A framework for relevant software development education

J.A. Liebenberg

10088873

Thesis submitted for the degree Philosophiae Doctor in Natural Science Education at the Potchefstroom Campus of the North-West University

Promoter: Prof HM Huisman

Co-promoter: Prof E Mentz

April 2015

Acknowledgements

A word of heartfelt thanks to the following persons, who, each in a particular capacity contributed in making this study possible:

- Prof. Magda Huisman for her enthusiasm, professional, though humane guidance and continued support.
- Prof. Elsa Mentz for her professional support, going the extra mile and contributing in her own special way.
- Dr. Estelle Taylor, prof. Tjaart Steyn and prof. Gilbert Groenewald for supporting me with study leave and other work matters.
- My husband, Christiaan, for his support and prayers whilst handling his own work and study pressures.
- My three children for unconditional love and understanding – you were and will always be my first priority.
- My dad, mom and sisters for their support – be it advise or just listening.
- Various friends and family members for support and encouragement.
- Isabel Swart for language editing and proofreading and ultimately – rapid response.
- Erika Fourie for statistical guidance and advice.
- Emily Mphoseloa for keeping the household afloat.

Above all our Lord and God for blessing me with this opportunity to study a mere speck in His wonderful creation.

Preface

This thesis is submitted in an article format in accordance with the General Academic Rules (A.5.1.1.2) of the North-West University. Four articles are included in this thesis.

I. Liebenberg, J., Huisman, M. & Mentz, E. 2015. The relevance of software development education for students. *IEEE Transactions on Education*. IEEE Early Access Articles. IEEE Xplore Digital Library. DOI=10.1109/TE.2014.2381599.

II. Liebenberg, J., Huisman, M. & Mentz, E. 2015. Industry's perception of the relevance of software development education. *TD The Journal for transdisciplinary research in Southern Africa*. (Accepted for publication).

III. Liebenberg, J., Huisman, M. & Mentz, E. 2015. Software: university courses vs workplace practice. *Industry and higher education.* 29(3):221-235.

IV. Liebenberg, J., Huisman, M. & Mentz, E. 2015. The relevance of software development education: students vs professionals. *Information Systems Management.* (Under review)

The co-authors of the articles in this thesis, Prof Magda Huisman (Promotor) and Prof Elsa Mentz (Co-Promoter), hereby give permission to the candidate, Mrs Janet Liebenberg, to include the articles as part of a Ph.D thesis. The contribution (advisory and supportive) of these co-authors was kept within reasonable limits, thereby enabling the candidate to submit this thesis for examination purposes. This thesis therefore serves as fulfilment of the requirements for the Ph.D degree in Natural Science Education within the School of Computer, Statistical and Mathematical Sciences in the Faculty of Natural Sciences at the North-West University, Potchefstroom campus.

Summary

It is widely acknowledged that there is a shortage of software developers with the right skills and knowledge. In respect of their university education, students want to take courses and carry out projects that clearly relate to their lives and their goals. The software development industry on the other hand, expects students to be educated in courses and projects, which are relevant for their professional career and equip them to be well-prepared for the workplace. In the middle, between the students and the industry, is the university that is expected to meet the needs of the students on the one side and the software industry on the other side.

The unique contribution of this research is the development of a framework for relevant software development education by addressing the question: How can universities ensure that software development education provides knowledge and skill sets that are relevant to both the software development industry and software development students? The literature study investigates the software development class, focusing on the students and the educators. Furthermore, a review of the software development workplace is done with attention to the software developers and their employers. The problems and challenges facing three role players in software development education, namely the students, the university and the industry are investigated. Lastly, the role of the university in relevant software development education is considered with a specific focus on curricula.

In the empirical study a questionnaire was developed to investigate the relevance of software development education from the perspective of the students. The questionnaire enquired about students' interests in each of a list of software development topics and further questions relating to students' views and needs for a relevant education are presented. The questionnaire was completed by 297 software development students and it was found that although a gap exists between students' needs and software development education, students' education does have a predominantly social relevance and also a moderate personal and professional relevance.

A second questionnaire was developed to investigate the relevance of software development education as it pertains to the software industry. The questionnaire enquired about 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 questionnaire was completed by 214 software development professionals and again it was found that there is a gap between the industry's needs and software development education. Questions related to the industry's needs, as well as an open-ended question at the end of the questionnaire offered rich

insights into the industry's view of its new graduates and the problems and challenges surrounding software development education. The quantitative data, as well as the qualitative data offered solutions to these problems and challenges.

The students' views are compared with the professional software developers' views to investigate the compatibility between the relevance of software development education for students and the relevance for the software industry. The analysis reveals matching and differing views.

A framework for relevant software development education was developed to address the gap between software development education and the students' needs, as well as the gap between software development education and the industry's needs. The problems and challenges that might cause SD education to be less relevant are presented and recommendations to industry and university for relevant software development education are made.

Key terms: software development education; software development students; software industry; software professionals; new recruits; computing curricula, university, relevance.

Table of Contents

Literature review

RELEVANT SOFTWARE DEVELOPMENT EDUCATION

Article 1

The relevance of software development education for students

Article 2

Industry's perception of the relevance of software development education

Article 3

Software: University Courses versus Workplace Practice

Article 4

The relevance of software development education: Students vs Professionals

A FRAMEWORK FOR RELEVANT SOFTWARE DEVELOPMENT EDUCATION

List of Tables

List of Figures

Chapter 1

ORIENTATION, RESEARCH DESIGN AND METHODOLOGY

1.1 PROBLEM STATEMENT

1.1.1 Research problem

The research problem investigated in this study is based on the realisation that universities need to ensure that software development education provides knowledge and skill sets that are relevant to both the software development industry and software development students. This study therefore aimed to develop a framework for relevant software development education.

1.1.2 Motivation

Bill Gates (2005) promotes the Gates Foundation's "3Rs" of education reform, which stand for "Rigour, Relevance and Relationships."

- Rigour – making sure all students are given a challenging curriculum that prepares them for college or work;
- Relevance – making sure students have courses and projects that clearly relate to their lives and their goals;
- Relationships – making sure children have a number of adults who know them, look out for them, and push them to achieve.

In software development (SD) education at universities a fair amount of attention is paid to Rigour. The Association for Computing Machinery (ACM), the Computer Society of the Institute for Electrical and Electronic Engineers (IEEE-CS), the Association for Information Systems (AIS), and the International Federation for Information Processing (IFIP) publish model curricula and guidelines for undergraduate degree programmes in Information Systems (Joint Task Force on Computing Curricula: Association for Computing Machinery (ACM) and IEEE Computer Society, 2013; Lunt *et al.*, 2008; Topi *et al*., 2010; Mulder & Van Weert, 2001).

Attention is paid to a lesser extent to the relevance of SD education. Relevance can be defined as: relation to the matter at hand / practical and especially social applicability / what is important in this time and situation (Oxford English Dictionary, 2011). Relevance in the educational context can be defined as the applicability of what is taught in respect of the needs and interests of students and society (Holbrook, 2009). Relevance in the software industry context, according to Wohlin and Regnill (1999), means "*that the education prepares students so that they are ready to cope with large-scale software development. It is also important that students are aware of the challenges and proven techniques related to industrial development of software*".

Students want to be educated at university in courses and projects that clearly relate to their lives and their goals (Holbrook, 2009). On the other hand, the software development industry expects students to be educated in courses and projects, which are relevant for their professional career and equip them to be well-prepared for the workplace (Moreno *et al.*, 2012).

There is a great shortage of information and communication technology professionals in South Africa. This is, however, also a worldwide phenomenon (Harris, 2012; McAllister, 2012; Bateman, 2013). The CareerJunction Index (CJI), which monitors online employment trends in South Africa, reported that the Information Technology industry is at the forefront of South Africa's career demand, as recruitment conditions for these skills are the most difficult of the 36 reported industries (CareerJunction Index, 2014). The South African government is even prepared to employ foreigners as software developers by including software developers in the Department of Home Affairs' published list of "scarce and critical skills", for which special exemption to normal immigration requirements has been affected (Department of Home Affairs, 2014). South Africa therefore needs more software developers.

Not only is there a shortage of SD workers, but students from computing disciplines are also graduating with a lack of the knowledge and skills that companies are searching for (Bateman, 2013). Regarding ICT skills in South Africa, "*not all graduates are prepared for the working environment, and don't always fit in. This is a gap that needs addressing*" (Mawson, 2010).

In the light of the shortage of software developers and in the light of the lack of the right knowledge and skills in the SD industry, software development education needs to be relevant; not only for the student but also for the industry. The research questions and subquestions this study therefore set out to answer were:

- What is the view of students regarding the relevance of SD education?
  - What is SD students' view of a career in software development?
  - Is there a gap between SD education and the student's needs from the perspective of the SD student?

    - What topics are relevant to SD students in their education?
- What is the view of the software industry regarding the relevance of SD education?
    - What is the industry's view of its new graduates?
    - Is there a gap between SD education and the workplace from the perspective of the industry?
    - What are the topics professional software developers learned in their formal education?
    - What topics are important to professional software developers in their actual workplace?
    - What are the problems and challenges surrounding software development education and what are the solutions to these problems and challenges?
- What is the compatibility between the views of SD students regarding the relevance of SD education and the views of SD professionals?

The author has developed a *Framework for relevant software development education* (see Chapter 7) to address the gap between software development education and the students' needs, as well as the gap between software development education and the industry's needs. The problems and challenges that might cause SD education to be less relevant are presented and recommendations to industry and university for relevant software development education are made.

The rest of this chapter is dedicated to providing a clear breakdown of the research design and methodology. As stated in the preface this thesis is submitted in an article format and since the articles do not allow for expanded descriptions of design and methodology, this chapter not only provides an orientation to the study but also describes the research design and methodology.

1.2 CLARIFICATION OF TERMINOLOGY

For the purpose of this study, it is necessary to explain and clarify certain key terms:

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

- *SD classes* will refer to the courses where the process of developing software is taught.
- *SD students* will refer to the students in the SD classes.
- *Software Industry* – the ISO and IEC define *software manufacturer* as a group of people who or organisations that develops/develop software, typically for distribution and use by other people or organisations (ISO/IEC, 19770-2, 2009). For the purpose of this study, *Software Industry* will refer to software manufacturers, as well as organisations where software is not the organisation's main product but software is developed for use within the organisation.

1.3 RESEARCH AIMS AND OBJECTIVES

The aim of the research is to investigate the relevance of software development education. This aim is operationalised as follows (to form pertinent objectives that have to be achieved):

1) to investigate the relevance of software development education as it pertains to students;
2) to investigate the relevance of software development education as it pertains to the software industry;
3) to investigate the compatibility between the relevance of software development education for students and the relevance for the software industry; and
4) to develop a framework for relevant software development education.

1.4 RESEARCH PARADIGM, DESIGN AND METHODOLOGY

Research is the creation of new knowledge, using an appropriate process, to the satisfaction of the users of the research (Oates, 2006).

1.4.1 Research paradigm and design

All research has certain philosophical underpinnings or takes place within a certain methodical research paradigm. A paradigm is a shared belief system that influences the kinds of knowledge researchers seek and how they interpret the evidence they collect (Johnson & Onwuegbuzie, 2004).

A pragmatic approach was followed in this study. Morgan (2007) states that the great strength of the pragmatic approach is its emphasis on the connection between philosophical concerns about the nature of knowledge and the technical concerns about the methods that we use to generate that knowledge. Pragmatism is a practical and applied research philosophy that allows the

researcher to make use of a combination of qualitative and quantitative methods (Teddlie & Tashakkori, 2003).

The empirical part of this study took place by implementing a mixed methods research design. Tashakkori and Creswell (2007) describe mixed methods as: "*Research in which the investigator collects and analyses data, integrates the findings, and draws inferences using both qualitative and quantitative approaches or methods in a single study or programme of inquiry*". Such work can help develop rich insights into various phenomena of interest that cannot be fully understood using only a quantitative or a qualitative method. Mixed methods research will often provide the most informative, complete, balanced, and useful research results (Venkatesh *et al*., 2013; Johnson *et al*., 2007).

Implementation of this mixed methods design implies that the qualitative and quantitative data are considered as equally important, in other words QUAN + QUAL, as presented by Hanson *et al*. (2005).

Creswell and Clark (2007) suggested four major types of mixed methods design:

- triangulation (i.e. merge qualitative and quantitative data to understand a research problem);
- embedded (i.e. use either qualitative or quantitative data to answer a research question within a largely quantitative or qualitative study);
- explanatory (i.e. use qualitative data to help explain or elaborate quantitative results); and
- exploratory (i.e. collect quantitative data to test and explain a relationship found in qualitative data).

In this study the type of mixed methods research was explanatory, as the objective of the qualitative investigation was to supplement the quantitative investigation and to better understand and explain the observations of the quantitative investigation.

1.4.2 Research methodology

In order to achieve the aims set out in 1.3, two methods were used, namely a literature study and an empirical investigation, ultimately resulting in the development of a framework for relevant software development education.

1.4.2.1 Empirical investigation

The empirical part of the study included a quantitative investigation, as well as a qualitative investigation.

In a quantitative investigation specific, focused questions are asked, numerical data of participants is collected, the numerical data is processed by using statistical procedures and the research is carried out in an impartial and objective manner (Cresswell, 2005). The quantitative investigation in this study was carried out using a survey research strategy where two questionnaires were used to collect data (Oates, 2006).

Qualitative researchers are interested in understanding the meaning people have constructed, that is, how they make sense of their world and experiences they have in the world (Merriam, 2009). According to McMillan (2000) the purpose of such research is to provide rich narrative descriptions of phenomena that enhance understanding and it is based on verbal narratives and observations rather than numbers. A basic qualitative design was used (Merriam, 2009) aiming at assisting the researcher to discover and gain understanding regarding the lived experiences of software developers.

1.4.2.1.1 Study participants

- **Software development students**

A convenience sample was used and the participants were all students in the SD classes at a university in South Africa. The participants included four academic year groups, most of whom were undergraduate students following the 3-year BSc in IT and CS programme in the Department of Computer Science and Information Systems. A subsequent fourth year BSc Honour's degree in CS and IS is offered and this degree gives access to a Master's degree in CS. The programmes are based on the Informatics Curriculum Framework of the International Federation for Information Processing (IFIP).

- **Software development professionals**

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.

1.4.2.1.2 Data collection and instrument

Both quantitative and qualitative data were collected in this study by making use of two questionnaires as a data collection method.

- **Software development students**

The data collection amongst the student participants took place by posting 386 questionnaires as an assignment on the e-learning system. The questionnaire was in an Excel workbook and the

students simply had to mark their answer with an X under the appropriate heading. They completed the questionnaire anonymously and once the questionnaires had been submitted on the e-learning system, the files were downloaded in bulk. The number of usable responses received totalled 297, with 276 being from BSc undergraduate students and 21 from students enrolled for the subsequent BSc(Hons), making for an overall response rate of 76.9%.

The process of developing the students' questionnaire progressed as follows: an initial list of questions was developed by writing both new items and adapting items from available surveys, such as ROSE (Schreiner & Sjøberg, 2004), the South African Qualifications Authority's (SAQA, 2000) list of "Critical Cross-field Outcomes" and topics commonly found in SD programmes. A further step of refining and enhancing the questions resulted in the questionnaire with a pool of 123 items (see Appendix A). The questionnaire consisted of a set of 57 attitude items followed by a set of items listing 66 core SD topics.

The first section of the questionnaire gathered information on the biographic data of the respondents. The remainder of the questionnaire was divided into four domains.

The first domain "Out of class" investigated personal relevance, with 12 items that gathered data on the students' out-of-class experiences, such as using the Internet and developing a software system. The participants were asked: "How often have you done this outside formal education?" with a five-point Likert response scale: Never / Once or twice / I don't know / Quite often / Very often.

The second domain "In class" investigated personal, social and professional relevance with 33 items, enquiring about their perceptions of their SD classes, such as their enjoyment and interest in the classes.

The third domain "My career" had 12 items that investigated social relevance, gathering data on their notions of a future career, such as what they thought would be expected from a good software developer. The second and third domain used a five-point Likert response scale from 1 (Strongly disagree) to 5 (Strongly agree).

The last domain "What I want" had 66 items to investigate professional relevance, testing their interest in curriculum topics such as specific programming languages or extreme programming. The participants were asked: "How interested are you in learning about the following?" on a five-point Likert response scale: Not interested / Slightly interested / Neutral / Quite interested / Very interested.

- **Software development professionals**

The data collection amongst the software development professionals took place by contacting them personally via e-mail and requesting them to complete the anonymous online survey. Some of the respondents indicated that they 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 totalled 214, which indicates a response rate of around 21%.

The questionnaire that was used for the SD students was used as a point of departure for the questionnaire for the SD professionals. A process of refining and enhancing the questions resulted in a list of 151 questions (see Appendix C). The questionnaire consisted of a set of 25 attitude items followed by two sets of items both listing the same 63 core SD topics. 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*) and the second set asked: "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*). An open-ended question at the end of the questionnaire asking for further comments on the education of software developers was included and the qualitative data was obtained through that question.

The qualitative data gathered in the open-ended question 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.

- **Software development students and software development professionals**

When the data of the students were compared with the data of the professionals, 89 corresponding items from each of the SD students' and SD professionals' questionnaires were included in the list (see Appendix A of Article 4 in Chapter 6). The list included 26 attitude items and the 63 core SD topics.

The reliability and validity of the questionnaires in this study was ensured as follows:

Reliability

Reliability is concerned with whether the measures show stability across the units of observation (Straub, 1989). The reliability of the questionnaires was determined by calculating the Cronbach alpha values. The Cronbach alpha values of the questionnaires for SD students (n=297), SD

professionals (n=214) and the whole group of respondents (n=511) were calculated and were found as Table 1-1, Table 1-2 and Table 1-3 show, to be reliable (α ≥ 0.60).

Table 1-1: Reliability coefficients of factors for SD students

Factor	Cronbach's alpha (α)
Out_of_class_Basic_use	0.743
Out_of_class_Advanced_use	0.605
In_class_Learn	0.876
In_class_Perceptions	0.745
In_class_Attitudes	0.853
In_class_Importance	0.724
In_class_Teaching	0.678
Career_Attitudes	0.843
Career_Skills	0.772
Real-time and systems programming	0.935
Mathematics and statistics	0.902
Software management	0.918
General software design	0.907
Web and Mobile technologies and Games	0.934
Computer science theory	0.934
Specialized application techniques	0.926
Software engineering methods	0.916
Essential subsystem design	0.897
Computer hardware and other electrical and computer engineering	0.946
Alternative software engineering methods	0.880

Table 1-2: Reliability coefficients of factors for SD professionals

Factor	Cronbach's alpha (α)
Critical outcomes required in software development modules	0.718
Positive attitude towards colleagues/management	0.793
Positive attitude towards tasks/work	0.666
Emotional/social skills required	0.700
Set1:	

Factor	Cronbach's alpha (α)
Information systems	0.800
Computer hardware and electrical and computer engineering	0.937
Software testing and maintenance	0.900
Computer science theory	0.912
Real-time and systems programming	0.888
Mathematics and statistics	0.933
Mobile technologies	0.946
Software development methodologies	0.915
Software management	0.882
General software design	0.853
Specialised application techniques	0.865
Web design and development	0.914
Hardware: Data transmission	0.851
Software engineering methods	0.901
Set2:	
Information systems	0.812
Computer hardware and electrical and computer engineering	0.907
Software testing and maintenance	0.833
Computer science theory	0.908
Real-time and systems programming	0.867
Mathematics and statistics	0.910
Mobile technologies	0.967
Software development methodologies	0.803
Software management	0.831
General software design	0.834
Specialised application techniques	0.884
Web design and development	0.911
Hardware: Data transmission	0.820
Software engineering methods	0.867

Table 1-3: Reliability coefficients of factors of whole group

Factors	Cronbach's alpha (α)
Information systems	0.842

Factors	Cronbach's alpha (α)
Computer hardware and electrical/computer engineering	0.948
Software testing and maintenance	0.857
Computer science theory	0.923
Real-time and systems programming	0.912
Mathematics and statistics	0.927
Mobile technologies	0.950
Software development methodologies	0.829
Software management	0.864
General software design	0.882
Specialised application techniques	0.925
Web design and development	0.901
Hardware: Data transmission	0.820
Software engineering methods	0.873
Critical outcomes required in courses	0.849
Knowledge of course requirements	0.606
Positive attitude towards work and colleagues	0.800
Emotional/social skills required	0.760

Content validity

Content validity is concerned with whether the instrument measures are drawn from all possible measures of the domain to be covered (Straub, 1989). In order to increase the content validity of the questionnaires used in this study, an initial list of questions was generated and sent to three industrial and two academic experts familiar with tertiary computing programs. The experts' feedback served as the basis for correcting, refining, and enhancing the questions.

In the students' questionnaire four items were added in the SD topics section. In the attitude items section one question was split in two, two items were rephrased and three items were omitted.

In the professionals' questionnaire three items were omitted in the SD topics section. In the attitude items section seven items were rephrased, two items were added and one item was omitted.

Construct validity

Construct validity is concerned with whether an instrument is measuring what it is supposed to be measuring or whether the measures show stability across methodologies (Oates, 2006,

Straub, 1989). The construct validity of the questionnaires was in all three supported by factor analysis.

1.4.2.1.3 Data analysis

- **Quantitative data**

The statistical data of the two questionnaires were analysed with the help of the Statistical Consultation Services of the North-West University (Potchefstroom campus).

The following statistical methods and techniques were used to analyse the data:

- Factor analysis was used to investigate the items of the two questionnaires in more detail to reduce the variables into a smaller number of factors.
- Basic analysis was done by calculating the mean values and standard deviation of each of the factors and items.
- T-tests were used to test for significant differences between means of two groups, such as the males and females.
- ANOVA tests were used to test for significant differences between means of more than two groups, such as the different age groups of the software development professionals.
- Chi-square tests and Spearman's rank correlation analysis were used to analyse relationships between the groups.

Software development students

Factor analysis was applied to the 123 items in the students' questionnaire to reduce the variables to a smaller number of factors. The 297 responses were examined using principal components factor analysis; the 123 attitude items yielded 20 interpretable factors, named according to their main context, and three single item variables (see Appendix B for the items in each factor).

Software development professionals

The variables in the professionals' questionnaire were reduced into a smaller number of factors by the use of factor analysis of the 151 items. The 214 responses were examined using principal components factor analysis as the extraction technique and Oblimin with Kaiser Normalization as the rotation method. Of the 25 attitude items, 10 items were being handled as single research variables and the remaining 15 items yielded four interpretable factors. The factor analysis of the two sets of 63 items was done while taking into account that the two sets needed to be comparable. The two sets 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.

Software development students and software development professionals

Factor analysis was used to investigate the 89 items in more detail to reduce the variables into a smaller number of factors. The 511 responses were examined using principal components factor analysis and the 89 attitude items yielded 18 interpretable factors and 11 items that were being handled as single research variables. X_1

A convenience sample instead of a random sample was used for both the questionnaires, and the p-values are therefore reported for the sake of completeness but are not interpreted. Effect sizes (d-values) indicating practical significance were calculated and reported when data was interpreted (Steyn *et al*., 1999). The effect sizes of the differences were calculated with Cohen's d-value by using the formula:

$$d = \left| \frac{X_1 - X_2}{s} \right|$$

where X_1 = mean of construct of one group;

X_2 = mean of construct of other group; and

s = pooled standard deviation.

Cohen's d indicates the differences as follows:

≈ 0.2 : small effect size

≈ 0.5 : medium effect size

≈ 0.8 : large effect size (practical significant difference)

- **Qualitative data**

The qualitative data obtained through this study was meant to supplement the quantitative data collected in this study, in order to better understand and explain the software professionals' view regarding the relevance of SD education. For the analysis of the qualitative data obtained through the open-ended question in the questionnaire and the e-mails from software developers computer-aided data analysis was done. Computer-aided qualitative analysis is the analysis of data with the aid of computer software developed to aid the researcher in storing, coding and organising of data (Oates, 2006). In this study the ATLAS.ti 7.1.4 computer program was used. Tools in Atlas.ti, such as the auto-coding tool, object manager, families and network views help the researcher to navigate through the data structures and concepts (Friese, 2013). By utilising these tools the data was then analysed as follows:

- The data of the open-ended question in the online survey and the data from the e-mails were assigned to a single hermeneutic unit;
- The relevant information was separated from the irrelevant information;
- The relevant information was broken into a number of text segments;

- The emerging themes were coded in a process of inductive categorisation and the text segments were linked to the coded themes;
- The codes were grouped into code families;
- Networks were drawn that reflect the meaning of the respondents' views (see Appendix D); and
- The networks were used to develop an overall description of SD professionals' views of SD education.

Since the product of qualitative research is richly descriptive (Merriam, 2009), the participant comments were analysed, interpreted and presented in the form of quotes.

1.4.2.2 Development of the framework

The research problems identified have culminated in the development of the proposed *Framework for relevant software development education* by addressing the question: How can universities ensure that software development education provides knowledge and skill sets that are relevant to both the software development industry and software development students?

After analysis of all the results obtained in the empirical investigation the researcher identified three role players and three layers to be represented in the framework.

The role players: on the one side are the software development students, on the other side the software development industry and in the middle, between the students and the industry, is the university.

The layers: at the top are the problems and challenges faced by the three role players, in the middle the needs of the students and the software industry and the bottom consists of recommendations to industry and university for relevant software development education.

A gap between software development education and the students' needs, as well as a gap between software development education and the industry's needs were established and the university is therefore at the core of the framework owing to the fact that the university is expected to meet the needs of the students on the one side and the software industry on the other side.

1.4.3 Ethical aspects of the research

Participation in the study by students through completion of the questionnaire was optional and participation was anonymous.

The software development professionals' participation in the study was completely voluntary and the online questionnaire was anonymous. The comments and suggestions of the respondents who emailed the researcher were kept confidential.

1.5 THESIS STRUCTURE

The structure of the thesis will be presented in article format as approved by the North-West University (Potchefstroom Campus) and will be as follows (see also Figure 1-1):

Chapter 1: Orientation, research design and methodology
This chapter contains the problem statement, the research aims, the research paradigm, design and methodology, as well as ethical aspects of the study. A preview of the chapters of the thesis is given and attention is paid to the contribution of the study.

Chapter 2: Literature review: Relevant software development education
This chapter presents the literature review on software development education and the stakeholders in SD education, namely the students, universities and the software industry.

Chapter 3 – Article 1: The relevance of software development education for students
This chapter is presented in the form of a manuscript published in the journal *IEEE Transactions on Education (TOE)*. The guidelines of this journal are presented in Appendix E.

Chapter 4 – Article 2: Industry's perception of the relevance of software development education
This chapter is presented in the form of a manuscript, which is accepted for publication in *TD The Journal for Transdisciplinary Research in Southern Africa*. The guidelines of this journal are presented in Appendix F.

Chapter 5 – Article 3: Software: university courses vs workplace practice
This chapter is presented in the form of a manuscript published in the journal *Industry and Higher Education*. The guidelines of this journal are presented in Appendix G.

Chapter 6 – Article 4: The relevance of software development education: students vs professionals
Chapter 6 is presented in the form of a manuscript submitted and reviewed at the journal *Information Systems Management*. The guidelines of this journal are presented in Appendix H.

Chapter 7: – A framework for relevant software development education
In Chapter 7 a recommended framework for relevant SD education is presented. Limitations to the study are presented with recommendations for future research.

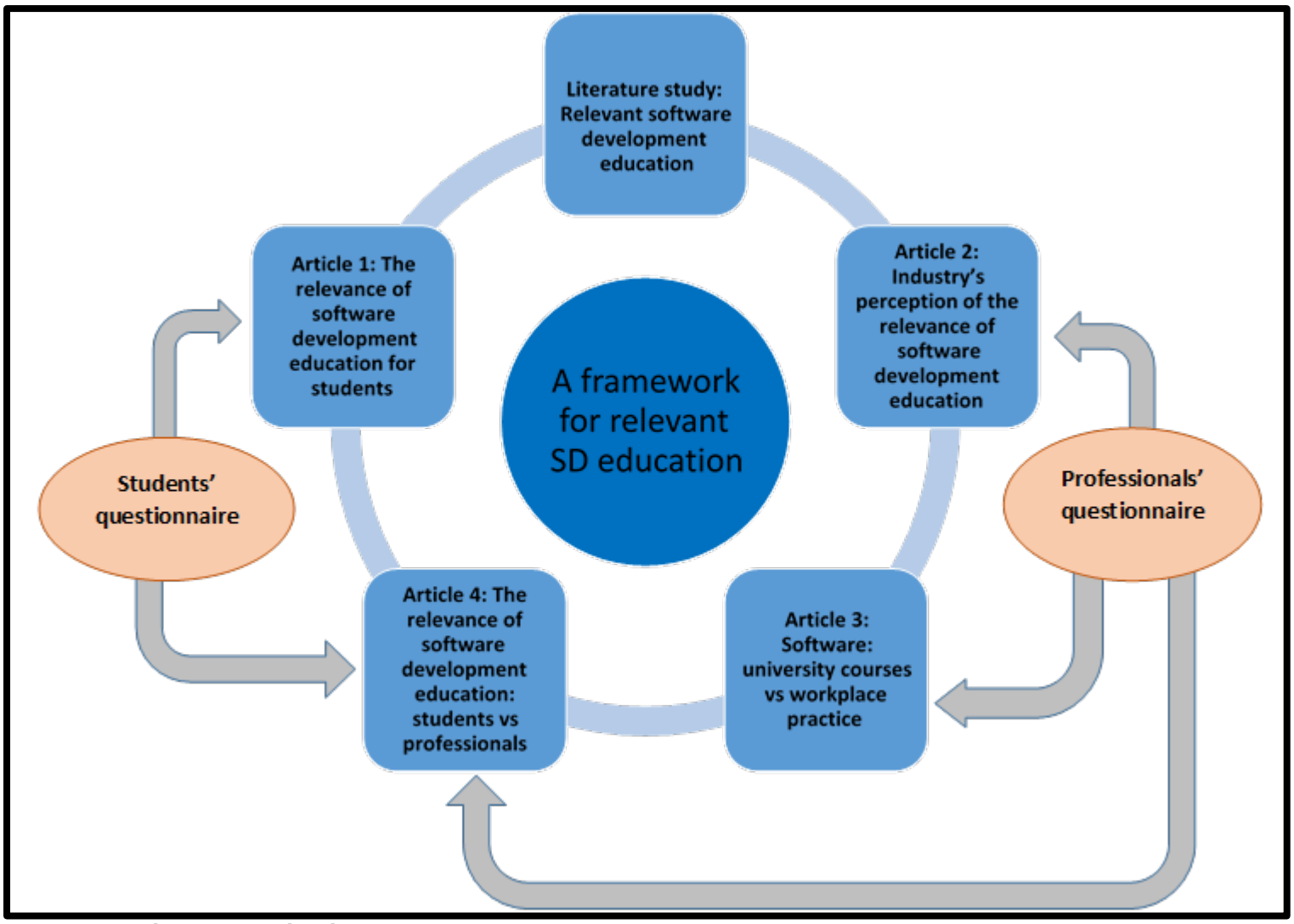


Figure 1-1: Thesis structure

The references of Chapters 1, 2, and 7 will be presented according to the Harvard style as prescribed by the North-West University (Potchefstroom Campus). The references of Chapters 3, 4, 5 and 6 are presented according to the requirements of the specific journal to which the articles were submitted for publication.

1.6 CONTRIBUTION OF THE STUDY

- The unique contribution of this research is the development of a framework for relevant software development education that can be used as a guide by all the stakeholders in SD education, ranging from students and universities (lecturers, developers of degree programmes/curricula) to industry (corporate trainers, management, employers).
- This study is the first to investigate students' views on the relevance of SD education and the university and software industry can therefore utilise this rare insight to improve the relevance of SD education.

- Students, new graduates and the university are informed by a picture of the view the industry has of the new graduates.
- The current knowledge and skills important to students and the software industry are presented, which can assist in curriculum development but also in in-house training and further education and training.
- The problems, challenges and solutions in respect of software development education are presented, to serve as a measure and guideline for all the role players in SD education.
- This study can lead to further research on factors that might influence the relevance of education in other related fields, such as engineering.

Chapter 2

Literature review

RELEVANT SOFTWARE DEVELOPMENT EDUCATION

2.1 INTRODUCTION

In a time of rapid technological changes the problem of how to educate software developers to do their jobs efficiently and properly remains a crucial open question for the future of the profession, keeping in mind the relevance of SD practices in the IT world. Lethbridge *et al.* (2007) argue that the majority of quality and budgetary issues with software have their root cause in human error or lack of skill. These issues in turn arise in large part from inadequate education. Therefore improving software development education should contribute towards improving software and software practice. In particular, software developer positions are important to education because they are common entry-level positions - graduates do not typically start their careers as project managers or consultants (Surakka, 2007).

Software developer Bill Gates (2005) focuses on his foundation's "3Rs" of "Rigor, Relevance and Relationships". The central pillar of relevance highlights that students need to be exposed to courses and projects that clearly relate to their lives and their goals. At the same time, the software 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).

2.2 THE CONCEPT OF RELEVANCE

Relevance is broadly defined as closely connected or appropriate to the matter in hand; or having significant and demonstrable bearing on the matter at hand; or practical and especially social applicability (Oxford English Dictionary, 2013; Merriam-Webster Dictionary, 2013).

Labaree (2008), who works in educational research, emphasises relevance as a function not only of person and purpose, but also of place and time. He argues that the question “useful to whom and for what?” needs to be answered because a wide array of actors is involved, including students, teachers, parents, curriculum developers and employers.

Relevance in the educational context can be defined as the applicability of what is taught in respect of the needs and interests of students and society (Holbrook, 2009). The process of instruction and learning is designed to make what is learnt relevant/current to the time so that it can be implemented in the social environment. As a result, the student then sees the learning as meaningful, timely, important and useful, and it builds on the intrinsic motivation of the student for self-concern, self-involvement, self-appreciation and self-development (Holbrook, 2009).

Holbrook (2003) suggests three relevance perspectives for students:

- Social relevance – the "useful in society" perspective, which is a perceived need for the society;
- Personal relevance – the "interest" perspective, which directly relates to concerns in the students' immediate environment or area of interest;
- Professional relevance – the "important for the course they are studying" perspective, which relates to the content of the curriculum that has to be interesting and useful to students.

Students need to see the relevance of teaching and learning, as it applies to them personally (their own lives, their career expectations, the wishes of their parents), or the relevance as it applies to society (wishes of the community, employers, the university) or as it applies to them professionally (the content/curriculum is meaningful) (Holbrook, 2003).

Relevance to the software industry as the employers of the students brings another perspective to the relevance of software development education. Industrial relevance implies that the education that students receive prepares them for large-scale software development, including the proven techniques and challenges related to industrial development of software (Wohlin & Regnill, 1999).

2.3 THE SOFTWARE DEVELOPMENT CLASS

In the next section, attention will be given to the two major role players in the SD class, namely the students and the educators.

2.3.1 The students

There is a new population emerging from young people born after the time when digital technologies began to be embedded in social life; sometime in the 1980s (Howe & Strauss, 2000). This group of young people is described as the Net generation, also known as the Millennial Generation, Generation Y or Digital Natives. These young people (especially people born in the US and Canada from the early 1980s to the late 1990s) have grown up with computers and the

Internet and therefore they have a natural aptitude and high skill levels when using new technologies (Oblinger & Oblinger, 2005; Cheese, 2008). Their early and omnipresent exposure to technology has defined their styles, their modes of communication, their learning preferences, their social choices, and their entertainment preferences (Saiedian, 2009).

Howe and Strauss (2000) described seven distinguishing traits of Millennials:

- Special - not only for themselves, but also for the future of society and the world.
- Sheltered - having been smothered with safety rules and devices.
- Confident - as a result of their trust and optimism. Often boast of power and potential.
- Team-oriented – there is an emphasis on group learning and tight peer bonds exist.
- Achieving - the result of higher school standards and an instilled sense of accountability.
- Pressured – they are pushed to study hard and take advantage of opportunities.
- Conventional – they take pride in good behaviour and are not rebellious. They are comfortable with their parents' values.

Some more characteristics of the Net generation have been described by Jones *et al*. (2010) and Oblinger and Oblinger (2005). The Net generation are individuals who believe it's cool to be smart; are fascinated by new technologies; are racially and ethnically diverse; value social networking; are not politically active, but community centered; expect quick rewards; are impatient with linear thinking; and display a novel capacity for multi-tasking.

Some challenges related to the Net generation have surfaced as well:

- shallowness of reading and TV-viewing habits;
- a comparative lack of critical thinking skills;
- naïve views on intellectual property and the authenticity of information found on the Internet; and
- high expectations combined with low satisfaction levels (Hartman *et al*., 2005).

Researchers describe the learning preferences of Digital Natives (Frand, 2000; Prensky, 2001; Prensky & Berry, 2001; Oblinger, 2003; Oblinger, 2008). Digital Natives prefer receiving information quickly; are masters at processing information rapidly; prefer multi-tasking; prefer non-linear access to information; have a low tolerance for lectures; prefer active rather than passive learning; rely heavily on communications technologies to access information and to carry out social and professional interactions; expect to be engaged by their environment with participatory, sensory-rich, experiential activities (either physical or virtual); are more oriented to visual media opportunities for input than previous generations; prefer to learn by doing rather than by telling or reading; and prefer to discover rather than be told.

Numerous people analyse the main traits of different generations, but Hoover (2009) warns that it can be a strong form of stereotyping and that not all university students fit into one mould. Bayne and Ross (2011) highlight the way in which the categorisation of the 'digital native' works to homogenise diverse and varied groups of individuals, using generational categorisation to over-determine student characteristics and relations to technology.

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. Furthermore, older students comprise a large and growing percentage of higher education. They may have information literacy characteristics and IT skills quite different from the typical Net generation. Higher education comprises a highly diverse and growing student body possessing a core set of technology-based skills, but beyond this core there is a diverse range of skills across the student population (Lorenzo *et al*., 2006; Kennedy *et al*., 2008; Jones *et al*., 2010).

The students in developing countries do not fit the description of the Net generation since Internet penetration for households in 2013 was 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; UN Broadband Commission, 2014). A computer skills assessment project at a South African university on over 4 000 first-year students in 2009 found that many students entering South African universities for the first time are not adequately equipped with the computer skills that they will need during their first year of study and African students are most at risk of being disadvantaged by their lack of prior skills (Nash, 2009).

Helsper and Eynon (2010) showed that breadth of use, experience, self-efficacy and education are just as, if not more, important than age in explaining how people become digital natives. Krause (2007) and Kennedy et al. (2008) reported on a study of first-year students in Australian universities, finding that their experience and understanding of technology vary significantly according to socio-economic background, age and gender and conclude that the assumption that all students entering university are digital natives is misleading and dangerous.

In 2001 Prensky published two papers on a new generation of students: the 'Digital Natives'. The basis of Prensky's argument was that this new group of university students was fundamentally different from any students that educators had seen before (Prensky, 2001; Prensky & Berry, 2001). This generational shift has consequences for approaches to learning and it suggests that teachers and educational institutions have a responsibility to change in response to the assumed demands of this new generation of learners (Jones *et al*., 2010).

The argument that educators and universities have to make radical changes because of students' *radically different approach to learning* is not new and it continues to have a contemporary significance despite being questioned and criticised by Bayne and Ross (2007), Kennedy *et al*. (2008) and Bennett *et al*. (2008), for example *and* even Prensky (2009) started to distance himself from it. Bayne and Ross (2011) call for a more carefully critical and nuanced understanding of the effects of new technologies on the practices and subject positions of learners and teachers in higher education. Bennett *et al*. (2008) state that academics are not empirically and theoretically informed and the argument can be compared to an academic form of a 'moral panic'. Bennett *et al.* (2008) propose that a more measured and disinterested approach is required to investigate 'digital natives' and their implications for education.

2.3.2 The educators

In higher education a popular notion exists to describe lecturers as 'digital immigrants' and that it is the duty of lecturers to adapt their methods to students' new way of learning (Howe & Strauss, 2000; Prensky, 2001; Prensky & Berry, 2001). The older lecturers are characterised as being at least one step behind and unable to reach the kinds of natural fluency that comes with having grown up with new digital technologies (Jones *et al*., 2010). The realities of the software industry for which the students need to prepare have shifted away from those of the foundational beliefs and practices of many of their educators and educators are expected to become familiar with the students' teaching and learning preferences *in order to remain relevant* (Saiedian, 2009).

Bayne and Ross (2011:159) state in their strong-worded critique: "*Teachers, we are told, have a duty to adapt their methods to this new way of learning – are required, in fact, to re-constitute themselves according to the terms of the 'native' in order to remain relevant and, presumably, employable (for example Prensky 2001, Oblinger 2003, Long 2005, Barnes et al., 2007, Thompson 2007).*" Bayne and Ross (2011:161) continue by stating that this argument de-privileges the role of the teacher: "*We would argue that the term 'occupying the commanding position' in this opposition is that of the 'native' (the 'future'), with the 'immigrant' (the 'past') taking the subordinate position. What we then see here is a structurally embedded de-privileging of the role of the teacher, aligned with the 'immigrant' position – the old, the past, the slow, the backward-looking, the association with modes of knowledge construction becoming 'obsolete', and dependent on analogue (print) technologies".*

Hartman *et al.* (2005) found in their study of a student population spanning three generations, namely Baby Boomers, Generation Xers and the Net generation, that what constitutes good teaching appears to be universal across the generations. Students believe that excellent teachers

- facilitate student learning;
- communicate ideas and information effectively;

- demonstrate genuine interest in student learning;
- organise their courses effectively;
- show respect and concern for their students; and
- assess student progress fairly and effectively.

This idea is further expounded on in the study by Hartman *et al. (2005*) in which they found that young students' expectations for involvement with faculty and other students override a desire to use technology. Oblinger and Oblinger (2005) found that differences among individuals are greater than dissimilarities between generations, so students in any age cohort will present a mixture of learning preferences.

2.4 THE SOFTWARE DEVELOPMENT WORKPLACE

Software and technical developments have made remarkable strides in the last few decades, and these developments continue unabated. It has profoundly transformed markets, industries, and society in general (Biztech Africa, 2013; Shaw *et al*., 2005). Not only is the dependence on software increasing, but the character of software production itself is changing – and with it the demands on software developers (Saiedian, 2009; Shaw *et al*., 2005; Gupta, 2005).

Holley (2008), the executive vice president and CIO of Tellabs, states regarding the impact of Generation Y: "*Their comfort with technology is second nature, in fact some would argue it's a birth right to use technology as they see fit, and expect those of us in technology to invent, design and deliver to meet their needs today and in the future*".

In the next sections the two role players in the software development workplace, namely the employees, i.e. the software developers and the employers will come under the spotlight.

2.4.1 Software developers

The new graduates entering the workplace mostly belong to the millennial generation or Generation Y. In the workplace, Millennials want flexible work schedules, and they do not like traditional office rules and hierarchies. They want continuous performance feedback and career advice from managers and they think that managers could learn from their young employees. What could be misinterpreted as the "self-importance" attitude of Millennials is actually an optimistic sense of having many new ideas and wanting to contribute, as well as a desire to have their technical skills and intellect tapped by managers.

Millennials want to wear jeans to work and especially in IT companies, the norm is to wear casual clothes, for example jeans, sneakers, flip-flops and sweatshirts – (Facebook CEO Mark Zuckerberg's famous hoodie and former Apple CEO Steve Jobs' black turtleneck with jeans are

good examples) (Schawbel, 2012). The older generations, on the other hand, are more prone to believing in the importance of maintaining a standard professional look in the workplace. It seems as if Millennials also prefer casual attire because they do not separate their personal and professional lives in the same way that the older generations do.

Much is written about the so-called “generational gap”. Young professionals or Millennials are joining Generation X (born 1960-1980) and Boomers (born 1940-1960) in modern organisations. The three generations are inherently different - they approach work, work/life balance, accountability, delegation, loyalty, authority, motivation and reward systems differently (Macon & Artley, 2009). However, Augustine (2001) states that a popular misconception exists that the workforce consists of generational tribes engaged in rivalries and conflicts with each other. Augustine (2001) admits that there might be differences, misunderstandings and tensions among workers from different generations, but the division and conflict are often exaggerated.

Murray (2015) pointed out a misconception that Millennials want to change jobs frequently. Millennials actually value job security more highly than previous generations, but they will not stay in a job they do not like. There is also a misconception that money does not matter to Millennials. A high-paying job is near the bottom of their list of work priorities - but the same applies to other generations, in nearly equal numbers (Murray, 2015).

In their study Soni *et al*. (2011) determined the differences in work commitment of software professionals from the X-generation and Y-generation (Millennials) and found that the two generations differed significantly on only three of the nine factors examined. Continuance commitment to the profession is significantly higher for Generation X than for Generation Y, meaning that Generation X feels obliged to stay in the software profession. The Generation Y group of employees has higher job involvement and they feel that they ought to remain with their organisation significantly more than the Generation X group.

2.4.2 Employers

Firstly, the needs of employers in the software development industry and secondly, the measures employers need to take to get what they need are considered in this section.

According to Spicer (2011), any business needs from its incoming recruits “Critical Cross- field Outcomes” - the skills and abilities that the South African Qualifications Authority (SAQA, 2000) requires to be achieved in all their registered qualifications. The SAQA (2000) lists seven outcomes, namely problem solving and creative thinking; being able to work in a team; the ability to manage oneself and one’s activities, being able to critically evaluate information; good written and verbal communication; an effective use of science and technology; and being able to

demonstrate an understanding of the world as a set of related systems by recognising that problem-solving contexts do not exist in isolation. These outcomes describe what can also be called soft skills or non-technical skills.

Professionals in the computing field are required to possess personal skills, technical expertise and life-long learning abilities (Fernandez-Sanz, 2009; Calitz, 2010). Employers are looking for professionals who can add value to their organisations by working successfully in interdisciplinary teams rather than focussing exclusively on a narrowly defined technical job (McKinnon & McCrae, 2012). Apart from the technical issues, software development includes many aspects of modern business, such as project management, design, quality control, and legal issues (Feldt *et al*., 2010).

There is little disagreement that discovering what motivates individuals and rewarding them on what they find important, are the key to successfully recruiting and retaining talent (Bunton & Brewer, 2012).

Total rewards is a promising approach to motivation management that has been adopted by technology-intensive firms, such as IBM and Microsoft (Rumpel & Medcof, 2006). O'Neal (1998) indicates that a total rewards strategy provides a solution to the complicated problem of recruiting and retaining employees. A total rewards framework has four categories and is laid out in a two by two quadrant. The upper two quadrants represent the transactional rewards of *pay* and *benefits*. The lower two quadrants - *learning and development* and *work environment* - represent the relational rewards.

- Pay includes direct financial items, such as base salary, variable pay incentives, stock and equity sharing, and monetary recognition.
- Benefits include indirect financial rewards, such as health care, retirement plans, savings plans, and paid time off.
- Learning and development include programmes and practices related to career development, supporting performance management and succession planning systems.
- Work environment includes programmes and practices related to organisational climate, performance support, work / life balance, such as flexible working arrangements, elements related to organisational reputation, elements related to challenging work and relationships with colleagues.

O'Neal (1998) found that Information Technology workers are far more focused on technology and their work environment than on pay and benefits. Rumpel and Medcof (2006) conducted research on high-tech companies that used the total rewards framework and found that research and development workers value all four reward quadrants, not just the traditional monetary

compensation and benefits packages, which have been the hallmark of traditional compensation practice offered by many firms. Additionally they found that rewards in the work environment quadrant are the most highly valued by these workers. Kochanski and Ledford (2001) explored the top 15 predictors of retention for technology and scientific professionals and found that pay was not the number one motivating factor, but that work environment and learning and development were. In their study with 658 industrial and 1,033 academic research and development teams in 11 countries Keller *et al*. (1996) found that work importance had the strongest effect upon productivity, followed by participation/cooperation in the team. Salary, opportunities for advancement and supervision had no consistent effect upon productivity.
"When it comes to motivating their employees, it can be said without question that Google stands out from the rest. Google was named the 2014 "Best Company to Work For" by the Great Place to Work Institute and Fortune Magazine. The organization topped the list for the fifth time." (Smith, 2014). The Google founders did their research and they were led to understand that when people feel truly valued and thoroughly supported they are really successful and loyal in their work. The result was the Google work culture with benefits, such as huge and plentiful perks, amazing freedom, flexibility and transparency and unconventional office designs.

When millennial employees are considered, the following can be said: Millennials value self-development and are goal-oriented and therefore Millennials constantly seek opportunities to learn and grow professionally. They can be an asset for an organisation since they have characteristics, which make them easier to manage than Generation X (Eisner, 2005). Employers must provide opportunities for lifelong learning, specifically utilising on-line courses and training materials (Krakovsky, 2010). However, as a result of the positive reinforcement and self-esteem building Millennials received from their parents, it is asserted that they may need help with accepting constructive criticism and managing conflict (Dolezalek, 2007).

In a pilot study Bunton and Brewer (2012) identified the workplace rewards and motivators that the millennial generation of Information Technology (IT) employees find attractive in organisations. They found that "respect for me as a person" was rated as most important and the motivator of "good pay" did not make the list of top motivators. However, when a total rewards methodology was applied, the motivators representing the total rewards category of pay were highest ranked. Additionally, the participants showed a statistical preference for motivators that represented the upper two quadrants of transactional rewards over the lower two quadrants of relational awards.

"Providing a challenging and engaging environment is critical for attracting and retaining (software) developers." (Bosch, 2015).

2.5 CLASS VS WORKPLACE

Merkofer and Murphy (2010) gave an overview of the key findings of an Accenture research study on the e-skills shortage in South Africa in which role players from the ICT industry and tertiary education sector were interviewed and surveyed. Although the study covered the whole spectrum of e-skills (Level 1 - (e-literacy); Level 2 - (e-skills); and Level 3 (ICT specialist), the results indicated that e-skills taught at tertiary level do not sufficiently prepare students for the expectations of their roles in employment.

Several studies suggest a gap between the knowledge and skills demanded by the industry and the knowledge and skills gained by graduates of university computing courses. The seminal study of Lethbridge (2000) on the specialised area of software development analysed the relevance and depth of the knowledge that software professionals had gained as part of their university education, and identified a significant mismatch between software education and industry in terms of the knowledge needed by software engineers to meet industry's requirements. Table 2-1 provides a summary of relevant studies that specifically investigated the knowledge and skills demanded by the industry versus the knowledge and skills gained by graduates.

Table 2-1: Studies on the knowledge and skills gap from the industry's perspective

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 systems design, software cost estimation, software metrics, software reliability and fault tolerance, and requirements gathering and analysis. Mathematical topics over-emphasised.
*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-emphasised.
Kim *et al*. (2006)	Examined IS/IT skills gaps from three perspectives: end users, academia, and IS/IT employers.	Gaps in project management, security, ERP, and the integration of soft skills.
*Surakka (2007)	Small survey of software developers, lecturers, and master's students about the relevance of different matters.	SD-specific issues, such as design, testing or configuration management considered as equally relevant by all three surveyed groups. Increased importance of web-related subjects

Author(s)	Study	Results
		and skills. Mathematical topics over-emphasised.
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 advertisements posted on Fortune 500 corporate websites.	Require skills related to development, software, social skills, and business.
Aasheim *et al*. (2009)	Surveyed and compared the perceptions academics have of the importance of various skills for entry-level IT workers with the view that IT managers have.	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, organisational skills and work experience in the same order of importance.
Calitz (2010)	Interviewed managers in businesses employing ICT graduates to identify graduate skill requirements.	Software development skills and business skills were emphasised, but a number of soft skills and technical skills were also identified. New skills, including business analysis and analytics, business process modelling, mobile application development and internationalisation skills were identified.
Gallagher *et al*. (2010)	Investigated the skills considered critical when hiring entry-level employees. A total of 104 senior IT managers were interviewed by 20 researchers in 94 non-IT companies.	A mix of skills is essential but the skills most critical are non-technical skills, such as project management, business-domain knowledge and relationship skills. Technical skills required are foundational skills, such as programming and essential skills including systems analysis, systems design, and IT architecture/standards.
Clark *et al*. (2011)	Case study on three cases that focused on the transition from university to work.	Noted a growth in roles focused upon management, particularly project management, planning and strategy and a decline in technical roles. They argue that to understand transitions, the field, the habitus of the individual and the capital need to be focused on.

Author(s)	Study	Results
*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.

* *Study on the knowledge and skills gap specifically for SD education*

Table 2-1 indicates that seven of the eleven studies found that professionals in the computing field lack non-technical or soft skills, such as communications skills (negotiation or giving presentations), client-facing capabilities, leadership skills and relationship skills. Nine out of the eleven studies in Table 1 found that business-domain knowledge and skills, such as project management, business consultancy, business analysis and analytics, business process modelling, entrepreneurial traits and internationalisation skills are needed in the industry. Computing professionals also lack technical expertise, such as HCI/user interfaces, real-time systems design, software (requirements gathering and analysis; cost estimation; metrics; development; design; testing; reliability and fault tolerance; configuration and release management), web-based programming, mobile application development, security and cryptography, ERP, operating systems, multimedia and computer graphics.

A number of studies have been conducted to define the critical skills that are needed to perform IT-related jobs properly. The research of Bullen *et al*. (2009) examined workforce trends in IT-provider companies and encountered problems in the following areas: graduates who are not trained in areas that the marketplace is seeking; thin pipeline for specific technical skills; increasing pressure to source IT capability; and lag in university responsiveness to the needs of the marketplace. Keil *et al*. (2013) identified 19 skills as being the most critical for IT project managers in (IT) projects and ranked them based on their relative importance. The top five skills identified were leadership, verbal communication skills, scope management, listening skills, and project planning. They pointed out the fact that none of the top five skills was technical in nature. The knowledge requirements for three management levels (i.e. supervisory, middle and top) at two kinds of Taiwan company (manufacturing and services), as well as critical IS professional activities were investigated by Wu *et al*. (2007). They found that the importance of professional activities was viewed in significantly different ways by each level of management but the views of the two types of industry were similar.

Lee *et al*. (2001) found that IS professionals' competencies change as they gain experience in the workplace and they are required to have higher levels of technical skill in the early stages and higher levels of non-technical skills in the later stages of their careers. In their study Bailey and

Stefaniak (2001) identified the knowledge, skills, and abilities needed by computer programmers - "*Computer programmers write code to create software programs. They turn the program designs created by software developers and engineers into instructions that a computer can follow.*" (Department of Labor, 2014). Bailey and Stefaniak (2001) found that 23 of the 85 identified skills were considered as highly important. None of these skills was language or tool specific, indicating that individuals possessing general skills that allow them to work with others, solve problems, and adapt to the rapidly changing environment will be in higher demand than those who possess a great deal of skill with particular tools. In addition, 13 of the 23 skills were technical skills while ten were soft skills. Business concepts were not rated very highly. The top three skills are the foundation of a computer programmer's job, namely the ability to modify, write, and debug code. The next three skills indicated that listening to others, problem solving and teamwork are highly valued.

Plice and Reinig (2007) found that emphasising technical topics at the expense of business content may provide short-term benefits in transitioning to the workforce, but it might inhibit career advancement as graduates assume greater managerial responsibilities. Emphasising communication and teamwork skills, while maintaining the existing curriculum balance between business and technical content, is indicated as an appropriate strategy to align the computing curricula with the needs of industry.

Industry's view on the relevance of SD education is therefore determined not only by the knowledge and skills (technical, non-technical and business) acquired by graduates at university but also by their awareness of the proven techniques, challenges and expectations of their roles in employment.

2.6 CHALLENGES FOR SOFTWARE DEVELOPMENT EDUCATION

In the rapid changing environment of Information Technology, the challenges facing SD education are definitely not few and far between. In the following section some of the thorny issues for SD education are considered.

2.6.1 Low enrolments

Educators are acutely aware of the worldwide decline in enrolment for computer-related courses at universities that has occurred since 2000 (Akbulut & Looney, 2007; Granger *et al*., 2007; Huang *et al*., 2008; McGettrick, 2009) and a similar trend has been noted in South Africa (Alexander *et al*., 2011). The National Plan for Higher Education in South Africa (Department of Education, 2001) emphasises an endemic shortage of high-level professional and managerial skills in South Africa, especially in IT, engineering, and technological and technical occupations. The

government is particularly keen to increase enrolments in the broad field of information and communications technology, which the Cabinet identified as a key focus area for skills development (Department of Education, 2001).

The decline in enrolment, with bright minds going into other disciplines, results in low levels of hiring of new PhD's, which will restrict the capacity for research in the field (Lethbridge *et al*., 2007). The impact of lower student numbers resulted in some universities reducing the number of academic teaching and research positions in computing departments, and even in some cases, the closure of such departments (Huang *et al*., 2008).

A few studies have investigated the attraction to and retention of students in courses in computer science, and they have identified motivation, culture, pre-college experience, and confidence issues as contributing factors. In addition, initial positive experience with computing, matching requirements of the discipline with perceived abilities, narrow perceptions on computing careers and career expectations were identified as key factors that influence students' decisions, firstly, to pursue courses in computer science and, secondly, to study the field further (Klawe, 2001; Margolis & Fisher, 2002; Tillberg & Cohoon, 2005; Akbulut & Looney, 2007; Hill *et al*, 2010). Alexander *et al.* (2011) found that the perceived importance of ''Interest in the career field'' when choosing a career remains very high for students from computer and other disciplines, which is important for this study where relevance and interest go hand in hand.

Huang *et al.* (2008) found that the outsourcing of IT jobs to low-wage economies resulted in IS and CS students being concerned about their job prospects after graduation, understandably fearing unemployment. Some IS/CS students changed their major and some students who were considering IS/CS were prompted to reconsider, leading to a significant decrease in enrolment in IS and CS majors in recent years.

Merkofer and Murphy (2009) reported on a study in South Africa in which higher education institutions closer to the more rural areas highlighted two groups of students who do not successfully follow through with their chosen ICT-related course - students who are under-prepared and those of lesser means. When insufficiently prepared students enter tertiary education, they find the course work difficult and drop out early in their first year of study. Students from more disadvantaged backgrounds struggle to pay fees and as a result tend to drop out later in the year.

Computing departments often struggle to retain students because computing classes often start with programming. This can deter students who find programming difficult or uninteresting, but who would be very interested in working in areas, such as information management, project management or systems management (Guzdial *et al*., 2009).

Goode *et al*. (2006) found that the scientific heart of computer science is *lost in translation* at the secondary school level, and as a result the field continues to lose the participation and interest of a wide group of students, especially females.

According to Forbes (Solomon, 2013), there are 29 billionaires in the world who are younger than 40. A large number of them are still in their twenties and most of them are in the technology sector. An alarming fact is that these rich young men often did not complete their university degree and some did not even finish high school. One example is the 17-year old Nick D'Aloisio who developed the Summly app when he was 15 and Yahoo! bought it for $30 million. Bill Gates, founder of Microsoft, and Steve Jobs and Steve Wozniak, co-founders of Apple, all left university before they completed a degree. Although wealth is not everything, the question is whether these young people perceived the university education they received as relevant.

2.6.2 Shortage of software developers

The consequence of a decline in enrolments is a reduction in the number of new graduates, with current and projected shortages of skilled professionals in almost all computer-related fields in many countries of the world (Harris, 2012; McAllister, 2012; Bateman, 2013; Granger *et al*., 2007; Seymour *et al*., 2005). Lethbridge *et al*. (2007) also draw attention to this shortage of software professionals, indicating that only 40% of computer industry workers actually have a computing-related education. Most practitioners are merely skilled at programming in a few popular languages or at using specific technology products, such as database management and web-development tools.

In view of the discriminatory Apartheid past of South Africa, the Employment Equity Act (Act no. 55 of 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 since there is a shortage of black ICT professionals (Calitz, 2010) and on top of that, the shortage of women in the computing disciplines is as much a reality in South Africa as it is a worldwide phenomenon (Harris, 2012).

Merkofer and Murphy (2009) attribute the reason for difficulties in sourcing candidates to a limited talent pool to a lack of skills in the market place, the increasing frequency of specialised technologies that are company and industry specific, high salary expectations of skilled candidates, poaching from competitors and the brain drain.

2.6.3 Computing careers

One of the factors contributing to the low attraction to and retention of students in computing courses is narrow perceptions on computing careers. People in computing careers are

stereotyped as nerds and students who love working with others and engaging in creative activities therefore do not choose to study computer science at universities (Klawe, 2001). A positive public image of computing must be promoted whereby good students are attracted and the public gains respect for the field and the professionals who practise within it (McGettrick *et al*., 2004; Lethbridge *et al*., 2007).

Researchers suggest many causes for the leaky pipeline in computing courses, one of which being the lack of accurate information about IT careers (Goode *et al*., 2006; Kekelis *et al*., 2005; McGettrick, 2009). The research in South Africa by Seymour *et al*. (2005) showed that secondary school learners do not know what a degree in IS entails, although they are convinced that remuneration and occupational benefits are positive factors in the pursuing of this career direction. Gupta and Houtz (2000) identified a lack of understanding of the skills required to succeed in IT careers among learners. Kekelis *et al*. (2005) found a lack of career guidance and support from family to be contributing factors for the leaky pipeline. Dealing with students' negative perceptions regarding a career in IS is a critical issue for the IS discipline and for some IS programmes it may even be an issue of survival (Frolick *et al*., 2005; Huang *et al*., 2008).

Galpin *et al*. (2007) found that Introversion, Thinking and Judging are more likely personality types in the computing disciplines than in the general population and these have an effect on the career choices of students. Feldt *et al*. (2010) state that personality is one of the variables to consider when trying to understand software engineering judgments, decision making and performance. People's personalities influence the effectiveness and efficiency with which they perform a predetermined role in the software process (Acuña & Juristo, 2004). If a student's personality type does not match his/her planned career the effect can be detrimental to both the student and his/her future employer.

Most students initially get a shock when they first arrive in the workplace. The transition from study at university to work after graduation cannot be reduced to a simple formula based upon graduate credentials and employability skills (Clark *et al*., 2011). Students' effective adaptation to the workplace is essential for a successful computing career.

2.6.4 Software development candidates

McGettrick *et al*. (2004) emphasise that the process of SD is a central element of the discipline of computing, an important practical skill for computing, and an essential component of the undergraduate curriculum. Moreover, a strong correlation exists between programming ability and other computing skills, such as skills in abstraction, conceptualisation, design and evaluation. Students who have the ability to become professional software developers need to be identified; the other students can develop their ability to contribute to the wider range of computing activities;

and the students who seem likely not to succeed can swiftly be identified and appropriately advised (McGettrick *et al*., 2004).

2.6.5 Gender

An issue closely related to the attraction and retention of students is the gender composition in computing classes, the male domination being a matter of great concern for SD educators (Margolis & Fisher, 2002; Tillberg & Cohoon, 2005; Blum & Frieze, 2005; Hill *et al.*, 2010). Females do view computers as important and useful, but they are not necessarily interested (Bovée *et al*., 2007). Females in South Africa showed very limited interest in technology-related careers (Seymour *et al*., 2005; Bovée *et al*., 2007). Margolis and Fisher (2002) reveal that most boys describe an early and persistent magnetic attraction between themselves and computers, while girls link their computer science interest to a larger societal framework, such as medicine or the arts much more frequently.

Not only are females not interested in computers, but those who might have started out being interested, get lost along the way. The pipeline shrinkage problem for females in computer science is a well-known and documented phenomenon where the ratio of females to males involved in computing shrinks dramatically from taking computer courses in secondary school on through university and into IT careers (Gürer & Camp, 2002; Mirjana *et al*., 2011). Women are more interested in the social and cultural applications of the computer, while men are more attracted to the nuts and bolts of computers. Men also get more hands-on experience of the computer than women and women have a smaller technical appreciation of computers (Carlson, 2006). Women tend to see computers as equipment, whereas men view it as a toy, and in the process, men tend to substitute social skills with computer skills while women are simply not prepared to make that sacrifice (AAUW, 2000).

Women do not like the competitive and antisocial environment of computers; instead they prefer collaboration and relevance to the real world (Chou & Tsai, 2007; Frieze, 2007). Moreover, women favour a contextualised curriculum in which computing and technology in general are perceived as tools for solving humanity's problems and enriching humanity's experiences (Tillberg & Cohoon, 2005; Shotick & Stephens, 2006). A great proportion of men on the other hand have a greater technical appreciation of computers and they enjoy playing computer games (Carlson, 2006).

In South Africa, female university IT students predicted they would receive lower grades for the course than males; in reality they received quite similar grades (Galpin *et al*., 2003). The overall conclusion from research is that females consistently underestimate their technology skills and academic abilities regardless, of what their skills really are (Warschauer, 2007; Hill *et al*, 2010).

Margolis *et al*. (2000) state: "*For girls, trying to find their place in a culture that challenges whether they are "really into it" and a curriculum that assumes their learning will occur in the same sequence and timing as their male peers, too many conclude that they "just aren't interested*". The AAUW (2000) confirms the notion with what they call the "*we can, but I don't want to*" syndrome.

2.6.6 Dimensions of the field

There are five different undergraduate degree programmes in computing covering computer science, information systems, software engineering, computer engineering, and information technology. Knowledge about specialties in terms of how practitioners work, the distinct roles in software development and what their different educational needs are, is needed. This will allow for better-tailored programmes, simpler models of computing and a smaller core resulting in the provision of appropriate education for each student (Lethbridge *et al*., 2007; Shaw, 2000; McGettrick *et al*., 2004). Students have insufficient information about the subject and related jobs, and cannot differentiate between the different computing sub-disciplines (Courte & Bishop-Clark, 2009; Gupta & Houtz, 2000). The research by Seymour *et al*. (2005) showed that secondary school learners do not know what an IS degree entails and in terms of CS they only have a partial understanding of the degree. The weighting of 60% allocated to programming in the school curriculum of the subject Information Technology might also contribute to the misconceptions of the computing fields (Department of Education, 2011).

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). The dimensions of the field need to be fully comprehended in order to focus education appropriately.

2.6.7 Educators

Universities are struggling to recruit good-quality staff, resulting in insufficient talent being channelled back into the education system. A relatively low-paying job as a lecturer in computing is not a great attraction and many young lecturers use their job as a stepping-stone to better paying jobs in the IT industry (Gupta, 2005).

Wohlin and Regnell (1999) believe that traditional "lectures from a book" alone will not achieve educational goals. For many higher education institutions, student-centred approaches are part of strategic change plans, but most institutions are still using teacher-centred approaches (Muianga *et al*., 2013). It is vital for educators to address the ways they teach. Wohlin and Regnell (1999) propose teaching methods with realistic scenarios of software development that mimic

industrial best practice to achieve deeper understanding. In addition, educators in the software field can do better at exploiting technology to support the learning process itself (Shaw, 2000; Gupta, 2005; Oblinger & Oblinger, 2005).

Distance delivery via a variety of mechanisms is here to stay and educators therefore need to adjust from the traditional lecture format (Mead, 2009; Shaw, 2000; McGettrick *et al*., 2004). When university education is discussed, the topic of MOOCs (Massive Open Online Courses) cannot be ignored. Cator (2013) argues that any innovation that creates new opportunity on a massive scale, also poses new challenges. SD students are often considered the ideal candidates for MOOCs but Cator (2013) points out that one challenge is to provide equitable access and the second is making sure learners know how to use the technology in ways that improve learning.

Just as students and curricula have to stay updated in the rapidly changing world of technology, similarly the educators should not fall behind (Lethbridge *et al*., 2007). If only teaching is the sole purpose of some educators, there is a risk that they will become isolated from new developments and from contacts with industry (Wohlin & Regnell, 1999). The knowledge levels of educators need to be maintained so that they do not teach outdated content to the Net generation of learners.

2.6.8 Rapid change

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 for universities. In an evolving field, such as SD, the challenge is a fast response to changes in technology and designing curriculum guidelines for the future, while supporting current practice (Lethbridge *et al*., 2007; Shaw, 2000).

Topi *et al.* (2010) claim that often the administration at an institution is not aware of the resources, computing hardware, software, course offerings and laboratory resources needed for a viable program. The administration may also be unaware of the specialised classroom technology, library resources, or laboratory assistants essential for the proper education of computing undergraduates. Finally, the administration might not recognise the rapid turnover of knowledge in the field and the need for resources to support constant retooling of lecturers.

2.6.9 Knowledge and skills

Students in the computing fields are graduating with a lack of the skills that companies are searching for and therefore ironically not only is there a shortage of software developers but also a noteworthy level of unemployment amongst computing graduates (Bateman, 2013).

In addition to educating new students, the knowledge levels of the existing workforce need to be raised by providing effective means for practitioners to keep their skills current (Lethbridge *et al*., 2007; Shaw, 2000; McGettrick *et al*., 2004).

Students need to “learn how to learn”, as their professional activity will develop through decades according to unpredictable and often rapidly variable scenarios. A deep-rooted ability to continuously learn from daily work and professional opportunities needs to be promoted. It is a profound and extremely complex pedagogical challenge that requires new education skills, methods, and approaches (Fuggetta, 2012; O’Grady, 2012).

2.6.10 Research and innovation

The quality of and respect for educational research are a challenge and to make matters worse, funding agencies only sporadically support educational research, and industry tends only to be interested in those aspects that have to do with training (Lethbridge *et al*., 2007). Educational research is often not recognised as ‘‘legitimate” technical research and Mead (2009) challenges educators to continue to push for the recognition of educational research as a legitimate field of research by raising the prestige and quality of educational research in computing.

Universities regularly face pressure from potential employers to deliver graduates with immediately useful skills, and in the process research and innovation in the field are sacrificed (Gupta, 2005; Shaw, 2000). Wohlin and Regnell (1999) propose that the academic staff and postgraduate students should be involved in both research and education. This would provide important feedback to the research and at the same time it would ensure that the latest findings in software development are included in the courses.

2.6.11 Pre-university issues

Universities recruit students from high schools but students come from a variety of backgrounds and they offer a wide range of entry qualifications. No universally accepted pre-university computing qualification for entry exists; many programmes require some level of mathematics but few, if any, require a prior qualification in computing (McGettrick *et al*., 2004).

The vocational nature of important aspects of the subject makes it a popular choice for many students but the motivation for this choice is not always academically driven. The wide exposure of learners to computers in many routine ways, such as game playing often misleads students about the nature of the subject (McGettrick *et al*., 2004).

2.7 THE ROLE OF THE UNIVERSITY IN RELEVANT SOFTWARE DEVELOPMENT EDUCATION

Gone are the classical times when students came from far and wide to sit at the knee of their tutor. Nowadays, a vision exists of higher education as market driven and determined by a culture of enterprise (Bayne & Ross, 2011). There are two stakeholders on either side of this market of higher education, with students on the one side and the industry on the other. Students want a relevant education and the industry wants future employees to receive a relevant education. The role of the university in providing a relevant education has always been and continues to be a relatively uncontentious role (Reisman, 2004).

Until the mid-1980s, parents were interested in seeing that their children obtained a well-rounded university education. University was a one-time experience, a place for socialisation and development of the life skills needed to understand and function in society (Wells & Sevilla, 2001; Reisman, 2004). Two major shifts changed the face of higher education. Firstly, an economic shift brought the non-traditional student to the university. Older people came back to university as a result of layoffs or changes in technology and these students are not looking for the "university experience". Secondly, there was a shift in the things the community values. Parents and their children started to value a career-focused education consistent with the needs and demands of industry, instead of a well-rounded education (Wells & Sevilla, 2001).

Traditionally, universities prepared students either for graduate work or for employment, but universities regularly face pressure from potential employers to deliver graduates with immediately useful skills (Gupta, 2005; Shaw, 2000).

"Many universities have established forums to encourage dialogue between industry and academia, in an attempt to bridge the gap between what the former wants and the latter teaches. These meetings are often unsuccessful, with industry annoyed at academia for not focusing on practical subjects, while academia sees industry as narrow and shallow" (Wells & Sevilla, 2001).

To effectively fill 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 correctly tackle the problems that they will face during their professional career (Loftus *et al*., 2011; Bothe *et al*., 2009).

Reisman (2004) argues that the primary qualification of new employees should be their solid undergraduate education where the curriculum focused on a common body of knowledge; a set

of basic topics and principles, in order that new employees can easily be trained by their employer for entry-level project work using particular (vendor) products. Shaw (2000) states that universities regularly face pressure from potential employers to sacrifice systematic understanding for immediately useful skills and each university must find its own balance between immediate and long-term knowledge.

Krakovsky (2010) states that: "*The model of the 18- to 22-year old going to a residential campus … and watching football games and that kind of stuff will be around for a long time, but the demand for other models is coming from lots of other areas and demographics".*

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

2.7.1 Curricula

The ACM, IEEE Computer Society and AIS have joined forces to regularly create and update curriculum guidelines for undergraduate degree programmes in the following computing disciplines, namely computer science, computer engineering, information systems, information technology and software engineering (Shackelford *et al*., 2006). Sahami *et al*. (2011) state that the development of curricular guidelines in the computing disciplines is particularly challenging given the rapid evolution and expansion of the field. Moreover, the growing diversity of topics in computing and the integration of computing with other disciplines create additional challenges and opportunities in defining computing curricula. Hassan (2008) warns that as a result of the increasing breadth of content introduced into the computing field, undergraduates may become multidisciplinary illiterates. The multidisciplinary nature of the content produces graduates as generalists instead of specialists. Hassan (2008) further warns that without a clear body of knowledge built within a succinct codified framework for students, the computing field will continue to struggle to recruit new members into the field.

The Joint Task Force on Computing Curricula (2013) emphasises that the education that students receive must adequately prepare them for the workplace in a more holistic way than simply conveying technical facts. Students will gain some soft skills and personal attributes through the general university experience (e.g. patience, time management, work ethic, and an appreciation for diversity), and others through specific curricula (Joint Task Force on Computing Curricula, 2013). Bailey and Stefaniak (2001) point out that it remains a struggle to incorporate opportunities that foster the development of necessary nontechnical skills into a traditional technical academic curriculum.

Gupta (2005) is of the opinion that universities continue to follow a curriculum which is incongruent with the latest international trends and technologies and students therefore require considerable training before they are able to contribute effectively to the organisation that employs them. It can be argued that 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 that were typically designed for Western realities and that do not address local needs (Bass & Heeks, 2011; Ezer, 2006).

This chapter is concluded with a quote from the paper of Alma Clayton-Pedersen and Nancy O'Neill (2005) entitled "*Curricula designed to meet 21st-century expectations*":

Future careers will require higher levels of education than in the past. That education must enable individuals to discover what they need to know rather than just having static knowledge. Society will need college graduates with mental agility and adaptability. If this is the goal of education, colleges and universities must re-examine how that goal is achieved.

The literature covered in this chapter investigated the software development class, focusing on the students and the educators. Furthermore, a review of the software development workplace was done with attention to the software developers and their employers. The problems and challenges facing three role players in software development education, namely the students, the university and the industry were investigated. Lastly, the role of the university in relevant software development education was considered with a specific focus on curricula.

In the literature no study appears to have investigated if there is a gap between students' needs and their education, or whether SD education is relevant from the perspective of the students. Furthermore, the view the industry has of the new graduates and the problems, challenges and solutions in respect of software development education has not been explored in detail. Moreover, despite reports of a gap between industry needs and software education, the gap has mostly been explored in developed countries and in quantitative studies. Therefore, there is a gap in literature that needs to be addressed.

Chapter 3

Article 1

The relevance of software development education for students

Mrs J. Liebenberg[a]

Prof. M. Huisman[a]

Prof. E. Mentz[b]

[a]**Department of Computer Science and Information Systems,**
Faculty of Natural Sciences,
North-West University,
Potchefstroom,
2520,
SOUTH AFRICA

[b]**Faculty of Education Sciences,**
North-West University,
Potchefstroom,
2520,
SOUTH AFRICA

Manuscript published in: ***IEEE Transactions on Education (TOE)***

The Relevance of Software Development Education for Students

Janet Liebenberg, Magda Huisman, and Elsa Mentz

***Abstract*—Despite a widely-acknowledged shortage of software developers, and reports of a gap between industry needs and software education, the possible gap between students' needs and software development education has not been explored in detail. In their university education, students want to take courses and carry out projects that clearly relate to their lives and their goals. This paper reports on a quantitative study of 297 software development students. The analysis of the results suggests that software development education has a predominantly social relevance to students and also has moderate personal and professional relevance. The following approaches are recommended to improve students' views of the relevance of software development education: use various learning environments; pay special attention to female students, students who did not have IT as a school subject, and students who rate their own academic performance as low; update educators on the latest developments; design programs to appeal to students and to meet societal demands.**

***Index Terms*—Computer industry, computer science education, curriculum development, engineering students.**

I. Introduction

SOUTH Africa has a shortage of software developers, which is a worldwide phenomenon. To remedy this, more software development students are needed, but will students take courses they do not find relevant?

In promoting education, software developer Bill Gates [1] focuses on his foundation's "3 R's": Rigor, Relevance, and Relationships. The central pillar of relevance highlights the need for students to be exposed to courses and projects that clearly relate to their lives and their goals.

Numerous studies have investigated the relevance of education at university level, but these have predominantly been concerned with ways to make education relevant to the needs of industry. Lethbridge [2] analyzed the relevance and depth of the knowledge that software professionals had gained as part of their university education, and identified a significant mismatch between software education and industry in terms of the knowledge needed by software engineers to meet industry's requirements. Other researchers reported similar gaps [3]–[9]; for Lethbridge [10], filling them is one of the most critical challenges for educators. In his paper "Higher Education: Who Cares What the Customer Wants?," Reisman [11] argues that studies in higher education often neglect to collect data from two of higher education's most important players—faculty and students. No study appears to have investigated if there is a gap between students' needs and their education, or whether software development (SD) education is relevant from the perspective of the students. In the light of the shortage of software developers, this study aimed to investigate the relevance of SD education to students.

Manuscript received July 10, 2014; revised September 15, 2014 and November 13, 2014; accepted December 04, 2014. The work of J. Liebenberg was supported by the National Research Foundation (NRF).

J. Liebenberg and M. Huisman are with the Department of Computer Science and Information Systems, North-West University, Potchefstroom 2520, South Africa (e-mail: janet.liebenberg@nwu.ac.za; magda.huisman@nwu.ac.za).

E. Mentz is with the Faculty of Education Sciences, North-West University, Potchefstroom 2520, South Africa (e-mail: elsa.mentz@nwu.ac.za).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TE.2014.2381599

A. Clarification of Terminology

For the purpose of this study, it is necessary to explain and clarify certain key terms.

Software development (SD)—The IEEE, ISO, and IEC define the software development process as the process by which user needs are translated into a software product [12]. For the purpose of this study, SD will refer to the orderly process of developing software products through successive phases.

SD classes will refer to the courses where the process of developing software is taught.

SD students will refer to the students taking the SD classes.

II. Conceptual Framework

A. Concept of Relevance

Relevance is broadly defined as being closely connected or appropriate to the matter in hand, or having practical and especially social applicability. Relevance in the educational context can be defined as the applicability of what is taught to the needs and interests of students and society [13].

Three relevance perspectives suggested by Holbrook [14] are used in this paper to analyze relevance:

- Social relevance—the "useful in society" perspective, which is a perceived societal need;
- Personal relevance—the "interest" perspective, which directly relates to concerns in the students' immediate environment or areas of interest;
- Professional relevance—the "important for the course they are studying" perspective; this relates to the content of the curriculum, which has to be interesting and useful to students.

B. Student in the Software Development Class

Most students in current university classes belong to the so-called Net generation, also known as Generation Y. The Net generation is characterized by students who may have never known life without the Internet. Their early and omnipresent exposure to technology has defined their styles, their modes of

communication, their learning preferences, their social choices, and their entertainment preferences [15].

A few studies have investigated the attraction of students to, and their retention in, courses in computing disciplines; they identified motivation, culture, pre-college experience, and confidence issues as contributing factors. Also, initial positive experiences with computing, matching requirements of the discipline with perceived abilities, narrow perceptions of computing careers, and career expectations were identified as key factors that influence students' decisions to, first, pursue courses in the computing fields and, second, to study the field further [16], [17].

An issue closely related to the attraction and retention of students is the gender composition in computing classes; that these are predominantly male is a matter of great concern for educators [17]–[19]. Women have been found to enjoy using existing systems more than developing new ones, and are attracted when they recognize computing as a form of communication, a means of creative self-expression, or as a path to an occupation where they may be of assistance. Moreover, women prefer a contextualized curriculum in which computing and technology in general are perceived as tools for solving humanity's problems and enriching humanity's experiences [19], [20]. Many of the men, however, have a greater technical appreciation of computers and enjoy playing computer games [16]–[18]. Margolis and Fisher [18] reveal that most men describe an early and persistent magnetic attraction between themselves and computers—for them the computer is the ultimate toy.

C. Software Development Education

In the last few decades, remarkable strides have been made in software and technical development; these developments continue unabated. Not only is the dependence on software increasing, but the character of software production itself is changing—and with it the demands on software developers [15], [21], [22]. The diversity of software applications requires adaptability in responding to client needs, and the diversity of clients and contexts requires the ability to discriminate among criteria for success. This presents new challenges for the education of software developers [21], [22].

Several studies suggest a gap between the knowledge and skills demanded by industry and the knowledge and skills gained by graduates of university computing courses. An analysis of computing curricula from an industrial perspective, in terms of technical and nontechnical knowledge and skills, revealed the following gaps: problem-solving and project management skills, knowledge of business, information technology (IT) business consultancy, security, end-user computing, soft skills related to core knowledge, knowledge related to leadership, and negotiation or giving presentations [3], [6], [8].

Lethbridge's [2] study identified gaps in technical knowledge between the education received and the knowledge required from the viewpoint of the SD industry. Lethbridge surveyed professionals with industry experience and found gaps in the following: HCI/user interfaces, real-time system design, software cost estimation, software metrics, software reliability and fault tolerance, and requirements gathering and analysis. Kitchenham [5] ran a similar study, but with software engineering (SE) graduates; the results were quite different, with gaps appearing to relate to Web-based programming, project management, configuration and release management, multimedia, security and cryptography, and computer graphics. Both studies found that mathematical topics appear to be taught in greater depth than required in industry. Another similar study was performed in the Finnish context by Surakka [7], who surveyed three sets of stakeholders (software developers, professors and lecturers, and Master's students) on the relevance of various matters. His results coincide with Lethbridge's [2] and Kitchenham's [5] on the excessive importance attached to mathematics-related topics at university, and with Kitchenham's findings of the increased importance of Web-related subjects and skills in industry.

Computing departments often struggle to retain students because computing classes often start with programming. This can deter students who find programming difficult or uninteresting, but who would be very interested in working in areas such as information management, project management, or system management [23].

Education for software developers is offered largely in traditional classroom formats. Software developers are still educated much as they have been for years past. However, a major challenge in educating software developers is that in courses whose primary emphasis is current technology, as that technology becomes obsolete, so will most of the knowledge taught. The changing character of software and external pressures on educational institutions will require changes in what, and how, software developers are taught [22].

Students need to see the relevance of teaching and learning as it applies to them personally (their own lives, their career expectations, the wishes of their parents), or as it applies to society (wishes of the community, employers, the university), or as it applies to them professionally (the content/curriculum is meaningful and interesting) [14]. The realities of the software industry for which the students need to prepare have shifted away from those of the foundational beliefs and practices of many of their educators. Educators need to become familiar with the students' learning challenges and investigate their distinctive qualities and personal preferences. Educators must identify the necessary ingredients for successful teaching and learning to improve teaching practices and course delivery methodologies [15]. This study therefore investigates the relevance of software development education from the point of view of students.

III. Research Method

A. Research Design and Participants

In this quantitative study, conducted at a university in South Africa, 386 questionnaires were posted as an assignment on the university's e-learning system to students in the SD classes of four academic year groups. A total of 297 usable responses were received, an overall response rate of 76.9%. The participants included 276 B.Sc. undergraduate students and 21 students enrolled for the subsequent B.Sc. (Hons.) in computer science (CS) and information systems (IS). The undergraduate students had a higher response rate (79.1%) than the graduates (56.8%). Most of the undergraduate students follow the 3-year B.Sc. in IT and CS program in the Department of Computer Science and Information Systems. A subsequent 1-year B.Sc. Honor's degree in CS and IS is offered, which gives access to a

TABLE I
PROFILE OF RESPONDENTS ($N = 297$)

		Number (%) of students
Gender	Male	222 (75%)
	Female	75 (25%)
Academic Year	1	145 (49%)
	2	76 (26%)
	3	55 (19%)
	4 (Hons)	21 (7%)
Self-rated academic performance	<= 59%	68 (25%)
	60% – 74%	158 (57%)
	>= 75%	50 (18%)
IT[1] as school subject	Yes	142 (48%)
	No	155 (52%)
Access to a computer since grade 1	Yes	170 (58%)
	No	123 (42%)

[1]*Information Technology (IT) is a subject that can be taken from Grade 10 to Grade 12 in South African schools, focusing primarily on programming skills.*

Master's degree in CS.[1] These programs are based on the Informatics Curriculum Framework of the International Federation for Information Processing (IFIP), so the results of this study could therefore also be of value to other university courses based on the IFIP framework. The IFIP framework refers to major widely accepted and widely implemented informatics curricula by professional organizations, such as the ACM, IEEE, AITP, and AIS.

Table I provides a summary of the biographic data.

B. Data Collection and Instrument

An initial list of questions was developed by both writing new items and adapting items from available surveys, such as ROSE [24], the SAQA's list of "Critical Cross-field Outcomes," and topics commonly found in computer science programs. Once the initial questions were generated, the instrument was sent to be refined by three industrial and two academic experts familiar with tertiary computing programs, who added additional topics. The experts' feedback served as the basis for correcting, refining, and enhancing the questions, resulting in a questionnaire with a pool of 123 items. The first section of the questionnaire gathered information on the biographic data of the respondents as shown in Table I. The remainder of the questionnaire was divided into four domains.

The first domain, "Out of class," investigated personal relevance, with 12 items that gathered data on the students' out-of-class experiences, such as using the Internet and developing a software system. The participants were asked: "How often have you done this outside formal education?" with a five-point Likert response scale: Never/Once or twice/I don't know/Quite often/Very often.

The second domain, "In class," investigated personal, social and professional relevance with 33 items, enquiring about their perceptions of their SD classes, such as their enjoyment and interest in the classes.

The third domain, "My career," had 12 items that investigated social relevance, gathering data on their notions of a future career, such as what they thought would be expected from a good software developer. The second and third domains used a five-point Likert response scale from 1 (Strongly disagree) to 5 (Strongly agree).

The last domain, "What I want," had 66 items to investigate professional relevance, testing their interest in curriculum topics such as specific programming languages or extreme programming. The participants were asked: "How interested are you in learning about the following?" on a five-point Likert response scale: Not interested/Slightly interested/Neutral/Quite interested/Very interested.

Factor analysis was applied to the 123 items to reduce the variables to a smaller number of factors. The 297 responses were examined using principal components factor analysis; the 123 attitude items yielded 20 interpretable factors, named according to their main context, and three single item variables. A Cronbach's α coefficient was calculated for each of the 20 factors and was found (see Table II) to be reliable ($\alpha \geq 0.60$).

The 23 factors were further divided into three perspectives of Personal relevance, Professional relevance, and Social relevance using Holbrook [14] as a guideline. A Cronbach's α coefficient was also calculated for the three perspectives and was found (see Table III) to be reliable ($\alpha \geq 0.60$). Table IV shows the division of the factors.

TABLE II
RELIABILITY COEFFICIENTS OF FACTORS

Factor	Cronbach's alpha (α)
Out_of_class_Basic_use	0.743
E-mail use	N/A
Internet use	N/A
Skype use	N/A
Out_of_class_Advanced_use	0.605
In_class_Learn	0.876
In_class_Perceptions	0.745
In_class_Attitudes	0.853
In_class_Importance	0.724
In_class_Teaching	0.678
Career_Attitudes	0.843
Career_Skills	0.772
Real-time and systems programming	0.935
Mathematics and statistics	0.902
Software management	0.918
General software design	0.907
Web and Mobile technologies and Games	0.934
Computer science theory	0.934
Specialized application techniques	0.926
Software engineering methods	0.916
Essential subsystem design	0.897
Computer hardware and other electrical and computer engineering	0.946
Alternative software engineering methods	0.880

**See http://tinyurl.com/SoftDevEd for the items in each factor*

TABLE III
RELIABILITY COEFFICIENTS OF PERSPECTIVES

Perspective	Cronbach's alpha (α)
Personal relevance	0.652
Professional relevance	0.909
Social relevance	0.704

C. Threats to Validity

The fact that all the survey respondents were from a single university can raise the question of the applicability of the

[1]See extracts of the programs from the yearbook of the Faculty of Natural Sciences at http://tinyurl.com/BSciITandCS

TABLE IV
BASIC ANALYSIS OF 23 FACTORS AND DIVISION OF RELEVANCE PERSPECTIVES

Factor	Relevance perspective	Mean*	Standard deviation
Internet use	Personal	4.862	0.456
E-mail use	Personal	4.539	0.881
Career_Attitudes	Social	4.488	0.534
Out_of_class_Basic_use	Personal	4.294	0.706
In_class_Learn	Professional	4.074	0.692
Career_Skills	Social	4.074	0.672
In_class_Importance	Social	3.889	0.664
Web and Mobile technologies and Games	Professional	3.835	0.996
In_class_Attitudes	Personal	3.759	0.810
General software design	Professional	3.759	1.020
Specialized application techniques	Professional	3.757	0.969
Real-time and systems programming	Professional	3.720	1.075
Essential subsystem design	Professional	3.696	1.069
Software engineering methods	Professional	3.527	0.989
Computer hardware and other electrical and computer engineering	Professional	3.517	1.049
Software management	Professional	3.456	1.078
In_class_Teaching	Professional	3.455	0.648
Mathematics and statistics	Professional	3.363	1.125
Alternative software engineering methods	Professional	3.338	1.179
Skype use	Personal	3.327	1.633
Computer science theory	Professional	3.199	1.215
In_class_Perceptions	Personal	2.944	0.882
Out_of_class_Advanced_use	Personal	2.697	1.068

* *Likert-style responses were ranked from 1 to 5 respectively*

study. However, the university's curricula are based on the IFIP framework that is linked to major, widely accepted, and widely implemented curricular efforts and to high-quality bodies of knowledge undertaken by professional organizations, such as the ACM, IEEE, AITP, and AIS.

The participants were asked to rate their academic performance on a percentage scale, which could be viewed as a threat to validity, but according to Ackerman [25], individuals generally have accurate views of their relative standing on abilities and knowledge. Furthermore, the participants' actual academic performance could not be compared sensibly—the first-year students had only completed three SD courses, whereas the Honor's students had completed up to 21 SD courses by the time they completed the survey.

An instrument that measures the relevance of SD education for students could not be found. The researchers adjusted surveys for measuring relevance to the industry and surveys measuring relevance to school learners and sent their finding to industrial and academic experts familiar with tertiary computing programs to make provision for validity. However, additional validation is recommended. Future work can include a study to design a standardized instrument to specifically address the relevance of software development education for students.

D. Data Analysis

Basic analysis was done by calculating the mean values and standard deviation of each of the 23 factors, as well as those of the three relevance perspectives. The statistical tests used in the analysis varied as necessary to match the metric being analyzed. Three groupings were identified based on gender, academic performance, and IT as school subject. All the groupings were tested for significant differences between means in the different factors using T-tests, except academic performance where an ANOVA test was used. Chi-square tests and Spearman's rank correlation analysis were also used to analyze relationships between the groupings and the factors. When the results of these interaction analyses are reported, only the significant interactions or primary effects will be discussed. Since a convenience sample was used instead of a random sample, the p-values will be reported for the sake of completeness but will not be interpreted.

TABLE V
BASIC ANALYSIS OF THE 3 RELEVANCE PERSPECTIVES

Relevance perspective	Mean	Std. Deviation
Personal_relevance	3.789	0.557
Professional_relevance	3.605	0.715
Social_relevance	4.155	0.501

IV. RESULTS AND DISCUSSION

A. General Results

Table IV shows that the mean values of 21 of the 23 factors are relatively high. Out_of_class_Advanced_use is the factor showing the lowest mean, which indicates that students have relatively little out-of-class experience with developing a software system for somebody, writing a computer program, and building a device. Further analysis indicated that gender played a significant role in the Out_of_class_Advanced_use of computers by students. The other factor showing a lower mean is In_class_Perceptions, indicating that students tend to have a perception that SD is a difficult subject area, that the volume of work is high, and that the instruction in an SD class is rigorous. In addition, some students were anxious/stressed when doing practicals; they found SD difficult to learn and were not confident that they would obtain their degree. Further analysis indicated that academic performance played a significant role in the In_class_Perceptions of students.

It is not surprising that these students had high mean values for Internet use, e-mail use, and Out_of_class_Basic_use. The use of Skype is lower, but the low access to computers (see Table I) and the high cost and slow speed of the Internet in South Africa might explain that figure.

The factor In_class_Learn had a high mean with some of the items contributing to the high mean being the importance of "Project work," "Teamwork," and "Communication skills." The students therefore view the so-called "soft skills" as relevant in their education.

Students showed the most interest in the subject topics in the factor Web and Mobile technologies and Games. This is a significant finding in view of the findings of Kitchenham [5] and Surakka [7] of the increased importance of Web-related subjects and skills in industry, but even more significant is the interest of students in Mobile technologies and Games since these topics have not emerged in studies in industry.

Table V shows SD education has high social relevance to these students and also has moderately high personal and professional relevance.

TABLE VI
GENDER DIFFERENCES IN VIEWS ON THE RELEVANCE OF SD EDUCATION

Men (N=222) Mean	SD	Women (N=75) Mean	SD	Effect size	P
Out_of_class_Basic_use					
4.44	0.55	3.87	0.93	0.61*	<0.001
Out_of_class_Advanced_use					
2.91	0.99	2.07	1.02	0.83**	<0.001
In_class _Attitudes					
3.93	0.69	3.25	0.94	0.72*	<0.001
Real-time and systems programming					
3.91	0.93	3.14	1.26	0.61*	<0.001
Computer hardware and other electrical and computer engineering					
3.71	0.91	2.93	1.21	0.64*	<0.001
Alternative software engineering methods					
3.52	1.05	2.79	1.38	0.52*	<0.001

* *medium practically significant difference (d>=0.5)*
** *large practically significant difference (d>=0.8)*

B. Gender

Gender differences were analyzed with a T-test. Table VI shows significant differences in means between the male and female students in six of the 23 factors. There is a large practically significant difference in Out_of_class_Advanced_use and a medium practically significant difference in Out_of_class_Basic_use of male and female students. The male students had significantly more out-of-class experience with developing a software system for somebody, writing a computer program, and building a device. In addition, it is the male students who play computer games, open devices to find out how they work, install programs on computers, read about computers in books or magazines, and download music from the Internet, significantly more than the female students. The other factor that shows a medium practically significant difference between male and female students is In_class_Attitudes. More than their female counterparts, the male students are interested in, like, and enjoy the SD courses; they and their parents want them to become software developers; and SD opened their eyes to new and exciting jobs. These differences in the genders' interests confirm the findings of numerous gender studies in computing.

With three factors showing a medium practically significant difference between male and female students' interest toward certain subject topics (Real-time and systems programming; Computer hardware and other electrical and computer engineering; Alternative software engineering methods), previous studies confirmed that male students exhibit a greater technical appreciation, and that female students prefer existing systems rather than alternative methods, such as extreme or real-time programming. Male students showed the least interest in Computer Science theory and the most interest in Web and Mobile technologies and Games topics. The female students were most interested in Mathematics and Statistics, which is in sharp contrast to the results of [3], [5], and [7] regarding the industry's view of the excessive importance attached to mathematics-related topics at university. The female students' interest in Mathematics might again be explained by their preference for existing systems since it is a familiar subject to them.

When the gender differences for the three relevance perspectives were analyzed by using a T-Test, a medium practically significant difference ($\mathrm{d} = 0.56, \mathrm{p} < 0.001$) between male and female students was found in terms of personal relevance. The fact that SD education has more personal relevance to the male students can be explained by their more intense use of the computer and their more positive attitude toward their classes.

TABLE VII
DIFFERENCES BASED ON SELF-RATED ACADEMIC PERFORMANCE

Factor		Effect size	p
In_class _Perceptions	<= 59% *vs* 60% –74%	0.36	
	<= 59%*vs*>= 75%	1.12**	<0.001
	60% – 74%*vs*>= 75%	0.28	

** *large practically significant difference (d>=0.8)*

C. Academic Performance

The questionnaire listed the applicable SD courses in the program; students were asked to rate their overall academic performance in those courses on the scales: 75%–100% = between 8 and 10; 70%–74% = 7; 60%–69% = 6; 50%–59% = 5; Below 50% = 4. The results were then reported in three groups: $<= 59\%(\mathrm{n} = 68)$; 60%–74% $(\mathrm{n} = 158)$; $>= 75\%(\mathrm{n} = 62)$.

The results of an ANOVA in Table VII indicate a practically significant difference between the $<= 59\%$ students and the $>= 75\%$ students in terms of their In_class_Perceptions. The $>= 75\%$ students had a significantly more positive perception of the SD class. As can be expected, the $<= 59\%$ students had a perception that SD is difficult, the volume of work is high, and the instruction in the SD class is rigorous. They were anxious/ stressed when doing practicals, found SD difficult to learn, and were not confident that they would obtain their degree.

A Spearman rank correlation analysis was used to analyze the correlation between the self-rated academic performance and all the factors and confirmed the results in Table VII with a medium practically significant relationship $(\mathrm{r} = -0.353, \mathrm{p} < 0.001)$ between the students' academic performance and their In_class_Perceptions. The lower the students rated their academic performance, the more negative the perception they had of the SD class.

When the differences in self-rated academic performance with the three relevance perspectives as variables were analyzed by using an ANOVA, no significant differences were found. Students' self-rated academic performance does not determine if SD education has more personal, social, or professional relevance to them.

D. IT as a School Subject

IT refers to one of the subjects that can be chosen from grades 10 to 12 in some South African high schools. The school subject IT focuses primarily on programming, which means that some of the students enter the SD classes with 3 years of experience in programming. The school subject IT is, however, not an admission requirement for the B.Sc. degree in IT and CS program.

A T-test was used to determine if there were significant differences between the 142 students who had IT as a school subject and the 155 students who were only formally introduced to SD once they came to university. Table VIII shows that there is a high practically significant difference in their Out_of_class_Advanced_use, with the students who had IT showing significantly

TABLE VIII
DIFFERENCE IN VIEWS BETWEEN STUDENTS WHO TOOK IT AS SCHOOL SUBJECT AND THOSE WHO DID NOT

Factors	Effect size	P
Out_of_class_Advanced_use	0.95**	<0.001
In_class_Perceptions	0.54*	<0.001
In_class_Attitudes	0.62*	<0.001

* medium practically significant difference (d>=0.5)
** large practically significant difference (d>=0.8)

more SD interest and experience outside their formal education. There was a medium practically significant difference between the two groups of students in terms of their In_class_Perceptions and their In_class_Attitudes. Confidence and initial positive experience have been identified as contributing factors in the attraction and retention of students in computing courses [16], [17]. The students without IT as a school subject were less confident that they would obtain their degree. Since not all students can take IT at school, this result might be a compliment to the schools offering IT because they contribute to students' initial positive experience, interest, and confidence, and therefore the relevance of SD education for students.

A Crosstab was used to determine if there was an association between self-rated academic performance and students who had taken IT; contrary to what might have been expected, the result indicated no statistically significant association. In other words, having taken IT as a school subject did not influence a student's academic performance in SD at university level. The above findings therefore suggest that although having taken IT as school subject does not impact students' academic performance, the students who did not take IT still seem to lack confidence, and their perceptions of and attitudes toward their SD classes are less positive.

When a T-test was done to analyze the differences between the students who took IT as a school subject and those who did not, with the three relevance perspectives as variables, a medium practically significant difference $(\mathrm{d} = 0.70, \mathrm{p} < 0.001)$ in personal relevance was found between the two groups. SD education has more personal relevance to the group who had taken IT as a school subject, and it is mostly explained by their advanced use of the computer, their In_class_Perceptions, as well as In_class_Attitudes.

V. CONCLUSION AND RECOMMENDATIONS

Labaree [26], who works in educational research, emphasizes that, when considering relevance, the question "useful to whom and for what?" needs to be answered because there is a wide array of stakeholders. In this study, the focus was on the largest stakeholder group in SD education, the SD students, including the females, poor performers, and students who did not have IT as school subject.

SD education has high social relevance to students; in other words, they view it as useful to society. Students acknowledge the fact that more software developers are needed and what they learn in the SD classes could lead to improved career opportunities for them.

SD students find moderately high personal relevance in their education, therefore their education does relate to concerns in their immediate environment or area of interest.

SD education also has moderately high professional relevance to students, meaning that the content of the curriculum is relatively interesting and useful to them. Students are less interested in topics such as computer science theory and more interested in Web and mobile technologies and games.

Certain students view SD education as more relevant than others. Male students, students who had IT as a school subject, and students who rate their academic performance as high $(>= 75\%)$ rate SD education as more relevant.

Effort should be made to further improve the relevance of SD education for all students. The following is recommended for SD education.

Use Different Learning Environments: For computing departments to offer relevant SD education, attention should not only be paid to what students are taught, but also to how they are taught.

- *Game-based learning.* Since students show such a great interest in Game development, computing departments at universities could explore game-based learning environments to satisfy students' appetite for games.
- *Project-based learning.* The students in this study view project work as important, and therefore project-based learning can also be used.
- *Pair learning.* Since students view teamwork and communication skills as important, pair learning (the use of pair programming in education) can therefore be exploited. Pair learning has been found to boost self-confidence and can therefore aid in teaching students who did not have IT as a school subject [27]

Pay Special Attention to Certain Groups of Students:

- *Female students*. Since most female students prefer to develop software that solves humanity's problems and enriches humanity's experiences, the SD curriculum and especially practical assignments and projects must be contextualized to appeal to the female students so that they can find more personal relevance in SD education. In addition, pair learning has been found to benefit specifically female students [28].
- *Students who did not have IT as school subject*. Lecturers must design and offer a bridging course early in students' first year to provide students who did not have IT with an initial positive experience of SD. Furthermore, students often learn more from each other than from lecturers, and the students who had IT as a school subject could be appointed as facilitators in the practical labs and assist the less experienced, as well as the students performing poorly.
- *Students who rate their academic performance as low.* University courses will always have poor performers, but in the light of the shortage of software developers, educators should attempt to retain these students by using different learning environments, such as pair learning, which has been found to improve academic performance [29].

Put in Place Mechanisms to Ensure That Educators Stay Abreast of the Latest Developments in the SD Field: These mechanisms can include visits by lecturers to the software industry, guest lectures by industry experts, and joint university–industry projects.

Design Programs to Appeal to Students and to Meet Societal Demands: There is a great demand for software developers and

students need to be informed from their early study years about the career possibilities and needs of the software industry. Curricular design must align the needs of students with the needs of industry. An emphasis on topics that do not interest students, such as computer science theory, can lead to a lower retention of students, but not emphasizing that topic could result in students being unprepared for the demands of the software industry. When students are aware of the specific needs of industry from their early study years, they might perceive the industry in a more positive light.

SD educators and curriculum developers should take cognizance of what makes SD education more relevant from the perspective of students in order to attract more students to SD classes—and to retain them. Enhancing the relevance of SD education is a profound and extremely complex pedagogical challenge that requires new education skills, methods, and approaches if the demand for software developers is to be met.

The paper focused on the needs of SD students; future research could include the needs of the SD industry to study the compatibility between student needs and industry needs.

ACKNOWLEDGMENT

Opinions expressed and conclusions arrived at are those of the author and are not necessarily to be attributed to the National Research Foundation (NRF).

REFERENCES

[1] B. Gates, "The 3 R's solution," 2005 [Online]. Available: http://www.gatesfoundation.org/Media-Center/Speeches/2005/02/Bill-Gates-2005-National-Education-Summit

[2] T. C. Lethbridge, "Priorities for the education and training of software engineers," *J. Syst. Softw.*, vol. 53, no. 1, pp. 53–71, 2000.

[3] A. Moreno, M. Sanchez-Segura, F. Medina-Dominguez, and L. Carvajal, "Balancing software engineering education and industrial needs," *J. Syst. Softw.*, vol. 85, no. 7, pp. 1607–1620, 2012.

[4] S. Lee, S. Koh, D. Yen, and H. Tang, "Perception gaps between IS academics and IS practitioners: An exploratory study," *Inf. Manage.*, vol. 40, no. 1, pp. 51–61, 2002.

[5] B. Kitchenham, D. Budgen, P. Brereton, and P. Woodall, "An investigation of software engineering curricula," *J. Syst. Softw.*, vol. 74, no. 3, pp. 325–335, 2005.

[6] Y. Kim, J. Hsu, and M. Stern, "An update on the IS/IT skills gap," *J. Inf. Syst. Educ.*, vol. 17, no. 4, p. 395, 2006.

[7] S. Surakka, "What subjects and skills are important for software developers?," *Commun. ACM.*, vol. 50, pp. 73–78, 2007.

[8] C. Lee and H. Han, "Analysis of skills requirement for entry-level programmer/analysts in Fortune 500 corporations," *J. Inf. Syst. Educ.*, vol. 19, no. 1, pp. 17–27, 2008.

[9] A. Mohan, D. Merle, C. Jackson, J. Lannin, and S. Nair, "Professional skills in the engineering curriculum," *IEEE Trans. Educ.*, vol. 53, no. 4, pp. 562–571, Nov. 2010.

[10] T. C. Lethbridge, J. Diaz-Herrera, R. LeBlanc, and J. B. Thompson, "Improving software practice through education: Challenges and future trends," in *Proc. IEEE FOSE*, Washington, DC, 2007, pp. 8–28.

[11] S. Reisman, "Higher education: who cares what the customer wants?," *IT Professional*, vol. 7, no. 6, pp. 62–64, Nov.–Dec. 2005.

[12] ISO/IEC/IEEE, Geneva, Switzerland, "ISO/IEC/IEEE 24765, 2010, 3.2758: Systems and software engineering—Vocabulary,".

[13] J. Holbrook, "Meeting challenges to sustainable development through science and technology education," *Sci. Educ. Int.*, vol. 20, no. 1, pp. 44–59, 2009.

[14] J. Holbrook, "Rethink science education," *Asia-Pacific Forum Sci. Learning Teaching*, vol. 4, no. 2, pp. 1–13, 2003.

[15] H. Saiedian, "Software engineering challenges of the "Net" generation," *J. Syst. Softw.*, vol. 82, no. 4, pp. 551–552, 2009.

[16] M. Klawe, "Refreshing the nerds," *Commun. ACM*, vol. 44, no. 7, pp. 67–68, 2001.

[17] L. Blum and C. Frieze, "In a more balanced computer science environment, similarity is the difference and computer science is the winner," *Comput. Res. News*, vol. 17, no. 3, p. 2, May 2005.

[18] J. Margolis and A. Fisher, *Unlocking the Clubhouse: Women in Computing*. Cambridge, MA, USA: MIT Press, 2002.

[19] H. Tillberg and J. Cohoon, "Attracting women to the CS major," *Frontiers, J. Women Studies*, vol. 26, no. 1, pp. 126–140, 2005.

[20] J. Shotick and P. R. Stephens, "Gender inequities of self-efficacy on task-specific computer applications in business," *J. Educ. Bus.*, vol. 81, no. 5, pp. 269–273, 2006.

[21] M. Shaw, J. Herbsleb, I. Ozkaya, and D. Root, "Deciding what to design: Closing a gap in software engineering education," in *Proc. ICSE*, 2005, pp. 28–58.

[22] M. Shaw, "Software engineering education: A roadmap," in *Proc. FOSE*, 2000, pp. 371–380.

[23] M. Guzdial *et al.*, "Future of Computing Education Summit," Arlington, VA, USA, Jun. 2009 [Online]. Available: http://www.acm.org/education/future-of-computing-education-summit/FoCES_web.pdf

[24] C. Schreiner and S. Sjøberg, "Sowing the seeds of ROSE. Background, rationale, questionnaire development and data collection for ROSE (The Relevance of Science Education)," *Acta Didactica*, vol. 4, pp. 1–80, 2004.

[25] P. L. Ackerman, M. E. Beier, and K. R. Bowen, "What we really know about our abilities and our knowledge," *Pers. Indiv. Differ.*, vol. 33, no. 4, pp. 587–605, Sep. 2002.

[26] D. Labaree, "Comments on Bulterman-Bos: The dysfunctional pursuit of relevance in education research," *Educ. Res.*, vol. 37, no. 7, pp. 421–423, 2008.

[27] C. Bishop-Clark, J. Courte, and E. Howard, "Programming in pairs with Alice to improve confidence, enjoyment, and achievement," *J. Educ. Comput. Res.*, vol. 34, no. 2, pp. 213–228, 2006.

[28] L. Werner, B. Hanks, and C. McDowell, "Pair programming helps female computer science students," *J. Educ. Resour. Comput.*, vol. 4, no. 1, Mar. 2004, art 4.

[29] C. McDowell, L. Werner, H. E. Bullock, and J. Fernald, "The impact of pair programming on student performance, perception and persistence," in *Proc. ICSE*, Portland, OR, USA, 2003, pp. 602–607.

Janet Liebenberg received the Master's degree in computer science education from the North-West University, Potchefstroom, South Africa, in 2010, and is currently pursuing the Ph.D. degree in computer science education at the North-West University.

In 2010, she became a Lecturer with the Computer Science (CS) and Information Systems (IS) Department, North-West University. Her research interests include the teaching and learning of programming and minorities in CS.

Magda Huisman received the Ph.D. degree in computer science (CS) and information systems (IS) from the Potchefstroom University for Christian Higher Education, Potchefstroom, South Africa, in 2001.

She is a Professor with the Department of CS and IS, North-West University, Potchefstroom, South Africa. She has published in journals such as *Management Information Systems Quarterly*, *Information & Management*, and the *International Journal of Web & Semantic Technology*. Her research interests include the use and effectiveness of systems development methodologies and the diffusion of information technologies.

Elsa Mentz received the Ph.D. degree in computer science education from the Potchefstroom University for Christian Higher Education, Potchefstroom, South Africa, in 2001.

She is a Professor with the Faculty of Education Sciences, North-West University, Potchefstroom, South Africa. Her research mainly focuses on cooperative learning in computer science (CS) education as well as the fostering of self-directed learning skills in the CS classroom.

Prof. Mentz is a member of AERA and ACM.

Chapter 4

Article 2

Industry's perception of the relevance of software development education

Mrs J. Liebenberg[a]

Prof. M. Huisman[a]

Prof. E. Mentz[b]

[a]**Department of Computer Science and Information Systems,**
Faculty of Natural Sciences,
North-West University,
Potchefstroom,
2520,
SOUTH AFRICA

[b]**Faculty of Education Sciences,**
North-West University,
Potchefstroom,
2520,
SOUTH AFRICA

Manuscript accepted for publication in: ***TD The Journal for Transdisciplinary Research in Southern Africa*****.**

Industry's perception of the relevance of software development education

J LIEBENBERG[1], M HUISMAN[2] AND E MENTZ[3]

ABSTRACT

It is widely acknowledged that there is a shortage of software developers, not only in South Africa, but also worldwide. Despite reports, in mostly quantitative studies, of a gap between the industry needs and software development education, the view the industry has of the new graduates and the problems, challenges and solutions in respect of software development education has not been explored in detail. This article reports on a mixed methods study of the relevance of software development education from the perspective of the industry. The analysis reveals some interesting views held by the industry, as well as by the different generations on their new recruits and the problems and challenges that are faced. The following solutions to the problems are suggested: teamwork; projects and experience; work-integrated-learning and mentoring; technical and soft skills; keeping up to date; career guidance; introspection; and generational awareness.

KEYWORDS

software development education, industry, computing curricula

1. INTRODUCTION

South Africa has a shortage of professionals with ICT skills (Harris, 2012) and this is a worldwide phenomenon (McAllister, 2012). The Career Junction Index (CJI), which monitors online employment trends in South Africa, found that the IT industry has the highest vacancy levels and that software development (SD) is at the forefront of the country's significantly high IT

[1] Corresponding author: Janet Liebenberg, Faculty of Natural Sciences, North-West University, Private Bag X6000, Potchefstroom, 2520, South Africa; *Tel*: +27-18-2992536, janet.liebenberg@nwu.ac.za

[2] Prof Magda Huisman, Faculty of Natural Sciences, North-West University, Private Bag X6000, Potchefstroom, 2520, South Africa, Tel: +27-18-2992537, magda.huisman@nwu.ac.za

[3] Prof Elsa Mentz, Faculty of Education Sciences, North-West University, Private Bag X6000, Potchefstroom, 2520, South Africa, *Tel*: +27-18-2994780, elsa.mentz@nwu.ac.za

demand (CareerJunction Index, 2014). Not only is there a shortage of SD workers, but students from computing disciplines are also graduating with a lack of the skills that companies are searching for (Bateman, 2013). Regarding ICT skills in South Africa: “not all graduates are prepared for the working environment, and don't always fit in. This is a gap that needs addressing” (Mawson, 2010:1).

The 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 on the other hand, the mission and 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. Reisman (2004) argues that the primary qualification of new employees should be their solid undergraduate education in which the curriculum focused on a common body of knowledge; a set of basic topics and principles. Employers can therefore build on that solid foundation and new employees can easily be trained by their employer for entry-level project work using particular (vendor) products. Several researchers have studied the knowledge and skills requirements for IT professionals through quantitative analysis but no mixed methods study could be found that studied the view the industry has of the new graduates and the problems and challenges connected to SD education. In addition, the solutions to the problems and challenges surrounding SD education have been discussed in papers with a view to the future but have not been researched in detail.

In view of the rapid pace at which technology is changing and in the light of the shortage of software developers, the present study investigated the relevance of software development education as it pertains to the industry. The research questions were:

- What is the industry’s view of its new graduates?
- Is there a gap between software development education and the workplace from the perspective of the industry?

- What are the problems and challenges surrounding software development education and what are the solutions to these problems and challenges?

The results of this study could help SD educators and curriculum developers to form a picture of the recent graduates from the perspective of the industry in order to relate the suggestions of the SD industry for relevant SD education to the academic preparation of future software developers. Furthermore, the results can give an indication of the responsibility and contribution of the software industry in respect of the education of new recruits. The SD students, especially those nearing the end of their studies, could use the results of this study to prepare for the expectations of their future employers. It can therefore result in more relevant software development education with regard to the industry and it might contribute to meeting the demand for software developers.

1.1 Terminology

The ACM, IEEE Computer Society and AIS have joined forces to regularly create and update curriculum guidelines for undergraduate degree programmes in the following computing disciplines - computer science, computer engineering, information systems, information technology and software engineering. However, the dynamic nature 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 necessary to explain and clarify certain key terms:

- *Software development (SD)* - the ISO (International Organization for Standardization) and the IEC (International Electrotechnical Commission) define *developer* as an individual or organisation that performs development activities (including requirements analysis, design, testing through acceptance) during the system or software life cycle process (ISO/IEC, 25000, 2014). For the purpose of this study *Software development* will refer to the process of developing software through successive phases in an orderly way.

Industry – the ISO and IEC define *software manufacturer* as a group of people or organisation that develops software, typically for distribution and use by other people or organisations (ISO/IEC,

19770-2, 2009). For the purpose of this study, *Industry* will refer to software manufacturers, as well as organisations where software is not the organisation's main product but software is developed for use within the organisation.

2. LITERATURE REVIEW

In this section, the background and literature regarding the new graduate in the workplace, the education new graduates received and the challenges will be discussed.

2.1 The new graduate in the workplace

Most new graduates belong to the so-called Net generation, also known as the Millennial Generation or Generation Y. The Net generation (especially people born in the US and Canada from the early 1980s to the late 1990s) is characterised by people who may have never known life without the Internet (Oblinger and Oblinger, 2005). Their early and omnipresent exposure to technology has defined their styles, their modes of communication, their learning preferences, their social choices and their entertainment preferences (Saiedian, 2009).

In the workplace, Millennials want flexible work schedules and they do not like traditional office rules and hierarchies. They want continuous performance feedback and career advice from managers and they think that managers could learn from their young employees. What could be misinterpreted as the "self-importance" attitude of Millennials is actually an optimistic sense of having many new ideas and a desire to not only contribute their ideas, but also to have their skills utilised by managers. Millennials want to wear jeans to work and especially in IT companies, the norm is to wear casual clothes, for example jeans, sneakers, flip-flops and sweatshirts – (Facebook CEO Mark Zuckerberg's famous hoodie and former Apple CEO Steve Jobs' black turtleneck with jeans are good examples) (Schawbel, 2012). The older generations, on the other hand, are more prone to believing in the importance of maintaining a standard professional look in the workplace. It seems as if Millennials also prefer casual attire because they do not separate their personal and professional lives in the same way that the older generations do.

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 and the latter group of students exhibit a high degree of technological diversity. They may have information literacy characteristics and IT skills quite different from the typical Net generation. Higher education comprises a highly diverse and growing student body possessing a core set of technology based skills, but beyond this core there is a diverse range of skills across the student population (Lorenzo *et al.*, 2007; Jones *et al.*, 2010; Kennedy *et al.*, 2008).

Much is written about the so-called "generational gap". Young professionals or Millennials are joining Generation X (born 1960-1980) and Boomers (born 1940-1960) in modern organisations. The three generations are inherently different - they approach work, work/life balance, accountability, delegation, loyalty, authority, motivation and reward systems differently (Macon and Artley, 2009). However, Augustine (2001) states that a popular misconception exists that the workforce consists of generational tribes engaged in rivalries and conflicts with each other. Augustine (2001) admits that there might be differences, misunderstandings and tensions among workers from different generations, but the division and conflict are often exaggerated.

Another misconception is that Millennials want to change jobs frequently (Murray, 2015). Millennials actually value job security more highly than previous generations and they also value job enjoyment and are inclined to leave a job they do not enjoy. There is also a misconception that money does not matter to Millennials. A high-paying job is near the bottom of their list of work priorities - but the same applies to other generations, in nearly equal numbers (Murray, 2015).

In their study Soni *et al.* (2011) determined the differences in work commitment of software professionals from the X-generation and Y-generation (Millennials). The study examined generational differences for the five types of work commitment - work involvement, job involvement, work group commitment, organisational commitment and professional commitment. Organisational commitment and professional commitment in turn had three components each - affective commitment, continuance commitment and normative commitment. Soni *et al.* (2011)

found that the two generations differed significantly on only three of the nine factors examined. Continuance commitment to the profession is significantly higher for Generation X than for Generation Y, meaning that Generation X feels obliged to stay in the software profession. The Generation Y group of employees has higher job involvement and their continuance commitment to the organisation is significantly higher than that of the Generation X group.

In higher education a popular notion exists to describe students as ‘digital natives’ and lecturers as ‘digital immigrants’. Students’ immersion in digital technologies creates in them a radically different approach to learning and it is expected from lecturers to adapt their methods to students’ new way of learning in order to remain relevant. However, Bayne and Ross (2011) call for a more cautiously critical and nuanced understanding of the effects of new technologies on the practices and subject positions of learners and teachers in higher education.

2.2 The education of new graduates

According to Spicer (2011), any business needs from its incoming recruits “Critical Cross- field Outcomes” - the skills and abilities that the South African Qualifications Authority (SAQA, 2000) requires to be achieved in all their registered qualifications. The SAQA (2000) lists seven outcomes, namely problem solving and creative thinking; being able to work in a team; the ability to manage oneself and one’s activities, being able to critically evaluate information; good written and verbal communication; an effective use of science and technology; and being able to demonstrate an understanding of the world as a set of related systems by recognising that problem-solving contexts do not exist in isolation. These outcomes describe what can also be called soft skills or non-technical skills.

Several studies suggest a gap between the knowledge and skills demanded by the industry and the knowledge and skills gained by graduates of university computing courses. Lethbridge (2000) analysed the relevance and depth of specific core topics that software professionals had received as part of their university education and identified a significant mismatch between software

education and industry in terms of the knowledge needed by software engineers to meet the industry's requirements. Other researchers reported similar gaps (Kitchenham *et al.*, 2005; Kim *et al.*, 2006; Surakka, 2007; Lee and Han, 2008; Aasheim *et al.*, 2009; Moreno *et al.*, 2012) and for Lethbridge *et al.* (2007) filling them is one of the most critical challenges for SD educators.

When Information Systems curricula were analysed from the perspective of the industry the following gaps in knowledge and skills were identified: problem-solving and project management skills, knowledge of business, IT business consultancy, security, end-user computing, soft skills related to core knowledge, knowledge related to leadership and negotiation or giving presentations (Lee and Han, 2008; Kim *et al.*, 2006; Moreno *et al.*, 2012). Aasheim *et al.* (2009) compared the perceptions academics have of the importance of various skills for entry-level IT workers with the view that IT managers have. They found that IT managers place more importance on issues related to hardware concepts, operating systems, leadership skills or entrepreneurial traits than academia. However, both groups ranked broader categories of skills - interpersonal skills, personal skills, technical skills, organisational skills and work experience - in the same order of importance.

The research of Bullen *et al.* (2009) examined workforce trends in IT-provider companies and encountered problems in the following areas: graduates who are not trained in areas that the marketplace is seeking; thin pipeline for specific technical skills; increasing pressure to source IT capability; and lag in university responsiveness to the needs of the marketplace.

To effectively fill 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 correctly tackle the problems that they will face during their professional career (Loftus *et al.*, 2011).

Industry's view on the relevance of SD education is determined not only by the knowledge and skills (technical, non-technical and business) acquired by graduates at university but also by their

awareness of the proven techniques, challenges and expectations of their roles in employment. Gallagher *et al.* (2010) investigated the skills considered critical when hiring entry-level employees and report that while both technical and non-technical skills are important, the skills most critical are non-technical skills, such as project management, business-domain knowledge and relationship skills. IS professionals' competencies change as they gain experience in the workplace and they are required to have higher levels of technical skill in early stages and higher levels of non-technical skill in the later stages of their careers (Lee *et al.*, 2001). Plice and Reinig (2007) found that emphasising technical topics at the expense of business content may provide short-term benefits in transitioning to the workforce, but it might inhibit career advancement as graduates assume greater managerial responsibilities. Emphasising communication and teamwork skills while maintaining the existing curriculum balance between business and technical content is indicated as an appropriate strategy to align the IS curriculum with the needs of industry.

Hassan (2008) warns that as a result of the increasing breadth of content introduced into the IS field, undergraduates may become multidisciplinary illiterates. The multidisciplinary nature of the content produces graduates as generalists instead of specialists. Hassan (2008) further warns that without a clear body of knowledge built within a succinct codified framework for students, the IS field will continue to struggle in recruiting new members into the field.

Software and technical developments have made remarkable strides in the last few decades. Not only is the dependence on software increasing, but the character of software production itself is changing – and with it the demands on software developers (Shaw *et al.*, 2005; Shaw, 2000; Saiedian, 2009). The diversity of software applications requires adaptability in responding to client needs and the diversity of clients and contexts requires the ability to discriminate between criteria for success. This presents new challenges for the education of software developers (Shaw *et al.*, 2005; Shaw, 2000).

2.3 Challenges

The challenges facing SD education have not been researched in empirical studies in detail but the British Computer Society (McGettrick *et al.*, 2004), Shaw (2000), Lethbridge *et al.* (2007) and Mead (2009) reported on some of the challenges and aspirations they foresee for computing education and software engineering education.

The perception of computing. Computing departments experienced a decline in enrolment that has occurred since 2000, with bright minds being attracted to other disciplines. Low enrolments result in low numbers of post-graduate students, which in turn will restrict the capacity for research in the field. A positive public image of computing must be promoted whereby good students are attracted and the public gains respect for the field and the professionals who practise within it (McGettrick *et al.*, 2004; Lethbridge *et al.*, 2007).

The dimensions of the field. There are five different undergraduate degree programmes in computing covering computer science, information systems, software engineering, computer engineering and information technology. Knowledge about specialties in terms of how practitioners work, the distinct roles in software development and what their different educational needs are, is needed. This will allow for better-tailored programmes, simpler models of computing and a smaller core resulting in the provision of appropriate education for each student (Lethbridge *et al.*, 2007; Shaw, 2000; McGettrick *et al.*, 2004).

Real-world industrial practices. A large collective effort should be launched to establish two-way knowledge and skill exchanges with companies from more industrial sectors and with more professionals within these companies in order to communicate these real-world industrial practices more effectively to students (Lethbridge *et al.*, 2007; Mead, 2009).

Evidence based education. Computing education should ensure that everything in the design and delivery of SD education must become increasingly based on carefully-evaluated evidence from industrial practice. Computing education models must be examined with workforce needs in mind (Lethbridge *et al.*, 2007).

Educator background. Just as students and curricula have to stay updated in the rapidly changing world of technology, similarly the educators should not stay behind. Educational institutions should ensure that educators have the necessary background (Lethbridge *et al.*, 2007).

Curriculum standards that are forward-looking. In an evolving field such as SD, the challenge is a fast response to changes in technology and designing curriculum guidelines for the future, while supporting current practice (Lethbridge *et al.*, 2007; Shaw, 2000).

Education for existing practitioners. In addition to educating new students, the knowledge levels of the existing workforce need to be raised by providing an infrastructure to provide support and guidance on a career-long basis for practitioners (Lethbridge *et al.*, 2007; Shaw, 2000; McGettrick *et al.*, 2004).

Information technology in the classroom. Educators in the software field can do better at exploiting technology to support the learning process itself (Shaw, 2000).

Educational research in computing. The prestige and quality of educational research in computing need to be raised (Lethbridge *et al.*, 2007). Educational research is often not recognised as "legitimate" technical research and Mead (2009) challenges educators to continue to push for recognition of educational research as a legitimate field of research.

Variety of delivery mechanisms. Educators need to adjust from the traditional lecture format because distance delivery via a variety of mechanisms is here to stay (Mead, 2009; Shaw, 2000; McGettrick *et al.*, 2004).

The literature was explored regarding new graduates, the possible gap between software development education and the workplace and the anticipated challenges faced with new graduates' education. As these topics were not explored in detail, this study therefore investigates the relevance of software development education from the point of view of the industry.

3. RESEARCH METHOD

In this section the terminology will first be explained and then the research design, the demographics of the participants, as well as the data collection and analysis are discussed.

3.1 Research design and participants

A mixed methods approach was used to conduct the research. Tashakkori and Creswell (2007:4) describe mixed methods as: "Research in which the investigator collects and analyses data, integrates the findings, and draws inferences using both qualitative and quantitative approaches or methods in a single study or programme of inquiry". Mixed methods research can aid in the development of rich insights into phenomena of interest that cannot be fully understood using only a quantitative or a qualitative method. Mixed methods research will often provide the most balanced, informative, complete 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; (2) embedded; (3) explanatory; and (4) exploratory. In this study the type of mixed methods research was explanatory, as 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 method study was conducted in South Africa and for the quantitative part of the study a survey was used and the qualitative data was acquired through the comments by survey respondents.

In 2013's last quarter 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 requested to complete the anonymous online survey. Some of the respondents indicated that they 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 totalled 214, which indicates a response rate of around 21%.

Table 1 provides a summary of the biographic data. The gender profile is a concern but not surprising with only 8% of the respondents being female and the results obtained from the females

can therefore not be generalised. The age profile indicates that 42% of the respondents are young, Generation Y software developers, 57% are Generation X software developers and a mere 1% are Boomers. In terms of the respondents' education 49% of them are in possession of an IT degree or degrees, with another 22.5% having related degrees. It is not uncommon in software development to find people with few qualifications (4.5% of the respondents). There are software developers who taught themselves to program and they provide proof of their knowledge, skills and experience to employers in the software industry through their work. The level of the education of the respondents was quite high with 38.5% of them possessing a post-graduate qualification and even one with a PhD. The work experience of respondents indicates that 70.5% of them have more than five years' work experience and 46% of them are involved in the hiring of new graduates. A common trend is for software developers to not stay long in one position or workplace and this trend also surfaced in this study with only 43% of the respondents working more than two years at their current employer.

Table 1: 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
Education	Matric	10 (4.5%)
	Certification	22 (10%)
	National Diploma	30 (14%)
	IT degree(s)	104 (49%)
	BSc/BCom	38 (18%)
	Engineering degree	10 (4.5%)
Level of education	Matric	10 (4.5%)
	Certification/Diploma	52 (24%)

	B-degree	70 (33%)
	Hons	54 (25%)
	Masters	28 (13%)
	PhD	1 (0.5%)
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%)

3.2 Data collection, instrument and analysis

An initial list of questions was developed by writing both new items and adapting items from available surveys, such as ROSE (Schreiner and Sjøberg, 2004) and the SAQA's list of "Critical Cross-field Outcomes". Once the initial questions were generated, they were sent to industrial and academic experts to refine the instrument. Feedback from this pilot study served as the basis for correcting, refining and enhancing the questions and it resulted in a questionnaire with a pool of 30 items. The first section of the questionnaire gathered information on the biographic data of the respondents as shown in Table 1. The questionnaire was further divided into two domains. The first domain, "Educational background of new recruits", had 16 items and enquired on what respondents view as important in SD classes, such as industry experience of lecturers. The second domain, "Career", gathered data on SD careers, such as what is expected from a good software

developer. The first and second domains were accompanied by a five-point Likert response scale from 1 (Strongly disagree) to 5 (Strongly agree).

Factor analysis was used to investigate the 30 items in more detail to reduce the variables into a smaller number of factors. The 214 responses were examined using principal components factor analysis as the extraction technique and Oblimin with Kaiser Normalization as the rotation method. After analysis of the results of the factor analysis, the researchers felt that some items were of lesser importance in answering the research questions, which resulted in five items being omitted. Of the remaining 25 attitude items, 10 items were being handled as single research variables and the remaining 15 items yielded four interpretable factors. Factors were named according to their main context. A Cronbach's α coefficient was calculated for each of the four factors and was found, as Table 2 shows, to be reliable ($\alpha \geq 0.60$).

Qualitative data was gathered by means of an open-ended question at the end of the questionnaire, asking for further comments on the education of software developers and 77 respondents added comments. Furthermore, there were 21 respondents who felt so strongly about the topic that they gave up their anonymity and e-mailed the researchers with more comments and suggestions.

Table 2: Factors* (with reliability coefficients) and items

Factor	**Cronbach's alpha (α)**
Critical outcomes required in software development modules.	0.718
Positive attitude towards colleagues/management.	0.793
Positive attitude towards tasks/work.	0.666
Emotional/social skills required.	0.700
Item	
Neat and tidy appearance required.	
Good set of exam results required.	
New recruits have the knowledge and skills to immediately contribute to the development of software.	
Students know what software developers do in the workplace.	
New recruits have to be trained/sent for training before they can contribute to the development of software.	
Our country needs more software developers.	

People from industry should be brought into software development classes.
Lecturers should have industry experience.
In the software development modules students should learn to use science and technology effectively.
Projects play an important role in the education of students.

** See Appendix A for the items in each factor*

Basic analysis of quantitative data was done by calculating the mean values and standard deviation of each of the 14 variables. Various statistical tests were used in the analysis to match the metric being analysed. Three groupings were identified based on gender, age and education. All the groupings were tested for significant differences between means in the different variables using T-tests and ANOVA tests. Chi-square tests and Spearman's rank correlation analysis were also used to analyse relationships between the groupings and the variables. When the results of the interaction analysis are reported, only the significant interactions or primary effects will typically be discussed. Since a convenience sample instead of a random sample was used, the p-values will be reported for the sake of completeness but will not be interpreted (Steyn *et al.*, 1999).

For the analysis of the qualitative data the ATLAS.ti 7.1.4 computer program was used and the data was analysed as follows:

- The data of the open-ended question in the online survey and the data from the e-mails were assigned to a single hermeneutic unit;
- The relevant information was separated from the irrelevant information;
- The relevant information was broken into a number of text segments;
- The emerging themes were coded in a process of inductive categorisation and the text segments were linked to the coded themes;
- The codes were grouped into code families;
- Networks were drawn that reflect the meaning of the respondents' views; and
- The networks were used to develop an overall description of SD professionals' views of SD education.

The code families and themes that emerged from the analysis were:

- New graduates: adaptation to the workplace; weaknesses; new graduates vs. unqualified recruits; passion; thirst for knowledge.
- The possible gap: true; false.
- The problems and challenges: who is to blame?; lacking skills and knowledge; rapid change.
- The solutions: teamwork; projects; experience; work-integrated-learning (WIL); mentoring; tips for teaching.

Since the product of qualitative research is richly descriptive (Merriam, 2009), the participant comments were analysed, interpreted and presented in the form of quotes.

4. RESULTS AND DISCUSSION

In this section, important data for each of the concepts are considered, as well as the qualitative data that provides a rich description of the information obtained.

4.1 The new graduate

From Table 3 it is clear that the respondents felt very strongly that new recruits must have a positive attitude towards colleagues and management, as well as a positive attitude towards tasks and work. Respondents also felt quite strongly about the emotional and social skills expected from new recruits when they enter the workplace. The respondents did not feel that a neat and tidy appearance is such an important requirement and respondents had a relatively neutral feeling about the standard of the examination results of students but the deviation was quite large.

Table 3: Basic analysis of new graduate factors and items

(I)tem / (F)actor	Mean*	Standard deviation
Positive attitude towards colleagues/management (F)	4.44	0.562
Positive attitude towards tasks/work (F)	4.55	0.458
Emotional/social skills required (F)	3.75	0.720
Neat and tidy appearance required (I)	2.79	1.111
Good set of examination results required (I)	3.03	1.041

* *Likert-style responses were ranked from 1 to 5 respectively*

Gender differences were analysed with a T-Test and the female respondents felt more strongly ($d = 0.47$, $p<0.05$) about a neat and tidy appearance required in the workplace. In addition, the T-Test showed that the female respondents felt more strongly ($d = 0.48$, $p<0.05$) that good examination results are important.

Generational differences were analysed with an ANOVA and the results in Table 4 indicate that the older software developers (Generation X) differed from the young software developers (Generation Y) in terms of new recruits' appearance. Surprisingly enough, it was the older generations that felt that a neat and tidy appearance is not such an important requirement in the workplace. This finding contradicts the view of Schawbel (2012) that the older generations believe that it is important to maintain a standard professional look in the workplace.

There is a medium practically significant difference between the 18 to 24 age group and the software developers over 40 in terms of their views on academic performance. The young software developers view a good set of examination results as significantly more important than the older respondents. This might be explained by the experience of Generation X where they had seen that it is not necessarily the academic giants of university that turned out to be the best and most innovative employees.

In terms of emotional and social skills required, Table 4 shows that there is also a medium practically significant difference between the 18 to 24 age group and the software developers over

40. The fact that the Generation X software developers view emotional and social skills as significantly less important than the young respondents might be explained by one of the three items in the factor, namely: "*To be a good software developer you have to have modern leadership skills, such as self-confidence and a preparedness to lead by example"*. Schawbel (2012) pointed out that Millennials' attitude could be misinterpreted as "self-importance" by the older generations and in the qualitative data this problem with the new recruits' attitude emerged as well.

Table 4: Differences between the age groups

(F)actor / (I)tem	Age groups	Effect size	p
Neat and tidy appearance required (I)	18-24 *vs* 25-29	0.53*	0.078
	18-24 *vs* 30-39	0.48	
	18-24 *vs* 40+	0.62*	
Good set of examination results required (I)	18-24 *vs* 25-29	0.41	0.114
	18-24 *vs* 30-39	0.23	
	18-24 *vs* 40+	0.57*	
Emotional/social skills required (F)	18-24 *vs* 25-29	0.25	0.061
	18-24 *vs* 30-39	0.46	
	18-24 *vs* 40+	0.54*	

* *Medium practically significant difference*

A Spearman rank correlation analysis found a medium practically significant relationship ($r = .416$, $p<0.001$) between the respondents' views of a developer's appearance and his/her examination results. Therefore, respondents who viewed a neat and tidy appearance to be important, also felt strongly about good examination results for software developer students.

The qualitative data revealed the following regarding the new graduates:

Adaptation to the workplace

New recruits adapt quickly to the workplace and they learn fast.

- *I have several varsity graduates with BSc Comp Sc qualifications who graduated in the last 2 to 3 years working for me and all of them are very intelligent young people who learn fast and have proven to be valuable members of my team. Having said that, however, that was NOT the case when they first arrived.*

- *Devs need the ability to learn quickly and adapt to constraints.*

Weaknesses

Respondents pointed out the weaknesses of new recruits. A great number of respondents pointed out that new recruits have theoretical but not applied knowledge and they lack problem-solving skills.

- *Lots of graduates/junior developers have excellent theory but have absolutely no clue how to implement it.*
- *Graduates seem to lack the analytical, logical and problem-solving abilities required*

Respondents felt that new recruits arrive in the workplace with a laid-back attitude and are emotionally immature. Respondents experience a lack of respect from the new recruits and a lot of attitude. This finding can probably be explained by the “generational gap” because as Schawbel (2012) pointed out, Millennials’ attitude could be misinterpreted as "self-importance", but Millennials see no reason for a strict hierarchy and most Millennials want to contribute, believing that everyone’s ideas should be heard.

- *They have a YOLO (You Only Live Once) attitude - I guess we were all young and impetuous once, but we need emotionally mature and stable professionals.*
- *Starters seem to have a lot of attitude and little respect. This makes it difficult to make them productive members of a team.*

Another weakness from the perspective of the software developers is that new recruits apply for and end up in a career that does not match their personality type.

- *Wrong career choice for their personality. Software developers are usually DISC "C" personalities (INTx in Myers Briggs). Project Managers and Architects are usually D/C - i.e. "Left Brainers". Business Analysts need more people / emotional skills.*

New graduates vs. unqualified recruits

The respondents, however, felt that new recruits without a degree had more weaknesses that probably cannot be rectified.

- *Unfortunately for the most time I have noticed a lack of architectural growth from degreeless developers in the long term, 3 years+. Their work often shows signs of dirty coding and lack of interface design skills. Lack of following standards and doing things the best way is often a sign of poor background. They also seem to project a type of unwarranted arrogance.*

Passion

Respondents commented that the best graduates are passionate about software development and the employers find that passion is lacking in some of the new recruits.

- *The best graduates (and thus the hardest to get) are the ones who unquestionably speak with an inbuilt passion for development.*
- *What ultimately stands out for me when I hire new people is their passion for what they do and hunger to learn more and constantly improve, as well as high quality work.*

Thirst for knowledge

In view of the rapid pace at which technology is changing, the respondents emphasised the fact that new recruits must have a desire to try out and learn new things.

- *Hunger for knowledge and the work ethic to see it through are essential - I'm probably learning more every day now after 30 years than I did at university.*
- *Software developers require a natural thirst to solve problems through their skills, as well as a natural want to learn new things.*
- *Hard work, thirst for knowledge and learn as many technologies as possible (be dynamic not conservative).*

4.2 The possible gap between software development education and the workplace

Table 5 shows that the respondents felt quite strongly about the shortage of software developers in South Africa and they strongly agreed that further training is required for new recruits. On the other hand, the mean values of 2 variables are relatively low with the respondents feeling that new

recruits cannot immediately contribute to the development of software and that students do not have a clear idea of what software developers do in the workplace.

Table 5: Basic analysis of gap items

Item	Mean*	Standard deviation
Our country needs more software developers.	4.32	0.890
New recruits have to be trained/sent for training before they can contribute to the development of software.	4.01	0.836
New recruits have the knowledge and skills to immediately contribute to the development of software.	2.35	1.063
Students know what software developers do in the workplace.	2.70	1.037

* *Likert-style responses were ranked from 1 to 5 respectively*

Differences based on work experience were analysed with a T-Test and a medium practically significant difference ($d = 0.50$, $p<0.001$) between the respondents with less than 10 years' experience and the rest of the respondents was found, in terms of the new recruits' knowledge and skills, to immediately contribute to the development of software. The respondents with less than 10 years of work experience had a significantly stronger opinion that new recruits can immediately contribute, whereas the more experienced software developers felt that new recruits do not have the knowledge and skills to immediately contribute to software development.

From the above it is not surprising that the results of an ANOVA in Table 6 indicates that the older the software developers were, the more they differed with the young software developers in terms of the knowledge and skills of new recruits. The older respondents felt that new recruits lack the knowledge and skills to immediately contribute to the development of software.

Table 6: Differences between the age groups

Item	Age groups	Effect size	P
New recruits have the knowledge and skills to immediately contribute to the development of software.	18-24 *vs* 25-29	0.43	<0.05
	18-24 *vs* 30-39	0.61*	
	18-24 *vs* 40+	0.77*	

* *Medium practically significant difference*

Furthermore, correlation techniques were used to analyse the correlation between the variables. A Spearman rank correlation analysis found a medium practically significant relationship (r =

0.376, p<0.001) between the respondents' views of the new recruits' knowledge and skills entering the workplace and students' knowledge of what software developers do in the workplace.

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 enrich understanding of a phenomenon but also open new avenues for future research (Venkatesh *et al.*, 2013). In the reported quantitative results above it showed that the respondents felt that there is a gap between SD education and industry needs and some of the qualitative results re-affirmed that notion. However, the comments of some respondents contradicted the idea of a gap between industry needs and SD education.

True - there is a gap

When looking at the qualitative data, some respondents felt strongly that there is a gap between SD education and industry needs. Words, such as "*huge gap*", "*big gap*", "*definitely a gap*", "*significant gap*", "*massive gap*" were used.

- *In my experience there is a massive gap between university grad knowledge and real-world industry needs.*
- *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.*
- *A piece of paper obtained by a student in IT in no way proves the skills and ability of that student. A huge misconception. It's not a good enough test of a good developer. Most of my best devs haven't finished their degrees...*
- *Industry vs. Training doesn't weigh up!*

Some respondents spoke of shock and horror at the graduates' knowledge and skills.

- *We brought on 3 graduates on a 3-month internship. I was shocked at how poor their actual skills were. Some of them knew a lot of theory but no practical application.*
- *Last year we brought on 3 graduates for an internship. I was horrified at how little they knew*

of real software development.

False – there is not really a gap

However, not all respondents felt negative about the knowledge and skills of graduates and they think that graduates have a broad basic knowledge, that it is not difficult to train them and they are a great asset in the long term.

- *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.*
- *Training them to be productive real-world developers has never been a particularly difficult or time-consuming task and in general, if they have earned a BSc Comp Sc, it does tend to indicate that they are of a higher calibre than most candidates and will prove to be a better hire in the long run.*
- *I have noticed that university gives developers a foundation of broad computer science knowledge, that without a degree very few individuals have. A degree also shows commitment to something that will take a while to complete and shows a character that is often necessary in long-term self-development of the individual.*

From the above it is clear that the respondents had opposing views regarding the gap between the new graduates' education and their workplace. These opposing views most probably originate from the respondents' views on the mission of the university. If they view the mission as educational as Reisman (2004) also pointed out, then they do not think there is a big gap, but if they view the university's mission as one of training graduates to immediately be productive software developers, then they experience a big gap.

4.3 The challenges in software development education

Analysis of the qualitative data highlighted a number of challenges and problems that the software developers in the industry foresee. "*We have a big problem in South Africa when it comes*

to software writing skills”.

Who is to blame?

Some respondents laid the blame for the problems with new software developers at the door of the university.

- *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.*

Other respondents blamed the software industry for the problems with new software developers.

- *No problem with the education. What falls flat is what happens with IT companies after graduation.*
- *There is a shortage of developers, but companies do not want to invest in graduates as they are afraid they will leave for better salary, etc. However, those same companies go to India, etc. for developers and further add to the shortage of developers. All a vicious circle, sadly.*

Lacking skills and knowledge

Respondents listed a number of skills that are lacking: “*no basic troubleshooting and investigation skills*”; “*No working experience at all*”; “*a lot of theory but no practical application*”; “*No teamwork experience*”; “*Too informal processes*”; and “*they have no concept of how real projects are managed, how projects are planned and timings are estimated”*.

One of the respondents echoed the claims of Gallagher *et al.* (2010), Lee and Han (2008) and Plice and Reinig (2007), namely that the main problem is that new recruits lack business knowledge. “*I think it very much depends on the industry you go into. In my personal experience - I have gone into investment banking (still working in IT though). I felt a big gap when I entered the working world purely because I wasn't prepared from a business knowledge perspective.”*

Rapid change

Respondents emphasised the continuing and rapid changes in the software industry and that the knowledge the graduates acquired at university is often outdated. Some respondents felt that

students should rather learn foundational principles and theories at university and others felt that university courses should regularly be updated.

- *The world is moving ever faster - what worked over the last 10 years won't necessarily work in the future*
- *...they also tend to do things in ways that are now considered old-fashioned and outdated and often had to be taught to do things differently to what they had learnt at university.*
- *Practical software development skills change rapidly so tertiary institutions should focus on general theory*
- *According to quickly changing software industry, varsity courses need to be revised every 3-4 years - to keep knowledge up to date.*

The words of a software development manager effectively summarises the above opinions: "*The general take away for me, when dealing with varsity graduates is not that they are being taught badly or incorrectly at varsity, but rather that what they are being taught always seems to be slightly out of date and behind current trends and technologies. 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. The market has a severe lack of solid theoretical computer science training.*"

4.4 The solutions to problems in software development education

All the variables listed in Table 7 have high means indicating that software developers viewed these variables as important solutions to the problems in educating software developers. Respondents felt that experts from industry should be used to teach students and that the lecturers should in turn gain experience in the workplace. Respondents also strongly agreed that the education of software developers requires what can be referred to as non-technical skills which in South Africa are known as critical outcomes.

Table 7: Basic analysis of solutions

(I)tem / (F)actor	Mean*	Standard deviation
People from industry should be brought into software development classes (I)	4.37	0.769
Lecturers should have industry experience (I)	4.43	0.752
In the software development modules students should learn to use science and technology effectively (I)	4.20	0.707
Projects play an important role in the education of students (I)	4.32	0.734
Critical outcomes required in software development modules (F)	4.40	0.488

** Likert-style responses were ranked from 1 to 5 respectively*

The qualitative data revealed some interesting and useful suggestions as solutions to the problems in educating software developers.

Teamwork

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.*

The above comments confirm the challenges Lethbridge *et al.* (2007) and Mead (2009) anticipated.

Projects

Respondents felt that students should receive more challenging practical assignments and projects during their years of study.

- *More practicals! Lots of graduates/junior developers have excellent theory but have absolutely no clue how to implement it.*
- *Practical SDLC experience in a team setting (systems development projects) is a crucial part of the learning process.*
- *They need to do more stressed-for-time projects. The projects must not be just about stuff they*

do know. It must challenge them.

Experience

Respondents stressed the fact that the software industry is looking for people with experience and not necessarily a degree - *"someone with a degree AND experience is the holy grail of HR managers"* - and a great deal of emphasis is placed on practical and real-life experience.

- *Give them the practical experience they need and projects related to current real-life environments*
- *Universities should bridge this gap by integrating more experience into the curriculum.*

Work-integrated-learning (WIL)

Several respondents suggested that work-integrated-learning in the form of a vacation job or internship could be a solution for effective software development education.

- *It should work the same as an engineering degree where the student needs a year of actually working for someone before the degree is awarded.*
- *Software Developers should have vacation jobs in relevant companies where they can get corporate experience in development.*
- *Important to reward future software devs if they do well - a holiday internship at a dev house would be the best reward.*

Mentoring

The education of software developers is not only the duty of the university, but the industry can play its part through a mentorship programme.

- *Graduates need to be mentored and coached in their first year of employment. This is lacking extensively!*

Tips for teaching

Respondents commented on what lecturers should focus on in the education of software developers, including soft skills and technical skills. Respondents confirmed the aspirations of Lethbridge *et al.* (2007) and Shaw (2000) that an important factor is for the university to keep up

to date within the rapid changing technology environment.

- *The education of software developers should emphasise soft skills as much as technical skills.*
- *Lecturers should be able to teach more up to date skills to students, and open their eyes to possible exposures to methodologies.*
- *My kids will MOST LIKELY also work in this industry and I hope that they will be taught the correct languages to make them appropriately skilled for the job market.*

One respondent pointed to the fact that software developers rarely start from scratch when writing a program and students should therefore be taught to utilise all the resources at their disposal.

- *...show them how to troubleshoot a problem properly and leverage the internet, Google, online resources, etc. properly to their advantage.*

A respondent wraps up some of the sentiments unveiled above: *"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 realise that it 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."*

5. CONCLUSION AND RECOMMENDATIONS

The first research question in this study was: What is the industry's view of its new graduates? The software industry views its new recruits as fast adapters to the workplace who learn fast while several new recruits have theoretical but not applied knowledge. The SD industry views emotional and social skills as important, although some new recruits are found to be emotionally immature and disrespectful. New recruits are observed as having a "self-importance" attitude, especially in the case of Generation X. The software developers also experience that new recruits enter the workplace with their personality type not matching their career choice. However, most new graduates have a passion and thirst for knowledge, whereas unqualified recruits have weaknesses that probably cannot be rectified. New recruits' positive attitude towards colleagues and

management, as well as towards tasks and work is important as far as the industry is concerned. New recruits' appearance is not perceived as too important, especially by Generation X, but women view a neat and tidy appearance as more important than men. Software developers feel relatively neutral about students' exam results, but women and Millennials view good examination results as important.

The second research question posed was: Is there a gap between software development education and the workplace from the perspective of the industry? Some people in the SD industry feel strongly that there is a gap between SD education and industry needs and that new recruits in the SD workplace are relatively unprepared to immediately start to work on projects, resulting in a need for training before they can make a constructive contribution. Some recruits do not know what the work of a software developer entails until they enter the workplace. Generation X software developers do not have confidence in the abilities of new recruits. However, some people in the SD industry think that graduates have a broad basic knowledge, that it is not difficult to train them and that they are a great asset in the long term. These opposite views can probably be explained by people's views on the mission of universities as either an educational institution or a vocational training institute.

The third and last research question was: What are the problems and challenges surrounding software development education and what are the solutions to these problems and challenges? The software industry confirms that it is challenged by a shortage of software developers. The university, as well as the software industry are to be blamed for the problems and challenges surrounding software development education. The problems and challenges identified from the perspective of the SD industry are the following. Graduates often lack business knowledge and they lack experience in teamwork, troubleshooting, investigation skills and general practical experience on real-life projects. The rapid pace at which technology is changing often causes the knowledge graduates acquire at university to be outdated.

The following solutions to the problems and challenges are recommended in order to offer relevant software development education as far as the industry is concerned:

Teamwork. The university and the IT industry should work together in creating up-to-date curricula. People from industry should be brought into software development classes and lecturers should acquire industry experience.

Projects and experience. Practical assignments and projects play an important role in the education of students and real-life and practical experience must be included in students' education.

Work-integrated-learning and mentoring. The software industry can be helpful in accommodating students to gain experience in the workplace during their studies and mentoring them once they have entered the workplace.

Technical and soft skills. Universities must examine their curricula to ensure that not only technical, but also soft skills are included.

Up to date. Universities must attempt to keep the curriculum abreast of the rapid change in technology. The challenge is a fast response to changes in technology and designing curriculum guidelines for the future, while supporting current practice.

Career guidance. Universities, in conjunction with the industry, should provide career guidance to students, so that they do not apply for and end up in a job that clashes with their personality type.

Introspection. The mission of the university is not that of a vocational training institute but rather a higher education institute. Both the software industry and the university must do introspection to establish where they are failing to contribute to the offering of relevant software development education.

Generational awareness. It is clear from the study that the older software developers or Generation X have a different, more negative view of new recruits. SD students, as well as new recruits or Millennials must be aware of the values and beliefs of the older generations, especially

since most of them will start work under a Generation Xer. On the other hand the older software developers must be aware of the traits, values and beliefs of the Millennials in order to get the most out of their new recruits.

This study contributes to the sparse empirical literature on the perspective of the SD industry of their new recruits, the problems and challenges that are faced and the solutions to the problems as far as the SD industry is concerned. In South Africa some universities are significantly higher ranked than others and the same quality of education is not offered in all universities and university departments. Some of the highly ranked universities might argue that the above recommendations are already implemented by them, but it is clear from this study that the SD industry perceives a gap between SD education and the workplace and consequently all universities should take note in order to further improve the quality and relevance of SD education. SD educators and curriculum developers must take the expectations and suggestions of the SD industry in the academic preparation of future software developers into account. The SD industry can ascertain which contributions towards the education of new recruits can be made by the industry. With reference to this study, the SD students can observe the perspective of the SD industry of their new recruits and therefore be better prepared for the expectations of their future employers. If each of the role players in SD education does his/her bit, it can result in more relevant software development education.

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*, 20(3):349-356.

Augustine, R.K. 2001. Thanks kiddo! A survival guide for professional generation Xers. *Organization Development Journal*, 19(2):19-26.

Bateman, K. 2013. The irony of an unemployment problem and an IT skills shortage within the IT industry, *ComputerWeekly.com*, Oct. http://www.computerweekly.com/blogs/itworks/2013/10/the-irony-of-an-unemployment-p.html Date of access: 30 Jul. 2014.

Bayne, S. and Ross, J. 2011. 'Digital Native' and 'Digital Immigrant' Discourses. (*In* Land, R. and Bayne, S., *eds*. Digital difference: Perspectives on online learning. Rotterdam: Sense Publishers, p. 159-169).

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*, 24:129-140.

CareerJunction Index. 2014. Executive Summary, Jul. http://www.careerjunction.co.za/cji/exec-pdf/CJI_Executive_Summary.pdf Date of access: 30 Jul. 2014.

Creswell, J.W. and Clark, V.L.P. 2007. *Designing and conducting mixed methods research*. Thousand Oaks, CA: Sage Publications.

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*, 53(6):144-148.

Gruner, S. 2014. On the historical semantics of the notion of software architecture. *TD The Journal for Transdisciplinary Research in Southern Africa*, 10(1):37-66.

Guzdial, M., Prey, J., Topi, H., Urban, J., Cassel. L. and Schneider, D. 2009. Future of Computing Education Summit, June 25-26, 2009. http://www.acm.org/education/future-of-computing-education-summit/FoCES_web.pdf Date of access: 30 Jul. 2014.

Harris, L. 2012. Mind the ICT skills gap. *Brainstorm,* Sep. http://www.brainstormmag.co.za/index.php?option=com_content&view=article&id=4699:mind-the-ict-skills-gap Date of access: 24 Jul. 2013.

Hassan, N.R. 2008. Courting multidisciplinary illiteracy. (*In* Proceedings of the Americas Conference on Information Systems (AMCIS 2008). Paper 245.) http://aisel.aisnet.org/amcis2008/245 Date of access: 11 Mar. 2015.

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.

Johnson, R.B., Onwuegbuzie, A.J. and Turner, L.A. 2007. Toward a definition of mixed methods research. *Journal of Mixed Methods Research*, 1(2):112-133.

Jones, C., Ramanau, R., Cross, S. and Healing, G. 2010. Net generation or Digital Natives: Is there a distinct new generation entering university? *Computers & Education*, 54(3):722–732.

Kennedy, G., Judd, T.S., Churchward, A., Gray, K. and Krause, K. 2008. First year students' experiences with technology: Are they really digital natives? 'Questioning the net generation: A collaborative project in Australian higher education', *Australasian Journal of Educational Technology*, 24(1):108-122.

Kim, Y., Hsu, J. and Stern, M. 2006. An update on the IS/IT skills gap. *Journal of Information Systems Education*, 17(4):395-402.

Kitchenham, B., Budgen, D., Brereton, P. and Woodall, P. 2005. An investigation of software engineering curricula. *Journal of Systems And Software*, 74(3):325-335.

Lee, C. and Han, H. 2008. Analysis of skills requirement for entry-level programmer/analysts in Fortune 500 corporations. *Journal of Information Systems Education*, 19(1):17-27.

Lee, S., Yen, D., Koh, S. and Havelka, D. 2001. Evolution of IS professionals' competency: an exploratory study. *Journal of Computer Information Systems*, 41(4):21-30.

Lethbridge, T.C. 2000. Priorities for the education and training of software engineers. *Journal of Systems and Software*, 53(1):53-71.

Lethbridge, T., Diaz-Herrera, J., LeBlanc, R. and Thompson, J.B. 2007. Improving software practice through education: Challenges and future trends. (*In* 2007 Future of Software Engineering. IEEE Computer Society, Washington, DC. p. 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, NY. p.105-110).

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*, 30(1):6-12.

Macon, M. and Artley, J.B. 2009. Can't we all just get along? A review of the challenges and opportunities in a multigenerational workforce. *International Journal of Business Research*, 9(6):90-94.

Mawson, N. 2010. ICT skills shortage to cost SA. *ITWeb*, http://www.itweb.co.za/index.php?option=com_content&view=article&id=29992 Date of access: 24 Jul. 2014.

McAllister, N. 2012. Here's how to solve America's developer shortage. *Infoworld*, http://www.infoworld.com/d/application-development/heres-how-solve-americas-developer-shortage-185042 Date of access: 12 May 2013.

McGettrick, A., Boyle, R., Ibbett, R., Lloyd, J., Lovegrove, G. and Mander, K. 2004. *Grand challenges in computing education*. British Computer Society, Swindon, United Kingdom.

Mead, N. 2009. Software engineering education: How far we've come and how far we have to go. *Journal of Systems and Software*, 82(4):571-575.

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*, 85(7):1607-1620.

Murray, A. 2015. What millennials do and don't want from their employers. *Fortune.Com*, http://fortune.com/2015/03/05/millenials-best-companies/ Business Source Premier. Date of access: 30 Mar. 2015.

Oblinger, D. and Oblinger, J. 2005. Educating the net generation. Louisville, CO: Educause. http://www.educause.edu/ir/library/pdf/pub7101.pdf Date of access: 30 Mar. 2015.

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*, 48(1):22.

Reisman, S. 2004. Higher education's role in job training. *IT Professional*, 6(1):6-7.

Saiedian, H. 2009. Software engineering challenges of the "Net" generation. *Journal of Systems and Software*, 82(4):551-552.

SAQA (South African Qualifications Authority). 2000. The National Qualifications Framework and Curriculum Development. http://www.saqa.org.za/docs/pol/2000/curriculum_dev.pdf Date of access: 16 Nov. 2011.

Schawbel, D. 2012. Millennials vs. Baby Boomers: Who would you rather hire? *Time*, Mar. http://business.time.com/2012/03/29/millennials-vs-baby-boomers-who-would-you-rather-hire/ Date of access: 31 Oct. 2013.

Schreiner, C. and Sjøberg, S. 2004. Sowing the seeds of ROSE. Background, rationale, questionnaire development and data collection for ROSE (The Relevance of Science Education) - a comparative study of students' views of science and science education. *Acta Didactica*, 4:1-120.

Shaw, M. 2000. Software engineering education: A roadmap. (*In* Proceedings of the Conference on the Future of Software Engineering, ACM, New York, NY. p. 371-380).

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, NY. p. 607-608).

Soni, S., Upadhyaya, M. and Kautish, P. 2011. Generational differences in work commitment of software professionals: Myth or reality? *Abhigyan*, 28(4):30-42.

Spicer, M. 2011. What does business expect from its incoming recruits? Do they meet our expectations? (*In* The International Conference of the International Association for Cognitive

Education in Southern Africa (IACESA), Keynote Address). http://www.businessleadership.org.za/gup/filez/Speeches_IACESA_Conference_17_February_2011.pdf Date of access: 31 Oct. 2013.

Steyn, A.G.W., Smit, C.F., du Toit, S.H.C. and Strasheim, C. 1999. Modern statistics in practice. Pretoria: Van Schaik.

Surakka, S. 2007. What subjects and skills are important for software developers? *Communications of the ACM*, 50:73-78.

Tashakkori, A. and Creswell, J.W. 2007. The new era of mixed methods (editorial). *Journal of Mixed Methods Research*, 1:3-7.

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*, 37(1):21-54.

APPENDIX A

Factor	**Questionnaire items**
Critical outcomes required in software development modules.	In the software development modules students should learn to work with others as a member of a team or group.
	The software development modules should require from students to organize and manage themselves effectively.
	In the software development modules students should learn to collect and critically evaluate information.
	In the software development modules students should learn to communicate effectively, both verbally and in writing.
	To be a good software developer you have to
Positive attitude towards colleagues/management.	have a willingness to listen
	have a willingness to take instructions
	have respect for others
	have a desire to succeed (realistically ambitious)
Positive attitude towards tasks/work.	be prepared to work hard
	be prepared to learn (a thirst for knowledge)
	have good time-management skills
	have a preparedness to take responsibility
Emotional/social skills required.	have modern leadership skills, such as self-confidence and a preparedness to lead by example
	have the ability to relate well to and to build relationships with others (emotional intelligence)

	have a reasonable level of general knowledge

Chapter 5

Article 3

Software: University Courses vs Workplace Practice

Mrs J. Liebenberg[a]

Prof. M. Huisman[a]

Prof. E. Mentz[b]

[a]**Department of Computer Science and Information Systems,**
Faculty of Natural Sciences,
North-West University,
Potchefstroom,
2520,
SOUTH AFRICA

[b]**Faculty of Education Sciences,**
North-West University,
Potchefstroom,
2520,
SOUTH AFRICA

Manuscript published in: ***Industry and Higher Education***

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

Chapter 6

Article 4

The relevance of software development education: Students vs Professionals

Mrs J. Liebenberg[a]

Prof. M. Huisman[a]

Prof. E. Mentz[b]

[a]**Department of Computer Science and Information Systems,**
Faculty of Natural Sciences,
North-West University,
Potchefstroom,
2520,
SOUTH AFRICA

[b]**Faculty of Education Sciences,**
North-West University,
Potchefstroom,
2520,
SOUTH AFRICA

Manuscript submitted and reviewed at the journal *Information Systems Management*. See Appendix H for proof of submission and review.

The Relevance of Software Development Education: Students vs Professionals

Janet Liebenberg[a], Magda Huisman[a] and Elsa Mentz[b]

[a]Department of CS and IS, North-West University, Potchefstroom campus, South Africa.
[b]Faculty of Education Sciences, North-West University, Potchefstroom campus, South Africa.

Abstract Software development students want a relevant education and the software industry wants future employees to receive a relevant education, but are these two relevance perspectives compatible? This paper reports on a quantitative study that investigated the compatibility of the views of software development students and professionals regarding the relevance of software development education. The analysis revealed matching and differing views and recommendations to industry and university for relevant software development education are made.

Keywords: software development education, software industry, software professionals, computing curricula.

Introduction

The overall levels of technical and soft skills needed from software developers are not sufficient to meet demand, not only in South Africa, but also worldwide (Biztech Africa, 2013; Connolly, 2013).

In promoting education, software developer Bill Gates (2005) focuses on his foundation's "3Rs" of "Rigor, Relevance and Relationships". The central pillar of relevance highlights the need for students to have courses and projects that clearly relate to their lives and their goals. At the same time, the software development (SD) industry expects students to be educated in courses and projects that are professionally relevant and that prepare them well for the workplace (Moreno, Sanchez-Segura, Medina-Dominguez, & Carvajal, 2012). Students want a relevant education and the industry wants future employees to receive a relevant education, but are these two relevance perspectives compatible?

Numerous studies have investigated the relevance of education at university level, but have been mostly concerned with ways to make the education relevant to the needs of industry. In his article entitled “Higher education: who cares what the customer wants?” Reisman (2004) argues that

studies in higher education often neglect to collect data from two of higher education's most important players, namely faculty and students. No study could be found that studied and compared both the views of SD students and SD professionals regarding the relevance of SD education. 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 compatibility of the view of SD students and the view of SD professionals regarding the relevance of SD education. The research questions were:

- What are SD students' views on the relevance of SD education?
- What are professional software developers' views on the relevance of SD education?
- Are the views of SD students regarding the relevance of SD education compatible with the views of SD professionals?

The results could help universities (SD educators, curriculum developers) and the SD industry (employers, corporate trainers) to form a picture of on the one hand, matching views and on the other hand differing views regarding the relevance of SD education. University and industry can relate the results of the study to the academic preparation of future software developers, as well as the continued education and training of software developers. It can therefore result in more relevant SD education with regard to students, as well as the software industry and might contribute to meeting the demand for skilled software developers.

Clarification of Terminology

For the purpose of this study, it is necessary to explain and clarify certain key terms:

Software development (SD) - the ISO (International Organization for Standardization) and the IEC (International Electrotechnical Commission) define developer as an individual or organisation 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 purpose of this study *Software development (SD)* 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 organisations that develops/develop software, typically for distribution and use by other people or organisations (ISO/IEC, 19770-2, 2009). For the purpose of this study, *Software Industry* will refer to software manufacturers, as well as organisations where software is not the organisation's main product but where software is developed for use within the organisation.

Conceptual Framework

Relevance is broadly defined as closely connected or appropriate to the matter in hand; or practical and especially social applicability (Oxford English Dictionary, 2014). Labaree (2008) emphasises relevance as a function not only of person and purpose, but also of place and time. He argues that the question "useful to whom and for what?" needs to be answered because a wide array of actors is involved. This study focused on relevance for two actors namely SD students and SD professionals.

The Student-Centric View on the Relevance of Software Development Education

Relevance in the educational context can be defined as the applicability of what is taught to the needs and interests of students and society (Holbrook, 2009). The process of instruction and learning is designed to make what is learnt relevant/current to the time so that it can be implemented in the social environment. As a result, the student then sees the learning as interesting, meaningful, timely, important and useful, and it builds on the intrinsic motivation of the student for self-concern, self-involvement, self-appreciation and self-development (Holbrook, 2009). Students need to see the relevance of teaching and learning, as it applies to them personally (their own lives, their career expectations, the wishes of their parents), or the relevance as it applies to society (wishes of the community, employers, the university) or as it applies to them professionally (the content/curriculum is meaningful and interesting) (Holbrook, 2003).

Students in current university classes are described by a great number of writers as the Net generation, also known as the Millennial Generation or Generation Y. The Net generation has unique modes of communication, learning preferences, social choices, and entertainment preferences defined by their early exposure to technology (Saiedian, 2009). Millennials want flexible work schedules, they do not like traditional office rules and hierarchies and they want continuous feedback and career advice from managers (Schawbel, 2012). Millenials want to wear jeans to work and especially in IT companies, the norm is to wear jeans, sneakers, flip-flops and sweatshirts - former Apple CEO Steve Jobs' black turtleneck with jeans and Facebook CEO Mark Zuckerberg's famous hoodie are good examples. However, not all today's students can be described as the Net generation, since not all students had and still have the benefit of cutting-edge technology. They may have information literacy characteristics and IT skills quite different from the typical Net generation (Lorenzo, Oblinger, & Dziuban, 2007).

Merkofer and Murphy (2009) reported on a study in South Africa where higher education institutions closer to the more rural areas highlighted two groups of students who do not successfully follow through with their chosen ICT related course - students who are under-

prepared and those of lesser means. When insufficiently prepared students enter tertiary education, they find the course work difficult and drop out early in their first year of study. Students from more disadvantaged backgrounds struggle to pay fees and as a result tend to drop out later in the year.

The attraction to and retention of students in computing courses are continuously under investigation, and some of the contributing factors identified are motivation, parental/peer pressure, culture, pre-college experience, and confidence. Furthermore, initial positive experience with computing, narrow perceptions on computing careers and career expectations, and matching perceived abilities with the requirements of the discipline were identified as key factors that influence students' decision, first, to pursue computing courses and, second, to study the field further (Klawe, 2001; Margolis & Fisher, 2002; Tillberg & Cohoon, 2005; Gupta, 2005).

An issue that is closely related to the attraction and retention of students is the gender composition in computing classes, with males dominating by far. Women are attracted when they recognise computing as a form of communication, a means of creative self-expression, or as the road to an occupation where they may be of service. A great proportion of men, on the other hand, describe an early and persistent magnetic attraction between themselves and computers; they have a greater technical appreciation of computers and they enjoy playing computer games (Klawe, 2001; Margolis & Fisher, 2002; Tillberg & Cohoon, 2005; Blum & Frieze, 2005).

From the above it is clear that some of the contributing factors involved in students' views on the relevance of SD education are their generation, preparedness for university, financial means, gender and the above-listed attraction and retention factors. Furthermore, relevance of education for students is related to their teaching and learning, their interests, their career expectations, the wishes of their parents/community/employers/university and the content/curriculum.

The Industry-Centric View on the Relevance of Software Development Education

Relevance in the industry context according to Wohlin and Regnill (1999) means "*that the education prepares students so that they are ready to cope with large-scale software development. It is also important that students are aware of the challenges and proven techniques related to industrial development of software*".

The South African Qualifications Authority (SAQA, 2000) requires "Critical Cross-field Outcomes" to be achieved in all their registered qualifications. According to Spicer (2011), any business needs from its incoming recruits these seven critical skills and abilities, namely problem solving and creative thinking; being able to work in a team; the ability to manage oneself and one's activities, being able to critically evaluate information; good written and verbal communication; an effective use of science and technology; and being able to demonstrate an understanding of the

world as a set of related systems by recognising that problem-solving contexts do not exist in isolation. These outcomes describe what can also be called soft skills or non-technical skills. Employers are looking for professionals who can add value to their organisations by working successfully in interdisciplinary teams rather than focussing exclusively on a narrowly defined technical job (McKinnon & McCrae, 2012).

Industry is, similar to the university, challenged to attract and retain skilled employees (Stokes, 2000). The research of Bullen, Abraham, Gallagher, Simon and Zwieg (2009) examined workforce trends in IT-provider companies and identified problems in the following areas: graduates who are not trained in areas that the industry is seeking; thin pipeline for specific technical skills; increasing pressure to source IT capability; and lag in university responsiveness to the needs of the industry.

Merkofer and Murphy (2010) gave an overview of the key findings of an Accenture research study on the e-skills shortage in South Africa where players from the ICT industry and tertiary education sector were interviewed and surveyed. Although the study covered the whole spectrum of e-skills (Level 1 - (e-literacy); Level 2 - (e-skills); and Level 3 (ICT specialist), the results indicated that e-skills taught at tertiary level do not sufficiently prepare students for the expectations of their roles in employment.

Several studies suggest 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 relevant studies.

Table 1 Studies on the knowledge and skills gap from the industry's perspective

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-emphasised.
Kitchenham, Budgen, Brereton, & Woodall (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-emphasised.
Kim, Hsu, & Stern (2006)	Examines IS/IT skills gaps from three perspectives: end users, academia, and IS/IT employers.	Gaps in project management, security, ERP, and the integration of soft skills.
Benamati & 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.

Author(s)	Study	Results
Lee & 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, Li, & Williams (2009)	Surveyed and compared the perceptions academics have of the importance of various skills for entry-level IT workers with the view that IT managers have.	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, organisational skills and work experience in the same order of importance.
Calitz (2010)	Interviewed managers in businesses employing ICT graduates to identify graduate skill requirements.	Software development skills and business skills were emphasised, but a number of soft skills and technical skills were also identified. New skills, including business analysis and analytics, business process modelling, mobile application development and internationalisation skills were identified.
Gallagher, Kaiser, Simon, Beath, & Goles (2010)	To investigate the premise that IT professionals should possess a varied set of skills, a total of 104 senior IT managers were interviewed by 20 researchers in 94 non-IT companies.	A mix of skills is essential for IT professionals but the skills most critical are non-technical skills, such as project management, business-domain knowledge and relationship skills.
Clark, Zukas, & Lent (2011)	Case study on three cases that focused on the transition from university to work.	Noted a growth in roles focused upon management particularly project management, planning and strategy and a decline in technical roles. They argue that to understand transitions, the field, the habitus of the individual and the capital need to be focused on.
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 concerns tasks associated with IT business consultancy, knowledge related to leadership, negotiation or giving presentations.

Table 1 indicates that seven of the ten studies found that professionals in the computing field lack non-technical or soft skills, such as communications skills (negotiation or giving presentations), client-facing capabilities, leadership skills and relationship skills. Nine out of the ten studies in Table 1 found that business-domain knowledge and skills such as project management, business consultancy, business analysis and analytics, business process modelling, entrepreneurial traits and internationalisation skills are needed in the industry. Computing professionals also lack technical expertise, such as HCI/user interfaces, real-time system design, software (requirements gathering and analysis; cost estimation; metrics; development; reliability and fault tolerance; configuration and release management), web-based programming, mobile application development, security and cryptography, ERP, operating systems, multimedia and computer graphics.

Industry's view on the relevance of SD education is therefore determined not only by the knowledge and skills (technical, non-technical and business) acquired by graduates at university but also by their awareness of the proven techniques, challenges and expectations of their roles in employment.

The Role of the University in Relevant Software Development Education

Traditionally, universities prepared students for either graduate work or for employment, but universities regularly face pressure from potential employers to deliver graduates with immediately useful skills, and in the process research and innovation in the field are sacrificed (Wells & Sevilla, 2001; Gupta, 2005; Reisman, 2004; Shaw, 2000). 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 (Topi, Valacich, Wright, Kaiser, Nunamaker, Sipior, & de Vreede, G., 2010). Lethbridge, Diaz-Herrera, LeBlanc and Thompson (2007) argue that most software quality and budgetary problems have their root cause in human error or lack of skill. These in turn arise in large part from inadequate education. Therefore by improving education, software and software practice should improve.

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 (Shaw, 2000; Gupta, 2005; McKinnon & McCrae, 2012). To effectively fill 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 correctly tackle the problems that they will face during their professional career (Loftus, Thomas, & Zander, 2011; Bothe, Budimac, Cortazar, Ivanović, & Zedan, 2009).

The Joint Task Force on Computing Curricula (2013) emphasises that the education that 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 (e.g., patience, time management, work ethic), but for the rest of the skills, provision must be made through specific curricula, although Bailey and Stefaniak (2001) point out that it remains a struggle to incorporate opportunities that foster the development of necessary nontechnical skills into a traditional technical academic curriculum.

Plice and Reinig (2007) found that emphasising communication and teamwork skills, while maintaining the existing curriculum balance between business and technical content, is indicated as an appropriate strategy to align the computing curricula with the needs of industry. Shaw (2000)

states that universities regularly face pressure from potential employers to sacrifice systematic understanding for immediately useful skills and each university must select its own balance between immediate and long-term knowledge.

Students want a relevant education and the industry wants future employees to receive a relevant education. The role of the university in providing a relevant education has always been and continues to be a relatively uncontentious role. However, the views of students and professionals regarding a relevant education might differ and the question is whether these two relevance perspectives are compatible.

Methodology / Data Collection

This quantitative study was conducted in South Africa and consisted of two groups of participants, namely SD students and SD professionals. In this section, the demographics of the participants, as well as the data collection and analysis are discussed.

Participants

***SD students*.** The target population was four academic year groups of the SD classes at a university in South Africa and 386 questionnaires were posted as an assignment on the e-learning system to these students. The number of usable responses received totalled 297, making for an overall response rate of 76.9%.

Table 2 Profile of software development students (n=297)

		Number (%) of students
Gender	Male	222 (75%)
	Female	75 (25%)
Academic Year	1	145 (49%)
	2	76 (26%)
	3	55 (19%)
	4 (Hons)	21 (7%)
Self-rated academic performance	<50%	7 (2%)
	50% - 59%	61 (21%)
	60% – 74%	158 (55%)
	>= 75%	62 (22%)
Access to a computer since Gr1	Yes	170 (58%)
	No	123 (42%)

Table 2 provides a summary of the biographic data. The gender profile is typical of most CS classes with only 25% of the respondents being women. The participants included 276 BSc undergraduate students and 21 students enrolled for the subsequent BSc(Hons) in CS and IS. Most of the undergraduate students follow the three-year BSc in IT and CS program in the Department of CS and IS. A subsequent one-year BSc (Hons) in CS and IS is offered and this degree gives access to a Master's degree in CS. The students were asked to self-rate their academic performance and 62 (22%) of the group thought they are distinction candidates, but seven (2%) students

expected to fail. These students are not your typical Net generation since only 58% of them had access to a computer from a relatively young age.

SD professionals. In 2013's last quarter 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 personally contacted via e-mail and requested to complete the anonymous online survey. The link of the survey was also forwarded by some of the respondents 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 totalled 214, which indicates a response rate of around 21%, although an exact figure cannot be determined.

Table 3 provides a summary of the biographic data. The gender profile is a concern but not surprising with only 8% of the respondents being female. The age profile indicates that 42% of the respondents are young (<30) software developers.

In terms of the respondents' education 49% of them are 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 qualifications with 4.5% of the respondents having only a high school certificate. The work experience of respondents indicates that 70.5% of them have more than five years' work experience.

Table 3 Profile of software development professionals (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%)
Education	Matric / High School Certificate	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%)

Data Collection, Instrument and Analysis

***SD students*.** An initial list of questions was developed by both writing new items and adapting items from available surveys, such as ROSE (Schreiner and Sjøberg 2004), the SAQA's list of "Critical Cross-field Outcomes" and topics commonly found in computer science programs. Once the initial questions were generated, they were sent to three industrial and two academic experts to refine the instrument and they added additional topics. Feedback from this pilot study served as the basis for correcting, refining, and enhancing the questions and it resulted in a questionnaire with a pool of 89 items. The first section of the questionnaires gathered information on the biographic data of the respondents as shown in Table 2.

The questionnaire was further divided into three domains (See Appendix A). The first domain "In class" with 14 items enquired on their perceptions of their SD classes, such as industry experience of lecturers. The second domain "My career" had 12 items and gathered data on their future career, such as what is expected from a good software developer. The first and second domain were accompanied by a five-point Likert response scale from 1 (Strongly disagree) to 5 (Strongly agree). The domain, "My career", included a question: "In your first job after graduation, which job would you ideally want?" with a list of 36 choices from which a maximum of three alternatives could be selected.

The last domain "Software topics" with 63 items investigated their interest in software topics. The participants were asked: "How interested are you in learning about the following?" with a five-point Likert response scale: Not interested / Slightly interested / Neutral / Quite interested / Very interested.

SD professionals. The questionnaire that was used for the SD students was used as a point of departure for the questionnaire for the SD professionals. The first section of the questionnaire gathered information on the biographic data of the respondents as shown in Table 3. The questionnaire was further also divided into the three domains (see Appendix A). The last domain "Software topics" 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).

SD students and SD professionals. Factor analysis was used to investigate the 89 items in more detail to reduce the variables into a smaller number of factors. The 511 responses were examined using principal components factor analysis and the 89 attitude items yielded 18 interpretable factors and 11 items were being handled as single research variables. Factors were named

according to their main context. A Cronbach's α coefficient was calculated for each of the factors and were found as Table 4 shows, to be reliable ($\alpha \geq 0.60$).

Table 4 Factors* (with reliability coefficients) and single variables

Factors	Cronbach's alpha (α)
Information systems	0.842
Computer hardware and electrical/computer engineering	0.948
Software testing and maintenance	0.857
Computer science theory	0.923
Real-time and systems programming	0.912
Mathematics and statistics	0.927
Mobile technologies	0.950
Software development methodologies	0.829
Software management	0.864
General software design	0.882
Specialised application techniques	0.925
Web design and development	0.901
Hardware: Data transmission	0.820
Software engineering methods	0.873
Critical outcomes required in courses	0.849
Knowledge of course requirements	0.606
Positive attitude towards work and colleagues	0.800
Emotional/social skills required	0.760
Single variables	
Software developers needed	
Industry experience of lecturers	
Project work	
People from industry teaching	
Good examination results	
Neat and tidy appearance	
Data warehousing	
Security and cryptography	
Game development	
HCI/user interfaces	
Databases	

** See Appendix A for the items in each factor and descriptive statistics*

Basic analysis of quantitative data was done by calculating the mean values and standard deviation of each of the 39 variables. The statistical tests used in the 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.

Threats to validity

In this study the focus was on SD professionals and SD students. All the students fitted the label "software development student", since they attended a software development course, but not all of them will end up pursuing a career in software development. However, the students' future careers cannot be predicted and their views and opinions can be of great value.

Furthermore, all the respondents from industry fitted the label "software developer professional", but what a software tester, web developer, software architect, project manager, or any other software-related professionals might view as important might vary. The researchers carefully selected the software development groups from LinkedIn and MyBroadband, and the views and opinions of these experienced software developers do not necessarily only represent their specific field, because a project manager, for instance will know what knowledge and skills are important in the software development team.

RESULTS AND DISCUSSION

In this section, important data for each of the research questions are considered.

Software development students' view

In Table 5 the results of the first two domains "In class" and "My career" are shown in the top section and the results of the third domain "Software topics" follow in the bottom section.

The top five ideal jobs from 36 choices for the software development students are software developer, computer game designer, computer and video game developer, systems analyst and software engineer. It is worth noting that game design and development are among the top jobs for these students although South Africa has a very small video game development industry.

Table 5 shows that the mean values of five of the 10 variables in the top section are relatively high. Students feel that a positive attitude towards work and colleagues is very important. Students also feel strongly about the need for emotional and social skills in the workplace and they agree that more software developers are needed in South Africa. The only variable showing a lower mean is that students feel that it is not too important for people from industry to come in and teach some of their classes. Further analysis showed that the more advanced the students were in their studies, the more importance they attached to industrial experience and knowledge brought to them in their classes.

Students showed the most interest in the topics *HCI/user interfaces, Game development and Specialised application techniques*. Further analysis showed that the male students had a significant influence on the result regarding Game development and it concurs with the findings of Margolis and Fisher (2002) and Tillberg and Cohoon (2005).

Students showed the least interest in the topics *Computer science theory, Mathematics and statistics* and *Software development methodologies*.

Table 5 Software development students' views

	Mean*	Std. Deviation
Positive attitude towards work and colleagues	4.501	0.530
Emotional/social skills	4.247	0.628
Software developers needed	4.207	0.875
Critical outcomes required in courses	4.051	0.717
Project work	4.020	0.954
Knowledge of course requirements	3.859	0.694
Industry experience of lecturers	3.776	0.940
Neat and tidy appearance	3.753	1.225
Good examination results	3.616	1.153
People from industry teaching	2.290	1.231
HCI/user interfaces	4.080	1.076
Game development	3.922	1.355
Specialised application techniques	3.823	0.992
Databases	3.813	1.120
Security and cryptography	3.811	1.223
Web design and development	3.764	1.153
General software design	3.758	1.020
Mobile technologies	3.755	1.126
Real-time and systems programming	3.720	1.075
Information systems	3.628	1.094
Software testing and maintenance	3.597	1.053
Data warehousing	3.561	1.196
Hardware: Data transmission	3.555	1.109
Computer hardware and electrical/computer engineering	3.506	1.072
Software management	3.501	1.069
Software engineering methods	3.441	1.042
Software development methodologies	3.338	1.179
Mathematics and statistics	3.316	1.231
Computer science theory	3.199	1.215

* *Likert-style responses were ranked from 1 to 5 respectively*

The "Software topics" domain had a comment box where students could list other topics that interest them and included: *More programming languages (×3); Ethical hacking; Natural Language Processing; GIS (Geo Information Systems); Physics; Calculus; Financial mathematics; Data analysis via Excel; Statistical Analysis Systems*. The last two listed topics namely *Data analysis via Excel* and *Statistical Analysis Systems* are an indication that these students might want to further their studies and go into research and they feel the curriculum should make provision for knowledge and skills in data analysis. Their thoughts confirm what Reisman (2004), Shaw (2000) and Gupta (2005) stated regarding the role of the university to not only prepare students for employment but also for research.

Although the following topics were included in the questionnaire, the students felt so strong about it that they added it again: *Security; Graphic design; App development; 3-D games; Electronics and robotics; Industrial electronics*.

Software development professionals' views

In Table 6, like in Table 5 the results of the first two domains "Educational background of new recruits" and "Career" are shown in the top section and the results of the third domain "Software topics" follow in the bottom section.

Table 6 shows that the mean values of six of the 10 variables in the top section are relatively high. The SD professionals feel that it is very important for lecturers to have industry experience in their armour. The professionals also feel strongly about a positive attitude towards work and colleagues and they agree that people from industry should come in and teach some of the university classes. The only variable showing a lower mean is that professionals feel that it is not too important for software developers to have a neat and tidy appearance and this result confirms Schawbel's (2012) findings.

The SD professionals attached the greatest importance to the topics Databases, General software design and Web design and development. The result on Web design and development concur with Kitchenham (2005) that the industry views Web-related subjects and skills as important. The professionals viewed Specialised application techniques, Computer hardware and electrical/computer engineering and Game development as the three least important topics.

Table 6 Software development professionals' views

	Mean*	Std. Deviation
Industry experience of lecturers	4.425	0.752
Positive attitude towards work and colleagues	4.410	0.477
People from industry teaching	4.369	0.769
Critical outcomes required in courses	4.346	0.440
Software developers needed	4.322	0.890
Project work	4.322	0.734
Emotional/social skills	3.778	0.679
Knowledge of course requirements	3.201	0.596
Good examination results	3.033	1.041
Neat and tidy appearance	2.790	1.112
Databases	4.336	0.924
General software design	3.977	0.783
Web design and development	3.770	1.117
Software testing and maintenance	3.738	0.831
HCI/user interfaces	3.617	1.131
Software engineering methods	3.564	0.955
Security and cryptography	3.486	1.141
Mobile technologies	3.386	1.346
Software management	3.254	0.958
Real-time and systems programming	3.169	1.000
Software development methodologies	3.159	1.041
Computer science theory	3.040	1.112
Data warehousing	3.037	1.214
Information systems	2.776	1.082
Hardware: Data transmission	2.680	1.094
Mathematics and statistics	2.584	1.068
Specialised application techniques	2.228	0.971

	Mean*	Std. Deviation
Computer hardware and electrical/computer engineering	1.985	0.882
Game development	1.636	0.982

** Likert-style responses were ranked from 1 to 5 respectively*

The SD professionals had added the following topics and skills in the comment box in the "Software topics" domain: *General problem solving; Practical SDLC experience in a team setting; 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; Distributed Systems and Parallel computing; SOLID, Domain Driven Design and Test-Driven Design; UI design; Version control; Industry relevant languages, i.e. C# and proper platforms if they come from JAVA; Business knowledge.* The comment regarding Business knowledge echoes the sentiments of numerous previous studies (Kitchenham, 2005; Benamati and Mahaney, 2007; Lee and Han 2008; Calitz, 2010). Since many software companies operate internationally it is essential they make use of version control systems for distributed and collaborative development and a respondent pointed out that version control knowledge is important in the workplace.

As was the case with the students, the professionals felt so strongly about certain topics that although they were included in the topics, they added them again: *Object-oriented design and design patterns; SCRUM; Agile and Extreme Programming and software practices; Math - calculus/algebra/matrices, etc. - data structures - algorithms - machine learning; Database design and development - SQL is paramount.*

The one respondent made an important comment: "*They should learn the basic fundamentals and not just the methods to provide quick solutions"*.

Software Development Students vs Software Development Professionals

The results of the SD students and professionals were compared to determine if the views of SD students regarding the relevance of SD education are compatible with the views of SD professionals. Differences were analysed with a T-Test and Table 7 shows significant differences in means between 13 variables. Eleven variables reveal that the mean values of the students are significantly higher than the mean values of the professionals and for only two variables the SD professionals had significant higher means. The top section of Table 7 shows the results of the first two domains "In class" and "Career" where significant differences were found, the middle section shows the results of the third domain "Software topics" where significant differences were found and the rest of the variables with insignificant differences follow in the bottom section.

Two variables in the top section of Table 7 showed large practically significant differences

between the students' views and the professionals' views. The professionals felt very strongly that it is important for people from industry (themselves) to assist in teaching the students whereas the students felt that it is not necessarily that important. However, the closer the students came to entering the workplace, the more they valued industrial experience brought to them in their classes. McKinnon and McCrae (2012) state that undergraduate students commonly do not think about their employability skills until they are about to graduate and the students in this study confirmed that notion. The students on the other hand felt significantly more strongly than the professionals that the critical outcomes (teamwork; self-organised; information collection and evaluation; communication skills; science and technology use; worldview of related systems) are important for students to learn.

There were also four variables showing a medium practically significant difference between the students' and professionals' views. Three of the four variables (*Neat and tidy appearance; Emotional/social skills; Good examination results*) indicated greater importance for the students than the professionals. It is noteworthy that numerous studies, including this study indicated the importance of soft skills for the industry and now this study found that soft skills (emotional/social skills and critical outcomes) have even greater importance for students. The fourth variable *Industry experience of lecturers* is much more important to the professionals than to the students and it is not surprising since this finding concurs with the finding that the professionals also view the variable that people from industry should come in and teach some classes as essential.

For the "Software topics" in the middle section of Table 7, three variables showed large practically significant differences between the students' views and the professionals' views. For the three topics Game development, Specialised application techniques and Computer hardware and electrical/computer engineering the students showed significantly more interest than the professionals regarding the importance of these three topics.

There were also four variables showing a medium practically significant difference between the students' and professionals' views. The students showed more interest in the topics Hardware: Data transmission; Information systems; Mathematics and statistics; and Real-time and systems programming than the professionals' views of importance of these topics.

Table 7 Students vs Professionals

	Mean of Students (n=297)	Mean of Professionals (n=214)	Effect size	p
People from industry teaching	2.290	4.369	1.690**	<0.001
Critical outcomes required in courses	3.859	3.201	0.948**	<0.001
Neat and tidy appearance	3.753	2.790	0.786*	<0.001
Emotional/social skills	4.247	3.778	0.691*	<0.001

	Mean of Students (n=297)	Mean of Professionals (n=214)	Effect size	p
Industry experience of lecturers	3.776	4.425	0.691*	<0.001
Good examination results	3.616	3.033	0.506*	<0.001
Game development	3.922	1.636	1.687**	<0.001
Specialised application techniques	3.823	2.228	1.609**	<0.001
Computer hardware and electrical/computer engineering	3.506	1.985	1.418**	<0.001
Hardware: Data transmission	3.555	2.680	0.789*	<0.001
Information systems	3.628	2.776	0.779*	<0.001
Mathematics and statistics	3.316	2.584	0.595*	<0.001
Real-time and systems programming	3.720	3.169	0.512*	<0.001
Databases	3.813	4.336	0.467	<0.001
Data warehousing	3.561	3.037	0.432	<0.001
Knowledge of course requirements	4.051	4.346	0.411	<0.001
HCI/user interfaces	4.080	3.617	0.410	<0.001
Project work	4.020	4.322	0.316	<0.001
Mobile technologies	3.755	3.386	0.275	<0.050
Security and cryptography	3.811	3.486	0.266	<0.050
Software management	3.501	3.254	0.231	<0.050
General software design	3.758	3.977	0.214	<0.050
Positive attitude towards work and colleagues	4.501	4.410	0.171	<0.050
Software development methodologies	3.338	3.159	0.152	>=0.050
Software testing and maintenance	3.597	3.738	0.134	>=0.050
Computer science theory	3.199	3.040	0.131	>=0.050
Software developers needed	4.207	4.322	0.129	>=0.050
Software engineering methods	3.441	3.564	0.118	>=0.050
Web design and development	3.764	3.770	0.005	>=0.050

* *medium practically significant difference (d>=0.5)*
** *large practically significant difference (d>=0.8)*

The variables where the students' views are most compatible with the professionals' views (d<0.2) are Web design and development; Software engineering methods; Software developers needed; Computer science theory; Software testing and maintenance; and Software development methodologies. The ten remaining variables had such small practically significant differences that it can also be said that the views of the students are compatible with the views of the professionals regarding these variables.

Conclusions and Recommendations

The above results paint a picture for all the stakeholders of SD education of on the one hand, matching views and on the other hand differing views of software students and software developers. SD students and SD professionals are unanimous in their views that *more software*

developers are needed and the students' views are also compatible with the professionals' regarding a *Positive attitude towards work and colleagues.* In terms of software topics, the students' interest views match the professionals' importance views with *Web design and development; Software engineering methods; Computer science theory; Software testing and maintenance*; and *Software development methodologies*. The university must take cognisance and ensure these topics are well covered in the curriculum because in the process both the students' interests and the industry's requirements are satisfied.

There is a mismatch between the students' and the professionals' views regarding *People from industry teaching* and *Industry experience of lecturers*. The university and industry must work together to provide 'real-world' group projects from students' first year of study. Not only do these projects of simulating the workplace provide them with the opportunity to solve problems and apply their theoretical knowledge but it engages students in reflecting on their future career from their early study years.

The students' views are largely incompatible with those of the professionals regarding the following topics: *Game development*; *Specialised application techniques* and *Computer hardware and electrical/computer engineering*. Students' expectations and conceptions of SD careers seem somewhat misguided when the results are contemplated, especially considering their interest in game development and their ideal job of being a game designer/developer in a country with a very small games industry. Students should gain experience in the industry much earlier in their education to eliminate possible misconceptions regarding their future career. Furthermore, it is paramount for lecturers to gain and maintain experience in the software industry through for instance a weekly "industry-day" in order for them to aid eliminating misconceptions and not to be contributing to them.

There is a mismatch between the students' and the professionals' views regarding soft skills (*Critical outcomes required in courses* and *Emotional/social skills)* with the students regarding soft skills significantly more important than the professionals. However, the importance for the industry of soft skills is proved in this study and numerous other studies and it should be a clear indication to universities to pay attention to the development of the soft skills of their students. It should furthermore be an indication to the software industry that their new recruits agree with them on the importance of soft skills and it might also reflect students' awareness of their own weaknesses and skills gaps.

Students indicated that statistical data analysis was found to be missing from the presented software topics. The role of the university is not only to prepare students for employment but also

for research and each university should prepare students for the workplace but preparation for research and innovation is essential.

The SD professionals indicated that knowledge of version control was found to be missing from the presented software topics and it is essential for universities to include it in their curricula before their students enter the world of distributed, collaborative software development. The software industry on the other hand must expect graduates to be underprepared in version control and they can put mechanisms in place, such as in-house training on version control to equip their new recruits.

Universities should consider increasing their coverage of the topics that interest the students, but also take cognisance of the topics regarded as important by the software industry. The software industry on the other hand must convey the important, but also the nonessential topics and issues to students and academia.

Industry-university collaboration should be launched and maintained to establish two-way knowledge and skill exchanges which will result in joint research and education projects and will keep educational offerings grounded in professional practice. It is essential to work towards compatible views regarding the relevance of software development education between industry and higher education's largest stakeholder, namely the students.

References

Aasheim, C., Li, L. & Williams, S. (2009). Knowledge and skill requirements for entry-level information technology workers: a comparison of industry and academia. *Journal of information systems education*, 20(3), 349-356.

Bailey, J., & Stefaniak, G. (2001). Industry perceptions of the knowledge, skills, and abilities needed by computer programmers. In *Proceedings of the 2001 ACM SIGCPR Conference on Computer Personnel Research* (SIGCPR '01), Serva, M. (Ed.). New York, NY, USA, pp. 93-99, ACM.

Benamati, J., & Mahaney, R. (2007). Current and future entry-level IT workforce needs in organizations. In *Proceedings of the 2007 ACM SIGMIS CPR Conference on Computer Personnel Research* (SIGMIS CPR '07). New York, NY, USA, pp. 101-104, ACM.

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

Blum, L., & Frieze, C. (2005). In a more balanced computer science environment, similarity is the difference and computer science is the winner. *Computing research news*, 17(3):2, May.

Bothe, K., Budimac, Z., Cortazar, R., Ivanović, M., & Zedan, H. (2009). Development of a modern curriculum in software engineering at master level across countries. *Computer Science and Information Systems*, 6(1), 1-21.

Bullen, C., Abraham, T., Gallagher, K., Simon, J.C., & Zwieg, P. (2009). IT workforce trends: implications for curriculum and hiring. *Communications of the association for information systems*, 24, 129-140.

Calitz, A.P. (2010). A model for the alignment of ICT education with business ICT skills requirements. Port Elizabeth: NMMU. (Thesis-DBA).

Clark, M., Zukas, M. & Lent, N. (2011). Becoming an IT Person: field, habitus and capital in the transition from university to work. *Vocations and Learning*, 4(2), 133-150.

Connolly, B. (2013, Feb 26). IT worker shortage continues as jobs remain unfilled. *CIO*. Retrieved October 31 2013 from http://www.cio.com.au/article/454650/it_worker_shortage_continues_jobs_remain_unfilled/

Gallagher, K.P., Kaiser, K.M., Simon, J.C., Beath, C.M., & Goles, T. (2010). The requisite variety of skills for IT professionals. *Communications of the ACM*, 53(6), 144-148.

Gates, B. (2005). The 3r's solution. Retrieved October 10 2014 from http://www.gatesfoundation.org/Media-Center/Speeches/2005/02/Bill-Gates-2005-National-Education-Summit

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*, 19(6), 423-431.

Holbrook, J. (2003). Rethink science education. *Asia-Pacific forum on science learning and teaching*, 4(2), 1-13.

Holbrook, J. (2009). Meeting challenges to sustainable development through science and technology education. *Science Education International*, 20(1), 44-59.

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.

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, NY, USA.

Kim, Y., Hsu, J., & Stern, M. (2006). An update on the IS/IT skills gap. *Journal of information systems education*, 17(4), 395-402.

Kitchenham, B., Budgen, D., Brereton, P., & Woodall, P. (2005). An investigation of software engineering curricula. *Journal of systems and software*, 74(3), 325-335.

Klawe, M. (2001). Refreshing the nerds. *Communications of the ACM*. 44(7), 67-68.

Labaree, D. F. (2008). Comments on Bulterman-Bos: the dysfunctional pursuit of relevance in education research. *Educational researcher*, 37(7), 421-423.

Lee, C., & Han, H. (2008). Analysis of skills requirement for entry-level programmer/analysts in Fortune 500 corporations. *Journal of information systems education*, 19(1), 17-27.

Lethbridge, T.C. (2000). Priorities for the education and training of software engineers. *Journal of systems and software*, 53(1), 53-71.

Lethbridge, T., Diaz-Herrera, J., LeBlanc, R., & Thompson, J.B. (2007). Improving software practice through education: Challenges and future trends. In *2007 Future of Software Engineering.*, Washington, DC. pp. 12-28 IEEE Computer Society.

Loftus, C., Thomas, L., & Zander, C. (2011). Can graduating students design: revisited. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (SIGCSE '11), New York, NY, pp. 105-110, ACM.

Lorenzo, G., Oblinger, D., & Dziuban, C. (2007). How choice, co-creation, and culture are changing what it means to be net savvy. *Educause Quarterly*, 30(1), 6-12.

Margolis, J., & Fisher, A. (2002). *Unlocking the clubhouse: Women in computing*. Cambridge, MA: MIT Press.

McKinnon, S., & McCrae, J. (2012). Closing the gap: preparing computing students for employment through embedding work-related learning in the taught curriculum. *Industry and Higher Education*, 26(4), 317-322.

Merkofer, P., & Murphy, A. (2009). The e-skills landscape in South Africa. *Zeitschrift für Politikberatung*, 2(4), 685-695.

Moreno, A., Sanchez-Segura, M., Medina-Dominguez, F., & Carvajal, L. (2012). Balancing software engineering education and industrial needs. *Journal of systems and software*, 85(7), 1607-1620.

Oxford English Dictionary. Relevance. (2013). Retrieved October 10 2013 from http://oxforddictionaries.com/definition/english/relevant

Plice, R. K., & Reinig, B. A. (2007). Aligning the information systems curriculum with the needs of industry and graduates. *Journal of computer information systems*, 48(1), 22.

Reisman, S. (2004). Higher education's role in job training. *IT Professional*, 6(1), 6-7.

Saiedian, H. (2009). Software engineering challenges of the "Net" generation", *Journal of systems and software*, 82(4), 551-552.

SAQA (South African Qualifications Authority). 2000. The National Qualifications Framework and Curriculum Development. Retrieved November 16 2011 from http://www.saqa.org.za/docs/pol/2000/curriculum_dev.pdf

Shaw, M. (2000). Software engineering education: A roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, New York, NY, pp. 371-380, ACM.

Schawbel, D. (2012, Mar 29). Millennials vs. Baby Boomers: who would you rather hire? *Time*. Retrieved October 31 2013 from http://business.time.com/2012/03/29/millennials-vs-baby-boomers-who-would-you-rather-hire/

Schreiner, C., & Sjøberg, S. (2004). Sowing the seeds of ROSE. Background, rationale, questionnaire development and data collection for ROSE (The Relevance of Science Education) - a comparative study of students' views of science and science education. *Acta Didactica*, 4, 1-120.

Spicer, M. (2011). What does business expect from its incoming recruits? Do they meet our expectations? In *The International Conference of the International Association for Cognitive Education in Southern Africa* (IACESA), Keynote Address. Retrieved October 31 2013 from http://www.businessleadership.org.za/gup/filez/Speeches_IACESA_Conference_17_February_2011.pdf

Stokes, S.L. (2000). Recruiting and retaining IT talent: Attracting and keeping IT talent. *Information Systems Management*, 17: 8-16.

Tillberg, H., & Cohoon, J. (2005). Attracting Women to the CS major. *Frontiers: a journal of women studies*, 26(1), 126-140.

Topi, H.; Valacich, J.S.; Wright, R.T.; Kaiser, K., Nunamaker, J.F.; Sipior, J.C.; & de Vreede, G. (2010) "IS 2010: Curriculum Guidelines for Undergraduate Degree Programs in Information Systems," *Communications of the Association for Information Systems*, 26, 359-428.

Wells, T.D. & Sevilla, C. (2001). Forming a Dialogue with Academia: Industry Requirements Versus Academic Programs, *Information Systems Management*, 18(1), 80-83.

Wohlin, C. & Regnell, B. (1999). Achieving Industrial Relevance in Software Engineering Education. In *Proceedings of the 12th Conference on Software Engineering Education and Training*, pp. 16-25.

Appendix A: Items and Descriptive Statistics

Factors	Items	Mean* of Students (n=297)	Mean* of Professionals (n=214)
Critical outcomes required in courses	*In the software development classes:*		
	... students should learn to work with others as a member of a team or group.	4.174	4.383
	...should require from students to organise and manage themselves effectively.	4.119	4.374
	... students should learn to collect and critically evaluate information.	4.020	4.491
	... students should learn to communicate effectively, both verbally and in writing.	3.997	4.360
	… students should learn to use science and technology effectively.	4.010	4.201
	… students must be able to demonstrate an understanding of the world as a set of related systems by recognising that problem-solving contexts do not exist in isolation.	3.976	4.266
Knowledge of course requirements	Students know what software developers do in the workplace.	3.839	2.696
	I know what the outcomes are for a software development degree.	3.710	3.495
	The university courses have high expectations of students.	3.945	3.178
	The instruction in the software development classes is relevant.	3.952	3.435
Industry experience of lecturers	Lecturers should have industry experience.	3.776	4.425
Project work	Projects play an important role in the education of students.	4.020	4.322
People from industry teaching	People from industry should be brought into software development classes	2.290	4.369
Good examination results	To be a good software developer you have to have a good set of exam results	3.616	3.033
Positive attitude towards work and colleagues	*To be a good software developer you have to:*		
	... have a good attitude including a willingness to listen and to take instructions	4.559	4.477
	... be prepared to work hard and to learn (a thirst for knowledge)	4.527	4.509
	... have good time-management skills	4.332	4.397
	... have respect for others	4.447	4.425
	... have a desire to succeed (realistically ambitious)	4.532	4.229
	... have a preparedness to take responsibility	4.600	4.423
Emotional/social skills	... have modern leadership skills like self-confidence and a preparedness to lead by example	4.135	3.645
	... have the ability to relate well to and to build relationships with others (emotional intelligence)	4.214	3.831

Factors	Items	Mean* of Students (n=297)	Mean* of Professionals (n=214)
	... have at least some idea of what career direction one wish to take	4.269	3.836
	... have a reasonable level of general knowledge	4.369	3.793
Neat and tidy appearance	... have a neat and tidy appearance	3.753	2.790
Software developers needed	More software developers needed in the country	4.207	4.322
Information systems	Information retrieval	3.574	3.322
	Decision support systems	3.572	2.720
	Expert systems	3.709	2.285
Computer hardware and electrical/computer engineering	Digital electronics and digital logic	3.662	2.192
	Microprocessor architecture	3.558	2.107
	Computer system architecture	3.673	2.495
	Analog electronics	3.217	1.603
	Digital signal processing	3.253	1.864
	Data acquisition	3.254	2.136
	Robotics	3.781	1.500
Software testing and maintenance	Performance measurement and analysis	3.571	3.505
	Testing, verification, and quality assurance	3.676	4.079
	Software reliability and fault tolerance	3.594	3.949
	Maintenance, reengineering, and reverse engineering	3.485	3.421
Computer science theory	Programming language theory	3.166	3.121
	Formal languages	3.249	2.995
	Computational complexity and algorithm analysis	3.211	3.159
	Information theory	3.146	2.883
Real-time and systems programming	Operating systems	3.842	3.196
	Systems programming	3.833	3.196
	Data transmission and networks	3.750	3.379
	Parallel and distributed processing	3.485	3.107
	Real-time system design	3.644	2.967
Mathematics and statistics	Discrete mathematics	3.237	2.491
	Probability and statistics	3.314	2.827
	Linear algebra and matrices	3.331	2.654
	Continuous mathematics	3.252	2.364
Mobile technologies	User interface design	3.824	3.393
	Mobile application development	3.801	3.336
	Security and privacy	3.723	3.477
	Compatibility	3.688	3.336
	Agile methods	3.268	3.500
	Extreme programming	3.505	2.696

Factors	Items	Mean* of Students (n=297)	Mean* of Professionals (n=214)
Software development methodologies	Scrum	3.128	3.280
Software management	Project management	3.782	3.322
	Software metrics	3.407	2.991
	Software cost estimation	3.349	3.154
	Configuration and release management	3.372	3.547
General software design	Data structures	3.574	3.981
	Algorithm design	3.594	3.668
	Software design and patterns	3.800	4.112
	Software architecture	3.760	4.089
	Object-oriented concepts and technology	3.856	4.364
	Specific programming languages	3.912	3.645
Specialised application techniques	Simulation	3.744	2.495
	Artificial intelligence	3.996	1.935
	Pattern recognition and image processing	3.754	2.248
	Computer graphics	4.035	2.187
	Parsing and compiler design	3.581	2.276
Web design and development	Web-based methods	3.647	3.477
	Interface design	3.822	3.720
	Security and privacy	3.849	3.935
	Web application development	3.841	3.949
Hardware: Data transmission	Network architecture and data transmission	3.675	2.916
	Telecommunications	3.424	2.444
Software engineering methods	Requirements gathering and analysis	3.364	3.874
	Formal specification methods	3.347	3.215
	Analysis and design methods	3.580	3.603
Data-warehousing	Data-warehousing	3.56	3.037
Security and cryptography	Security and cryptography	3.81	3.486
Game development	Game development	3.92	1.636
Human-computer interaction/user interfaces	Human-computer interaction/user interfaces	4.08	3.617
Essential subsystem design: Databases	Essential subsystem design: Databases	3.81	4.336

** Likert-style responses were ranked from 1 to 5 respectively*

Chapter 7

A framework for relevant software development education

7.1 SYNOPSIS OF STUDY

The research report comprised an introductory chapter, followed by a literature study and then four articles were presented. A synopsis of the first two chapters and each article will firstly be given, followed by a recommended framework for relevant SD education.

Chapter 1 was devoted to an orientation in terms of SD education and the need for measures to offer relevant SD education from the perspectives of both the software industry and SD students. These gave rise to the problem statement, the aim of the research and the research design. The research aims and objectives of the study as set out in chapter 1 were to determine the following:

- To investigate the relevance of software development education as it pertains to students.
- To investigate the relevance of software development education as it pertains to the software industry.
- To investigate the compatibility between the relevance of software development education for students and the relevance for the software industry.
- To develop a framework for relevant software development education

Chapter 2 was devoted to a literature review of software development education. The review of the literature was led by the stakeholders in SD education namely the students, universities and the software industry. Attention was paid to the software development class including the students and the educators. Next followed a look at the software developers and employers in the software development workplace. The literature regarding the SD class compared to

the SD workplace was considered as well as the challenges facing SD education. Lastly the role of the university in relevant software development education was considered with particular attention to SD curricula.

In Chapter 3 the manuscript entitled “The relevance of software development education for students” and published in the journal *IEEE Transactions on Education* was presented. This paper aimed to investigate the relevance of SD education from the perspective of the students.

In Chapter 4 the manuscript entitled “Industry’s perception of the relevance of software development education” and accepted for publication in *TD The Journal for Transdisciplinary Research in Southern Africa* was presented. The research questions this paper aimed to answer were:

- What is the SD industry’s view of its new graduates?
- Is there a gap between software development education and the workplace from the perspective of the industry?
- What are the problems and challenges surrounding SD education and what are the solutions to these problems and challenges?

In Chapter 5 the manuscript entitled “Software: university courses vs workplace practice” and published in the journal *Industry and Higher Education* was presented. The research questions were:

- What are the topics professional software developers learned in their formal education?
- What topics are important to professional software developers in their actual workplace?
- Is there a gap between SD education and the workplace from the perspective of the software industry?

In Chapter 6 the manuscript entitled “The relevance of software development education: students vs professionals” submitted to the journal *Information Systems Management* was presented. The research questions that were addressed in this paper were:

- What are SD students’ views on the relevance of SD education?
- What are professional software developers’ views on the relevance of SD education?

- Are the views of SD students regarding the relevance of SD education compatible with the views of SD professionals?

This chapter contains a recommended framework for relevant SD education derived from the research results emerging mainly from the empirical investigation but also from the literature study.

7.2 A FRAMEWORK FOR RELEVANT SOFTWARE DEVELOPMENT EDUCATION

A framework is defined as a “broad overview, outline, or skeleton of interlinked items which supports a particular approach to a specific objective, and serves as a guide that can be modified as required by adding or deleting items” (BusinessDictionary.com, 2011).
In technical terms, a software framework is defined as “a platform for developing software applications. It provides a foundation on which software developers can build programs for a specific platform. This streamlines the development process since programmers don't need to reinvent the wheel each time they develop a new application” (TechTerms.com, 2014). The above definitions were used as a guideline to create the framework in this chapter.

The framework is presented on the next page, followed by a detailed discussion of the interlinked items in the framework.

A FRAMEWORK FOR RELEVANT SOFTWARE DEVELOPMENT EDUCATION

Challenges

- Narrow perceptions on computing careers.
- Mismatch between personality type and career choice.
- Mismatch between ability and career choice.
- Disparity between entry qualifications.
- Misguided career expectations.
- Unsuccessful with studies.
- Adaptation to the workplace.

- Gap between industry needs and software education.
- Gap between students' interests and software education.
- Lack of research and innovation.
- Low enrolment.
- Gender composition in classes.
- Educators with insufficient talent and experience.
- Rapid change in technology.
- Generational gap.

- Shortage of software developers.
- Rapid change in character of software production.
- Lack of right knowledge and skills.
- High employee turnovers.
- Generational gap.

Challenges

Software development students

University

Software development industry

Students' interest

Non-technical knowledge and skills

- Time-management.
- Emotional intelligence / Social skills.
- Teamwork.
- Leadership.
- Organisational and managerial.
- Communicate - verbally and in writing.

Technical knowledge and skills

- HCI/user interfaces.
- Game development.
- Specialised application techniques.
- Databases.
- Security and cryptography.
- Different programming languages.

Technical knowledge and skills

- Databases.
- General software design.
- Web design and development.
- Software testing and maintenance.
- HCI/user interfaces.
- Version control.
- Problem solving.
- SQL.

Non-technical knowledge and skills

- Time-management.
- Teamwork.
- Organisational and managerial.
- Communicate - verbally and in writing.
- Emotional intelligence / Social skills.
- Leadership.

Industry's needs

Recommendations

A.Content

- Revise curriculum frequently.
- Include soft skills.
- Include business skills.
- Include research knowledge and skills.
- Align students' interest and industry's needs.
- Contextualised curriculum.

B.Teaching and learning

- Project-based learning.
- Work-integrated learning.
- Pair learning.
- Practical assignments.
- Game-based learning.
- Use experienced students as facilitators.
- Variety of delivery mechanisms.

C.Collaboration

- Lecturer visits to the software industry.
- Guest lectures by industry experts.
- Joint university-industry projects.
- Joint university-industry curriculum.

D.Attract and retain

- Provide career guidance.
- Inform about needs of the industry.
- Provide access to a diverse range of students.
- Offer bridging courses.
- Attention to: females; poor performers; no school IT.
- Generational awareness.

E.Real world

- Give assignments with a real-life context.
- Expose students to real world projects.

F.Life-long learning

- Appoint mentors in the workplace.
- Provision for continuing education.
- Keep educators updated.

Recommendations

RECOMMENDATIONS FOR RELEVANT SOFTWARE DEVELOPMENT

7.2.1 The challenges and problems in software development

This study focused on relevance of SD education for students and the industry; however it was firstly necessary to investigate the problems and challenges that might cause SD education to be less relevant. The problems and challenges facing three role players in SD education namely the SD students, the university and the SD industry were investigated. Tables 7-1 to 7-3 summarise the problems faced by students, the university and the industry respectively. The last column in each table indicates the origin of the information either from the literature study or from the empirical study or both.

7.2.1.1 Software development students

The problems faced by SD students mainly concern their future career, but their academic performance in order for them to reach their anticipated career is a challenge for any student. Table 7-1 presents the challenges and problems software development students are facing.

Table 7-1: Challenges and problems for software development students

Challenge/Problem	Discussion	See
Narrow perceptions on computing careers.	One of the contributing factors for the attraction to and retention of students in computing courses is narrow perceptions on computing careers. People in computing careers are stereotyped as nerds and students who love working with others and engaging in creative activities therefore do not choose to study computer science at universities.	• (Klawe, 2001)
Mismatch between personality type and career choice.	One of the software professionals in the study stated: "*Wrong career choice for their personality. Software developers are usually DISC "C" personalities (INTx in Myers Briggs). Project Managers and Architects are usually D/C - i.e. "Left Brainers". Business Analysts need more people / emotional skills*." Introversion, Thinking and Judging are more likely personality types in the computing disciplines than in the general population and these have an effect on career choices.	• Article 2 • (Galpin *et al.,* 2007)
Mismatch between ability and career choice.	Not all students have the ability to succeed as software developers. Students who have the ability to become professional software developers need to be identified; the other students can develop their ability to contribute to the wider range of computing activities; and the students who seem likely not to	• Article 1 • (McGettrick *et al*., 2004)

Challenge/Problem	Discussion	See
	succeed can swiftly be identified and appropriately advised.	
Disparity between entry qualifications.	Universities recruit students from high schools but students come from a variety of backgrounds and they offer a wide range of entry qualifications.	• Article 1 • (McGettrick *et al*., 2004)
Misguided career expectations.	Students' expectations and conceptions of SD careers seem somewhat misguided when the results are contemplated, especially considering their interest in game development and their ideal job of being a game designer/developer in a country with a very small games industry.	• Article 4
Unsuccessful with studies.	Numerous studies deal with academic performance of students, but in South Africa two groups of students who do not successfully follow through with their computing course were pointed out - students who are under-prepared for university study and students of lesser means.	• Article 1 • (Merkofer & Murphy, 2009)
Adaptation to the workplace.	New graduates adapt quickly to the workplace and they learn fast in comparison with recruits without degrees. However, the transition from university study into graduate work cannot be reduced to a simple formula based upon graduate credentials and employability skills and most students initially get a shock when they first arrive in the workplace.	• Article 2 • (Clark *et al*., 2011).

7.2.1.2 The university

The university as deliverer of SD education finds itself in the middle between the students on the one side and the industry on the other side and the role of the university is often viewed differently from the two sides. Universities are challenged to fulfil their role and Table 7-2 describes some of the challenges in fulfilling that role.

Table 7-2 Challenges and problems for the university

Challenge/Problem	Discussion	See
Gap between industry needs and software education.	This study and several other studies suggest a gap between the knowledge and skills demanded by the industry and the knowledge and skills gained by graduates of university computing courses.	• Article 2 • See also Table 1 in Article 3
Gap between students' interest	Students need to be exposed to courses and projects that interest them and clearly relate to	• Article 1

Challenge/Problem	Discussion	See
and software education.	their lives and their goals. The university is challenged to offer interesting courses to students.	
Lack of research and innovation.	The role of the university is not only to prepare students for employment but also for research and each university should prepare students for the workplace but preparation for research and innovation is essential.	• Article 4
Low enrolment.	Educators are acutely aware of the decline in enrolment that has occurred since 2000 with bright minds going into other disciplines. Low enrolments result in low levels of hiring of new PhD's, which will restrict the capacity for research in the field.	• (Lethbridge *et al*., 2007)
Gender composition in classes.	The male domination in computing classes is a matter of great concern for educators.	• Article 1 • (Margolis & Fisher, 2002)
Educators with insufficient talent and experience.	Universities are struggling to recruit good-quality staff resulting in insufficient talent being channelled back into the education system. The attractions of a relatively low-paying job as a lecturer in computing are not great and many young lecturers use their job as a stepping-stone to better paying jobs in the IT industry.	• (Gupta, 2005)
Rapid change in technology.	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 for universities.	• Articles 2, 3
Generational gap.	Most students belong to Generation Y and educators to Generation X. This generational shift has consequences for approaches to learning and it suggests that teachers and educational institutions have a responsibility to change in response to the assumed demands of this new generation of learners.	• (Jones *et al*., 2010)

7.2.1.3 The software development industry

The software development industry is a relatively young, fast-growing and rapid-changing industry that presents unique problems and challenges discussed in Table 7-3.

Table 7-3 Challenges and problems for the software development industry

Challenge/Problem	Discussion	See
Shortage of software developers.	Both the students and software development professionals in this study agree that there is a shortage of software developers, but this shortage is a worldwide phenomenon.	• Articles 1, 2
Rapid change in character of software production.	The continuing and rapid changes in the software industry is a reality that causes new recruits to arrive in the industry with already out-dated knowledge and software professionals being challenged to stay up to date with the knowledge and skills required in their job.	• Article 2 • (Shaw, 2000)
Lack of right knowledge and skills.	Students in the computing fields are graduating with a lack of the skills that companies are searching for and therefore ironically there is a shortage of software developers but also a noteworthy level of unemployment amongst computing graduates.	• Article 3 • (Bateman, 2013)
High employee turnovers.	A software professional in the study stated "*companies do not want to invest in graduates as they are afraid they will leave for better salaries*". Software companies are challenged by high employee turnovers and the management of the loss of skills and knowledge.	• Article 2
Generational gap.	Young professionals or Millennials are joining Generation X and Boomers in modern organisations. The three generations are inherently different - they approach work, work/life balance, accountability, delegation, loyalty, authority, motivation and reward systems differently resulting in sporadic differences, misunderstandings and tensions among workers from different generations.	• Article 2 • (Macon & Artley, 2009)

7.2.2 Interests and needs

This study focused on relevance for SD students and the SD industry and it was therefore necessary to investigate the knowledge and skills that interest students on the one hand and the knowledge and skills needed by the industry on the other hand. Table 7-4 and Table 7-5 list the technical and non-technical knowledge and skills that interest students and are needed by the industry respectively.

The first five topics in the technical knowledge columns came from the survey results and are listed in order of importance, but the other technical topics arose from the qualitative data. The skills in the non-technical knowledge columns came from the survey results and are listed in order of importance.

Article 1 and article 4 contained the results from the empirical study with the software development students. The first column in Table 7-4 shows the top five technical knowledge and skills that interest students and furthermore the students commented that they are interested in different programming languages. The non-technical knowledge and skills perceived as important by the students are listed in the second column in Table 7-4.

Table 7-4 Software development students' interests

Technical knowledge	Non-technical knowledge and skills
HCI/user interfaces.	Time-management.
Game development.	Emotional intelligence / Social skills.
Specialised application techniques.	Teamwork.
Databases.	Leadership.
Security and cryptography.	Organisational and managerial.
Different programming languages.	Communicate - verbally and in writing.

Articles 3 and 4 contained the results from the empirical study with the software development professionals. Table 7-5 shows the technical and the non-technical knowledge and skills that the software development professionals perceive as important. The top five topics in the first column from the survey data are followed by Version control, Problem solving, SQL and Business knowledge which the software development professionals commented that they need those knowledge and skills in the workplace.

Table 7-5 Software development industry's needs

Technical knowledge	Non-technical knowledge and skills
Databases.	Time-management.
General software design.	Teamwork.
Web design and development.	Organisational and managerial.

Software testing and maintenance.	Communicate - verbally and in writing.
HCI/user interfaces.	Emotional intelligence / Social skills.
Version control.	Leadership.
Problem solving.	
SQL.	
Business knowledge.	

In terms of technical knowledge it is noteworthy that both the students and professionals have Databases and HCI/user interfaces on their lists. In the comments of the professionals, SQL also emerged quite often which is a clear indication of the importance of Databases and the related query languages. It is also important to note that both the students and professionals view Time management and Teamwork as important non-technical skills.

7.2.3 Recommendations for relevant software development education

The underlying problems and challenges that might cause SD education to be less relevant to students and the industry were investigated from the viewpoint of SD students, the university and the SD industry. Furthermore, the relevant knowledge and skills that interest students and the relevant knowledge and skills needed by the industry were presented. Finally, recommendations for relevant software development education are made that round off the complete framework for relevant software development education.

Table 7-6 Recommendations for relevant software development education

Recommendation	Discussion	See
A. Content		
Revise curriculum frequently.	Universities must attempt to keep the curriculum abreast of the rapid change in technology. The challenge is a fast response to changes in technology and designing curriculum guidelines for the future, while supporting current practice.	Articles 2, 3
Include soft skills.	Universities must examine their curricula to ensure that not only technical, but also soft skills are included in their curriculums. The software industry and students view soft skills as important and students are aware of their own weaknesses and skills gaps.	Articles 2, 3, 4
Include business skills.	The software industry finds business knowledge and skills severely lacking from new recruits. The	Article 3

Recommendation	Discussion	See
	universities must ensure that not only technical, but also business skills are included in their curricula.	
Include research knowledge and skills.	The role of the university is not only to prepare students for employment but also for research and each university should prepare students for the workplace but preparation for research and innovation is essential.	Article 4
Align students' interest and industry's needs.	The university must take cognisance of the students' interests and the industry's needs and maintain the balance in the curriculum because in the process both the students' interest and the industry's requirements are satisfied.	Article 4
Contextualised curriculum.	Since most female students prefer to develop software that solves humanity's problems and enriches humanity's experiences, the SD curriculum and especially practical assignments and projects must be contextualised to appeal to the female students so that also they can find more relevance in SD education.	Article 1
B. Teaching and learning		
Project-based learning.	The students in this study view Project work as important and therefore project-based learning can be utilised. Not only do these projects of simulating the workplace provide them with the opportunity to solve problems and apply their theoretical knowledge but it engages students in reflecting on their future career from their early study years.	Article 1
Work-integrated learning.	The software industry can be helpful in accommodating students to gain experience in the workplace during their studies and mentoring them once they have entered the workplace. Students should gain experience in the industry much earlier in their education to eliminate possible misconceptions regarding their future career.	Articles 2, 4
Pair learning.	Pair learning (the use of pair programming in education) has been found to boost self-confidence and can therefore aid in teaching the students who did not have IT as school subject (Bishop-Clark *et al*., 2006, Salleh *et al.,* 2014). In addition, pair learning has been found to benefit female students and to improve academic performance (Werner *et al*., 2004; McDowell *et al*., 2003; Plonka *et al.*, 2015).	Article 1

Recommendation	Discussion	See
Practical assignments.	Practical assignments play an important role in the education of students and students should frequently receive challenging practical assignments and projects during their study years.	Article 2
Game-based learning.	Since students show such a great interest in Game development, computing departments at universities could explore game-based learning environments to satisfy students' appetite for games.	Article 1
Use experienced students as facilitators.	Students often learn more from each other than from lecturers, and the students who had IT as school subject could be appointed as facilitators in the practical labs and assist the less experienced students, as well as the students performing poorly.	Article 1
Variety of delivery mechanisms.	One of the respondents stated "…*MOOC's. They draw the masses and I think it forces universities to change their approach towards education".* Educators need to adjust from the traditional lecture format because computing classes need to be delivered with a variety of methods and to a variety of students.	Article 3
C. Collaboration		
Visits by lecturers to the software industry.	It is paramount for lecturers to gain and maintain experience in the software industry through for instance a weekly "industry-day" in order for them to aid eliminating misconceptions and not to be contributing to them.	Articles 1, 2, 4
Guest lectures by industry experts.	People from industry can be brought into SD classes.	Articles 1, 2
Joint university-industry projects.	Students should gain experience in the industry much earlier in their education to eliminate possible misconceptions regarding their future career.	Articles 1, 4
Joint university-industry curriculum.	The university and the IT industry should work together in creating up to date curriculums.	Articles 2, 3
D. Attract and retain		
Provide career guidance.	Students need to be informed from their early study years about the career possibilities and needs of the software industry. Universities, in conjunction with the industry, should provide career guidance to	Articles 1, 2, 4

Recommendation	Discussion	See
	students, so that they do not apply for and end up in a job that clashes with their personality type.	
Inform students about needs of the industry.	When students are aware of the specific needs of industry from their early study years, they might perceive the industry in a more positive light.	Article 1
Provide access to a diverse range of students.	SD education must be made accessible to a diverse range of students including minority groups.	Article 3
Offer bridging courses.	Lecturers must design and offer a bridging course early in students' first year, to provide students who did not have IT as school subject, with an initial positive experience of SD.	Article 1
Attention to: females; poor performers; no school IT.	In the light of the shortage of software developers, educators should pay special attention to female students, students with poor academic performance and students who did not have IT as school subject in an attempt to retain these students by providing a positive experience of SD.	Article 1
Generational awareness.	SD students, as well as new recruits or Millennials must be aware of the values and beliefs of the older generations. On the other hand the older software developers (Generation X) must be aware of the traits, values and beliefs of the Millennials in order to get the most out of their new recruits	Article 2
E. Real world projects		
Give assignments with a real-life context.	The tools, techniques, knowledge, and practical experience of real-world software projects need to be taught.	Article 2
Expose students to real world projects.	The university and industry must work together to provide 'real-world' group projects from students' first year of study. Not only do these projects of simulating the workplace provide them with the opportunity to solve problems and apply their theoretical knowledge but it engages students in reflecting on their future career from their early study years.	Article 3, 4
F. Life-long learning		
Appoint mentors in the workplace.	The software industry can be helpful in accommodating students to gain experience in the workplace during their studies and mentoring them once they have entered the workplace.	Article 2

Recommendation	Discussion	See
Provision for 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 on the forefront of the latest developments in the SD field.	Article 3
Keep educators updated.	Mechanisms must be put in place so that educators stay on the forefront of the latest developments in the SD field.	Article 4

When referring back to the definition of a framework: a platform with interlinked items was created that supports the education of software development students with an objective to make SD education relevant from the perspective of both students and the software industry. The framework can serve as a guide for all the stakeholders in SD education ranging from students and universities (lecturers, developers of degree programmes/curricula) to industry (corporate trainers, management, employers). The framework can streamline the education process to be relevant and the stakeholders can modify the framework to fit their perspective by adding or deleting items.

7.3 LIMITATIONS OF THE STUDY

A few limitations were identified in the empirical study. These limitations were:

In this study the focus was on SD professionals and SD students. All the students fitted the label "software development student", since they attended a software development course, but not all of them will end up pursuing a career in software development. However, the students' future careers cannot be predicted and their views and opinions can be of great value.

In the questionnaire to the SD professionals the focus was on software developers, and all 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 sector, service industries, etc.) and there would be many different needs to address.

The fact that all the survey respondents were from a single university can raise the question of the applicability of the study. However, the university's curricula are based on the IFIP framework that is linked to major, widely-accepted and widely-implemented curricular efforts and to high

quality bodies of knowledge undertaken by professional organizations, such as ACM, IEEE, AITP and AIS.

An instrument that measures the relevance of SD education for students could not be found. The researchers adjusted surveys for measuring relevance to the industry and surveys measuring relevance to school learners and sent their finding to industrial and academic experts familiar with tertiary computing programs to make provision for validity. However, additional validation is recommended.

The students, especially the first years, might not have been familiar with all the software development topics presented in the questionnaire. This might have resulted in them not showing interest in a certain topic for the simple reason that they do not know what that topic entails.

The respondents of the SD professionals' questionnaire had a low representation of Baby Boomers (born 1940-1960). Although the views and opinions of the older software developers are important, it is not clear whether the Boomers' representation in the study might be a true reflection of their representation in the workplace.

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 can therefore not be used as a remedy to fix a single degree programme, but as a guideline for role players in SD ranging from lecturers, developers of degree programmes/curricula to corporate trainers.

7.4 RECOMMENDATIONS FOR FURTHER RESEARCH

- A study to design a standardised instrument to specifically address the relevance of software development education for students.
- The study involving the students could be extended to include qualitative data gathered through interviews or focus groups.
- A similar study can be done on students from various universities and on students that show a greater probability of being software developers in their future career.
- A similar investigation can be done but on SD professionals from various countries.
- A study to determine the specific Mathematics knowledge and skills requirements of the software industry.

BIBLIOGRAPHY

AAUW (American Association of University Women) educational foundation: commission on technology, gender and teacher education. 2000. Tech-Savvy: educating girls in the new computer age. Washington, WA.

Aasheim, C., Li, L. & Williams, S. 2009. Knowledge and skill requirements for entry-level information technology workers: a comparison of industry and academia. *Journal of information systems education*, 20(3):349-356.

Ackerman, P.L., Beier, M.E. & Bowen, K.R. 2002. What we really know about our abilities and our knowledge. *Personality and individual differences,* 33(4):587–605.

Acts (See South Africa)

Acuña, S.T. & Juristo, N. 2004. Assigning people to roles in software projects. *Software: Practice and experience*, 34(7):675-696.

Akbulut, A.Y. & Looney, C.A. 2007. Inspiring students to pursue computing degrees. *Communications of the ACM*, 50(10):67–71.

Alexander, P., Holmner, M., Lotriet, H., Matthee, M., Pieterse, H., Naidoo, S., Twinomurinzi, H. & Jordaan, D. 2011. Factors affecting career choice: comparison between students from computer and other disciplines. *Journal of science education and technology*, 20(3):300-315.

Augustine, R.K. 2001. Thanks kiddo! A survival guide for professional generation Xers. *Organization development journal*, 19(2):19-26.

Bailey, J. & Stefaniak, G. 2001. Industry perceptions of the knowledge, skills, and abilities needed by computer programmers. (*In* Serva, M., *ed*. Proceedings of the 2001 ACM SIGCPR conference on computer personnel research (SIGCPR '01), ACM, New York, NY, USA. p. 93-99).

Barnes, K., Marateo, R. & Pixy Ferris, S. 2007. Teaching and learning with the net generation. *Innovate: Journal of online education*, 3(4).

Bass, J. & 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*, 48(5):1-39.

Bateman, K. 2013. The irony of an unemployment problem and an IT skills shortage within the IT industry. *ComputerWeekly.com*, 29 Oct. http://www.computerweekly.com/blogs/itworks/2013/10/the-irony-of-an-unemployment-p.html Date of access: 1 Nov. 2013.

Bayne, S. & Ross, J. 2007. The 'digital native'and 'digital immigrant': a dangerous opposition. Annual Conference of the Society for Research into Higher Education (SRHE), Brighton, UK, 11–13 December 2007.

Bayne, S. & Ross, J. 2011. 'Digital Native' and 'Digital Immigrant' Discourses. (*In* Land, R. & Bayne, S., *eds*. Digital difference: Perspectives on online learning. Rotterdam: Sense Publishers, p. 159-169).

Benamati, J. & Mahaney, R.C. 2007. Current and future entry-level IT workforce needs in organizations. (*In* Proceedings of the 2007 ACM SIGMIS CPR conference on computer personnel research: The global information technology workforce (SIGMIS CPR '07). ACM, New York, p. 101-104).

Bennett, S.J., Maton, K.A. & Kervin, L.K. 2008. The 'digital natives' debate: a critical review of the evidence. *British journal of educational technology*, 39(5):775-786.

Bishop-Clark, C., Courte, J. & Howard, E. 2006. Programming in pairs with Alice to improve confidence, enjoyment, and achievement. *Journal of educational computing research*, 34(2):213–228.

Biztech Africa. 2013. Africa's high end ICT skills shortfall grows. 25 Nov. http://www.biztechafrica.com/article/africas-high-end-ict-skills-shortfall-grows/7302/#.UqwYi_SnqSA Date of access: 21 Jan. 2014.

Blum, L. & Frieze, C. 2005. In a more balanced computer science environment, similarity is the difference and computer science is the winner. *Computing research news*, 17(3):2, May.

Bosch, C. 2015. Software developers a rare commodity. ITWeb. http://www.itweb.co.za/index.php?option=com_content&view=article&id=141230:Software-developers-a-rare-commodity&catid=250 Date of access: 20 Feb. 2015.

Bothe, K., Budimac, Z., Cortazar, R., Ivanović, M. & Zedan, H. 2009. Development of a modern curriculum in software engineering at master level across countries. *Computer science and information systems*, 6(1):1-21.

Bovée, C., Voogt, J. & Meelissen, M. 2007. Computer attitudes of primary and secondary students in South Africa. *Computers in human behavior*, 23:1762–1776.

Bullen, C., Abraham, T., Gallagher, K., Simon, J.C. & Zwieg, P. 2009. IT workforce trends: implications for curriculum and hiring. *Communications of the association for information systems*, 24:129-140.

Bunton, T.E. & Brewer, J.L. 2012. Discovering workplace motivators for the millennial generation of IT employees. (*In* Proceedings of the 1st Annual conference on Research in information technology (RIIT '12). ACM, New York, NY, p. 13-18).

BusinessDictionary.com. 2011. Framework. http://www.businessdictionary.com/definition/framework.html Date of access: 1 Nov. 2014.

Calitz, A.P. 2010. A model for the alignment of ICT education with business ICT skills requirements. Port Elizabeth: NMMU. (Thesis - DBA).

CareerJunction Index. 2014. Executive Summary. Jul. http:// www.cji.co.za Date of access: 17 Aug. 2014.

Carlson, S. 2006. Colleges work to attract and support women in technology majors. *The Chronicle: Information Technology*, 52(19):235. http://chronicle.com Date of access: 5 Jun. 2006.

Cator, K. 2013. Education is having its Internet moment. MOOCs. LinkedIn. 8 Aug. http://www.linkedin.com/today/post/article/20130808192147-2906843-education-is-having-its-internet-moment Date of access: 12 Aug. 2013.

Cheese, P. 2008. Netting the net generation. *Businessweek.com*. 13 Mar. http://www.businessweek.com/stories/2008-03-13/netting-the-net-generationbusinessweek-business-news-stock-market-and-financial-advice Date of access: 25 Jul. 2013.

Chou, C. & Tsai, M. 2007. Gender differences in Taiwan high school students' computer game playing. *Computers in human behavior*, 23(1):812–824.

Clancy, M., Stasko, J., Guzdial, M., Fincher, S. & Dale, N. 2001. Models and areas for CS education research. *Computer science education*, 11(12):323-341.

Clark, M., Zukas, M. & Lent, N. 2011. Becoming an IT Person: field, habitus and capital in the transition from university to work. *Vocations and learning*, 4(2):133-150.

Clayton-Pedersen, A. & O'Neill, N. 2005. Curricula designed to meet 21st-century expectations (*In* Oblinger, D. & Oblinger, J., eds. Educating the net generation. Louisville, CO: Educause. p. 6.1-6.15).

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/ Date of access: 31 Oct. 2013.

Courte, J. & Bishop-Clark, C. 2009. Do students differentiate between computing disciplines? (*In* Paper presented at the SIGCSE '09, March 3–7, 2009, Chattanooga, Tennessee, USA)

Creswell, J. W. 2005. Educational research: Planning, conducting, and evaluating quantitative and qualitative research. 2nd ed. Upper Saddle River, NJ: Merrill.

Creswell, J.W. & Clark, V.L.P. 2007. Designing and conducting mixed methods research. Thousand Oaks, CA: Sage Publications.

Cushing, J., Bryant, R., Orr, G., Spengler, S., Tuttle, S. & Yasuhara, K. 2006. NSF'S Integrative Computing Education and Research (ICER) initiative: whither the northwest. *Journal of computing sciences in colleges,* 22(2):49-51.

Department of Education ***see*** South Africa. Department of Education.

Department of Home Affairs ***see*** South Africa. Department of Home Affairs.

Department of Labor ***see*** United States. Department of Labor.

Dolezalek, H. 2007. Generational Series, Part 2: XY Vision. MANAGESmarter. http://www.strategichrinc.com/articles/article-managesmart060807.pdf Date of access 26 Mar. 2015.

Eisner, S.P. 2005. Managing generation Y. *SAM Advanced management journal*, 70(4):4-15.

Ezer, J. 2006. India and the USA: A comparison through the lens of model IT curricula. *Journal of information technology education*, 5:429-440.

Feldt, R., Angelis, L., Torkar, R. & Samuelsson, M. 2010. Links between the personalities, views and attitudes of software engineers. Information and software technology, 52:611-624.

Fernandez-Sanz, L. 2009. Personal skills for computing professionals. *Computer*, 42(10):110-112.

Frand, J. L. 2000. The information-age mindset. Changes in students and implications for higher education. *EDUCAUSE Review*, 35(5):15-24.

Friese, S. 2013. ATLAS. ti 7 user guide and reference. Berlin: Scientific Software Development GmbH. http://atlasti.com/wp-content/uploads/2014/05/atlasti_v7_manual_en_201409.pdf Date of access: 24 Jul. 2014.

Frieze, C. 2007. The critical role of culture and environment as determinants of women's participation in computer science. Pittsburgh: Carnegie Mellon University. (Thesis – D.Phil.)

Frolick, M.N., Chen, L. & Janz, B.D. 2005. Supply and demand of IS faculty: a longitudinal study. *Communications of the Association of Information Systems*, 15:674–695.

Fuggetta, A. 2012. 3+1 Challenges for the future of universities. *Journal of systems and software*, 85(10):2417-2424.

Gallagher, K.P., Kaiser, K.M., Simon, J.C., Beath, C.M. & Goles, T. 2010. The requisite variety of skills for IT professionals. *Communications of the ACM*, 53(6):144-148.

Galpin, V., Sanders, I., Turner, H & Venter, B. 2003. Gender and educational background and their effect on computer self-efficacy and perceptions. Technical Report TR-Wits-CS-2003-0, School of Computer Science, University of the Witwatersrand, Apr. http://www.cs.wits.ac.za/ Date of access: 27 Jul. 2008.

Galpin, V, Sanders, I & Chen, P. 2007. Learning styles and personality types of computer science students at a South African university. *ACM SIGCSE Bulletin*, 39(3):201-205.

Gates, B. 2005. The 3r's solution. http://www.gatesfoundation.org/Media-Center/Speeches/2005/02/Bill-Gates-2005-National-Education-Summit Date of access: 14 Nov. 2011.

Goode, J., Estrella, R. & Margolis, J. 2006. Lost in translation: Gender and high school computer science. (*In* Aspray, W. & Cohoon, J., *eds*. Women in IT: reasons on the under-representation. Cambridge, MA.: MIT Press. p. 89–114.)

Granger, M.J., Dick, G., Jacobson, C.M. & Van Slyke, C. 2007. Information systems enrolments: challenges and strategies. *Journal of information systems education*, 18(3):303–311.

Gruner, S. 2014. On the historical semantics of the notion of software architecture. *TD The journal for transdisciplinary research in Southern Africa*, 10(1):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*, 19(6):423-431.

Gupta, U. & Houtz, L. 2000. High school students' perceptions of information technology skills and careers. *Journal of industrial technology*, 16(4):2-8.

Gürer, D. & Camp, T. 2002. An ACM-W literature review on women in computing. *ACM SIGCSE bulletin*, 34(2):121–127.

Guzdial, M., Prey, J., Topi, H., Urban, J., Cassel. L. & Schneider, D. 2009. Future of Computing Education Summit, June 25-26, 2009. http://www.acm.org/education/future-of-computing-education-summit/FoCES_web.pdf Date of access: 1 Nov. 2013.

Hanson, W.E., Creswell, J.W., Clark, V.L.P., Petska, K.S. & Creswell, J.D. 2005. Mixed methods research designs in counseling psychology. *Journal of counseling psychology*, 52(2):224.

Harris, L. 2012. Mind the ICT skills gap. *Brainstorm*, 3 Sep. http://www.brainstormmag.co.za/index.php?option=com_content&view=article&id=4699:mind-the-ict-skills-gap Date of access: 24 Jul. 2013.

Hartman, J., Moskal, P. & Dziuban, C. 2005. Preparing the academy of today for the learner of tomorrow. (*In* Oblinger, D. & Oblinger, J., *eds*. Educating the net generation. Louisville, CO: Educause. p. 6.1-6.15).

Hassan, N.R. 2008. Courting multidisciplinary illiteracy. (*In* Proceedings of the Americas Conference on Information Systems (AMCIS 2008). Paper 245.) http://aisel.aisnet.org/amcis2008/245 Date of access: 11 Mar. 2015.

Helsper, E.J. & Eynon, R. 2010. Digital natives: where is the evidence? *British educational research journal*, 36(3):503-520.

Hill, C., Corbett, C. & St Rose, A. 2010. Why so few? Women in science, technology, engineering, and mathematics. American Association of University Women. Washington, DC: AAUW research report.

Holbrook, J. 2003. Rethink science education. *Asia-Pacific forum on science learning and teaching*, 4(2):1-13.

Holbrook, J. 2009. Meeting challenges to sustainable development through science and technology education. *Science education international*, 20(1):44-59.

Holley, J. 2008. Generation Y: understanding the trend and planning for the impact. (*In* 32nd Annual IEEE International Computer Software and Applications Conference. COMPSAC'08, p. 2-2).

Hoover, E. 2009. The millennial muddle: How stereotyping students became a thriving industry and a bundle of contradictions. *The chronicle of higher education*. 11 Oct. http://chronicle.com/article/The-Millennial-Muddle-How/48772/ Date of access: 24 Jul. 2013.

Howe, N. & Strauss, W. 2000. Millennials rising: The next greatest generation. New York: Vintage Books.

Howe, N. & Strauss, W. 2007. The next 20 years. *Harvard Business Review*, 85:41-52.

Huang, W.W., Greene, J. & Day, J. 2008. Outsourcing and the decrease of IS program enrollment. *Communications of the ACM*, 51(6):101–104.

ISO/IEC/IEEE 24765. 2010. 3.2758: Systems and software engineering - Vocabulary. ISO/IEC/IEEE, Geneva, Switzerland.

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.

Johnson, R.B. & Onwuegbuzie, A.J. 2004. Mixed methods research: a research paradigm whose time has come. *Educational researcher*, 33(7):14-26.

Johnson, R.B., Onwuegbuzie, A.J. & Turner, L.A. 2007. Toward a Definition of Mixed methods research. *Journal of mixed methods research*, 1(2):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, NY, USA.

Jones, C., Ramanau, R., Cross, S. & Healing, G. 2010. Net generation or Digital Natives: Is there a distinct new generation entering university? *Computers & Education*, 54(3):722–732.

Keil, M., Lee, H. & Deng, T. 2013. Understanding the most critical skills for managing IT projects: A Delphi study of IT project managers. *Information & management*, 50:398–414.

Kekelis, L.S., Ancheta, R.W & Heber, E. 2005. Hurdles in the pipeline: girls and technology careers. *Frontiers: a journal of women studies*, 26(1):99–109.

Keller, R.T., Julian, S.D. & Kedia, B.L. 1996. A multinational study of work climate, job satisfaction, and the productivity of R&D teams. *IEEE Transactions on Engineering Management*, 43(1):48-55.

Kennedy, G., Judd, T.S., Churchward, A., Gray, K. & Krause, K. 2008. First year students' experiences with technology: Are they really digital natives? *Australasian journal of educational technology*, 24(1):108-122.

Kim, Y., Hsu, J. & Stern, M. 2006. An update on the IS/IT skills gap. *Journal of information systems education*, 17(4):395-402.

Kitchenham, B., Budgen, D., Brereton, P. & Woodall, P. 2005. An investigation of software engineering curricula. *Journal of systems and software*, 74(3):325-335.

Klawe, M. 2001. Refreshing the nerds. *Communications of the ACM*, 44(7):67-68.

Kochanski, J. & Ledford, G. 2001. "How to keep me" - retaining technical professionals. *Research technology management*, 44(3):31.

Krakovsky, M. 2010. Degrees, distance, and dollars. *Communications of the ACM*, 53(9):18-19.

Krause, K. 2007. Who is the e-generation and how are they faring in higher education? (*In* Lockard, J. & Pegrum, M., *eds.* Brave new classrooms: Democratic education and the Internet. New York: Peter Lang. p. 125–139).

Labaree, D.F. 2008. Comments on Bulterman-Bos: the dysfunctional pursuit of relevance in education research. *Educational researcher*, 37(7):421-423.

Lee, C. & Han, H. 2008. Analysis of skills requirement for entry-level programmer/analysts in Fortune 500 corporations. *Journal of information systems education*, 19(1):17-27.

Lee, S., Koh, S., Yen, D. & Tang, H. 2002. Perception gaps between IS academics and IS practitioners: An exploratory study. *Information Management*, 40(1):51–61.

Lee, S., Yen, D., Koh, S. & Havelka, D. 2001. Evolution of IS professionals' competency: an exploratory study. *Journal of computer information systems*, 41(4):21-30.

Lethbridge, T.C. 2000. Priorities for the education and training of software engineers. *Journal of systems and software*, 53(1):53-71.

Lethbridge, T., Diaz-Herrera, J., LeBlanc, R. & Thompson, J.B. 2007. Improving software practice through education: Challenges and future trends. (*In* 2007 Future of Software Engineering. IEEE Computer Society, Washington, DC. p. 12-28).

Loftus, C., Thomas, L. & 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, NY. p.105-110).

Long, S.A. 2005. Digital natives: If you aren't one, get to know one. *New library world*, 106(1210/1211): 187–189.

Lorenzo, G., Oblinger, D. & Dziuban, C. 2007. How choice, co-creation, and culture are changing what it means to be net savvy. *Educause quarterly*, 30(1):6-12.

Lunt, B., Ekstrom, J., Gorka, S., Hislop, G., Kamali, R., Lawson, E., LeBlanc, R., Miller, J. & Reichgelt, H. 2008. Curriculum guidelines for undergraduate degree programs in information technology. ftp://jaran.undip.ac.id/pustaka/kurikulum-acm/IT2008%20Curriculum_2.pdf Date of access: 10 Sep. 2014.

Macon, M. & Artley, J.B. 2009. Can't we all just get along? A review of the challenges and opportunities in a multigenerational workforce. *International journal of business research*, 9(6):90-94.

Margolis, J. & Fisher, A. 2002. Unlocking the Clubhouse: Women in Computing. Cambridge, MA: MIT Press.

Mawson, N. 2010. ICT skills shortage to cost SA. ITWeb. http://www.itweb.co.za/index.php?option=com_content&view=article&id=29992 Date of access: 24 Jul. 2014.

McAllister, N. 2012. Here's how to solve America's developer shortage. Infoworld, http://www.infoworld.com/d/application-development/heres-how-solve-americas-developer-shortage-185042 Date of access: 12 May 2013.

McDowell, C., Werner, L., Bullock, H. & Fernald, J. 2003. The impact of pair programming on student performance, perception and persistence. (*In* Proceedings of the International Conference on Software Engineering, p. 602-607).

McGettrick, A., Boyle, R., Ibbett, R., Lloyd, J., Lovegrove, G. & Mander, K. 2004. Grand challenges in computing education. British Computer Society, Swindon, United Kingdom.

McGettrick, A. 2009. Computing education matters. *Communications of the ACM*, 52(4):5.

McKinnon, S. & McCrae, J. 2012. Closing the gap: preparing computing students for employment through embedding work-related learning in the taught curriculum. *Industry and higher education*, 26(4):317-322.

McMillan, J.H. 2000. Educational research: fundamentals for the consumer. 3rd ed. USA: Addison Wesley Longman.

Mead, N. 2009. Software engineering education: How far we've come and how far we have to go. *Journal of systems and software*, 82(4):571-575.

Merkofer, P. & Murphy, A. 2009. The e-skills landscape in South Africa. *Zeitschrift für politikberatung*, 2(4):685-695.

Merriam, S.B. 2009. Qualitative research. A guide to design and implementation. San Francisco, CA: Jossey-Bass.

Merriam-Webster Dictionary. 2013. Relevance. www.merriamwebster.com/dictionary/relevance Date of access: 26 Aug. 2013.

Mirjana, I., Zoran, P., Anja, S. & Zoran, B. 2011. The IT gender gap: Experience, motivation and differences in undergraduate studies of computer science. *Turkish online journal of distance education*, 12(2):170-186.

Mohan, A., Merle, D., Jackson, C., Lannin, J. & Nair, S. 2010. Professional skills in the engineering curriculum. *IEEE Transactions on education*, 53(4):562–571.

Mooketsi, B.E. & Chigona, W. 2014. Different shades of success: educator perception of government strategy on e-education in South Africa. *The electronic journal of information systems in developing countries*, 64(8):1-15.

Moreno, A., Sanchez-Segura, M., Medina-Dominguez, F. & Carvajal, L. 2012. Balancing software engineering education and industrial needs. *Journal of systems and software*, 85(7):1607-1620.

Morgan, D.L. 2007. Paradigms lost and pragmatism regained. Methodological implications of combining qualitative and quantitative methods. *Journal of mixed methods research*, 1(1):48-76.

Muianga, X., Hansson, H., Nilsson, A., Mondlane, A., Mutimucuio, I. & Guambe, A. 2013. ICT in education in Africa - myth or reality: a case study of Mozambican higher education institutions. *The African journal of information systems*, 5(3), Art. 5.

Mulder, F. & Van Weert, T. 2001. IFIP/UNESCO's informatics curriculum framework 2000 for higher education. *ACM SIGCSE Bulletin*, 33(4):75-83.

Murray, A. 2015. What millennials do and don't want from their employers. *Fortune.Com*, http://fortune.com/2015/03/05/millenials-best-companies/ Business Source Premier. Date of access: 30 Mar. 2015.

Nash, J. 2009. Computer skills of first-year students at a South African university. (*In* Proceedings of the 2009 Annual Conference of the Southern African Computer Lecturers' Association (SACLA '09). ACM, New York, NY, p. 88-92).

Oates, B.J. 2006. Researching information systems and computing. London: Sage.

Oblinger, D. 2003. Boomers, Gen-Xers & Millennials. Understanding the new students. *EDUCAUSE Review*, 38(4), 37-47.

Oblinger, D. & Oblinger, J. 2005. Educating the net generation. Louisville, CO: Educause. http://www.educause.edu/ir/library/pdf/pub7101.pdf Date of access: 30 Mar. 2015.

Oblinger, D. 2008. Emerging technologies for learning. *Becta*, 3:11-29.

O'Grady, M. 2012. Practical problem-based learning in computing education. *ACM Transactions on computing education*. 12(3), Article 10:1-16.

O'Neal, S. 1998. The phenomenon of total rewards. *ACA Journal*, 7:6-18.

Oxford English Dictionary. 2013. Relevance. http://oxforddictionaries.com/definition/english/relevant Date of access: 10 October 2011.

Plice, R.K. & Reinig, B.A. 2007. Aligning the information systems curriculum with the needs of industry and graduates. *Journal of computer information systems*, 48(1):22.

Plonka, L., Sharp, H., Van der Linden, J. & Dittrich, Y. 2015. Knowledge transfer in pair programming: An in-depth analysis, International journal of human-computer studies, 73:66-78.

Prensky, M. 2001. Digital natives, digital immigrants part 1. *On the horizon*, 9(5):1-6.

Prensky, M. & Berry, B.D. 2001. Do they really think differently? *On the horizon*, 9(6):1-9.

Prensky, M. 2009. H. Sapiens Digital: From digital immigrants and digital natives to digital wisdom. *Innovate: Journal of online education*, 5(3).

Reisman, S. 2004. Higher education's role in job training. *IT Professional*, 6(1):6-7.

Rumpel, S. & Medcof, J. 2006. Total rewards: good fit for tech workers. *Research technology management*, 49(5):27.

Sahami, M., Guzdial, M., McGettrick, A. & Roach, S. 2011. Setting the stage for computing curricula 2013: computer science--report from the ACM/IEEE-CS joint task force. (*In* Proceedings of the 42nd ACM technical symposium on computer science education, p. 161-162).

Salleh, N., Mendes, E. & Grundy, J. 2014. Investigating the effects of personality traits on pair programming in a higher education setting through a family of experiments. *Empirical Software Engineering*, 19(3):714-752.

Saiedian, H. 2009. Software engineering challenges of the “Net” generation. *Journal of systems and software*, 82(4):551-552.

SAQA (South African Qualifications Authority). 2000. The National Qualifications Framework and Curriculum Development http://www.saqa.org.za/docs/pol/2000/curriculum_dev.pdf Date of access: 16 Nov. 2011.

Schawbel, D. 2012. Millennials vs. Baby Boomers: who would you rather hire? *Time*, Mar. http://business.time.com/2012/03/29/millennials-vs-baby-boomers-who-would-yourather-hire/ Date of access: 31 Oct. 2013.

Schreiner, C. & Sjøberg, S. 2004. Sowing the seeds of ROSE. Background, rationale, questionnaire development and data collection for ROSE (The Relevance of Science Education) - a comparative study of students' views of science and science education. *Acta didactica*, 4:1-120.

Seymour, L., Hart, M., Haralambous, P., Natha, T. & Weng, C. 2005. Inclination of scholars to major in Information Systems or Computer Science. *South African computer journal*, 35:97–106.

Shackelford, R., McGettrick, A., Sloan, R., Topi, H., Davies, G., Kamali, R., Cross, J., Impagliazzo, J., LeBlanc, R. & Lunt, B. 2006. Computing curricula 2005: The overview report. *ACM SIGCSE Bulletin,* 38(1):456-457.

Shaw, M. 2000. Software engineering education: A roadmap. (*In* Proceedings of the conference on The future of software engineering (FOSE '00), p. 371-380).

Shaw, M., Herbsleb, J., Ozkaya, I. & Root, D. 2005. Deciding what to design: Closing a gap in software engineering education. (*In* Inverardi, P & Jazayeri, M., *eds*. Software engineering education in the modern age. ICSE 2005 Education Track, p. 28 – 58).

Shotick, J. & Stephens, P.R. 2006. Gender inequities of self-efficacy on task-specific computer applications in business. *Journal of education for business*, 81(5):269–273.

Smith, M. 2014. People management. The Google way of motivating employees. Entrepreneurial insights. http://www.entrepreneurial-insights.com/google-way-motivating-employees/ Date of access: 10 Nov. 2014.

Soni, S., Upadhyaya, M. & Kautish, P. 2011. Generational differences in work commitment of software professionals: myth or reality? *Abhigyan*, 28(4):30-42.

South Africa. Department of Home Affairs. 2014. Immigration Act (Act No. 13 of 2002). Skills or qualifications determined to be critical for the republic of South Africa in relation to an application for critical skills visa or permanent residence permit. *Government gazette,* 37716:12-25, 3 Jun.

South Africa. Department of Education. 2001. National plan for higher education. http:// www.polity.org.za/html/govdocs/misc/higheredu1.htm Date of access: 10 Mar. 2008.

South Africa. The Employment Equity Act (No. 55 of 1998). Pretoria: Government Printer.

Spicer, M. 2011. What does business expect from its incoming recruits? Do they meet our expectations? (*In* The international conference of the International Association for Cognitive Education in Southern Africa (IACESA), Keynote Address). http://www.businessleadership. org.za/gup/filez/Speeches_IACESA_Conference_17_February_2011.pdf Date of access: 31 Oct. 2013.

Steyn, A.G.W., Smit, C.F., du Toit, S.H.C. & Strasheim, C. 1999. Modern statistics in practice. Pretoria: Van Schaik.

Stokes, S.L. 2000. Recruiting and retaining IT talent: Attracting and keeping IT talent. *Information systems management*, 17:8-16.

Straub, D.W. 1989. Validating instruments in MIS research. *MIS quarterly*, 13:147-169.

Surakka, S. 2007. What subjects and skills are important for software developers? *Communications of the ACM*, 50:73-78.

Tashakkori, A. & Creswell J.W. 2007. The new era of mixed methods (editorial). *Journal of mixed methods research*, 1:3-7.

TechTerms.com. 2014. Software framework. http://www.techterms.com/definition/framework Date of access: 1 Nov. 2014.

Teddlie, C. & Tashakkori, A. 2003. Major issues and controversies in the use of mixed methods in the social and behavioral sciences. (*In* Tashakkori, A & Teddlie, C., *eds*. Handbook of mixed methods in social & behavioral research. Thousand Oaks, CA: Sage. p. 3-50).

Thompson, J. 2007. Is education 1.0 ready for web 2.0 students? *Innovate: Journal of online education*, 3(4).

Tillberg, H. & Cohoon, J. 2005. Attracting women to the CS major. *Frontiers: a journal of women studies*, 26(1):126-140.

Topi, H., Valacich, J.S., Wright, R.T., Kaiser, K., Nunamaker, J.F., Sipior, J.C. & de Vreede, G.J. 2010. IS 2010: Curriculum guidelines for undergraduate degree programs in Information Systems. *Communications of the Association for Information Systems,* 26, Art. 18.

Trauth, E.M., Quesenberry, J.L. & Huang, H. 2008. A multicultural analysis of factors influencing career choice for women in the information technology workforce. *Journal of global information management*, 16(4):1–23.

UN Broadband Commission. 2013. The state of broadband 2013: universalizing broadband. A report by the Broadband Commission. Sep. 2013. http://www.broadbandcommission.org/ documents/bb-annualreport2013.pdf

UN Broadband Commission. 2014. The state of broadband 2014: broadband for all. A report by the Broadband Commission. Sep. 2014. http://www.broadbandcommission. org/ documents/bb-annualreport2014.pdf

United States. Department of Labor: Bureau of Labor Statisics. 2014. Occupational outlook handbook. http://www.bls.gov/ooh/computer-and-information-technology/home.htm Date of access: 9 May 2014.

Venkatesh, V., Brown, S.A. & Bala, H. 2013. Bridging the qualitative-quantitative divide: guidelines for conducting mixed methods research in information systems. *MIS quarterly*, 37(1):21-54.

Wade, R. 2002. Bridging the digital divide: new route to development or new form of dependency. *Global governance*, 8(4):443-466.

Warschauer, M. 2007. A teacher's place in the digital divide. *Yearbook of the national society for the study of education*, 106(2):147–166.

Wells, T.D. & Sevilla, C. 2001. Forming a dialogue with academia: industry requirements versus academic programs. *Information systems management*, 18(1):80-83.

Werner, L., Hanks, B. & McDowell, C. 2004. Pair programming helps female computer science students. *ACM Journal on educational resources in computing*, 4(1), art 4.

Wohlin, C. & Regnell, B. 1999. Achieving industrial relevance in software engineering education. (*In* Proceedings of the 12th Conference on Software Engineering Education and Training, p. 16-25).

Wu, J.H., Chen, Y.C. & Chang, J. 2007. Critical IS professional activities and skills/knowledge: A perspective of IS managers. *Computers in human behavior*, 23(6):2945-2965.

Appendix A

Students' questionnaire

The relevance of software development education

We want to ask you a few questions about yourself. Your answers will be treated confidentially and the information will only be used to gain insights into student opinions and preferences as a group

A. Biographical information

(Give your answer with an X)

Gender?	
Male	
Female	

How would you classify your ethnic background?	
African/Black	
White	
Coloured	
Indian/Asian	
I don't wish to say	
Other (please specify)	

What is your year of birth?	

What <u>academic</u> year of study are you currently in?	

Are you currently an Undergraduate or Postgraduate student?	
Undergraduate	
Postgraduate	

Please grade your overall academic performance for this semester's software development modules on a scale from 1 - 10, where 10 represents 'excellent results', 5 stands for 'average'. If your overall academic performance is between:	
75% - 100%, your score = between 8 & 10; 70% - 74%, your score = 7; 60% - 69%, your score = 6; 50% - 59%, your score = 5; Below 50% = 4	

I Had IT as a school subject:	
Yes	
No	

I had access to a computer at home since Grade 1 (or earlier)	
Yes	
No	

B. My out-of-class experiences

How often have you done this outside formal education?
(Give your answer with an X on each line. If you do not understand, leave the line blank.)

I have …	**Never**	**Once or twice**	**I don't know**	**Quite often**	**Very often**
written a computer program					
developed a software system for somebody					
read about computers in books or magazines					
installed a program on a computer					
searched the internet for information					
communicated with friends and family via e-mail					
communicated with friends and family via skype					
played computer games					
used a dictionary, encyclopaedia, etc. on a computer					
downloaded music from the internet					
opened a device (radio, watch, computer, telephone, etc.) to find out how it works					
built a device (robot, radio, computer, etc.)					

C. My software development classes

To what extent do you agree with the following statements? (Give your answer with a X on each row. If you do not understand, leave the line blank.)	**Strongly disagree**	**Disagree**	**Neutral**	**Agree**	**Strongly agree**
I'm confident that I will obtain my degree					
I enjoy working with computers					
I am anxious/stressed when I do the practicals					
Software development is a difficult subject area					
Software development is interesting					
Software development is rather easy for me to learn					

Software development has opened my eyes to new and exciting jobs					
I like Software development better than most other subjects					
I think everybody should learn Software development					
My parents wish for me to become a software developer					
The things that I learn in Software development will be helpful in my everyday life					
I think that the Software development I learn will improve my career chances					
Our country need more software developers					
I would like to become a software developer					
The instruction in the Software development class is rigorous					
Some of the lecturers should have industrial experience					
I learn something new every day in the Software development classes					
This course has high expectations for all students					
Projects play an important role in the education of students					
The instruction in the Software development class is relevant					
People from industry should be brought into the classes					
The same staff are involved in teaching and research					
I know what the outcomes are for the Software development degree					
The volume of work in the Software development modules is high					
In the software development modules we learn to identify and solve problems using critical and creative thinking					
In the software development modules we learn to work with others as a member of a team or group					
The software development modules require from students to organise and manage themselves and their activities effectively					
In the software development modules we learn to collect and critically evaluate information					
In the software development modules we learn to communicate effectively, both verbally and in writing					
In the software development modules we learn to use science and technology effectively					

In the software development modules students must be able to demonstrate an understanding of the world as a set of related systems by recognising that problem-solving contexts do not exist in isolation.					
I knew what software developers do in the workplace, before I started my studies					
I know what software developers do in the workplace now that I am busy with my studies.					

D. My future job

To what extent do you agree with the following statements? (Give your answer with an X)

To be a good software developer you have to:	**Strongly disagree**	**Disagree**	**Neutral**	**Agree**	**Strongly agree**
have a good set of exam results					
be prepared to work hard and to learn (a thirst for knowledge)					
have a good attitude, including a willingness to listen and to take instructions					
have respect for others					
good time-management skills					
a neat and tidy appearance					
have a desire to make something of oneself through hard work and application, a desire to succeed (realistically ambitious)					
have at least some idea of what career direction one wish to take					
have modern leadership skills like self confidence and a preparedness to lead by example					
have the ability to relate well to and to build relationships with others (emotional intelligence)					
have a reasonable level of general knowledge					
have a preparedness to take responsibility					

E. What I want to learn about

How interested are you in learning about the following? *(Give your answer with an X on each row. If you do not understand, leave the line blank.)*		**Not interested**	**Slightly interested**	**Neutral**	**Quite interested**	**Very interested**
General software design	Data structures					
	Algorithm design					
	Software design and patterns					
	Software architecture					

	Object-oriented concepts and technology					
	Specific programming languages					
Software engineering methods	Requirements gathering and analysis					
	Formal specification methods					
	Analysis and design methods					
	Performance measurement and analysis					
	Testing, verification, and quality assurance					
	Software reliability and fault tolerance					
	Maintenance, reengineering, and reverse engineering					
Alternative software engineering methods	Agile methods					
	Extreme programming					
	Scrum					
	Web-based methods					
Software management	Project management					
	Software metrics					
	Software cost estimation					
	Configuration and release management					
	Process standards such as CMMI, ISO9000/3					
Essential subsystem design	Human-computer interaction/user interfaces					
	Databases					
	File management					
	Datawarehousing					
Specialized application techniques	Computational methods for numerical problems					
	Simulation					
	Artificial intelligence					
	Pattern recognition and image processing					
	Computer graphics					
	Parsing and compiler design					
	Information retrieval					
	Security and cryptography					
	Decision support systems					
	Expert systems					
Real-time and systems programming	Operating systems					
	Systems programming					
	Data transmission and networks					
	Parallel and distributed processing					
	Real-time system design					

Computer hardware	Digital electronics and digital logic					
	Microprocessor architecture					
	Computer system architecture					
	Network architecture and data transmission					
	Telecommunications					
Other electrical and computer engineering	Analog electronics					
	Digital signal processing					
	Data acquisition					
	Robotics					
Web	Interface design					
	Security and privacy					
	Web application development					
Mobile technologies	User interface design					
	Mobile application development					
	Security and privacy					
	Compatibility					
Computer science theory	Programming language theory					
	Formal languages					
	Computational complexity and algorithm analysis					
	Information theory					
Games	Game development					
Discrete mathematics						
Probability and statistics						
Linear algebra and matrices						
Continuous mathematics						
Other (please specify)						

Appendix B

Students' questionnaire: Items in each factor

Factor	Questionnaire items	Mean	SD
How often have you done this outside formal education?			
1 - Never / 2 - Once or twice / 3 - I don't know / 4 - Quite often / 5 - Very often			
	played computer games	4.52	0.92
	opened a device (radio, watch, computer, telephone, etc.) to find out how it works	3.97	1.29
Basic computer use	installed a program on a computer	4.7	0.74
	read about computers in books or magazines	3.78	1.27
	downloaded music from the internet	4.44	1.02
	used a dictionary, encyclopedia, etc. on a computer	4.36	0.99
E-mail use	communicated with friends and family via e-mail	4.54	0.88
Internet use	searched the internet for information	4.86	0.46
Skype use	communicated with friends and family via skype	3.33	1.63
	developed a software system for somebody	2.02	1.29
Advanced computer use	written a computer program	3.44	1.44
	built a device (robot, radio, computer, etc.)	2.62	1.55
To what extent do you agree with the following statements? ***1 - Strongly disagree / 2 - Disagree / 3 - Neutral / 4 - Agree / 5 - Strongly agree***			
In_class_Learn	In the software development modules we learn to collect and critically evaluate information	4.02	0.96
	In the software development modules we learn to communicate effectively, both verbally and in writing	4	0.92
	In the software development modules we learn to use science and technology effectively	4.01	0.91
	In the software development modules students must be able to demonstrate an understanding of the world as a set of related systems by recognising that problem-solving contexts do not exist in isolation.	3.98	0.9
	In the software development modules we learn to work with others as a member of a team or group	4.17	0.95
	The software development modules require from students to organise and manage themselves and their activities effectively	4.12	0.86
	In the software development modules we learn to identify and solve problems using critical and creative thinking	4.21	0.84
	Projects play an important role in the education of students	4.02	0.95

Factor	Questionnaire items	Mean	SD
In_class_Perceptions	I am anxious/stressed when I do the practicals	2.85	1.42
	Software development is a difficult subject area	2.97	1.33
	The volume of work in the Software development modules is high	3.08	1.22
	Software development is rather easy for me to learn	3.37	1.23
	The instruction in the Software development class is rigorous	3.36	1.02
	I'm confident that I will obtain my degree	2.8	1.52
In_class_Attitudes	I like Software development better than most other subjects	3.8	1.29
	I would like to become a software developer	3.7	1.37
	My parents wish for me to become a software developer	2.81	1.28
	Software development has opened my eyes to new and exciting jobs	3.97	1.06
	I enjoy working with computers	4.5	0.84
	Software development is interesting	4.27	0.94
In_class_Importance	I learn something new every day in the Software development classes	3.84	1.06
	I think that the Software development I learn will improve my career chances	4.28	0.86
	Our country need more software developers	4.21	0.88
	I think everybody should learn Software development	3.11	1.29
	The things that I learn in Software development will be helpful in my everyday life	3.95	0.98
	This course has high expectations for all students	3.95	0.97
In_class_Teaching	The same staff are involved in teaching and research	3.44	1.04
	People from industry should be brought into the classes	2.29	1.23
	I know what the outcomes are for the Software development degree	3.71	1.12
	I know what software developers do in the workplace now that I am busy with my studies.	3.84	1.16
	I knew what software developers do in the workplace, before I started my studies	3.15	1.33
	The instruction in the Software development class is relevant	3.95	0.93
	Some of the lecturers should have industrial experience	3.78	0.94
To be a good software developer you have to:			
Career_Attitudes	have a good attitude, including a willingness to listen and to take instructions	4.56	0.62
	be prepared to work hard and to learn (a thirst for knowledge)	4.53	0.65
	have a desire to make something of oneself through hard work and application, a desire to succeed (realistically ambitious)	4.53	0.68
	good time-management skills	4.33	0.89
	have a preparedness to take responsibility	4.6	0.63
	have a reasonable level of general knowledge	4.37	0.74
Career_Skills	a neat and tidy appearance	3.75	1.22
	have modern leadership skills like self-confidence and a preparedness to lead by example	4.14	0.88

Factor	Questionnaire items	Mean	SD
	have the ability to relate well to and to build relationships with others (emotional intelligence)	4.21	0.84
	have a good set of exam results	3.62	1.15
	have respect for others	4.45	0.79
	have at least some idea of what career direction one wish to take	4.27	0.85
How interested are you in learning about the following? **1 - Not interested / 2 - Slightly interested / 3 - Neutral / 4 - Quite interested / 5 - Very interested**			
Real-time and systems programming	Data transmission and networks	3.75	1.18
	Parallel and distributed processing	3.49	1.23
	Real-time system design	3.64	1.19
	Systems programming	3.83	1.22
	Operating systems	3.84	1.2
Mathematics and statistics	Continuous mathematics	3.25	1.38
	Linear algebra and matrices	3.33	1.35
	Discrete mathematics	3.24	1.33
	Probability and statistics	3.31	1.33
	Computational methods for numerical problems	3.58	1.26
Software management	Configuration and release management	3.37	1.21
	Software cost estimation	3.35	1.24
	Process standards such as CMMI, ISO9000/3	3.21	1.28
	Software metrics	3.41	1.18
	Project management	3.78	1.23
General software design	Software design and patterns	3.8	1.2
	Algorithm design	3.59	1.31
	Software architecture	3.76	1.21
	Object-oriented concepts and technology	3.86	1.19
	Data structures	3.57	1.23
	Specific programming languages	3.91	1.25
Web and Mobile technologies + Games	Web application development	3.84	1.29
	User interface design	3.82	1.23
	Interface design	3.82	1.28
	Mobile application development	3.8	1.24
	Mobile technologies - Security and privacy	3.72	1.24
	Compatibility	3.69	1.21
	Web - Security and privacy	3.85	1.26
	Game development	3.92	1.36
	Web-based methods	3.65	1.29
	Human-computer interaction/user interfaces	4.08	1.08
Computer science theory	Formal languages	3.25	1.3
	Computational complexity and algorithm analysis	3.21	1.32

Factor	Questionnaire items	Mean	SD
	Information theory	3.15	1.34
	Programming language theory	3.17	1.35
Specialized application techniques	Decision support systems	3.57	1.2
	Information retrieval	3.57	1.21
	Security and cryptography	3.81	1.22
	Pattern recognition and image processing	3.75	1.22
	Computer graphics	4.03	1.13
	Parsing and compiler design	3.58	1.24
	Artificial intelligence	4	1.19
	Expert systems	3.71	1.24
	Simulation	3.74	1.19
Software engineering methods	Testing, verification, and quality assurance	3.68	1.17
	Analysis and design methods	3.58	1.13
	Requirements gathering and analysis	3.36	1.18
	Software reliability and fault tolerance	3.59	1.22
	Performance measurement and analysis	3.57	1.16
	Formal specification methods	3.35	1.12
	Maintenance, reengineering, and reverse engineering	3.49	1.33
Essential subsystem design	File management	3.69	1.16
	Data-warehousing	3.56	1.2
	Databases	3.81	1.12
Computer hardware and other electrical and computer engineering	Microprocessor architecture	3.56	1.26
	Computer system architecture	3.67	1.22
	Digital electronics and digital logic	3.66	1.22
	Network architecture and data transmission	3.68	1.21
	Digital signal processing	3.25	1.28
	Analog electronics	3.22	1.27
	Telecommunications	3.42	1.23
	Robotics	3.78	1.32
	Data acquisition	3.25	1.24
Alternative software engineering methods	Scrum	3.13	1.22
	Agile methods	3.27	1.24
	Extreme programming	3.51	1.4

Appendix C

Software professionals' questionnaire

Software Development Education

Welcome to the Software Development Education Survey!

This survey gives you an opportunity to impact the education of software developers. The data from this survey allows educators to understand what they need to offer students if they want to attract and retain the best talent and prepare them for their future career. And as you are most likely to be an employer or colleague of these software development students in the future, you now have a chance to tell educators what you would like. Responses are only used as part of an aggregate and anonymity and confidentiality are strictly protected.

Software Development Education

What is your gender?: *

☒ Male ☒ Female

How would you classify your ethnic background?: *

☒ African/Black ☒ White ☒ Coloured ☒ Indian/Asian ☒ Other

What is your age category?: *

☒ 18-24 ☒ 25-29 ☒ 30-39 ☒ 40-49 ☒ 50-59 ☒ 60+

Which one of the following best describe your education?: *

☒ 3-year B.Sc in IT/CS/IS degree

☒ 3-year B.Sc in IT/CS/IS degree & Honours

☒ 3-year B.Sc in IT/CS/IS degree & Honours & M.Sc

☒ 3-year B.Sc in IT/CS/IS degree & Honours & M.Sc & Ph.D

☒ 3-year B.Sc/B.Com/B.Ing/B.A. degree

☒ 3-year B.Sc/B.Com/B.Ing/B.A. degree & Honours

☒ 3-year B.Sc/B.Com/B.Ing/B.A. degree & Honours & Masters

☒ 3-year B.Sc/B.Com/B.Ing/B.A. degree & Honours & Masters & Ph.D

☒ Certification with Microsoft/Oracle

☒ Other (please specify) ☒

How many years' experience do you have in your current line of work?: *

☒ 0-4 ☒ 5-9 ☒ 10-14 ☒ 15-19 ☒ 20-29 ☒ 30-39 ☒ 40+

How long (in years) have you worked at your current employer?: *

☒ 0-2 ☒ 3-4 ☒ 5-6 ☒ 7-9 ☒ 10-14 ☒ 15-19 ☒ 20-29 ☒ 30-39 ☒ 40+

Are you involved in the hiring of new graduates?: *

☒ Yes ☒ No

Are you part of management?: *

☒ Yes ☒ No

Is software your company's primary product?: *

☒ Yes ☒ No

Software Development Education

To what extent do you agree with the following statements?: *	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
Students know what software developers do in the workplace.	☒	☒	☒	☒	☒
Lecturers should have industry experience.	☒	☒	☒	☒	☒
The university courses have high expectations of students.	☒	☒	☒	☒	☒
Projects play an important role in the education of students.	☒	☒	☒	☒	☒
The instruction in the Software development classes is relevant.	☒	☒	☒	☒	☒
People from industry should be brought into software development classes.	☒	☒	☒	☒	☒
I know what the outcomes are for a Software development degree.	☒	☒	☒	☒	☒
In the software development modules students should learn to work with others as a member of a team or group.	☒	☒	☒	☒	☒
The software development modules should require from students to organise and manage themselves effectively.	☒	☒	☒	☒	☒
In the software development modules students should learn to collect and critically evaluate information.	☒	☒	☒	☒	☒
In the software development modules students should learn to communicate effectively, both verbally and in writing.	☒	☒	☒	☒	☒
In the software development modules students should learn to use science and technology effectively.	☒	☒	☒	☒	☒
In the software development modules students must be able to demonstrate an understanding of the world as a set of related systems by recognising that problem-solving contexts do not exist in isolation.	☒	☒	☒	☒	☒
Our country needs more software developers.	☒	☒	☒	☒	☒
New recruits have the knowledge and skills to immediately contribute to the development of software.	☒	☒	☒	☒	☒
New recruits have to be trained/sent for training before they can contribute to the development of software.	☒	☒	☒	☒	☒

Software Development Education

How much did you learn about this in your formal education?

General software design: *	Learned nothing at all	Became vaguely familiar	Moderate working knowledge	Learned a lot	Learned in depth; became expert
Data structures	☒	☒	☒	☒	☒
Algorithm design	☒	☒	☒	☒	☒
Software design and patterns	☒	☒	☒	☒	☒
Software architecture	☒	☒	☒	☒	☒
Object-oriented concepts and technology	☒	☒	☒	☒	☒
Specific programming languages	☒	☒	☒	☒	☒

Software engineering methods: *	Learned nothing at all	Became vaguely familiar	Moderate working knowledge	Learned a lot	Learned in depth; became expert
Requirements gathering and analysis	☒	☒	☒	☒	☒
Formal specification methods	☒	☒	☒	☒	☒
Analysis and design methods	☒	☒	☒	☒	☒
Performance measurement and analysis	☒	☒	☒	☒	☒

General software design: *	Learned nothing at all	Became vaguely familiar	Moderate working knowledge	Learned a lot	Learned in depth; became expert
Testing, verification, and quality assurance	X	X	X	X	X
Software reliability and fault tolerance	X	X	X	X	X
Maintenance, reengineering, and reverse engineering	X	X	X	X	X
Agile methods	X	X	X	X	X
Extreme programming	X	X	X	X	X
Scrum	X	X	X	X	X
Web-based methods	X	X	X	X	X

Software management: *	Learned nothing at all	Became vaguely familiar	Moderate working knowledge	Learned a lot	Learned in depth; became expert
Project management	X	X	X	X	X
Software metrics	X	X	X	X	X
Software cost estimation	X	X	X	X	X
Configuration and release management	X	X	X	X	X

Essential subsystem design: *	Learned nothing at all	Became vaguely familiar	Moderate working knowledge	Learned a lot	Learned in depth; became expert
Human-computer interaction/user interfaces	X	X	X	X	X
Databases	X	X	X	X	X
Datawarehousing	X	X	X	X	X

Specialised application techniques: *	Learned nothing at all	Became vaguely familiar	Moderate working knowledge	Learned a lot	Learned in depth; became expert
Simulation	X	X	X	X	X
Artificial intelligence	X	X	X	X	X
Pattern recognition and image processing	X	X	X	X	X
Computer graphics	X	X	X	X	X
Parsing and compiler design	X	X	X	X	X
Information retrieval	X	X	X	X	X
Security and cryptography	X	X	X	X	X
Decision support systems	X	X	X	X	X
Expert systems	X	X	X	X	X

Real-time and systems programming: *	Learned nothing at all	Became vaguely familiar	Moderate working knowledge	Learned a lot	Learned in depth; became expert
Operating systems	X	X	X	X	X
Systems programming	X	X	X	X	X
Data transmission and networks	X	X	X	X	X

Real-time and systems programming: *	Learned nothing at all	Became vaguely familiar	Moderate working knowledge	Learned a lot	Learned in depth; became expert
Parallel and distributed processing	☒	☒	☒	☒	☒
Real-time system design	☒	☒	☒	☒	☒

Computer hardware: *	Learned nothing at all	Became vaguely familiar	Moderate working knowledge	Learned a lot	Learned in depth; became expert
Digital electronics and digital logic	☒	☒	☒	☒	☒
Microprocessor architecture	☒	☒	☒	☒	☒
Computer system architecture	☒	☒	☒	☒	☒
Network architecture and data transmission	☒	☒	☒	☒	☒
Telecommunications	☒	☒	☒	☒	☒

Other electrical and computer engineering: *	Learned nothing at all	Became vaguely familiar	Moderate working knowledge	Learned a lot	Learned in depth; became expert
Analog electronics	☒	☒	☒	☒	☒
Digital signal processing	☒	☒	☒	☒	☒
Data acquisition	☒	☒	☒	☒	☒
Robotics	☒	☒	☒	☒	☒

Web: *	Learned nothing at all	Became vaguely familiar	Moderate working knowledge	Learned a lot	Learned in depth; became expert
Interface design	☒	☒	☒	☒	☒
Security and privacy	☒	☒	☒	☒	☒
Web application development	☒	☒	☒	☒	☒

Mobile technologies: *	Learned nothing at all	Became vaguely familiar	Moderate working knowledge	Learned a lot	Learned in depth; became expert
User interface design	☒	☒	☒	☒	☒
Mobile application development	☒	☒	☒	☒	☒
Security and privacy	☒	☒	☒	☒	☒
Compatibility	☒	☒	☒	☒	☒

Computer science theory: *	Learned nothing at all	Became vaguely familiar	Moderate working knowledge	Learned a lot	Learned in depth; became expert
Programming language theory	☒	☒	☒	☒	☒
Formal languages	☒	☒	☒	☒	☒
Computational complexity and algorithm analysis	☒	☒	☒	☒	☒
Information theory	☒	☒	☒	☒	☒

Games:*	Learned nothing at all	Became vaguely familiar	Moderate working knowledge	Learned a lot	Learned in depth; became expert
Game development	X	X	X	X	X

Mathematics and statistics:*	Learned nothing at all	Became vaguely familiar	Moderate working knowledge	Learned a lot	Learned in depth; became expert
Discrete mathematics	X	X	X	X	X
Probability and statistics	X	X	X	X	X
Linear algebra and matrices	X	X	X	X	X
Continuous mathematics	X	X	X	X	X

Software Development Education

How important have the details of this specific material been to you in your career as a software developer?

General software design:*	No importance	Occasionally important	Moderately important	Very important	Essential
Data structures	X	X	X	X	X
Algorithm design	X	X	X	X	X
Software design and patterns	X	X	X	X	X
Software architecture	X	X	X	X	X
Object-oriented concepts and technology	X	X	X	X	X
Specific programming languages	X	X	X	X	X

Software engineering methods:*	No importance	Occasionally important	Moderately important	Very important	Essential
Requirements gathering and analysis	X	X	X	X	X
Formal specification methods	X	X	X	X	X
Analysis and design methods	X	X	X	X	X
Performance measurement and analysis	X	X	X	X	X
Testing, verification, and quality assurance	X	X	X	X	X
Software reliability and fault tolerance	X	X	X	X	X
Maintenance, reengineering, and reverse engineering	X	X	X	X	X
Agile methods	X	X	X	X	X
Extreme programming	X	X	X	X	X
Scrum	X	X	X	X	X
Web-based methods	X	X	X	X	X

Software management:*	No importance	Occasionally important	Moderately important	Very important	Essential
Project management	X	X	X	X	X
Software metrics	X	X	X	X	X

Software management: *	No importance	Occasionally important	Moderately important	Very important	Essential
Software cost estimation	X	X	X	X	X
Configuration and release management	X	X	X	X	X

Essential subsystem design: *	No importance	Occasionally important	Moderately important	Very important	Essential
Human-computer interaction/user interfaces	X	X	X	X	X
Databases	X	X	X	X	X
Datawarehousing	X	X	X	X	X

Specialised application techniques: *	No importance	Occasionally important	Moderately important	Very important	Essential
Simulation	X	X	X	X	X
Artificial intelligence	X	X	X	X	X
Pattern recognition and image processing	X	X	X	X	X
Computer graphics	X	X	X	X	X
Parsing and compiler design	X	X	X	X	X
Information retrieval	X	X	X	X	X
Security and cryptography	X	X	X	X	X
Decision support systems	X	X	X	X	X
Expert systems	X	X	X	X	X

Real-time and systems programming: *	No importance	Occasionally important	Moderately important	Very important	Essential
Operating systems	X	X	X	X	X
Systems programming	X	X	X	X	X
Data transmission and networks	X	X	X	X	X
Parallel and distributed processing	X	X	X	X	X
Real-time system design	X	X	X	X	X

Computer hardware: *	No importance	Occasionally important	Moderately important	Very important	Essential
Digital electronics and digital logic	X	X	X	X	X
Microprocessor architecture	X	X	X	X	X
Computer system architecture	X	X	X	X	X
Network architecture and data transmission	X	X	X	X	X
Telecommunications	X	X	X	X	X

Other electrical and computer engineering: *	No importance	Occasionally important	Moderately important	Very important	Essential
Analog electronics	X	X	X	X	X
Digital signal processing	X	X	X	X	X

Other electrical and computer engineering: *	No importance	Occasionally important	Moderately important	Very important	Essential
Data acquisition	☒	☒	☒	☒	☒
Robotics	☒	☒	☒	☒	☒

Web: *	No importance	Occasionally important	Moderately important	Very important	Essential
Interface design	☒	☒	☒	☒	☒
Security and privacy	☒	☒	☒	☒	☒
Web application development	☒	☒	☒	☒	☒

Mobile technologies: *	No importance	Occasionally important	Moderately important	Very important	Essential
User interface design	☒	☒	☒	☒	☒
Mobile application development	☒	☒	☒	☒	☒
Security and privacy	☒	☒	☒	☒	☒
Compatibility	☒	☒	☒	☒	☒

Computer science theory: *	No importance	Occasionally important	Moderately important	Very important	Essential
Programming language theory	☒	☒	☒	☒	☒
Formal languages	☒	☒	☒	☒	☒
Computational complexity and algorithm analysis	☒	☒	☒	☒	☒
Information theory	☒	☒	☒	☒	☒

Games: *	No importance	Occasionally important	Moderately important	Very important	Essential
Game development	☒	☒	☒	☒	☒

Mathematics and statistics: *	No importance	Occasionally important	Moderately important	Very important	Essential
Discrete mathematics	☒	☒	☒	☒	☒
Probability and statistics	☒	☒	☒	☒	☒
Linear algebra and matrices	☒	☒	☒	☒	☒
Continuous mathematics	☒	☒	☒	☒	☒

Software Development Education

To what extent do you agree with the following statements?

To be a good software developer you have to: *	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
have a good set of exam results	☒	☒	☒	☒	☒
be prepared to work hard	☒	☒	☒	☒	☒
be prepared to learn (a thirst for knowledge)	☒	☒	☒	☒	☒

To be a good software developer you have to: *	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
have a willingness to listen					
have a willingness to take instructions					
have respect for others					
good time-management skills					
a neat and tidy appearance					
have a desire to succeed (realistically ambitious)					
have at least some idea of what career direction one wish to take					
have modern leadership skills like self confidence and a preparedness to lead by example					
have the ability to relate well to and to build relationships with others (emotional intelligence)					
have a reasonable level of general knowledge					
have a preparedness to take responsibility					

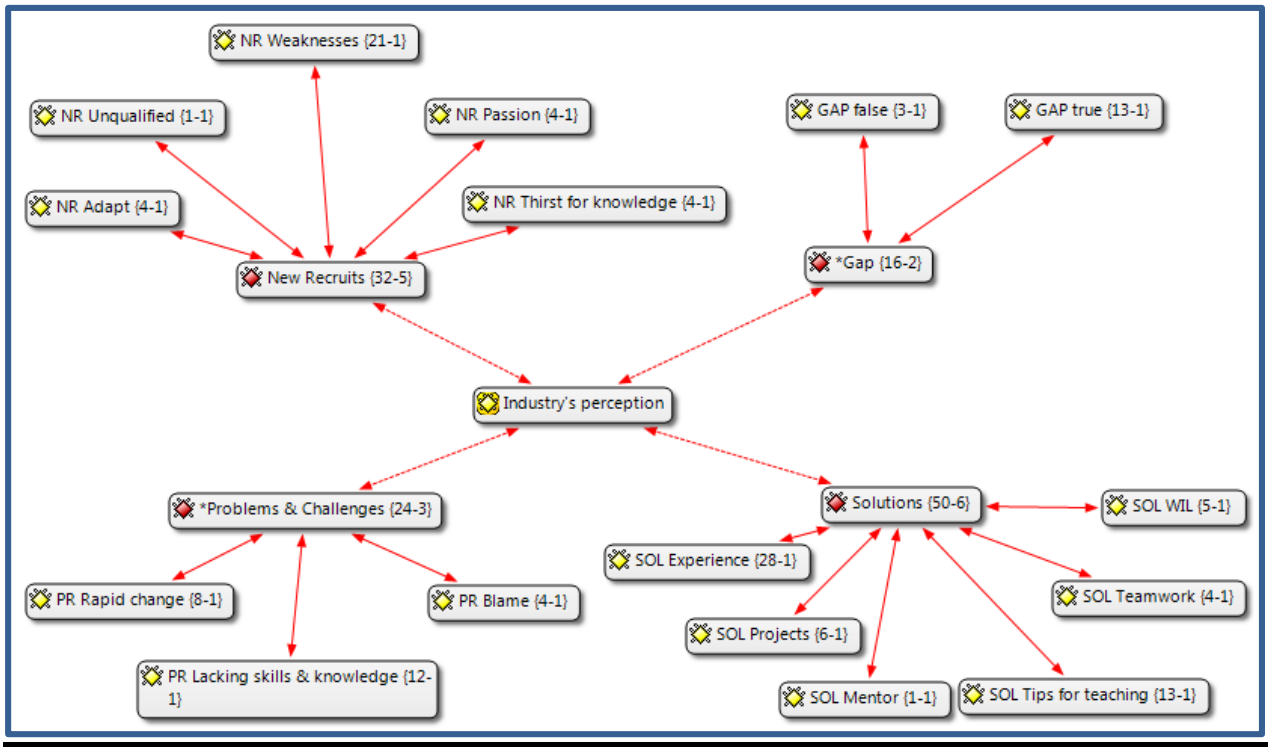
Do you have any further comments on the education of software developers?:

You submit this survey strictly anonymous, but we would like to know which companies are our research partners. Please fill in the name of your company/affiliation if you choose so.:

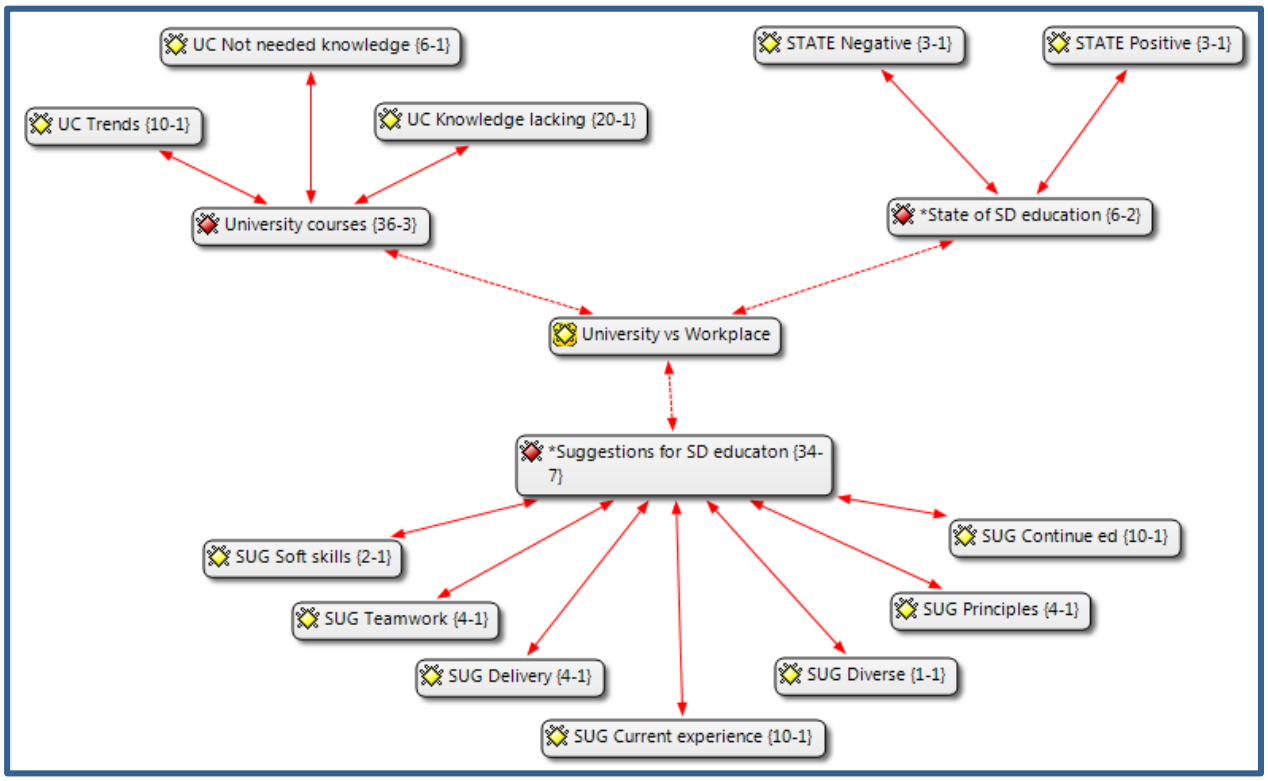
Appendix D

Networks of code families in Atlas.ti

Network for Article 2



Network for Article 3



Appendix E

Author guidelines

IEEE Transactions on Education

The “INFORMATION FOR AUTHORS” for *IEEE Transactions on Education* is presented on the next page.

See also the 52 page IEEE Editorial Style Manual at https://www.ieee.org/documents/style_manual.pdf

INFORMATION FOR AUTHORS

The aims of the IEEE TRANSACTIONS ON EDUCATION are both scientific and educational, grounded in the theory and practice of electrical and computer engineering. The scope covers education research, methods, materials, programs, and technology in electrical engineering, computer engineering, and fields within the scope of interest of IEEE. Manuscripts submitted to the TRANSACTIONS should clearly embrace one or more of these topic areas. The IEEE TRANSACTIONS ON EDUCATION is published quarterly with a distribution in excess of 3500 copies. Subscribers include engineering educators and libraries, both in university and industry.

A. Manuscript Submission

- All manuscripts must be submitted online at the IEEE TRANSACTIONS ON EDUCATION's Manuscript Central site, at http://mc.manuscriptcentral.com/te-ieee.
- Submission of material will be construed as indicating that it has not been copyrighted, published, submitted, or presented elsewhere, unless explicit notice to the contrary is given. All submissions undergo, and must pass, an originality check. Authors must complete an IEEE Copyright Transfer Form before publication and are requested to complete this upon submission.
- Two manuscript types are accepted, Regular or Special Issue. Special Issue papers are ONLY accepted in response to a published call for papers (to be found in the print version of the journal and at the IEEE Xplore site for the TRANSACTIONS, http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=13).
- Manuscripts are reviewed under one of three areas of scholarship: Discovery, Application, and Integration. Authors must select the appropriate area upon submission, and before preparing their manuscript should first read the information at http://sites.ieee.org/review-criteria-toe/

B. Manuscript Format

- Manuscripts must be submitted in the journal two-column format, as a Word or a PDF document.
- IEEE templates, guidelines, and Style Manual are available in the IEEE Author's Toolbox (http://www.ieee.org/publications_standards/publications/authors/authors_journals.html).
- Manuscripts should be no longer than six journal pages.
- Manuscripts must be written in the third person, avoiding the use of the first person (I, we, our).
- Manuscripts should use first-, second- and third-order headings, formatted as per the Style Manual.
- All acronyms, abbreviations, subscripts, superscripts, Greek letters, mathematical symbols, and any nonstandard symbols must be defined when first used in the text.
- Complete bibliographies, formatted according to the Style Manual, should follow the References.
- Manuscripts should be spell-checked in US English. (Hint: To change the spell check language, select the entire document, go to Language on the Tools drop-down menu, and select US English.)
- Manuscripts are expected to meet the highest standards of writing quality. Authors should absolutely consider having a native English speaker review the manuscript before submission.

C. Manuscript Abstract

The abstract must be a concise yet comprehensive reflection of what is in your article. In particular:

- The abstract must be self-contained, without abbreviations, footnotes, or references. It should be a microcosm of the full article.
- The abstract must be between 150–250 words. Be sure that you adhere to these limits; otherwise, you will need to edit your abstract accordingly.
- The abstract must be written as one paragraph, and should not contain displayed mathematical equations or tabular material.
- The abstract should include three or four different keywords or phrases, as this will help readers to find it. It is important to avoid over-repetition of such phrases as this can result in a page being rejected by search engines.
- Ensure that your abstract reads well and is grammatically correct.

D. Figures and Page Charges

- IEEE will automatically process color figures to appear in grayscale in the print edition of the journal, unless color printing is specifically requested. (Color printing will incur a charge of $1045.00 (setup) plus an additional $62.50 per color figure. Figures will appear in color on the online IEEE Xplore site free of charge.)
- Authors not requesting color printing should therefore make figures color-independent, and remove specific color references (e.g., "the blue line in the figure").
- If the printed length of the paper exceeds six pages, mandatory charges will be levied of $200 for each excess page.

E. Final Manuscript Files

- Final files for accepted manuscripts can be uploaded **either** in Word, TeX, LaTeX, or WordPerfect (use standard Macros). IEEE can process most commercial software, but not page layout programs. Do not send PostScript files.
- *If the source file is in Word and contains the figures, separate figure files are not required.*
- *Using Word will make the final editing process a great deal simpler for the author.*

F. Open Access

This publication is a hybrid journal, allowing either Traditional manuscript submission or Open Access (author-pays OA) manuscript submission. Upon submission, if you choose to have your manuscript be an Open Access article, you commit to pay the discounted $1,750 OA fee if your manuscript is accepted for publication in order to enable unrestricted public access. Any other application charges (such as over-length page charge and/or charge for the use of color in the print format) will be billed separately once the manuscript formatting is complete but prior to the publication. If you would like your manuscript to be a Traditional submission, your article will be available to qualified subscribers and purchasers via IEEE Xplore. No OA payment is required for Traditional submission.

Digital Object Identifier 10.1109/TE.2010.2089213

Appendix F

Author guidelines

TD The Journal for Transdisciplinary Research in Southern Africa

The “Editorial policy” for *TD The Journal for Transdisciplinary Research in Southern Africa* follows on the next two pages.

Editorial policy

1. TD is an international transdisciplinary journal for research in all fields of scientific endeavour. It is published and edited in the Vaal Triangle Faculty of North-West University in South Africa.
2. Contributions may be in the natural sciences or the humanities. Articles, in which transdisciplinary collaboration between natural and the social or human sciences are explored, are most welcome.
3. The term transdisciplinarity is meant to imply the integrated use of conventional scientific theory and methodology in an effort to explore quantum frontiers of new knowledge in all spheres of scientific endeavour.
4. Regionally editorial content can be based on empirical research in Southern Africa.
5. Authors can make individual contributions or submit work, done in teams.
6. TD is a peer reviewed journal. Contributions of authors will be subject to review by two or more reviewers in disciplines used in the research and writing of an article.
7. Language of the journal: Articles may be in any of the 11 official languages of South Africa. It could also be in any of the major international languages, e.g. French, Italian, Japanese, Dutch, German, Portuguese and Spanish.
8. A maximum of 30 per cent of the editorial content of each edition of the journal may be in a non-English language.
9. Abstracts: Contributions must be accompanied by an abstract of not more than 250 words in the language in which the article is written. Should the text not be in English, an abstract in English (250 words), as well as an executive summary of the article content (about 1 500 word) should accompany the article.
10. Titles of articles: The titles of articles should preferably not exceed 20 words.
11. Names of authors: The names of authors and their institutional affiliation must accompany all contributions. Authors also have to enclose their telephone and fax numbers, email addresses and postal addresses.
12. Reference system: The reference system of authors will be respected by the editorial management, providing it is current in the academic writing in the particular disciplines used to research and write the article. References must also be clear, lucid and comprehensible for a general academic audience of readers. The Harvard and APA reference styles are acceptable. The conventional footnote system of references may also be used.

13. Illustrations: Editorial material, with illustrations, photographs, tables and graphs is welcome. The illustrations should however be of a high-density quality. Should the files be large, they have to be posted in separate emails and appropriately numbered in sequence.

14. Articles should be posted to the editorial secretary electronically at johann.tempelhoff@nwu.ac.za. Notification of receipt of material will take place within 48 hours.

15. Text format: Text must be in 12pt text, with double spacing. Text should preferably be in Microsoft Word.

16. The length of articles should preferably not exceed 8 000 to 10 000 word or 15 to 20 journal pages.

17. Articles that have been published previously in other journals may not be republished in the journal.

Appendix G

Author guidelines

Industry & Higher Education

Submissions - Notes for authors

Please send submissions by e-mail to John Edmondson, Industry and Higher Education, IP Publishing Ltd, 4th Floor, Hamilton House, Mabledon Place, Bloomsbury, London WC1H 9BB, UK. jedmondson(a)ippublishing.com

Type and length of contributions

The major part of the journal is taken up by papers between 4,000 and 8,000 words long. These should be analytical and evaluative in approach and not simply descriptive. Other contributions include opinion or 'viewpoint' pieces (1,500-3,000 words); case studies of specific ventures or programmes (1,500-3,000 words); brief factual summaries of reports, agency programmes, educational institutions, etc (1,000-2,000 words); and letters to the editors.

Presentation

Submissions should be double-spaced. They can be sent either by e-mail to the **editor** or by post (in which case one hard copy and a disk or CD should be enclosed). Papers should preferably be sent in Word (please note that PDF versions are not acceptable for review purposes). The title page should contain full names of the authors, their professional status or affiliation and the address to which they wish correspondence to be sent. There should be an abstract of about 100 words at the beginning of the paper. The text should be organized under appropriate cross-headings and where possible these should not be more than 800 words apart.

Between 3 and 6 keywords should appear below the abstract, highlighting the main topics of the paper.

References should follow the Harvard system. That is, they should be shown within the text as the author's surname (or authors' surnames) followed by a comma and the year of publication, all in round brackets: for example, (Smith, 1998). For textual citations, where there are two authors please use the word 'and', not the ampersand (thus: '(Smith and Jones, 2012)'. Where there are more than two authors, please use the first-named author only, followed by 'et al' in italics (thus: Smith et al, 2012). At the end of the article a bibliographical list should be supplied, organized alphabetically by author (surnames followed by initials - all authors should be named). Bibliographic information should be given in the order indicated by the following examples:

Articles: Woollard, D. (2010), ‘Towards a theory of university entrepreneurship’, *Industry and Higher Education*, Vol 24, No 6, pp 413–427.

Books: Viale, R., and Etzkowitz, H., eds (2010), *The Capitalization of Knowledge*, Edward Elgar, Cheltenham.

Notes should be numbered consecutively in the text and typed in plain text at the end of the paper (not as footnotes on text pages).

Figures and tables should be presented separately on separate sheets at the end of the text. Each figure or table must be referred to in the text - the first reference will be used to locate the figure or table in the final printed version.

Prior Publication

Articles are received on the understanding that they are original contributions, and have not been published officially, either in print or electronic form, or submitted for publication elsewhere. In this respect, 'discussion' or 'working' papers, conference presentations and proceedings are not considered to be official publications, unless they have been formally deemed so by conference organizers, or presented as edited works through recognized publishing channels. If in doubt, authors are asked to draw the attention of the Editor to any prior dissemination of the paper in their letter of submission. Please note that articles should not be posted on personal Websites or social networking sites before or after submission.

Refereeing

Other than research notes, reports, and personal opinion pieces, articles will be refereed. Papers by authors who are not academics (eg submissions from industry) will also be subject to review before acceptance, but their distinct nature and aims will be fully taken into account.

Copyright

Wherever possible, authors are asked to assign copyright to IP Publishing Ltd. Relevant authors' rights are protected.

AUTHOR CHECKLIST FOR FINAL VERSIONS

Before sending us your final version, please be sure that the presentation of the paper conforms to the following basic guidelines. Please also refer to the submission guidelines at www.ippublishing.com.

□ The **text** should be double-spaced.

□ **Sections** should not be numbered: the text should be divided by subheadings, as detailed in our notes for authors.

□ The **title page** must include author names as they are to appear in the journal and full contact details (*full* mailing address, phone, fax, e-mail) for at least the corresponding author. For all non-corresponding authors, at minimum there should be an affiliation (department and institution or equivalent, city and country). Fuller details are not essential for non-corresponding authors, but will be welcome and will be reproduced in the published version.

□ **Tables and figures** should be presented on separate pages (using 'page breaks') at the end of the paper and not in the body of the text. We shall place them as near as possible to the first textual reference to them. Tables and figures must all have captions. Colour coding is best avoided in figures, as the print

edition of the journal is in black and white. If figures were originally prepared in Excel, we can often get better results from the Excel file than from the imported graphic in the Word document, so it is advisable to supply the Excel files as well in such cases.

□ **Tables** should be prepared using the Word 'insert Table' tool and not using the text tabs, which cause instability and alignment errors. They should also not be presented as graphics as this can cause editing problems.

□ **Numbering of tables and figures.** Tables and figures should be numbered separately (that is, Table 1 and Figure 1, not Table 1 and Figure 2). In all cases they should be numbered consecutively in the order in which they appear in the paper. Please note that maps, photos, charts etc should all be called figures (not Map 1, Photo 1, etc). There must be a textual reference to all figures and tables.

□ All **acronyms** should be spelt out in full the first time they are used, with the acronym in parentheses.

□ All references to the work or comments of others should be supported by full bibliographical citations in the reference list (please see our notes for authors for referencing style).

□ Please make sure that your paper has an **abstract** of around 100-150 words and between 3 and 6 keywords.

□ **The paper should be submitted in Word**. A **PDF** would also be appreciated so that we can use this as a reference copy in case of software glitches.

□ Please ensure that you have included any **acknowledgements** you may wish to make.

Appendix H

Author guidelines and proof of submission and review

Information Systems Management

Instructions for authors

SCHOLARONE MANUSCRIPTS™

This journal uses ScholarOne Manuscripts (previously Manuscript Central) to peer review manuscript submissions. Please read the guide for ScholarOne authors before making a submission. Complete guidelines for preparing and submitting your manuscript to this journal are provided below.

Aims and Scope. ***Information Systems Management (ISM)*** places a high value on article content that communicates advanced practices to address current IS management challenges and innovative applications of new, but proven, information technologies. The methodological orientation is towards field research informed by published literature, but primarily based on case study, surveys, and field experiences of IS experts. All prospective authors are encouraged to look at a recent ISM issue for writing style and format.

Please note that *Information Systems Management* uses CrossCheck™ software to screen papers for unoriginal material. By submitting your paper to *Information Systems Management* you are agreeing to any necessary originality checks your paper may have to undergo during the peer review and production processes.

Preparation of Manuscripts. Information and instructions are provided below for preparation of manuscripts, submission of manuscripts, and the journal review process. Please direct any questions to the Editorial Office at ism@villanova.edu ; however, please do not send manuscripts to this address. All manuscripts should be submitted electronically via the ScholarOne Manuscripts website located at http://mc.manuscriptcentral.com/UISM. ScholarOne Manuscripts allows for rapid and easy submission of original and revised manuscripts, as well as facilitating the review process and internal communication between authors, editors, and reviewers via a web-based platform. For ScholarOne Manuscripts technical support, contact them by e-mail or phone at http://scholarone.com/services/support.
A typical paper length is about 7500 words, exclusive of exhibits. The review process for ISM is double-blind; therefore, **PLEASE REMOVE AUTHOR NAME(S), THEIR BIOGRAPHICAL NOTES, OR ANY ACKNOWLEDGEMENTS FROM YOUR MANUSCRIPT FILE**. Do include a title, a short title of 50 characters or less, abstract, and keywords within your manuscript file. Short biographies, with contact information, for all authors should be submitted as a "Supplemental File" at the time of submission, along with an indication of the corresponding author.

Authors are responsible for obtaining permission to reproduce copyrighted material from other sources and must sign an agreement for transfer of copyright to the publisher upon acceptance. If you plan to use graphics or tables from another publication, you must secure reprint permission from the appropriate publisher. Such permission is also needed for quotes of 50 words or more, or more than 400 words of material quoted from one source. All accepted manuscripts, artwork, and photographs become the property of the publisher.

The journal only accepts submissions as electronic versions of MS-Word or Adobe PDF files. Please make sure that the text is double-spaced and in 12-point Times Roman font. Do *NOT* include page numbers, line numbers, or any information in the manuscript header or footer. Keep acronyms and abbreviations to a minimum and define those that do appear in the text. Manuscripts must use American spellings. The appropriate usage of em-dashes (a form of parenthetical punctuation) and en-dashes (for numerical ranges) is appreciated.

Footnotes are not typically used; if you need to include them, use the endnote format.

Figures, Tables, and Graphics. Figures, tables, and other graphics should each be in separate MS-Word or Adobe PDF files. All tables and figures must be mentioned in the text of a paper, and include a caption. All illustrations must be clear enough to be read when printed in black-and-white. Note that uploading of higher resolution images may take significantly longer; therefore authors are strongly encouraged to submit lower resolution images during the review process to streamline uploading.

Illustration Print Options. Illustrations submitted (line drawings, halftones, photos, photomicrographs, etc.) should be clean originals or digital files. Digital files are recommended for highest quality reproduction and should follow these guidelines:

- 300 dpi or higher
- sized to fit on journal page
- EPS, TIFF, or PSD format only
- submitted as separate files, not embedded in text files

Color Reproduction. Color illustrations will be considered for publication; however, the author will be required to bear the full cost involved in color art reproduction. Color art can be purchased for online only conversion and reproduction or for print + online reproduction. Color reprints can only be ordered if print + online reproduction costs are paid. Rates for color art reproduction are: Online Only Reproduction: $225 for the first page of color; $100 per page for each of the next three pages of color. A maximum charge of $525 applies. Print + Online Reproduction: $900 for the first page of color; $450 per page for the next three pages of color. A custom quote will be provided for articles with more than 4 pages of color.

References. References should appear at the end of the article in the latest edition of APA style. Cite in text the author and date, separated by a comma, in parentheses: (Batra, 2006). Use an ampersand, rather than "and" for multiple authors. For works with more than one author, provide all author names for the first citation in the text: (McCoy, Galletta, & King, 2007). Then use "et al." for subsequent citations of the same work: (McCoy et al., 2007).

Examples:

Journal Article: Gray, P. & Hovav, A.Z.(2007). The IS Organization of the Future: Four Scenarios for 2020.*Information Systems Management,* 24(2), 113–120. doi: 10.1080/10580530701220967

Book: Volonino, L., Anzaldua, R., & Godwin, J. (2007). *Computer Forensics: Principles and Practice.* Upper Saddle River, NJ: Prentice-Hall.
Book Chapter: Volonino, L. (2007). Security. In E. Turban, D. Leidner, E. McLean, & J. Wetherbe (Eds.)*Information Technology for Management: Transforming Organizations in the Digital Economy* (pp. 622–655). New York: John Wiley & Sons.

Conference Proceedings: Bullen, C.V., Goles, T., & Kaiser, K. (2006). The Impact of Sourcing on the IT Workforce Pipeline. In *Proceedings of the Twelfth Americas Conference on Information Systems,* pp. 3213–3221, Acapulco, Mexico, August 2006. Association for Information Systems.

Electronic Sources: Roberts, P. (2003, May 7). Earthlink wins $16 million in spam case. *PCWorld*. Retrieved October 25 2005 from PCWorld Web site: http://www.pcworld.com.

Manuscript Submission. All manuscripts should be submitted electronically via the Scholar One Manuscripts website located at http://mc.manuscriptcentral.com/uism. ScholarOne Manuscripts allows for rapid and easy submission of original and revised manuscripts, as well as reviewing and internal communication between authors, editors, and reviewers via a web-based platform.

To submit a manuscript, please follow the instructions below:
1. Launch your web browser (Internet Explorer 5 or higher or Netscape 6 or higher) and go to the ScholarOne Manuscript homepage (http://mc.manuscriptcentral.com/uism). Log-in, or click the "Create Account" option if you are a first-time user of ISM ScholarOne Manuscriptl. If you have a ScholarOne Manuscripts account for another journal, you will still need to create a new account for the ISM site.

2. Log-in and select "Author Center."

3. Click the “Submit a Manuscript” link in the menu bar. Enter data and answer questions as prompted. Click on the “Next” button on each screen to save your work and advance to the next screen. Be sure to recommend 3 to 5 preferred Reviewers by providing the name, email address, institution, department, and phone for each. Please designate 1 to 3 preferred Senior Editors by selecting each from the drop down list. Note that recommended Reviewers and Senior Editors will be used at the discretion of the Editor-in-Chief. You will be prompted to upload your files. Click on the “Browse” button and locate the file on your computer. Select the description of the file in the drop down next to the Browse button. Please use the file description “Supplemental File” when uploading your short author biography and contact information. When you have selected all files you wish to upload, click the “Upload” button.

Review your submission (in both PDF and HTML formats) before sending to the Editor-in- Chief. **PLEASE CHECK TO MAKE SURE THAT YOUR MANUSCRIPT SUBMISSION IS BLINDED.** All author name(s), their biographical notes, or any acknowledgments should NOT appear in your manuscript file. Do include a title, abstract, and keywords within your manuscript file. Note that ScholarOne Manuscripts will automatically assign each page a running head and footer, as well as page numbers and line numbers to your manuscript. Click the “Submit” button when you are done reviewing. Authors are responsible for verifying all files have uploaded correctly.

You may stop a submission at any phase and save it to submit later. After submission, the corresponding author will receive a confirmation via e-mail acknowledging receipt of the manuscript, including an assigned manuscript number that should be included in all subsequent correspondence. You can also log-on to ScholarOne Manuscripts any time to check the status of your manuscript. You will receive an e-mail once a decision has been made on your manuscript.

Revised manuscripts should be submitted via Scholar One Manuscripts Author Center to ensure that they are linked to the original submission.

Journal Review Process. All papers undergo an initial review by the journal’s Editor-in-Chief. After a positive initial review, papers undergo a double-blind peer review process by two or more subject matter experts. Reviews are received by the Editor-in-Chief, who then provides the author with the final recommendation: a conditional acceptance, a request for further revisions, or rejection.

Prior to publication, all accepted papers are edited for style and grammar by Taylor & Francis’s editors. The author is also requested to complete a copyright form.

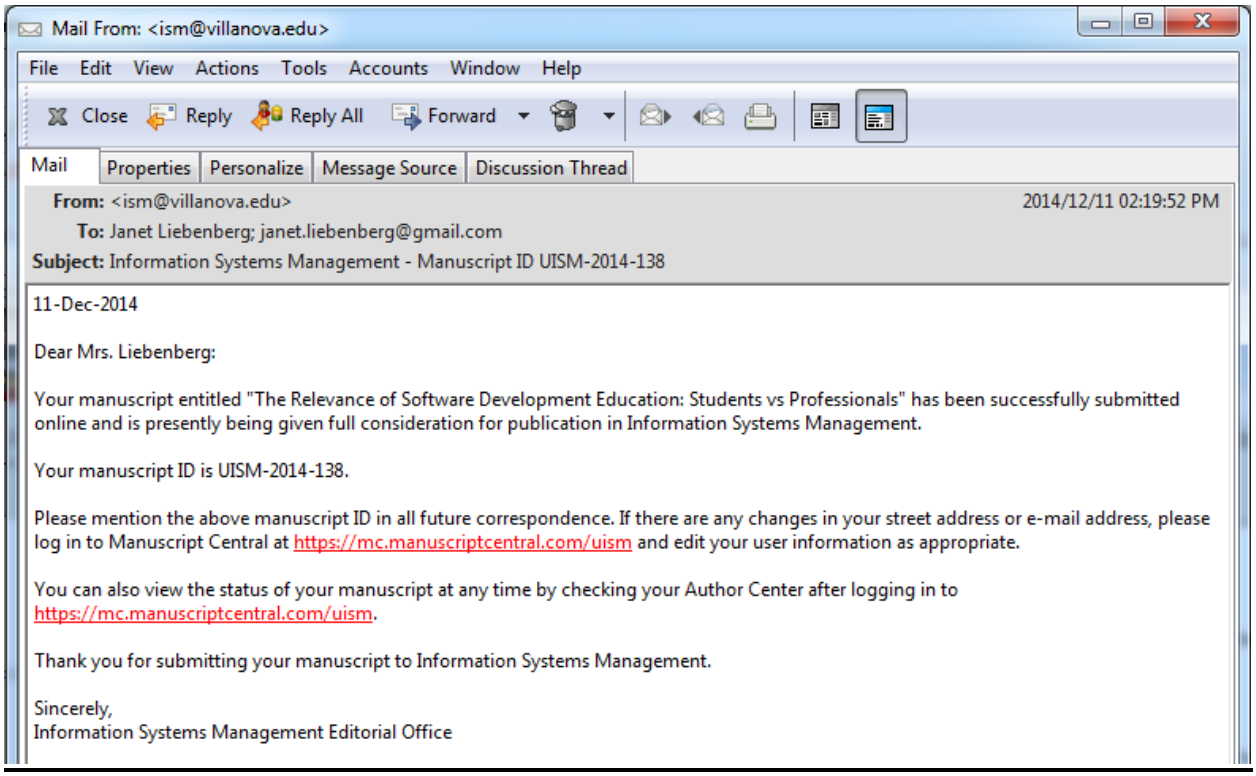
Proofs. Page proofs are sent to the corresponding author. They must be carefully checked and returned within 48 hours of receipt.

Complimentary Policy and Reprints. Authors for whom we receive a valid email address will be provided an opportunity to purchase reprints of individual articles, or copies of the complete print issue. These authors will also be given complimentary access to their final article on *Taylor & Francis Online.*

Open access. Taylor & Francis Open Select provides authors or their research sponsors and funders with the option of paying a publishing fee and thereby making an article permanently available for free online access –*open access* – immediately on publication to anyone, anywhere, at any time. This option is made available once an article has been accepted in peer review. Full details of our Open Access program

Visit our Author Services website for further resources and guides to the complete publication process and beyond.

Proof of Submission



Mail From: <ism@villanova.edu>

File Edit View Actions Tools Accounts Window Help

Close Reply Reply All Forward

Mail | Properties | Personalize | Message Source | Discussion Thread

From: <ism@villanova.edu> 2014/12/11 02:19:52 PM

To: Janet Liebenberg; janet.liebenberg@gmail.com

Subject: Information Systems Management - Manuscript ID UISM-2014-138

11-Dec-2014

Dear Mrs. Liebenberg:

Your manuscript entitled "The Relevance of Software Development Education: Students vs Professionals" has been successfully submitted online and is presently being given full consideration for publication in Information Systems Management.

Your manuscript ID is UISM-2014-138.

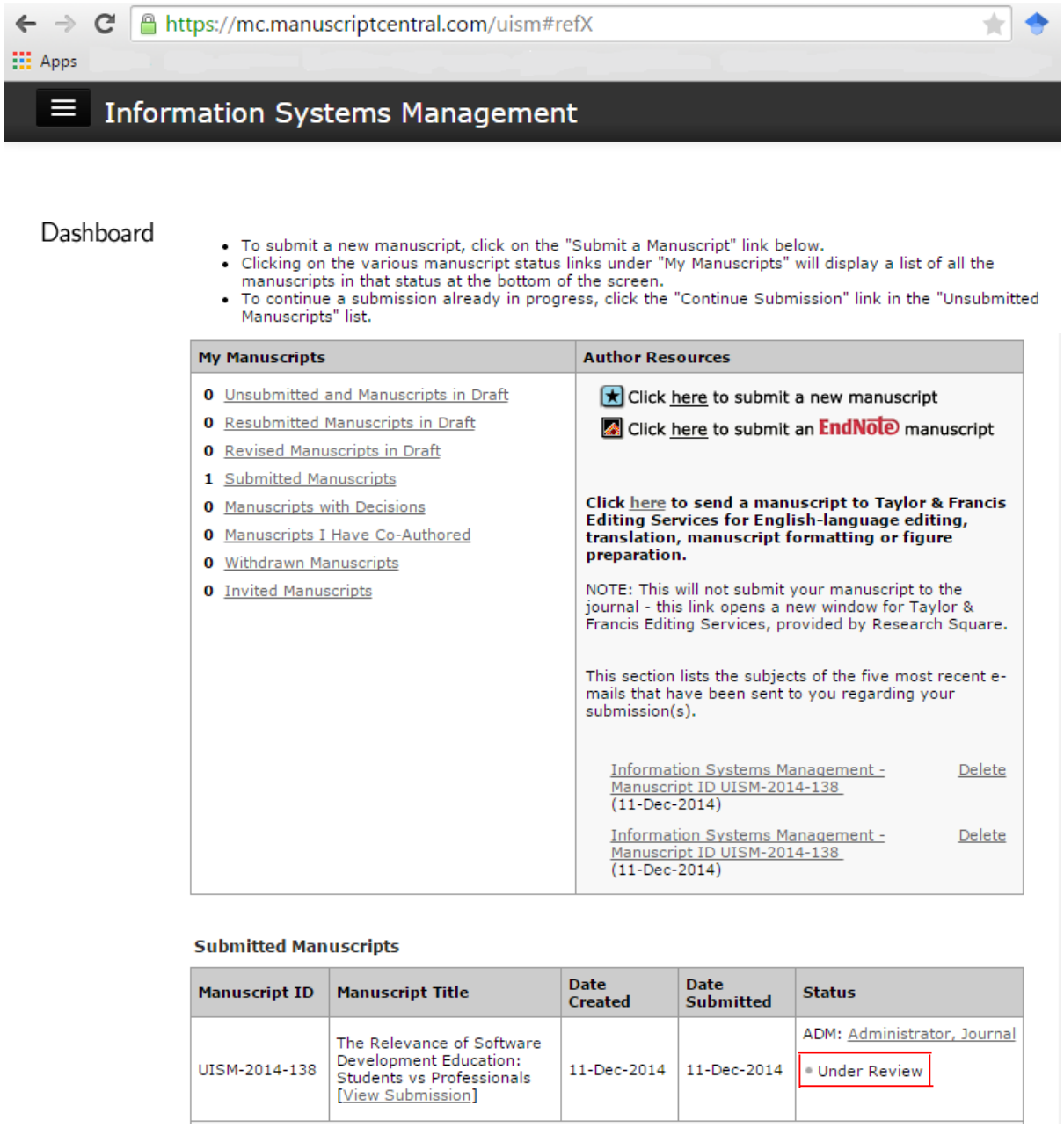
Please mention the above manuscript ID in all future correspondence. If there are any changes in your street address or e-mail address, please log in to Manuscript Central at https://mc.manuscriptcentral.com/uism and edit your user information as appropriate.

You can also view the status of your manuscript at any time by checking your Author Center after logging in to https://mc.manuscriptcentral.com/uism.

Thank you for submitting your manuscript to Information Systems Management.

Sincerely,
Information Systems Management Editorial Office

Proof of review



Appendix I

Confirmation from language editor

This serves to confirm that I, Isabella Johanna Swart, registered with and accredited as translator by the South African Translators' Institute, registration no. 1001128, language edited the following thesis.

A FRAMEWORK FOR RELEVANT SOFTWARE DEVELOPMENT EDUCATION

by

Janet Liebenberg

I Swart

Dr Isabel J Swart

18 MAY 2015

Date

23 Poinsettia Close
Van der Stel Park
Dormehlsdrift
GEORGE
6529
Tel: (044) 873 0111
Cell: 082 718 4210
e-mail: isaswart@telkomsa.net