Teaching of Essential Professional Skills Required by IT Graduates

Carin Venter
School of Computer Science and Information Systems, Faculty of Natural and Agricultural Sciences, North-West University
South Africa

Abstract

Information technology (IT) professionals, such as software developers, require more than technical skills, e.g. software design and programming, to excel at their profession. Software developers work in diverse and cross-functional teams; they must, for example, elicit appropriate business requirements; design and develop suitable software artefacts, and provide training to end-users. A delicate balance between technical and non-technical (professional) skills are key to software success (or failure). IT students must thus acquire necessary professional skills, in addition to technical skills, to be well-rounded and productive employees upon entering workplaces. This study reflects on a BSc/BCom IT degree offered at a leading South African university; it aims to determine whether IT students acquired the mandatory non-technical skills that they needed. So, the researcher interviewed recent IT graduates to reflect on the professional skills they felt they lacked, and incorporation thereof into the curriculum. Actions were then taken, based on the responses of the participants, and teaching of professional skills was successfully incorporated into the curriculum.

1. Introduction

Individuals that choose IT-related careers are usually naturally technically oriented. They are thus typically interested in the options (and challenges) offered by field the science, technology, engineering, and mathematics (STEM) [12]. Similarly, software developers choose this career because they enjoy the technical challenges accompanying the development of new software artefacts. They must acquire very specific and specialised skills. As examples, they must have solid analytical, problem solving, software design and programming skills. They learn these at tertiary education institutions such as universities; the institutions must continuously provide sufficient quantities of (suitably qualified) graduates for the continually increasing demands of the growing IT industry [11].

Unfortunately, literature indicate, and have been indicating for quite some time now, that software development projects often fail; it is a world-wide trend indicating a continuing struggle to design, develop and deliver successful (user-acceptable) software [9; 16]. It seems that developers generally develop software that are technically good; however, it may then still fail due to low acceptance rates by end-users, when users reject it simply because they *feel that it does not meet their specific needs* [2; 5]. It is thus "other" (non-technical) factors that lead to software failure and, to circumvent it, the "other" issues must be identified so that developers can adapt and work towards software development success.

Non-technical issues will require specific non-technical skills; these must be identified to enable developers to learn these skills, so as to effectively deal with "other" issues impacting negatively on project success—the researcher will refer to these as "professional skills" for the remainder of this paper.

So, in this study, the researcher aimed to identify the professional skills that recent graduates, which are currently working as software developers, felt they were lacking upon graduation, and entering their workplaces. The aim is to identify the non-technical skills that hindered them to develop user-acceptable software for end-users, in order to enable the university to incorporate it into the IT degree. The researcher applied critical social heuristics, as a reflective practice, to reflect on the research problem.

This paper is organised as follows: Section 1 introduces the study. The research problem is stated in Section 2. Section 3 discusses the action research approach that was followed. Section 4 gives a brief overview of the key concepts of the study, i.e. professional skills required by software developers; the IT degree explored in this study; and critical social heuristics, positioned in the critical systems thinking paradigm, as the theoretical lens applied in this study. Section 5 discusses the outcomes of the interview; it identifies the professional skills that the

graduates lacked. Section 6 discusses the actions that were taken, i.e. next steps following on the outcome of the interviews in terms of incorporating teaching of professional skills into the IT curriculum. Section 7 reflects on the outcome of the study and, lastly, Section 8 gives a summary and conclusion.

2. The problem statement

It has been pointed out that academic institutions may fail to adequately prepare students for industry and their future places of work [19]. It is crucial to continuously reflect on what is required from students upon entering industry, so that institutions can respond timeously, so as to effectively create the next generation workforce [20]. Considering this, the researcher maintains that software developers that completed typical IT degrees may have obtained sufficient technical skills during their tuition; however, they did not necessarily acquire mandatory non-technical (professional) skills that are also required from them to be successful in their future careers, upon entering industry.

The IT industry grows at a tremendous rate and, as such, the demand for IT professionals, including software developers, grows daily. Post-secondary educational institutions do their best to keep up with the demand [11]. They endeavor to continue to deliver accomplished graduates that are ready for industry; however, something seems to be amiss. The trend of software development projects that, according to literature, continue to fail indicate that software developers struggle to design, develop and deliver software that is technically good *as well as user-acceptable*; literature affirms that these software mostly fail due to low acceptance rates, i.e. when users feel that their needs have not been met, rather than due to technical infeasibility [2; 5; 9; 16].

Empirical data gathered for this study supports this premise: Students that are admitted for an IT degree must generally have strong technical and/or academic backgrounds, and hence must meet high minimum entry criteria; at the university where the study was conducted, entry criteria for academic programmes that aim to prepare IT professionals, e.g. software developers, are also reasonably steep; examinations are equally difficult. For example, at this university, analysis of the IT curriculum's content and delivery, that consists mainly of software design, development and implementation, confirmed that assessments include frequent testing of theory *and* appropriate practical application of a range of software planning, design and development, as well as abstract and functional programming concepts. Upon completion of this degree students should thus be able to develop efficacious software for end-users.

Employers refer to students that graduate from this university as "strong software developers and programmers"; they are in demand and reaped up by industry upon graduation. The participants chosen for this study were all recent graduates that are currently employed as software developers. All of them have impeccable academic records. The author of this paper therefore safely presumes that these graduates were (technically) sufficiently prepared upon graduation.

Yet, the participants (recent graduates) confirmed that struggled to develop user-centric/user-acceptable software; they often have to do re-work on projects when end-users seem to endlessly change scopes/ requirements for software projects to include what they (the developers) perceive as new, emerging requirements. However, the end-users continue to insist that the requirements are not new, and that they (the development team) merely failed to understand appropriately from the onset of the project. This causes problematic issues such as schedule delays, budget overspends, uncertainty, conflict, and low morale within and amongst development teams, and also conflict with management as well as end-users. This correlates well with the literature confirming some of the major stressors of IT specialists, i.e. "Deadlines: issues relating to the need to complete projects within schedule"; "Coworkers: power struggles and conflicts that may result from working with others"; and "User demands: pressures put on staff by users, such as dealing with the IS user interface" [6].

Hence, the questions that the researcher aims to answer in this study are: What other, non-technical (professional) skills should the IT students be taught, to enable them to effectively deal with the non-technical aspects that negatively impact upon their ability to design and develop software that end-users will accept and use, without re-work, schedule delays and budget overruns? And how could teaching of these be effectively incorporated into the curriculum, i.e. without over loading students with too much additional study material?

3. The research approach

The researcher applied an action research (AR) approach. AR is an interactive form of knowledge development - it is practical and participatory, and it focuses on change/intervention [4]. AR was suitable for this study - the researcher was able to intervene in the identified problem context; plan and take suitable actions; and also evaluate the effectiveness of these actions that aimed to improve the stated problem, and answer the research questions.

The study was structured according to similar AR work, as proposed and conducted by authors that also work in the IT fields [3; 4]. Holistic and reflective research entails three elements: a method (M), which embodies an underlying theoretical framework (F) that is to be applied to improve/resolve an identified

problem, i.e. an identified area of concern (A)—the FMA framework is illustrated in Fig 1 [4].

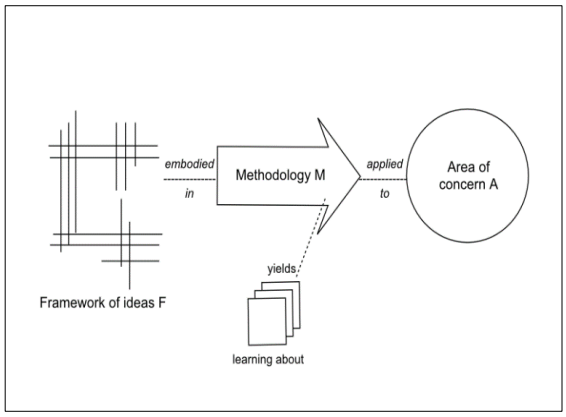

Figure 1. FMA Illustration: Elements in Research [4]

Secondly, AR phases as prescribed by [3] were followed and combined with the FMA framework of [4] as follows: first, problem context, i.e. the area of concern (A), was defined; second, relevant actions were planned by means of a method (M) and guided by a theoretical framework (F) to improve upon the problem (A); third, actions were taken, as planned; fourth, outcomes of the implemented actions and the extent to which the research questions were answered and the problem (A) was resolved through (M) were assessed; and fifth, learning was specified and the next iteration of action planned and executed.

The execution of the AR steps in this study are discussed next.

3.1. Definition of the area of concern

The area of concern (A) is discussed in Section 2; it gives an overview of the problem statement and states the questions that this study aimed to answer. Also, key concepts of the study are discussed in Section 4 to create a shared understanding of these.

3.2. Action planning

To plan the action(s) to be taken to improve the identified problem context and answer the research questions, the researcher did an in-depth case study in a company that employs a number of recent graduates from the university where the study was conducted. The case study approach is suitable for research, such as this and in this context, i.e. to focus on a single instance of an area of investigation that aims to obtain rich and detailed insight into relevant complex relationships and (e.g. organisational) processes [14]. In this case, the researcher wanted to explore the perceptions of recent graduates that successfully completed a BSc/BCom IT degree at the university, and currently work as software designers and/or developers.

She therefore interviewed recent graduates that graduated within the last 5 years with a BSc/BCom IT degree from the university, and are currently employed to design and develop software for (in-house) end-users.

The software development team in the company where the case study was conducted consisted of a total of nine software designers and developers; six of them completed the undergraduate (3 year) degree (and thus graduated with a basic BSc/BCom IT degree), and also completed/were in the process of completing the optional 4th (Honours) year level. These six were chosen as participants and they were invited to be interviewed. They were chosen because they have completed all the modules that are regarded as core modules to qualify as software developers in industry—the structure of the degree and curriculum is discussed in Section 4.2—and they were actively working as software designers and developers.

Roles were not well defined in the company, and the employees were expected to be able to fulfil any role, as required, in a software development project. This was a frustration for the participants, but provided an additional benefit to this study, since all the participants were approximately equally exposed to the entire spectrum of software design and/or development during their employment, and therefore were all relatively equally knowledgeable about the processes and procedures of the company. The company followed an agile development approach to develop software.

The participants were interviewed as a group; the group interview had the added advantage of allowing the researcher to observe how they interact with each other. So it thus also gave an indication of their perceived interpersonal skills in a group context (as an example of a required professional skills that is relevant to all types of work that involves some sort of team work, as would also be the case when working as part of a software development team); so, the aim was to identify whether any particular participant could potentially be perceived as 'socially awkward', i.e. lacking basic interpersonal skills. Fortunately, none appeared to be 'socially awkward'; all contributed equally and shared views openly yet politely. The researcher also had an opportunity to follow up with individual participants after the group interview to clarify statements if/where required.

Critical systems thinking (CST) was applied as the theoretical lens (F) to explore the viewpoints of the participants, and critical social heuristics (CSH), as a framework positioned in the CST paradigm, was used to structure the interview questions; it was also used as a framework to reflect on the participants' responses. CSH provided a structure whereby the stated problem could be explored from the following views: who/what makes software and/or software development purposeful and measurable for end-

users; who is, vs. who ought to, control resources; who/what is, vs. who/what ought to be, considered relevant experts/ expertise to guarantee successful application and continued use of new software; and who/what should identify/ensure emancipation of stakeholders that may be affected, yet are uninvolved [17]. Participants were guided during the interview to reflect on their software development work from these viewpoints. The outcome of the interviews are discussed in Section 5.

3.3. Action taking

Next steps (actions) that were taken in this study, as per the outcomes of the interview, are discussed in Section 6. The researcher implemented ideas formulated, as per the information gathered from the participants. She then reflected on the outcome of the implemented actions; it is discussed in Section 7.

4. Key concepts of the study

The key concepts of the study are discussed next; the aim of these sections are to create a shared understanding of the following: professional skills required by software developers, as per the literature; a brief overview of the BSc/BCom IT degree taught at the university where the study was conducted; and a short overview of CSH to position in this study.

4.1. Professional skills for developers

This study is grounded in the conviction that software development is by nature a collaborative process - it acknowledges that specific "soft skills enhances the likelihood of an individual's success and contributes positively towards the common goal of the project"; yet, it seems that, in general, IT professionals "tend to be poor at verbalizing how the task affects the people involved… the greatest different between software engineers and the general population is the percentage who take action based on what they think rather than what they feel, a fact that does not help in bringing software engineers closer to their customers" [1]. The literature often refer to these non-technical skills as "soft skills"; however, the researcher argues that the term "soft skills" may be limiting and therefore, in this study, she rather refers to "professional skills" to encompass relevant non-technical skills required by an individual, which complements his/her technical abilities and aids achievement of given (technical) tasks.

Limited research has been done in terms of particular professional skills required by software developers [1], but authors agree that the degree to which software development teams comprehend and aptly interpret specific needs and requirements of end-users drives software development success (or failure); e.g., they must accurately understand and interpret end-users' business requirements, so as to design and develop successful software; they must also be able to work effectively in teams to achieve this [1; 10; 13]. The most prominent skills that overlap for all individuals involved in software development projects are: communication (and listening) skills; analytical and problem solving skills; interpersonal skills; and the ability to be a team player [1; 13].

Essential professional skills required by software developers do not refer, in the context of this study, to personality traits of people that are software developers; these are also important and should not be ignored when allocating team members to projects [1]—refer also to the rationale for having a group interview so as to determine whether the participants appeared to have sufficient interpersonal skills to work in a group context—however, personality traits of individuals are not the focus of this study.

4.2. The IT degree

The degree offered at the university entails a 3 year (undergraduate) programme, with an optional 4th (Honours level) year that allows students to specialise. The university's degrees are governed by the South African governing body for academic institutions (SAQA) [15].

The curriculum includes a number of core modules, i.e. introduction to programming (Python); basic programming (Java/C#); advanced programming (Java/C#); data analytics; security; data structures; business/systems analysis; basic statistics; basic mathematics; communication skills; business management; accounting and databases (MySQL). The fourth (optional) year of the curriculum also includes the following elective modules (that have all been completed by the participants), i.e. advanced databases (Oracle); data warehousing (the Kimball lifecycle approach; SQL); and information systems engineering (with project management and software development approaches).

4.2. Critical social heuristics

Critical social heuristics (CSH) operationalises the premises of the critical systems thinking (CST) paradigm [17]. CST aims to aid social improvement through awareness, complementarism and liberation from oppressive structures [8]. CSH is a critical systems methodology that enables critical reflection; it facilitates identification of underpinning boundary judgements and associated normative implications of a problem, and guides rational application of improvement actions [18]. CSH enables a problem solver to determine what *is/has been* done, and what *ought to be* done to improve a problem context [7]. Reflection from a CSH perspective aims to be

holistic and systematic, and is, as such, to be done from the perspectives of four categories, i.e. the basis of motivation; the basis of power/control; the basis of expertise; and the basis of legitimacy on behalf of uninvolved, yet affected stakeholders [17]. Problems can be analysed effectively from the perspectives of these four categories, according to involved and affected (yet not involved) stakeholders to ensure legitimate design and implementation of systems.

For this study the researcher therefore applied CSH as the theoretical lens to systematically explore the viewpoints of the participants in terms of their actual experience, vs. what they felt should have been in place that could have improved their circumstances upon starting work as software developers. So, the participants were asked to comment on the following aspects of their work: who/what makes software and software development purposeful and measurable for them and/or for end-users; who is, vs. who ought to, control software development resources (e.g. processes, funds, tools); who/what is, vs. who/what ought to be, considered relevant experts/expertise to guarantee successful application and continued use of new software; and who are affected, yet uninvolved, and who/what should identify/ensure liberation of them.

5. Interview outcomes

The outcome is also structured according to the viewpoints of the participants as per CSH categories that also guided the discussions with the participants. It is discussed next.

5.1. Purposeful and measureable software

The participants acknowledged that, when starting with a new software development project, they find it difficult to identify the exact purpose of the new piece of software, and how it will ultimately be measured by end-users, at the onset of the project. Software are often used across departments, and representatives of those departments have different views about the relevant purpose(s).

They also say that there were often perceived incongruities between what they, as designers and developers, initially thought it was that the end-users wanted, and what the end-users would be willing to accept as usable and/or worthy software in the end. These disparities had very little to do with technical issues, but rather with functional issues and what the developers regard as "nice-to-haves"; these would, for example, be confusions regarding page lay-outs or buttons, color schemes, reporting formats etc.

The participants said that they derive from this that they were not taught suitable communication and active listening skills; they did not learn how to be "asking the correct questions", therefore they often feel that they do not really "know what they [the clients/end-users] want" from the final product that are asked to develop. A few of the participants felt that they were able to succesfully derive business requirements from end-users; however, most of them acknowledged that they felt incapable to properly "…structure those questions…", especially where the software will serve cross-functional departments/ teams/diverse end-users, e.g. financial, auditing; project management, and processing/manufacturing department; and/or operational and tactical/strategic management.

The participants agreed that "…if you ask the correct questions you can move towards the right code", and therefore feel that this should be incorporated (integrated) into the curriculum. One participant indicated that, upon entering the work place and immediately after graduation "we did not have the skills to ask the correct questions…" therefore their clients often want to continuously "…keep changing, keep changing, keep changing…" so projects' scopes were difficult to fix—it results in scope changes when end-users see prototypes of the product, and then continue to add requirements to the scope of the project, without increasing the amount of time allowed to finish the final product. Continuous re-work and then, ultimately, limited time to finish final software projects, results in failed software in the sense that the end-users claim that it does not meet their requirements (which, must be noted, were unclear from the onset of the project). So, they wanted to learn how to properly incorporate changes through change management.

5.2. Proper controlling of resources

Change management, as a systematic approach to dealing with transitions within an organisation, or on a project where approved changes go through a structured justification and approval process, is not part of any of the subjects included in the degree offered by the university. The students are given fixed scope projects as practical assignments; the scopes of these projects do not vary during the time that they have to complete them, and they do not have to deal with managers/end-users that move goal posts; so they did not learn to cope with project scope changes and/or people that initiate these scope changes.

Ultimately, the participants did not feel that they were in control of their own software development projects. Scope changes that resulted in schedule delays and budget overruns were blamed on them; but they felt that end-users also had to take responsibility for some of these. So, they also wanted to learn to be assertive, yet still professional, when communicating with end-users (that are often their seniors), for example, to explain the impact (in terms of time, resources and effort) of every requested scope change timeously. They also wanted to be able

to implement proper change management processes for their projects.

5.3. Relevant experts/expertise

The participants acknowledged "…that you get these excellent technical students entering the workplace, but then how to communicate, how to sit in a meeting and voice your concerns or your opinions. How do you do it from day one? It's a skill that you need to acquire. If you can't get it on university level, it's very difficult out in the workplace. It's a dominant world." They want to learn how to be seen as team players, and not just technical IT people that are "…solitary in the sense that we don't like to interact with people…", i.e. they want to be "debunking the notion of us as IT… people who don't really care about the client so to say… if we can start by getting rid of that idea from us as IT people, then I think we can grow and actually get communication skills…"

The participants agreed that they want to learn communication (interviewing and listening) skills as part of the practical software development modules, and where they develop actual software artefacts. They want it integrated into existing modules, and not necessarily added as an additional module to the curriculum. This way they can learn to specifically communicate (listen) as is required in their area of expertise, rather than only in general.

Requirements analysis/gathering, documentation of business requirements, and interviewing of end-users, are a definite part of a few of the software development modules (e.g. business/systems analysis and data warehousing); however, these were, according to the participants "rushed through" in order to allow students to spend more time on the technical aspects of system/software analysis, design and development.

The participants realised now, in retrospect, that it would have been beneficial to have more practical project work that integrated individual modules and ran "across" individual yet interlinking modules, to integrate concepts business analysis; requirements gathering for software design and development; interviewing of end-users (*what* to ask in addition to *how* to ask the right questions); and design and development actual databases for "real" clients/end-users, rather than "dummy" case studies/ scenarios; they feel that this will definitely help in "[i]ntegrating the technical side and the soft side.".

5.4. Liberation of the affected

The affected, in the context of this study, is the IT students that do not learn the relevant non-technical, professional skills that they also need to acquire to be successful software developers upon graduation. The oppressing structure to be liberated from is the lack of professional skills teachings in the IT curriculum. However, the current technical skills that they learn, must remain part of the curriculum, as these are also crucial for software developers. So, a balance must be found between teaching of the technical skills (currently included) and required professional skills (to be included), without overburdening the students. The credit-load of the IT degree, as governed by SAQA [15], should not be exceeded.

6. Actions taken (next steps)

Two natural next steps followed upon the outcome of the initial in-depth case study: Firstly, a second (last) semester 3rd year module where the IT students must work in teams to develop a (pre-defined) software artefact was identified. The project usually entails a software application to be designed and developed, based on specifications provided by the lecturer. The aim is to integrate all technical students acquired, and allow students to apply a range of skills taught to them over a period of 2½ years.

However, for this study, the university worked in collaboration with members of the community and a real-life project, that would also improve the community, was identified. A random sample of ten students were selected, and they were given this community (real-life) software development project, instead of the usual "dummy" project. Three staff members and one community member were also involved in this project - the community member played the role of project sponsor; the three staff member were in the roles of project manager, project/module coordinator and project technical lead respectively. The project was overseen by the school director. The team followed an agile development approach with regular sprint and progress meetings.

The team had tight deadlines to work towards. They had to follow all the steps of a software development project, i.e. starting at analysis of the business requirements (which were not yet clearly defined as the community project entailed the development of a mobile application, but the exact details and scope had to be defined still, as is often the case in real software development projects). So, the team had to investigate technological platforms and tools, document their work, and give regular feedback to the project sponsor. The project sponsor (with minimal background and knowledge regarding IT and software development) was interviewed to determine business requirements (that continuously changed throughout the project, much to the frustration of the project team as well as the project sponsor, again, as is the case with actual software development projects - refer to previous discussions in this regard). As a result, functional and technical specifications continuously changed.

At the end of the semester, the students were interviewed to determine what they have learned from this project - it is discussed in Section 7.

7. Reflection and specification of learning

At the end of the semester, the mobile application was, to the disappointment of both the students and the project sponsor, as well as the involved lecturers, not finished. What started off as a seemingly very simple mobile application to be developed, evolved into a frenzy of scope changes, documentation, re-work and schedule delays that frustrated the students and staff, as the project sponsor changed business requirements as well as the technological platform half-way through the project.

The students agreed that they did not learn a lot of technical skills by means of this project (but it must be noted that this was luckily not the aim of the project). They did, however, agree that they learnt valuable lessons in terms of business analysis (asking the right questions in the beginning of the project, rather than to assume that they know what the client wanted and expected from them); to communicate more effectively within the team and with the client; how to listen more effectively; and how to properly plan from the onset of the project. These correlate well with the professional skills that the interviewed graduates (refer to Section 5) felt they were lacking.

When interviewing the remainder of the students that did not participate in the community project, but continued with the module and completed a dummy project, as per previous years, these students also felt that they did not learn any new technical skills (as expected), but merely applied the skills that they learnt in other modules to complete their projects (as is the aim of this module). However, they did not have the added advantage of learning any of the professional skills that came with working with an actual, real client.

So, it can be concluded that incorporating a project that simulated a real-world scenario, such as the community project, yielded success and taught the students mandatory professional skills required by software development professionals. This project will be developed further in consecutive years to ensure that it be completed successfully. Similar projects will also be identified so that the whole class can benefit from these.

An additional (and unexpected) outcome of this study was that the lecturers involved in this study said that they also gained valuable practical "real-world" experience, which they feel can be ploughed back to improve their future classes and teachings.

8. Summary and conclusion

This study reflected on professional skills that software developers should acquire to effectively develop successful software. These should ideally be learned as part of academic tuition, since lack of these non-technical skills hinder software developers and software development success. Students learn technical capabilities to develop technically good, robust software. Yet, the technical skills are not well-balanced with required non-technical professional skills. The outcome is then frustration and failed software projects.

This study reveals shortcomings that recent IT graduates felt that would have, if incorporated as part of the IT degree, helped them to be more successful. They want to learn to communicate more effectively (assertively) in team meetings and with clients/end-users; they want to be able to aptly guide interviews so as to gain a proper understanding of business requirements upfront (to minimise scope changes and develop better software in limited allowed time frames). They also want to learn to manage changes that result from scope changes.

Participants indicated that it would have helped them to have more exposure to projects that mimic real-life scenarios, integrate teachings of different modules, and hence acquire a balance of technical and professional skills. An identified community project, where a real-life project was given to a number of students for a 3rd project module, yielded positive results. The students (and staff) learnt valuable lessons and acquired professional skills.

9. References

[1] Ahmed, F., Capretz, L.F., Bouktif, S., and Campbell, P., 2015. Soft skills and software development: A reflection from the software industry. *arXiv preprint arXiv:1507.06873*.

[2] Avison, D. and Fitzgerald, G., 2006. *Information systems development: methodologies, techniques & tools*. McGraw-Hill, London.

[3] Baskerville, R.L., 1999. Investigating information systems with action research. *Communications of the Association for Information Systems* 2, 19, 1-32.

[4] Checkland, P. and Holwell, S., 1998. *Information, systems, and information systems: making sense of the field*. Wiley, Chichester.

[5] Clegg, B. and Shaw, D., 2008. Using process-oriented holonic (PrOH) modelling to increase understanding of information systems. *Information Systems Journal* 18, 5, 447-477. DOI= http://dx.doi.org/10.1111/j.1365-2575.2008.00308.x.

[6] Colomo-Palacios, R., Casado-Lumbreras, C., Misra, S., and Soto-Acosta, P., 2014. Career Abandonment Intentions among Software Workers.

Human Factors & Ergonomics in Manufacturing & Service Industries 24, 6, 641-655. DOI= http://dx.doi.org/10.1002/hfm.20509.

[7] Flood, R.L. and Jackson, M.C., 1991. *Creative Problem Solving: Total Systems Intervention*. Wiley, Chichester.

[8] Flood, R.L. and Jackson, M.C., 1991. Total systems intervention: A practical face to critical systems thinking. *Systems Practice* 4, 3, 197-213. DOI= http://dx.doi.org/10.1007/BF01059565.

[9] Hwang, M.I. and Hongjiang, X., 2007. The Effect of Implementation Factors on Data Warehousing Success: An Exploratory Study. *Journal of Information, Information Technology & Organizations* 2, 1-14.

[10] Jalil, R., Khalid, J., Maryam, M., Khalid, M., Cheema, S.N., and Iqbal, I., 2019. Requirement Elicitation for Bespoke Software Development: A Review Paper. In *Intelligent Technologies and Applications*, I.S. BAJWA, F. KAMAREDDINE and A. COSTA Eds. Springer Singapore, Singapore, 805-821.

[11] Mahalepa, T., 2016. Developing guidelines for business intelligence modules in information technology programmes at universities using critical systems heuristics North-West University.

[12] Miller, K., Sonnert, G., and Sadler, P., 2018. The Influence of Students' Participation in STEM Competitions on Their Interest in STEM Careers. *International Journal of Science Education, Part B: Communication and Public Engagement* 8, 2, 95-114.

[13] Mtsweni, E.S., Hörne, T., and Van Der Poll, J.A., 2016. Soft Skills for Software Project Team Members. *International Journal of Computer Theory and Engineering* 8, 2, 150.

[14] Oates, B.J., 2006. *Researching Information Systems and Computing*. SAGE.

[15] Saqa, 2012. The South African Qualifications Authority: Level Descriptors for the South African National Qualifications Framework.

[16] Shehzad, B., Awan, K.M., Lali, M.I.U., and Aslam, W., 2017. Identification of Patterns in Failure of Software Projects. *J. Inf. Sci. Eng.* 33, 6, 1465-1479.

[17] Ulrich, W., 1983. *Critical Heuristics of Social Planning*. Haupt, Bern.

[18] Ulrich, W., 2003. Beyond methodology choice: Critical systems thinking as critically systemic discourse. *Journal of the Operational Research Society* 54, 4, 325-342. DOI= http://dx.doi.org/10.1057/palgrave.jors.2601518.

[19] Wixom, B., Ariyachandra, T., Douglas, D., Goul, M., and Gupta, B., 2014. The Current State of Business Intelligence in Academia: The Arrival of Big Data. *Communications of the Association for Information Systems* 34, 1-13.

[20] Wixom, B., Ariyachandra, T., Douglas, D.E., Goul, M., Gupta, B., Iyer, L.S., Kulkarni, U.R., Mooney, J.G., Phillips-Wren, G.E., and Turetken, O., 2014. The current state of business intelligence in academia: The arrival of big data. *CAIS* 34, 1.

