A Pedagogical Intervention Based on Agile Software Development Methodology

Sanjay Ranjeeth
Ashley Marimuthu
Manoj Maharaj

Abstract

In recent times, the activity of software development has been tagged as being embroiled in a crisis because of the inability of software developers to deliver quality software. In response, the software engineering (SE) community have opted to discard the traditional processes that underpin software development in preference for a set of processes that have been termed as agile methodologies. The underlying philosophy of the agile approach is that the software development process should enhance the possibility of constant interaction with the customer and also be adept at accommodating changing customer requirements. In this article, we examine the pedagogical implications of using the agile approach as part of an academic programme. We also report on students' acceptance of the agile approach as a methodological framework for the development of an information system as part of their capstone major project course. A purposive sampling strategy was employed to conduct a survey with final year Information Systems & Technology students at the Pietermaritzburg and Westville campuses of the University of KwaZulu-Natal. A 71% response rate was achieved. A combined academic framework consisting of behavioural science and design science theory was used to operationalise acceptance of agile methodology. The results from each of the criteria used to quantify acceptance of agile methodology indicate a high level of acceptance of agile methodology within the IS student community.

Keywords: software development, information systems, agile methodology, behavioural science, design science, constructivism, connectivism

Introduction

The broad discipline of information technology (IT) is anchored around the core activity of software development which in turn, is historically grounded within computer science (Shackelford *et al.* 2006). The focus in computer science is on the delivery of functional software underpinned by a strong mathematical component that ensures optimum usage of the processing capacity of a computer (Glass 1994). In an overview report on academic curricula in computing disciplines released by the Association for Computing Machinery (ACM) (Shackelford *et al*. 2006), a post-modernistic view of software development is adopted. The recommended strategy for disciplines such as software engineering (SE) and information systems (IS) is that these disciplines should focus on the non-functional aspects of software development. As a consequence, software systems began to incorporate a social dimension where the usability thereof began to assume as much attention as functionality. The software development process models have also begun to adopt a "business-like" demeanour where the imperative is that quality software should be developed on time, within budget and to satisfy requirements that have been stipulated by the customer. Recently, the activity of software development has been tagged as being embroiled in a crisis because of the inability of software developers to deliver quality software that is usable and in accordance with customer's expectations of the system (Parnas 1994; Glass 1994; Schach 2008; Pressman 2010).In response to this dilemma, the software engineering community has opted to discard the traditional processes that underpin software development in preference of a set of processes that have been termed agile methodologies. The underlying philosophy of the agile approach is that the software development process should enhance the possibility of constant interaction with the customer with a view to efficiently accommodating changing customer requirements.

Data recently released in the "State of Agile Development" survey (VersionOne 2011), indicate a global acceptance of the agile approach as the current de-facto software process model of choice. It is reported in Cohn (2012), that according to the Standish Group 2011 report, software applications developed through the agile process have three times the success rate of the traditional waterfall method and the agile process could be viewed as a possible solution to the problem of failed software projects.

In order to align the undergraduate curriculum offered by the Discipline of Information Systems & Technology at the University of KwaZulu-Natal (UKZN) towards the latest trends in industry, the capstone project module offered at third year level has been re-designed in accordance with the dictates of Extreme Programming (XP), a popular agile methodology. The choice of XP is informed by claims made by Bergin, *et al.* (2004) that XP has a positive influence on the learning of computer programming and it facilitates the use of constructivism as a pedagogical strategy

In this article, we examine the pedagogical implications of using the agile approach as part of an academic programme. We also report on the students' acceptance of the agile approach towards the building of an information system as part of their capstone project course.

The Research Questions

The following research questions have been used to underpin the current study.

- What does Agile Methodology of software development entail?
- What are the pedagogical challenges of implementing the Agile Methodology as part of a capstone module?
- What is the students' level of acceptance of Agile Methodology?
- How well did the students comply with the requirements of XP?

The Academic Dilemma

There is a growing body of opinion that suggests that research within the disciplines of management studies as well as information systems is severely lacking in relevance (Davenport & Markus 1999; de Villiers *et al.* 2007; Holcombe & Thomson 2007), so much so that research produced at universities will have minimal or no impact on the practitioner community. Despite the acknowledgement by some members of the academic community of the lack of relevance of academic research in the above mentioned disciplines, there is still an exclusive focus on academic rigour with very little consideration being given to the relevance of the research effort (de Villiers *et al.* 2007;Worrall *et al.* 2007) . This has resulted in a steady decline in the amount of funding that academics are generating from business because the research produced is lacking in "real world" relevance. There seems to be a strong preference within parts of the IS academic community to engage with the "social dimensions of phenomena" (Pinch 2008), and to relegate the technology to a "black box" status. As a consequence, "…the richness of important and interesting IS research questions has been lost or severely limited" (Niederman & March 2012). A further issue that compromises the relevancy and currency of IS research is that of the time delay, as identified in Knight *et al*. (2008), between the problem inception and the publication of the results of an attempted solution in an academic journal. The academic community seems to wait for emerging trends in the practitioner sector before any research in that area is conducted and eventually published. Hence, it is actually the academic community that is always playing "catch-up" thereby trivialising the value of academic research to the practitioner community (van Loggerenberg 2007).

An ideal resolution to this dilemma of keeping abreast of technology change as well as bridging the gap between the world of practitioners and the world of academics is to incorporate the technologies and methodologies that are current, from a practitioner perspective, into research and development projects that drive academic curricula of universities. This strategy would entail a revisit to the relevance versus rigour debate because it would entail a resurrection of the importance of producing IS research that is current and relevant (widely discussed in the Alternation Journal, titled Themes in Management Studies (2007)). This solution strategy has been extensively deliberated upon and endorsed by Rosemann and Vessey (2008); van Loggerenberg (2007); Jami and Shaikh (2005); Fällman and Grönland (2002); Benbasat and Zmud (2003); Davenport and Markus (1999) and Lee (1999). As a consequence of the imperative to produce relevant IS research, Hevner *et al*. (2004) made reference to the two main domains of IS research. The first domain is behavioural science, where theories exist to explain the usage of IT artefacts in an organisational context. The dominant theories in this domain focus on the usage, perceived usefulness or intended usage of IT artefacts (referred to as the Technology Acceptance Model (TAM) proposed in Davis (1985)) or the utility value and information quality delivered by an IT artefact (referred to as the Information System Success Model proposed by Delone and Mclean (2004)). The second domain of IS research is design science, where the focus is on the development of an IT artefact, encompassing an evaluation of the feasibility of the development process. In order to enhance the relevance of IS research, the design science domain needs to be given just as much prominence as the behavioural science domain (Hevner *et al*. 2004; Niederman & March 2012; Kuechler & Vaishnavi 2011; Wieringa & Moralı 2012). This approach of delving into the "black box" whereby IS researchers view behavioural and design science as interdependent (Niederman & March 2012) can only serve to add an element of "richness" and broaden the impact of IS research.

In accordance with these sentiments, the current study incorporates elements of the technological and social science realms to investigate the applicability of the agile approach towards software development in an educational context. From a technological perspective, the XP process was subjected to an inquiry regarding its effectiveness as a software process model to develop an information system. From a social science perspective, the effectiveness of the strategies adopted to teach the essence of XP as part of a capstone module was also analysed.

Software Process Models

The development of software is underpinned by software process models that are adaptations of the generic software life-cycle model referred to as the Waterfall model which was proposed by Royce (1970). Many of these adaptations have been given extensive coverage in software engineering texts written by Schach (2008), Pressman (2010) and Sommerville (2007). The main theme emanating from these texts is that the Waterfall model of software development is characterised by a linear or sequential approach consisting of various stages of development. From an overview perspective, these development stages consist of requirements, analysis, design, implementation, testing and maintenance. The Waterfall process is quite rigidly structured and does not easily handle changing software requirements; which entails the developer going back to the requirements stage (the analogy used here is that it is not natural to flow up a waterfall). The Waterfall model is a process oriented model where the process of development is given priority over the possibility of entertaining changing user requirements. In its purest form, the Waterfall model has been subjected to severe criticism from the SE community. These criticisms, which have been summarised in Parnas and Clements (1985), include the following:

- A system's users seldom know exactly what they want and cannot articulate all they know.
- Even if the system's users could state all requirements, there are many details that they can only discover once they are well into implementation.
- Even if the system's users knew all these details, as humans we can master only so much complexity.
- Even if the system's users could master all this complexity, external forces lead to changes in requirements some of which may invalidate earlier decisions.

These factors prompted the SE community to look at other process models. A possible contender was the iterative process model that began to receive attention after a proposal by Basili and Turner (1975) that software development should follow an "iterative enhancement" technique. The idea here is that the software process model has to accommodate changing user requirements as well as deliver functionality incrementally to the user rather than as a complete finished product. This endorsement of iterative software development by the SE community prompted a group of software engineers to formalise the set of iterative development models as the Agile Methodology for software development. The Agile Methodology is informed by a set of core principles and values that is documented by Beck *et al*. (2001) in what is referred to as the Agile Manifesto. The essence of the manifesto is that software development should prioritise:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation

- Customer collaboration over contract negotiation
- Responding to change over following a plan

The Agile Methodology comprises a set of methods that subscribes to the core principles of the Agile Manifesto. A detailed analysis of these agile methods are provided in Dybå and Dingsøyr (2008). It is reported in van Valkenhoef *et al*. (2011), Dybå and Dingsøyr (2008) as well as Vijayasarathy and Turk (2008), that one of the more popular agile methods is Extreme Programming (XP). An overview of the XP process model is provided in Beck (1999), and some of the significant aspects of XP include the following:

- System development is driven by user stories. A user story is essentially communication between the customer (person who commissioned the development of the system) and the system developers. A user story is a brief, concise description of the functionality required by the customer.
- A set of user stories are developed and released for customer review. This is referred to as one of many iterations of the system until it is fully developed.
- The customer must be available for consultation with the development team, thereby entrenching the idea of greater customer interaction.
- A customer provides test criteria that will determine whether a user story has been developed to the customer's preference.
- All production code is written by two people using a single computer. This strategy is referred to as pair programming, an agile computer programming strategy that is given extensive coverage in Hulkko and Abrahamsson (2005) and Vanhanen and Korpi (2007).
- There is no overall, architectural design. The system design evolves with the development. There is constant re-factoring of the system design as the system evolves.

The XP method of software development was introduced as part of the learning experience for students registered for the Major Project capstone module offered by the Discipline of Information Systems and Technology (IS&T) at UKZN. In this article, we report on the pedagogical challenges of incorporating an agile approach such as XP into a Major Project capstone module as well as the effectiveness of the XP process in developing an information system, from a student perspective.

The Essence of the Major Project

The Major Project is a reference to a set of final year modules where IS&T students are required to work in groups of 4 or 5 and build an information system for an organisation (typically, a local business) that is prepared to serve as a client. The ultimate purpose of the system is to provide organisational decision making support where business reports are accessible on a front-end or Web based platform. In order to achieve this objective, the system would have to initially capture and process as much data as possible from core business transactions so that the data can be analysed from many different perspectives.

The Major Project provides an ideal opportunity to allow academics as well as students to bridge the divide between the practitioner and academic worlds. It is reported in Holcombe and Thomson (2007) that at the University of Sheffield, a similar strategy was used to provide students with a large scale project and real client, thus motivating the students as well as providing academic staff with a viable opportunity to engage in current, “cutting edge” research. The value inherent in the Major Project from a student perspective is well documented in Strode and Clark (2007). Lynch *et al*. (2007) analysed student and academic perspectives of the Major Project undertaken at academic institutions in the United Kingdom, South Africa and Australia. They reported that in all three countries, the Major Project was highly endorsed by academics as “…it recognises the need for industrial experience and learning of applied skills, and therefore make these projects a compulsory part of the curriculum”. The group work aspect of the Major Project was also endorsed by Mahnič (2008). The paper asserts that the Major Project exercise is not just about technical skills, but also provides a platform for the acquisition of skills such as teamwork, leadership, planning as well as the production of formal documentation and an opportunity for students to obtain the experience of doing a project presentation to a formally instituted panel consisting of academics as well as industry representatives.

The Pedagogical Challenge of the Major Project

In order to report on students' acceptance of the agile approach towards systems development, it became imperative to ensure that the student cohort chosen for the current study subscribed to the principles of agile methodology. However, getting the student cohort to abide by the dictates of the XP process model in developing the Major Project system presented itself as a pedagogical challenge. The XP approach of attaching less significance to documentation could possibly result in a "cowboy" development style (Ferreira & Cohen 2008) where there is complete disregard for any formal aspects of software development such as requirements gathering, design, planned development, testing and continuous consultation with the system stakeholders. Wellington (2005) warned that the most significant challenge in employing XP in a Major Project course is to ensure that every student abides by the principles of agility and XP and respects these development models as process driven. The temptation to "dive into coding" (Sewchurran 2007) under the banner of being agile needs to be guarded against. The pedagogical challenges of conducting a Major Project exercise as part of a capstone module are well documented by Sewchurran *et al*. (2006) from their experiences at the University of Cape Town (UCT). However, a significant aspect of these challenges was the problem experienced in trying to get students to "internalise" the essence of the agile approach and to engage with the methodology in a conventional manner thereby ensuring the development of a system that conforms to the customer's requirements and expectations. As part of the agenda for the current study, a brief report is provided on the strategy used in overcoming the pedagogical challenge of incorporating the agile software process model as part of the Major Project capstone module.

The Pedagogical Strategy Used to Incorporate Agility

From a theoretical perspective, the educational theories of constructivism and connectivism were deemed to be most appropriate as descriptors of the strategy adopted to present the Major Project course. The learning theory of Constructivism (emanating from contributions by the likes of Piaget, Vygotsky, Bruner and Dewey) allude to the activity of constructing one's own knowledge from personal experiences rather than becoming dependant on an intake of passive knowledge (Applefield *et al.* 2000). Development of a fully functional business information system is quite an undertaking, something that you would not expect many students to have experienced. From a constructivist perspective, this could be seen as a significant disadvantage to the students. In order to minimise this disadvantage, we adopted a strategy of simulating this experience by focusing lectures and practical sessions on the development of a generic point of sales system, a strategy also used quite successfully by Demuth *et al*. (2002) for a similar teaching agenda. This exercise provided students with the opportunity to "construct" the appropriate cognitive structures that would facilitate an awareness of the requirements for the development of the actual Major Project system.

In an attempt to get the students to internalise and identify with the principles of XP, in response to the concerns raised by Sewchurran (2007) and Wellington (2005), we adopted a connectivist approach. The introduction to agile methodology and XP was conducted via a series of lecture presentations. It was quite evident during these lecture sessions that the terminology and methodological explanations used in these lecture sessions only served to increase the abstractionism inherent in the whole concept of agility. Hence, there was certainly a need for a formal pedagogical intervention.

From an educational theory perspective, we decided to use connectivism as our underpinning theoretical model so that the process of knowledge construction regarding XP could be facilitated. However, this knowledge construction had to be guided or "cajoled". The basic tenet of connectivism (Siemens 2005) is that learning takes place when individuals establish "connections" between elements in the learning domain in order to construct new knowledge. Hence, we needed to present the elements that underpin XP to students within a problem-solving context so that they could create their own knowledge regarding XP (within the parameters of the Agile Methodology). An opportunity presented itself, courtesy of the sentiments expressed in Beck (2008), that in order for the agile approach to be successful, there has to be adequate software support (referred to as a software tool) to underpin the software process model. In response to this opinion, Microsoft (2012) released Windows Team Foundation Server (WTFS), a software tool that is designed to support the entire agile process model. After having conducted an inspection of the trial version of WTFS, the authors realised that using WTFS as a software project management tool would ensure that an XP approach would be enforced in the building of the software artefact. This conclusion was based on the support for aspects of agile development and XP that formed the core functionality of WTFS. These included aspects such as user stories, tests cases, release dates, main and navigator programmer (a reference to pair programming). Acquisition of WTFS would incur a significant cost, time and effort overhead to the IS&T division and there was no guarantee that students would use it to underpin their Major Project effort.

In order to resolve this dilemma, we decided to use the practical sessions to get the students to build a scaled down version of WTFS which we referred to as the User Story Application (USA). This strategy served the dual purpose of ensuring that each Major Project group had their own customised software project management tool as well as sufficient knowledge of the components that were used in the building of that tool. In this way, the entire agile approach comprising of aspects such as user stories, test cases and pair programming became an integral part of the vocabulary used by students in the Major Project course. The concept of XP and agile development now seems to have gained widespread acceptance by the Major Project student cohort thereby achieving the objective of reducing the abstractionism inherent in these concepts.

Academic Framework Underpinning the Acceptance of Agile Methodology

It is reported in Chan and Thong (2009) that the constructs of TAM, perceived ease of use (PEOU) and perceived usefulness (PU), are generic enough to be readily used for examining the acceptability of software development methodology (SDM). The current study leveraged off the adaptable nature of these constructs to provide a guiding framework to investigate the acceptability of the core activities that underpin the agile methodology for software development. From an IS research perspective, this approach falls within the ambit of behavioural science research. However, the current study also involves an incursion into the actual software development process and as such, hovers on the periphery of design science research as well. Design science research is anchored around the basic tenet that an innovative IT artefact is developed and becomes the source of inquiry from a research perspective (Wieringa & Moralı 2012; Kuechler & Vaishnavi 2011; Kautz 2011; Niederman & March 2012). However, this interpretation of design science research is still very technically oriented and does little to bridge the gap between technical and social aspects of IS research (Niederman & March 2012).

Table 1: Constructs of Agile Methodology Classified According to an IS Research Framework

	Behavioural Science	
	Perceived Usefulness (PU)	Perceived Ease of Use (PEOU)
Design Science	User Stories are effective in capturing user requirements	Compiling a set of user stories is easy to do
	Test Cases are effective in ensuring that the system works correctly	Test cases are easy to construct and implement
	The time allocated to the analysis phase was sufficient	Refactoring the database is easily accommodated
	An evolving system design is effective in directing the development process	There is no need for a specific design phase
	The “quick route” to system implementation improves the prospect of refining user requirements	The “quick route” to system implementation makes the system easier to develop

In order to address this situation, Niederman and March (2012) propose a second dimension to design research where the software design and construction process itself may be viewed as the artefact of inquiry. By doing so the IS research community would be making a practical and relevant contribution to the software design and development process. In accordance with this assertion, the current study adopts a similar stance by viewing the agile software development process model, as embodied by XP, as the source of inquiry. From an IS research paradigm perspective, this kind of approach can be classified as an amalgamation of the behavioural and design science domains. The overriding academic framework comprising of TAM may be classified as part of behavioural science whilst the exploration of specific aspects of agile methodology can be classified as part of the design science framework.

The academic framework for the current study is underpinned by the dimensions of TAM that are operationalised via references to specific aspects of the agile software process model. This overriding framework was used to inform the design of the research instrument that comprises of a questionnaire.

Table 1 above illustrates the relevance of the academic framework to the design of the questionnaire.

Data Collection & Analysis

A purposive sampling strategy was adopted in order to obtain responses from final year IS&T students. The data collection instrument was a questionnaire that was designed to elicit students' perceptions on aspects of agile methodology. The questionnaire was designed so that perceptions on aspects of agile methodology (alluded to in Table 1) could be quantified on a 5 point Likert Scale ranging from "strongly agree" (coded as 1) to "strongly disagree" (coded as 5). The questions were phrased positively towards the constructs of agile methodology and were classified along the dimensions of PU and PEOU (in accordance with the acceptance framework of TAM). The population consisted of 135 students and there was a response rate of 71%.

It is reported in Sekaran and Bougie (2010) that several questions may be used to measure a single concept. In order to obtain a measure of quantification, "… scores on the original question have to be combined into a single score" (Sekaran & Bougie 2010). In accordance with this suggestion, the analysis of the responses was conducted by collapsing the individual measures of the perception variables into 2 single dependent variables that represented the mean of the individual responses. The dependant variables represented PU and PEOU. An affirmation of the internal validity was obtained by conducting a Cronbach alpha test for these variables. According to Sekaran and Bougie (2010), a Cronbach alpha in excess of 0.6 indicates an acceptable level of cohesiveness with respect to the grouping of questions. A set of 5 questions was used to operationalise the PEOU variable. The Cronbach alpha value obtained was 0.64 and fell within the acceptable range alluded to by Sekaran and Bougie. The histogram representing the PEOU variable is shown in Figure 1.

The summary data from Figure 1 (mean =2.52; median=2.4; mode=2.2) indicate a majority acceptance (75% of responses were below 3 and 50% of the responses were below 2.3) of the ease of using agile methodology. While these results are sufficient to indicate acceptance of the PEOU of agile methodology, the low Cronbach alpha value obtained for the PEOU variable became a source of concern as well as a catalyst for further inquiry. Upon closer scrutiny of the data, it becomes apparent that 2 questions did not seem to fit well with the remaining 3 questions. These 2 questions required responses to the following statements:

- There is no need for a specific design phase (a reference to the whole concept of not having a "big up front design" that is part of the agile strategy).
- Refactoring/changing the database design to accommodate changing user requirements is easy to achieve.

If these 2 questions are removed from the original set of questions, then the Cronbach alpha value increases to 0.78 which is indicative of much better cohesiveness with regards to the grouping of questions. A frequency count of the refined set of questions used to measure the PEOU variable is displayed in Figure 2.

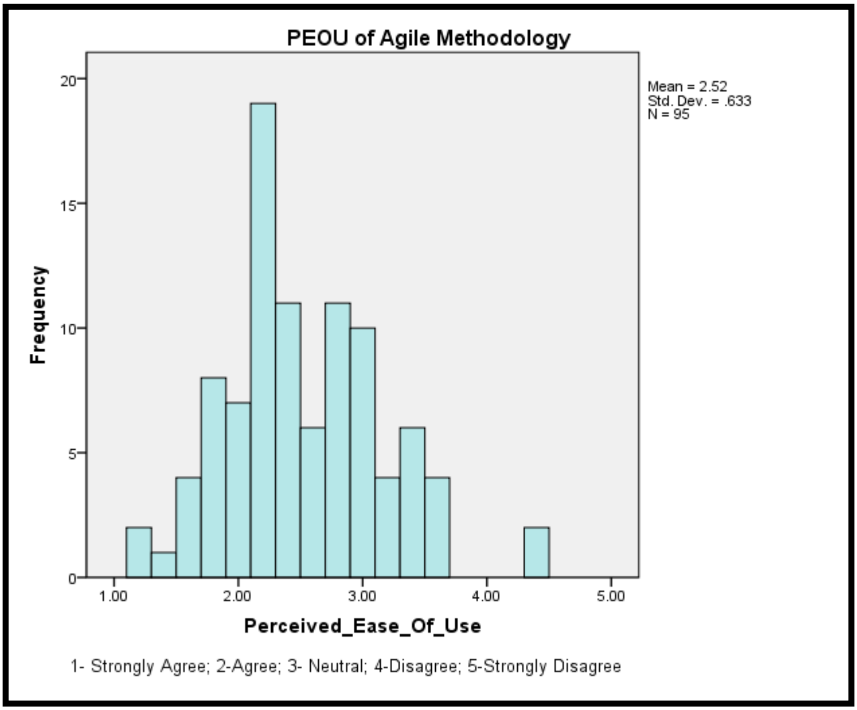

Figure 1: Frequency Count of PEOU of Agile Methodology

The summary data from Figure 2 (mean=2.09; mode=2.0; median=2.0) indicate a higher level of acceptance (88% of the responses were below 3 and 74% of the responses were below 2.3) of the ease of using agile methodology.

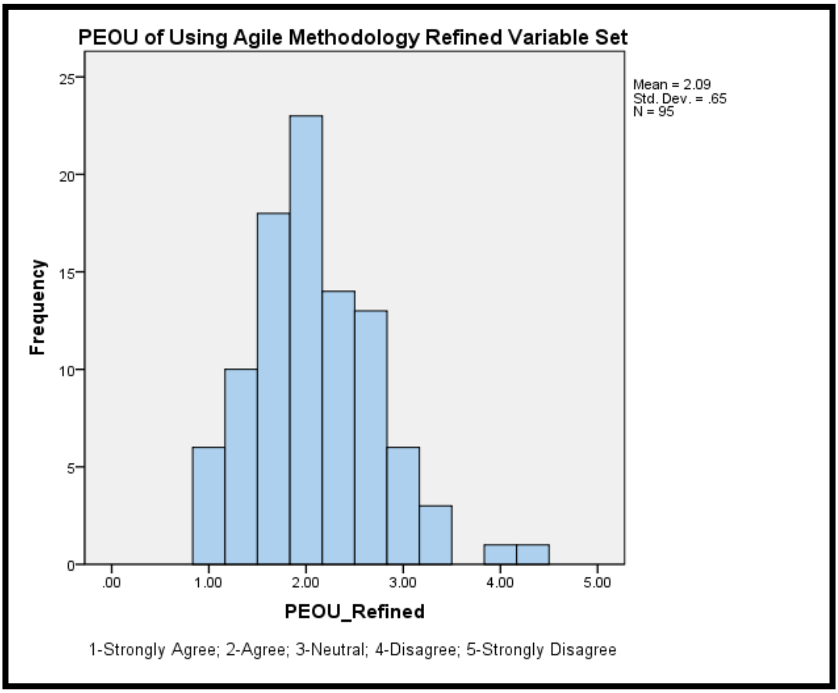

Figure 2: Frequency Count of PEOU with of Agile Methodology Using a Reduced Variable Set

The PU variable was operationalised using a strategy similar to the one used for the operationalization of the PEOU variable. A set of 5 questions was used to derive a quantitative value for PU. The Cronbach alpha value obtained was 0.71 and fell within the acceptable range alluded to by Sekaran and Bougie. The histogram representing the PU variable is shown in Figure 3.

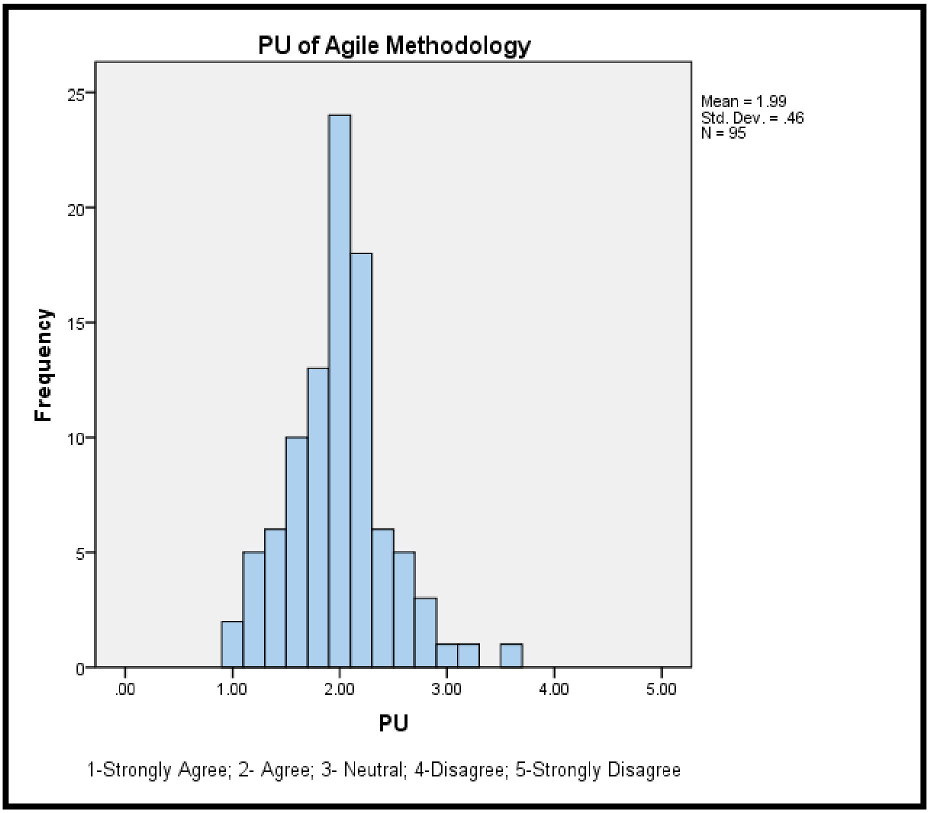

Figure 3: Frequency Count of Acceptance of the Perceived Usefulness of Agile Methodology

The summary data from Figure 3 (mean=1.99; mode=2.0; median=2.0) indicate a high level of endorsement (97% of the responses were below 3 and 82% of the responses were below 2.3) of the usability of agile methodology.

In order to operationalise the level of engagement with agile methodology, students were required to provide a response with respect to their participation in "agile activities", more specifically elements that underpinned XP. This included aspects such as frequency of participation in group meetings, frequency of involvement with identification and development of user stories, frequency of meetings with the system client/ business owner as well as the frequency of participation in pair programming. These frequency values were summed and expressed as percentage values (illustrated in Figure 4) reflecting the students' level of engagement with XP concepts.

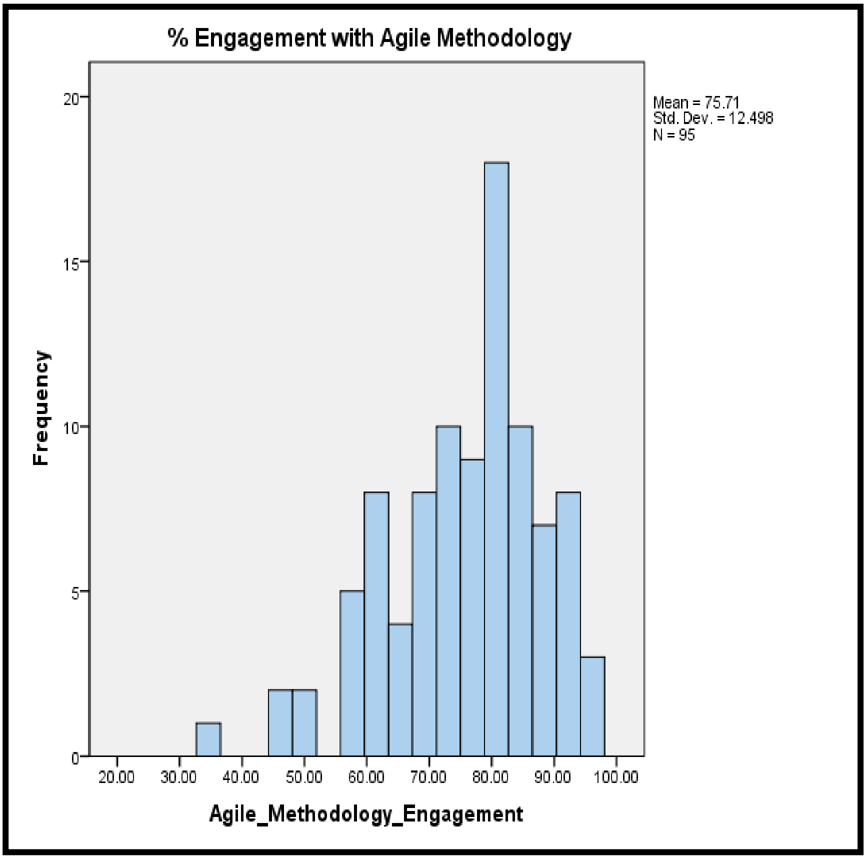

Figure 4: Level of Engagement with Agile Methodology

The median value of 77% as well as negatively skewed distribution illustrated in Figure 4 is indicative of a high level of engagement with the elements of XP.

Discussion and Conclusion

In this article, we have contextualised the objectives of this study by providing a justification for the research topic. It is envisaged that IS researchers will make an effort to leverage off the wider range of IS research topics that will become available as a consequence of the strategy of integrating the 2 distinct IS research paradigms of behavioural and design science. Whilst the current research agenda has a dominantly explorative demeanour, the empirical evidence provided suggest that the outcome of such research efforts can be beneficial to the IS academic and practitioner community.

The literature review as well as the data collection and analysis efforts have jointly contributed towards the provision of a solution, within the framework of this study, to the research problems that were identified at the outset. This assertion is corroborated by the summary provided in Table 2.

Table 2: Summary of Research Problems and Outcomes

Research Question	Research Method	Research Outcome
What does the agile methodology entail?	Literature Review	A definition and listing of main characteristics of the agile methodology
What are the pedagogical challenges of implementing the agile methodology?	Literature Review	A strategy is required to discourage “cowboy style coding”(Wellington 2005); A strategy is required to facilitate the internalisation of agile processes (Sewchurran 2007)

What is the students' level of acceptance of Agile Methodology?	Quantitative data collection and analysis	A high level of acceptance of agile methodology is reported on the basis of the data analysis (74% acceptance of PEOU and 82% acceptance of PU)
How well did the students comply with the requirements of XP?	Quantitative data collection and analysis	A high level of compliance is reported (An average engagement level of 75% is reported)

Table 2 provides an overview of the outcome of this study as well as an indicator that agile methodology will be endorsed as a successful process model for software development. However, the areas of concern, as highlighted by inconsistent data responses, are that of not adopting a "big design up front" (BDUF) as well as constant database re-factoring in order to accommodate customer requirements that may have changed during the system development process. With regards to the BDUF issue, a possible source of rationalisation lies in the approach that is adopted in teaching the systems development process at IS undergraduate level. The traditional "offering" consists of systems analysis and design that is delivered as part of the Systems Development Life Cycle (SDLC) through prescribed texts such as those written by Satzinger, Jackson and Burd as well as Bentley and Whitten. After having been accustomed to the routine of having a BDUF for an entire year, the notion of not starting with comprehensive design models for the system will create a "disorienting moment" (Hughes 2008) thereby resulting in a response that may be inconsistent with the other responses provided. The second inconsistent response emanates from the concern that any re-factoring of a software system will generate regression errors (Schach 2008; Sommerville 2007; Mens & Tourwé 2004) that may be difficult to resolve. The .Net framework also implements a "disconnected" data architecture that creates a memory resident "snapshot" of the database. Any change to the database structure will require re-generation of the memory resident copy of the database as well as re-coding of data structures designed to facilitate database processing. Hence, the strategy of database re-factoring will invariably receive mixed responses from the student cohort. The anomalous responses regarding the strategy of adopting an evolutionary modelling style as opposed to implementing a BDUF strategy as well as the whole issue of constant database re-factoring and the impact it has on system success is a viable area for future research concerning agile methodology.

From an overview perspective, the biggest challenge of implementing an agile approach towards systems development lies in the behavioural realm. The "lightweight" and flexible nature of the agile approach could be perceived as an opportunity to trivialise the methodological component of agile methodology in favour of development practice that is not "plan-driven" (such as "cowboy" style coding) under the "banner" of agile methodology. Whilst these remarks have been made on the basis of the literature review and the empirical evidence that was reviewed as part of the current study, they have also been endorsed in an interview that the researchers conducted with IBM Research Fellow, Grady Booch (Skype interview, June 11th 2012) where he emphasised the "socio-technical" nature of agile methodology. The term "socio-technical" is a reference to the recognition/incorporation of the behavioural traits of the software development team towards the technical aspects of agile methodology as a critical success factor in determining the success of an IS project developed using an agile approach. Hence, while many research efforts may attempt to quantify the success of the agile approach towards software development, it is equally important to ascertain whether the software development team adhered to the principles of agile methodology before the methodology itself is evaluated.

References

Applefield, JM, R Huber & M Moallem 2000. Constructivism in Theory and Practice: Toward a Better Understanding. *The High School Journal* 84: 35 - 53.

Basili, VR & AJ Turner 1975. Iterative Enhancement: A Practical Technique for Software Development. *IEEE Transactions on Software Engineering* 1: 390 - 396.

Beck, K 1999. Embracing Change with Extreme Programming. *Computer* 32**,** 70 - 77.

Beck, K 2008. Tools for Agility. *Microsoft White Papers* June: 1 -12.

Beck, K, M Beedle, AV Bennekum, A Cockburn, W Cunningham, M Fowler, J Grenning, J Highsmith, A Hunt & R Jeffries 2001. *Agile Manifesto.* Available at: http://agilemanifesto.org/.

Benbasat, I. & RW Zmud 2003. The Identity Crisis within the IS Discipline: Defining and Communicating the Discipline's Core Properties. *MIS Quarterly* 27,2: 183 - 194.

Bergin, J, J Caristi, Y Dubinsky, O Hazzan & L Williams 2004. *Teaching Software Development Methods: The Case of Extreme Programming.* Paper presented at the ACM. SIGCSE Bulletin.

Chan, FKY & JYL Thong 2009. Acceptance of Agile Methodologies: A Critical Review and Conceptual Framework. *Decision Support Systems* 46: 803 - 814.

Cohn, M 2012. *Agile Succeeds Three Times More Often Than Waterfall*. Available at: http://www.mountaingoatsoftware.com/blog/agile-succeeds -three-times-more-often-than-waterfall. (Accessed in July 2012.)

Davenport, TH & ML Markus 1999. Rigor vs. Relevance Revisited: Response to Benbasat and Zmud. *MIS Quarterly* 23,1: 19 - 23.

Davis, FD 1985. *A Technology Acceptance Model for Empirically Testing New End-user Information Systems: Theory and Results.* Massachusetts: Institute of Technology, Sloan School of Management.

De Villiers, MRR, S Lubbe & R Klopper 2007. Action Research: The Participative Researcher or Experiential Approach 1. *Alter*nation 14,1: 218 - 242.

Delone, WH. & ER McLean 2004. Measuring e-Commerce Success: Applying the DeLone & McLean Information Systems Success Model. *International Journal of Electronic Commerce* 9: 31 - 47.

Demuth, B, M Fischer & H Hussmann 2002. Experience in Early and Late Software Engineering Project Courses. *IEEE* 241-248.

Dybå, T & T Dingsøyr. 2008. Empirical Studies of Agile Software Development: A Systematic Review. *Information and Software Technology* 50: 833 - 859.

Fällman, D & A Grönland 2002. Rigor and Relevance Remodeled. *Citeseer* 10-13.

Ferreira, C & J Cohen 2008. Agile Systems Development and Stakeholder Satisfaction: A South African Empirical Study. Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology. Wilderness, South Africa: ACM.

Glass, RL 1994. The Software-research Crisis. *Software IEEE* 11: 42 - 47.

Hevner, AR, ST March, J Park & S Ram 2004. Design Science in Information Systems Research. *MIS Quarterly* 28: 75 - 105.

Holcombe, M & C Thomson 2007. 20 Years of Teaching and 7 Years of Research: Research when you Teach. *Proceedingsof the 3rd South-East European Workshop on Formal Methods*. South-East European Research Centre, Thessaloniki.

Hughes, E 2008. Taking First-Year Students to Court: Disorienting Moments as Catalysts for Change. *Wash UJL & Pol'y* 28: 11.

Hulkko, H & PA Abrahamsson 2005. Multiple Case Study on the Impact of Pair Programming on Product Quality. *ACM* 495-504.

Jami, S & Z Shaikh 2005. Teaching Computer Science Courses Using Extreme Programming (XP) Methodology. *IEEE* 1-6.

Kautz, K 2011. Investigating the Design Process: Participatory Design in Agile Software Development. *Information Technology & People* 24: 217 - 235.

Knight, LV, TA Steinbach & Y Levy 2008. Selecting an Appropriate Publication Outlet: A Comprehensive Model of Journal Selection Criteria for Researchers in a Broad Range of Academic Disciplines. *International Journal of Doctoral Studies* 3: 59 - 79.

Kuechler, B. & V Vaishnavi 2011. Promoting Relevance in IS Research: An Informing System for Design Science Research. *Informing Science: The International Journal of an Emerging Transdiscipline* 14: 125 - 138.

Lee, AS 1999. Rigor and Relevance in MIS Research: Beyond the Approach of Positivism Alone. *MIS Quarterly* 29-33.

Lynch, K, A Heinze & E Scott 2007. Information Technology Team Projects in Higher Education: An International Viewpoint. *Journal of Information Technology Education* 6: 181 - 198.

Mahnič, V 2008. Teaching Information System Technology in Partnership with IT Companies. *Organizacija* 41: 71 - 78.

Mens, T & T Tourwé 2004. A Survey of Software Refactoring. *Software Engineering, IEEE Transactions* 30: 126 - 139.

Microsoft. 2012. *Agile Software Development.* Microsoft. Available at: http://msdn.microsoft.com/en-us/vstudio/gg605177. (Accessed in February 2012).

Niederman, F & ST March 2012. Design Science and the Accumulation of Knowledge in the Information Systems Discipline. *ACM Trans. Manage. Inf. Syst.* 3: 1 - 15.

Parnas, D & P Clements 1985. A Rational Design Process: How and Why to Fake it. *Formal Methods and Software Development* 80 - 100.

Parnas, DL 1994. Software Aging. *IEEE Computer Society Press* 279 - 287.

Pinch, T 2008. Technology and Institutions: Living in a Material World. *Theory and Society* 37: 461 - 483.

Pressman, RS 2010. *Software Engineering: A Practitioner's Approach.* Boston: McGraw-Hill Higher Education.

Rosemann, M. & I Vessey 2008. Toward Improving the Relevance of Information Systems Research to Practice: The Role of Applicability Checks. *Management Information Systems Quarterly* 32: 1.

Royce, WW 1970 Managing the Development of Large Software Systems. *Proceedings of IEEE WESCON* 26:8.

Schach, SR 2008. *Object-oriented Software Engineering.* Boston: McGraw-Hill.

Sekaran, U & R Bougie 2010. *Research Methods for Business: A Skill-building Approach.* Hoboken, NJ & Chichester: John Wiley.

Sewchurran, K 2007. Recovering Student Misunderstanding about Agile Development Practices. *Proceedings of the Computer Science and IT Education Conference, Mauritius. Informing Science Institute* 671-679.

Sewchurran, K, M Eccles & E Scott 2006. *Teaching Software Development to Information Systems Majors Using an Action Research Process.* Cape Town, South Africa: SACLA, 2006.

Shackelford, R, A Mcgettrick, R Sloan, H Topi, G Davies, R Kamali, J Cross, J Impagliazzo, R LeBlanc & B Lunt 2006. Computing Curricula 2005: The Overview Report. *ACM SIGCSE Bulletin* 38: 456 - 457.

Siemens, G 2005. Connectivism: A Learning Theory for the Digital Age. *International Journal of Instructional Technology and Distance Learning* 2: 3 - 10.

Sommerville, I 2007. *Software Engineering*. St. Andrews: Addison-Wesley.
Strode, D & J Clark 2007. Methodology in Software Development Capstone Projects. 20th Annual Conference of the NACCQ, Nelson, NZ.
Van Loggerenberg, J 2007. Those Who Can, Do, and Those Who Cannot, Teach. *Alter*nation 14,1: 277 - 291.
Van Valkenhoef, G, T Tervonen, B De Brock & D Postmus 2011. Quantitative Release Planning in Extreme Programming. *Information and Software Technology* 11: 1227 - 1235.
Vanhanen, J & H Korpi 2007. Experiences of Using Pair Programming in an Agile Project. *IEEE* 274b -274b.
Versionone 2011. *State of Agile Development*. Available at: http://www.versionone.com/state_of_agile_development_survey/11/.
Vijayasarathy, L & D Turk 2008. Agile Software Development: A Survey of Early Adopters. *Journal of Information Technology Management* 19: 1-8.
Wellington, CA 2005 Managing a Project Course Using Extreme Programming . *IEEE* T3G-1.
Wieringa, R & A Moralı, 2012. Technical Action Research as a Validation Method in Information Systems Design Science. *Design Science Research in Information Systems. Advances in Theory and Practice* 220 - 238.
Worrall, L, S Lubbe & R Klopper 2007. Academic Research and Management Practice: Is the Relevance Gap Closing? *Alter*nation 14: 292 - 316.

Sanjay Ranjeeth
Department of Information Systems & Technology
University of KwaZulu-Natal
Pietermaritzburg, South Africa
ranjeeths@ukzn.ac.za

Ashley Marimuthu
Department of Information Systems & Technology
University of KwaZulu-Natal
Durban, South Africa
marimuthum@ukzn.ac.za

Manoj Maharaj
Department of Information Systems & Technology
University of KwaZulu-Natal
Durban, South Africa
maharajms@ukzn.ac.za