University for the Common Good

Software systems engineering: a journey to contemporary agile and beyond, do people matter?

AlQaisi, Raid; Gray, Eddie; Steves, Bonnie

Published in:
Software Quality Management XXV

Publication date:
2017

Document Version
Peer reviewed version

Link to publication in ResearchOnline

Citation for published version (Harvard):
AlQaisi, R, Gray, E & Steves, B 2017, Software systems engineering: a journey to contemporary agile and beyond, do people matter? in P Marchbank, M Ross & G Staples (eds), *Software Quality Management XXV: Achieving Software Quality in Development and Use.* British Computer Society, pp. 159-173.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
If you believe that this document breaches copyright please view our takedown policy at https://edshare.gcu.ac.uk/id/eprint/5179 for details of how to contact us.

Download date: 29. Apr. 2020

Software Systems Engineering: A Journey to Contemporary Agile and Beyond, Do People Matter?

Raid AlQaisi[1], Eddie Gray[1], Bonnie Steves[2]

[1]School of Engineering and Built Environment, Glasgow Caledonian University, Glasgow, G4 0BA, UK.
raid.alqaisi@gcu.ac.uk, e.gray@gcu.ac.uk

[2]The Graduate School, Glasgow Caledonian University, Glasgow, G4 0BA, UK.
b.steves@gcu.ac.uk

Abstract

This paper explores the evolutionary journey of the approaches and processes of Software Systems Engineering (SWSE), from the Traditional methods to Contemporary Agile methods and looks at elements and issues of importance for future development. It does this through a comprehensive search of the literature and review of the findings of an empirical study "*Echoes from the Field*" [1,2], which investigates the common practices and issues within the SWSE industry. Particularly, the paper identifies that the human-element is common across all the different SWSE approaches and processes current and past, and that people and their behaviour can have a significant effect on the success of the SWSE endeavour. In the literature, there is a focus on developing the approaches of SWSE to gain a successful outcome, yet very little focus on modelling the effect of human behaviour on the outcome. To understand the human-element effect in SWSE, established theories of human behaviour from the management and psychology disciplines, such as the Toyota Production System (TPS) or Lean, Knowledge Creation Theory (KCT) and Emotional Intelligence (EI), are adapted. The paper concludes that it is not really the approach or process that is the key to success, but rather it is the people who use these approaches that can make the approach succeed or fail. Finally, one of the authors reflects on his experiences of the changes in SWSE since he attended the first Software Quality Management (SQM) conference 25 years ago.

Keywords: Human-Element, Agile, Lean, Emotional Intelligence (EI), and the Knowledge Creation Theory (KCT).

1.0 Introduction

It is intriguing to view the evolution of Software Systems Engineering (SWSE) over the decades, from its early beginnings in the 1950's to its current state. At first glance, it appears that the various processes and approaches to software systems development and maintenance are quite different from each other. Each SWSE process has different organisation of the four common basic process activities: specification, development, validation, and evolution, as defined by Sommerville [3]. Yet some of the issues with the different SWSE approaches are common across the different approaches, both past and current.

Such issues are almost constant in their existence across all types of software systems development processes over the past 60 years and are most often human-related. It is recognised in the literature, that the human-element in software development plays an integral part in the success of the software systems development endeavour. SWSE is, after all, a human-centric craft [1]. Cockburn [4] highlights that the fundamental characteristics of "people" have a first-order effect on software development, not a lower-order effect. Such a view on the importance of the human-element in SWSE was noticed, as early as 1971, in Weinberg's [5] book. What is fascinating is that despite this recognition of the "people" factor, developments of new approaches remain focused on the process, rather than on the people. Sadly, this is the prominent view in the field, continuing to seek the silver bullet solution through the development of process [6]. This paper seeks to understand the effect of the human-element on the success of the software development and answer the question, "*Do people really matter?*"

Section 2.0 describes the evolution of SWSE approaches and processes over the last 60 years and highlights some of the common features and issues. Section 3.0 tackles how SWSE can be seen as a human-centric activity. Section 4.0 discusses some of the empirical findings of the research project "*Echoes from the Field*" [1,2] which explores with SWSE practitioners the common practices and issues found within the industry. It describes three key themes relating to people that affect SWSE practice and outcome. To understand the human-related issues, knowledge of the SWSE discipline is brought together with knowledge from other disciplines such as the Japanese management theories of Toyota Production System (TPS) also known as Lean, the Knowledge Creation Theory (KCT) and Emotional Intelligence (EI) theory from psychology [1]. Section 5.0 describes these theories and how they model the effects of human behaviour on people's creative work.

Insight into the human-element effect in SWSE is drawn from adapting the related theories of TPS, KCT and EI to software systems creation, in Section 6.0. It is here that the question in the paper's title, "*Do people matter?*" is answered. A deliberately contradictive viewpoint of the standard Agile vs Traditional approaches case is provided, with the inclusion of the human-element which is the common link in all processes and approaches. The paper finishes with the conclusions in Section 7.0. In a special Reflective Coda (Section 8.0), one of authors, Mr Eddie Gray, gives his reflections on the commonalities and changes in SWSE since he attended the first Software Quality Management (SQM) conference 25 years ago.

2.0 Software Systems Engineering – Journey of Development

According to the literature, the journey of the development of SWSE began with what is known as the Traditional approaches such as code and fix [7], then moved on to the first published software systems development approach known as the phased concept in 1956 by Benington [8], followed by the well-known Waterfall approach by Royce [9] in 1970.

With the need for delivering complex software systems to meet requirements within time and budget constraints, various processes and approaches evolved such as the Spiral Model [7] in 1988 and the Dynamic Systems Development Methods (DSDM) in 1994 [10,11]. Such processes and approaches evolved to overcome the weaknesses of previous processes/approaches, to cope with the rapid evolution of systems, stakeholder demands, technology, and continuously changing environments [12].

From 2001 new approaches to software development began to appear in the literature, known as the Contemporary Agile approaches, that followed the Manifesto for Agile Software Development of 2001 [13]. Examples include Scrum [14], eXtreme Programming (XP) [15], Lean Software Development [16], and Disciplined Agile Delivery (DAD) [17]. In theory, the use of the Agile approach, enables individuals in the development teams to be empowered, more trusted, more liberated in their task selection, and most importantly, consulted in the software systems development process. Additionally, unnecessary documentation is reduced to the minimum [1,13].

What is interesting is the appearance of various methods before and after 2001 that propose a different view from either Agile or Traditional. For example, the Gilb's Evolutionary (Evo) [18] software development method was proposed ahead of the Manifesto for Agile of 2001 as early as 1988. It focuses on early delivery of high value to stakeholders by promoting incremental iterative software development with an emphasis on clear requirements and delivering value [1]. Another example, in 2003 was the suggested balance of both Traditional and Agile approaches for a particular project in a particular organisation using a risk management tool [11].

One interesting recent change is the evolution of the Spiral Model into the Incremental Commitment Spiral Model (ICSM) in 2014 [19]. The ICSM design is based on the old spiral model design, but with the inclusion of symbolising systems thinking, best practices, and engineering practices. It covers the full system development lifecycle, starting from the exploration phase to the operation phase.

It has the aim of better integrating hardware, software, and human-interaction to cope with the rapid pace of change and to add value for the system users [1,19].

It can be argued that all processes and approaches have more or less the same principles and aims, *i.e.* to produce software systems on time, within budget, with minimal errors, that meet the requirements and add value to stakeholders. However, they differ in their operation [1,3,20,21].

Both Traditional and Agile approaches have successes and failures in delivering software systems. For example, both Traditional and Agile have the failures of technical-debt and scope-creep [1,22]. There is no silver bullet [6] solution; however, software systems engineers are advised to search and find the best silver bullet to develop a certain software system in a certain time and context [1].

A review of the literature shows that most, if not all, publications are proponents of certain approaches trying to advocate the use of a particular approach over other approaches. For example, Sutherland [23] in his book about Scrum claims that "*The way we work is wrong; this is the solution.*" [23, Back cover]. Whereas, AlQaisi [1] argues, along the ideas of Cockburn (2000), … *[Software Engineering] SWE approaches generally try to approach software development on an abstract level view at the process level, rather than at the holistic level to consider the wider sociotechnical environment. Even when taking the holistic view, the focus remains on the mechanistic SWE approach as a first order component rather than having the people as a first order component.*"[1 p34].

The one common element between all methods is the human-element [1]. As can be noticed from the literature, there is still a great deal of focus on the process/approach view and how to implement it, with no direct consideration, at least explicitly on the human-element dynamics [1]. The following section discusses how SWSE is human-centric.

3.0 Software Systems Engineering is Human-Centric

Software systems engineers are the innovators and creators of the software systems. Software development is human-centric and highly dependent on the human knowledge, innovation, and judgement, and to a large extent is influenced by the psychology and emotions of the engineers [1,2,24]. The findings of the empirical study "*Echoes from the Field*" [1] indicate that the human-centric view is especially important when using the Agile approach for software systems development. Yet, the literature on SWSE has only limited publications researching into the importance of the human-element in software systems development [1,2,25,26,27].

As early as 1971, Weinberg [5] argues the importance of the psychology of the engineers in software systems development. For example, he describes how social factors cause different levels of performance amongst software systems engineers.

Another example is the effect of the environment on individuals and teams. What is most interesting is that his speculations and ideas are still valid and are true reflections of contemporary SWSE practices and deficiencies [1]. Boehm in 1991[28] discusses how people and their initiative make projects succeed "*Good people, with good skills and good judgment, are what make projects work.*" [28 p.41]. Cockburn in 2000 [4] characterises software systems engineers (People) as a first order non-linear component in software systems development before the process/approach. He argues that it is not the methodological equation of how to develop software, it is the effect of "People" on methodologies [1,4]. DeMarco and Lister 1988-2013 [25] discuss how major issues in software development are human related, such as false assumptions, poor specification of requirements, and lack of top management support, not technical. Therefore, such issues are complicated and hard to solve [1]. Nevertheless, identifying, acknowledging, and dealing with them increases the chances of success. The human-side appears to matter much more than the technical side for a successful outcome [1,2,4,25].

In 2001 Constantine [26] highlights that change in people's behaviour is slower than technological change. Consequently, the people side of technology shows almost stable issues compared to the issues of technological change. He argues that projects are still running late, over budget, and not delivering the required software system, where people are still facing the same organisational and cultural challenges. Good software, successfully delivered comes from the people, not from the technical tools. [1,26].

In 2012 Broza [27] discusses the human-side of Agile, and how to build the personal leadership skills of the Agile team. One interesting idea by Broza is his model of the evolution of the 3P's (Product, People, and Process). He states the focus on the three P's differs through time. For example, in the early manufacturing theories, the most important P was the process; the second most important P was the product, leaving the people the least important of all. Conversely, in the contemporary Agile software development, the focus has shifted the order of importance of the 3P's for successful outcome to (People, Product, and Process) [1,27]. The argument is that focusing on the process and product over people does not work in a knowledge based and therefore people based field such as SWSE.

The following section will discuss some of the findings of the empirical study "*Echoes from the Field*" based on investigation of the current practices in the SWSE industry.

4.0 Brief Discussion of Some Findings from the "Echoes from the Field" Study

This section summarises some related findings from the empirical study "*Echoes from the Field*". Further details on the study design and implementation can be found in [1,2,24]. Figure 1 shows the thematic map of the inductive thematic analysis of the findings.

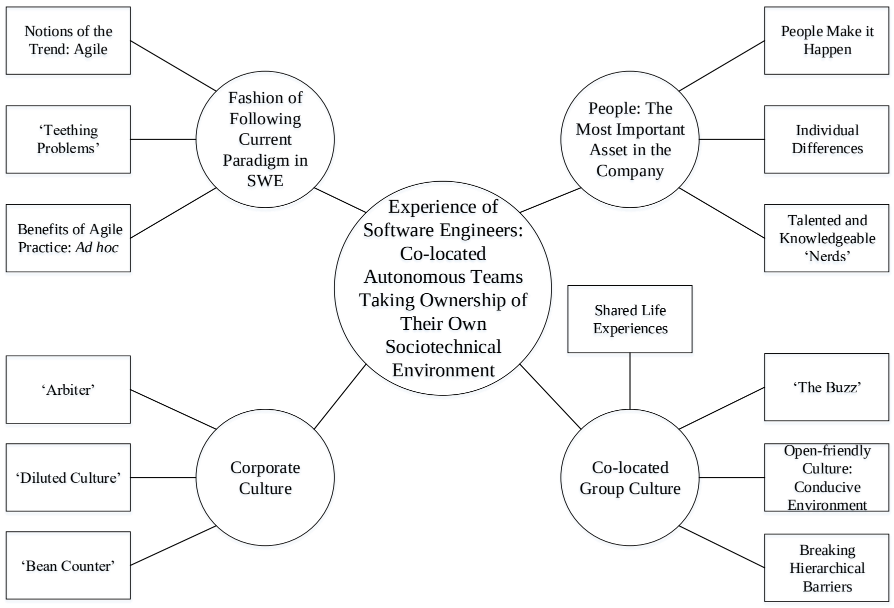
Notions of the Trend: Agile
'Teething Problems'
Benefits of Agile Practice: *Ad hoc*
Fashion of Following Current Paradigm in SWE
People Make it Happen
Individual Differences
Talented and Knowledgeable 'Nerds'
People: The Most Important Asset in the Company
Experience of Software Engineers: Co-located Autonomous Teams Taking Ownership of Their Own Sociotechnical Environment
Shared Life Experiences
'Arbiter'
'Diluted Culture'
'Bean Counter'
Corporate Culture
'The Buzz'
Open-friendly Culture: Conducive Environment
Breaking Hierarchical Barriers
Co-located Group Culture

Figure 1: The Thematic Map of the Inductive Thematic Analysis from the Study "Echoes from the Field" [1, p171].

The study [1] showed there is a gap between real-life experience and SWSE theories. Most of the current software development methodologies/processes focus on the technical processes rather than on the people who implement and execute these processes, with the assumption that the processes will always be implemented accurately by the people. Such a view neglects the fact that people are different as well as emotionally and psychologically influenced [1]. Thus, process implementation varies between individuals, teams, and organisations; as well as across time with the same individuals, teams, and organisations. This is an important consideration when implementing and practicing SWSE [1].

In the study [1], software systems engineers describe their experience of practicing SWSE as being members of co-located autonomous teams taking ownership of their own environment. The following three key themes relating to people were identified to affect SWSE practice and outcome: 1) Teams work autonomously 2) Varying depth of the implementation of Agile 3) Apprenticeship style knowledge exchange and training.

Theme 1: "**Teams Work Autonomously**". Most of the software systems cannot be delivered by any one person single-handedly. The successful delivery of a software project requires the collaborative group effort of a project team [1]. The participants in the study preferred to work within an autonomous team with ownership, strong internal bonds and high level of synergy as well as lively group dynamics. Such autonomous teams were suggested to be both more Agile and more productive. Some participants described that at the team level, there are more informal and *ad hoc* communications. Some argued that using Agile as a development approach increased group dynamics, team autonomy, synergy, and team unity; whereas with the old approach, namely Waterfall, the environment felt like everyone was competing against one another [1].

Theme 2: "**Varying Depth of the Implementation of Agile**". An interesting aspect of the experience of software systems engineers practicing Agile SWSE, was the confusion between the theory and practice of the Agile development approaches. For instance, some teams would apply Agile close to the theory, whereas others found this challenging.

It is interesting to note that implementing and practicing Agile methods differs between organisations as well. For example, organisation Alpha in the study [1, 2] is an older company that has moved to Agile from an older traditional SWSE approach such as the Waterfall. Whereas, organisation Beta is a young start-up company with slightly less experienced software engineers compared to Alpha. The younger software engineers at Beta started software development with Agile mainly using Scrum. As a result of their different history, Alpha had more issues implementing Agile than Beta did; however, both organisations preferred Agile over other processes/approaches. Both organisations perceived the results of using Agile as a success [1].

Theme 3: "**Apprenticeship Style Knowledge Exchange and Training**". According to the participants, autonomous teams were generally based on co-locating within 'physical proximity', *i.e.* at the same physical space on the same site [1]. This co-location resulted in a rich interaction, both formal and informal, between people that was mostly face-to-face. Such an environment is conducive to learning and exchanging knowledge between individuals. A related matter pointed out by the participants was the effect of the training budget-cuts on the group learning. Interestingly, software systems engineers and managers found solutions to the situation, by being creative and practising a form of apprenticeship that shares knowledge and experience together. In the example provided by the participants, junior team members shadowed more experienced team members to gain their training and learning [1].

The following section describes theories from related disciplines that can be adapted to improve SWSE practices.

5.0 Related Disciplines and Adapted Theories

It has been shown that the human-element of SWSE should be treated as of first order importance when dealing with software development and maintenance

[1,4,5]. Yet, in the SWSE literature, there are no theoretical explanations or justifications of why the human-element potentially helps success or causes failure.

Fortunately, there are well-established theories in the fields of industrial production, management and psychology that can provide greater understanding of the complexity of human behaviour and can help explain the importance of the human-element in SWSE development. What follows is a brief summary of the most relevant theories.

5.1 Toyota Production System (Lean)

Known as Lean or the 'Toyota Production System' (TPS) [29], its adoption and use is wide spread around the globe in various industries and businesses. According to Liker [30], the "Toyota Way" has what is called the 4P's: Philosophy, People/Partners, Process and Problem solving; for organisational excellence and for creating learning enterprise [30,31]. Toyota culture is based on "human systems" that are put in place to infuse its founding principles of excellence, mutual prosperity, and trust with all people inside and outside the company. In brief, people are the heart and soul of the "Toyota Way". The core of TPS is the relentless pursuit to eliminate waste or "Muda" [29,30]. Toyota invests long term in systems of people, technology, and processes that combine to realise high customer value. TPS can be summarised as a culture of respecting people, continuous improvement (Kaizen) and eliminating waste. It is a system that is designed to provide people with the tools they need to continually improve their work [30]. Toyota is serious about long-term thinking philosophy. The Toyota problem solving process is based on the Deming's Plan-Do-Study-Act (PDSA) cycle [29,30,32].

5.2 Knowledge Creation Theory

The Knowledge Creation Theory (KCT) by Takeuchi and Nonaka [33] is based on the notion of that knowledge is created and can be passed on by establishing trusted autonomous teams with cross-functional areas of expertise. Takeuchi and Nonaka state that knowledge creation should be at the centre of the organisation's human resources "human-element" strategy. Companies are not machines, but living organisms that have their own fundamental purpose, collective sense of identity, and their own self-knowledge that shapes them [1]. Furthermore, Nonaka [34] describes two types of human knowledge. The first one is the explicit hard systematic objective and formal 'quantifiable' type that can be articulated in formal language *i.e.* codified in the procedures. The second type is the tacit subjective knowledge that is highly subjective and depends on the hunches, insights, and intuitions of the employees in the organisations. The second type is hard to articulate in formal language. Nonaka and Takeuchi [35] state that the interaction between tacit and explicit knowledge is the key dynamic of knowledge creation in business organisations. They argue that organisational knowledge creation is a spiral process in which tacit knowledge interacts with explicit knowledge repeatedly, creating the knowledge spiral.

5.3 Emotional Intelligence

Goleman [36] first fashioned the theory and term of Emotional Intelligence (EI) in his book entitled "*Emotional Intelligence: Why It Can Matter More Than IQ*". He claimed that EI ability counted for 66% of the ability needed for all jobs, and 85% for leadership jobs. This concept was supported by research output from the industry [37]. EI can be defined as the ability to recognise emotions; "feelings" of oneself and those of others, for self-motivation, and for managing emotions well in oneself and in handling relationships [36]. EI encompasses personal (individual level) competencies that include self-awareness, self-regulation, and self-motivation; as well as, group (social) competencies that include empathy, people (social) skills, and handling relationships [1,36].

6.0 Understanding the Human-Element Effect in Software Systems Engineering Using Related Theories

In his PhD thesis, AlQaisi [1] took the established theories of human behaviour from other disciplines, namely TPS/Lean, KCT and EI, and adapted them in order to explain and provide a deeper understanding of the effects of the human-element on SWSE.

According to the findings of his study [1], software development practices require a high level of EI as in any other activities involving people. For example, software engineers described their feelings of excitement, motivation, frustration, confusion, and sarcasm; and how these related to practice, environment, the surrounding people, policies, freedom to create, and restrictions [1].

Emotions management and EI theory are important aspects to consider when understanding the effect of human behaviour and interaction in SWSE. SWSE is not an individual effort. It is a collective effort of a team of software engineers working and interacting together to achieve a common goal of developing successful software systems. The participating CEO of organisation Beta commented on how software engineers are very emotional people, the opposite of what people usually think about software engineers [1]. People are different and vary in their interaction and communications dynamics.

Understanding the variety of these human dynamics is vital for successful teams. The people dynamics and interactions are an integral part of the human behaviour theories KCT, TPS, and EI. People, including teams, interact together to create and share knowledge. Software systems development is a creative and a knowledge based field. Therefore understanding the effects of human behaviour on that knowledge creation using the theories of KCT, TPS and EI might explain why the Traditional and Agile approaches have both strengths and weaknesses that are related to the people factor which according to the studies outlined above is shown to be more important than the process factor.

Generally, in the SWSE literature, one typically finds listing of the strengths of the Agile approaches in comparison with listing of the weaknesses of the Traditional approaches. As such, all the traits focus on the process. Accordingly, in Figure 2 it is intentional that the diagram is constructed with an alternative focus, indicating that both Agile and Traditional approaches sometimes succeed and sometimes fail and that Traditional approaches have strengths and Agile approaches also have weaknesses. The key element in identifying routes to success is factoring in the middle Human-Element of Figure 2.

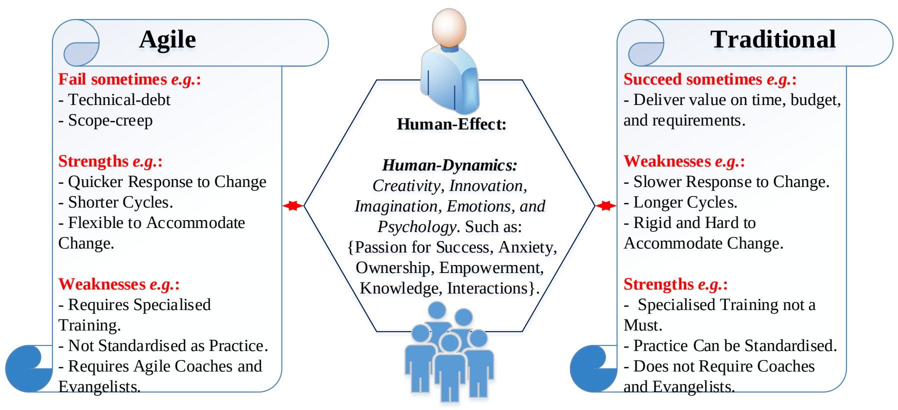
Agile
Fail sometimes e.g.:
- Technical-debt
- Scope-creep
Strengths e.g.:
- Quicker Response to Change
- Shorter Cycles.
- Flexible to Accommodate Change.
Weaknesses e.g.:
- Requires Specialised Training.
- Not Standardised as Practice.
- Requires Agile Coaches and Evangelists.
Human-Effect:
Human-Dynamics:
Creativity, Innovation, Imagination, Emotions, and Psychology. Such as:
{Passion for Success, Anxiety, Ownership, Empowerment, Knowledge, Interactions}.
Traditional
Succeed sometimes e.g.:
- Deliver value on time, budget, and requirements.
Weaknesses e.g.:
- Slower Response to Change.
- Longer Cycles.
- Rigid and Hard to Accommodate Change.
Strengths e.g.:
- Specialised Training not a Must.
- Practice Can be Standardised.
- Does not Require Coaches and Evangelists.

Figure 2: Agile versus Traditional View: Human-Effect within Software Systems Engineering.

Raising the consideration of the contradictive view as in Figure 2 is the intention of this paper. All processes and approaches have failures and successes; however, the common link is the people who create the potential for the success or failure of implementing any software systems development approach. What is interesting is that a simple literature search for Agile software systems project failure did not result in papers discussing such an important matter. Returning to the 3P's example by Broza [27] in Section 3.0, Agile approaches shifted the order of the 3P's to focus on people as in (People, Product, Process). However, in practice and in the literature according to the findings of the study [1,2], the focus is still on the process as a first order P. All evidence suggests that focusing on the process and product over people does not work in such a creative and knowledge-based field as SWSE.

7.0 Conclusion

It can be argued that Software Systems Engineering (SWSE) can be considered as an interdisciplinary complex field that is heavily reliant on the human innovation, creativity, and imagination, and affected by the psychology and emotions of the software engineers [1]. The traditional view in the software engineering field, with software development approaches such as Agile and the Waterfall for example, is focused on the technical side. However, as discussed in [1,2,24] that narrowly focused view should be expanded to encompass the wider sociotechnical environment that includes not only the technical side, but also the human social side. By focusing only on the technical process, the context in which SWSE is practiced is missing. SWSE is human-dependent, and thus is highly reliant on the context [1]. The human-element dimension is central to SWSE and its successful practice.

All approaches and processes can potentially succeed or fail depending on people. It is not about Agile, nor the Traditional approaches, but rather the people who practice and use these approaches. Thus, the success of any approach, regardless of being Agile or Traditional is based on the software systems engineers - "People". According to the findings of the study "*Echoes from the Field*" [1], it can be argued that it is important to acknowledge the human-element of SWSE as a first order component of successful practice and development.

Software systems engineers preferred as they described: to be co-located within autonomous teams that take ownership of their own sociotechnical environment and SWSE approach. Once people are trusted and have a sense of ownership, they will do their best to make it happen. So, it is the people who choose the appropriate process to develop a particular system, in a particular time, within a particular context; where the process could be changed across time and context. It is therefore the people's ability to shape their sociotechnical environment and approach that makes the approach or process successful.

8.0 Reflective Coda by Eddie Gray

I am in a privileged position to have attended not only the 1st International Conference on SQM, in 1993 as a presenter but many more over the 25 years to the present day and can therefore reflect on past and recent trends and identify what has faded away and what has become more "popular". The cover of the Proceedings of the 1st International Conference on SQM, 1993 pretty much summed up the "fashion" or state-of-the-art in SQM in 1993 with its emphasis on standards for software quality and software process improvement, ranging from ISO0 9001 QMS through CMM to TickIT and TQM. Over the years these have all transitioned respectively to ISO 9001:2015, TickITplus and CMMI. How and why has this transition happened?

8.1 Into the Seventh Decade of Software Process Improvement Activity

Solomon says "*There is nothing new under the sun*".

Software process improvement is not a new idea or aim. The first reference to improving the programming process appeared in 1951 [38]. To date there has been an intensive 66 year effort to improve the software process.

Various technical innovations have been introduced in the more recent decades, for example CASE tools, various programming paradigms, and formal methods and so on. The most notable innovations were met with great expectations and various ‘experts’ greeting each in turn as the panacea to solve most problems. Although growth in system size and functional complexity have been facilitated, through the years many techniques and tools have been tried and failed to deliver substantial global improvements to the software process. None have lived up to the real expectations. Technology alone then is clearly not enough.

Since the mid-1980s interest in the software process and its improvement has increased substantially, with the most significant contributing factor being the ongoing efforts of the Software Engineering Institute (SEI) at Carnegie-Mellon University, Pittsburgh, USA. Leading figures in this camp, such as Humphrey [39] and Curtis [40], have been directly responsible for increased industry awareness. Bill Curtis [40] of the SEI spoke of the need for PROCESS, PEOPLE and TOOLS (TECHNOLOGY), as three prongs of a concerted software productivity and quality effort.

Based on experience and evidence over the years, the trend has been to recognise more explicitly that IT systems and software systems are sociotechnical systems - they cannot be understood without a sense of the relationship between the social aspects (organisation and people) and the technical aspects (hardware and software) of the system. The social and the technical aspects of a system are inextricably linked. Although this is a strand of systems thinking that dates back to the early 1950s, it is becoming increasingly recognised, as evidenced in the earlier sections of this paper.

This trend is also reflected in the topic of my presentation at the 1st International Conference on SQM, in 1993, the Capability Maturity Model (CMM) which developed into Capability Maturity Model Integration (CMMI) for Development. The current CMMI model is divided into 22 process areas, of which 5 cover organisational aspects and many of the others cover management aspects.

The CMMI started life in 1987 as the Capability Maturity Model (CMM), a project at SEI. The CMM for Software was first published in 1991 and is based on a checklist of critical success factors in software development projects during the late 1970s and early 1980s. Its success led to the development of CMMs for a variety of subjects beyond software. The proliferation of new models was confusing, so the government funded a two-year project that involved more than 200 industry and academic experts to create a single, extensible framework that integrated systems engineering, software engineering, and product development. The result was CMMI. The most important thing to understand about the CMMI-

DEV is that it is a model. It is not a process or a prescription to be followed. It is a set of organisational behaviours that have proven to be of merit in software development and systems engineering. Why use such a model? What is its purpose? And how best should it be used? These are critical questions and are perhaps the most misunderstood issues with CMMI.

Best practices are intrinsically a matter of opinion since there is no known right way to develop or evolve software properly. When people get together and decide on best practice, it could be viewed that what is agreed upon is synthetic, from imagination to a significant extent, and indeed arising from their opinion(s) of software 'engineering' state-of-the-art in the 1990s. Perhaps it is not the best practice processes choices that are important here for success, so much as that people get together and agree, and are motivated to implement and make a success of their best practice choices.

9.0 References

1 AlQaisi R, Software and Systems Adaptive Value Engineering (2SAVE) Framework Based on the Empirical Study 'Echoes from the Field', PhD Thesis, Glasgow Caledonian University, Glasgow, UK, 2015

2 AlQaisi R, Gray E, Echoes from the Field: An Empirical Study of Contemporary Software Engineering Practices in Some Software Development Organisations in the UK, proceedings of SQM 2013, pp 47-62, London, 2013

3 Sommerville I, Software Engineering, 9th Edition, Addison-Wesley 2011, ISBN-13: 978-0137053469

4 Cockburn A R, Characterizing People as Non-Linear, First-Order Components in Software Development, the 4th International Multi-Conference on Systems, Cybernetics and Informatics, Orlando, Florida, USA, 2000

5 Weinberg G M, The Psychology of Computer Programming, First Edition, Litton Educational Publishing 1971, ISBN-13: 978-0442292645

6 Brooks F P Jr (1987). No Silver Bullet Essence and Accidents of Software Engineering. Computer, 20(4), pp 10-19

7 Boehm B (1988). A Spiral Model of Software Development and Enhancement. Computer, 21(5), pp 61-72

8 Benington H D, (1983). Production of Large Computer Programs, Annals of the History of Computing, 5(4), pp 350-361

9 Royce W, Managing the Development of Large Software Systems, Proceedings of the IEEE/WESCON conference, pp 328-339, USA, 1970

10 Abrahamsson P, Outi S, Ronkainen J, (2002). Agile Software Development Methods. First Edition. Finland: VTT Electonics

11 Boehm B, Turner R B, Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley 2003, ISBN-13: 978-0321186126

12 AlQaisi R, Gray E, Empirical study of unpredictable culture in reality: Delivering value to stakeholders or developing requirements for senior management, Proceedings of the APCOSEC 2013 Systems Engineering annual conference, pp 1-15, Yokohama, Japan, 2013

13 The Agile Alliance. (2001). Manifesto for Agile Software Development. Retrieved 22nd June 2013, from the Agile Alliance: http://agilemanifesto.org/
14 Scrum Alliance. (2010). Scrum Guide by Schwaber K, Sutherland J. Retrieved 19th April 2010, from the Scrum Alliance: https://www.scrumalliance.org/why-scrum/scrum-resources
15 Beck K, Extreme Programming Explained: Embrace Change, First Edition, Addison-Wesley 2000, ISBN-13: 978-0201616415
16 Poppendieck M, Poppendieck T, Lean Software Development An Agile Toolkit, First Edition, Addison-Wesley 2003 ISBN-13: 978-0321150783
17 Ambler S, Lines M, Disciplined Agile Delivery; A Practitioner's Guide to Agile Software Delivery in the Enterprise, First Edition, IBM Press 2012 ISBN-13: 978-0132810135
18 Gilb. (2014). Evo Manuscript Edition by Gilb K T. Retrieved 30th May 2013, from Gilb: www.gilb.com
19 Boehm B, Lane J A, Koolmanojwong S, Turner R, Incremental Commitment Spiral Model: Principles and Practices for Successful Systems and Software, First Edition, Addison Wesley 2014, USA. ISBN-13: 978-0321808226
20 Copeland L, Choosing The Best Of The Plan-Driven and Agile Development Methods, Conference & EXPO Presentation, Better Software Conference & EXPO 2008, Presentation, USA, 2008
21 Gilb T, Competitive Engineering: A Handbook for Systems Engineering Requirements Engineering, and Software Engineering Using Planguage, First Edition, Elsevier 2005, ISBN-13: 978-0750665070
22 Microsoft. (2011). Ten Year Agile Retrospective: How We Can Improve in the Next Ten Years, by Sutherland J. Retrieved 10th September 2014, from Microsoft: http://msdn.microsoft.com/en-us/library/hh350860.aspx
23 Sutherland J, SCRUM: A Revolutionary Approach to Building Teams, Beating Deadlines and Boosting Productivity, First Edition, Crown Publishing Group 2014, ISBN-13: 978-1847941084
24 AlQaisi R, Gray E, Moffat D, Wang B, ‘Echoes From The Field’ Study Outcome and Discussion: Reflections on Software Systems Engineering Practice, Proceedings of the 26th Annual INCOSE International Symposium 2013, pp 1.15, Edinburgh, UK, 2016
25 DeMarco T, Lister T, Peopleware: Productive Projects and Teams, Third Edition, Addison-Wesley 2013, ISBN-13: 978-0321934116
26 Constantine L L, The Peopleware Papers, First Edition, Yourdon Press 2001, ISBN: 978-0130601230
27 Broza G, The Human Side of Agile - How to Help Your Team Deliver, First Edition, 3P Vantage Media 2012, ISBN-13: 978-0988001626
28 Boehm B (1991). Software Risk Management: Principles and Practices, Software, IEEE, 8(1), pp 32-41
29 Ohno T, Toyota Production System: Beyond Large-Scale Production, First Edition, CRC Press 1988, ISBN-13: 978-0915299140
30 Liker J K, The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer, First Edition, McGraw-Hill 2004, ISBN-13: 978-0071392310

31 Liker J K, Hoseus M, Toyota Culture: The Heart and Soul of the Toyota Way, First Edition, McGraw-Hill 2008, ISBN-13: 978-0071492171
32 Moen R D, Norman C L (2010). Circling Back: Clearing up Myths About the Deming Cycle and Seeing How it Keeps Evolving, Quality Progress, 43(11), pp. 22-28
33 Takeuchi H, Nonaka I (1986) The New New Product Development Game, Harvard Business Review, 64(1), pp. 137-146
34 Nonaka I (1991). The Knowledge-Creating Company, Harvard Business Review, 69(6), pp. 96-104
35 Nonaka I, Takeuchi H, The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation, First Edition, Oxford University Press 1995, ISBN-13: 978-0195092691
36 Goleman D, Emotional Intelligence: Why It Can Matter More Than IQ, First Edition, Bloomsbury Publishing 1996, ISBN-13: 978-0747526223
37 Kunnanatt J T (2004). Emotional Intelligence: The New Science of Interpersonal Effectiveness, Human Resource Development Quarterly, 15(4), pp. 489-495
38 Wilkes MV, Wheeler D J, Gill S. The Preparation of Programs for an Electronic Digital Computer, Addison-Wesley 1951, ISBN-13: 9780938228035
39 Humphrey W S, Managing the Software Process, First Edition, SEI series on Software Engineering, Addison Wesley 1989, ISBN-13: 978-0201180954
40 Curtis W, Software process improvement and the superior software organisation, proceedings of the Software Process Modelling in Practice conference 1993, pp. 22-23, London, UK, 1993