Approaches to Software Process Improvement: A State-of-the-Art Review

Ismail Keshta*
Computer Science and Information Systems Department, College of Applied Sciences, AlMaarefa University, Riyadh, Kingdom of Saudi Arabia.

* Corresponding author. Email: imohamed@mcst.edu.sa
Manuscript submitted August 31, 2019; accepted October 31, 2019.
doi: 10.17706/jsw.14.11.519-529

Abstract: This paper introduces the domain of software process improvement (SPI) by including a review of it as an approach that can improve software quality. In addition, it reviews work that has been carried out in the SPI area, as well as problems that can limit how successful SPI initiatives are in various highlighted companies.

Key words: Software quality, software process improvement (SPI), capability maturity model, SPICE, ISO/IEC 15504.

1. Introduction

Software quality has been given considerable attention in the key areas of industry and academia due to its important role in modern-day living and business. The software quality assurance group has embraced a model that consists of a total of 14 quality factors in 3 stages of the development life cycle, namely, performance, design, and adaptation [1]. Even though creating software that adheres to every one of these various factors is difficult, this model provides an excellent frame of reference to comprehend software quality.

A number of software engineering researchers are now paying close attention to the software development process because of the recent rise in the overall importance of software products and the current demand for better software quality within the industry. Continual assessments and constant improvements are required during this process to satisfy the customers' requirements, as well as those of the stakeholders. These various improvements will lead to high-quality software being created by companies. The quality of the software processes that are utilized by companies significantly influences the software product's overall quality [2]. Thus, one of the major challenges for software organizations is to acquire software that is of high-enough quality to meet their customers' requirements [3][4].

Many researchers applied the software process improvement (SPI) concept to focus on software quality [5] [6]. Ashrafi [5], for example, investigated the kind of impact that SPI methodologies had on software quality. García-Mireles et al. [6] stated that when software development companies implement SPI, they try to improve software quality. SPI is seen as a vital aspect of the optimization of the software development process, particularly for small- and medium-sized organizations [7]-[10]. Niazi et al. [11] stated that one of the software industry's biggest challenges is the design of SPI implementation initiatives that will effectively help small- and medium-sized organizations. Research efforts have so far concentrated on the implementation of SPI standards/frameworks to increase software quality and increase productivity [12]-

[45]. Two examples of well-known and established SPI standards are Capability Maturity Model Integration (CMMI) and ISO/IEC 15504.

We will introduce SPI in this paper and review it as an approach that can improve software quality. We will also review the work undertaken in the SPI area and highlight the problems that can affect the success of SPI initiatives in various organizations.

We will organize our paper as follows: Section 2 will provide some background on SPI. Section 3 will identify the most widely accepted approaches to process improvement. Section 4 will highlight the current challenges to SPI. Section 5 will describe our conclusions and future work.

2. Software Process Improvement

Attempts have been made by software companies to improve software quality for a number of decades because of the questionable quality image of many of their products [13]. As a result, these companies have been stretched to a greater degree than ever before, and so the motto of many of them has become customer satisfaction [14]. Software quality is vital as software is becoming increasingly crucial to our daily lives [15], [16]. Most software organizations struggle to provide quality software on schedule and within budget [17]. Although different approaches have been developed to address these software quality issues effectively, SPI is the most widely used method [18].

It provides organizations with a powerfully effective way to assess their ability to develop software systems and, therefore, identify their own weaknesses and strengths. This puts them in a good position to start a process improvement program that includes clearly defined, achievable goals that can demonstrate their achievement. SPI's underlying theme is that it can help organizations to define and understand their current software development processes, which enables them to determine the various areas that need to be controlled and manipulated to achieve a specific product effect [19].

SPI has been defined by Fox [20] as "a set of process oriented quality management systems that apply a unified set of theories, tools, methods and techniques in conjunction with attitudes, values and model problem solution." Meanwhile, Rico's [21] definition of SPI is "the discipline of characterizing, defining, measuring, and improving software management and engineering processes, leading to successful software engineering management, higher product quality, greater product innovation, faster cycle times, and lower development costs, simultaneously." In addition, Sommerville [22] has stated that "SPI involves understanding existing processes and changing these processes to improve product quality and/or reduce costs and development time." Sommerville added that "process improvement does not simply mean adopting particular methods or tools or using some model of a process which has been used elsewhere. Process improvement should always be seen as an activity that is specific to an organization or a part of a larger organization" [21]. In Rico's [21] and Sommerville's definitions [21], the focus is on heightening the quality of the software while also decreasing the time it takes to develop the product and the cost. These seem to be more complete definitions, and therefore, it is more appropriate to use them in this research project.

3. Approaches to Software Process Improvement

3.1. Capability Maturity Model

Founded in 1984, the Software Engineering Institute (SEI) is funded by the US government and based at Carnegie-Mellon University, Pittsburgh. The purpose of this organization is to set up protocols and establish methodologies in the software development field.

In 1986, it began the development of a process maturity framework. This was in response to the US Department of Defense's request for a method that would enable them to assess the ability of contractors to

make software that was both on budget and on time. The Capability Maturity Model (CMM) was one of SEI's early developments [23] that aimed to improve the software processes of organizations. CMM was largely started by Watts Humphrey, who wrote the first book on CMM [24]. SEI then started to refine their work with the assistance of Mark Paulk, as well as his team of researchers and engineers.

CMM's primary aim is to determine the capability of an organization by measuring the degree to which its processes are managed and defined. CMM evolved from an ad-hoc immature software process to one that is optimized, mature, and disciplined. It is structured into maturity levels that range from level 1 to level 5. Level 1 is Initial, level 2 is Repeatable, level 3 is Defined, level 4 is Managed, and level 5 is Optimizing (see Fig. 1). A maturity level is defined by Paulk *et al*. [23]) as "a well-defined evolutionary plateau towards achieving a mature software process." Software process maturity is defined by them as being "the extent to which a specific process is explicitly defined, managed, measured, controlled and effective" [23]. A different state of maturity is expressed by each maturity level in an organization. Level 1 is the lowest maturity state, and level 5 is the highest. The five levels are made up of a number of key process areas (KPAs) (see Fig. 1), and several (generally three or four) goals are defined for each process area (PA). Many key practices are identified under one of five sections, which are called "common features." These are attributes used to determine the repeatability and effectiveness of the institutionalization or implementation of a KPA. An organization has to satisfy the goals of the KPAs for that level and also the lower levels if it is to reach a certain maturity level. These key practices assist by enabling an organization to understand how it can achieve the maturity goals. They also serve as examples of the various activities that have to be addressed while improvements are being made to a software process.

Organizations can carry out an "appraisal" to judge their process weaknesses, strengths, and maturity level. Self-assessments or evaluations prescribed via SEI are used to determine an organization's capability. Two surveys have been developed by SEI—the Software Process Assessment (SPA) and Software Contractor Evaluation (SCE)—which rate and compare organizations against CMM. SCE is an appraisal or audit that is conducted by assessors who have been trained to identify contract software developers that are qualified. SPA, on the other hand, is an assessment that helps organizations to evaluate the maturity of their software process, as well as identify the key areas that need to improve.

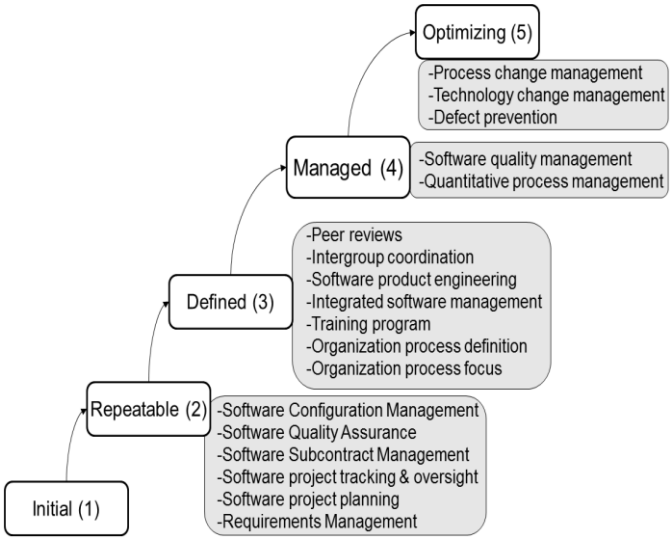

Fig. 1. Key process areas by maturity level.

CMM generally covers planning, engineering, maintaining, and managing software processes. It is accepted as being the de-facto standard in big North American companies, and its popularity is now rapidly

growing throughout Europe. Indeed, CMM has become a valuable SPI model that has detailed definitions of important process areas and key practices, as well as defined levels. The model fails to show how those key practices can be to implemented, however, and does not suggest any efficient implementation strategies. Neither does it address any issues that relate to human resources, such as hiring, selecting, and motivating staff.

CMM is questioned by many companies because of the lack of resources available to small companies for SPI, and it is thought to be more suitable for large organizations [25]. Some researchers have decided to tailor it to small projects [26] while others have applied CMM to a micro team with limited resources [27]. Hareton and Terence [26] have argued that CMM needs to be tailored to small projects, as this will encourage more companies to use it. They created a process framework for small projects that is based on CMM's eight KPAs and obtained positive results regarding software quality. Batista and Dias [27] wrote about a case study in which CMM was applied to a micro team of fewer than 10 people that had limited resources. They focused mainly on the achievement of the KPA objectives of level 2. The results showed a clear improvement had occurred in the processes. This demonstrates that it is possible for SPI to be achieved by a small team that has limited resources.

3.2. Capability Maturity Model Integration

SEI's latest model is CMMI for development version 1.3 [28]. The CMMI project was launched to solve the difficulty of using multiple CMMs. It combines three source models into one single improvement framework that accommodates multiple disciplines and can support two representations (continuous and staged) because of its flexibility. These three source models are the Capability Maturity Model for Software, Electronic Industries, the Alliance Interim Standard (EIA/IS) 731, and the Integrated Product Development Capability Maturity Model (IPD-CMM).

SEI released CMMI version 1.3 in 2010. It was built using knowledge of the best systems and software development practices of high-maturity CMM organizations that had been practicing the model for a long time, along with information from a number of well-regarded and popular models. Figure 2 shows CMMI's history.

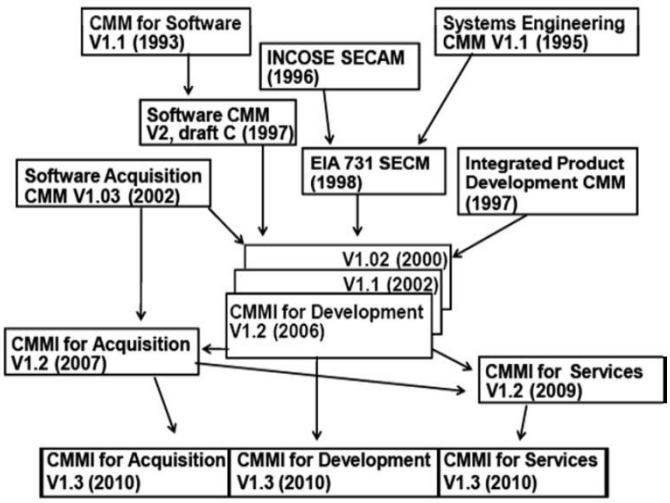

Fig. 2. History of CMMI [28].

CMMI's core objective is to discover an organization's capability by identifying the degree to which its processes are both managed and defined. CMMI has evolved from an immature software process to become a process that is disciplined, optimized, and mature. It is structured into five maturity levels, which are Initial, Defined, Managed, Quantitatively Managed, and Optimizing (see Fig. 3).

CMMI for development includes different PAs, which are categorized across five maturity levels that describe an organization's various states of maturity. CMMI level 1 is the lowest state while level 5 is the highest.

Each of the five levels consists of several PAs, and each has different goals (both generic and specific). Various practices are identified under each goal. The goals of the PAs for that level and all of the lower levels have to be satisfied if an organization is to qualify for a particular maturity level. These practices help an organization to determine how to achieve the maturity goals, and they serve as examples of the kind of activities that have to be addressed when an organization is carrying out an SPI program.

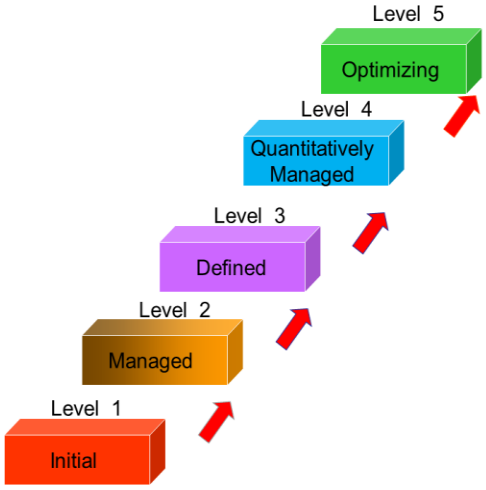

Fig. 3. CMMI maturity levels.

CMMI has continuous and staged representations. Fig. 4 shows a CMMI model that has a staged representation. In this staged representation, the maturity levels provide the company with a recommended order for approaching the improvement process in stages. It is the maturity levels that organize the PAs, as shown in Fig. 4. There are both specific and generic goals within the PAs, as well as specific and generic practices.

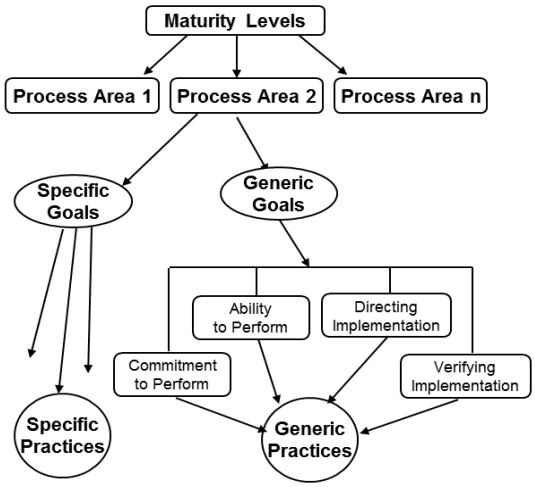

Fig. 4. CMMI model with a staged representation.

Fig. 5 shows a CMMI model that has a continuous representation. The specific goals organize the specific practices, and it is the generic goals that organize the generic practices. Each generic and specific practice corresponds to a different capability level.

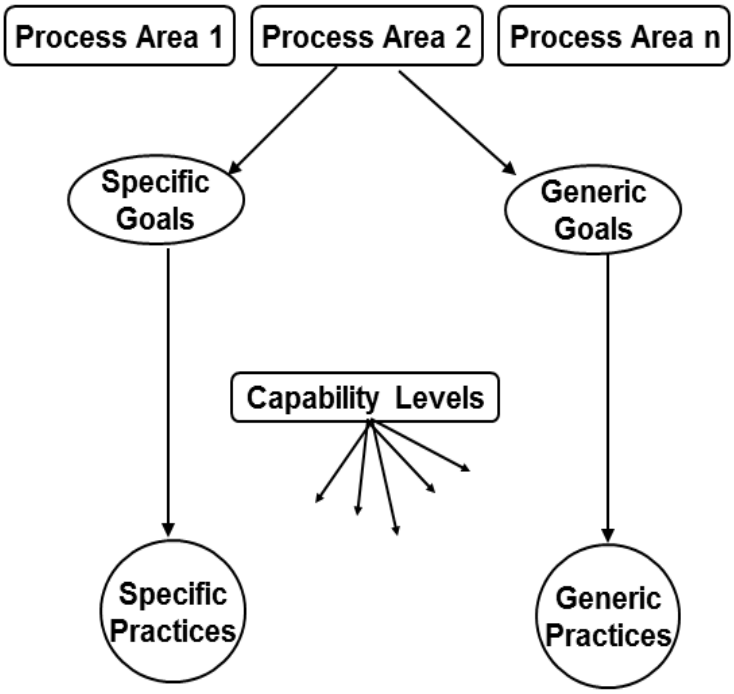

Fig. 5. CMMI model with a continuous representation.

CMMI is a valuable improvement framework that has defined levels, Pas, and key practices. However, similar to its predecessor, it does not show companies how they can implement key practices, nor does it recommend any implementation strategies that would be effective.

3.3. SPICE (ISO/IEC 15504)

SPICE is a set of international standards that has been designed for software process assessment [29]. It aims to harmonize a number of approaches to the assessment of the software process and also encourages self-assessment. The intention is to deliver some reliable evaluations of the capability of the process. These evaluations are not only repeatable but also provide the kind of results that enable organizations to make valid comparisons between a number of different assessments. SPICE provides a mechanism for the interchange of assessment results that are based on various process models. This is achieved by defining a reference model for the software processes, as well as process capability. SPICE consists of nine parts:

- Part 1: The concepts and an introductory guide
- Part 2: A reference model for the processes, as well as process capability. This part defines the set of processes and provides a framework for evaluating the processes' capability. This is achieved by assessing the process attributes that are structured into capability levels.
- Part 3: Performing an assessment
- Part 4: A guide to performing these assessments
- Part 5: The assessment model and indicator guidance
- Part 6: A guide to the competency of the assessors
- Part 7: A guide for use in the improvement process
- Part 8: A guide for use in determining the capability of the supplier process
- Part 9: Vocabulary

The key section of SPICE is Part 2, where a set of universal software engineering processes are documented in the reference model to provide an organization with a common basis for the various software process assessment methods and models. This makes it possible to report the assessment results in a common context.

The process capability is expressed in terms of the process attributes that are grouped into capability levels, with each level representing an incremental evolution in the control and management of the processes. Table 1 shows both the capability levels and the associated process attributes.

Table 1. SPICE Capability Levels and Process Attributes

Capability Level	Capability Description	Process Attributes
0	Incomplete process	–
1	Performed process	Process performance
2	Managed process	Performance management Work product management
3	Established process	Process definition and tailoring Process resource
4	Predictable process	Process measurement Process control
5	Optimizing process	Process change Continuous improvement

The process is evaluated by making an assessment of the selected processes' capability attributes. The assessment output contains process profiles and a capability level rating (which is optional) for every process instance that is assessed. The process assessment is either carried out by a team that has at least one qualified assessor or is performed on a continuous basis by using tools verified by the qualified assessor and are suitable for data collection. SPICE successfully harmonizes a number of existing approaches to the improvement of the process, but it does not provide specific recommendations for improvement. Instead, it leaves it to the practicing company to determine what the specific improvement path will be.

3.4. ISO 9000

ISO 9000 is a series of quality system standards [30] that aim to create common standards for both quality assurance and quality management. As they are generic, these standards can be applied to organizations in the manufacturing and service industries, whatever their size is or how complex their product/service is.

ISO 9001, "Quality Systems—The Standard Model for Quality Assurance in Design and Development, Installation, Production, and Servicing," is applicable to both software maintenance and development [31]. This model is utilized when the supplier needs to give assurances that it will conform to specific requirements during the design, development, installation, production, and servicing stages. These are not product-specific requirements, and the main aim is to provide customer satisfaction by avoiding instances of nonconformity at any of the stages, from design right through to servicing. This is a prescriptive model that encourages companies to examine the internal quality management systems within their own organization. ISO 9000-3, "Quality Management and Quality Assurance Standards—Guidelines for the Application of ISO 9001 to the Development, Supply, and Maintenance of Software," was published in 1991 by the International Standards Organization [32]. ISO 9000-3 works by interpreting 9001 for software, while ISO 9001 2000 replaced the ISO 9001 1994 standard [30]. Furthermore, the old quality standards of ISO 9001 1994 and ISO 9003 1994 are now obsolete as they have all been discontinued. Although ISO 9001 2000 has proved to be an effective standard for quality management, it has failed to address any of the SPI

implementation issues.

3.5. Trillium

Trillium is an assessment model that is based on SEI's CMM and developed by Northern Telecom and Bell Northern Research in 1991 [33]. Its set of practices are derived from a benchmarking exercise, and it possesses eight capability areas that are able to span the five Trillium capability levels. Different practices have been designed under each of the capability levels. Although it was designed for embedded software systems, like telecommunications systems, Trillium can also be applied to other parts of the software industry, such as various management information systems.

3.6. Bootstrap

Bootstrap a methodology for assessing and improving software process quality [34]. It is primarily used in Europe, and its main features include the underlying process model, assessment model, capability levels for evaluation, process improvement guidelines, and rating and scoring principles.

4. Current Challenges of Software Process Improvement

Many organizations in the software industry focus on gaining high maturity levels without making any improvements to their process capability, due to the level of competition that currently exists in the industry. Measuring organizational capability with maturity levels is difficult as the maturity framework fails to look deeply into every organizational practice [35]. There is, in addition, a need to adjust the appraisal methods and the CMMI framework [35].

The standards and models used to improve the software development process have significantly advanced over the past decade. Research reveals that the efforts exerted to develop these standards and models help in the production of high-quality software. They can also increase productivity and reduce both time and costs [36]-[38]. Unfortunately, these advances have not been matched by any similar advances in the adoption of the standards and models [39], [41], [42]. This situation means that many SPI efforts have only had limited success. Research has shown that 67% of all SPI managers would like to receive guidance on "how" they could implement SPI activities effectively, rather than be given advice on which SPI activities they should implement [42]. Even though the SPI implementation process is important, there has been little empirical research conducted into developing ways that organizations can implement SPI programs effectively [35]. This condition suggests that SPI's current problem is the lack of effective strategies that organizations can use to successfully implement the standards or models, rather than any lack of standards or models.

Some SPI models, such as CMMI, are seen as being more suitable for large organizations with enough resources to invest, as smaller organizations have found it hard to tailor some of CMMI's recommendations [41], [43], [44]. Leung stated that this adaptability issue can stop a few companies from embarking on SPI [39]. Hareton and Terence [43] have argued that CMM needs to be tailored to small projects to encourage more companies to use it.

The success of SPI initiatives in different companies can be limited by other difficulties, such as the expense of SPI, the lack of resources, the lack of support, the lack of immediate success, and organizational politics [41], [44]. As a result, adopting SPI initiatives can be difficult for companies. Therefore, researchers constantly need to be aware of what is really undermining the SPI implementation process and organizations from successfully implementing SPI standards and models.

A thorough literature review showed that one SPI topic was missing, that is, empirical studies only concentrate on "what" activities are needed instead of "how" companies can implement them. The current research has shown that just identifying "what" activities are needed to implement SPI is no longer

sufficient. If SPI programs are to be successfully implemented in the future, organizations need to know "how" they can do it [35], [42]. Therefore, attention to the "how" is vital.

5. Conclusion

SPI has been examined in this paper, and a number of approaches have been described. In addition, a thorough literature review has revealed what is missing in SPI. In other words, although many SPI models and standards exist, very little attention has actually been paid to their effective and successful implementation. Several advances have been achieved in the development of SPI models and standards, such as CMM, CMMI, and ISO SPICE. However, these standards and models have not been adopted sufficiently, which means that many SPI efforts have achieved only limited success. SPI's current problem, therefore, is not a lack of standards or models, but instead the lack of an effective strategy to successfully implement the models or standards. The literature also reveals that many companies have adopted ad-hoc methods to implement SPI initiatives in real life, instead of any rigorous systematic methods. No approach has been identified so far that would specifically help organizations to design effective implementation initiatives for SPI.

The literature also highlights the fact that most existing research focuses on "what" activities can be used by organizations to implement SPI, instead of "how" the companies can implement these activities. SPI implementation is vital, and this importance demands that it should be recognized as a complicated process. Attention to "how" it can be implemented is crucial in the successful implementation of SPI, and organizations should use an organized set of activities to determine their own SPI implementation maturity.

Further research into SPI implementation is therefore needed to ensure a higher quality of SPI implementation, reduce the time and cost of this implementation, and, just as importantly, increase the satisfaction levels of SPI practitioners.

Acknowledgment

The author would like to acknowledge the support provided by AlMaarefa University while conducting this research work

References

[1] SQA. (1999). *Handbook of Software Quality Assurance*. Prentice Hall.

[2] Halvorsen, C. P., & Reidar, C. (2001). A taxonomy to compare SPI frameworks. *Software Process Technology*. 217-235, Springer Berlin Heidelberg.

[3] Kitchenham, B., & S. Pfleeger, (1996). Software quality: The elusive target. *IEEE Software*, *13(1)*, 12-21.

[4] Scacchi, W. (2001). Process models in software engineering. *Encyclopedia of Software Eng*. 993-1005.

[5] Ashrafi, N. (2003). The impact of software process improvement on quality: In theory and practice. *Information & Management*, *40(7)*, 677-690.

[6] García-Mireles, Gabriel, A., *et al*. (2013). The influence of process quality on product usability: A systematic review. *CLEI Electronic Journal*.

[7] Iqbal, J., Ahmad, R. B., Nasir, M. H. N. M., Niazi, M., Shamshirband, S., & Noor, M. A. (2015). Software SMEs' unofficial readiness for CMMI®-based software process improvement. *Software Quality Journal*, 1-27.

[8] Keshta, I., Niazi, M., & Alshayeb, M. (2017). Towards implementation of requirements management specific practices (SP1. 3 and SP1. 4) for Saudi Arabian small and medium sized software development organizations. *IEEE Access*.

[9] Keshta, I., Niazi, M., & Alshayeb, M. (2018). Towards implementation of process and product quality

assurance process area for Saudi Arabian small and medium sized software development organizations. *IEEE Access*.

[10] Rahmani, H., Ashkan, S., & Abdullah, K. (2016). CIP-UQIM: A unified model for quality improvement in software SME's based on CMMI level 2 and 3. *Information and Software Technology*, *71*, 27-57.

[11] Niazi, M., & Babar, M. (2009). Identifying high perceived value practices of CMMI level 2: An empirical study. *Information and Software Technology Journal*, *51(8)*, 1231-1243.

[12] Dutra, E., & Gleison, S. (2015). Software process improvement implementation risks: A qualitative study based on software development maturity models implementations in Brazil. *Product-Focused Software Process Improvement*.

[13] Zahran, S. (1998). Software process improvement — Practical guidelines for business success. Addison-Wesley.

[14] Paulk, M., Weber, C., Curtis, B., & Chrissis, M. (1994). A high maturity example: Space shuttle onboard software, in the capability maturity model: Guidelines for improving software process. Addison-Wesley.

[15] Pitterman, B. (2000). Telcordia technologies: The journey to high maturity. *IEEE Software*.

[16] Yamamura, G. (1999). Software process satisfied employees.

[17] Jiang, J., Klein, G., Hwang, H. G., Huang, J., & Hung, S. (2004). An exploration of the relationship between software development process maturity and project performance, *Information and Management*, (*41*). 279-288.

[18] SEI. (2004). Process maturity profile. Software Engineering Institute Carnegie Mellon University,

[19] Humphrey, W. S. (1995). A discipline for software engineering. Addison Wesley.

[20] Fox, C., & Frakes, W. (1997). The quality approach: Is it delivering. *Communications of the ACM*, *40(6)*, 25-29.

[21] Rico, D. (1997). SPI: Impacting the bottom line by using powerful solutions. Retrieved from: http://davidfrico.com/spipaperpdf.htm.

[22] Sommerville. *Software Engineering*, 9th ed. Addison-Wesley.

[23] Paulk, M., Curtis, B., Chrissis, M., & Weber, C. (1993). Capability maturity model for software. Version 1.1. CMU/SEI-93-TR-24, Software Engineering Institute USA.

[24] Humphrey, W. (1989). *Managing the Software Process*. Addison-Wesley.

[25] Cater-Steel, A. (2001). Process improvement in four small software companies. *Proceedings of the 13th Australian Software Engineering Conference*.

[26] Hareton, K. N. L., & Terence, C. F. Y. (2001). A process framework for small projects, software process-improvement and practice, *6*, 67-83.

[27] Batista, J., & Dias, D. F. (2000). Software process improvement in a very small team: A case with CMM. *Software Process-Improvement and Practice*, *5*, 243-250.

[28] CMMI for Development Version 3. Retrieved from: http://www.sei.cmu.edu/reports/10tr033.pdf.

[29] ISO/IEC-15504. (1998). Information technology — Software process assessment. Technical report - Type 2.

[30] ISO-9001. (2008). Quality management systems requirements. Retrieved from: http://www.iso.org/iso/catalogue_detail?csnumber=46486: 20/NOV/

[31] Ince, D. (1994). *ISO 9001 and Software Quality Assurance*. McGraw-Hill.

[32] ANSI/ASQC-Q9000-3. (1991). American National Standard, American Society of Quality, Milwaukee.

[33] April, A., & François, C. (1995). Trillium: A model for the assessment of telecom software system development and maintenance capability. *Proceedings of the Software Engineering Standards Symposium*.

[34] Pasi, K. (2001). BOOTSTRAP 3.0 - A SPICE conformant software process assessment methdology, *IEEE*

Computer Society, 95-107.
[35] Chrissis, M., Konrad, M., & Shrum, S. (2009). CMMI guidelines for process integration and product improvement - CMMI for development version 1.2. Addison Wesley.
[36] AMRI. (2013). CMMI benefits. Retrieved from: http://www.lamri.com/resources/CMMI%20Benefits.pdf
[37] Lepmets, M. (2010). Which process model practices support project success? *Proceedings of the 17th European Conference on Communications in Computer and Information Science*.
[38] Shih, S.-P., Shaw, R.-S., Fu, T.-Y. F., & Cheng, C.-P. (2013). A systematic study of change management during CMMI implementation: A modified activity theory perspective, project management journal.
[39] Leung, H. (1999). Slow change of information system development practice. *Software Quality Journal*, *8 (3)*,197-210.
[40] Niazi, M. (2006). Software process improvement: A road to success. *Proceedings of the 7th International Conference on Product Focused Software Process Improvement*.
[41] Staples, M., Niazi, M., Jeffery, R., Abrahams, A., Byatt, P., & Murphy, R. (2007). An exploratory study of why organizations do not adopt CMMI. *Journal of Systems and Software*, *80(6)*, 883-895.
[42] Herbsleb, J. D., & Goldenson, D. R. (1996). A systematic survey of CMM experience and results. *Proceedings of the 18th International Conference on Software Engineering (ICSE-18)*.
[43] Hareton, K. N. L., & Terence, C. F. Y. (2001). A process framework for small projects, software process-improvement and practice. *6*, 67-83.
[44] Niazi, M., Ali-Barbar, M., & Verner, J. M. (2010). Software process improvement barriers: A cross-cultural comparison. *Information and Software Technology*, *52(11)*, 1204-1216.
[45] Niazi, M. (2015). A comparative study of software process improvement implementation success factors. *Journal of Software: Evolution and Process*.
[46] Akbar, M. A., Sang, J., Khan, A. A., & Shafiq, M. (2019). Towards the guidelines for requirements change management in global software development: Client-vendor perspective.

Ismail Keshta received his B.Sc. and the M.Sc. degrees in computer engineering and his Ph.D. in computer science and engineering from the King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia, in 2009, 2011, and 2016, respectively. He was a lecturer in the Computer Engineering Department of KFUPM from 2012 to 2016. Prior to that, in 2011, he was a lecturer in Princess Nourah bint Abdulrahman University and Imam Muhammad ibn Saud Islamic University, Riyadh, Saudi Arabia. He is currently an assistant professor in the computer science and information systems department of AlMaarefa University, Riyadh, Saudi Arabia. His research interests include software process improvement, modeling, and intelligent systems.