Software Process Improvement (SPI) In Small Software Firms

Mejhem Yousef Al Tarawneh
Dept. of Information Technology, College of Arts & Sciences, Universiti Utara Malaysia
mejhem1981@yahoo.com

Mohd Syazwan Abdullah
Dept. of Information Technology, College of Arts & Sciences, Universiti Utara Malaysia
syazwan@uum.edu.my

Abdul Bashah Mat Ali
Dept. of Information Technology, College of Arts & Sciences, Universiti Utara Malaysia
bashah@uum.edu.my

Key words: software process improvement, small software firms, future research, critical success factors

ABSTRACT

Small software firms need to improve their software process by adopting a suitable SPI model. However, all the traditional SPI models and standards were created for large firms and software houses. This paper presents the generic information about software processes, software process improvement and small software firms to highlight the challenges faced by the small software firms in applying traditional SPI models. This paper also discusses how future work can be done to solve this problem. This is achieved by reviewing literatures to determine the SPI activities, characteristics of small software firms and the critical success factors of software process and SPI, and using these features in developing new software process model for small software firms.

1. Introduction

The use of technology techniques have increased in our life. As a result we have to manage these techniques to get all the benefits of the technologies. Small software firms form the important sectors that are needed to manage and develop their software processes, because small firms play a fundamental role in most countries economies, and they represent up to 85 percent of all software firms in the US, Canada, China, India, Finland, Ireland, and many other countries [1]. But the problem here is that all the traditional software process improvement (SPI) models and standard for assessment were developed to improve software process in large and very firms that have more than 100 employees, and the small software firms cannot afford these models [2]. Furthermore, there is lack of research about the use of SPI in small firms [3]. This paper discusses these issues and proposes new software process model that can be used in small software firms. This model is based on traditional software process development models and SPI traditional model such as Capability Maturity Model (CMM), Capability Maturity Model Integration (CMMI), International Organization for Standardization (ISO), and Software Process Improvement and Capability Determination (SPICE).

2. Software process

Saiedian and Carr [4] point out that 'software process' refers to a set of tools, practices and methods to produce software products according to specific plan. The main objective of software process is to provide the suitable organizational stability and good control [5]. Although there are lots of software process definitions, all these definitions have the same aim of helping software engineers to develop software of high quality. Pressman [7] defines the software process as a framework of tasks to built high quality software. Sommerville [6] summarized the software process as a structure of activities to development software systems and pointed out that software process consists of the following four activities:

A- Software Specification
This activity used to establish the required services from the system, and determine the constraints of system operations and development. Software Specification has two levels: (1) level for high-end users and (2) customer needs level for system developers.

B- Software Design and Implementation
This activity is used to convert and translate the system specifications to the executable system.

C- Software Validation
This activity is used to show if the system is achieving its specifications and meeting customer needs through testing process.

D- Software Evolution
This activity is used to maintain and develop the system so that the system can meet circumstantial changes such as requirements changes and customer needs.

2.1 Software Development Process Models
Sommerville [6] defines the software development process model as an abstract representation of a process that presents the description of a process from some particular perspective. He summarized the software development process models as a following:

A- Waterfall Model (Software Life Cycle)
This model is the first model in software process development by Rock in 1970 and is called the 'Waterfall', because its numbers of separate specifications and development activities worked as a cascade.
Waterfall activities:
1- Requirements analysis and definition:
Establish the systems services, goals and constraints by consultation with system users.
2- System and software design:
System software using to partition the requirements to software and hardware systems. Software design helping to define and describe the fundamentals and relationships of the software system abstraction.
3- Implementation and unit testing:
Convert the design to a set of programs by implementation, and checking these program units by testing unit to achieve all units' specifications.
4- Integration and system testing:
In this phase, all program units are integrated to create the main system, and test this system to ensure the achieving the software requirements, and deliver the system to the customers.
5- Operation and maintenance:
Install the system and using real data to check the system. The maintenance help to correct the errors in the systems that couldn't be discover in previous phases.

B- Evolutionary Development Models
These models are based on developing an initial implementation and show users how to navigate the users' comments and refining that by many versions to achieve the suitable version for users. Moreover, these models have separated activities such as specification, development and validation, and these activities executed through rabid feedback. These models are divided into two types as a follow:
1- Exploratory development:
This type starts with understanding requirements by an initial outline specification and by working with customers to explore the requirements. Then, the need features are added to achieve the desirable system. An incremental model such as Extreme Program is an example of this method.

2- Throw-away prototyping:
This type starts with poorly understood requirements and explains the needed requirements. Spiral model is an example of this method.

C- Formal system development

This model is based on formal mathematical notations for system specifications and uses a series of transformations for design, implementation and testing.

D-Reuse-Oriented development (Component-based software engineering)

This model uses existing components or COTS (Commercial-off-the-shelf) systems to develop the process by systematic reuse. It has four stages as a following:

1- Component analysis: By knowing the requirements specification, this phase search and specify the suitable components to implement these requirements.

2- Requirements modification: Modify the requirements to be suitable with existing components.

3- System design with reuse: Design the new framework of the system or reuse the existing framework.

4- Development and integration: Develop the software which cannot be bought and integrate the components and COST to create the system.

3. Software Process Improvement

Because of the increasing of use software in all aspects of our life, the software firms need to manage and develop their software processes to meet the challenges of continuously changing user requirements to satisfy the customers needed within set time restraints, low cost, while maintaining high quality. According to BAe [8], most software firms facing tough competitions struggle to develop the quality of its software within specific time and suitable budget to achieve business needs to satisfy its customers. Allen and others [9] also believed that the increasing use of software systems that lead to the complexity of these systems and the need to understand and manage the software development process to ensure high quality, suitable cost and maximize productivity.

According to Wang and King [10] the SPI is a systemic procedure for improving the performance of an existing process system by changing or updating the process. Sommerville [11] point out that the software process improvement is used to understand the current processes and doing changes on process to improve the product quality, reduce cost or accelerate schedules.

3.1 Software Process Improvement Traditional Models and Standards

Most researches in the world have focused on some of the generic process improvement traditional and standards models such as CMM, CMMI, ISO, SPICE and BOOTSTRAP. This section discusses general information about these models as a follow:

A- Capability Maturity Model

CMM model was developed by the U.S. Department of Defense at Software Engineering Institute (SEI) of Carnegie Mellon University. This model focus on managing the process and the main objective of this model is to develop a process maturity framework to help the organization to improve their software process by using five maturity levels (Initial level, Repeatable level, Defined level, Managed level and Optimizing level [12].

B- Capability Maturity Model Integration

CMMI improvement model was created by Software Engineering Institute by combining the CMM models (SW- CMM V2.0, integrated product development (IPD), and system engineering CMM (SE-CMM)) [13]. According to Yao and Lee [14] this model was used as a guideline for improving the process in the organizations. They also point out that this model was written specially for the software industry and describes the software process in detail. Furthermore, this model focuses on supplier to improve the internal software process.

C- Software Process Improvement and Capability Determination

The SPICE is the major international initiative to support the development of an International Standard for Software Process Assessment .The first version of the standard was released in 1995 and the goal of the SPICE project was to develop a standard that would be applicable to both process improvement and capability determination in different application domains [15].

D- International Organization for Standardization

In 1987 the ISO published the first edition of ISO 9000 Quality System Standards and revised this model in 1994 and 2000 [16]. The purpose is to guide the software development and maintenance. ISO 9000 is a quality system for software development stages including design, development, production, installation, and servicing. ISO 9000 series is the standard used to provide the guidance of quality management (by ISO 9000 and ISO 9004) and quality assurance by ISO 9001, 9002 and, ISO 9003 [16].

E- BOOTSTRAP

The bootstrap is a methodology developed in the ESPRIT (the European Strategic Programme for Research) in Information Technology project from October 1991 to February 1993. After the ESPRIT ended, the Bootstrap Institute developed this methodology for this [17]. The main goal of this model to support and help (start up) application of software engineering technology in the software industry [18].

3.2 SPI Critical Success Factors

Most researchers have used the concept of Critical Success Factors (CSFs) to identify areas where attention must be focused. Since Rockart [19] introduced the concept, CSFs studies have been shown to be useful in the analysis of the implementation and use of information systems and management practices. Some studies refer to the critical success factors and critical barriers as both enabling and prohibitive. There are a lot of classifications of these critical success factors. However, Hall and others [20] derived all the critical factors that where founded by SPI researchers to four groups as a following:

A- SPI Economic Factors

Hersh [21] warned that it is not easy to measure the value of process improvement in terms of lower risk, staff monthly productivity, improved quality, or customer satisfaction. Many publications in the past have claimed to have determined the return on investment for process improvement. Recently however, high costs and inadequate re-sourcing have been found to be the greatest hindrance to SPI success.

B- SPI People Issues

There is growing awareness of the important role of individuals in SPI programs as the literature reflects. This is stressed by Komiyama, Sunazuka and Koyama [22] who claim that the process determines the success of the outcome of the software project, and that all personnel must be interested in the process. Some researcher mentioned these people issues as a following: (1) Management commitment and SPI leadership, (2) Staff involvement, (3) Mentors, (4) Training and expertise, and (5) Motivation. [23]

C- SPI Organizational Factors

Many researchers have derived these factors into six, such as [24]. They point out the six organizational factors in SPI (human, political, cultural, goals, and change management). However, Aileen [13] distributed these factors to three dimensions, which focused on communication between the employees and the availability of resources to achieve all needed improvement. She also focused on the business strategy that is used in firms.

D- SPI Implementation Factors

There are a variety of implementation factors which can cause well-planned SPI initiatives to result in failure such as setting realistic objectives, SPI infrastructure, Evaluation and Readiness [25].

4. Small Software Firms

Small software firms represented a high Proportion of firms in most countries all over the world. They represent more than 85% of all software firms in the US, Canada, China, India, Finland, Ireland, and many other countries [1]. Small firms are less hierarchical and have the organizational flexibility and freedom to take more risks than larger ones who operate on a more aggressive business plan. [26]. As for the size of firms is depends on the number of employees, and this number different between countries. According to Fayad and others [27], the small software firms have fewer than 50 employees. Laporte and others [28] determined this number to be fewer than 60 employees. Depending on an empirical study in Australia by Hofer [29], the size of small software firms is between 10 to 50 employees. Then from the pervious analysis we can conclude that the expected size of small software firms is between 10 to 50 employees. Moreover, Hoofers [29] explains the methods and techniques that are used in small software firms, as shown in Table 1. We can conclude that the object oriented programs (OOP), object oriented design (OOD), object oriented analyses (OOA) such as C++, and Component based software development (CBD)such as JAVA are the most common methods used by small firms.

Also according to Hoofers [29], when we look to Table 2, we can make conclusions about some of the generic characteristics of small software firms. It can be concluded that the strongest characteristics that are recognized in more than 86% of firms, customer support, dynamic and flexible company policies, and that internal project meetings are held regularly. Further, it has been established that quality management is important.

5. Software Process Improvement in Small Software Firms

Small firms represent the majority of all firms in most countries and have many processes which need to be developed. They need SPI

methods and techniques	**Ratio**
OOP	92%
OOA/OOD	92%
CBD	63%
UML	48%
COM/DCOM	37%
Automated Testing	25%
Design Pattern	29%
CORBA	22%
Extreme Programming	12%
Analysis Pattern	12%
Re-factoring	12%
Pair Programming	5%

Table 1: Usage of methods and techniques in small software firms.

Characteristics	**Approximately Ratio in small firms (%)**
internal project meetings are held regularly	90%
serve mainly regular customers	65%
projects often last longer than planned	50%
employees often work overtime	73%
marketing is an important part of the company philosophy	75%
investing in training of employees	78%
quality management is important	87%
continuous documentation of all tasks	6%
traditionally structured company	52%
teamwork is important	99%
customer involvement all the time	80%
develop software for many different domains	50%
always newest technology	80%
dynamic and flexible company	94%
customer support is important	95%
often use new methods and techniques	75%

Table 2: characteristics of small software firms.

to achieve all goals and quality assurance for its products, customers satisfaction, reduce cost, and time. However, the main problem here is that no SPI traditional model can be used to improve their software processes, since all these models are designed for large and very large firms [8]. There are many researches who consider the use of SPI in small software firms to be very difficult, and they focused primarily on the larger firms.

5.1 Lack of Research in Small Software Firms

Most researchers focused are on large and very large firms because most of these firms have enough investment to improve its software processes by using SPI traditional models. Thus, the small software firms do not have enough researches to solve their problem of improving their software process. Lobo & Jones [31] emphasize that the empirical research into the rate and success of implementation of SPI in small software firms are always considered as being inadequate. Oscar Pedreira [2] points out in his survey about the empirical studies in the digital libraries that there is 20% of empirical studies about small firms and 80% about large firms. Small software firms need a lot of specific and focused research to improve their software processes.

5.2 Difficulty to Implement SPI Models and Standards in Small Software Firms

SPI traditional models need a lot of activities and requirements, but most small software firms can't afford these [1]. According to Guerrero and Eterovic, small software firms have a lack understanding of the success factors of SPI and do not have enough people to perform all the SPI activities. Therefore, they find themselves to be very far from implementing formal SPI traditional models. Hofer [29] point out that the main problems in small software firms for implementing SPI formal models are factors such as a lack of management and resources, a lack methods and techniques, and a low number of human aspects. One of the main problems in implementing SPI traditional models in small software firms is that the number of employees cannot support the activities of improvement [30]. Therefore, we can conclude that the main problems for implementation of SPI traditional models in small software firms include a low level of SPI experience, lack of resources, and the high cost by using SPI traditional models.

5.3 Future Research

The proposed future research is aimed at helping small software firms in general to improve their software processes.

To help small software firms, we have to determine the characteristics of these firms depending on literature reviews because most small software firms have the same characteristics. Then, we need to determine the SP activities to check the SPI critical success factors. Depending on the critical success factors of SPI, we will discuss all software process models that are used in small software firms to choose the most suitable one that can help small software firms to improve their software processes. We will study all SPI traditional models and choose or adapt the most suitable ones for small firms.

When we have identified the SP model and SPI model, we will need to compare the SP activities of selected SP models with process areas of selected SPI models to determine the missing activities of SP models with the SPI models. We will then modify the SP activities to achieve all process areas of SPI model depending on the activities of other SP models. After this modification, we will determine the new SP model requirements and this requires administrating questionnaires on small firms to check whether the new model meets their expectation. After the analysis of the questionnaires, we will be familiar with the user requirements. Then, we can determine the final requirements for implementing the SPI model for small software firms, and depending on these requirements we will know what activities in selected SP model need to be modified and what activities need to be added to achieve all the key process areas of SPI selected model. Figure 1 shows how this is done.

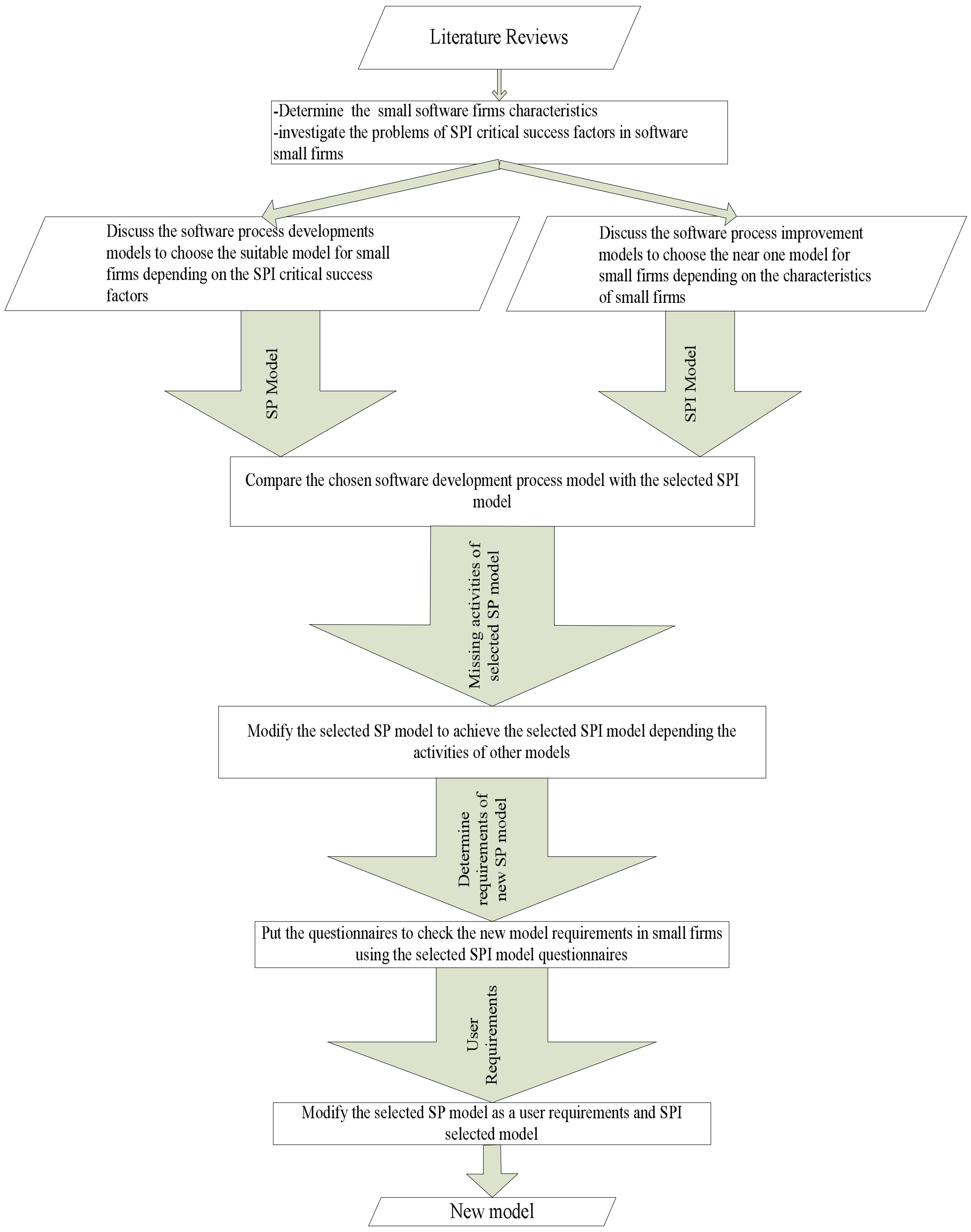
Literature Reviews
-Determine the small software firms characteristics
-investigate the problems of SPI critical success factors in software small firms
Discuss the software process developments models to choose the suitable model for small firms depending on the SPI critical success factors
Discuss the software process improvement models to choose the near one model for small firms depending on the characteristics of small firms
SP Model
SPI Model
Compare the chosen software development process model with the selected SPI model
Missing activities of selected SP model
Modify the selected SP model to achieve the selected SPI model depending the activities of other models
Determine requirements of new SP model
Put the questionnaires to check the new model requirements in small firms using the selected SPI model questionnaires
User Requirements
Modify the selected SP model as a user requirements and SPI selected model
New model

Figure 1: development stages of SP model for small software firms.

6. Conclusion

Small software firms represent a high proportion of software firms around the world. However, these firms do not have the suitable software process model to achieve all key process areas of one of SPI traditional models since these models are created to help large and very large firms.

Small software firms need to have suitable software process models that can achieve all the activities of a selected SPI traditional model. This paper discussed this problem and how it can be solved. It depends on the comparison between software process models and the characteristics of small software firms, as well as and getting the features required by small firms on SPI model. Then a new SP model will be developed based on these requirements.

References:

[1] Richardson, and C. von Wangenheim, "Why Are Small Software Organizations Different?," IEEE SOFTWARE, vol. 1, 2007, pp. 9.

[2] K. Malaivongs, "Software Process Improvement in Thailand," 2008. http://www.drkanchit.com/presentations/200805_ProcessImprovement_NewKeyNoteAddress.pdf

[3] O. Pedreira, et al., "A systematic review of software process tailoring," ACM SIGSOFT Software Engineering Notes, vol. 32, no. 3, 2007, pp. 1-6.

[4] H. Saiedian and N. Carr, "Characterizing a software process maturity model for small organizations," ACM SIGICE Bulletin, vol. 23, no. 1, 1997, pp. 2-11.

[5] B. Wong and S. Hasan, "Software Process Improvement in Bangladesh'," Software Engineering Research and Practice, ed. Arabnia, HR and Reza, H., SERP, 2006, pp. 26-29.

[6] I. Sommerville, "software process", in *Software Engineering*, *6th edn*, Addison-Wesley, 2001, chapter 3, pp.43-50.

[7] R. S. Pressman, Software Engineering: A Practitioner's Approach, 6th international edn, McGraw-Hill Education, Singapore, 2005, p.53.

[8] D. Bae, "Panel: Software Process Improvement for Small Organizations," COMPSAC, 2007.

[9] P. Allen, et al., "PRISMS: an approach to software process improvement for small to medium enterprises," Proceedings of the Third International Conference on Quality Software (QSIC'03), pp. 211-214.

[10] Y. Wang and G. King, *Software Engineering Processes: Principles and Applications*, CRC Press LLC, Boca Raton, FL, USA, 2000, pp.42.

[11] I. Sommerville, "software process", in *Software Engineering*, *6th edn*, Addison-Wesley, 2001, chapter 25, pp.558.

[12] C. Lutteroth, et al., "A maturity model for computing education Ninth Australasian Computing Education Conference (ACE2007), Ballarat, Victoria, Australia, pp. 107-114.

[13] A. Cater-Steel, "An evaluation of software development practice and assessment-based process improvement in small software development firms," Ph.D. dissertation, Sch..Com. Info. Tech, Griffith Uni, Australia, 2004.

[14] Y. Yao and H. Lee, "Applying ISO 9001 and CMMI in quality-oriented knowledge management for software process improvement," International Journal of Electronic Business Management, vol. 2, no. 2, 2004, pp. 140-151.

[15] K. El Emam, et al., *SPICE: The theory and practice of software process improvement and capability determination*, IEEE Computer Society Press Los Alamitos, CA, USA, 1997.

[16] S. Winistorfer and H. Steudel, “ISO 9000: Issues for the structural composite lumber industry,” Forest Products Journal, vol. 47, 1997, pp.43-47.

[17] D.Robben,” TPI, BOOTSTRAP and testing”,’ Sogeti Nederland B.V., Vianen, the Netherlands’, 2000.

[18] D. Vasiljevic and S. Skoog, “A software process improvement framework for small organizations: A research approach,” Master’s thesis, Dept.Soft. Eng.and.Com. Sci. Blekinge Institute of Technology, Ronneby, Sweden, 2003.

[19] J. Rockart, “Chief executives define their own data needs,” Harvard Business Review, vol. 57, no. 2, 1979, pp. 81-93.

[20] T. Hall, et al., “Implementing software process improvement: an empirical study,” Software Process: Improvement and Practice, vol. 7, no. 1, 2002, pp. 3-15.

[21] A. Hersh, “Where's the return on process improvement?,” IEEE SOFTWARE, vol. 10, no. 4, 1993. p. 12.

[22] T. Komiyama, et al., “Software process assessment and improvement in NEC-current status and future direction,” Software Process: Improvement and Practice, vol. 5, no. 1, 2000.

[23] N. Baddoo and T. Hall, “Motivators of software process improvement: an analysis of practitioners' views,” The Journal of Systems & Software, vol. 62, no. 2, 2002, pp. 85-96.

[24] J.Brietzke, A.Rabelo,”resistance factors in software process improvement”,’CLEI ELECTRONIC journal’, volume 9, number 1, 2006, paper 4.

[25] T. Kaltio and A. Kinnula, “Deploying the defined SW process,” Software Process: Improvement and Practice, vol. 5, no. 1, 2000, pp. 65-83.

[26] Winger, Alan R.,” Is Big Really Bad? Business Economics”, 29, 1994, pp. 38-42.

[27] M. Fayad, et al., “Thinking objectively: software engineering in the small,” Communications of the ACM, vol. 43, no. 3, 2000, pp. 115-118.

[28] C. Laporte, et al., “Initiating Software Process Improvement in Small Enterprises: Experiments with Micro-Evaluation Framework,” 2005, pp. 153–163.

[29] C. Hofer, “Software development in Austria: results of an empirical study among small and very small enterprises,” 2002, pp. 361-366.

[30] P. Grunbacher,”A software assessment process for small software enterprises”,’ In Proceedings of EUROMICRO97’, Budapest, Hungary, September. IEEE Computer Society Press, 1997, pp.123-128.

[31] M. Xydias-Lobo and J. Jones, Quality Initiatives and Business Growth in Australian Manufacturing SMEs: an Exploratory Investigation, School of Commerce, Flinders University, 2003.