Assessment of factors affecting the software process improvement in small organizations

Bakir Karahodža, Elma Avdagić-Golub, Alem Čolaković

University of Sarajevo, Faculty of Traffic and Communications, Zmaja od Bosne 8, 71000 Sarajevo, Bosnia and Herzegovina

Abstract

Software process improvement implies a set of complex and systematic activities of software engineering. It requires theory and models established in management, technical and social sciences. The improvement is based on the assumption that the organization if it owns mature and capable processes, would be able to deliver quality software on time and in line with predicted costs. The maturity models are initially aimed for implementation in enterprise software organizations, government organizations and within the military industry. Their complexity and the size make them difficult to use in small software organizations and companies. In such organizations the interest for use and the efforts to make an efficient and effective organization is always presented, though. In this paper, the basic and derived capability maturity models are described and cases from their implementation are analyzed, along with assessment of results of such projects in business practices. The problem of the software process improvement in small organizations is described, extracting the risks and recommendations for its enhancement. These recommendations are provided in order to set up a foundation for implementation of these models in a specific managerial and organizational environment characterized by small organizations.

Keywords: *software engineering, software quality, software process improvement, CMM models, small organizations, change management*

1 Introduction

Software process consists of a complex set of activities that should result in the delivery of a quality software product or service. In addition to the quality, the software project must be completed within the established deadlines and within the planned budget. People, procedures, and many different components must be well coordinated to achieve such goals. The management practices identify software processes as manageable and subject to improvement. There is a continuous development path of models for software process improvement. IBM began with development of such models to improve software quality in the early 1980s. Along with the IBM team, Humphrey [1] developed an original concept that later served as the basis for many feature models, standards, and methodologies. Humphrey found that the quality of software is directly related to the quality of processes used in its development. To improve the software development process, Humphrey attempted to implement the continuous improvement cycle Plan-Do-Check-Act (PDCA) by Deming [2]. After the implementation of such a new concept, the quality of the software was not significantly improved, though. It has been found that improved software development practices do not survive until organizational behavior changes in a way that is supported and maintained. The conclusion was that organizations must remove obstacles to continuous improvement in a unique and specific order if they are to succeed.

In the late 1980s, the Software Engineering Institute (SEI) at Carnegie Mellon University, also began work on developing directives to establish and improve a high-performance software organization. The research was funded by the US Department of Defense (DoD), wanting to ensure the success of the project through the maturity assessment of their subcontractors. The project continued on developing a maturity framework originally started by IBM and Humphrey. The renewed and expanded model was named The Capability Maturity Model for Software (SW-CMM) and retained the original idea of a multilevel model. The SEI recognized the chance to promote a novel framework outside the military sector of American industry, which succeeded in accepting new ideas [3].

Despite the wide interest in the application of CMM based framework for improving software processes, it was found that small businesses, small organizations, and

Published: 30 April 2022
Corresponding author: *Bakir Karahodža (bakir.karahodza@fsk.unsa.ba)*

small projects encounter certain difficulties in its implementation [4]. CMM models mostly relate to software practice in large projects and software organizations. In addition, many of these practices are unsuitable for small projects, which are prevalent not only in small businesses and small software organizations but also in large businesses. Small projects within such organizations often act as small organizations, i.e. independent cost centers. Although the authors of CMM have repeatedly emphasized CMM was created for any project and any organization, there is a general conclusion that CMM cannot be applied in an integral form in small and medium organizations [5].

Software process improvement (SPI) is defined as a complex, systematic, and highly professional activity of software engineering that requires theory and models, skilled technical staff and managers, and motivated and ready top management [1]. From the very definition, one can recognize some of the principles, but also the preconditions for SPI. Given that the software process is by definition a complex set of activities and operations, it can be concluded that any attempt to improve these processes is at least as demanding. The challenge thus becomes multiplied: perform SPI in a small software organization and try to identify a large set of constraints and then overcome them.

The rest of the paper is organized as follows: Section “Capability maturity models” summarizes and discusses several capability maturity models, such as CMM and CMMI. In “IDEAL process improvement life cycle” we detail one of the most often used approaches to continuous improvement and describe its steps. In “Software process improvement in small organizations” we present and discuss SPI in small organizations, and describe the common factors constraining or enabling the process improvement, along with proposed prerequisites with success. We conclude the article in “Conclusion”.

2 Capability maturity models

In the last several decades, from the first edition of CMM for Software ver.1.0 to the present, CMM-based frameworks have traversed a complex path from a specialized model for evaluating bidders for the U.S. military industry, up to a framework with a broad set of guidelines for software process improvement. Each CMM model is designed as a so-called maturity model. The maturity model is defined as a structured set of elements that describe the characteristics of effective processes [6]. The idea of the SEI was to provide a place where an organization could start, i.e. from where it can be launched in activities to improve software processes. They took long lasting knowledge and experience in SPI along with already adopted terminology and common vision within the academic and business community.

According to the SEI, CMM is defined as “a reference model of process maturity in a specified discipline, used to improve and assess a group’s ability to perform those disciplines” [7]. Also, CMM models are described as "a set of public criteria that describe the characteristics of an organization that has successfully implemented process improvements" [8].

CMM models differ from each other in:

- Disciplines: software engineering, systems engineering, etc.
- Definitions of maturity, i.e. ways to improve the process, and
- Structures: phase or continuous.

Inspired by the success of CMM-based software improvements, various organizations have sought to apply this concept to other critical engineering disciplines. As they were the most competent with their basic knowledge of maturity models, SEI initiated the coordination of international efforts to develop the *Systems Engineering Capability Maturity Model* (SE-CMM) and the *Integrated Product Development Capability Maturity Model* (IPD-CMM).

In addition to these models, *People CMM* and *Software Acquisition CMM* have been developed, which are similar to CMM for software with their architecture, basic principles, and support practices. Several other projects were later launched in an attempt to create an international standard for process management. The project started in 1991 and was named SPICE (Software Process Improvement and Capability Determination), and since 1993 it has been covered by ISO / IEC 15504 [9] covered by the ISO and the International Electrotechnical Commission (IEC).

The rest of the chapter describes the SEI models in more detail, especially CMM for Software, which was first developed, following with the CMM Integration.

2.1 Software CMM

The Software CMM (SW-CMM) is a first developed maturity model, and very quickly achieved broad success. It later became a foundation for all other maturity models. In 1987, the SEI (Software Engineering Institute) was commissioned by the US Department of Defense to develop a method for determining the capabilities of software contractors who bid for the Air Force. As a result of these activities, a questionnaire (now known as the Maturity Questionnaire - MD) and a bidder's evaluation method (The Software Capability Evaluation - SCE) were first developed.

CMM is based on five levels of process maturity, ranging from level one (1) to level five (5). The level one is the initial level of maturity - the lowest degree and each subsequent level of maturity consist of key process areas (KPA), such as e.g. Requirements Management (RM) or Software Project Planning (SPP). These areas are organized by certain characteristics. Each KPA has specific goals that must be met and all KPAs at some level must be met to demonstrate that the level of maturity in the organization has been achieved. Achieving the KPA at one level allows the organization to initiate improvement activities at the next, higher level.

In this way, SW-CMM reached the desired outcome: customers are able to identify the advantages, disadvantages, and potential risks of working with their software vendors - each level indicates the state of maturity of the entire software organization. The SW-CMM is written in a hierarchical form containing an abstraction of axioms and universal knowledge applicable to software engineering and project management with detailed instructions and examples (Figure 1).

The five levels and key process areas that describe them are as follows:

1. Level – **Initial** describes software processes as ad-hoc, and often chaotic. Only a few processes have been defined, and success practically depends on individual efforts and so-called heroes within the organization. It takes tremendous effort from management and employees to overcome difficulties in software development that is constantly unpredictable. Plans, budget, functionality and product quality change depending on the motivation of individuals, and their skills and knowledge.
2. Level – **Repeatable** establishes the basic management processes for software development projects: monitoring costs, monitoring product plans and functionality. The necessary process discipline is established to replicate previous successes on projects with similar applications, although project-specific processes may differ. The basic elements of management control are built-in and the development itself can be described as disciplined because the monitoring and planning of the process are more stable.
3. Level – **Defined** documents, standardizes and integrates software processes for both management and engineering processes into standard software processes within the organization. All projects use proven and customized versions of the standard processes within the organization. An education program is implemented throughout the organization so that managers and other employees acquire the necessary knowledge and skills in projects. Development processes are defined as standardized and consistent, and software engineering and managerial jobs are stable and repeatable.
4. Level – **Quantitatively Manageable** collects detailed procedures for software processes and product quality. Productivity and quality are measured in each of the major software process activities and in all projects. These measurements are defined and consistent. The causes and consequences of the whole development process are already very well-known and there are no unpredictable situations. Corrective actions are used in case of approaching the set product quality limits.
5. Level – **Optimizing** continuously improves processes by enabling quantitative feedback from processes, innovative ideas, and technologies. The data on the effectiveness of software processes are used to conduct a cost-benefit analysis of new technologies and proposed changes in the software production process. Continuous improvements are being sought to expand capabilities and areas of application.

The Key Process Areas (KPA) in the CMM are met by achieving the objectives described in the key procedures and examples. There are also informative components that provide guidance on model interpretation. There are 52 objectives and 316 key practices for 18 key process areas. Practices and examples describe what good engineering and management practices are, but they are not exclusive in how to implement processes. Under the term “general characteristics”, the CMM identifies institutional activities that stabilize all key process areas. Concepts such as measurement, training, documented procedures, executive policies, top management support, appropriate tools, verification of practices, and continuous process improvement are gradually included within levels. Examples of KPAs are as follows (Level 2):

- Requirements Management - RM
- Software Project Planning - SPP
- Software Project Tracking & Oversight - SPTO
- Software Subcontract Management - SSM
- Software Quality Assurance - SQA
- Software Configuration Management - SCM.

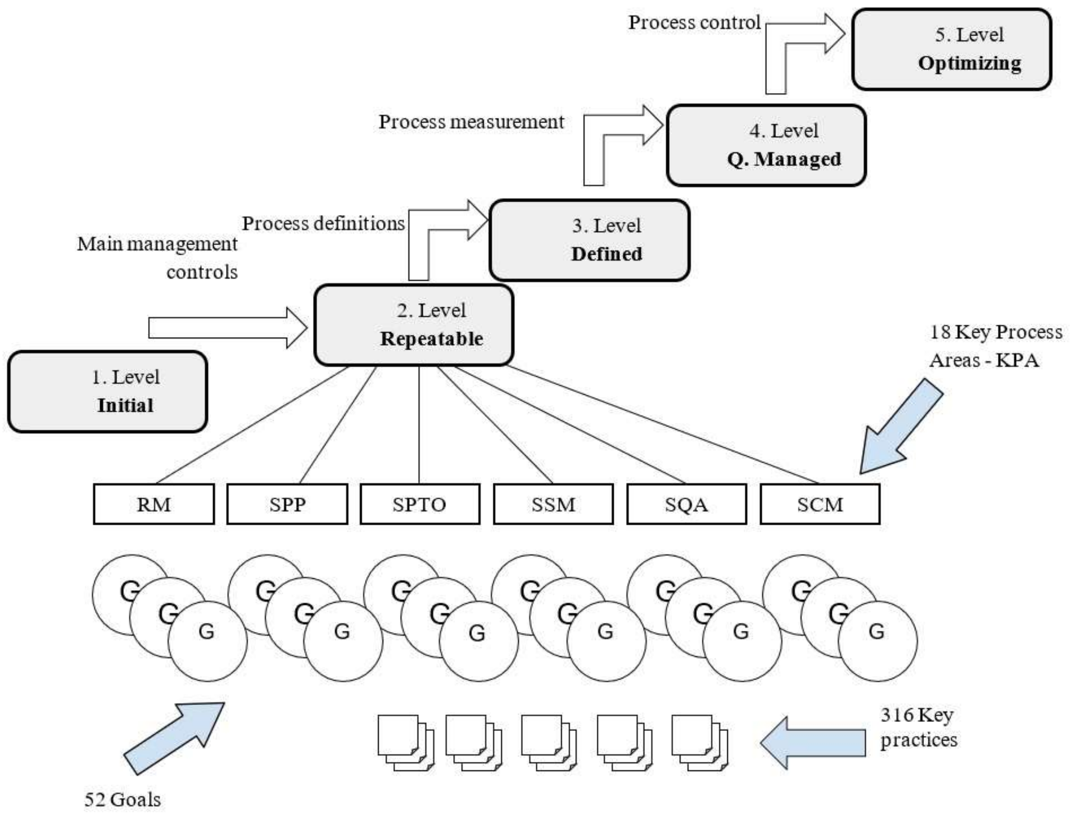

Figure 1. Hierarchical structure of the SW-CMM model

2.2 Software CMM Integration

SW-CMM has achieved acquisition and expansion in the commercial IT sector, certain criticisms, redesigns, releases of legacy models, as well as innovative versions issued by the SEI itself. Models for other disciplines such as systems engineering, integrated product development, and others have been developed. Although many organizations found these models useful, they also faced problems caused by inconsistencies when integrating with other models, such as the ISO family of standards and other process improvement programs. For that reason, in 2000 the initiative to integrate various CMM frameworks into a single set of integrated models was announced. The result of this initiative is the creation of CMM Integration (CMMI). This model has undergone several editions and has further popularized this approach to software process improvement [10]. CMMI terminology is very similar to the original version of CMM. For example, Key Practices are now defined as Specific Practices. Seven new Process Areas have been added, four of which relate to Integrated Product and Process Development (IPPD) and Supplier Sourcing (SS). There are now 25 Process Areas (PAs), unlike the CMM which had 18 Key Process Areas (KPAs).

The major change is in the way it is implemented and applied. CMMI can be applied through continuous or phased presentation. Continuous presentation allows the organization to choose the sequence of improvements that best suit their business goals and mitigate potential risks. The phased presentation provides a previously proven sequence of improvement, starting with basic managerial practices and advancing through successive levels, where each serves as a foundation for transition to the next level.

In a continuous presentation, a given process area is defined by its Capability level and each Process area can exist in any of the six capability levels, independent of other process areas (Figure 2). Here, the term "level of maturity" refers to a predefined group of process areas that exist at the same level of maturity, while the term "level of capability" refers to only one process area.

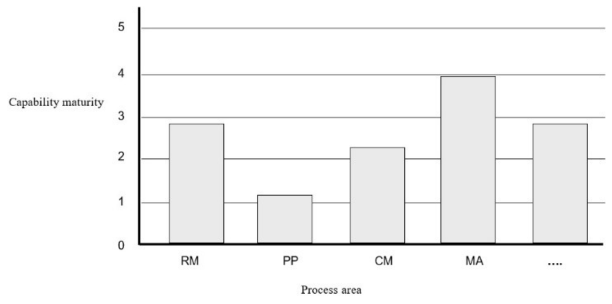

Figure 2. CMMI continuous presentation of capability levels

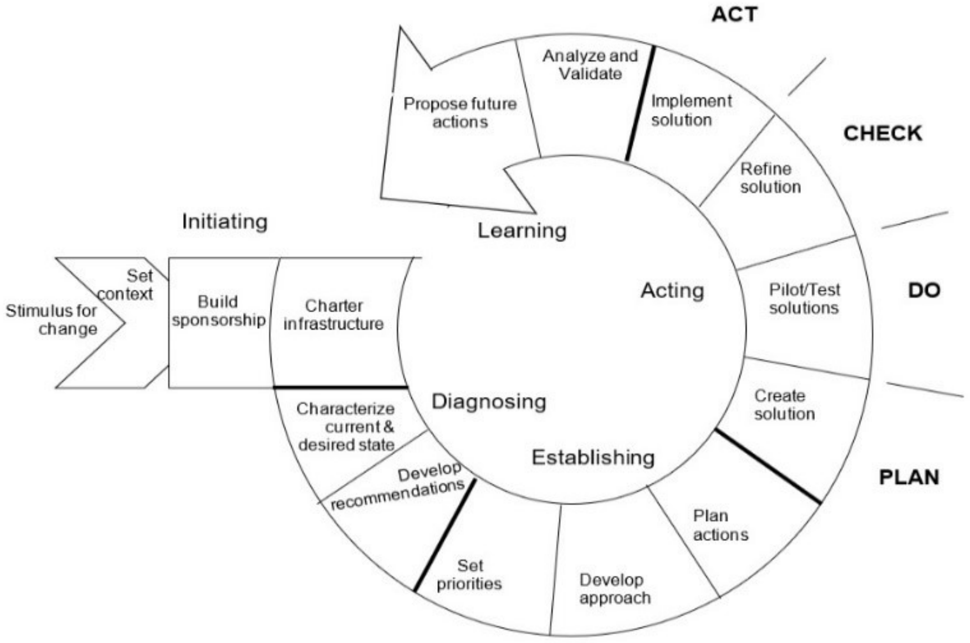

Figure 3. IDEAL process improvement life cycle

3 IDEAL process improvement life cycle

During the implementation of CMM models in various organizations, the need for specific guidelines for the implementation of the model is recognized. The reason for creating a specific approach is obvious. The attempt to improve is so complex and its effect is so far-reaching that it requires a specialized and systematic approach to manage a kind of improvement lifecycle. In order to fulfill such requirements, SEI has developed the IDEAL model. It's aimed *to complete the lifecycle of software process improvement* based on the CMM framework in an organization.

IDEL provides an approach to continuous improvement by highlighting the steps needed to establish an improvement program. Following IDEAL's phases, activities, and principles, the model provides a form of a disciplined engineering approach to improvement. The focus is set on the management of improvement programs, as well as establishment of a long-term improvement strategy. The model consists of five phases:

1. **I**nitiating–Laying the groundwork for the improvement.
2. **D**iagnosing – Determining where the organization is and where it wants to be.
3. **E**stablishing– Planning a specific way to achieve the desired state.
4. **A**cting– Execution according to plan.
5. **L**earning– Learning from experience and improving readiness to adopt new technologies in the future.

Each of these five phases consists of several activities that complement each phase. The similarity of the phases of the IDEAL cycle with the Deming Plan - Do - Check - Act cycle is obvious (Figure 3 [11]):

– **Initiating** Phase. The critical groundwork of the improvement program is being laid. The business reasons for taking the program are being refined and clearly presented. The contributions of the improvement program according to the business goals are identified, as well as the relations with other tasks in the organization. Management support is provided and the necessary resources are allocated. The necessary infrastructure for implementation management is finally being launched. Activities at this stage are critical. If done well and completed, the activities that follow can continue with minimal deviations. If they are poorly or incompletely done, then time, effort, and resources will be wasted in the following stages.

– **Diagnosing** Phase. The diagnostic phase develops a more complete understanding of the improvement work. During this phase a more complete understanding of the work to improve the development, as well as two characteristics of the organization: the current state of the organization and the desired future state. This is the phase for which the SEI has developed methods to measure the current organizational level. It was originally a set of CBA-IPI (CMM Based Appraisal for Internal Process Improvement) methods. Since 2002 the SCAMPI (Standard CMMI Appraisal Method for Process Improvement) set of methods has been used. The defined organizational state is used to develop an approach for business practices improvement.

– **Establishing** Phase. The purpose of this phase is to develop a detailed work plan. Priorities are set as a reflection on the recommendations created during the diagnostic phase, as well as organizational operations and requirements for the operating environment. A priority-based approach is developed, whilst actions, checkpoints, desired outcomes and responsibilities are finally specified in the action plan.

– **Acting** Phase. Activities in this phase help the organization to implement the work that was conceptualized and planned in the previous three phases. These activities usually require more time and more resources than all the other phases combined.

– **Learning** Phase. This phase completes the improvement cycle. One of the goals of the IDEAL model is continuous improvement and readiness to implement changes. In the Learning phase, the overall experience from the IDEAL model is reviewed to determine what has been met, whether the attempt has met the objectives, and how the organization can implement change in a more effective/efficient way in the future.

4 Software process improvement in small organizations

Related studies highlight many of the factors that affect SPI projects in small organizations. In our approach, we gather and analyze the most important factors for their successful implementation. Based on the selected factors, we categorize each within one of three major categories. Later, we extract and compare prerequisites for success aligned with the structural organization of KPA found in CMM/CMMI framework.

The original focus of the CMM was on enterprise organizations and contracting large projects with the government. However, there are a huge number of companies that are not able to successfully start and complete activities to improve the software process, due to their internal structure, and especially the number of employees. These companies also have an interest in obtaining an assessment of the maturity of their processes in such a way as to enable them to participate in projects seeking such companies as a condition of bidding for the contract.

One of the first challenges for small organizations is that their primary business goal is survival. Even after realizing that the status quo is unsatisfactory and that SPI will help the organization, and after finding resources and allocating individual responsibility for change, using CMM remains a difficult business decision. According to Paulk [4], small organizations tend to believe “We are all competent .. We all communicate with one another. We are all heroes”. This is how the author describes a very common situation within small organizations trying to explain their internal software process during the appraisal.

The term “small and medium organizations” is often the subject of discussion. The SEI describes small projects as “3-4 months’ duration with 5 or fewer people”. Other authors [12] define a small organization with less than 50 software designers and developers and small projects with less than 20 employees. Somewhat different, certain European authors favoring the SPICE program define small companies with less than 15 people, and medium between 15 and 50 people [13].

The key point is that the small organizations, same as large, have issues with undocumented requirements, resource allocation, training, and product documentation. Regardless of these challenges, they tend to act in an agile and efficient way [14]. Usually, small teams are more productive than large ones - they get closer faster and have fewer communication problems [15]. Some tend to use minimal processes and rely on human skills, while others insist on rigorous use of procedures, planned processes, and methodological steps, techniques, and tools. The dilemma remains open - how much process discipline is needed and what is its role in small organizations and small teams.

4.1 Common factors constraining or enabling the process improvement

Since the first appearance of failures and open questions in the implementation of SPI projects in small organizations, the large number of papers that have researched this topic emerged. We find that even in the period of intensive spread of CMM models in the industrial environment, some of the open issues were defined, along with the key factors for success. In the early CMM work, Humphrey [1], [16] defined differences in expectations of results from the SPI project in a characteristic environment such as small companies. Later, Abbot [17] identified six keys to SPI, and Jonhson and Brodman [18] identified seven challenges in small organizations. Subsequent papers present similar approaches that further describe and categorize success factors in a variety of ways, such as Wongsai [14], Conradia and Fuggetta [19], Kautz [20], Rifkin [21], Allison [22], Taupe [23], Kuhrmann [24], Duba [25], and Alfaro [26]. It is noticeable that a certain number of papers connect success factors with the principles of the general theory of change management.

Based on these researches, some of the general principles and rules for SPI projects are identified and presented. In Table 1., the mapping between extracted factors and prerequisites for success needed to adopt and exploit each factor as enabling is presented. The factors are categorized in one of three categories: Business, Process and People.

Table 1. SPI success factors

Category	**Factor**	**Description**
Business	Gains from improvement are cumulative	Although they produce significant technical and organizational costs, SPI gains accumulate over time if the organization continues to maintain SPI in a systematic and consistent manner.
Business	Cost benefit analysis	Aim for an internal cost benefit model including depreciation of expenses. Perform one experimental SPI attempt in the short term and then inform management that a usable result may require more time than expected.
Business	Focus on development goals and innovations	SPI initiative should be primarily focused and consistent with business goals and strategies in order to ensure a positive effect on the organization's performance that leads to improved software products.
Process	Empirical models and approaches are hypothesis	Although different approaches and concepts of improvement are presented, each should be treated as a hypothesis due to imperfect evidence.
Process	Improvement is always performed in a different environment	There is no specific and universal model that can be completely copied, nor is there a specific methodology that can always be followed - organizational uniqueness.
Process	Measurable goals	The goal measurement is used in two ways: monitoring the quality of the software product along with customer satisfactions, and external monitoring and certification.
Process	Learning	SPI is about learning, not control. The SPI team should work independently of the quality assurance team. Use information about how people really work, not how they describe it. A reward system for reporting problems or suggesting ideas for improvements should be established.
Process	Implementation of automated software support	Automated support for software processes is usually overemphasized. Only stable processes, e.g. inspection, testing, configuration management are suitable for automated support.
Process	Internal resources and resistant to change	The ratio of used resources and resistance to change is inversely proportional. With reduction in organizational size, resistance becomes more of an issue.
People	Domination of the sociological component	Software development and software processes take place in an environment that has the characteristics of technical and sociological systems. However, SPI is run exclusively by humans and technological features of the system are negligible here.
People	Motivation for change	Designers and developers are motivated for change; if possible, start with bottom-up concrete initiatives, and continuously increase their participation through situational learning.
Process	Process improvement champion	Process improvement champion who has experiences that they can draw on to deliver the improvement should be selected. Their political strength within all stakeholders is important.

Underlying the various frameworks and approaches to SPI are, in fact, similar factors that have been taken into consideration. In addition to these principles, the emphasis is on providing preconditions for successful programs.

4.2 Prerequisites for success

Using any CMM framework alone will not raise an organization's level of process maturity enough. Although each approach provides different empirical models, they all in fact share the same fundamental assumptions for success. Only when these conditions are met, various types of SPI project risks can be significantly reduced.

The partial answer in enabling prerequisites for success can also be found in structural organization of KPA in CMM/CMMI framework. They are organized according to certain characteristics: *Commitment to action*, *Ability to change*, *Executive activities*, *Measurement and analysis*, *and Verification of implementation*. Here, we emphasize the *Ability to change*, which should provide an answer to the following question: What prerequisites must exist within a project or organization to implement software processes? It contains practices related to resources, training, orientation, organizational structure, and utilities.

It is necessary to create such an environment that will adapt and provide the preconditions for success, regardless of the individual method used. As previously pointed out, these methods must be treated as hypotheses. The preconditions are general, familiar, and simple, but if considered carefully, it is clear that they are not always easy to provide.

If an organization agrees that these prerequisites for success must be met before any improvement program is launched, then there are important implications for the SPI program. Primarily, the program or project manager is partially relieved of responsibility for the success or failure of the program. Although the first prerequisite is leadership, the adoption of these principles has overcome this definition and the focus is evenly distributed on other prerequisites:

1. *Leadership*. The starting point for any successful project is leadership, focused on clearly defined goals and objectives. Unfortunately, this is hard to find in most small organizations and what is lacking even in large organizations. Leaders must be ready to remove all ambiguities and direct the organization towards defined goals and objectives.
2. *Commitment*. Management must show commitment to the adopted policy of improvement by its example so that other people can follow them. Also, commitment is required from every member of the organization. The level of this commitment may vary between individuals, but management must be balanced and build a team that will stand behind the project.
3. *Honesty*. Many organizations only rhetorically work on improvement projects and cannot face the truth about their environment. Organizations that are not honest in their intentions, as well as their shortcomings and failures, are too weak to take any action for improvement.
4. *Training*. Once the guidelines are adopted, the organization must mobilize to support them. Training on the chosen model or approach should show people the behaviors that are expected of them. The training itself often does not teach behavior, so the organization must carefully separate training on individual topics of behavior in specific situations.
5. *Professionalism*. Requirements identification, cost estimation, systems engineering, systems testing, and project management are all formal professions today. Each of these professions implies certain knowledge, so well-trained and skilled professionals must be appointed to such professional positions.

It is obvious that the dominance of the sociological component within all prerequisites causes their relation to factors from the category “People” from Table 1, whether they refer to SPI managers, developers, or other people. Such people must be skilled in software engineering and able to automatically adjust their activities to the optimal goals of the system during the program. The set of solutions or the solution to a recognized problem, as well as the time and effort required, will vary the most due to the different approaches of individuals and teams.

5 Conclusion

CMM models are one of the standard frameworks for software processes improvement. Along with their first introduction into the industrial community, they started a discussion about new approaches to achieve maximum improvement, with minimal costs, and in different types of software organizations.

The quality of the processes, as well as the use of statistical controls to maintain continuous progress, is constantly emphasized. However, the software is not similar to any other product and it is difficult to compare the path to its improvement with other products. Software development, like other design and engineering jobs, is not mechanized or disciplined production. It contains a strong creative component that includes human and social interaction and that cannot be fully planned in a standard or detailed process model.

An additional problem is the implementation of CMM-based models in small and medium-sized companies and organizations. Frequent ambiguity of the model and its complexity, in addition to all the previously identified risks, leave a lot of space for failure and abandonment of CMM-based improvement programs. In this paper the assumptions that an organization must meet in order to approach such general efforts are analyzed.

CMM does not describe how to create an effective software organization. It contains behaviors or best practices that such successful companies should demonstrate. If a company or organization is CMM compatible, it is not a guarantee that the software project will be successful, although it is true that this compatibility can increase the chances of the project being successful.

In order to reach the desired level of maturity, it is necessary to start and lead improvement programs. The empirical IDEAL model for improvement created in SEI provides a good foundation for initiating and leading a cycle of continuous improvement. This model uses already recognized techniques and methods of general management and change management, which together create an environment for overcoming business problems and creating a successful organization.

Small companies and organizations involved in software development and maintenance initially have a higher risk when implementing improvement programs based on CMM models. Characteristic factors for the success or failure of SPIs are most often related to general change management activities. SPI programs are, in fact, related to change management in a software organization. Although considered dynamic and ready for constant change in the business environment, risks of failure are pronounced when it comes to small organizations.

Software process improvement and project risk reduction require exceptional efforts to change organizational culture. If the organization is not able to provide such preconditions, then it is probably not worth spending resources on SPI.

References

[1] W. S. Humphrey, "Characterizing the software process: a maturity framework," *IEEE software,* vol. 5, no. 2, pp. 73-79, 1988.

[2] W. E. Deming, Out of the crisis, Cambridge, Mass.: Massachusetts Institute of Technology, Center for Advanced Engineering Study, 1986.

[3] C.Tully, "European SPI-Glass," *IEEE Software Process Newsletter,* no. 12, 1998.

[4] M. C. Paulk, "Using the Software CMM in Small Organizations," in *Proceedings of the 8th International Conference on Software Quality*, Portland, 1998.

[5] J. D.L. and B. J.G., "Tailoring the CMM for Small Business, Small Organizations, and Small Projects," *Software Process Newsletter, IEEE Computer Society Technical Council on Software Engineering,* no. 8, Winter 1997.

[6] Konrad, M. Chrissis, J. Ferguson, S. Garcia, W. Hefley, D. Kitson and M. Paulk, "Capability Maturity Modeling at the SEI," *Software Process: Improvement and Practice,* vol. 2, no. 1, pp. 21-34, 1996.

[7] P. Mark C., "A History of the Capability Maturity Model for Software," *ASQ Software Quality Professional,* vol. 12, no. 1, pp. 5-19, 2009.

[8] M. C. Paulk, "The Capability Maturity Model – A Summary," *Crosstalk: the Journal of Defense Software Engineering,* vol. 12, no. 5, p. 4, 1999.

[9] ISO, "Software Process Assessment - Part 4: Guidance on use for process improvement and process capability determination. Technical report. ISO/IEC 15504-4:2004," International Organization for Standardization, 2004.

[10] SEI, "CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1) Staged Representation. Technical report, CMU/SEI-2002-TR-012 ESC-TR-2002-012," Software Engineering Institute, 2002.

[11] C. Valentine and I. Richardson, "A practical application of the IDEAL model," *Software Process: Improvement and Practice,* vol. 9, no. 3, pp. 123-132., 2004.

[12] A. Laryd and T. Orci, "Dynamic CMM for Small Organizations," in *Proceedings of the First Argentine Symposium on Software Engineering*, Tandil, Argentina, 2000.

[13] R. V. Horvat, I. Rozman and J. Gyorkos, "Managing the Complexity of SPI in small companies," *Software Process: Improvement and Practice,* vol. 5, no. 1, p. 45–54, 2000.

[14] N. Wongsai, S. Veeraporn and W. Rattana, "Factors of influence in software process improvement: An ISO/IEC 29110 for very-small entities," in *2015 7th International Conference on Information Technology and Electrical Engineering (ICITEE), pp. 12-17. IEEE*, 2015.

[15] K. Steffen, P. Dietmar, J. Kristjan, D. Philipp, M. Jürgen and K. Marco, "How has SPI changed in times of agile development? Results from a multi-method study," *Journal of Software: Evolution and Process,* vol. 31, no. 11, 2019.

[16] W. S. Humphrey, Managing the software process, Boston: Addison-Wesley Longman Publishing Co., Inc., 1989.

[17] J. J. Abbott, "Software Process Improvement in a Small Commercial Software Company," in *1997 Software Engineering Process Group Conference*, San Jose, CA, 17-20 March 1997.

[18] D. L. Johnson and J. G. Brodman, "Applying the CMM to Small Organizations and Small Projects," in *1998 Software Engineering Process Group Conference, Chicago*, Chicago, IL, 9-12 March 1998.

[19] H. Conradi and F. Alfonso, "Improving software process improvement," *IEEE software,* vol. 19, no. 4, pp. 92-99, 2002.

[20] K. Kautz, "Software process improvement in very small enterprises: does it pay off?," *Software Process: Improvement and Practice,* vol. 4, no. 4, pp. 209-226, 1998.

[21] S. Rifkin, "Is process improvement irrelevant to produce new era software?," in *European Conference on Software Quality, pp. 13-16. Springer*, Berlin, Heidelberg, 2002.

[22] I. Allison, "Organizational factors shaping software process improvement in small-medium sized software teams: A multi-case analysis," in *2010 Seventh International Conference on the Quality of Information and Communications Technology, pp. 418-423. IEEE*, 2010.

[23] T. Micheal and A. Yirsaw, "Factors affecting development process in small software companies," in *2019 IEEE/ACM Symposium on Software Engineering in Africa (SEiA), pp. 16-23. IEEE*, 2019.

[24] K. Marco, D. Philipp and M. Jürgen, "Software process improvement: a systematic mapping study on the state of the art," *PeerJ Computer Science,* no. 2, 2016.

[25] D. Tore, "Factors of software process improvement success in small and large organizations: an empirical study in the scandinavian context," *ACM SIGSOFT Software Engineering Notes,* vol. 28, no. 5, pp. 148-157, 2003.

[26] F. Alfaro, S. Cynthia and D. Abraham, "CMMI Adoption and Retention Factors: A Systematic Literature Review," in *International Conference on Software Process Improvement, pp. 15-28. Springer*, Cham, 2021.