A Longitudinal Investigation about Issues and Problems of Software Engineering in Turkey

Melih Kirlidog, North-West University (Vaal Triangle Campus), Vanderbijlpark, South Africa, melihk76@gmail.com
Meric Aykol, Istanbul University, Istanbul, Turkey, meric.aykol@gmail.com

Abstract

This article reports the findings of a longitudinal research about the maturity of software practices in Turkey. Two surveys were conducted in 2001 and 2008 in order to get the perceived maturity levels in SWEBOK's knowledge areas of software engineering. SPICE capability levels were used for gauging maturity levels. It was found that software practices in the country are usually far from being mature and not much has changed in the time period of 2001-2008.

Keywords: CMMI, SPICE, SWEBOK, Turkey, software quality

Introduction

Software development constitutes one of the most striking dilemmas in our age: although software is associated with speed and automation of complicated tasks, its development mainly bears resemblance to traditional crafts such as painting. In traditional crafts there are some traits that cannot be automated or "sped up." Those crafts rely on the intuition of the craftsman and that intuition is required to pre-design the finished product in mind. This pre-design is not lost from the "sight" by the craftsman during the entire development process. Although a craftsman can develop his/her intuition in time her/his mind must be prone to accommodate the required skills that come naturally and there is little room for speeding up the product development process beyond a certain threshold. Just like the gifted craftsman's talent that comes by birth, some software developers are much better than the others. Quoting Sackman et al (1968) in his seminal book about software development, Brooks (1995) reports that best and worst programmers' productivity varies about 1:10 and their speed varies about 1:5. Brooks also argues that software development is a *creative* work and unlike the hardware development, there is no *silver bullet* of technology or management technique that can increase the productivity in software development in short time.

Yet, such a dependency on natural skills is relevant particularly on individual level. Excluding the Free and Open Software development (Stallmann, 2009) where individual level engagement is significant, large software development tasks are always organizational issues. Recruitment and management of those developers require a high level of organizational effectiveness. Another reason that emphasizes the significance of organizational effectiveness is that unlike cotton picking which can be partitioned, software development is a task that cannot be partitioned, because it involves extensive sequential constraints and communication requirements increases exponentially with the size of the project (Brooks, 1995).

Quality of software is a complicated phenomenon affected by a myriad of factors. Those factors determine not only how well the software is designed, but also the level of compliance of the final product to the design. Since software production involves special types of processes where craftsmanship is important, quality of software is directly related with the quality of the processes used in its production.

Software quality and increasing an organization's effectiveness in software development has been subject to intensive research. This body of research is usually centered on the concept of Software Process Improvement (SPI) (Pino et al, 2008; Niazi & Babar, 2009; Staples et al, 2007). Some assessment frameworks like CMMI (Capability Maturity Model Integration) (Chrissis et al, 2003) and SPICE (Software Process Improvement and Capability Determination) (Rout et al, 2007) have been developed for SPI endeavors. IEEE's Software Engineering Body of Knowledge (see www.SWEBOK.org) (Bourque, et al, 1999) is another SPI approach that defines the required knowledge and recommended practices in software engineering.

This article reports the findings of two research surveys that have been conducted in 2001 and 2008 in Turkey. It aims to interpret the results of the surveys using SPICE as a guideline, thereby attempting to understand the maturity level of the software engineering practices in Turkish companies. The 2001 study was conducted by the Turkish Quality Association's (TQA) Total Quality Management in Software Working Group and it was based on the ten software engineering "Knowledge Areas" of the SWEBOK. This research was reported by Aytac et al (2003). The 2008 study was conducted with the same survey instrument by the permission of the TQA. Using the same instrument in both surveys allowed a longitudinal comparison to be made in the time frame of 2001-2008.

This article is organized as follows: The next section elaborates the approaches for determination of software maturity and it is followed by the literature study about those approaches. The next section describes the methodology and data collection in this research. It is followed by the analysis of the findings and conclusion sections.

Software maturity determination

Determining the capabilities of software developing organizations is a formidable task that involves logically integrating a myriad of parameters about the organizations and their activities. CMMI, the most commonly used approach, has been evolved from CMM (Capability Maturity Model) that was introduced by Carnegie Mellon Software Engineering Institute (SEI) in 1987. Methodologies like CMM can be regarded as a response to the perceived "software crisis" of the 1980s (McFadden & Discenza, 1987; Georgiadou, 2003)**.** The crisis was a result of two interrelated events: there was a fast spread of computers with scientific and commercial applications and the power of computers increased even faster. As a result, software development was required for a very broad range of applications and the software had to be much more sophisticated than previous decades for accommodating the powerful hardware as well as satisfying the ever-increasing user demands. Such pressing requirements, in turn, resulted in inadequate and poor quality software with unsystematic software development practices, cost overruns, and long queues of backlogs. (It must also be stated that some authors like Glass (2006) do not agree on the notion of "software crisis" as perceived in the 1980s. Nevertheless, there was consensus about a pressing requirement for a methodological approach to software development in that decade and beyond.)

CMM is a process-oriented methodology and it was initially developed to assess the process capabilities of the software contractor companies that serve the US Government. Although the model concentrates on the process of software development it can also be used for other software-related processes such as project and risk management. Watts Humprey, one of the key developers of the CMM, has published the main tenets of the model in his book (Humprey, 1989) and the model was explained in detail by some other developers (Paulk et al, 1995). CMM assesses the companies according to their maturity levels in performing their software-related *processes* and the model has five distinct levels with clearly defined characteristics (Tayntor, 2007):

1. Initial – results are unpredictable, because they are dependent on individuals' skills and efforts.
2. Repeatable – basic processes have been established on a project level, making it possible to replicate performance on similar projects.
3. Defined – standard processes have been integrated across the IT organization and are used consistently on all IT projects.
4. Managed – detailed measurements and quantitative controls make it possible to predict results.

5. Optimizing – the organization actively seeks to improve the process through innovation.

CMM has a formal assessment process for not only determining at what level an organization is, but also the steps that must be taken to reach the next level. The assessment of an organization is performed by visiting experts and the organization is graded in one of the five levels explained above. CMM contains Key Process Areas (KPAs) in each level and each KPA consists of five activities, namely goals, commitment, ability, measurement, and verification. There are also clearly defined milestones and deliverables in each level and a company must satisfy the relevant requirements in order to promote from one level to another.

The development of CMM was ceased in 1997 and the broader concept of CMMI was introduced in 2002 again by the SEI. Unlike the CMM which is purely a staged framework CMMI has both staged and continuous representations. CMMI certification is currently sought by several public and private organizations for software bidding all over the world.

Unlike CMMI which grades the entire organization SPICE defines capability levels for *individual processes* in the organization. SPICE was initially developed in 1993 and had its current form of ISO/IEC 15504 standard as a result of the joint effort of International Organization for Standardization and International Electrotechnical Commission. SPICE is both an assessment method and a process model.

SPICE is consistent with CMMI. Rout and Tuffley (2007) report the results of an analysis of the relationship between CMMI and SPICE. The authors argue that CMMI addresses most of the matters addressed in the Measurement Framework as defined in ISO/IEC 15504, providing a basis for conversion of data.

The capability levels of SPICE are as follows:

0. Incomplete process
1. Performed process
2. Managed process
3. Established process
4. Predictable process
5. Optimizing process

Software development can be regarded as an engineering activity whose focus is the cost-effective development of high-quality software systems. However, unlike other engineering disciplines where the final product is tangible, software is an abstract and intangible product. Beyond the nature of the final product, this intangibility is also reflected in the software development process as software is not constrained by materials nor governed by physical laws (Sommerville, 2007). Thus, software engineering differs from traditional engineering disciplines due to its emphasis on abstract concepts such as *design, management*, and *processes*.

These abstract concepts are addressed by the ten knowledge areas of the SWEBOK. In the 2004 version of the SWEBOK the knowledge areas are grouped under software design and management. The structure of SWEBOK is as follows (for sub-items for each knowledge area see http://www2.computer.org/portal/web/swebok/html/ch1#Ref9):

a) Software design
    1) Software requirements
    2) Software design
    3) Software construction
    4) Software testing
    5) Software maintenance
b) Software management
    1) Software configuration management

2) Software engineering management
3) Software engineering process
4) Software engineering tools and methods
5) Software quality

Previous studies

Although systematic approaches like CMMI and SPICE offer important benefits to software developing organizations they require significant money and effort with no guarantee of success at the end (Kautz & Nielsen, 2000). Bigger companies that can command larger resources are more prone to success than the smaller ones in certification. In a literature review about the software process improvement attempt of small and medium enterprises (SME) Pino et al (2008) found that it is more difficult for the SMEs to be successfully certified in models like CMMI or SPICE.

There are a number of studies that investigate organizations according to their maturity levels in software-related tasks. The European Software Institute (ESI) developed "software best practices questionnaire" in 1997 (see www2.umassd.edu/SWPI/esi/tr-sbpqaor3.pdf) that aimed to collect data about the adoption of software practices in European organizations. The questionnaire contained 42 questions organized in five sections, namely organizational issues, standards and procedures, metrics, control of the development process, and tools and technology. 394 organizations from 20 countries answered the questionnaire. The results revealed that about half of the organizations that participated in the research adopted the best practices and there were significant variations among countries and sectors in terms of the adoption levels (Dutta et al, 1999).

Wilkie et al (2005) have investigated six small and medium software companies in Northern Ireland about the perceived value of six of the seven CMMI process areas in maturity level 2, namely requirements management, configuration management, project planning, project monitoring and control, measurement and analysis, and process and product quality assurance. As can be expected, the authors found that the companies tended to focus on end product quality assurance rather than process quality assurance and relied more heavily on individual developer competence as opposed to process. In a similar study, Niazi and Babar (2009) examined 46 software practitioners working in Vietnamese and Malaysian SMEs about their perception for the CMMI level 2 practices. The authors found that some requirements management practices were perceived to have a higher value than the others and recommended "tailoring" existing frameworks to match the requirements of the SMEs based on the perceived value of the practices.

Methodology and data collection

As stated above, the 2001 and the 2008 surveys contain the same questions and they are based on the SWEBOK's ten knowledge areas. The questionnaires have been prepared in Turkish language. In developing the questionnaires the SWEBOK technical terms were translated into Turkish language by using the Software Quality Assurance Dictionary of the Turkish Informatics Foundation. The surveys mainly addressed practitioners rather than organizations. In other words, practitioners were asked to reflect their thoughts about software practices in their organizations and data for some organizations appeared in the surveys by more than one respondents. As a result, although there could be conflicting views about the software practices in a specific organization, this approach allowed gaining insight about a wide range of organizations in operating in several industries as well as software companies.

The questionnaires contain one hundred questions plus six questions for determining the demographics of the participants. The participants were assured about their anonymity to third parties so that objective results would be obtained rather than formal corporate declarations. The last question in each knowledge area is a generic one used to check the validity of the answers in the relevant area.

Data collection for the 2001 survey was conducted between the dates of 12 November – 31 December 2001. The declared objective of this survey was to identify problem areas which hamper the growth and competitiveness of Turkish Software Industry. TQA intended to use the results as a basis for prioritizing improvement actions and suggesting policies to cure identified deficiencies. There were 68 usable entries in the survey and the participants worked in 50 organizations.

The 2008 data collection process was conducted between the dates of 14 September and 10 November 2008. The researchers have determined 100 ICT professionals who actively work in diverse areas of software engineering and invited them to participate in the research with e-mail or personnel communication. Data collection was performed through a web-based online system and 81 people have participated in the research. Disregarding six entries for various reasons such as quite sparse answers to the questions, there were 75 usable entries. Demographics of the participants of the 2001 and 2008 research are shown in Table 1:

Table 1: Demographics of the participants

		2001	**2008**
Participants' job definition	Software engineer	22	20
	Quality or process engineer	7	4
	Other	39	51
Participants' job experience	0-1 years	5	1
	2-4 years	16	12
	5-9 years	23	24
	10-19 years	20	27
	=> 20 years	4	11
Participants' main activity	Software contracting company	30	44
	Software developer for internal use	24	28
	Packaged software developer company	14	3
Participants' organizations: Sector	Production	34	25
	Finance	14	14
	Services	14	28
	Public	6	8
Participants' organizations: Number of software personnel	1-9	20	30
	10-49	19	22
	50-99	16	9
	=>100	13	14
Total		**68**	**75**

The table shows that the majority of the participants are employees or owners of software companies. Only 24 of the 68 participants in 2001 and 28 of the 75 participants are software developers for internal use. Thus, it can be argued that the results are particularly relevant for Turkish software industry.

The questionnaire is organized in two parts. The first part is formed by the first nine questions in each knowledge area and the tenth question in each knowledge area constitutes the second part.

The first part aims to get a deep insight into the software practices with quite detailed questions. Unlike traditional questionnaires with Likert scale, this part of the survey has been designed in a flexible way. This means that each question had different set of answers and the participants can mark more than one item in a group. For example, the eighth question in the software requirements knowledge area is as follows:

8. Which of the following best characterizes the practice of requirements change management in your organization?

- We cannot claim that we perform change management.
- We rely on personal interest and skills of developers.
- Requirement changes are performed by designated personnel but are not documented.
- We follow an institutionalized but ineffective change management process.
- We follow an institutionalized process, record all changes, and inform all related parties.
- None of the above.

Although this non-uniformity does not allow for an advanced statistical analysis, it facilitates getting a deeper insight into the problem surveyed. What is important in this organization is to foresee and include all possible cases. Since there could be cases that are not covered by any of the options in a question, most questions have a "none of the above" option in this part.

Unlike the first part which is too detailed to be analyzed in an article, the second group is uniform across the ten knowledge areas and they reflect the scale levels of SPICE mentioned above. In other words, after detailing their organizations' software practices in the first nine questions for each knowledge area of the SWEBOK, the tenth questions for each knowledge area aim to get the general perceptions of the participants about their organizations' ranking in terms of the SPICE levels. Thus, the tenth questions can be regarded as a summary of their corresponding knowledge areas. Possible answers and their corresponding SPICE levels in this group are shown in Table 2:

Table 2: SPICE levels

SPICE level	
0. Incomplete process	We cannot claim to be successful in this area.
1. Performed process	We achieve results, but in an ad hoc fashion and mostly by individual effort.
2. Managed process	We achieve results in a planned fashion, but we don't have an institutionalized process.
3. Established process	We achieve results by following an institutionalized process.
4. Predictable process	We achieve results by following an institutionalized and quantitatively controlled process.
5. Optimizing process	We achieve results by following an institutionalized, quantitatively controlled, and continuously improving process.

Analysis of this group, which is the subject of the next section, gives an idea about software practices in Turkish organizations.

Findings and discussion

It was attempted to calculate and compare the perceived sophistication in the ten knowledge areas. This was realized by a weighted calculation approach where a single value was calculated for each knowledge area. The value is based on the answers given to the tenth questions in each knowledge area and it reflects the SPICE levels of zero to five. The calculation is performed as follows: the percentage of respondents for each level is found and this percentage is multiplied by the number designating that level. Then, results for each level are added to find the general value of the knowledge area (summation errors in all tables are due to rounding.) Since the inputs in this calculation are based in SPICE levels, the outputs also can be regarded to reflect the SPICE level of Turkish organizations in each knowledge area. The calculations are performed separately for the 2001 and 2008 surveys. As an example, Table 3 shows the calculation for the Software Requirements knowledge area.

Table 3: Calculation of the perceived sophistication levels for Software Requirements knowledge area

SPICE level	2001	2008	Coefficient	2001 Point	2008 point
0. Incomplete process	2%	6%	0	-	-
1. Performed process	17%	19%	1	0.17	0.19
2. Managed process	31%	37%	2	0.62	0.73
3. Established process	36%	21%	3	1.07	0.62
4. Predictable process	2%	6%	4	0.10	0.25
5. Optimizing process	12%	11%	5	0.60	0.56
Total points				**2.55**	**2.35**

This calculation is performed for all knowledge areas and results are presented in the "overall" columns of Tables 4-6. The tables also contain detail calculations for divisions such as participants' job definition, size and main activities of the organizations. Overall values are repeated in each table for easy comparison within the table.

Table 4: The perceived sophistication levels detailed for participants' job definition

	Overall		**Software engineers**		**Quality or process engineers**		**Others**	
SWEBOK knowledge area	**2001**	**2008**	**2001**	**2008**	**2001**	**2008**	**2001**	**2008**
1. Software requirements	2.55	2.35	2.83	2.00	2.25	3.25	2.46	2.38
2. Software design	2.32	2.22	2.85	2.20	1.75	2.67	2.15	2.19
3. Software construction	2.65	2.49	3.00	2.30	2.25	4.00	2.46	2.45
4. Software testing	2.33	2.08	2.69	1.80	2.25	3.00	2.14	2.04
5. Software maintenance	2.58	2.51	2.93	3.00	1.80	3.00	2.53	2.42
6. Software configuration management	1.74	1.96	1.73	1.50	3.00	4.00	1.57	1.78
7. Software engineering management	2.18	2.28	2.22	2.00	2.75	3.00	2.05	2.27
8. Software engineering process	2.03	1.96	1.50	3.00	2.40	2.67	2.20	1.63
9. Software eng. tools and methods	1.71	2.00	1.60	3.50	1.50	3.00	1.80	1.75
10. Software quality	1.71	1.77	1.71	1.00	1.67	2.75	1.67	1.68
Average	**2.18**	**2.16**	**2.31**	**2.23**	**2.16**	**3.13**	**2.10**	**2.06**

Table 5: The perceived sophistication levels detailed for the main activity of participants' organizations

SWEBOK knowledge area	Overall		Developers for internal use		Software contractors		Packaged software developers	
	2001	**2008**	**2001**	**2008**	**2001**	**2008**	**2001**	**2008**
1. Software requirements	2.55	2.35	2.07	2.09	2.81	2.56	2.83	1.67
2. Software design	2.32	2.22	1.60	2.00	2.86	2.32	2.25	2.50
3. Software construction	2.65	2.49	2.14	2.23	2.87	2.64	2.89	2.00
4. Software testing	2.33	2.08	2.00	1.82	2.53	2.21	2.43	2.00
5. Software maintenance	2.58	2.51	1.54	2.71	3.00	2.40	3.43	3.00
6. Software configuration management	1.74	1.96	1.67	1.25	1.89	2.18	1.25	2.50
7. Software engineering management	2.18	2.28	1.58	2.33	2.53	2.33	2.50	1.50
8. Software engineering process	2.03	1.96	1.83	1.78	2.13	2.00	2.13	2.00
9. Software eng. tools and methods	1.71	2.00	1.50	2.00	1.57	2.00	2.25	2.00
10. Software quality	1.71	1.77	1.75	1.63	1.55	1.84	2.00	2.00
Average	**2.18**	**2.16**	**1.77**	**1.98**	**2.37**	**2.25**	**2.40**	**2.12**

Table 6: The perceived sophistication levels detailed for the organizations' size in terms of personnel numbers

SWEBOK knowledge area	Overall		1-9		10-49		50-99		=> 100	
	2001	**2008**	**2001**	**2008**	**2001**	**2008**	**2001**	**2008**	**2001**	**2008**
1. Software requirements	2.55	2.35	2.13	1.78	3.25	2.24	2.63	2.50	2.55	3.62
2. Software design	2.32	2.22	1.73	2.18	3.30	1.79	2.60	2.67	1.89	3.00
3. Software construction	2.65	2.49	2.00	2.60	3.00	2.00	3.00	2.20	2.78	3.38
4. Software testing	2.33	2.08	1.33	1.79	3.44	1.92	2.67	2.25	2.22	3.00
5. Software maintenance	2.58	2.51	1.80	2.38	3.00	2.27	3.30	2.80	2.30	3.14
6. Software configuration management	1.74	1.96	0.91	1.25	2.25	1.90	1.71	1.75	2.33	3.20
7. Software engineering management	2.18	2.28	1.11	1.80	2.67	2.25	2.78	1.50	2.33	3.22
8. Software engineering process	2.03	1.96	1.13	1.63	2.00	1.50	2.73	2.00	2.00	3.00
9. Software eng. tools and methods	1.71	2.00	1.11	1.78	1.33	2.00	2.00	1.50	2.22	2.75
10. Software quality	1.71	1.77	1.17	1.22	1.64	2.00	1.92	1.00	2.20	2.60
Average	**2.18**	**2.16**	**1.44**	**1.84**	**2.59**	**1.99**	**2.53**	**2.02**	**2.28**	**3.09**

The overall results show that the 2001 and the 2008 surveys are mainly consistent and there are only slight differences between them. Yet, during this time there has been an accelerated pace of diffusion of computer applications in all areas and it could be expected that the sophistication levels of software practices would also increase. Thus, this is not good news for the Turkish software industry which has the stated objective of opening into foreign markets. The general averages of 2.18 and 2.16

indicate SPICE level 2 where the results are achieved in a planned fashion, but institutionalized process is lacking. This level of sophistication is obviously inadequate for developing a world class software industry. Further, on the average, the knowledge area of software quality has consistently the least point among others in both surveys and there has been only a very slight improvement in this area. This is also not encouraging for the industry.

On the overall, there have been slight improvements in the elapsed period in four knowledge areas, namely software configuration management, software engineering management, software engineering tools and methods, and software quality. Software engineering tools and methods have seen the highest improvement by increasing from 1.71 to 2.00. This could be regarded as an encouraging development, because using proper tools and methods may result in improvement in other knowledge areas.

The knowledge areas of software requirements, software construction, and software maintenance have comparatively higher points. However, these areas could be regarded as “lower level” or core areas in software engineering. Other areas such as software configuration management, software engineering process, and software engineering management are related to the “upper level” or management of the processes. Comparatively lower points in the latter group indicate low level of development of software practices in the country.

Table 4 shows the divisions according to the job definition and it is striking to note that although software engineers and other personnel do not indicate any substantial improvement from 2001 to 2008, quality and process engineers perceive a stark increase in the average points by 2.16 to 3.13. This could perhaps be interpreted as stemmed for concentrating on their job tasks where the primary mission is to improve software quality. Quality personnel’s satisfaction with their work does not seem to be shared with others.

Details according to the main activity of the participants’ organization are shown in Table 5. Developers for internal use get less point (1.77-1.98) than software contractors (2.37-2.25) and packaged software developers (2.40-2.12). This shows that the organizations that develop software for the market tend to be more quality-conscious than the ones that develop bespoke software.

Table 6 shows the sophistication levels detailed according to the size of the organizations in terms of personnel numbers. In Turkey, software companies are generally small in size and very few of them employ more than one hundred people. As can be expected, very small organizations with 1-9 employees have the least points (1.44-1.84). However, although points generally increase in larger organizations, a recognizable pattern such as the increase of points proportional to the increase in size does not exist. It is worth noting that the robust increase of points for the organizations with more than one hundred employees (2.28-3.09). This can be an important clue for the determination of larger organizations to open up into export markets.

Conclusion and future work

Although there is a wide body of literature about software quality some of which is about quality improvement approaches such as CMMI and SWEBOK, to our best knowledge, this article is the first attempt to use SWEBOK for gauging the software process and product quality of companies in a country.

As stated, the survey instrument used in this research has two parts. While the first part allowed getting a deep insight into the issues and problems, the second part facilitated quantitatively analyzing and comparing the knowledge areas in software engineering. Since the first part contains comprehensive material that cannot be covered within the context of this article it will be the topic of a future article.

The two surveys indicate that the there are serious problems and inadequacies in the area of software development and maintenance in Turkey. Further, the situation has not improved during the time frame of seven years between 2001 and 2008. Software projects are not professionally handled and they are not conducted with an approach that emphasizes the processes. An overwhelming majority of the organizations do not follow any standard in software development and maintenance efforts. The most common method of knowledge transfer is not formal training-in-job; rather, it is the knowledge transfer from the craftsman to the apprentice. It is an irony that although software development is mainly a craft, knowledge transfer in software engineering can not be effectively handled like the ones in traditional crafts.

Software development tools and methods are usually either unknown to the developers or they are not used even if they are known. In the latter case it is usually to avoid the time and effort overhead that comes with the tools and methods. Their absence makes it very difficult to benchmark development or maintenance efforts.

Software design is usually performed by project manager whose task is often not to *manage* the projects, but rather to work as a developer or integrator. This is particularly problematic for large-scale projects and usually leads to serious flaws in the crucial management process.

Software quality is proportional to the resources devoted to it. Since the Turkish software industry is extremely fragmented with thousands of micro-sized organizations there is usually a severe lack of resources that can be directed for quality improvement. Although larger organizations have more resources for quality assurance they are quite few in numbers. The net result is a general overlook to the quality issues in software engineering.

Although there have been some progress in some areas such as large organizations' increasing awareness for quality issues, there have been little or no improvement in software engineering issues in Turkey in the last decade. Turkish software organizations need to relinquish amateurism and get accustomed to present-day tools and methods of software engineering in order to transform themselves into global players.

References

Aytac, T., Ikiz, S. and Aykol, M. (2003), "A SPICE-oriented, SWEBOK based software process based assessment on a national scale: Turkish sector software survey – 2001," Proceedings of the Joint ESA-3rd International SPICE Conference on Process Assessment and Improvement, 17-21 March 2003, Noordwijk, The Netherlands.

Bourque, P., Dupuis, R. and Abran, A. (1999), "The guide to the software engineering body of knowledge," *IEEE Software*, 16 (6), 35-44.

Brooks, F. P. (1995) The Mythical Man-Month, Anniversary Ed., Addison-Wesley, NY.

Chrissis, M. B., Konrad, M. and Schrumm, S. (2003) CMMI: Guidelines for Process Integration and Product Improvement, Addison-Wesley, Boston, MA.

Dutta, S., Lee, M. and Van Wassenhove, L., (1999), "Software engineering in Europe: a study of best practices," *IEEE Software*, 16 (3), 82-90.

Georgiadou, E. (2003), "Software process and product improvement," *Cybernetics and System Analysis*, 39 (1), 125-142.

Glass, R. L. (2006), "The Standish Report: does it really describe a software crisis?" *Communications of the ACM*, 49 (8), 15-16.

Humprey, W. S. (1989) Managing the Software Process, Addison-Wesley Longman Publishing Co., Boston, MA.

Kautz, K. and Nielsen, P. A. (2000), “Implementing software process improvement: two cases of technology transfer.” Proceedings of the 33rd Hawaii Conference on System Sciences, vol. 7, Maui, USA, 1–10.

McFadden, F. and Discenza, R. (1987), “Confronting the software crisis,” *Business Horizons*, 30 (6), 68-73.

Niazi, M. and Babar, M. A. (2009), “Identifying high perceived value practices of CMMI level 2: An empirical study,” *Information and Software Technology*, 51 (8), 1231-1243.

Paulk, M. C., Weber, C., Curtis, B. and Chrissis, M. B. (1995) The Capability Maturity Model: Guidelines for Improving Software Process, Addison-Wesley, Reading, MA.

Pino, F. J., Garcia, F. and Piattini, M. (2008), “Software process improvement in small and medium software enterprises: a systematic review,” *Software Quality Journal,* 16 (2), 237–261.

Rout, T. P. and Tuffley, A. (2007), “Harmonizing ISO/IEC 15504 and CMMI,” *Software Process Improvement and Practice,* 12, 361–371.

Rout, T. P., El-Emam, K., Fusani, M., Goldenson, D. and Jung, H. (2007), “SPICE in retrospect: developing a standard for process assessment,” *The Journal of Systems and Software,* 80 (9), 1483-1493.

Sackman, H., Erikson, W. J. and Grant, E. E. (1968), “Exploratory experimental studies comparing online and offline programming performance,” *Communications of the ACM*, 11 (1), 3-11.

Sommerville, I. (2007) Software Engineering, 8th ed. Addison-Wesley, London.

Stallmann, R. (2009), “Why ‘open source’ misses the point of free software,” *Communications of the ACM*, 52 (6), 31-33.

Staples, M., Niazi, M., Jeffery, R., Abrahams, A., Byatt, P. and Murphy, R. (2007), “An exploratory study of why organizations do not adopt CMMI,” *The Journal of Systems and Software*, 80 (6), 883–895.

Tayntor, C. B. (2007) Six Sigma Software Development, Auerbach Publications, Boca Raton, FL.

Wilkie, F. G., McFall, D. and McCaffery, F. (2005), “An evaluation of CMMI process areas for small-to-medium-sized software development organizations,” *Software Process Improvement and Practice,* 10 (2), 189-201.