Evaluation of Quantitative Process and Software Quality Management in the Nigerian Software-House

Moses Kehinde Aregbesola
Department of Mathematics and Computer Science
Elizade University,
Ilara-Mokin, Ondo State, Nigeria

ABSTRACT

The Software Engineering Institute (SEI) Capability Maturity Model Integration (CMMI) is made up of 5 maturity levels namely Initial, Managed, Defined, Quantitatively Managed, and Optimizing. Each maturity level consists of Key Process Areas (KPA) each of which in turn consists of key practices. Recent studies have shown that many developing countries rank poorly on this maturity model. The current study evaluated the level of performance of the practices associated with the two KPAs at maturity level 4 in a typical Nigerian software-house. The SEI CMMI Organizational Process Performance and Quantitative Project Management KPAs at maturity level 4 (Quantitatively Managed) are depicted as Software Quality Management (SQM) and Quantitative Process Management (QPM) respectively in the SEI maturity questionnaire employed in conducting the study. The survey study was conducted across 30 different software-houses within the country. The study equally employed the action research approach with some of the selected companies which were nominated for more detailed investigation. The study revealed weak performances in both KPAs but with a better performance of the practices associated with the SQM KPA. The findings from the current study were observed to be consistent with findings from similar studies in other developing countries. The causes of the observed weak performances, including unawareness of the existence of laid down international standards, non-adherence to such standards and inadequate knowledge about the required process improvement techniques, were discussed. Solutions, such as the institutionalisation of formal standard indicators for QPM and SQM with associated functional definitions, measurement methods, and analysis models, were proffered to address the causes of the weak performances experienced in both KPAs. Also, adequate resources in terms of time, budget, bureaucracy, tools, training, organisational framework, senior management support, common understanding and patience was equally advocated for QPM and SQM practices so as to forestall the occurrence of any unforeseen overhead. It was equally suggested that quality reviews for assessing software quality should be performed as often as possible and should secure the full support of organisational top management. Quality management activities should also be separated from project management activities so as to guarantee the independence of the quality management reviews. Finally, organizational top managements were encouraged to enforce strict adherence to QPM and SQM practices across the length and breadth of the organization. It is believed that if the proffered solutions are adopted, the software-houses will rank higher on the CMMI maturity scale and most likely experience better patronage.

Keywords

Capability Maturity Model, CMMI, Key Process Areas, KPA, Quantitative Process Management, QPM, Software Quality Management, SQM, Quality Management, QM, Software Process, Software Industry, Nigerian Software-House, Nigerian Software Industry.

1. INTRODUCTION

The production of software presents developing countries with many potential benefits such as job creation and foreign exchange [1, 2]. However, many software companies in developing countries major in selling software products and services to the domestic market, typically representing a survival strategy more than a developmental one.

Nigeria is a federal republic in West Africa consisting of about twenty percent of black Africa's population and with a typical African state profile with regards human development indicators and technological infrastructure [3]. Considering her large population, Nigeria constitutes a strategic market for software applications in the African continent and her software industry plays a strategic role in the West African software experience.

The study of Soriyan and Heeks [3] on software process methodologies employed by software companies in Nigeria revealed a significant reliance on formal software methods developed in-house rather than reliance on industry standards. Similar observations were equally made in comparable studies affecting other developing countries such as Turkey [4]. Heeks [1, 2] equally revealed that many software systems in developing countries experience some form of partial or total failure as a result of a design-reality gap caused by the absence of a functional software process model or the use of an immature one.

In agreement with a similar study by Sowunmi *et al.* [4] on Turkey, the study of Aregbesola and Akinkunmi [5, 6] and Aregbesola, Akinkunmi and Akinola [7] which evaluated the Nigerian software industry placed its software process at the CMMI maturity level 1. The current study is therefore focused on evaluating the performance of the Nigerian software industry on portions of the SEI CMMI software process model not hitherto covered in previous studies. The Software Quality Management (SQM) and Quantitative Process Management (QPM) are the key process areas of interest in the current study. The evaluation was conducted using the SEI Maturity Questionnaire [8] as the main tool for information gathering. The survey study was conducted across 30 software companies in Nigeria, employing action research with some of the companies earmarked for more detailed case studies.

2. QUANTITATIVE PROCESS MANAGEMENT AND SOFTWARE QUALITY MANAGEMENT

The Capability Maturity Model Integration (CMMI) is an outstanding global framework for process enhancement employed by competitive organizations that wish to achieve high performance in their operational activities [9, 10]. The CMMI is often adopted as the preferred approach for process improvement across several industries including but not limited to software development and engineering. The CMMI is made up of 5 maturity levels, that is, Initial, Managed, Defined, Quantitatively Managed, and Optimizing. The individual maturity levels each consist of Key Process Areas (KPA) which in turn consists of a number of associated key practices. The Organizational Process Performance and Quantitative Project Management key process areas of the CMMI are represented as Software Quality Management (SQM) and Quantitative Process Management (QPM) respectively in the maturity questionnaire [8] employed in the current study. Quantitative Process Management (QPM) and Software Quality Management (SQM) are the KPAs at maturity level 4, Quantitatively Managed. While QPM is in the management process category, SQM is in the engineering process category [11, 12, 9, 13]. To forestall the mythical misconceptions in some circles about the CMMI not being agile, a number of works including those of Mogre and Salunkhe [14] as well as Glover and Dennie [9] looked at a view on how to be agile with CMMI. That is, implementing CMMI in an Agile environment.

2.1. Quantitative Process Management

Quantitative Process Management (QPM) is mainly aimed at controlling the process performance of the software project quantitatively [15, 16]. By software process performance, the actual results achieved from following a software process is what is being referring to. QPM brings a comprehensive measurement program to the practices of Organization Process Definition, Integrated Software Management, Intergroup Coordination, and Peer Reviews KPAs [15, 17, 18, 19]. Several studies including those of Campo and Smith [20], INCOSE [21], Madachy [22], Niessink and Vliet [23], Kinnula [24], Sargut [25], Keraminiyage *et al.* [26], and Paulk *et al.* [15, 16] discussed the QPM as a KPA concerned with instituting goals for the performance of the project's defined software process described in the integrated software management KPA, measuring the process performance, analyzing the measurements taken, and making all necessary adjustments possible within acceptable limits as appropriate. Upon achieving a stable process performance within acceptable limits, a new baseline is then established for controlling process performance quantitatively using the project's defined software process, the associated measurements, and the acceptable limits. The process capability of the organizations standard software process is then characterized using the process performance data collected from the software projects and subsequently described in the organization process definition KPA. The process capability is typically the process performance, described by the range of expected results, a new project within the organisation can expect to attain from following a software process. Each subsequent project then employs the capability data in establishing and revising process performance goals as well as to examine the performance of the projects' defined software processes. Typical issues and questions associated with the exploratory data analysis involved in initiating quantitative process management was the focus of the study of Paulk [27]. Operational definitions, process consistency, aggregation, and organizational implications were some of the issues considered as important for analyst to cover as a roadmap in implementing quantitative process management.

A common challenge observed in many software companies was captured in the work of Hikichi *et al.* [28] who studied software companies that performed some form of QPM. The study observed that in many organizations, the definition of standard indicators was briefly described in natural language, giving very little or no formal representations of the associated functional definitions, measurement methods, or analysis models. Popa [29] explained that QPM in a typical audit processes is developed on audit process customization, audit measurement, history data and indicators together with their statistical control. Popa [29] in agreement with earlier studies described QPM in informatics audit processes as consisting of three key activities: establishing the goals for audit process performance; analyzing the result indicators; and implementation of process adjustments to maintain it within acceptable limits.

2.2.Software Quality Management

Software Quality Management (SQM) on the other hand has the sole purpose of developing a quantitative understanding of the project's software products' quality and achieving specific quality goals [15]. Quantitative goals are established for the software products on the basis of the needs of the organization, the customer, and the end users. To achieve the set goals, the organization establishes strategies and plans, while the project carefully adjusts its defined software process to realize the quality goals [26, 15, 16].

Ebert and Dumke [30] described quality management (QM) as not just a task, but a tradition that needs to be deep-seated within a company's culture, and involves all planned systematic events and processes for creating, monitoring and assuring quality. QM is aimed at monitoring and refining the development process on the premise that the quality of the development process directly impacts the quality of the final product [4].

The concept of software quality was first introduced formally at Bell Laboratories in the year 1916, and gradually pervaded the software industries later in the 1970s [31]. Murugesan [32] described quality as an essential requirement for software products and a necessity for business survival in the software industry. It has also been described as a complex concept that can sometimes be ambiguous and difficult to measure. Sommerville [33] explained that the concept of quality in software engineering does not imply the exact same thing as it does in other engineering fields (like manufacturing) where it is restrained in definition to meeting predefined specifications. In software engineering, quality should be tailored towards specific customer requirements and organizational standards [33]. In software engineering, quality has been described as meaning "meeting requirements" and "fitness for use"; implying that the software product meets the user requirements in the requirements specification, and performs the required tasks to the user's satisfaction.

SQM therefore entails defining quality goals for the software products, instituting strategies for accomplishing these goals, as well as observing and modifying the software plans, software work products, activities, and quality goals to fulfill the user's requirements and need for a product of high quality [15]. The requirements engineering process and the resultant documentation are very important to SQM since the quality

structure is built around it. With growing stress on the importance of product quality, process maturity, and continual process improvements, robust quality focus is beginning to emerge in all phases of the software development lifecycle [4]. Sommerville [34] enumerated SQM activities to include: Quality assurance, which establishes organisational procedures and standards for quality; Quality planning, which selects applicable procedures and standards for a particular project and performs necessary modifies them as required; and Quality control, which ensures that procedures and standards are followed by the software development team. Humphrey [35] defined an essential roadmap (of eight steps) for the consistent production of quality software to include: Establishing quality policies, goals, and plans; Proper training, coaching, and support for developers and their teams; Establishing and maintaining a requirements quality-management process; Establishing and maintaining statistical control of the software engineering process; Reviewing, inspecting, and evaluating all product artifacts; Evaluating all defects for correction and identifying, fixing, and preventing other similar problems; Establishing and maintaining a configuration management and change control system; and Continually improving the development process. Sommerville [34] equally identified reviews as the most widely used approach for assessing software quality, and advocated that quality management should be separated form project management so as to guarantee the independence of the quality review process.

A number of studies including those of Elgebeely [36] and Sowunmi *et al.* [4] have outline some of the challenges inhibiting the proper implementation of SQM practices in industries to include the following: tight schedules characterized by strict deadlines, QM budget overhead, QM bureaucratic overhead, QM time overhead, inadequate tools for process automation, a low level of awareness about the process, inadequate organizational SQM training, weak or inexistent organisational framework for SQM, difficulty in gaining senior management support, contradicting view-points among key stakeholders, developer ego, impatience of management, and past failed attempts at implementing SQM.

2.3. Interaction between the two QPM and SQM

Although the relationship between Quantitative Process Management (QPM) and Software Quality Management (SQM) has been conspicuous at several stages in the build to this point, the current section attempts to spell out this relationship in clear terms. The work of Paulk *et al.* [15, 16] will be very instrumental in achieving this.

The practices of the SQM KPA are based on the practices of the Integrated Software Management and Software Product Engineering KPAs [19], which are focused on establishing and implementing the project's defined software process, and the QPM KPA, which is focused on establishing a quantitative understanding of the ability of the project's defined software process to achieve the desired results [26, 15, 16]. To further elucidate on this relationship between the QPM and SQM KPAs, the following is an extraction from the work of Paulk *et al.* [15, 16]:

The key process areas at Level 4 focus on establishing a quantitative understanding of both the software process and the software work products being built. The two key process areas at this level, Quantitative Process Management and Software Quality Management, are highly interdependent, as is described below:

The purpose of Quantitative Process Management is to control the process performance of the software project quantitatively. …

The purpose of Software Quality Management is to develop a quantitative understanding of the quality of the project's software products and achieve specific quality goals. Software Quality Management applies a comprehensive measurement program to the software work products described in Software Product Engineering.

3. RESEARCH METHODOLOGY

The research methodologies employed in the current study are discussed in this section. Two major research approaches were adopted namely survey research and action research. The survey was based on the research objectives of evaluating the level of performance of the Quantitative Process Management and Software Quality Management KPA practices in the Nigerian software industry.

The survey covered a total of thirty Nigerian software companies. Twenty seven of the companies were based in Lagos state, South-Western Nigeria, while three of them were based in Asaba, in the South-South geo-political region of the country. Twenty six of the thirty selected companies (that is 86.67%) eventually partook in the study. The sampling method is stratified to the extent that according to Soriyan and Heeks [3] and Soriyan *et al.* [37], the majority of Nigerian software companies are located in Lagos, the commercial nerve centre of the country. An abridged version of the SEI Maturity Questionnaire [8] was used to gather information about the level of performance of the practices associated with the QPM and SQM KPAs within the companies studied. This instrument was administered to solutions developers and software project managers in the industry. This instrument served as the key data collection tool for the survey.

Certain of the selected companies were considered for more detailed investigation using the action research methodology. A direct observation of their activities and environment was performed alongside participation in the actual software development process and measurement of process-related phenomena. These activities were performed in the nominated companies for over a period of time so as to get firsthand information about the actual practices adopted in the companies. Information about the companies and their operations were gathered via print and electronic documentation. Both structured and unstructured interviews were also used to solicit for more information and further clarifications.

4. RESULTS

The results of the current study are as shown in Tables 1 and 2. The results are equally graphically represented as depicted by Figures 1 and 2. The results are presented in percentages of actual responses. The averages for each response option are shown in bold at the last row of each table. Discussions and resultant conclusions from these results are presented in the subsequent sections.

Key Practices of the Quantitative Process Management (QPM) KPA

Q1. Does the project follow a documented plan for conducting quantitative process management?

Q2. Is the performance of the project’s defined software process controlled quantitatively (e.g., through the use of quantitative analytic methods)?

Q3. Is the process capability of the organization's standard software process known in quantitative terms?

Q4. Does the project follow a written organizational policy for measuring and controlling the performance of the project's defined software process (e.g., projects plan for how to identify, analyze, and control special causes of variations)?

Q5. Are adequate resources provided for quantitative process management activities (e.g., funding, software support tools, and organizational measurement program)?

Q6. Are measurements used to determine the status of the quantitative process management activities (e.g., cost of quantitative process management activities and accomplishment of milestones for quantitative process management activities)?

Q7. Are the activities for quantitative process management reviewed with the project manager on both a periodic and event-driven basis?

Table 1: Performance of key practices of the QPM KPA

	Responses			
Q	Yes	No	NA	DK
Q1	0	21	2	3
Q2	2	22	1	1
Q3	4	20	2	0
Q4	0	18	5	3
Q5	6	12	6	2
Q6	3	17	0	6
Q7	0	23	1	2
%	8.24	73.08	9.34	9.34

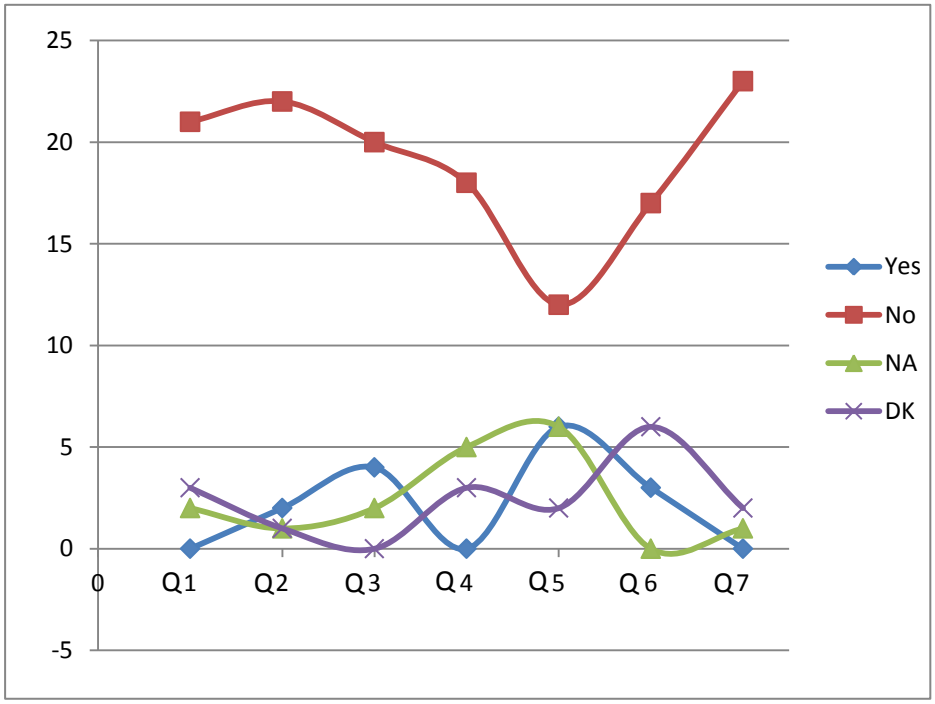

Figure 1: Chart of performance against key practices in QPM

Key Practices of the Software Quality Management (SQM) KPA

Q1. Are the activities for managing software quality planned for the project?

Q2. Does the project use measurable and prioritized goals for managing the quality of its software products (e.g., functionality, reliability, maintainability and usability)?

Q3. Are measurements of quality compared to goals for software product quality to determine if the quality goals are satisfied?

Q4. Does the project follow a written organizational policy for managing software quality?

Q5. Do members of the software engineering group and other software-related groups receive required training in software quality management (e.g., training in collecting measurement data and benefits of quantitatively managing product quality)?

Q6. Are measurements used to determine the status of the activities for managing software quality (e.g., the cost of poor quality)?

Q7. Are the activities performed for software quality management reviewed with senior management on a periodic basis?

Table 2: Performance of key practices of the SQM KPA

	Responses			
Q	Yes	No	NA	DK
Q1	6	11	4	5
Q2	8	10	4	4
Q3	12	10	2	2
Q4	0	22	3	1
Q5	5	18	2	1
Q6	7	9	2	8
Q7	6	12	3	5
%	24.18	50.55	10.99	14.29

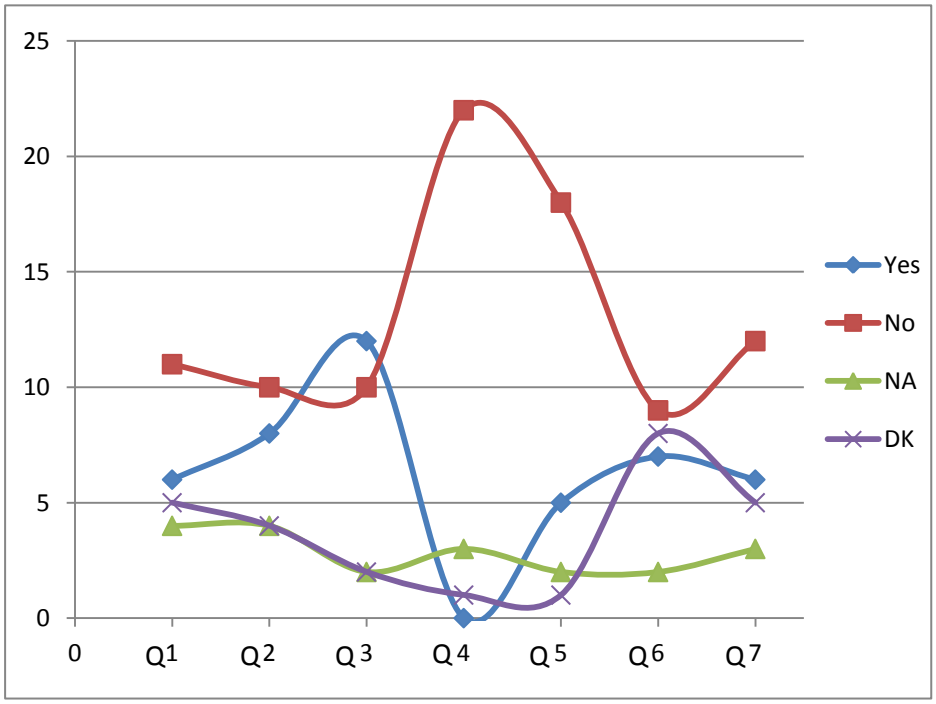

Figure 2: Chart of performance against key practices in SQM

5. DISCUSSION

The results depicted in Tables 1 and 2 show a high degree of non-performance of key practices in the associated with both Quantitative Process Management (QPM) and Software Quality Management (SQM) key process areas (KPAs). The already established strong interdependence between the two KPAs would easily account for the similarity in performance of practices associated with both KPAs, even though the performance of SQM is better than that of QPM. These results were consistent with the findings of the study of Sowunmi *et al.* [4] which although mainly focused on quality assurance equally observed that a number of other KPAs, in this case QPM and SQM, and their associated practices, were quite neglected in many software companies in developing countries. A number of studies have equally shown that quality management processes are generally not strictly adhered to by software organizations and that this non-adherence reduces the overall quality of the final software product which might actually account for low patronage in the industry. A view point about one of the major causes of the poor performance of these KPAs, which is equally supported by a number of other studies including Heeks [1, 2], Soriyan and Heeks [3], Sommerville [34], and Sowunmi *et al.* [4], is that many software practitioners are unaware of the laid down standards by international standards organizations and the required process improvement techniques. Due to this poor awareness, the self-alluded standards of their in-house methods, which are limited to their local experience and knowledge, are not aligned to software standards of accredited bodies, which are an encapsulation of best practice. Even where the awareness exists, Sowunmi *et al.* [4] and a number of other studies have shown that quality management processes are generally not strictly adhered to by software organizations.

Besides all the aforementioned, some organizations equally approach the concept of software quality from a simplistic view-point which results in a number of issues such as: Tension between customer quality requirements, such as efficiency and reliability, and developer quality requirements, such as maintainability and reusability; Difficulty in specifying some quality requirements in an unambiguous way; and the usual incompleteness and frequent inconsistency of software specifications [34]. It is equally observed that the association of the QPM and SQM KPAs with the software process maturity level 4 (Quantitatively Managed) could equally account for the poor implementation of the KPAs since the studies of Aregbesola and Akinkunmi [5, 6], Aregbesola *et al.* [7], Aregbesola and Onwudebelu [38], and Aregbesola and Oluwade [39] stated that the Nigerian software industry is currently at maturity level 1.

6. CONCLUSION

This paper has concentrated on the appraisal of the performance of two KPAs at the quantitatively managed SEI CMMI software process maturity level, namely, Quantitative Process Management (QPM) and Software Quality Management (SQM). By using survey and action research methodologies, it was illustrated that the performance of the two KPAs in the Nigerian software industry was quite weak, although better performance was recorded for the SQM key practices than for those of the QPM KPA. These KPAs should therefore be accorded the needed attention so as to strengthen them for optimal performance. Improving the performance of the practices associated with the QPM and SQM KPAs will go a long way in improving the software process maturity level of the Nigerian software industry.

To improve the performance of the practices at these KPAs, a number of challenges will have to be addressed. One of such is the institutionalisation of formal standard indicators for QPM and SQM with associated functional definitions, measurement methods, and analysis models, rather than a vaguely described definition in some form of ambiguous natural language with very little or no formal representations. Also, adequate resources in terms of time, budget, bureaucracy, tools, training, organisational framework, senior management support, common understanding and patience must be allocated to QPM and SQM practices so as to forestall the occurrence of any unplanned overheads.

Organizations should ensure that Operational definitions, Process consistency and aggregation, and Organizational implications are considered in the exploratory data analysis involved in initiating QPM. Organizations should equally see to it that they: institute quality policies, goals, and plans; implement proper training and support for the development team; create and maintain a requirements quality-management process; develop and maintain statistical control of the software engineering process; review, inspect, and appraise every piece of product; assess all identified flaws for rectification and preventing similar reoccurrence; create and sustain a configuration management and change control structure; and continually improve the software production process.

Quality reviews for assessing software quality should be performed as often as possible and should secure the full support of organisational top management. Quality management activities should also be separated from project management activities so as to guarantee the independence of the quality management reviews.

After all these measures have been put in place, organizational top management should then ensure that quantitative process management and software quality management practices are strictly enforced and adhered to across the length and breadth of the organization. The strict adherence to these quantitative quality management practices would improve the quality of the final deliverable software product as well as the overall ranking of the software-houses and subsequently boost patronage.

7. REFERENCES

[1] Heeks, R.B. (1999) Software strategies in developing countries, Communications of the ACM, 42(6), 15-20

[2] Heeks, R.B. (2002) i-Development not e-development, Journal of International Development, 14(1): 1-12.

[3] Soriyan H. A. and Heeks R. (2004). A Profile of Nigeria's Software Industry. Development Informatics Working Paper No 21, Institute for Development Policy and Management, University of Manchester.

[4] Sowunmi O. Y., Misra S., Fernandez-Sanz L. , Crawford B. and Soto R. (2016). An empirical evaluation of software quality assurance practices and challenges in a developing country: a comparison of Nigeria and Turkey. SpringerPlus20165:1921. DOI: 10.1186/s40064-016-3575-5.

[5] Aregbesola M. K. and Akinkunmi B. O. (2010a). Software Process Implementation – A focus on the Nigerian Software Industry. Journal of Research in Physical Sciences, Vol. 6, No. 2, pp. 9 – 14.

[6] Aregbesola M. K. and Akinkunmi B. O. (2010b). Software Process Implementation – A focus on the Nigerian Software Industry. International Research and Development Institute (IRDI), World Congress on Research and Development, Conference Center, University of Ibadan, 5th - 8th October. Vol. 5, No. 6, pg.111-116.

[7] Aregbesola M. K., Akinkunmi B. O., and Akinola O. S. (2011). Process Maturity Assessment of the Nigerian Software Industry. International Journal of Advances in Engineering and Technology (IJAET), Vol.1, Issue 4, pp. 10-25.

[8] Zubrow D., William H., Jane S. and Dennis G. (1994). Maturity Questionnaire. Special Report CMU/SEI-94-SR-7, June 1994.

[9] Glover M. T. and Dennie D. (2017). CMMI–Agile Process Combo: How to be Agile with CMMI. Excellence in Measurement Technology.

[10] O'Neill D. (2017). In Search of a Modern Software Life Cycle: Secure DevOps Foundations for Large-Scale Software Systems. CrossTalk March/April 2017: Modern Process Trends.

[11] Jalote P. (2002). Managing Software Projects. Addison-Wesley.

[12] CMMI Product Team (2006). CMMI for Development, Version 1.2 - CMMI-DEV, V1.2. Software Engineering Institute, Carnegie Mellon University.

[13] Hurst J. (2017). The Capability Maturity Model and Its Applications. SANS Software Security with Frank Kim.

[14] Mogre A. and Salunkhe S. (2014). Effective CMMi Implementation in Agile environment with fresh team. Atos, Mumbai, India.

[15] Paulk M. C., Weber C. V., Garcia S. M., Chrissis M. B., and Bush M. (1993). Key Practices of the Capability Maturity Model, Version 1.1. Technical Report CMU/SEI-93-TR-025 ESC-TR-93-178, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania.

[16] Paulk M. C., Weber C. V., Curtis B., & Chrissis M. B. (1995). The Capability Maturity Model: Guidelines for Improving the Software Process. Addison – Wesley, Boston.

[17] Aregbesola M. K. (2017a). Experiential Appraisal of Organizational Process Focus and Process Definition in Nigerian Software Companies. Journal of Scientific and Engineering Research. In press.

[18] Aregbesola M. K. (2017b). Investigating Training Program and Intergroup Coordination in relation to Peer Review in Nigerian Software Companies. Journal of Scientific and Engineering Research. In press.

[19] Aregbesola M. K. (2017c). Integrated Software Management and Product Engineering. Manuscript submitted for publication.

[20] Campo M. and Smith K. (2012). NDIA CMMI Technology Conference & Users Group. November 5-8, Denver, Colorado.

[21] INCOSE (2010). A Basic Introduction to Measurement Concepts and Use for Systems Engineering. International Council on Systems Engineering (INCOSE) Systems Engineering Measurement Primer v2.0. 5th November.

[22] Madachy R. (2000). Process Management and Software Quality Management. USC Computer Science CS577b.

[23] Niessink F. and Vliet H. (1999). The Vrije Universiteit IT Service Capability Maturity Model. Faculty of Sciences, Division of Mathematics and Computer Science Vrije Universiteit Amsterdam.

[24] Kinnula A. (2001). Software Process Engineering Systems: Models and Industry Cases. Department of Information Processing Science, University of Oulu.

[25] Sargut K. U. (2003). Application of Statistical Process Control to Software Development Processes via Control Charts. A Master of Science Thesis submitted to the graduate school of Informatics of the Moddle East Technical University.

[26] Keraminiyage K., Amaratunga D. and Haigh R. (2005). Achieving higher capability maturity in construction process improvement. In: 2nd International Conference for Postgraduate Researchers of the Built And Natural Environment (PRoBE), 16th - 17th November, Glasgow, Scotland. (Unpublished).

[27] Paulk M. C. (1999). Toward Quantitative Process Management With Exploratory Data Analysis. International Conference on Software Quality, Cambridge, MA. pp. 1-7.

[28] Hikichi K., Yonemitsu T., Fukuchi Y., Fushida K. and Iida H.(2005). An assistance method of incorporating quantitative management indicator into software development process. Hitachi, Ltd., Shinagawa, Tokyo, Japan.

[29] Popa M. (2011). Techniques and Methods to Improve the Audit Process of the Distributed Informatics Systems Based on Metric System. Informatica Economică. vol. 15, no. 2, pp 69 -78.

[30] Ebert C., Dumke R. (2011). Software Measurement. Springer, New York.

[31] Lewis W. E. (2004). Software testing and continuous quality improvement. Auerbach Publications, Boca Raton.

[32] Murugesan S. (1994). Attitude towards testing: a key contributor to software quality. In: Proceeding of 1st international conference on software testing, reliability and quality assurance. IEEE, New Delhi, pp 111–115.

[33] Sommerville I. (2007). Software engineering. Addison-Wesley, England.

[34] Sommerville I. (2006). Quality Management. Software Engineering, 8th edition. Chapter 27 Slides. Addison-Wesley, England.

[35] Humphrey W. S. (2008). The Software Quality Challenge. CROSSTALK: The Journal of Defense Software Engineering. pp 4 - 9.

[36] Elgebeely A. R. (2013). Software quality challenges and practice recommendations. In: IBM. http://www.ibm.com/developerworks/rational/library/software-quality-challenges-practice-recommendations/

[37] Soriyan H. A., Mursu A. and Korpela M. (2000). Information system development methodologies: gender issues in a developing economy. In: Women, Work and Computerization, E. Balka & R. Smith (eds.), Kluwer Academic, Boston, MA, 146-154.

[38] Aregbesola M. K. and Onwudebelu U. (2011). Typical Software Quality Assurance and Quality Management Issues in the Nigerian Software Industry. National Association for Science, Humanities & Education Research, 8th National Conference, University of Ado Ekiti, Ado Ekiti, September 14-17.

[39] Aregbesola M. K. and Oluwade B. A. (2014). An Experimental Evaluation of Defect Prevention and Change Management in Software Process Optimization in the Nigerian Software Industry. ARPN Journal of Systems and Software Vol.4, No.1, pp. 5-11.