JCGIRM Journal of Corporate Governance, Insurance, and Risk Management (JCGIRM)
2014, Volume 1, Series 2

Pages 66-76

Improving Unified Process Methodology by Implementing New Quality Management Discipline

Dr Boris Todorović, PhD*

Director, Axelyos Solutions, Bosnia & Herzegovina

ABSTRACT

Unified process, a leading software development methodology, allows project teams to incrementally build their software and structurally defines project roles, phases, iterations and disciplines. One of the issues that arise when applying unified process is absence of discipline for quality assurance and control. This research aims to define new discipline entitled "quality management of software development" and its processes, in order to produce modified version of unified process, suitable for continually controlling quality in software development projects. This discipline will integrate ISO 9126 "Software engineering – product quality", which is an international standard for addressing software quality and quality control tools, as proposed by Project Management Book of Knowledge 2010. Main hypothesis of this research is that, by defining and integrating new discipline of quality management, project teams that employ this new, modified version of unified process, will be able to produce software of higher quality level. Experimental research is conducted on four software development projects, ranging from 2009 to 2010, two of which use standard, and two of which use modified unified process model. Research results show higher software quality levels in two projects that use modified unified process methodology.

ARTICLE INFO

Keywords:
Unified process, quality management, software quality assurance, quality control, ISO 9126, Project Management Body of Knowledge

**Corresponding author:*
boris@axelyos.com
(Dr Boris Todorović)

Article Submitted 12-01-2014
Article Accepted 24-02-2014

**Article previously published in EJEM 2014, vol 1, No. 2

Introduction

Unified process is a leading software development methodology, used by hundreds of thousands of software development teams across the world. It is an iterative methodology, which allows project teams to incrementally build their software. Unified process defines project roles (human resource organization), phases and iterations (used for scheduling) and disciplines, which are used to organize type of work undertaken in a project. Among others, disciplines include requirement specifications, analysis and design and project management. One of the issues that arise when applying unified process is absence of discipline for quality assurance and control. This paper aims to address this problem by further researching on capabilities to improve standard Unified Process model with existing standards for quality control and assurance.

Literature overview

Overview of unified process

Unified Process is an iterative software development methodology which came formally into existence in 1999, when Ivar Jacobson, Grady Booch and James Rumbaugh published their book "*The Unified*

Software Development Process" (Jacobson, Booch, & Rumbaugh, 1999). This methodology became very popular worldwide, especially after IBM adopted its own variant entitled Rational Unified Process. As an iterative methodology, it is frequently used as it overcomes major problems of sequential software development approach such as: managing of rapidly changing requirements, risk management (strategy, tools & techniques), poor project organization, formal documentation, inability to adopt to changing business environments, poor testing performance and record, etc., as identified by Xiong (2011).

Unified process is organized into a *static* and *dynamic structure*. Static structure deals with organizing project work into disciplines (such as analysis and design, or testing), which are superset of work type to be done (Kroll & Kruchten, 2003). Each of them defines a set of activities, or tasks to be completed and artifacts, documentation and code to be produced. Finally static structure covers project roles, which serve to group project members into teams, based on their specialty (e.g. analyst, project manager, developer). Dynamic structure deals with time - project phases, iterations and milestones.

Focusing on quality, it is obvious that Unified Process defines a discipline called "testing". This discipline describes how to do testing, or quality control. As all iterative and agile methodologies do, Unified Process uses concept of continuous testing (quality control). Since most common problem of sequential software development approach is that quality control is done only at the end of the project, Unified Process focuses on doing quality control continually throughout the project. Main driver for this concept is fact that if a bug or defect is identified in early phase of the project, it will be cheaper and faster to fix it, than in later phases. However, we must note that Unified Process is not a standard and, as such, does not define a set of standardized tools to do quality control with. Also, it primarily deals with quality control and not quality management and improvement, not giving project managers a tools and standard to deal with quality (Hindle, Godfrey, & Holt, 2010). On the other hand, Unified Process never discusses what are characteristics of quality software, how to structure or to test these non-functional requirements (Losavio, Chririnos, Matteo, Levy and Ramande-Cherif, 2004).

Project Management Body of Knowledge and quality tools

In order to improve Unified Process, as out base software development methodology, from a quality management perspective, we introduce *Project Management Body of Knowledge (PMBOK)*. This is the top most standard for project managers since late 1980s. Created by The Project Management Institute (PMI), it defines nine knowledge areas, among them, quality management knowledge area. Each knowledge area defines processes and tools which are needed for project managers to perform well. As a basic concept, PMBOK introduces six project management constraints – time, resources, costs, quality, scope and risks. By balancing these constraints, project managers can drive themselves through projects.

Quality management knowledge area defines three processes: *quality planning*, *quality control* and *quality assurance*. They represent logical steps towards achieving high levels of quality within projects. Three main goals that PMBOK defines for all projects are: strive to satisfy client's requirements, to produce error-free products (software) and continuously improve quality. PMBOK recommends plan-do-check-act, Six Sigma and Total Quality Management (TQM) as basis for continuous quality improvement (Phillips, 2007).

Tague (2004, p. 15) reflect on PMBOK recommendations with inclusion of seven basic quality control tools, as defined by Ishikawa (1990): cause and effect (fishbone) diagram, control charts, flowchart diagrams, histograms, Pareto diagrams, run charts and scatter charts. These tools help by presenting data visually and organizing them using proved methods, in order to help project and quality managers to

assess control measurements, to project trends, model root causes of the problems, see correlation between two variables, etc.

ISO 9126 Software engineering – product quality

Although PMBOK recommends quality tools, it also fails to define a standard way to measure software quality level. Also, the problem is how to look at the software quality as it's not a material good or product. To standardize this International Standards Organization (ISO) defined ISO 9126 standard, entitled "*software engineering – product quality*". This standard covers software quality characteristics and quality evaluation process.

ISO/IEC (2002) standard 9126 defines six general characteristics of quality software, as follows:

- *Functionality* – are the required functions available in the software?
- *Reliability* – how reliable is the software?
- *Usability* – is the software easy to use?
- *Efficiency* – how efficient is the software?
- *Maintainability* – how easy is to modify the software?
- *Portability* – how easy is to transfer to another environment?

These six characteristics are divided into sub-characteristics, which further refine questions asked to evaluate software quality level in certain area.

Software quality evaluation is split into: *preparation* and *evaluation* (as illustrated in the figure 1). Preparing software quality evaluation encompasses four sub-phases, where quality requirements are defined for the project; then metrics for measuring certain quality properties are chosen; levels for ranking are defined (range of values which are treated as satisfactory or unsatisfactory); and evaluation criteria (summarizing results) are defined. Evaluation process encompasses three sub-phases: measurement, in which metrics are applied to software and give out certain quantitative results; ranking, in which it is evaluated if they are satisfactory or unsatisfactory; and assessment.

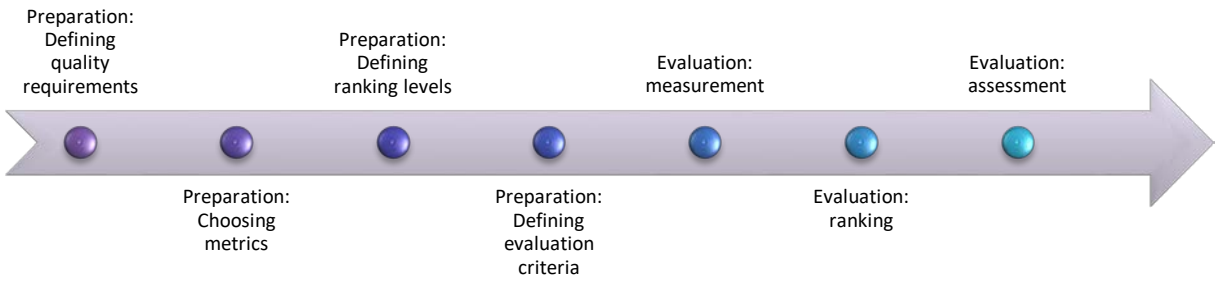

Figure 1 - Software quality evaluation process in ISO 9126 standard, ISO/IEC (2002)

Research methodology

ISO 9126 standard is present for over two decades in theory and used by large number of companies worldwide. There are number of software development methodologies used, so there are relatively few known authors that tried to improve Unified Process with quality management aspect in mind. In one of the most relevant articles, Losavio et al. (2004) presented an overview of possible integration of Unified Process methodology and ISO 9126 standard, but only in a perspective of building system architecture. Also, authors did not provide model or guidelines on how to practically integrate ISO 9126 into Unified

Process. This research aims to provide exactly that - a structured model for improving Unified Process methodology by integrating ISO 9126 standard into a new discipline.

Unified Process is adaptable methodology and there are many variants of it all over the world. IBM created Rational Unified Process, while other organizations developed Open Unified Process, Enterprise Unified Process and even a mix called Agile Unified Process (integrates agile with iterative software development approach). Since Unified Process is extendable and adaptable, we will be implementing a completely new discipline in Unified Process. This new discipline, called *quality management*, will cover all tasks, activities and artifacts used for purpose of managing software quality. New project role is also defined – a project *quality manager*, who will oversee entire quality management discipline and processes, reporting directly to project manager.

In order to adequately research proposed modified Unified Process methodology, this paper presents experimental research. Experimental research was conducted at Republic of Srpska Securities Commission[1] (RSSEC in further text), which is the capital market regulatory body in Republic of Srpska, Bosnia and Herzegovina. Author of this paper was employed by RSSEC as information technology project manager, leading development of multiple software development project.

Experimental research will compare results from four software development project at RSSEC, ranging from 2009 to 2010. Two project were using standard Unified Process model and other two were using new, modified Unified Process model with new quality management discipline, throughout project lifecycle. Since only two project employing modified model have been continually tracked and evaluated for software quality (actually have software quality level data measured and evaluated), we will evaluate two previous project that used standard model using same metrics to put in perspective level of quality between those project.

Main hypothesis of this research is that, by defining and integrating new discipline of quality management, project teams that employ modified version of Unified Process, will be able to produce software of higher quality level.

QUALITY EVALUATION PROCESS

Defining requirements and metrics

First step in quality evaluation process is definition of non-functional requirements and software quality requirements. This step is completed by extending using use case specifications with minimum acceptable level of quality. After project goes through phase of user acceptance testing, these quality will be used to evaluate is software meets acceptable minimum values.

As a next step, quality manager proposes a set of metrics from ISO 9126 standard. This standard defines three sets of metrics: internal, external and in-use metrics. During software development phase, internal set of metrics are used (ISO/IEC, 2002). For each of six general software quality characteristics, there are many sub-characteristics and many metrics. Main goal of this step is to identify suitable metrics to evaluate software. Quality manager produces list of metrics for reference, with their original.

[1] More details regarding Republic of Srpska Securities Commission are available on official Web site: www.secrs.gov.ba

Defining rating levels and evaluation criteria

Defining rating levels is a step during which acceptable range of results from evaluations is defined. Rating levels depend on metric and the result it produces. For example, metric for evaluating suitability, entitled "functional adequacy" is calculated by following equation (ISO/IEC, 2002):

$$X = 1 - {}^{A}/_{B}$$

Equation 1 - Example of equation used to calculate result for a metric

In this equation, A represents number of functions where problems were detected, while B represents total number of tested functions. Results of this equation are in range of 0 and 1, or represented mathematically: $\mathbf{0} \leq \boldsymbol{X} \leq \mathbf{1}$. When setting rating levels, we are defining minimum acceptable value, below which we treat result as unacceptable. In this example, acceptable result will be only if $\boldsymbol{X} \geq \mathbf{0,75}$.

Defining evaluation criteria follows rating level definition and describes quantitatively how results from different metrics will be summarized. Recommended approach for summarizing results is using weighted scoring model (Saphire, 2008). Using this approach, each metric will be assigned weight. This process produces evaluation planning sheet, as in example below:

Characteristic	Sub-characteristic	Metric type	Metric name	Ranking level	Evaluation criteria (1-10)
Functionality	**Suitability**	**External**	Functional adequacy	$X \geq 0{,}75$	R=5
			Functional implementation completeness	$X \geq 0{,}90$	R=10
		Internal	Functional adequacy	$X \geq 0{,}75$	R=5
			Functional implementation correctness	$X \geq 0{,}75$	R=10
			Functional specification stability	$X \geq 0{,}25$	R=2

Table 1 - Example of evaluation planning sheet

Measurement, ranking and assessment

Following step in process is organizing project team, with quality manager and project manager defining new work assignments for project team member. Essentially, team members will be given role of quality controllers. Modifying project plan, work break down structure and other planning elements is done in this process, allowing time and resources to do measurements. Quality controllers will be required to take measurements and quantitatively calculate and note results of specific measurement. Measurements are done by using existing Unified Process structure: at the end of each phase and at the end of each iteration.

Important aspect of model is best practice and recommended measurement form. Following table gives template used to record measurement results for one use case specification:

Code/title of use case	K.01.001 – Adding new user to database
Characteristics	Functionality
Sub-characteristic	Suitability
Metric type	Internal
Metric	Functional implementation correctness
Measurement range	5 measurements 01/06/2010 – 17/06/2010

#	Date and time	Author	Phase	Iteration	Result
1.	01/06/2010 16:50	John Smith	Elaboration	E1	0.35652
2.	02/06/2010 12:00	Jane Doe	Elaboration	E2	0.75485
3.	02/06/2010 14:45	John Smith	Elaboration	E3	0.75000
4.	15/06/2010 12:36	Mark Jones	Elaboration	E4	0.80545

5.	17/06/2010 09:00	Jane Doe	Construction	C1	0.85000
6.	...	...	...	...	...

Table 2 - Template for recording measurement results for single use case

Based on measurements through project lifecycle, quality manager can summarize measurements by use cases, characteristics, sub-characteristics and other information, creating various graphical and pivot/cross table representations of quality levels and trends. Besides pivot charting, control charts (recommended by PMBOK) can be used to identify variations in software quality levels. Also, they define lower and upper limits, which clearly identifies if process is out of control. This is essential for project managers to timely react to problems.

Each measurements add news results, which need to be ranked – put in perspective with minimum acceptable quality levels. Ranking is done on metric, iteration and phase levels. Ranking document adds simple pass or fail grades to results.

Analysis of research results

Presentation of results

For purpose of presenting research results, we will summarize ranking data in effective manner using *radar chart*. By employing this chart, percentage of compliant use cases is clearly visible and gives sense about quality level of whole project. Summarizing data by use case on sub-characteristic level is done using following formula:

$$UC_{\ll Sub-characteristic \gg} = \frac{\sum S_{M1} * R_{M1} + S_{M2} * R_{M2} + \cdots + S_{Mn} * R_{Mn}}{\sum R_{M1} + R_{M2} + \cdots + R_{Mn}}$$

Equation 2 - Formula used to summarize evaluation results on sub-characteristics level

Where S_{Mn} represents either acceptable (value 1) or unacceptable result for metric (value 0), and R_{Mn} represents evaluation criteria for metric. We are using previously defined evaluation criteria to summarize results by employing weighted scoring model.

Standard Unified Process Model

Two projects that employed standard Unified Process model have been evaluated after their development has been completed. As proposed by ISO 9126 (2002) external metric are used to evaluate software quality during testing stages of software development lifecycle or during operation phase. As seen in figure 2, two projects do not reach top values in sub-characteristics, except for co-existence, installability and adaptability.

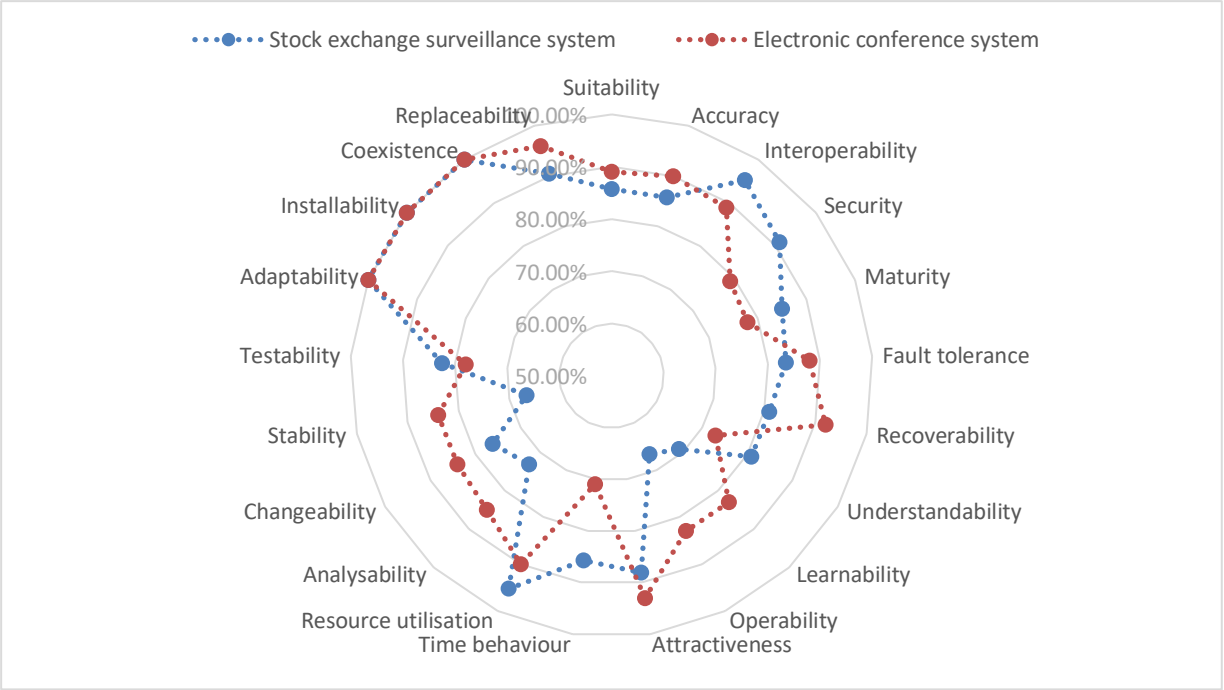

Figure 2 - Software quality level in two projects that used standard Unified Process model

Figure 3 represents data obtained by looking at each project at metric level, and then calculating standard deviation and summarizing it at sub-characteristic level. This chart shows that there is deviation present in quality levels.

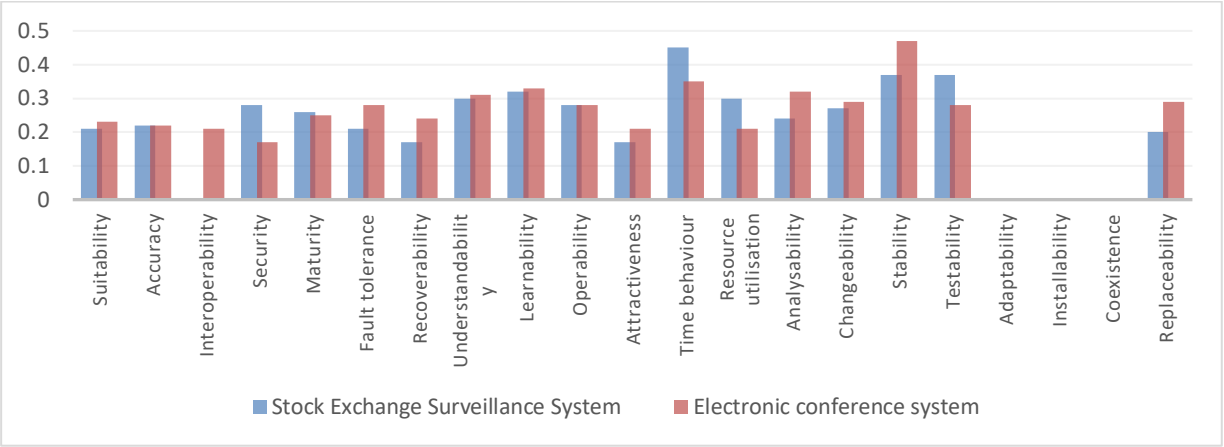

Figure 3 - Standard deviation at metric level, summarized at quality sub-characteristic level in two projects that used standard Unified Process model

Modified unified process model

Two projects that employed modified Unified Process model have been evaluated during their development, starting from first phase of process (inception). As proposed by ISO 9126 (2002) internal metric are used to evaluate software quality during development stages, by testing non-executable code (source code). As seen in figure 4, two projects reach top values (greater than 95%) in nearly all sub-characteristics.

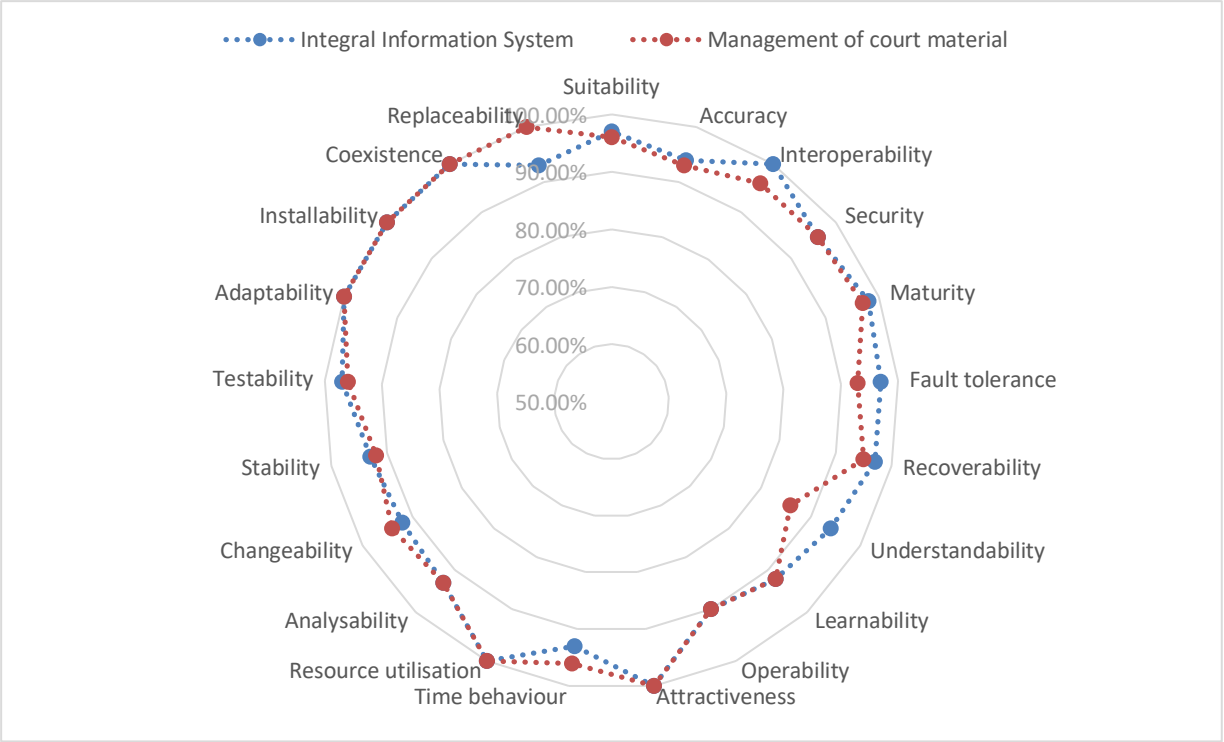

Figure 4 - Software quality level in two projects that used modified Unified Process model

As presented in figure 5 standard deviation is lower than in projects that were using standard Unified Process mode, although some variation is still present.

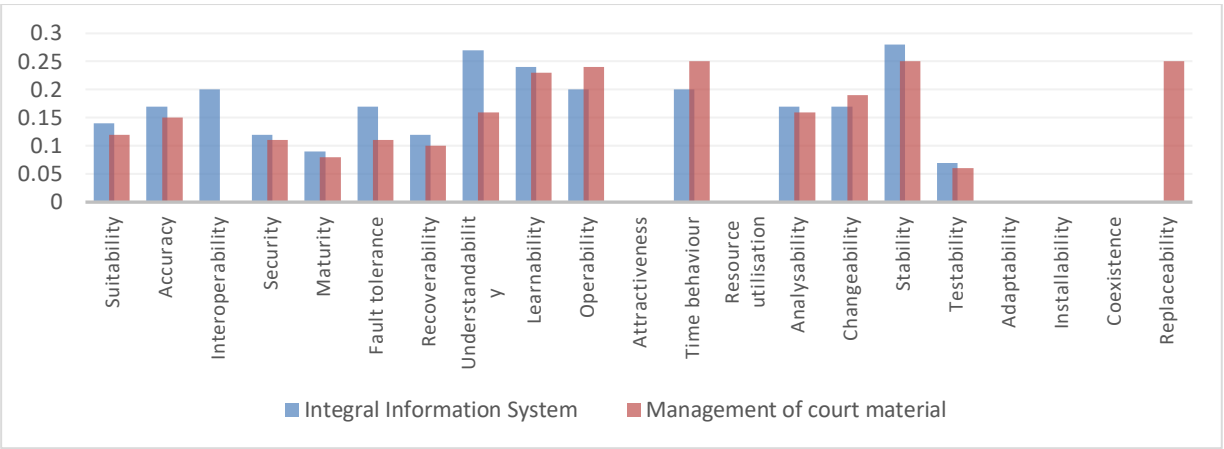

Figure 5 - Standard deviation at metric level, summarized at quality sub-characteristic level in two projects that used modified Unified Process model

DISCUSSION

In order to prove our main hypothesis, we have to prove that modified Unified Process model improved software quality levels. In research results we have provided software quality levels for both models. Figure 6 visualizes software quality levels in a single chart, grouped by software quality characteristics and summarized by model (including all projects which model was applied to).

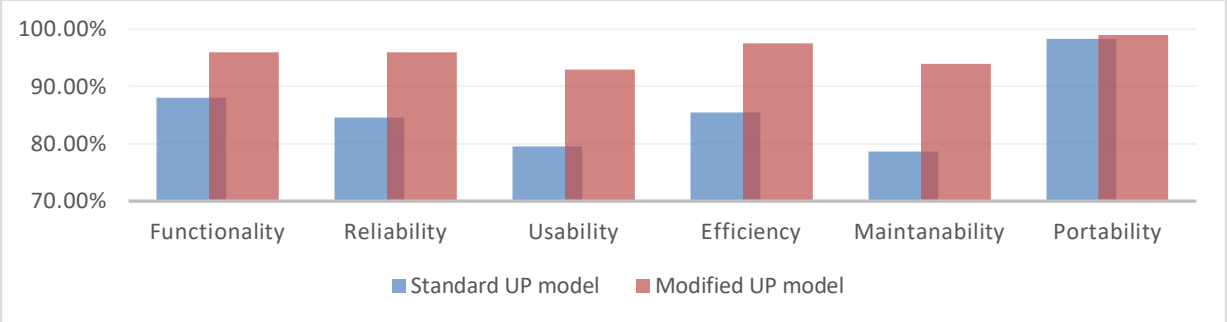

Figure 6 - Comparing software quality level of projects developed using standard and modified Unified Process model, grouped by quality characteristics

As we can see, modified Unified Process model, which integrates Unified Process, PMBOK and ISO 9126 indeed *produces higher software quality level*. Software quality improvement ranges from 0.69% to 15.32%, averaging at 10%. Looking at data in figures 4 and 5, we can conclude that applying modified model contributed to normalizing quality levels, reducing standard deviation and achieving high quality levels in nearly all characteristics.

Radice (2000) indicates in his research that applying tools for statistical process control (Ishikawa tools, recommended by PMBOK), is critical to achieve process control, which is one of the factors for achieving better final software quality levels in modified model. Apart from that, it is important that modified model used these tools continually and evaluated quality in each of the Unified Process phases, as recommended by Gibbs (2007).

If we make reference to Capability Maturity Model, it defines five maturity levels and criteria by which a process or model can be ranked (Miyachi, 2001). Standard Unified Process model is at level 3 (defined), while modified model conforms to all requirements of level 4 (managed) and two requirements of level 5 (optimized), because all of the key processes are implemented: quantitative process management, quality management, defect prevention and management of technology change.

Conclusion

This paper presented structured and systematic process by which new modified Unified Process methodology was built. Integrating Unified Process with quality tools recommended by PMBOK and ISO 9126 standard, we were able to produce a model which was set to improve software quality. Using experimental research, we confirmed that modified model was more successful and helped project team achieve high software quality levels, when compared to standard model. Identified improvement on model basis was more than 10%, indicating there was not a case of statistical errors, but a case of major quality improvement.

This research did not explicitly change existing ISO 9126 models, however, it identified several possible research directions in order to further improve modified Unified Process model. First, ISO 9126 uses both subjective and objective metric gathering tools, with subjective tools being user/consumer

interviews. These tools can generate subjective evaluations of software and further research can potentially identify any tools that can generate more objective data.

Future research will also take existing automated quality control tools and try to create infrastructure for optimizing project workloads of quality controllers and managers by automating or partially automating evaluation process. Modern integrated development environments (IDEs) such as Microsoft Visual Studio and Eclipse allow test automations, but within functional requirements.

REFERENCE:

CHARBONNEAU, S. (2004): Software Project Management - A Mapping between RUP and the PMBOK. Xeleration Software Corporation.

GIBBS, R. D. (2007): Project Management with the IBM® Rational Unified Process®: Lessons from the Trenches. Upper Saddle River, Prentice Hall Professional: (pp. 177).

HINDLE, A., GODFREY, M. W. and HOLT, R. C. (2010): Software Process Recovery using Recovered Unified Process Views. IEEE International Conference on Software Maintenance (ICSM), IEEE: (pp. 2)

ISHIKAWA, K. (1990): Introduction to Quality Control. Tokyo, 3A Corp: (pp. 98).

ISO/IEC. (2002): ISO 9126-3: Software engineering – Product quality – Part 2 (Internal metrics, ISO/IEC: (pp. 3, 12, 14).

JACOBSON, I., BOOCH, G. and RUMBAUGH, J. (1999): The Unified Software Development Process. Reading, MA, Addison Wesley: (pp. 10).

KROLL, P. and KRUCHTEN, P. (2003): Rational Unified Process Made Easy: A Practitioner's Guide to the RUP, The. Boston, MA, Addison Wesley.

LOSAVIO, F., CHRIRINOS, L., MATTEO, A., LEVY, N. and RAMANDE-CHERIF, A. (2004): Designing Quality Architecture: Incorporating ISO Standards Into the Unified Process. Winter: (pp. 27).

MIYACHI, C. (2001): Enhancing the Software Improvement Processes through Object-Process Methodology. Massachusetts Institute of Technology: (pp. 19).

O'REGAN, G. (2002): A practical approach to software quality. Springer.

PHILLIPS, J. (2007): CAPM/PMP Project Management All-in-One Exam Guide. McGraw Hill Professional: (pp. 279).

RADICE, R. (2000): Statistical Process Control in Level 4 and 5 Organizations Worldwide. Proceedings of the 12th Annual Software Technology Conference. Salt Lake City, Utah, USA, Software Technology Conference.

SAPHIRE. (2008): Public Deliverables - Delivery of the Assessment Criteria for Software Validation. Accessed on 2010-09-21, Saphire - Intelligent Healthcare Monitoring, http://www.srdc.metu.edu.tr/webpage/projects/saphire/: (pp. 13).

TAGUE, N. R. (2004): Seven Basic Quality Tools. Milwaukee, Wisconsin, ASQ Quality Press: (pp. 15).

WESTFALL, L. (2010): The Certified Software Quality Engineer Handbook. Milwaukee, ASQ Quality Press.

XIONG, J. (2011): New Software Engineering Paradigm Based on Complexity Science: An Introduction to NSE. Springer: (pp. 34).