Journal of Computer Science 1 (3): 316-322, 2005
ISSN 1549-3636

SQIMSO: Quality Improvement for Small Software Organizations

[1, 2]Rabih Zeineddine and [2]Nashat Mansour
[1]Central Bank of Lebanon, Department of Information Technology
P.O. Box 11-5544, Beirut, Lebanon
[2]Division of Computer Science and Mathematics, Lebanese American University
P.O. Box 13-5053 Beirut, Lebanon

Abstract: Software quality improvement process remains incomplete if it is not initiated and conducted through a wide improvement program that considers process quality improvement, product quality improvement and evolution of human resources. But, small software organizations are not capable of bearing the cost of establishing software process improvement programs. In this work, we propose a new software quality improvement model for small organizations, SQIMSO, based on three major issues. The first issue is that every improvement program should be wide enough to include the three main trends. The second issue is that any process quality model should answer the question "How to do" things. The third issue is that any suggested quality model should be cost-effective and practical enough to be implemented by small software organizations. SQIMSO also draws upon international quality standards, models, experiences and on a local field survey.

Key words: Software Engineering, Software Quality, Process Quality Model, Quality for Small Software Organizations

INTRODUCTION

Quality and Process Models: Quality models and standards have been developed to improve product quality, satisfy business goals and standardize software engineering practices and procedures. Their certifications have been sought due to management improvement strategies, governments' policies, market competition and customer's requirements. However, the implementation of quality models and standards involves time and cost overheads that are large, especially for small organizations.

The evolution of international quality models was intended to improve quality and to provide more guidelines for reducing overhead involved. ISO 9001 [1] required establishing, documenting and maintaining a software process. The Capability Maturity Model (CMM) was intended to improve software processes using a set of recommended practices in a number of key process areas [2, 3]. ISO 9000-3 preceded CMM guidance by introducing "Guidelines for the application of ISO 9001 to the development, supply and maintenance of software" [4]. LOGOS tailored CMM Model to suit small organization by addressing documentation overload, layered management, Limited resources and other factors [5]. BOOTSTRAP is a software process assessment and improvement that is based on benchmarking and determination of current achievement levels to assess suitable improvement programs and action planning [6]. The main objectives of the SPICE project were to put the first draft for software process assessment and to conduct early trials [7]. Based on SPICE, a new framework called SPIRE was created [8]. Trying to adapt the SPICE model to small size organizations, SPIRE adapted the language but used the same structure. Extreme Programming (XP) is actually a deliberate and disciplined approach that suits risky projects with dynamic requirements [9]. XP is based on simplicity, continual communication with customer and rapid feedback mechanisms and it only suits "small to medium sized teams building software with vague or rapidly changing requirements" [10].

International and Local Surveys: There have been a number of international surveys concerning international standards and Capability Maturity Models and their strengths and weaknesses. A survey was done in the United Kingdom by Hall and Wilson [11] about quality system formality, ISO Standards and quality certification for large-to small-scale software companies. The negative comments on quality certification were: it was irrelevant to quality, less important than having quality infrastructure and culture, just about producing software consistently and was not necessarily for high quality and not useful without practitioner professionalism and ability. Another survey in Scotland [12] led to similar results; an analyst programmer put it as follows "We have to fill a lot of forms, but it hasn't affected the way I write software, not at all".

A survey was done in Germany by Stelzer [13] on 21 software houses that were ISO 9001 certified. The objectives were to assess the value of an ISO 9001

certificate for buyers of software and software-related services. The major findings of the survey were as follows: some interpreted standards clauses strictly others very widely; deviations from the standards were not pointed out; a certificate did not guarantee that the quality system fulfilled all ISO 9001 requirements; an ISO 9001 certificate was not an indicator of the quality of the products, the processes, or the quality system; ISO was a way for certification bodies to hold small companies to ransom.

Another survey was done in Europe by Stelzer [14] based on interviews and reports from 36 European software houses. The major findings were as follows: not the technical contents of ISO 9000 made process improvements; the culture created by wide improvement program seemed to be more important; implementing an ISO 9000 quality system was a helpful impetus for a cultural change.

The lessons of the Australian experience surveyed by Wilson [15] can be summarized as follows: lack of understanding of ISO 9000; rigorous and documented processes and procedures do not guarantee quality products; three aspects are important to quality: people, process and product; much of the criticism directed to ISO 9000 was the result of human error resulted from either misinterpretation or misguided management.

We have also carried out a survey in Lebanon to figure out the existing software engineering practices concerning customer-supplier relationship, requirements acquisition processes, development processes, faced problems, standards and models used and organization's willingness to invest in and initiate quality improvement program. The survey included 32 software companies. The findings of this survey are: small software houses face many problems that compel them to think first how to survive before thinking how to start quality improvement programs; the major factor is economic; quality systems need good level of qualification, experience, training and probably additional staff, which implies additional cost.

Objectives: The experiences presented by the local and international surveys lead to the following remarks: (a) product quality or software process quality improvement needs a wide improvement program including building quality culture and quality system infrastructure that will facilitate the implementation process. (b) Nothing is achieved without practitioner professionalism and ability; i.e., provide "know how" and the applicable methods. (c) ISO 9001 certificate is not an indicator of the quality of the products, the processes, or the quality system. (d) Practitioners have to fill a lot of forms, but it hasn't affected the way they write software; thus, procedures and methods should be carefully selected.

Small software organizations face serious problems when dealing with the challenges of existing models because they cannot bear the costs. To give small software organizations the chance to have efficient and applicable quality improvement systems a new model should be developed. This model should avoid all the drawbacks and deficiencies of the existing international standards and should offer quality culture by adding a new factor that helps practitioners implement "what to do" by knowing "how to do". In this work, we propose SQIMSO as a model that suits small software organizations.

SOFTWARE QUALITY MODEL FOR SMALL ORGANIZATIONS (SQIMSO)

Motivation: In SQIMSO, three main critical issues are considered. The first critical issue is that every improvement program should be wide enough to include three main factors which are process quality, product quality and human resources management. The second issue is that any process quality model should answer the question *"How to do?"*. The third issue is that any suggested quality Model should be practical enough to be implemented by small software organizations for saving cost and time without decreasing the quality turnover level.

Based on international quality standards, models, surveys and on the local field survey, the new software quality improvement model SQIMSO is introduced. SQIMSO processes model structure is based on SPICE processes model structure but adapted to fit the practices of small organizations. Using a new framework, the processes entirely reestablished based on ISO 9001, ISO 9000-3, CMM, SPICE, SPIRE, software engineering practices, fields studies and surveys findings and our software engineering field experience. Conceptually, SQIMSO discusses quality improvement strategy from three dimensions that are process, capability and method. A new factor, methods, is added to facilitate initiating and implementing software process improvements.

SQIMSO is designed for small software organizations that are not capable of bearing the cost of establishing software process improvement programs especially when new people are needed. It offers an improvement program through established and documented standard processes that can be adopted and maintained to ensure quality conformance. It does not expect from the practitioner the knowledge of international standards or quality models practices and maturity is not required, but is expected to be gained. SQIMSO is designed to offer guidance and suggested actions are explained thoroughly. Mainly, methods are defined to provide guidance and suggest a way of implementation.

Many practitioners need to know how to do things to avoid implementation faults. Although quality model formalism may limit creativity, but until knowledge and skills are attained it is better to have a quality system that tells how to do things. However, since some practitioners like to do things their own way, the "how

to do" suggestions remain informative in their nature. The suggested methods are not recipes to be abided by. They provide sufficient guidelines that can be adapted during implementation to suit practical work. Practitioners may add or modify suggested methods, but elimination is not preferred. However, the practitioner should ensure that the "what to do" requirements are to be achieved.

Further, SQIMSO emphasizes continuous improvement through quality values and culture. It also emphasizes: efficient communication to meet the new designed quality system; teamwork and fair distribution of jobs and responsibilities; empowering employees with necessary knowledge and skills; giving more value to working people as well as software process and the product itself.

SQIMSO Concepts and Structure

Process: A software process can be defined as an environment of capable interrelated resources managing a sequence of activities using appropriate methods and practices to develop a software product that conforms to customers' requirements.

The requirements are not the only input to the process. In addition, supplier objectives add to customer requirements in order to construct the process input. In fact, this factor is important because it is related directly to the process itself and provides guidance for assessors to know what are the limits and restrictions for software process improvements.

Based on ISO TR 15504 (SPICE), SPIRE Model and other suggested software process improvement frameworks, the software processes in SQIMSO are subdivided into software sub-processes. In SPIRE, sub-processes differ in impact; thus, one can focus on sub-processes that meet business goals. This view is somehow deficient because neglecting some important sub-processes will affect the stability and adequacy of quality models. Another alternative can be to combine sub-processes requirement and to eliminate irrelative parts since we are dealing only with one scale of organizations. In small organizations, it shouldn't be left to practitioners to evaluate and decide what to use and what to ignore. Naturally, the success of a software sub-processes depends on the outcome of its preceding sub-process. Therefore, any hidden neglect may lead to deficiencies and rework. If practice indicates that a sub-process is weak in terms of its practical applicability, this sub-process should be reviewed and adapted to facilitate implementation, instead of avoiding it.

In SQIMSO the Organization processes of the SPIRE model are combined and embedded into other adapted categories to fit small scale. The management process category plays an important role in process categories interaction and control. In order to facilitate determining process requirements and guidance, a practical process framework is established within SQIMSO. The answer to "when to do things?" is reflected in the sequence of customer-supplier and engineering processes. For other processes they provide baselines, when needed, to show process support activities sequence. The overall structure of SQIMSO processes is defined in Table 1.

Table 1: SQIMSO Model Process Structure

Process Category	Process ID	Process Title
Customer-Supplier	SQIMSO-C1	Acquisition
	SQIMSO-C2	Supplier Selection
	SQIMSO-C3	Requirements Elicitation
Engineering	SQIMSO-E1	Requirement Analysis and Specification
	SQIMSO-E2	Software Planning
	SQIMSO-E3	Software Design
	SQIMSO-E4	Software Implementation
	SQIMSO-E5	Software Testing and Integration
	SQIMSO-E6	Software Maintenance
Support	SQIMSO-S1	Documentation
	SQIMSO-S2	Configuration Management
	SQIMSO-S3	Quality Assurance
	SQIMSO-S4	Joint Review
	SQIMSO-S5	Methods Selection
Management	SQIMSO-M1	Project Management
	SQIMSO-M2	Improvement Program
	SQIMSO-M3	Human Resources Management

SQIMSO's Process framework consists of the following elements:

* Purpose and Justification
* Process Inputs.
* Who is involved?
* What to do?
* How to do?
* Process Outputs.
* What to be ensured?

Capability: Process capability is defined as the range of expected results that can be achieved by following a process or the ability of a process to achieve a required goal. The process capability can be measured by maturity key process areas to determine the level of current capability and maturity achievement. This measurement helps organization's management to develop appropriate quality improvement strategy. the levels of capability and maturity will be used as measurement tools to determine the current status of work environment. Passing from level to another will lead the organization to achieve better process quality results.

Methods: Methods are the procedures, patterns, techniques, metrics, roadmaps and all other tools used to develop software products that meet customer's requirements and achieve supplier's business objectives. They may be used to develop resources, monitor deficiencies and enhance engineering and management approaches. Methods answer the important question *"How to do things?"* In order to achieve process and

business objectives, methods should be selected, adopted, adapted and used.

Methods have a great influence on product quality conformity and sometime wrong selection may lead to project failure. They are an essential part of industry's best practices. Methods vary depending on the characteristics of the project. Methods selection may be tedious but this activity should be performed to ensure implementing best practices. Once methods are selected and defined it is the role of quality assurance teams to control methods implementation and to assess and report any deficiencies. Although the methods selection process should be established, maintained and controlled by software quality teams, developers' input and feedback should also be considered.

EXAMPLES OF PROCESSES

Requirements Elicitation Process (SQIMSO-C3)

Purpose and Justification: The purpose of this process is to identify customer requirements and collect the necessary information about existing running systems and procedural works. This process is very critical because its success does not depend only on supplier work but also on customer response and efficient coordination.

Process Inputs?
* Contract Contents
  - System boundaries and Environment
  - Financial and liabilities Obligations
* Customer Adequate Requirements
* Existing Customer Business Processes
  - Operational and procedural workflow.
  - Dataflow and document flow.

Who should be involved?
* From Supplier Side: Project Leader; Analysts; Designers.
* From Customer Side: Top Management; Line Management; Operational Staff.

What to do?
In order to achieve better requirements elicitation the supplier should:
* Assign responsibilities and select the qualified people to perform the job.
* Prepare an investigation plan.
* Select the appropriate method to excavate facts.
* Conduct and Track investigation plan.
* Hold joint review sessions to approve elicited requirements.
* Manage customer requirements changes.

How to do?
The supplier should
* Assign responsibilities for managing requirements elicitation
* Prepare an investigation plan and inform the customer about its phases and all other related factors such as requested reports, peoples, on site visits etc…
* Define the methods to be used in investigation process,
* Interviewers must be selected carefully, because they affect both fact finding performance and organization image. The interviewers should be: impartial; precise and impersonal; tactful; positive; qualified; skilled in dealing with personalities.

Afterwards, review sessions session should be held with customer to check and control any anomalies.

Process Outputs?
* Product Requirements,
* Business Process Identification and Classification,
* Existing Workflow and data flow description documents,
* Collected Forms, Layout, Documents and Reports.

What to be ensured?
During investigation supplier should ensure that:
* Facts are recorded clearly and precisely,
* Answers are fully considered,
* Interviews are made with the right peoples,
* Any contradictions in answers are solved and clarified,
* Interviews covered all related areas of organization,
* Interrelationship and interfaces are covered.

Requirements Analysis and Specifications Process (SQIMSO-E1)

Purpose and Justification: The purpose of Requirements Analysis and Specifications process is to analyze Software requirements and specify its elements in defined description to form the sufficient inputs for design process.

Process Inputs
* Product Requirements,
* Existing Workflow and data flow description documents,
* Collected Forms, Layout, Documents and Reports.

Who should be involved?
Project Leader; Analysts; Designers.

What to do?
Analysts should:
* Determine what to computerize,
* Define Data Flow and Data Elements
* Define transfer states,

* Determine functional parameters and conditions,
* Determine and describe used logic and algorithms,
* Explore and determine input and output physical resources,
* Define interfaces and operational environment.
* Determine specifications criterion for verification and testing,
* Review and verify outcome documents according to contract contents
* Get customer approval on requirements specifications.

How to do?
Developers can use Data Flow Diagrams, truth and decision tables, finite state machine, organization charts, flowcharts, or any modeling techniques to describe requirement specifications. It is very important to select the appropriate description method according to subject been described and whatever techniques is used it should be self-descriptive. At this stage objects or entities should be nominated and relationships should be defined. Terminology adopted by requirement analysis and specification process should be standardized and specified to be used all over the developed system.

Process Outputs?
* Data Flow Diagrams
* Entities and Data Elements Descriptions
* Processes nature and logic descriptions
* Data Stores and sizing
* Input and Output Specifications
* Hardware requirements

What to be ensured?
The supplier should ensure that:
requirements specifications are fully stated,
requirements are specified in details,
any omissions, contradictions, or ambiguity are solved, operational environment is specified, all specifications meet with requirements purpose, descriptive tools are standardized and are commonly used and understood by the customer, all internal reviews and joint review meeting are documented and tracked, any changes in the requirements are controlled and reflected.

Configuration Management Process (Sqimso-S2)

Purpose and Justification: Software Configuration Management is a set of activities that have been developed to manage and control changes throughout the entire software life cycle. It is responsible to identify software configuration items, control various versions, control changes, audit software configuration and report all changes.

Process Inputs?
* Configuration Items
* Product Strategy Plan

Who should be involved?
* Project Manager
* Project Leader
* Configuration Management Administrator
* Software Quality Assurance Members

What to do?
* Define the types of elements that should be identified as configuration items.
* Determine the mechanism to identify and control configuration items and product versions.
* Define configuration item descriptions and status.
* Determine the mechanism to control changes and provide coordination for updating.
* Maintain version releases and relative support.
* Maintain historical information and track released product with customers.

How to do?
* Establish Baseline throughout the entire engineering processes:
* System specifications
  - Object models
  - Process models
  - Data models
  - Executable documents
* Planning
  - Human resources allocation
  - Time schedule
* Design Specifications
  - Architectural Design
  - Detailed Design
  - Interface Design
* Coding
  - Modules
  - Procedures, Macros
  - Input/Output screens
  - Reports
* User Manual
* Database Description
  - Schema and file structure
* Maintenance Documents
* Identification description includes:
  - Fundamental and technical specifications
  - Effective development tools
  - Interfaces
  - Documents
  - Status
  - Files related to configuration items.
* To manage control versions configuration an evolution graph can be used, but instead of just mentioning version name or number, a document reference can be added.
* Use central server repository decomposed into project subdirectories that is decomposed in its turn into version subdirectories and assign authorization for configuration administration.

* For each developer Create workspace separated from central repository.
* Prepare change request form that helps controlling changes procedures and that contain:
  - Date and time of the request
  - Project
  - Version
  - Module
  - Brief description for the Module
  - Developer assigned
  - Developer workspace
  - Changes description
* Check-Out Configuration Management
  - Date and time stamp
* Testing and integration results and approval
* Check-In Configuration Management
  - Date and time stamp

Process Outputs?
* Configuration Management Baseline
* Product Configuration Documentation
* Development Storage Configuration

What to be ensured?
* Do not overlap versioned source or object phase subdirectories.
* Ensure that changes are released according to an official request form.
* Ensure source access control to provide more security
* Ensure that simultaneous update is controlled.
* Establish the baseline and control its changes.
* Record version of tools used.
* Ensure that Configuration Management Baseline is conducted effectively throughout the whole development cycle life.

Improvement Program Process (Sqimso-M2)

Purpose and Justification: The purpose of this process is to explain a way of implementing Software Quality Model *SQIMSO* as an improvement program. This process does not suggest starting improvement program from scratch, but the implementer should understand and consider what exists in his organization and determine what are the significant deviations from standard Processes; then, assess whether suggested processes could be integrated in the development work.

Process Inputs?
* Software Quality Improvement Model
* Organization Resources
* Capable Pilot Team

Who is involved?
* Management
  - Pilot Team
* Software Quality Assurance Members

What to do?
Assign pilot team to study and understand SQIMSO Model.
* Identify implementation opportunities.
* Define the scope of pilot project and allocate adequate resources.
* Conduct pilot project development
* Assess improvements feedback and analyze impact of implementation and lessons learned and report to concerned management.
* Setup and conduct training
* Change development process accordingly

How to do?
* Select a project at low risk and assign development task to credible development team.
* Pilot project team should be:
  * Methodological and systematic
  * Flexible and dynamic
  * Willing to improve
  * Skilled and experienced in software development and maintenance
  * Objective and reasonable.

Process outputs?
* Adopted and Maintained Software Quality Improvement Model
* Product Quality
* Software Process Quality
* Efficient Human Resources Management

What to be ensured?
Ensure that:
* Personal resistance to change is considered and handled
* Status Quo analysis is done before implementing the new model.
* Weaknesses of the organization are not considered as Quality Model weaknesses.
* Assigning the right credible developers.
* Pilot project team is formed from within the organization.
* Assessment is not built on current process defensive view.
* Resources factors changes and overall turnover are reported objectively
* Software Quality Assurance team are involved in the pilot project.
* Pilot project is given enough time to achieve its purpose.
* The improvement program is intended to improve quality and not to misevaluate existing development process.

CONCLUSION

Small organizations are not capable of bearing the cost of establishing software process improvement programs and the existing software process standards or models are not easily applicable. Thus, a practical and comprehensive quality improvement model is proposed. SQIMSO provides a practical framework that gives small software organizations the opportunity to have a feasible quality improvement model to improve their software product quality and to provide software practitioners with quality culture and skills. Future work would involve empirical evaluation of this process model and identifying appropriate metrics for such evaluation.

ACKNOWLEDGEMENT

We thank the National Council for Scientific Research for partially supporting this work.

REFERENCES

1. ISO 9001, ANSI/ISO/ASQC Q9001-1994. Quality Systems-Models for Quality Assurance in Design Development, Production, Installation and Servicing.
2. Paulk. M., 1995. How ISO 9001 compares with the CMM? IEEE Software, pp: 74-83.
3. Paulk, M., B. Curtis and M. Chrisis, 1993. Capability Maturity Model Version 1.1. IEEE Software, pp: 18-27.
4. ISO 9000-3, 1991. International Standards, ISO 9000-3:1991(E). Quality management and quality assurance standards, Part 3, Guidelines for the application of ISO 9001 to the development, supply and maintenance of software.
5. Johnson, D. and J. Brodman, 1999. Tailoring the CMM for Small Organizations and Small Projects. In: Elements of Software Process Assessment and Improvement, edited by K. El Emam and N. Madhavji, IEEE Computer Society, pp: 239-257.
6. Steinen, H., 1999. Software Process Assessment and Improvement. In: Elements of Software Process Assessment and Improvement, edited by K. El Emam and N. Madhavji, Edward Brothers, USA: IEEE Computer Society, pp: 57-75.
7. El Emam, K., Drouin, J. and Melo, W., (Ed.), 1998. SPICE, The Theory and Practice of Software Improvement and Capability Determination. IEEE Computer Society.
8. Sanders, M., 1998. SPIRE Project, Centre for Software Engineering, Dublin, Ireland.
9. Wells, D., 2001. www.extremeprogramming.org/what.html.
10. Paulk, M., 2001. Extreme Programming from a CMM Perspective, IEEE Software, Nov-Dec., 19-26.
11. Hall, T. and D. Wilson, 1997. Views of software quality: a report field. IEEE Proc. Software Engineering, 144: 111-118.
12. Beirne, M., A. Panteli and H. Ramsay, 1997. Going soft on quality? Process management in the Scottish software industry. Software Quality J., 6: 195-209.
13. Stelzer, D., W. Mellis and G. Herzwurm, 1997. A critical look at ISO 9000 software quality management. Software Quality J., 6: 65-79.
14. Wilson, D. and T. Hall, 1997. Perception of software quality: A pilot study. Software Quality J., 7: 67-75.
15. Wilson, D., 1997. Software quality assurance in Australia: An update. Software Quality J., 6:81-87.