Framework of Software Quality Management Using Object Oriented Software Agent

Anand Kr Pandey [1], Dr. A.S. Saxena [2] and Rashmi Pandey [3]

**[1] Research Scholar, Mewar University,
Chittorgarh, Rajasthan, India**

**[2] Prof. Dept of CSE, M.P.C.T.,
Gwalior, Madhya Pradesh, India**

**[3] School of Computer Application, ITM University,
Gwalior, Madhya Pradesh, India**

Abstract

Development of software is a scientific and economic problem, particularly the design of complex systems whichrequire evolving methods and approaches. Agent technology is currently one of the most active and vibrant areas of IT research and development. Object-oriented Software Engineering (OOSE) has become an active area of research in recent years. In this paper, we review the framework of software quality management using object-oriented methodology concepts for software agents.The software specification acts as a bridge between customers, architects, software developers and testers. Using object-oriented concept of software agent and its standard it may offer benefits even if the system is implemented without an object-based language or framework . We propose and discuss a software agent framework, specifically to support software quality management. Although still in its initial phases, research indicates some promise in enabling software developers to meet market expectations and produce projects timeously, within budget and to users' satisfaction. However, the software quality management environment has also changed and is continuously evolving. Currently software projects are developed and deployed in distributed, pervasive and collaborative environments and its quality should be managed by applying its best standard. From the point of view of software engineering this framework and its standards are applying for developing the software projects.We discuss the standard and benefits that can be gained by using object-oriented concepts, and where the concepts require further development.

Keywords: Agent, Framework, Design, Software Quality Management, Autonomy

1. Introduction

The software quality management and its standard acts as a bridge between customers, architects, software developers and testers. The terms "objects" and "oriented" in something like the modern sense of object-oriented programming seem to make their first appearance at MIT in the late 1950s and early 1960s. This process is the part of software analysis and software analysis often occurs after requirements gathering, and before architectural design. Software analysis is the activity of understanding the requirements, so that decisions can be made about how to architect it. Software quality management and assurance is gaining increasing attention throughout the software lifecycle. While software testing has become mainstream in professional software development, sophisticated quality management in the early phases of software development is still not evolved that far. Existing methods for software analysis and quality management include data flow diagrams, state transition diagrams, and object-oriented modeling. These methods have all been applied successfully, and all have their own restrictions and limitations.

Now a days software agents are probably the fastest growing area of Information Technology (IT). Software agents, by definition, are active, independent components. Most agents are designed to act as or for the user to help execute some task or operation [1]. In this paper, we discuss the review of designing framework for software quality management and its standards. The development of Software Project Management (SPM) as an independent application area and field of study. SPM includes, amongst other things, the management of all issues involved in the development of a software project, namely scope and objective identification, planning, evaluation, project development approaches, software effort and cost estimation, activity planning, monitoring and control, risk management, resource allocation, as well as managing contracts, teams of people and quality [2]. In this work, we focus on our *software specification framework* and demonstrate its strong integration of constructive and analytic software engineering methodology. For better and efficient software quality management and programming service, software can be defined by differentiating basic environment for launching, basic service and extended service. Software agent technology offers a promising solution to addressing software quality management

Problems in a distributed environment. According to this technology, software agents are used to support the development of SPM systems in which data, control, expertise, or resources are distributed. Software agent technology provides a natural metaphor for support in a distributed team environment, where software agents can help the project manager and team members to monitor and coordinate tasks, to apply quality control measures, to validate and verify, as well as to ensure proper change control [1]. We specifically concern ourselves with the question of how software agents can be used to improve quality management in a distributed environment. Here we discussed

about software quality and constructing quality using specification method. The next section contains a background study and a discussion on software quality management in the context of the software project management framework.

2. What is Software Quality Management

The first question arise that what is software quality? The first is that quality means "meeting requirements." of customer, because customer defines quality as to whether the product or service does what the customer needs. The professional views about software quality reflects conformance to requirement and fitness for use. Software quality is something that everyone wants. Manager know that they want high quality, software developers know they want to produce a quality product, and user's insist that software work consistently and be reliable [2]. We can say that a quality product is one which meets its requirements and satisfies the user. The aim of Software Quality Management (SQM) is to manage the quality of software and of its development process. To mantain the quality we have to ensure that the required level of quality is achieved in a software product and for manage all the activities and processes we have to encourage a company-wide "Quality Culture" where quality is viewed as everyone's responsibility.

The Software Management Body of Knowledge defines software quality management as the processes required to ensure that the software will satisfy the needs for which it was undertaken. It includes all activities of the overall management function that determine the qualitative policy, objectives and responsibility, and implements these by means of quality planning, quality assurance, quality control and quality improvement, within the quality system [3]. Quality management not only includes the concepts, tools and methods of quality assurance, but also validation and verification. The SQM should have following features.

(i) **Quality planning:** determining which quality standards are relevant to this specific project and deciding how these standards will be met.

(ii) **Quality assurance:** involves evaluating overall performance regularly to ensure conformance to the set standards. Quality audits or reviews can support this function.

(iii) **Quality control:** monitoring the activities and end results of the project to ensure compliance with the standards utilizing various available tools and techniques.

However, software quality management should not be considered as a separate developmental phase but should be an inextricable part of all phases and all processes during software project management.

3. Role of Object Oriented Software Agent in SQM

Software agents, by definition, are active, independent components. Most agents are designed to act as or for the user to help execute some task or operation. The object-oriented software agent paradigm has some flaws related to design and implementation of multi-agent systems, we also believe that it is still the most practical programming language to implement the agent technology. Each developer and researcher in the agents field adopts their own definition of an Agent.Object oriented software agent technology offers a promising solution to addressing software quality management problems in a distributed environment. According to this technology, software agents are used to support the development of SPM systems in which data, control, expertise, or resources are distributed [4]. Software agent technology provides a natural metaphor for support in a distributed team environment, where software agents can help the project manager and team members to monitor and coordinate tasks, to apply quality control measures, to validate and verify, as well as to ensure proper change control. Software agents can be grouped, according to specific characteristics, into different software agent classes. Object-oriented concept developed as the dominant programming methodology in the early and mid 1990s when programming languages supporting the techniques became widely available.

In this paper the use of object oriented software agents is investigated as a potential tool for improving the quality management of SPM processes. Whether or not an agent has a user interface, depends on whether it collaborates with humans, other agents or hosts. Software agent technology is being explored as a promising way to support and implement complex distributed systems. In this section, we briefly consider how agent technology is currently being deployed in SQM by considering some application examples and standards [3]. This research approach is based on the fact that object-oriented software engineering has proved to be extremely powerful for building complex systems, which promote modularity, maintainability, and reusability.

3.1 Features Of Software Agents

There is a minimum set of common features that typify a software agent.Various features have proposed different definitions of agents, these commonly include concepts such as [5]:

- *Autonomy:* Agents should be able to perform the majority of their problem solving tasks without the direct intervention of humans or other agents, and they should have a degree of control over their own actions and their own internal state.
- *Social ability:* Agents should be able to interact with other software agents and humans in order to complete their own problem solving and to help others with their activities where appropriate.
- *Perceptive:* A software agent is perceptive; it is able to perceive and respond to changes in its environment.
- *Reactivity*: Agents perceive the context in which they operate and react to it appropriately.
- *Monitoring:* A software agent can be used to monitor a computer system and software system to make sure that it is performing and functioning correctly.
- *Interaction:* An object oriented agent can communicates with the environment and other agents by means of sensors and effectors.

The autonomy characteristic of a software agent distinguishes it from general software programs. Autonomy in agents implies that the software agent has the ability to perform its tasks without direct control, or at least with minimum supervision, in which case it will be a semi-autonomous software agent.

4 SQM Framework

The software agents are used to control and monitor several activities execution at various sites in an open source platform supporting distributed software engineering processes.

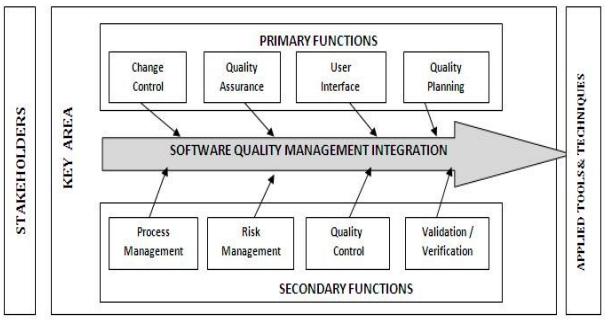

Fig 1: Framework of Software Quality Management

The design of the framework of SQM using object oriented model for overall system, based on components (specialised agents), which simplifies the design and programming of agents. The proposed approach for designing framework encourages the separate handling of object oriented modeling concerns, and provides a disciplined scheme for their composition [6]. An object oriented software agent can collaborate with other agents in order to use the counterpart services via some communication language, it provide the abstractions that explicitly capture the structure. This framework technology support the ability to build applications by selecting and assembling objects from libraries.

Above figure 1 shows the framework for SQM. The following specialised factors of working agents are used in our discussion making for the software quality management framework that we present in this section. Here multiple methods are used for variable parametres. The following responsible factors are as:

(i) *Change Control:* Change control is a systematic approach to managing all changes made to a product or system for designing framework. It is a formal process used to ensure that changes to a product or system are introduced in a controlled and coordinated manner.

(ii)

(iii) *Quality Assurance:* It is a process-centered approach to ensuring that a company or organization is providing the best possible products or services. consists of a means of monitoring the software engineering processes and methods used to ensure quality.

(iv) *User Interface:* The user interface is one of the most important parts of any program because it determines how easily you can make the program do what you want. A powerful program with a poorly designed user interface has little value. The user interface, in the SQM is the space where interaction between user and model occurs.

(v) *Quality planning* consists of determining which quality standards are relevant to this specific project and deciding how these standards will be met.

(vi) *Process management:* This is the ensemble of activities of planning and monitoring the performance of a process during the activities of applied tools and techniques on SQM.

(vii) *Risk Management:* Software Quality Management (SQM) is concerned with all the processes, methods, and practices that affect quality during designing framework when producing, supporting, and operating software. Essentially, SQM is risk management, identifying and addressing all the factors that can negatively impact all those processes, methods, and practices. While often the most visible form of managing risk, testing developed code is only one of many aspects that SQM considers. The strategies to manage risk typically include transferring the risk to another party or avoiding the risk or reducing the negative effect or probability of the risk, or even accepting some or all of the potential or actual consequences of a particular risk.

(viii) *Quality Control:* Quality control (QC) is a procedure or set of rules and functions intended to ensure that a designed framework or performed service adheres to a defined set of quality criteria or meets the requirements of the client or customer.

(ix) *Validation & verification:* In software quality management, software testing, and software engineering, verification and validation (V&V) is the process of checking that a software system meets specifications and that it fulfills its intended purpose. It may also be referred to as software quality control. Verification and Validation are independent procedures that are used together for checking that a product, service, or system meets requirements and specifications and that it fulfills its intended purpose.

5 Conclusion

In this paper we investigated an approach for designing framework of software quality management using object oriented software agent technology to address the challenges proposed in the software project management arena. The concept of a agent is most useful as a tool to analyze systems, not as a prescription. We focused on one of the key elements of Software quality Management, namely software quality management, and designed a generic agent framework to address all the tasks of this key element. This framework forms a basis for other key elements, and could be adapted into individual frameworks and then coordinated by an overall multi-agent system to achieve the objectives of Software Project Management. Our framework follows an approach of agent teams being composed of specialised software agents.

References

[1] A. Graesser and S. Franklin, "Is it an agent, or just a Program?" A Taxonomy for Autonomous Agents". In Proceedings of the Third International Workshop On Agents Theories, Architecture, and Languages, Springer-Verlag, 1996.

[2] Chen, F, Nunamaker, J. F., Romano, N. C. & Briggs, R. O. (2003). "A collaborative project management architecture". Proceedings of the 36th Hawaii International Conference on System Sciences. Big Island, Hawaii.

[3] D'Inverno, M. & Luck, M. (2001). "Understanding Agent Systems". Berlin: Springer-Verlag.

[4] Leonard Foner, "What's an Agent Anyway?", Agents Memo 93-01, Agents Group MIT Media Lab, 1993.

[5] Michael Wooldridge and Nick Jennings, Dept. of Computer science, Manchester Metropolitan University, IEEE Review, January 1996.

[6] P. Maes, Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back. MIT Press, 1991, ISBN 0-262-63135-0.

First Author Anand K. Pandey, MCA, M.Tech., is a research scholar for Ph.D. program from Mewad University, Rajasthan. He has been published 05 papers in reffered journals. He has 10 years of teaching experiance and also written a book titled "Simulation and Modelling" for BE, MCA and M.Tech. students.

Second Author Dr. A. S. Saxena, prof in Dept of C.S.E. at M.P.C.T. Gwalior has 40 years of teaching experiance and also published 10 research papers in reffered journals. He has been guided 06 Ph.D. students.

Third Author Rashmi Pandey, MCA, is currently working as a Asst. Prof. in computer application dept at Institute of Applied Science and Computer Application, Gwalior. have been published 6 papers in reffered journals. She has 7 years of teaching experiance.