

# Software Quality Assurance in a Remote Client/Contractor Context (May 2005)\*

A.H. Black, R.J. Foss  
DEPARTMENT OF COMPUTER SCIENCE  
*Rhodes University, Grahamstown*  
E-Mail: [g00b0795@campus.ru.ac.za](mailto:g00b0795@campus.ru.ac.za)  
Phone: +27 46 6038619, Fax: +27 46 636 1915

**Abstract—With the reliance on information technology and the software that this technology utilizes increasing every day, it is of paramount importance that software developed be of an acceptable quality. This quality can be achieved through the utilization of various software engineering standards and guidelines. The question is, to what extent do these standards and guidelines need to be utilized and how are these standards and guidelines implemented?**

**This research focuses on how guidelines developed by standardization bodies and the unified process developed by Rational can be integrated to achieve a suitable process and version control system within the context of a remote client/contractor small team environment.**

**Index Terms—IEEE, ISO, Project Management, Quality Assurance**

## I. INTRODUCTION

THERE exist today numerous standards and guidelines for the management and operation of software development projects, each of which have their own merits and application areas.

The two main software standardization bodies in place today are The International Organization for Standardization (ISO) and The Institute of Electrical and Electronics Engineers, Inc (IEEE). ISO developed a set of standards known as ISO 9001 and this set is titled “Quality systems – Model for quality assurance in design, development, production, installation and servicing” [9]. ISO then published a guide for this standard, ISO 9000-3, which is titled “Quality management systems – Part 3 – Guide for the application of ISO 9001 to the development, supply and maintenance of software” [9]. Subsequently, ISO refined and updated this guide and published ISO 90003 in 2004 [4]. The IEEE has also developed numerous standards and guidelines to aid in quality assurance for software development [3].

The Rational Unified Process (RUP) is the most widely utilized set of software development guidelines [10]. RUP provides a software development process which helps to ensure software quality and the development of a repeatable process [11].

Code versioning systems have traditionally been used to store source code, but it is possible for these version control mechanisms to store almost any information. Versioning systems are particularly useful when numerous developers and team members are working on a single project, requiring all artifacts for the project [2].

This paper describes how a repeatable process has been developed to ensure software quality. The process has been tailored towards a remote client/contractor small team environment with integrated version control, utilizing the unified processes developed by Rational.

## II. CONTEXT

The Audio Engineering Group (AEG) at Rhodes University in the Computer Science Department has, and is currently involved in the implementation of projects for overseas clients. The research in this paper is based upon these projects and the impact quality assurance has on them.

In conducting this investigation into the impact of quality assurance on these projects, the standards and guidelines have been examined and implemented. Along with this investigation into how these factors affect the projects, certain processes and solutions have been developed to aid in the development of a repeatable project process.

As the project work is done for an overseas customer, the unique aspects of contract work for an overseas client/contractor is the focus. RUP provides a very extensive level of detail for project processes and is traditionally focused at teams larger than that of a typical small project team [8]. The AEG usually has about 4 people working on a project at a time, so the characteristic of a small team utilizing the RUP processes is another unique aspect of this research.

Communications between overseas clients and AEG are normally done through email and occasional phone calls. These communications are done to communicate project status and also to send project artifacts.

\* This work was undertaken in the Distributed Multimedia Centre of Excellence at Rhodes University, with financial support from Telkom SA, Business Connexion, Comverse, Verso Technologies, THRIP, and the National Research Foundation.

The team members of the AEG are not all located in the same city, so further emails to project members are required to relay project artifacts. An FTP server was set up for the management of software packages and for the clients to check on the status of the work.

It became evident through the course of the research that there was a requirement for remote process and version management control. This would give the client the ability to check on the status of the project as a whole and check on certain projects artifacts. This too would give the team members who are remotely located access to all project artifacts and aid them in seamlessly working together on the same project.

### III. PROCESS

As previously mentioned, RUP is one of the most widely used software development processes. It provides a very extensive level of detail for the software development process; it is well documented and very concise, providing templates and examples for numerous project artifacts [8].

The RUP architecture can be seen in figure 1. The process itself is broken into various phases, iterations and workflows.

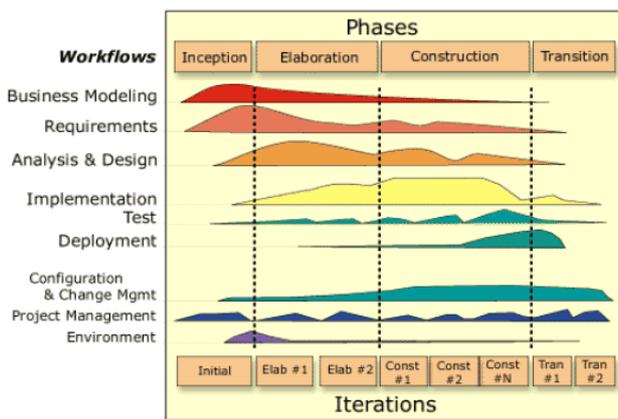


Figure 1: The RUP architecture

The horizontal and vertical axes in figure 1 represent the following [11]:

- The horizontal axis represents the life of the project and is broken into phases and iterations. The phases are the main 4 stages of the project and iterations are the steps within each phase to complete the phase.
- The vertical axis represents the workflows. These are the main activities within the project and each workflow involves various activities, artifacts and workers.

The horizontal axis is broken into 4 main phases and each of these phases represents a particular stage in the project [11]:

- The inception phase involves defining the project as a whole and initiating plans for the progression of the project

- The elaboration phase involves building on the project plans from the inception phase and developing certain design and architecture analysis of the problem
- The construction phase involves solving the actual problem the project has set out to achieve
- The transition phase involves the handover of the end product and the correction of any problems that may arise.

As can be seen in figure 1 certain workflows involve more work in certain phases. For example the implementation workflow involves more work in the construction phase than the requirements workflow and vice versa. Within each of these phases there are iterations which are progressive iterative steps through the phase to ensure its completion.

As previously mentioned the vertical axis represents the workflows and within each workflow various activities, artifacts and workers are involved, which can be seen in figure 2.

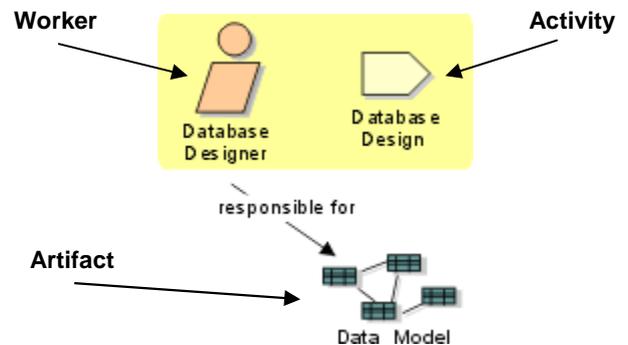


Figure 2: Worker, activity and artifact

The worker represents a team member responsible for a particular role in the project. The activity is something that the worker is required to perform to accomplish their role. The artifact is the product of a worker accomplishing their role and performing their activities.

For each of the workers, RUP provides detailed information regarding exactly what activities they should perform and what these activities entail. For the artifacts, RUP provides detailed information regarding what the artifacts are and why they should be created, along with some template and example documents.

The information that RUP provides for use in the software development process is not all intended to be used. The whole idea behind RUP is to tailor the process towards individual project needs and to use as much detail as seems fit for the individual project. This was a prime motivator for utilizing RUP as the process control approach in the remote client small team context.

### IV. VERSION MANAGEMENT

Version Management has been used for a lengthy period of time now in software development projects to store source code and help aid in the development of systems where there are numerous developers.

There are many such products available and they are utilized in conjunction with many development environments. Two of the most well known version management systems are Concurrent Versions System (CVS) [2] and Microsoft SourceSafe [13].

#### A. Concurrent Versions System (CVS)

CVS is an open source network-transparent version control system and is available on both the Microsoft Windows and Linux environments and has numerous front-ends developed for these environments. CVS is a powerful, stable and reliable version management system.

CVS has a client/server architecture which allows developers to access the repository from anywhere with an Internet connection [2]. The version management system tracks all access to the repository and builds log and history files and also has the capability to show the differences between different versions of a file stored within the repository.

#### B. Microsoft SourceSafe

Microsoft SourceSafe is a version control system similar to that of CVS. SourceSafe has the ability to integrate directly into many development environments within the Microsoft Windows environment. However numerous features found in CVS are lacking in SourceSafe. SourceSafe lacks branching support, cannot be safely extended, and has reliability issues on busy networks or slow connections [5].

For these reasons CVS was chosen as the tool utilized within the solution offered by this research.

Traditionally version management has only been utilized as an implementation scheme for source control. These systems are quite capable of handling files of any kind and a move towards versioning all project artifacts has begun. One such product which is similar in nature to the solution offered by this research is Merant Professional. This product offers a version control and project tracking tool for software development projects [6].

The version manager provided by Merant Professional allows users to access the repository via the Internet using a graphical user interface (GUI). Through this GUI users can check out project artifacts from the repository, utilize these artifacts, and commit any changes to the artifacts back into the repository. The version manager keeps track of the status of artifacts and who is currently working on the artifacts. The tracker offered by Merant Professional is also of a GUI nature and allows users to login to the tracker system and post queries and comments on a particular project or project artifact.

What was required by the AEG was a system that could easily integrate with the processes offered by RUP and incorporate the version control offered by CVS. Merant Professional was a close contender for what was required, but the system seemed very disjoint in the version control and project tracker and did not integrate with process

control. What was required was a single solution offering all the benefits of version control with CVS and process control offered by RUP. For this, the Project Process Control and Versioning System (PPCVS) was developed.

### V. PROJECT PROCESS CONTROL AND VERSIONING SYSTEM (PPCVS)

The PPCVS was developed to meet the needs of the AEG. The PPCVS provides both the best of what RUP has to offer and the version control offered by CVS all in one solution, which is accessed through the use of a GUI interface. The PPCVS allows for the seamless interaction of all team members on a project. The PPCVS also provides valuable status information to clients on the project as a whole and the ability to check on the status of certain project artifacts within a project.

This system comprises two crucial elements, the repository itself and the interface to the repository. The interface to the repository was developed in the C# .NET development environment and the main window for the application can be seen in figure 3.

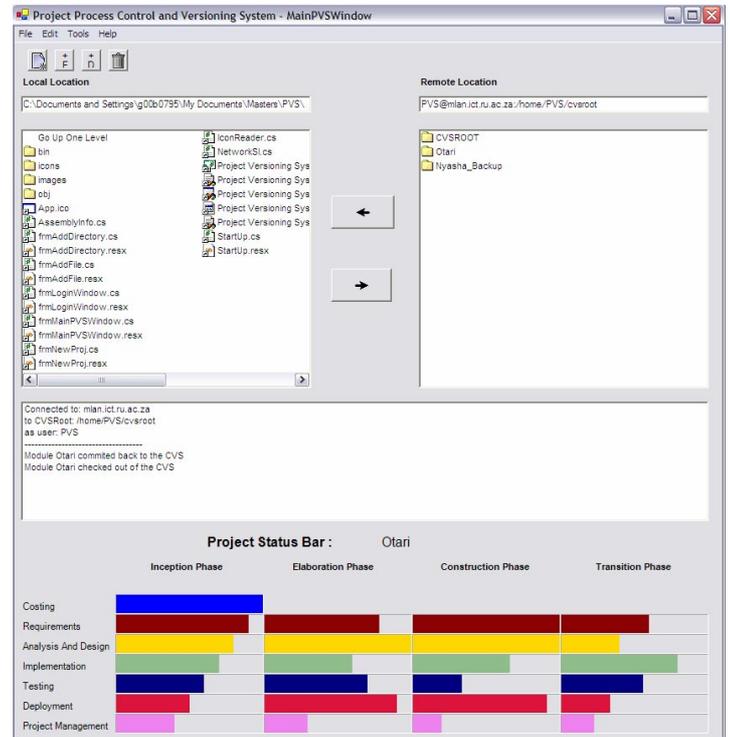


Figure 3: The main PPCVS window

The repository is a CVS running on a Linux Server. The repository houses all the project artifacts and is accessible through the use of the PPCVS or a normal front-end CVS client.

The basic object model for the system can be seen in figure 4. As can be seen in figure 4 the main functionality of the system is to provide the user with the ability to create a new project in the repository and check out and commit changes to the project artifacts. Along with this, there is the functionality to add files and directories to an existing module within the repository.

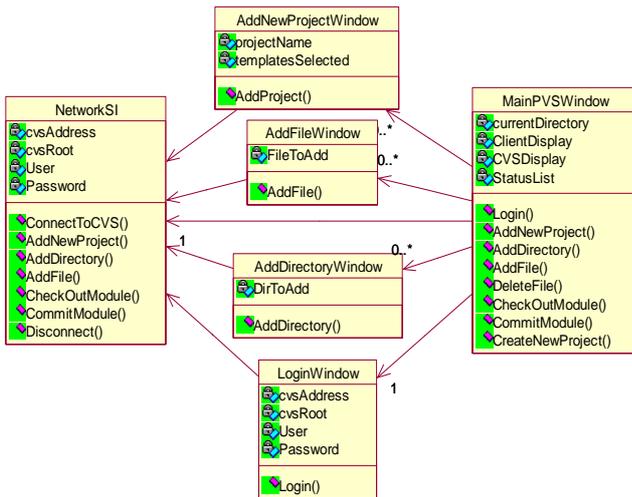


Figure 4: The PPCVS object model

In figure 1 the left hand pane represents the files on the local client machine and the right hand pane shows the project artifacts on the CVS Server. The user is able to select a module or individual project artifact and check it out onto the local machine by clicking on the arrow pointing towards the left hand pane. Once the user has finished editing or viewing the project artifact, they are able to select the artifact and click on the arrow facing towards the CVS Server artifacts. This will commit any changes to the artifacts back to the server.

#### A. The create new project process

When the user decides to create a new project within the repository he is taken through a series of graphical selection panes from which he can select various template and information documents which have been developed from RUP. These documents have been selected from RUP, based on their effectiveness in the project process for projects undertaken by the AEG. Not all the template and reference material provided by the system is taken from RUP, some of the documentation is from the IEEE and ISO standards set and some created from scratch.

The systems requirement specification template was created from the IEEE recommended practice for requirement specification [12]. This recommendation was a far more concise document and fitted the AEG requirements elicitation process.

Some of the RUP workflows have been removed and one extra workflow added to that seen in figure 1. The costing workflow was added to the process. This was a necessary addition, as the AEG needed to cost projects to tender for contracts, and make allowances for resource allocation. As this process would only take place at the start of every project it would only have one phase, that being the inception phase. The costing technique chosen for this process is the Constructive Cost Model II (COCOMO II). This costing model allows for the input of various project attributes and project projections and outputs a project cost based in person hours [1]. Previously the AEG was doing this costing technique by estimating cost based on previous projects done. COCOMO provided a solid process based on

best practices. For this process a Microsoft Excel spreadsheet was developed, which allowed for the inputting of data and the calculation of the project cost.

In table 1 the project artifact set can be seen, the inspiration for this set being partly taken from a paper written by M. Hirsh entitled "Making RUP Agile" [8].

TABLE 1  
THE PPCVS PROJECT ARTIFACT SET

| Costing             | Costing Spreadsheet               |
|---------------------|-----------------------------------|
| Requirements        | Vision Document                   |
|                     | System Requirements Specification |
|                     | Use Case Model                    |
| Analysis and Design | Software Architecture Document    |
|                     | Design Model                      |
| Implementation      | Implementation Model              |
| Testing             | Defect List                       |
| Deployment          | Release Notes                     |
|                     | Installation Artifacts            |
| Project Management  | Software Development Plan         |
|                     | Iteration Plan                    |
|                     | Iteration Assessment              |
|                     | Microsoft Project Status File     |

Once the user has selected the template and reference artifacts he wishes to have in his new project, the new project module is created within the repository with all the selected artifacts. After this is done, the project is available for all team members to use.

#### B. The Project Status Bar (PSB)

In figure 1 the Project Status Bar (PSB) is located at the bottom of the window. This bar is used to display the status of a current project. The bar takes a similar form to that of the RUP architecture shown in figure 1. The workflows have been edited to only show the required workflows necessary for the AEG projects. The PSB can be seen in figure 5.

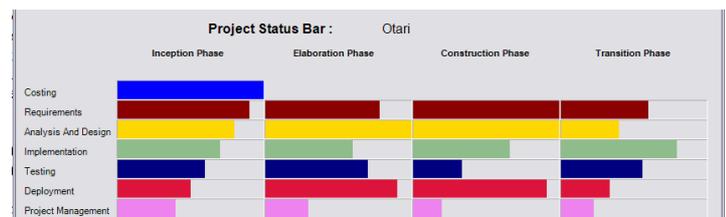


Figure 5: The Project Status Bar

The seven workflows all have associated artifacts which can be found in table 1. Each bar represents a phase within the workflow and the amount of work that has been completed within that phase of the workflow.

The information for the PSB is obtained from an XML file, which represents the project schedule drawn up in Microsoft Project. Each project that is created by the PPCVS is given a template Microsoft Project file which has the 7 workflows within the four phases of the RUP life cycle. The project manager is responsible for the updating of this file in the repository. This then makes project status information readily available to anyone using the PPCVS. By clicking on a particular bar, the artifacts associated with that workflow's phase are opened. This makes the process of checking on unfinished work extremely easy.

## VI. FUTURE WORK

There is no reliable Application Programmer Interface (API) for the communication between the C# .NET development environment and a Linux or Windows based CVS. The one API that is available is SharpCVSLib and is hosted on SourceForge.Net [7]. It is still in its alpha version of development and has numerous bugs and problems. The communications between the PPCVS and the Linux CVS is done through code developed for the purpose of the PPCVS only and the development of a complete and concise API would be very beneficial to the C# development community.

## VII. CONCLUSION

The PPCVS has provided the AEG with a repeatable process that utilizes the RUP methodology, uses effective version management and project status review. This enables all team members to work together irrespective of their locations. The artifact set that has been compiled for the system has been tailored towards the small team, so that no excessive work is done on artifacts that are not required. This also enables the team to very easily and quickly start a new project. The clients can very easily login to the PPCVS, check on the status of the project using the PSB, and view certain project artifacts to ensure they are satisfactory.

The use of RUP has attributed to many successes and process improvements in the software engineering field. Version control provides any project with a solid basis for control over the development of source code and project artifacts. RUP lacked a version control mechanism and the PPCVS has provided this bridge between these two highly valuable project processes and aided in the development of an effective project process for the AEG.

## REFERENCES

- [1] COCOMO II Program Affiliates. 'COCOMO II Model Definition Manual', 1999
- [2] CVS Homepage. 'Concurrent Versions System' Available: <http://cvshome.org>
- [3] IEEE Homepage. 'The Institute of Electrical and Electronics Engineers, Inc.' Available: <http://www.ieee.org>
- [4] ISO Homepage. 'The International Organization for Standardization' Available: <http://www.iso.org>
- [5] M. Bolton. 'Visual SourceSafe Version Control: Unsafe at any Speed?'. Available: <http://www.michaelbolton.net/testing/VSSDefects.html>
- [6] Merant. 'Merant Build For Professional User's Guide', Hillsboro, Oregon, 2003.
- [7] M. Krueger, C. Harbour. 'SharpCVSLib'. Available: <http://sourceforge.net/projects/sharpcvslib>
- [8] M. Hirsh. 'Making RUP Agile', presented at the 2002 Conference on Object Oriented Programming Systems Languages and Applications, Seattle, Washington.
- [9] O. Oskarsson, R.L. Glass. 'An ISO 9000 Approach To Building Quality Software', Prentice Hall, 1996.
- [10] P. Kroll. 'The Rational Unified Process: An industry-wide platform for best practices'. Available: <http://www-128.ibm.com/developerworks/rational/library/873.html>
- [11] P. Kruchten. 'The Rational Unified Process An Introduction', Addison-Wesley, 2000.
- [12] The Institute of Electrical and Electronics Engineers, Inc. 'IEEE Recommended Practice for Software Requirements Specifications', IEEE Std 830-1998, 25 June 1998.
- [13] Visual SourceSafe Website. 'Microsoft Visual Studio Developer Centre', Microsoft. Available: <http://msdn.microsoft.com/vstudio/previous/ssafe/default.aspx>

**Angus Hugh Black** has completed an Honours degree in Computer Science at Rhodes University. He is currently reading towards a Masters degree at the same institute and is a Telkom bursar.

**Richard John Foss** is an associate professor in the Computer Science Department of Rhodes University. His research interests include the management of small projects in the field of IEEE 1394 based audio systems.

