Functional Automation – Rational ClearCase™

**2009**

Yinan Ma

Swinburne Software Guru Pty Ltd

11/2/2009



Contents

[Executive Summary 2](#_Toc244910294)

[Target Audience 2](#_Toc244910295)

[Scope 3](#_Toc244910296)

[Purpose 3](#_Toc244910297)

[Recommendations 3](#_Toc244910298)

[Company Background 3](#_Toc244910299)

[Project Description 6](#_Toc244910300)

[Client 7](#_Toc244910301)

[Project Team 7](#_Toc244910302)

[Stakeholders 8](#_Toc244910303)

[Goals 10](#_Toc244910304)

[Objectives 10](#_Toc244910305)

[Project Environment 12](#_Toc244910306)

[Deployment environment 12](#_Toc244910307)

[Software Tools 13](#_Toc244910308)

[Project SCM Activities 14](#_Toc244910309)

[Collaboration 14](#_Toc244910310)

[Manageability 14](#_Toc244910311)

[Project Assets Baselines 14](#_Toc244910312)

[Procedures for changing baselines 15](#_Toc244910313)

[Procedures for processing change requests and approvals-change classification scheme 15](#_Toc244910314)

[Level of Control 15](#_Toc244910315)

[Handling Document Revision 16](#_Toc244910316)

[Major functionalities – Overview of ClearCase functional automation 16](#_Toc244910317)

[Accessing, maintaining and managing project assets 17](#_Toc244910318)

[UCM, base ClearCase – automating and integrating change management process 18](#_Toc244910319)

[VOB, Snapshot and Dynamic Views – automating workspace management 19](#_Toc244910320)

[Baseline - automating the creation process 20](#_Toc244910321)

[Automating software build process 21](#_Toc244910322)

[Branching and merging – automating parallel development processes 22](#_Toc244910323)

[Meta data – automating policies and annotations 23](#_Toc244910324)

[Report Generation – automating analysis and decision making 24](#_Toc244910325)

[Multi-platform – automating team member training and adoption 24](#_Toc244910326)

[ClearCase Multi-Site – automating distributed work process 25](#_Toc244910327)

[Recommended automations 26](#_Toc244910328)

[Version Control and workspace management – VOBs and Views 26](#_Toc244910329)

[Parallel Development 26](#_Toc244910330)

[Build avoidance 26](#_Toc244910331)

[Baseline 27](#_Toc244910332)

[Meta data 27](#_Toc244910333)

[Report Generation 27](#_Toc244910334)

[Conclusion 28](#_Toc244910335)

[References 30](#_Toc244910336)

# Executive Summary

## Target Audience

The audience of this report is project leaders, management and interested individuals who are considering implementing software configuration management tools for their project. The audience should have an understanding of real world software project concerns and issues and preferably have technical knowledge in the field of software processes and engineering.

## Scope

The scope of this report addresses a medium size project and software configuration functions relevant to the size of this project. Details of the project descriptions are provided in this report.

## Purpose

The purpose of this report is to provide a sufficient understanding and educate the reader on the benefits of implementing ClearCase as tool for managing software versioning and collaboration issues.

## Recommendations

We recommend the reader to learn ClearCase and find the best way to utilise the tool for their project. We also recommend the reader to further develop their understanding of software configuration issues and concerns and use ClearCase to address these concerns.

# Company Background

Swinburne Software Guru Master Students Pty. Ltd is a new company that will provide IT solutions and advice in the manufacturing, distribution and services industries. Swinburne Software Guru Master Students Pty. Ltd has a great achievement for over 10 years within the industry. Swinburne Software Guru Master Students Pty. Ltd is a proprietary limited organization owned by Adam Smith and John Doe.

Mr.Adam Smith and Mr.John Doe both have left their respective jobs in order to specialize in custom software package solution to medium and large sized businesses.   
  
Mr.Smith’s previous employment was with Swinsys Corp acting as an Software engineer. Mr. Doe’s previous employment was with S&S Software acting as chief information officer.

Swinburne Software Guru Master Students Pty. Ltd*.* Targets its market from small to medium sized companies and government organizations within the Southern part of Victoria including Melbourne and surrounding areas. Swinburne Software Guru Master Students Pty. Ltd*.* seeks major contracts with medium sized firms.

Those contracts will be served with the assistance of strategic alliances, both with other Information Technology and engineering companies such as Swinsys Corp. and S&S Software as well as other professional groups.

This strategy gives the company the flexibility it needs to successfully secure and complete projects with varying elements. By using existing contracts and joint ventures with other consulting firms, Swinburne Software Guru Master Students Pty. Ltd. is in a position to corner the on-site assessments market in Victoria. By year 3, Swinburne Software Guru Master Students Pty. Ltd. will expand to other markets such as Queensland, and other region of Australia.

The primary objectives of Swinburne Software Guru Master Students Pty. Ltd. over the next year are to:

* Generate one new client contract a month by networking with key industry leaders, conducting seminars and workshops, and joining key environmental agencies;
* Generate a net profit of $35,000 in the 2010 fiscal year of operations by developing a strong client base and keeping overhead costs to a minimum;
* Develop and conduct 10 Information Technology seminars that meets the needs of the surrounding communities.

Swinburne Software Guru Master Students Pty. Ltd. Headquarters is located at:

**280 Collins Street, Suite 1003  
Melbourne 3000  
Victoria**

The office space is leased and accommodates the necessary office equipment such as computers, fax machine, photo copier and other supportive equipments.

Swinburne Software Guru Master Students Pty. Ltd has other regional branches in Sydney and Auckland. These branches are geographically separated to help the organization gains market value within the Australia and Oceanic region.

Swinburne Software Guru Master Students Pty. Ltd. currently employ up to 1000 staffs including programmer, system analyst, database programmer, designer, legal officers, and other supporting staffs. This number of staff is believed can help Swinburne Software Guru Master Students Pty. Ltd to be the leading IT solution organization in Victoria.

# Project Description

Swinburne Software Guru needs an online portal website to facilitate and promote interactions among its internal staff in an effort to increase working productivity. Through this portal site, Swinburne Software aims at actively promoting and discussing ideas that inspire and support creative leadership. This ‘information portal’ will include articles, blogs, podcasts and videos both submitted by internal staff or fed by external resources utilizing various web services and Web 2.0 technology.

## *Client*

Swinburne Software Guru Master Students Pty. Ltd and associated organisations are the primary clients of the project.

## Project Team

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Position | Internal Extension | Email | Direct Involvement |
| Adam Virzi | Programmer | 6134 | [avirzi@ssgm.com](mailto:avirzi@ssgm.com) | Yes |
| James Addison | Programmer | 6124 | [jaddison@ssgm.com](mailto:jaddison@ssgm.com) | Yes |
| Ian Ma | Project Lead | 6100 | [ima@ssgm.com](mailto:ima@ssgm.com) | Yes |
| Matthew Davis | CIO | 6124 | [mdavis@ssgm.com](mailto:mdavis@ssgm.com) | No |
| Jessica Kyle | Graphic Artist/Designer | 6024 | [jkyle@ssgm.com](mailto:jkyle@ssgm.com) | Yes |
| Joey Torvalds | CMS Consultant | - | [jtorvalds@drupalfun.com](mailto:jtorvalds@drupalfun.com) | Yes |
| Joseph Lang | QA | 6075 | [jlang@ssgm.com](mailto:jlang@ssgm.com) | Yes |

## Stakeholders

|  |  |  |  |
| --- | --- | --- | --- |
| Organisation | Role | Notes: |  |
| Swinburne Software Guru | Client and project initiator | Swinburne Software Guru initiated this project to address existing culture and organisational needs. |  |
| Swinburne Software Guru Employees | End user, evaluator, potential survey target, training and support | End user of the outcome of the project. Potential targets for survey and training rollout |  |
| Drupal Fun | Web technology Consultant | External organisation contracted to provide CMS and hosting advices and services. |  |
| AussieHost | DNS and hosting provider | Service provider for DNS and database hosting. |  |
| Swinburne University | Financial provider | Provides funding for the project. |  |
| Deloitte | Accounting and auditing | Provides accounting and auditing services including team member payroll and funding allocation. |  |

## Goals

The main goal of this project is to develop an online portal system that would enable Swinburne Software Guru Master Pty. Ltd to share information, ideas and thoughts among its staff and increase productivity and efficiency in terms of culture, organisational and knowledge management.

The development of this online portal system as a part of Swinburne Software Guru Master Students Pty. Ltd main website would facilitate creation of a internal forum for actively promoting and discussing ideas that inspire and support creative leadership and thereby enabling Swinburne Software Guru Master Students Pty. Ltd to demonstrate their own expertise and thought leadership.

This information portal is to be designed in such a manner so that it facilitates posting of articles, blogs, podcasts and videos that reflect Swinburne Software Guru Master Students Pty. Ltd own and others’ expert views on a wide range of timely topics relating to Technology breakthrough, IT development and productivity.

## *Objectives*

The objective of the project is the successful creation and acceptance of the Swinburne Software Guru Social and Networking Portal website.

1. To develop an ‘information portal’ which will include articles, blogs, pod casts, etc, that reflect Swinburne Software Guru Master Students Pty. Ltd’s own and others’ expert views (e.g. business commentators, academics) on a wide range of timely topics relating to Information and Technology breakthrough.
2. Create, design and deliver a virtual space that will foster and grow a community of thought leaders by supporting like-minded individuals in the B2B space to engage, interact and connect with each other.
3. To provide a system with restricted access rights and level of access control to ensure appropriate use.
4. The system should have straight forward and extensible installation, maintenance support and deployment.
5. To encourage and promote exchange of ideas and thoughts. As well as improving relationship between staff.

## *Project Environment*

### Deployment environment

|  |  |
| --- | --- |
| Hardware | PC portable computers  PC Desktop computers  Apple IMac, IBook and IPhone |
| Operating Platforms | Windows XP Service Pack 3  Apple OS X Leopard and Snow Leopard  All Linux distributions |
| Internet Browser | IE explorer 7 to 9  Firefox 2 – 3.5  Apple Safari |

### Software Tools

|  |  |
| --- | --- |
| **Category** | **Software** |
| Browser | IE Mozilla Firefox  Safari |
| Programming and Graphic developement tools | Adobe Dreamweaver Adobe Photoshop Adobe Fireworks Adobe Flash  IBM Rational ClearCase and ClearQuest |
| Database Engine | PostgreSQL |
| Office tools | Microsoft Office Suite |

## Project SCM Activities

Before analysing which SCM activities that worth to be automated, a brief description of all relevant activities in the context of our project is needed. This section outlines specified and potential project activities.

### Collaboration

The project team must be able to share and contribute equally to the project. We need to implement effective and easy to use solution for sharing and controlling work flow, resources and processes.

### Manageability

The project leaders, support staff and management must be able to produce and audit all performance and resource utilization metrics.

### Project Assets Baselines

Identify various asset baselines for the project

Baselines are established to designate significant milestones during the engineering and development cycle, as well as to document requirements, design and applicable processes. Multiple baselines are defined to capture and version all assets up to the point of the milestones in questions.

### Procedures for changing baselines

There are three classifications of baselines, informal checkpoints, integration and verification, and release.

For informal baselines the Configuration Manager must notified of the change. The change must be tested and built by the developer, then built by the Configuration Manager to ensure that the build process is not broken by the change. The Configuration Manager then baselines the configurations with a “checkpoint” label.

Integration and verification baselines must be approved by the team leader and the Software Engineering Manager.

### Procedures for processing change requests and approvals-change classification scheme

This activity includes:

* Change reporting documentation
* Change control flow diagram

### Level of Control

Software Baselines require the approval of the configuration manager, a description of the change, and a completed successful build that incorporates the change.

Software Release requires consensus among the Software Development Team leads and the approval of the Software Engineering Manager, as well as detailed descriptions of change, and a complete tested build that incorporates the change.

Software Release to customer requires Change Control Board approval, detailed description of changes and test reports, a review of the integration and compatibility matrix, and completion of required customer documentation.

### Handling Document Revision

Documents in the Software Development team are identified as Configuration Items and are controlled in the same way as source code. If a document is required to be released company-wide, or if the document is a process detail that requires approval from the change control board, the document will be assigned a part number and undergo review and release process.

# Major functionalities – Overview of ClearCase functional automation

This section provides an overview of the key automation functions in ClearCase and related Rational™ Suite integrations. First, we will outline each of the functions and then provide an explanation and rationale using configuration management methodology.

Then we will justify each of them in context of our project and provide you with a selection rationale to aid management decisions.

## Accessing, maintaining and managing project assets

We needed a tool to facilitate two basic features for software developers. We need to be able to share code/work amongst the team and we need to guard and rollback against unstable progressions.

We define project assets as any articles, source code, documentations, designs and also directory structure of the project file system.

The ability to manage abstract assets is what sets ClearCase apart from traditional version control softwares.

The outcome of managing project assets is achieving efficiency.

### UCM, base ClearCase – automating and integrating change management process

Change management is an essential function of software project management. The ability to modify, store and deliver software according to the changing needs of the project requirement is critical for project success.

ClearQuest is the de facto tool within Rational™ suite to provide change management. However, ClearCase uses change management models to facilitate integration with ClearQuest.

UCM (Unified Change Management) is the out-of-box change management process model. UCM is activity based; project assets that are changed during a change activity are mapped and audited accordingly.

Base ClearCase is a term that refers to a set of tools within ClearCase to configure a tailor-made change management solution that suits the need of the project.

Both UCM and base ClearCase functions as an effective change management model that can be further used and integrated with other tools.

### VOB, Snapshot and Dynamic Views – automating workspace management

Workspace is an abstract collection of project assets and resources that facilitates individuals to work as a cohesive unit.

Workspace enables an individual developer with the ability to work and later on integration his work with others.

ClearCase takes a “Sandbox” approach when dealing with workspace management.

VOB (Versioned Object Base) is ClearCase’s terminology for a software repository. Developers use the check-in and check-out functions to interact with VOBs. VOB keep track of all versioned assets and auditing information.

To put it simply, the major advantage of using ClearCase *View* feature is that it guarantees project consistency and stability.

Each *View* maintains and keeps its own changes; developers working with other *Views* are not affected by changes that occurred within a particular *View*. However, they may access other views provided they have the appropriate access permissions.

ClearCase View is available as Snapshot and Dynamic modes. In Snapshot model, ClearCase *View* retrieves and copies all items that are listed in the configuration specification of the project.

Dynamic mode uses ClearCase’s MVFS (multi-version file system) and facilitates immediate and transparent access to data in the repository.

ClearCase *View* automates workspace management by providing version selection, source retrieval and knowledge sharing.

### Baseline - automating the creation process

In ClearCase, a baseline is a set of versioned project assets that is consistent with a particular project mile stone.

An example of a project baseline would consist of all project source code files of beta 0.1 and possibly also including alpha 0.2’s forked changes.

ClearCase provides baseline function for both the UCM and base ClearCase interfaces.

The important distinction between the two baseline functions is that base ClearCase requires the developer to configure *config spec* of his workspace (view); *config spec* is a set of rules that describes and controls the versioned project assets. An example of a *config spec* rule would be the requirement that all source code of module 101 should only include code version 0.11.

UCM do not require individual developer to do this, versioned specification and related requirements of the project baseline are configured by the maintainer or manager of the project prior to usage.

Using the baseline automates the development creation process. By providing all team members the right set of versioned assets; the project management ensures the consistency and stability of the project code base.

### Automating software build process

Space and time restrictions are common to all software engineering projects. Traditional version control systems such as CVS implements *source* control but do not automate the software building process.

A scalable and reasonably complex project would undergo many changes and releases during of the duration. Different branching within the project with different configuration of features would require different builds for release.

We needed a tool to automate this process; the alternative is using a manual and ad hoc process where we would manually link and combine source files for compilation. The main functional requirement of this automation is build avoidance; or in laymen terms: avoid building/compiling unchanged part of the project assets.

In ClearCase, the function of automating software building is accomplished by using the tool clearmake and omake.

clearmake is ClearCase’s implementation of the popular UNIX utility *make* with improved features.

omake is another tool which is unique to ClearCase. It is best used in the dynamic view interface.

## Branching and merging – automating parallel development processes

In small projects, we can keep track of changes and versioning manually or implement multiple version control systems; this method quickly becomes impractical when project scales.

Modern SCM tools typically offer branching and merging support for project assets. The purpose of branching is to facilitate parallel development on one project.

We can have multiple branches for a single project; preferably have a single master branch to act as a baseline.

The main impetus of this function is the need to have manageability when variations of the source code occur due to multiple changes and versioning.

ClearCase implements direct branching and merging functions in base ClearCase process model. UCM use internal streaming method to manage branches.

Merging is the function of combining development branches. ClearCase will automatically combine differences across the branches. In case of conflict (where differences contradict each other), the maintainer of the project must manually resolve each conflict.

## Meta data – automating policies and annotations

Development procedures and policies are useful functions and should be a part of all project processes.

ClearCase automates policies in two ways. By annotating the project assets and by assigning trigger and locking functions on key functions such as check-in, check-outs, reporting etc.

In the annotation method, project manager assigns meta-data to project assets. Meta data can be attributes, branch types, hyperlinks, version information and much more. Effectively, we are attaching labels to project assets. From this, we can group and associates labels of baseline assets for example.

Attaching labels to different versions will preserve relationship between versions. This is a powerful function and we strongly recommend integrating this function/practice in future projects.

Trigger and lock are also useful ClearCase features.

Triggers can be attached to project assets; for example, we can set a trigger to automatically check if a check-in item conforms to the required documentation procedure.

We can lock a particular branch when a release is imminent; so that no new source files are checked in. This would be very useful in situations where we need to ensure that the proper testing is completed on all development.

Annotation labels and trigger/lock combined together would give project maintainer and management powerful ways to define and manipulate development standards and procedures.

## Report Generation – automating analysis and decision making

ClearCase can use the *report builder* to specify *report parameters* and report viewer to output and format the result of a report.

ClearCase’s reporting tools are fully programmable either by specifying reporting procedures or by utilising the ClearCase programming interface. This offers power flexibility on generating and keep track of project performance/resource metrics.

We recommend specifying and keep a standard set of report templates for all projects.

## Multi-platform – automating team member training and adoption

ClearCase is available for deployment on both Windows and UNIX –derived operating systems. This means that VOBs and Views used on one platform are fully compatible and accessible from ClearCase on other platforms. This greatly aids training and adoption as usage is not limited by operating platform.

## ClearCase Multi-Site – automating distributed work process

ClearCase by default uses centralized VOB repositories for source and version control. For projects with geographically distributed members, ClearCase Multi-Site offers VOB replication and mirroring.

However, ClearCase Multi-Site incurs additional licensing costs and we would recommend multi-site based on resource and requirement needs.

## Recommended automations

### Version Control and workspace management – VOBs and Views

ClearCase provide sophisticated and streamlined version control and workspace management via VOB repository and workspace views. We recommend this function as it is an essential component of ClearCase usage. We also recommend using the UCM model for rapid integration with ClearQuest.

### Parallel Development

ClearCase offers efficient branching, merging for parallel development. We also recommend using UCM’s integrated stream model for transparent and safe branching management. We recommend this function as this would increase code base consistency and work productivity.

### Build avoidance

ClearCase use clearmake and omake for achieving build avoidance. We recommend this function as doing so would reduce resource and time requirements and offer consistent releases for project deliverables.

### Baseline

ClearCase baselines are essential for automating the creation process. We recommend this function because it can facilitate fast and reliable branching and also enforcing milestone policies.

### Meta data

We recommend assigning labels, triggers and locks for enforcing versioning and policies. By doing so, we also enforce delivery standards.

### Report Generation

We recommend utilizing reporting builder and viewer to audit and measure project metrics. This would greatly aid decision makers and will provide an insight into the state of the project progression.

# Conclusion

We observed that majority of software professionals use SCM tools to automate versioning and collaboration processes during their development. It is important to conclude that implementing SCM functions in projects have a positive benefit in terms of productivity, code base consistency and metric collection and reporting.

ClearCase is the tool that we feel can achieve this positivity without impacting too much on costs and time. ClearCase provides out-of-box usage model and integrates well into other software tools which we feel is essential for project teams that do not wish to retrain their staff.

ClearCase is also multi-platform and flexible enough for further customisation.

We recommend project managers and leaders to keep a set of build scripts, directory templates, documentation standards, report templates for deployment in future projects. This should be easy to accomplish given the fact that ClearCase keep tracks of all project assets, project leaders should use ClearCase to record their scripts and templates in all projects.

We do not recommend teams to invest in ClearCase multi-site unless the team can utilise the benefits of a distributed system as the additional licensing costs will impact negatively on project budgets and resources.

ClearCase performance is a concern early in our investigation but we believe that the features and robustness of ClearCase out weights any performance problems. In practice, we have observed that medium and large sized projects benefited the most from ClearCase features and if configured and maintained properly, performance should not have a significant impact on project delivery.

# References

ClearCase Concepts Guide

<http://techpubs.sgi.com/library/dynaweb_docs/0620/SGI_Developer/books/ClrC_CG/sgi_html/ch01.html>

ClearCase Usage Tutorial

<https://publib.boulder.ibm.com/infocenter/cchelp/v7r1m0/index.jsp?topic=/com.ibm.rational.clearcase.tutorial.doc/topics/default_topic.htm>

ClearCase Build Performance Degradation

[www.cmi.com/PDF/clearcase\_bpd\_report.pdf](http://www.cmi.com/PDF/clearcase_bpd_report.pdf)

Rational ClearCase – Administrator’s Guide, Version 2000.05.00 and later, UNIX/Windows Edition

Using ClearCase for software projects management

<http://manuals.it.swin.edu.au/rational/Rational%20ClearCase%20LT/cc_proj.pdf>

ClearCase, Introduction to features and usage

<http://manuals.it.swin.edu.au/rational/Rational%20ClearCase%20LT/cc_intro.pdf>

Simplifying product line development using UCM streams

<http://www.ibm.com/developerworks/rational/library/1748.html>