



Productivity and Quality With CVS

Case Study

BuildMonkey
<http://www.buildmonkey.com>

Table of Contents

Table of Contents.....	2
Overview.....	3
Benefits and Drawbacks of CVS.....	4
Benefits of CVS.....	4
Drawbacks of CVS.....	4
Mitigating The Risk.....	5
The BuildMonkey Way.....	5
Customer Experience.....	7
Management Information.....	7
Increased Productivity.....	7
Reduced Cost of Defects.....	7
Ease Of Use Promotes Wider Use.....	8
Clearly Defined Processes.....	8
Configuration Management.....	8
Segmentation.....	8
Under The Hood.....	8
Repository Administration.....	8
About BuildMonkey.....	10

Overview

The problems that can arise on software development projects using CVS as the source control and Configuration Management repository are well known, yet CVS continues to be a credible choice for many Project Managers mainly due to it being free and because of the wide availability of developers who are experienced in its use.

Even though the software is free, this does not mean that it does not have a cost associated with it – on a large project the cost of using CVS can be considerable.

It is nonetheless possible to exploit the capabilities of CVS to provide functionality more commonly found in its more expensive counterparts and, if properly implemented, CVS can meet the needs of the most complex of projects.

This document briefly considers the drawbacks, and benefits, of CVS and goes on to describe the BuildMonkey methodologies and software that reduce the cost, complexity and risk of managing software projects in a CVS environment.

Benefits and Drawbacks of CVS

The main points arising from the use of CVS are covered below.

Benefits of CVS

- It is free and open-source;
- It is available on a wide variety of platforms;
- The skill pool of developers experienced in the use of CVS is very large;
- It can be installed and set up very quickly indeed;
- It has a transparent repository, allowing manual modification of repository revisions in the event of problems;
- It has no special backup considerations;

Drawbacks of CVS

- Management information (such as statistics) is very difficult to obtain, where the capability exists at all;
- Breaking a build through checking in erroneous code is very easy to do;
- Rolling back changes once they have been checked in to the repository is very difficult to do;
- The user interface, both command-line and gui, requires technical skill to be used even for the simplest of tasks (such as checking in a document);
- It does not support multiple repositories. Every developer requires network access to a single repository, and this can pose a security risk;
- It does not natively handle branching or multiple threads of development. Branch capabilities exist, but these are clever mutations of the underlying RCS mechanisms (which does not support branching);
- It relies on unix-level permissions and it is therefore difficult to segment or “ring-fence” teams of developers working on different parts of an application.

Mitigating The Risk

It is possible, through the use of robust server-side plugins and shell scripts, to enhance the capabilities of CVS to be more like those of its commercial counterparts (such as Clearcase).

If these are implemented correctly, CVS can provide excellent functionality and can comfortably service the project needs. If implemented wrongly, they can severely impede project stability and progress. As the television programmes would say: *“these people are specially trained – don’t try this at home”*.

The BuildMonkey Way

We view software development projects as assembly lines in a factory. Sometimes there may be more than one assembly line, or more than one factory, but the principles are the same:

Factory	Development Project
Blueprints and plans are provided to the factory.	Blueprints and plans would normally consist of requirements and, sometimes, wire frames or other design materials.
Delivery terms are decided upon.	Choices are made with regard to target platform and architecture.
The manufacturing process is determined, with regard to order, machine placement and quality control procedures.	The development methodology is determined, with regard to coding and documentation standards, CM environment and build processes.
Plant and machinery are obtained and calibrated.	Development tools are obtained and configured (e.g. IDEs, design tools, CM tools such as CVS).
Raw materials are assembled on the production line to create finished, saleable, goods. Quality assurance is built into each stage of the assembly.	Code is developed, compiled and integrated to create an application. Testing is built in to the development process to ensure that the developed code meets functional and non-functional requirements.
Finished goods are packaged and shipped.	The application is packaged and delivered to the customer.

Table 1: Comparison of Software Development and Factory Production

The industrial revolution, with its effects on productivity and production, is well-known. The revolution in manufacturing due to automation and robotics – bringing immense gains in productivity and output – is equally well known.

The BuildMonkey team believes that a similar revolution is underway in the field of software production. That's right, "production" and not "development". We know that *Software and Development are two different things*:

Software is the end objective – a releasable application that will deliver a return on investment;

Development is the means by which that objective is achieved.

Development is only one stage in a multi-stage process – the process of software production. This includes:

- Design and coding
- Configuration Management and version control
- Software construction
- Packaging and deployment
- Testing
- Integration
- Documentation

In other words, to regard a software project purely as "development" may introduce a level of risk since the other aspects of the production process must also be considered.

Like the revolution in manufacturing, automating large parts of the software production process provides massive gains in productivity and quality.

This automation, based on deep technical understanding of the software production process, is performed by a BuildMaster. Like the WebMasters of the 1990's, whose job it was to create and maintain those millions of new web sites, it is the job of a BuildMaster to create and maintain efficient software production environments.

We have off-the-shelf software and scripts that allows us to create a CVS repository that will meet project – and management – objectives and make the software production process less opaque.

The BuildMonkey team are specialist BuildMasters, with many years of experience in this area. When all you do is BuildMastering, you get pretty good at BuildMastering.

Customer Experience

Software development projects can often seem to take on a life of their own. Sometimes, apparent progress can be enormous in return for very little effort and at other times it can seem to be the reverse. It can be very difficult for Project Management to accurately gauge project progress¹ against key milestones.

BuildMonkey provide Project Management with meaningful management information to allow decisions to be made based on objective data, and to allow progress to be accurately gauged and measured:

- What changed, when and by whom?
- How many new defects were introduced by the addition of module X?
- How many man days did defect 'y' take to fix?
- How much of the code had to be rewritten to add or remove feature A?
- Can we recreate the build that was put into test last Tuesday?
- Why did this work yesterday and not today? What changed last night?

Measurable productivity gains are achieved through the use of our BuildMonkey software, which automates large parts of the development cycle.

We are the original BuildMasters, and always the best.

Management Information

If the CVS repository is set up correctly, then it is possible to extract meaningful management and statistical information on the code base through the use of CVS server-side plugins and shell scripts.

Increased Productivity

Productivity is considerably increased through process automation, which also ensures utter consistency and repeatability.

Reduced Cost of Defects

The cost of remedying defects is proportional to time taken to discover it. If a new defect can be discovered immediately when a piece of code is modified, the cost of detecting and fixing it is considerably less than if it is discovered several weeks later during a formal test cycle.

By automating the tests, and integrating them into the build, defects are discovered as soon as they occur – when the cost of fixing them is lower.

¹ It is a well-known metric that technical resources are always '90% finished, one or two small things to tidy up'

Ease Of Use Promotes Wider Use

By providing a web-based interface to the CVS repository, it is possible for non-technical and occasional users to be given read-only (or read-write) access to the repository.

This allows, for example, project administration staff to maintain documentation and other artifacts that are essential to the project without having to learn the intricacies of CVS in order to do so.

Clearly Defined Processes

Clearly defined and automated processes provide stability and consistency during the project life-cycle. Boundaries, hand-overs and sign-offs are well-understood and this facilitates effective teamworking.

Configuration Management

Configuration Management is more than simply checking code in and out of the repository, as there are many artifacts that must be considered part of the “release” at any given time:

- Documentation
- Application configuration files
- Database schema and seed data
- Third-party libraries
- Third-party applications

By producing a Release Manifest, it is possible to reproduce any point in the project life cycle and – importantly – to be able to detect changes to running environments (e.g. changes to the production environment).

Segmentation

It is often necessary to segment development teams for some reason – perhaps due to geographical or organisational constraints, or due to the fact that different parts of the application require different levels of security clearance (as can be the case on Government projects).

It is possible to create a segmented CVS infrastructure that will fit even the most complex segmentation requirements, and back this up with the appropriate levels of security for those different environments.

Under The Hood

Repository Administration

Because of the transparent architecture of CVS, it is possible to apply plugins and other server-side technologies to facilitate the required functionality. Such plugins, however, need to be:

- Performed with care;
- Performed by experienced technical resources;
- Adequately documented;
- Able to be seamlessly removed
- Invisible to end-users

Many of these plugins have already been produced by BuildMonkey and are available off-the-shelf.

Figure 1 shows the topology of a BuildMonkey deployment in a CVS environment.

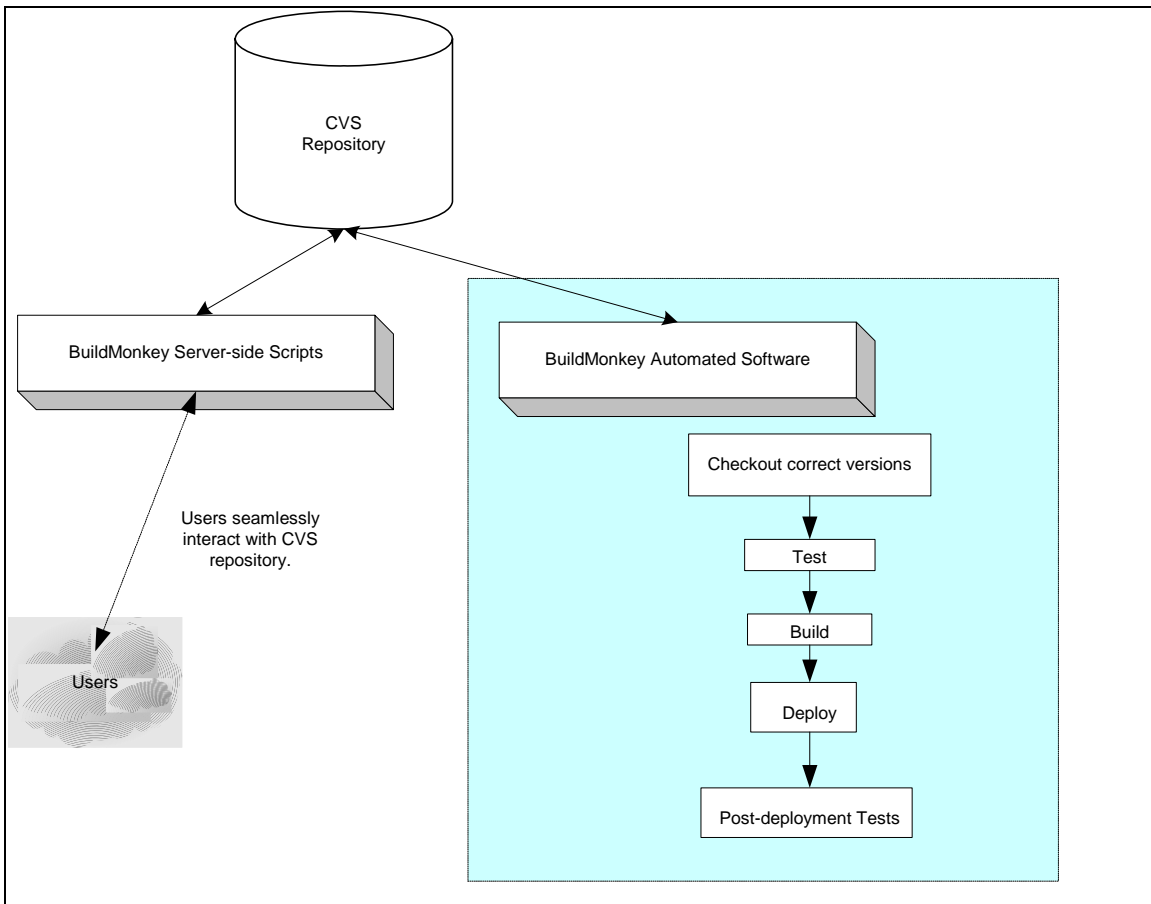


Figure 1: BuildMonkey Software Production Topology

BuildMonkey. Ship on time, on budget, with less defects

About BuildMonkey

BuildMonkey are the market leaders in Build, SCM and Deployment.

Formed in 1999, and with many Fortune 500 and FTSE 100 blue-chip clients, we are the original and the best.

All of the concepts described in this paper have been encapsulated in a suite of off-the-shelf tools and associated processes to facilitate rapid implementation of the Best Practices set out in this paper.

We are passionate about solving the problems which plague software development. We know that, with very little effort, it is possible for software to be delivered on-time, on-budget and free of defects.