



Integrating Agile Methods into your Process Driven Lifecycle

A technical whitepaper demonstrating the benefits of using Openmake® Meister® to manage Agile builds in a CA® Software Change Manager process driven lifecycle

OpenMake Software
213 W. Institute Place, #404
Chicago, IL 60610
800.359.8049
312.440.9545
Email: sales@openmakesoftware.com
www.Openmakesoftware.com



INTEGRATING AGILE METHODS INTO YOUR PROCESS DRIVEN LIFECYCLE	I
INTRODUCTION.....	1
CHALLENGES FOR THE PROCESS DRIVEN LIFECYCLE IN AN AGILE ENVIRONMENT	2
BUILDS IN A PROCESS DRIVEN LIFECYCLE	3
CA SCM AND MEISTER SUPPORTS BOTH AGILE AND WATERFALL.....	5
CONFIGURING CA SCM UDPS.....	6
The Dev State and Continuous Integration.....	6
Setting up the Continuous Integration Server.....	7
MAINTAINING A WATERFALL PROMOTE PROCESS	8
CONCLUSION	9
CORPORATE OVERVIEW	10
Additional Information.....	10
Company Overview.....	10

Introduction

This document discusses how to integrate agile software builds into your process driven lifecycle using both OpenMake Meister and CA Software Change Manager (SCM). Software Configuration Management involves the tracking of source code changes moving from development through production release. The software build process takes the managed source code and creates the binaries that execute in your testing and production environments. Without an integrated build step, there is no audit trail that bridges your binaries back to the source code managed by CA SCM. Matching your source to your binaries is the primary function of the build process.

This document will review how to incorporate an agile build into your process driven lifecycle. It will address both a waterfall and agile approach to software development, explaining how the build step can be integrated using either approach.

This document does not attempt to provide a complete discussion of each of the many features and benefits that Meister can provide to the users of CA SCM. For a more in-depth review of Meister and its features, please review the Meister datasheet and Meister Enterprise Build Management White Paper. Both are available either from a OpenMake Software representative or in downloadable form at www.OpenMakeSoftware.com.

Challenges for the Process Driven Lifecycle in an Agile Environment

A waterfall process and an agile process are not always complimentary. In the waterfall approach, there are tight controls on how and when an application is released to production, including approval stop gates along the way. There is good reason for this. Most organizations have learned that frequent releases to production can cause a production environment to bounce up and down like a misguided ball across a dangerous street. Agile development methodologies have different goals. The ultimate goal of agile is to release a software application into production as quickly and cost efficiently as possible.

Agile development really means "frequent iterations". Frequent iterations means that the application is built as a complete unit, and a release is performed for testing. The more often builds are done, called "continuous integration builds", the quicker developers find problems caused by any two developers coding in opposite directions. This is critical.

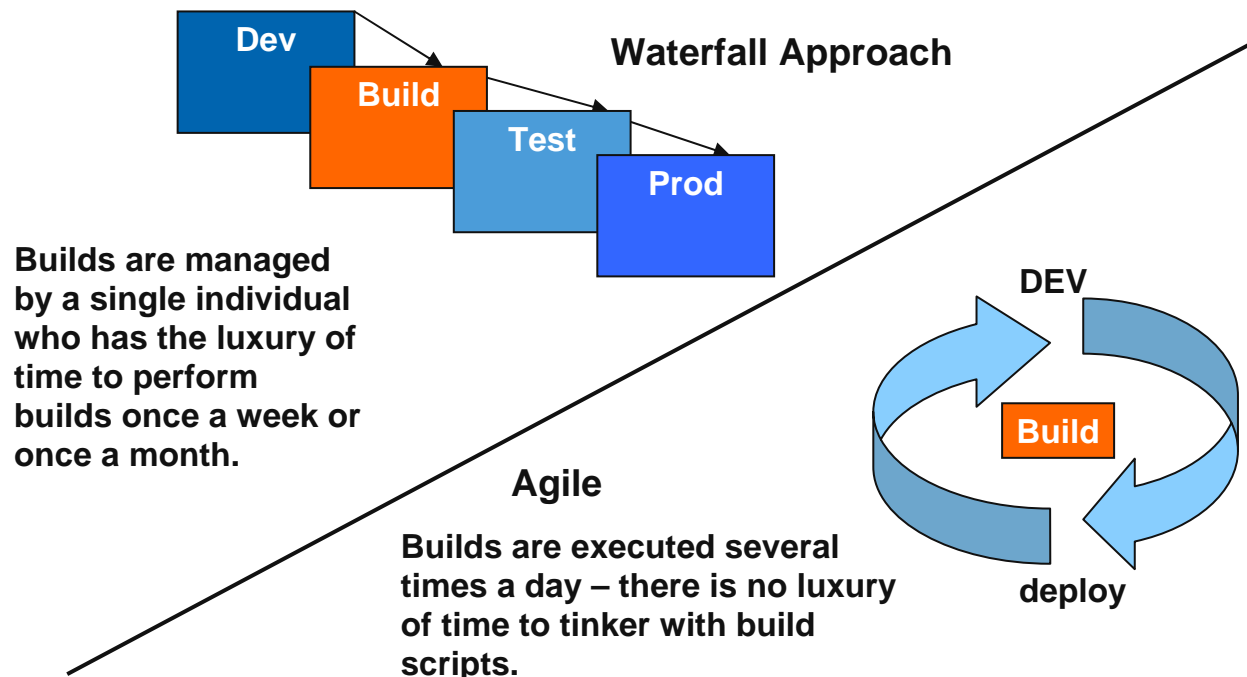
More tightly controlled environments do not need to be in conflict with agile practices. Developers can create an Agile environment while conforming to a waterfall release approach. What is most important is that developers are given the control and flexibility they need in the development processes to support improved development techniques.

This document will provide information on how you can allow your development teams to move toward more agile practices while preserving your lifecycle driven processes using OpenMake Meister and CA SCM.

Builds in a Process Driven Lifecycle

So where do builds fit in the overall development to release lifecycle. The simple answer is "everywhere". Developers execute compiles on a frequent basis while performing their everyday duties. In agile methodologies, developers run their builds locally, called pre-commit or pre-flight builds, as well as perform integrated team builds called continuous integration builds. In a waterfall approach a pre-test build is performed when code is promoted into the testing environment. In some cases, a final, or pre-production build, is executed just prior to production release.

First, let's examine where builds are executed in a standard waterfall lifecycle versus an Agile Lifecycle. In the diagram below, you see the traditional Dev to Prod stages approach representing a lifecycle model. In this approach, integration builds are performed by a Team Lead on a weekly, bi-weekly or even monthly basis. The team lead has the luxury of time to maintain build scripts and to tweak the build until it is successful. Build scripts are managed by the Team Lead, and he alone is responsible for producing a successful build as scheduled, i.e., weekly or monthly. Builds are also executed by each developer, however they typically execute a build that creates only a single binary, often using their IDEs. The integration build and the developer build are out of sync until the Team Lead addresses the integration build scripts and gets everything working as a full build together. The length of time between builds allows this process to work. In most cases several weeks pass before a new clean build is needed. Even though this makes it simpler to manage full builds, it is a core problem for developers attempting to navigate their application and integrate their changes.



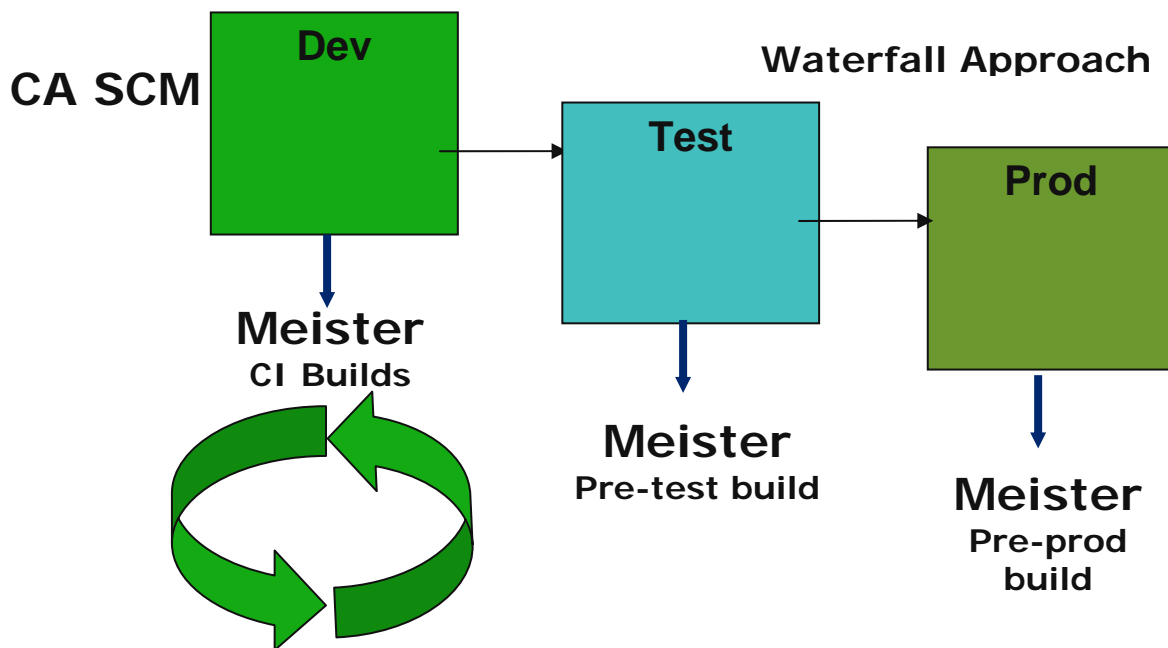
In the Agile approach builds are executed much more frequently, as often as every time source code is checked into CA SCM. This means that there is no luxury of time to update build scripts or to tweak the build until it works correctly. The build must work every time source code is integrated back into CA SCM or the continuous integration build process fails. There is nothing continuous about broken builds. Developers perform this continuous integration process to determine if their unique changes have adversely impacted the overall application. By doing this on a frequent basis, there is less code fixing required. In a waterfall approach, it may be weeks before a developer finds out that his coding changes have broken the build. The sooner the developer discovers this, the easier it is to fix the problems resulting from the impact.

CA SCM and Meister support both a waterfall and agile approach to software development. In fact, CA SCM's process driven lifecycle can support both agile and a more traditional waterfall method for the same application project. In fact, by creating the continuous integration build process, the waterfall build becomes more consistent and improves the development to release process overall.

CA SCM and Meister Supports Both Agile and Waterfall

Out of the box, CA SCM and Meister are integrated and ready to go. You will need to setup your builds using Meister before you define your build processes for either the agile or waterfall States. To setup the States, you define a CA SCM process that calls Meister to support both Agile and Waterfall build methods. There are many ways that this can be done, however this document gives a best practice approach to achieving this goal.

The following diagram shows how both methods can be supported based on the CA SCM State. In this method, the DEV State supports a continuous integration process using Meister. The Test and Prod States call Meister to execute a full build for pre-production, which suits the waterfall approach.



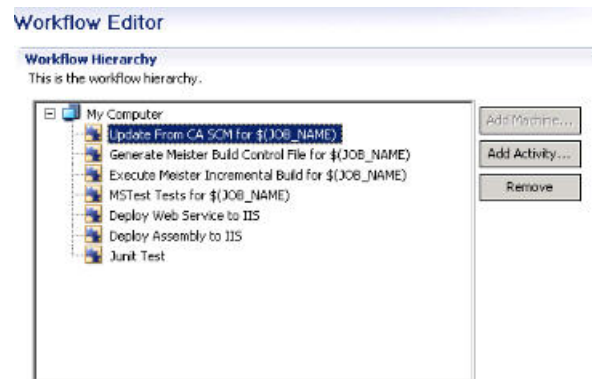
Configuring CA SCM UDPs

In order to support both a continuous integration and waterfall approach, the developers CA SCM State must be configured to support a continuous integration process. Providing a Continuous Integration process within the Development State will not impact the other States in the process, for example Test or Prod.

The Dev State and Continuous Integration

With Continuous Integration developers often need to do more than just call a build on a check-in. The build itself will have pre and post activities that need to be executed as part of the build. To do this, Meister uses workflow technology to support the pre and post commands around the build.

The Meister Workflow Editor allows you to execute workflow activities in a particular order around your build. Developers can be given access to define these pre and post activities within Meister. This keeps a separation of duties between developers and your SCM administrators. The SCM administrator can restrict access to the CA SCM UDPs, but give full access to creating workflows within Meister. By providing the developers this level of control, they can configure their Continuous Integration activities independent from any process defined by the SCM team.

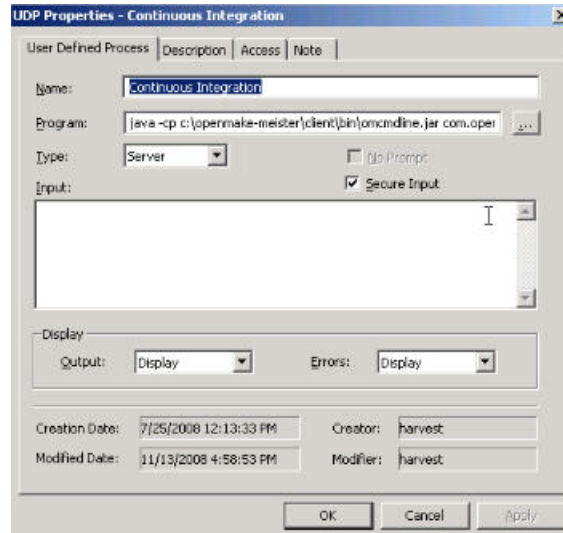


Common workflow activities defined by developers include "Updates from CA SCM", "Generating Build Control File", "Execute Meister Build", "Deploy to Server" and "Execute Test". Developers need to execute the build and immediately deploy the new objects to a development server so they can see their changes in action.

Setting up the Continuous Integration Server

To execute a continuous integration process, something must "trigger" the process to occur. The standard trigger is a "check-in" to CA SCM. This can be configured in two ways. A CA SCM UDP can be called upon a Check-in. The UDP would be configured to call a Meister "command line" Workflow. As soon as a CA SCM "Update" occurs, then the Continuous Integration process can be called.

With Meister, your continuous integration build can be performed incrementally, referred to as "build avoidance". This means that only the files that have been changed will be re-compiled and re-link with all dependent modules updated as part of the build. This means that there is little reason to define "wait" or "quiet periods". Because the build is not doing a full build, the incremental build called by the continuous integration process will be executed in as short as time as possible, sometimes less than 10 minutes.



Alternatively, Meister can be used to set up the continuous integration server process. To setup a continuous integration build with Meister developers use the `ContinuousBuild.pl` program provided by Meister. This program will initiate builds by monitoring the CS SCM repository to determine if files have been changed. If it is determined that files have changed, it will initiate a Meister Command Line Workflow running on the machine where the Continuous Integration process has been defined, most often a separate build server managed by the developers. The parameters that are required for the Continuous Build are *Sleep Time* and the *Configuration* file name. The configuration file defines how the Continuous Integration Server will act.

The following syntax is used for the `ContinuousBuild.pl` command:

```
perl -S ContinuousBuild.pl <seconds to sleep> <configuration file>
```

In the following example, CA SCM would be checked every 10 seconds. A build will be started if a change is found in the repository specified in the *Configuration* file.

```
perl -S ContinuousBuild.pl 10 c:\mojo\continuousbuild.cfg
```

The *Configuration* file has the following syntax:

Name=<Display Name for the SCM repository>

SCMcmd=<SCM Command used to determine file changes>

MD5ResultsFile=<Fully Qualified Path to store the MD5 Results in>

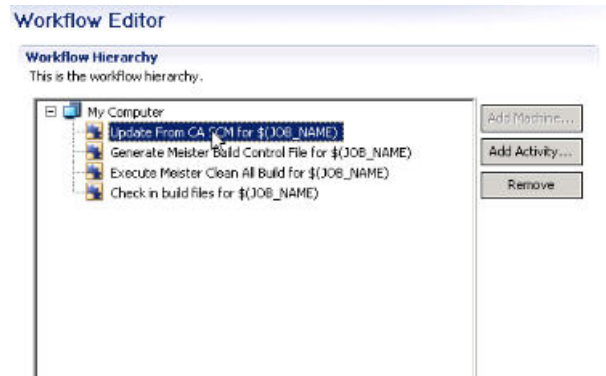
BuildCmd=<Command Used to start a build>

For a full description of how to implement the Continuous Integration features of Meister, go to <http://www.openmakesoftware.com/help/meister/> and view the node "Performing Continuous Integration".

Maintaining a Waterfall Promote Process

Once your developers are ready to move their source code forward and request a release, your standard approval process can be initiated. Your standard process for managing approvals can be established. It is highly recommended that a pre-production build be re-executed and called on the Promote process.

Using a similar method as the continuous integration process, Meister can be called to execute the build, including any pre or post steps that are needed to support the pre-production build. For example, Meister can manage the pre and post steps of "Updating from CA SCM", "Generating Meister Build Control file", "Executing a Meister Clean Build" and "Checking in the built files". Unlike the continuous integration build, the Pre-Production build can be configured to perform a "clean all" build, meaning all objects are re-created. It can also be configured to execute a Build Audit report showing a full audit trail of what files were used in the pre-production build.



Conclusion

Managing both an agile and waterfall environment need not be mutually exclusive. The key is to provide developers an environment that they can define and control, particularly around the continuous integration process. CA SCM and Meister can provide the tools needed to support both agile and waterfall practices, making users on both side of the infamous "wall" happy. Developers can use Meister's Build Services to automate the creation of build scripts directly from their IDEs, perform a check-in with CA SCM and trigger a continuous integration build. Meister can also support their pre-commit builds, and define pre and post commands around the continuous build without needing access to configure or define the CA SCM UDPs.

Embracing agile in your development state has many benefits for the developers. But preserving your production releases are just as critical to ensure that source code is properly managed, approved and your pre-production builds are audited. This more tightly controlled method provides the full traceability needed to ensure the health of your production environment and validate that the source code managed by CA SCM was indeed used by Meister to create your production executables.

Corporate Overview

OpenMake Software, the leader in application build and release management technology, is a provider of build to release software and consulting services designed to manage the risks associated with developing and deploying software applications. OpenMake Software specializes in the design and implementation of reliable and repeatable application build processes for Global 2000 organizations.

Additional Information

For more information on all the features Meister can bring to the CI build processes please visit <http://www.openmakesoftware.com/build-management-white-papers/> for additional whitepapers and information.

For a free continuous integration server with build to release workflow processing – go to www.openmakesoftware.com/get-mojo/

Company Overview

Meister has been revealing the "secret art" of the build to release process through a reusable, community-developed knowledgebase since 1995. Meister is used by Global 2000 customers worldwide, representing approximately 40,000 end users.

North American Headquarters

General Info: request-info@openmakesoftware.com

OpenMake Software
213 W. Institute Place, #404
Chicago, IL 60610
Ph: 800.359.8049 – 312.440.9545