



Implementing a Repeatable Build to Release Process for UCLA

A case study of
OpenMake® Meister's implementation at UCLA

OpenMake Software
213 W. Institute Place #404, Chicago
IL 60610
Tel: 800.359.8049
312.440.9545
Fax: 312.440.9543
Email: sales@OpenMakeSoftware.com

www.OpenMakeSoftware.com

UCLA's challenge was to standardize how builds and releases were processed, regardless of the development language used.

Summary

This case study provides an overview of UCLA's implementation of OpenMake Meister. It gives an overview of how Meister's Build Services and Build Methods were used to standardize the building and release of multi-language applications, including Visual Studio, Visual Studio .Net and Eclipse Java.

The Challenge of managing software applications written in multiple languages and IDEs

When the Regents of the University of California, Los Angeles campus reviewed the best way to standardize their build to release process, they chose OpenMake Software to do the job. Their challenge – standardizing the process of production readiness for 10 mission critical applications written by multiple teams using diverse languages including Microsoft Visual Studio, Microsoft Visual Studio .Net and Eclipse Java.

OpenMake Software's Meister was selected as the solution of choice because of its ability to manage the build engines of multiple development tools in a standardized repeatable method. It allowed a central production control team to manage the build and release of objects built using both Visual Studio and Eclipse Java, without needing to manage ad hoc, non-transparent scripts.

The Project Approach

The project was approached by dividing it into 4 major phases. The first phase defined how the build to release process would be implemented, regardless of team or development language. This planning defined how SCM tools would be leveraged, the definition of the workflow, and the roles and responsibilities of both the development teams and the production control teams.

Secondly, a pilot application written in Visual Studio was built. The Visual Studio pilot defined a standard build configuration for Visual Studio objects, for example, the use of production ready compile flags and the linking of production level third party libraries.

Similarly, a third phase which included the building of a pilot Java application accomplished similar goals. A standard process for building Java based applications using Eclipse was defined including the use of production level java compile options, and the use of production level java packages.

Once a standard process was defined, and tested using both Visual Studio and Java based applications, the process was ready to be rolled out to the remaining applications. The fourth and largest phase of the project involved building applications using both Visual Studio and Java, coordinating the building of shared objects between the applications, and standardizing when and where common modules were to be shared between the teams and the coordination of these applications in the march toward production release.

The Results – A Synchronized and Repeatable Build to Release Process

The UCLA team improved their overall build to release process by removing the dependency on a "point and click" IDE build; to a standardized build that could be executed automatically outside of the IDEs for both Visual Studio and Eclipse. This was accomplished by implementing standard OpenMake Meister Build Services that support the management of builds from all locations including inside the IDE running on the developer's machine, outside the IDE executing on a development team's build server or on a secured pre-production build machine. OpenMake Meister Build Methods removed the dependency on the "point and click" effort required by their developers to create the binaries. It provided a method to automatically synchronize the developer's individual IDEs build with the builds executing outside of the IDEs. With this process, both an automated and synchronized "team build" can now be performed by the developers and promoted to production control for the scheduled execution of "production ready" builds.

OpenMake Build Services mash up the Individuals IDE Build with the "team" or "pre-production" build executed outside of the IDE.

Build Services - Critical Components of the Process

The use of OpenMake Meister Build Services and Build Methods was critical in the success of the UCLA effort. Using Build Services and Build Methods, the UCLA team could standardize, reuse and repeat how similar objects were created. They no longer needed to depend on ad hoc, non reusable build scripts that could not be standardized across Visual Studio and Java.

A total of 10 OpenMake Projects were created and the use of 5 primary Build Methods were implemented including:

- Visual Basic DLL
- Visual Basic EXE
- .Net DLL
- .Net EXE
- Eclipse JAR

OpenMake Build Methods manage build engines such as MSBuild and Ant.

The implementation of OpenMake Meister at UCLA solved their core build to release issues which revolved around the lack of repeatability and transparency in the build to release process. By exposing application inter-dependencies and impact, synchronizing the developers individual build with the pre-production build, managing multi-language builds and

OpenMake Meister
standardized the
UCLA build to release
process from Visual
Studio to Eclipse.

improving build performance for better support of emergency release, OpenMake Meister helped UCLA achieve their ultimate goal of build to release repeatability and production readiness for mission critical applications. It also provided them the flexibility and consistency needed to manage their current and future application development activities, regardless of the development tools chosen by the development teams. With their current process managed by Openmake Meister, UCLA will be able to continue with a standard build to release practice even when new development tools and requirements are added to their mix.

For More Information about OpenMake Software and OpenMake Meister go to www.OpenMakeSoftware.com or contact us at 312.440.9545/800.359.8049