



How OpenMake Software is Different from Electric Cloud

OpenMake Meister is the only build management tool that actually manages builds for cross platforms and cross languages. From Make and MSBuild to Ant, Meister improves your build speeds and quality by managing at the build engine level. Meister provides you the power tools needed to address the core problem set of build management and create standardized and repeatable builds. We are not experts in Electric Cloud products, but we can give you a high level overview of their products in comparison to ours.

Competitive Features in Core Competency of Build Management

The following features review the core competency of build management. Electric Cloud is one of the only build solution vendors, other than OpenMake Software, that addresses the core function of the build process.

Cross Platform and Cross Language Support

Electric Cloud offers a variety of tools. Their flagship product Electric Accelerator is focused on managing at the build level, similar to Meister. It speeds up processing by managing dependencies and distributing the build across multiple machines. It was designed to improve the build engine processing for Gnumake and Nmake; therefore it does not handle cross languages such as java/Ant, or managed code based engines such as MSBuild.

In contrast, OpenMake Meister improves build processing for cross languages and cross platforms. It improves the quality and build speeds for Ant/Java, Visual Studio 6.0 (Nmake), Visual Studio .NET (DevEnv), Visual Studio 2005 and 2008 (MSBuild), Visual Basic, Eclipse, IBM RAD, Borland, C-Unix, C++, Gnumake, Linux, HP, Sun, AIX and z/OS. Meister is an enterprise based solution that goes far beyond improving builds for C and C++ applications that rely on a scripted Gnumake or Nmake process.

Project and Source Code Dependency Management

Electric Cloud's Electric Insight exposes all the critical source code dependency gathering that occurs when Electric Accelerator executes a build. Because it is based on Electric Accelerator it can only perform this critical feature on builds that rely on Gnumake or Nmake, which is primarily C and C++ applications. We believe the ability to manage and expose dependencies is a core component of a build management solution. It is the heart of a build.

OpenMake Meister supports Project and Source Code Dependency Management for all languages, including Java, Visual Studio .Net, Visual Studio 6.0, Visual Studio 2005, Visual Studio 2008, Visual Basic, IBM-RAD, Eclipse, C, C++, Assembler and COBOL. Because of Meister's cross language features, it can perform dependency management across languages, for example between a .Net and a Java application. In addition, managing dependencies at this deep level allows Meister to perform builds using "build avoidance". This means that objects that are "up to date" are not re-created every time a build is executed. This is the most efficient and accurate method of running builds, allowing build times to be reduced from hours to minutes. Yes, 10 minute builds are possible with Meister – our customers do them everyday.

Workflow Management and Job Scheduling

As more and more developers have begun to address problems associated to builds, they quickly come to the realization that they have many scripts that need to be executed in a particular order. This need has driven build solution companies to add job scheduling features to their product offerings.

Electric Cloud's Electric Commander is a workflow processing tool that performs basic job scheduling functionality with remote "agents" allowing steps in a workflow to execute across multiple machines. Developers design the scripts to be executed and Electric Commander executes the script in the order defined. The scripts can be written in any command line language such as Ant, Make or any shell command. As part of this scheduling and workflow processing, a continuous integration server component is added. These features are not unlike the free open source tools such as CruiseControl and free commercial tools such as OpenMake Mojo.

Workflow processing and Job Scheduling is a focus now for many companies looking at build management solutions. **Both OpenMake Mojo and OpenMake Meister offers these features including the continuous integration processing.**

What Electric Cloud Says About OpenMake Software

"OpenMake is a scriptless build tool and you would need to throw out your scripts and start again with Meister"

This is simply false. OpenMake Meister is not a scriptless build tool. You can implement Meister in the same way as you would Electric Commander using your existing build scripts and creating a workflow around those scripts, with scheduling, continuous integration and centralized logging. Meister also provides you with Build Services. Build Services allow your developers to write reusable build scripts, and substantially improve the quality and consistency of your builds. For example, these reusable build scripts can integrate directly to your IDEs allowing your IDE build and the builds running outside of your IDE to stay in sync. When our customers roll out Meister, they begin using their current scripts. As their build process matures, they begin developing Meister reusable build scripts reducing the number of build scripts from hundreds down to a handful. This is where your ROI begins to maximize and your build quality and speed really improves.

Meister goes beyond simple job scheduling functionality and allows you to address the core of the builds, at the lowest level, the build engine.

"OpenMake requires investment in professional services for implementation and significant training for developers to use this system."

Meister offers many benefits to developers - one main benefit being its integration with popular IDEs. Because of this developers can quickly learn to implement and navigate through the Meister features without learning a new user interface. We recognize that our competitors look for areas that they can use against us, and this happens to be one of those "grasping at straws" arguments.

The simple fact is that the opposite is true. When developers implement Meister, they can do it much faster than our consultants, because they know their applications. Most teams get the basic workflow processing working within the first day. When they begin using the build services, they can improve their build speed, quality and audit within a week resulting in build scripts that are NEVER out of date, even when refactoring has changed the build script logic.

"OpenMake Meister does not have a 3-Tier Architecture so it cannot support all the teams that need to access the system from all areas."

A three tiered architecture refers to a client a server and a data store. Meister has all three. We have a client component with a centrally managed knowledge base server running on Tomcatt, OC4J, or Websphere, with an XML based data store. This technology is what allows our customers to support large globally distributed teams and to run an infinite number of concurrent processes. We have customers that do just that. I believe what Electric Cloud is referring to is the use of our XML data store versus a database. A database is not required to provide global access, just a centralized data store with an efficient server processing.

"There is no concept of projects, so reuse and distribution of standard shared procedures is difficult."

This statement just shows a lack of knowledge about Meister. Meister's entire framework is based on reuse and each application has Projects. In terms of sharing procedures, our "workflows" are defined as "public" and "private". Meister allows you to setup groups and define who has access to workflows. You can also restrict access to Projects and dependency directories.