

Productivity Improvement Using Subversion, Netbeans and Maven 2

Abstract: *Software development is one of the most complex tasks in the world. Achieving quality productivity in this field requires knowledge about the right tools and their usage. This paper discusses three tools available for the Java development community which in the author's experience, improved productivity in his organization.*

Author: *Subhash Chandran S*

Last Updated: *2nd March 2006*



Subversion

Subversionⁱ is a popular OpenSource Software Configuration Management (SCM) tool. This was developed primarily as a replacement to the ubiquitous CVS. The best part of the tool is that the commands are similar to that of CVS, but the working more powerful and easier than CVS. There is only a minor learning curve.

Our team practices the best parts of Extreme Programming(XP). And a tool like subversion helps us convert the principles into practices.

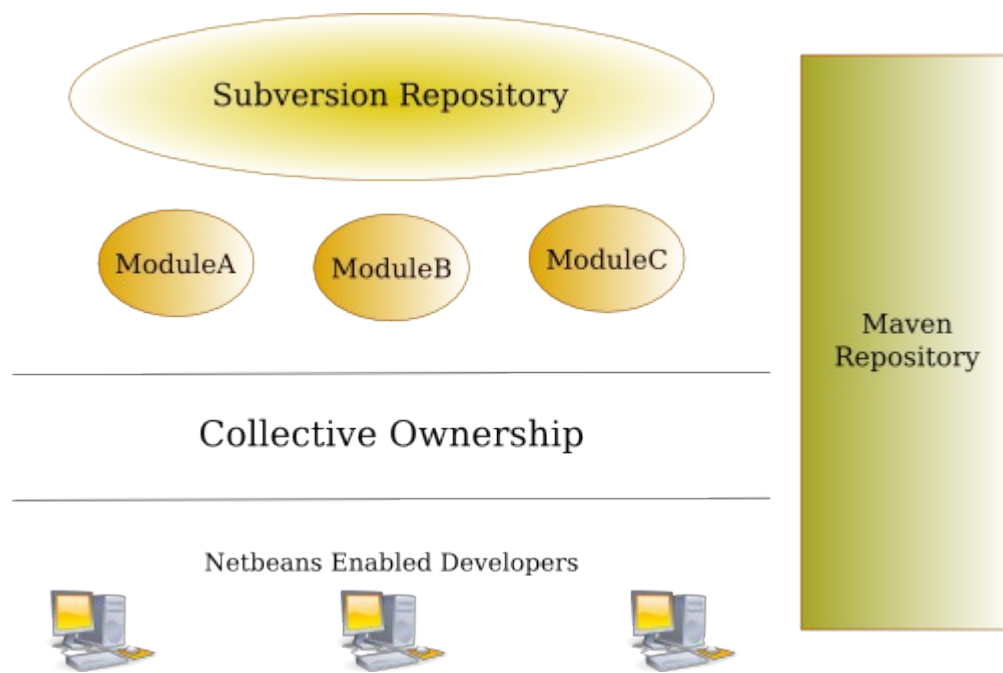
Netbeans

Netbeansⁱⁱ is one of the oldest IDE for Javaⁱⁱⁱ. This software was notoriously known to be a memory hog. But since version 4, this has changed. This is a worthy contender to tools like Eclipse and IntelliJ Idea. It has support for Extreme Programming (XP) concepts like refactoring. This is an OpenSource Software.

Maven2

“Maven^{iv} is a software project management and comprehension tool.”^v Basically it is a build tool. But it has modules for release management, documentation management, site generation, quality reports. To use all these features, one has to just describe the project using the POM, or the Project Object Model. This is basically a XML file describing the project. Using well-defined project folder structure, and naming conventions, the developer is relieved of knowing or defining low level details like jar filename, or source directory.

Their Combined Power...



We follow a aggressive commit policy in our team. We commit often, and build often. It is unfortunate that many people regard a SCM program as a “official” deployment tool, where only a “perfect” code gets committed.

In XP, we have the concept called Collective Ownership. In the words of Kent Beck, it means: *Anybody who sees an opportunity to add value to any portion of the code is required to do so at any time.*^{vi}

This is one practice in XP which we follow with religious vigor. The principle of Collective Ownership weights upon us the responsibility to write the right code, the readable way. It also ensures that small bugs get fixed faster, minor integration problems gets resolved immediately. The awareness that our code is available for others for modification builds an internal quality consciousness.

Subversion makes available the source code of the team to all developers. So the next logical step for any member in the team is to checkout the source and – if need be – to modify and build it. Previously we were using Ant for the same. We had spent much time writing redundant Ant build.xml, and guessing what naming conventions and folder structures we were supposed to use in the process! Also having Ant meant having dependency Jars cluttered in multiple locations in our subversion repository, with most of their version information lost. With the advent of Maven 2 all this has changed. We now just describe our project in our POM. We no longer spend time copy/pasting redundant code from one *build.xml* to another. We no longer publish Jar naming standard, project folder structure, and other low level details. The magic of Maven happens by defining a rigorous directory structure and having a wonderful dependency management module.

Project dependencies are published in public repository. So for our project we created our own internal public repository. The built modules are updated to this repository. Modules dependent on them automatically downloads these updated Jars and builds the project.

Maven has plugins for creating deployable WARs, EARs and other J2EE modules. It has an excellent release management tool. Creating a release is just issuing a maven command – it updates the SCM repository by tagging, updates the POM with the version number, copies the distribution to the release server.

Netbeans as an IDE has improved over the ages. It has excellent support for refactoring, another wonderful practice followed by XP community with enthusiasm. Netbeans has been my choice for some time, because of the SUN® tag, and the pure Java codebase – and I personally find it better designed when compared to Eclipse. And Netbeans has a Maven plugin, MevenIDE^{vii} (note the “e” instead of “a” after “M”). One feature of the plugin which impressed us – we can open any Maven project, without any configuration. That is no project classpath setup, no compiler version setup, no source directory setup. None! Just open the project!

So with MevenIDE installed one can imagine the power of Netbeans. And combine it with Maven and Subversion, we have a killer combination! The latest code of any module is available on the subversion repository, any person with access can checkout the code, modify it – using the power of the modern IDE Netbeans, build the same using Maven and release.

Did anyone say we are *not* living in the perfect world?

- i <http://subversion.tigris.org/>
- ii <http://www.netbeans.org/>
- iii <http://www.netbeans.org/about/history.html>
- iv <http://maven.apache.org/>
- v As described in the Maven site (as on 2nd March 2006): <http://maven.apache.org/>
- vi Kent Beck, *Extreme Programming Explained: Embrace Change*, Addison Wesley, 2000.
- vii <http://docs.codehaus.org/display/MEVENIDE/MevenideNetbeans2.0>