Barriers to Building Eclipse with Maven

Introduction

Over the course of the past year or so, Jason van Zyl, Jeff McAffer (IBM / Equinox), Kim Moir (IBM / Platform), and John Casey (among others) have had a series of conversations about supporting Eclipse builds through Maven. Since both Maven and Eclipse have somewhat similar ideas about distribution of plugins and dependency components, and because Eclipse is a large and diverse community who could benefit from build standardization, it seems natural to pursue the goal of having Eclipse itself build using Maven.

Specific Barriers to Entry

The following are a list of issues relayed to me (John) from Jeff and company, by way of Kim. These are seen as key features that Maven will have to support in order for Eclipse's various projects (particularly RCP/platform/equinox) to build:

duplicated metadata

Maven must support reading project information directly from plugins.xml, features.xml, and manifests in order to avoid the need to maintain manually two pieces of metadata per eclipse project.

artifact resolution using running Eclipse/basebuilder instance

During a normal Eclipse PDE build, plugins which are already in the IDE instance are used as dependencies for projects as needed. Maven should support resolving against installed plugins, features, etc. of an existing build environment.

dependency OS/architecture

Some dependencies are platform-specific, such as SWT. Maven would have to allow expression of this, and handle it correctly.

NOTE: This might be feasible using profiles...need to checkout the RCP build to try it.

resolve feature sources from SCM and build-on-demand

In some builds, dependency features are resolved in source form from SCM and built prior to the main build running.

NOTE: We've done something like this for c-builds, and may be able to make it a little more elegant using Buckminster's materialization code.

building system-dependent fragments

When building certain projects, parts of the project are "fragmented" off into platform-specific sub-project builds that only run if the platform is correct. Maven must support such conditional sub-project builds to support Eclipse.

NOTE: This sounds like a set of profiles that inject modules, much like the profiles in the Maestro build, but with different activation.

multi-JVm support at build-time

Some builds require that certain sub-builds be constructed using one JVM, while other sub-builds use a different JVM. Since these must build from a single command-line, Maven must support the notion of JVM-per-project configuration to support this.

NOTE: We might be able to work around this initially by using maven-invoker.

filtering of feature.xml

feature.xml, one of the proposed sources of POM information, may in some cases need to be filtered to set versions based on some external property. I'm assuming that these filtered values would then make their way into the live build process...

jars within jars

Many plugin builds involve bundling dependency jars within the plugin jar itself.

NOTE: This should be a breeze with the assembly plugin, but we may want to make a descriptorRef available via a plugin-dependency or something to support this better.

resolve binary plugins

Two things related to this:

1. same as artifact resolution from existing instances, mentioned above
2. resolving plugins from existing Eclipse update sites...which would mean having an artifact resolver that could understand site.xml files.