Eclipse Planning Input

Ideas and requests for future Eclipse versions that would facilitate a tight integration with Gradle. Collected as input for Lars Vogel who is going to participate in high-level planning for Eclipse IDE.

- JDT should support multiple classpath roots per project. Each classpath root must have its own configuration of dependencies. Some configurations extend other configurations.
- Improve WTP through better documentation and with hooks that 3rd party plugins can reliably integrate with. Make it more consistent with other (pure Java) projects.
- Better support for hierarchical project structures.
- Facilitate external build of Eclipse RCP applications/plugins.
- Communication of supported versions/requirement/EOL policies.

Classpath

An IntelliJ module distinguishes between source and test classpath roots, which is far more powerful than a single classpath root per project as modeled in Eclipse. Currently, it is not possible to properly map a Gradle project to a single Eclipse project.

In addition, IntelliJ has the general concept of *compile*, *runtime*, *test, and provided* scopes used for project and external dependencies. In Eclipse, that can be partially modeled with launch configurations and classpath containers, but there are limitations to that.

NetBeans is even more flexible because the classpath is decoupled from a project and it is possible to have a project with separate classpaths for sources, tests, and optionally for other things like integration tests.

We favor the NetBeans flexibility since it matches the flexibility of Gradle. Gradle allows custom configurations, where each configuration comes with its own source sets, its own dependencies, and usually with inheritance of dependencies from other configurations.

See also https://bugs.eclipse.org/bugs/show_bug.cgi?id=105372 [buildpath] JDT should support multiple classpaths per project

WTP vs JavaSE

In Eclipse, it is a very different task to model basic Java SE projects and to model Java SE/EE projects once WTP becomes involved. The project classpath for a simple Java library is

different when everything is just Java SE and when it is a dependency of a WAR (or another Java EE archive). Using classpath containers with these projects is likely a problem.
Facets are a good idea but they are limited to WTP.

Hierarchical projects

It is possible to do this now at the cost of resource filters. Maybe it is something we can live with. We mentioned this in previous interviews.

Eclipse PDE/Building

It is hard to set up a project for Eclipse plugin development that works in Eclipse/PDE *and* with a build tool of your choice. Currently, you can choose between Ant, Maven/Tycho, and Gradle/Bnd. But, each one of these has its problems in the context of Eclipse/PDE.

Eclipse/PDE has various requirements that often contradict common practices of the different build systems. As a consequence, a setup that allows to develop with Eclipse/PDE and that is also compatible with an automated build is difficult to achieve, considering the output must be the same. Sometimes PDE goes against Eclipse itself (missing support for linked resources). https://bugs.eclipse.org/bugs/show_bug.cgi?id=153023 'Expose flexible bundle root location in bundle creation wizards' is another example of a bug report showing the limitations.

Versions/Requirements

There is a cost related to supporting multiple Eclipse versions in the Gradle plugin. We are not sure if we can really ignore Eclipse 3.x and focus on Eclipse 4.x. Easier development as in 4.x can help us to provide a better user experience. But, there still seem to be many software teams that to use Eclipse 3.x or commercial tools based on it.

Having some statistics on the Eclipse usages across versions and some information about the end of life policies would be very beneficial to decide on what Eclipse versions to support.