Position Paper: A Desire for a Point of Integration

Ed Warnicke - Cisco Systems

October 21, 2005

1 Introduction

We have a need for a modern platform on which to standardize development efforts across the lifecycle of a large number of software products. In order to achieve this end, it is necessary to be able to accommodate the diversity of existing tools in many tool chains, and the existing skill set of the people performing tasks in product lifecycles.

What this effectively means is that support is needed for developing in C, C++, Java, Tcl, Perl, Python, PHP, and Ruby in order to provide support for existing development processes.

Because many of the most valuable tools bubble up from the people closest to a task, developers need to be able to develop extensions to Eclipse in the languages they are comfortable with. For many of developers this is not Java.

2 Language Support

Currently there is excellent support of Java in Eclipse. Support for C/C++ is also good (although there are scaling issues for very large projects). There exists support of Perl (http://http://e-p-i-c.sourceforge.net/), Python (http://rubyeclipse.sourceforge.net/), PHP (http://www.phpeclipse.de/), of varying (but improving) quality. There is no known Eclipse support for TCL.

3 Extending Eclipse in non-Java Languages

We have a tremendous number of developers who are much more comfortable in TCL, Perl, or Python than in Java. Being able to extend Eclipse in those languages would enable a great deal of innovation at the edge.

Looking at existing JVM based implementations of TCL (jacl), Python (jython), and Ruby (JRuby) it would seem possible to provide JVM hosted extension in those languages. Many other languages can be embedded using C, and thus could potentially be accessed via JNI. Obviously existing JAVA class structures would need to be respected.

It would seem that the leading impediment to being able to use non-Java language implementations of plugins for Eclipse center around the use of the constructor pattern for object creation. Because most extension points expect plugins providing extensions to at some point specify a concrete class that satisfies some interface, and because that concrete class specification is intrinsically the specification of a Java class, it becomes very difficult to specify an object implemented in a non-Java language.

One approach might be to allow for the specification of an AbstractFactory with parameters rather than specifying a concrete class. In this way a class like PythonAbstractFactory could be written which, when provided with appropriate initialization parameters could interpret a python module and return an object from that interpreters namespace. In this way a small number of language specific AbstractFactory classes could allow a user to write all of their plugin code in the language of their choice, and not have to write any Java. This same tactic could allow for possible bindings to non-JVM hostable interpreters using JNI.