Compiled Language Debug as Extensions to Platform Debug vs. Parallel Hierarchy

Position Paper for Eclipse Languages Workshop

Martin Imrisek

October 13, 2005

The debugger implementer faces a dilemma of choice when implementing debug support for their favourite compiled language today – Does one implement a debugger to the Eclipse platform APIs or to CDT's C Debugger Interface (CDI) APIs?

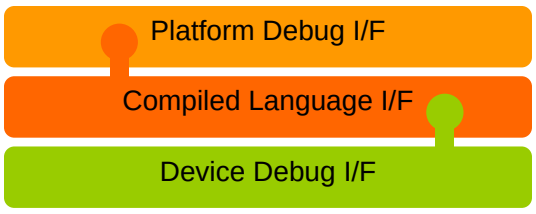
The Platform and CDI API and debug models are very similar and in fact they parallel each other closely as a result of the history of the development and evolution of the platform APIs and CDT. The CDI has the concepts needed to implement a debugger for a compiled language, concepts such as instruction stepping, dis-assembly for example, but the CDT itself is geared towards C/C++ support with little granularity for integrating support for other languages.

So as debugger developers we have this dilemma of choice – choose to implement to the Platform and one does not get needed support for instruction stepping and code disassembly. Choose CDI and one does not get to easily integrate the other aspects of language support such as language specific editors.

At this time I don't think it makes sense to carry these parallel debug hierarchies forward or to encourage other projects to create parallel hierarchies to the platform or CDT to add extensions that do not fit into either one.

As implementers of debuggers we are interested in a hierarchy of debug APIs that provide core abstractions (which the platform does well) with compiled language debug capabilities being extensions on the core APIs. The compiled language debug APIs , and likely some abstract implementation as well, could be shared amongst the CDT and other compiled language debug projects.
As we are currently targeting device debugging with the CDT we similarly see that the compiled language debug APIs could be extended with support for features that are specific to device debugging which is part of the discussions in the DSDP project.

In the scenario of this hierarchy CDI would cease to exist and instead CDT debuggers would be written to the Compiled Language interfaces.



There are probably concepts and capabilities in the CDI interface that should be migrated to the platform so further analysis is needed to identify these precisely.

An example of an important concept that is currently handled abstractly in CDI but not in the platform is the notion of an 'Address'. The platform's notion of an address is a integer or a BigInteger, while CDI's is an abstraction – an opaque object. This is significant because CDI is being used to debug platforms with paged memory architectures and therefore cannot represent addresses as a single integer.

Moving this concept of an address as an opaque object is required to be able to get a hierarchy of interfaces as in the diagram – however this specific change would result in significant rippling through the platform, JDT, PDE and other projects that rely on the platform debug APIs.