ACM SIGAda 2007 Conference Workshop Report on Hibachi - the Eclipse Ada Development Toolset

Thomas Grosman
grosman@aonix.fr

Understanding Eclipse

Eclipse is more than just a development environment (IDE) for Java, now used for a variety of languages including Ada and other tool sets. Eclipse is also an opensource community and a large ecosystem focused around an extensible open framework of services, tools and runtimes for building, deploying and managing software across the lifecycle.

The Eclipse platform is written in Java, freely available, extensible via an extension point/plugin mechanism with well-defined APIs. Extensions may be open source, proprietary, or a combination of both. Eclipse includes many useful plugins "out of the box", such as JDT- a Java IDE, as well as many best-of-breed functionalities such as a full-feature integrated CVS client. Eclipse projects follow Eclipse development methodology (transparency, meritocracy, "code talks"). Projects within the Eclipse Management Organization, like Hibachi, are open source under the Eclipse Public License (EPL) and use a public Bugzilla server for managing most aspects of the project (bugs, patches, enhancements, development plan, release plan, etc.)

An Eclipse project *community* is made up of users, testers, contributors and committers. Eclipse.org consists of over 60 open source projects, with over 800 committers. Eclipse is used for such applications as Enterprise Development, Embedded and Device Development, Rich Client Platform, Application Frameworks, Application Lifecycle Management (ALM) and Language IDEs. Examples of Eclipse open source projects include Hibachi (Ada development), JDT (Java development), CDT (C/C++ development), Photran (FORTRAN), UML2, PDE (Plugin Development Environment), Subversive (Configuration Management), TPTP (Testing/Profiling Tools) and DSDP (Embedded Development).

Committers, who are devoted developers, have write access to the code repository, are nominated and elected by current committers, are responsible for project architecture and direction and are responsive to community needs. *Adopters* use services and extension points to work with their own “add-on”, influence Hibachi architecture and direction and may be committers or contributors. *Users* are end users and may not even be aware that they are using Eclipse due to RCP or branding.

The Eclipse *ecosystem* consists of the Eclipse Foundation, the Eclipse Management Organization, Eclipse Members (Strategic, Add-in Provider or Associate), Eclipse projects, adopter/integrators and users. Eclipse projects are developed under a well-defined but flexible development process and lifecycle, following certain principles ("just enough process"). These principles include:

1. A committer may not, through action or inaction, violate IP cleanliness
2. A committer may not, through action or inaction, disenfranchise contributors
3. A committer may not, through action or inaction, surprise the membership

Projects mature through the following lifecycle states: Pre-Proposal, Proposal, Incubation, Mature, Top-Level, or Archived (frozen). Graduation from one status to another is the result of community reviews of the project.

Why Hibachi?

Hibachi is a full-featured IDE for developing Ada native and embedded applications, providing an open framework for the integration and use of other tools used during the lifecycle of large-scale Ada application development. These tools include but are not limited to Static Analysis, Modeling, Testing and Verification, Performance Analysis, Documentation, Refactoring and Configuration Management. Hibachi is independent of the underlying Ada compiler technology. Hibachi's design goal is to be architected in such a way as to allow integrators the possibility to extend or replace the functionality it provides.

The rationale for such a vendor neutral open source extensible Ada development environment includes customer/market demand, an understanding by vendors of their value added proposition, strong competition from other language IDEs, the ability to take advantage of the force multiplier of a community larger than a single vendors, the attraction to more third party integrators, as well as the ability to simplify synchronizing updates.

Hibachi introduces a new paradigm in the Ada industry, that of "coopetition" (collaboration on a product from industry competitors).

Hibachi Status

Hibachi was approved by the EMO as an official project on September 24, 2007. It is based on the AonixADT technology, currently supports ObjectAda and GNAT (Nov 2007), and is available under EPL.

The initial Hibachi committers include developers from Aonix, AdaCore, DDC-I and CohesionForce. Organizations interested in participation as contributors or adopters include Praxis High Integrity Systems, OCSystems, Green Hills Software, the Institute for Software (Switzerland), the Military University of Technology (Poland), and the GnuAda and MacAda groups. Hibachi will initially be released on Windows, Linux and SPARC Solaris. The Eclipse project infrastructure is in place and being developed. The infrastructure includes the project home page (**www.eclipse.org/hibachi**), source downloads (**wiki.eclipse.org/CVS_Howto**) and access (**http://dev.eclipse.org/viewcvs/index.cgi/org.eclipse.hibachi/?root=Tools_Project**), newsgroup (**eclipse.tools.hibachi**), development mailing list (**hibachi-dev@eclipse.org**), bug and patch tracking (**https://bugs.eclipse.org**) and project wiki (**http://wiki.eclipse.org/index.php/Hibachi**).

The first downloadable release of Hibachi, scheduled to be available in Q4 2007(v0.5), will include the full AonixADT code base. It will be available on Windows, Linux and Solaris under Europa (Eclipse 3.3), and include support for ObjectAda and GNAT.

Initial Hibachi functionality includes support for a project explorer and builder, an Ada object navigator, Ada configuration control, wizards, multiple Ada toolchains (ObjectAda and GNAT), software update manager, language block completion, structure highlighting, object navigation (declaration, definition, package specification, package body, where referenced), source code completion, formatting (syntactic, semantic, indentation, comment toggling), application launching, searchable and dynamic help, file comparison (local history, other files) source control (CVS, Subversion, Clearcase, …) textual and semantic searching, bookmarks, problems (with source navigation), tasks, and multi-language debugging including task inspection (stack, locals, suspend/kill), attach to running process, variable/expression/memory/register inspection and modification, and breakpoints (conditional, disable, action on break).

David Phillips of CohesionForce and Greg Gicca of AdaCore, both companies which have committed resource to Hibachi, gave short presentations of their organization's view of Hibachi and contribution to the project.

Future Direction

The Hibachi Roadmap includes an open API for additional toolchains and support for additional toolchains, (v0.7); CDT technology rapprochement, more APIs and more stable APIs (v0.7); Ada 2005 support and DSDP support for Eclipse 3.4 (v1.0).

Future features will include unit testing, "where used" reimplementation, Ada 2005 support, refactoring (AST development), cheat sheets, reformatter (in Java), Mouse over information improvements, generic ARM lookup, more wizards (builder, stub generation), code coverage extension points, code folding compiler error hints, automatic "With" insertion, unit dependency browser, and call tree browser.