Product update: on the road to Ada05

Sigada 2003
San Diego 12-10-03
Recent Developments in GNAT technology

► New targets
► New configurations
► New tools
► New language!
New Targets

► GNAT for the Rockwell AARM chip
  • Completed
  • Does not use GCC back-end

► GNAT for Itanium / Gnu-linux
  • Ready for test drive

► Gnat for Itanium / VMS
  • In progress
  • Indispensable to complete VMS port
New Configurations

► Ravenscar
  • Profile and light-weight library

► AE653

► Configurable run-time
  • From bare board to full Ada

► All new releases based on GCC 3
New Tools

► The projects facility

► Mixed language compilation

► GPS

► Interfacing to Eclipse
GNAT Project Facility

- Plain ASCII Text Control Files
- Integrated with gnatmake, gnatfind, gnatpp & gps
- Controls entire compiler tool chain
- Supports Mixed-Language Applications
- Supports Modular Hierarchical Designs
  - Uses an Ada-like Package Model – “withs” bring in other projects
  - Subsystems in APEX, Subprojects in MULTI
- Supports Project Extensions and Project Inheritance
- Supports Alternate Views & Builds
  - Configuration variables control various kinds of builds
- Supports Alternate Naming Conventions
- Supports building objects, libraries and executables
Gpr2make is a tool which converts gnat project files into makefiles:

- Proj.gpr -> Makefile.proj
- This makefile
  - Makes use of Makefile.prolog & Makefile.generic
  - Will call gnatmake to compile all ada units
  - Will use make rules to invoke compilers and other tools on units using other languages
  - Uses special features of GNU make
  - Guarantees exactly correct dependencies (using information generated by G++ compiler).
GPS: The GNAT Programming System

- Developer-friendly
- Modern GUI
- All tools tightly integrated
- Multi-platform: Linux, Unix, Windows
- Multi-language: Ada, C, C++
- Native & cross development
- Extensible
- Open technology
GPS: The Programmers' Companion

▶ Developer-friendly
  • Intuitive interface
  • Easy to learn and use
  • Allows to develop, build, maintain large complex systems

▶ Modern GUI
  • Same look’n’feel for all tools and on all platforms
  • Can plug your favorite look’n’feel (e.g. Windows)

▶ All software-engineering tools are tightly integrated
  • Tools interact in a coordinated fashion
  • Launch all tools from the same interface
  • User’s extensions tightly integrated
GPS Tools

- Language sensitive editor
- Source code navigation
- Source-level debugger
- Project, dependency & call graphs
- Version control (RCS, CVS, ClearCase, ...)
- Project & configuration manager

- Class, type & instantiation graphs
- Unit testing
- Reformatting tool
- Documentation generator
- Metrics
- Refactoring tool

- GUI builder
- Real-time event monitoring
GPS & Heterogeneous Environments

While developing for remote targets (compile, debug, …)

Run GPS on your local machines

While developing for remote targets (compile, debug, …)
GPS is an Open Environment

► All tools also available from the command line
  • You can use them in text-only mode (e.g. for use in scripts)
  • Call them from your own tools

► All formats use plain text
  • E.g. the project file

► Ability to plug in your own tools
GPS is Customizable & Extensible

► You can change key bindings, shortcuts, font colors, etc

► You can add a menu or a button to invoke a new tool

► You can extend the editor to understand a new language

► Easy to plug-in 3rd party tools

► The ultimate extensibility: You can plug your modules in GPS
GPS Architecture

LOC

200,000 Ada
- 90,000 GNSA
- 50,000 GVD
- 60,000 GPS

100,000 Ada

400,000 C

GtkAda
Thick Ada binding

Gtk
toolkit for creating graphical user interfaces

X11 or Win32
graphic libraries

Linux, Unix, Windows
Integration of GPS into Eclipse

- Simple invocation of GPS executing independently
- Invokes GPS on selected file from within Eclipse
- Uses built-in Eclipse support for “external tools”
  - Dedicated button indicates GPS
- Full GPS project support as long as:
  - GPS project file name is same as the Eclipse project name
  - GPS project file is located in Eclipse project directory
Dedicated buttons
with Identities;
use Identities;

package Cognicenti is

  task type Philosopher is
    entry Identify( Me : in Identifier );
  end Philosopher;

  Philosophers : array( Identifier ) of Philosopher;

  procedure Shut_Down;

end Cognicenti;
Schedules/Capabilities (GPS)

► GPS 1.4.0

- Full support for Ada, C and C++ editing
- Full support for version control management
  - CVS
  - Clearcase (static views)
- Full support for Ada intelligent browsing
- Extensive support for C and C++ browsing
  - Some limitations for templates, name spaces, macros
- Full build capabilities for Ada, C, C++ builds
- Full support for Ada and C debugging
- Extensive support for C++ debugging
  - Some limitations for templates, name spaces, macros
Schedules/Capabilities (GPS)

► GPS 1.4.0

► Released November, 2003

• Windows NT
• Windows 2000
• Windows XP
• Solaris
• GNU/Linux x86
  - Redhat Enterprise edition
  - Novell/SUSE
• Other targets to follow
  - HPUX
  - True 64 (DEC Unix)
  - Other targets as needed
Schedules/Capabilities (GCC)

► GNAT 5.01a available now for nearly all targets
  • Supports Ada and C fully, can interface to C++
► GNAT 5.02a scheduled for early Q1` 2004
  • C++ compilers added for
    - Solaris
    - GNU/Linux x86
    - Windows 2000, NT, XP
► GNAT 5.02a1 scheduled for Q2 2004
  • Full support for C++ offered on selected targets
  • C++ capability added on additional targets
Towards Ada05

► Implementation of major Ada05 enhancements is in progress

► Driven by:
  • Customer demand
  • Resource planning
  • Implementor's itch
Object-Oriented facilities

► Cyclic type structures
  • Fully implemented (*limited_with* clauses)
  • Years of design, days of implementation effort
  • Supercedes *with_type* clauses

► Interfaces
  • Under study, proposal still fluid, implementation effort probably non-trivial

► Prefix notation
  • No implementation difficulties, no enthusiasm

► Accidental overriding
  • Early implementation discarded, new one underway

► Extensions of protected / task types
  • Unlikely (for now)
Access types

► Anonymous access types
  • In progress

► Downward closures and anonymous access to subprograms
  • Preliminary design
General purpose capabilities

► Aggregates for limited types
  • Fully implemented
  • Constructors?

► Pragma Unsuppress
  • Early GNAT pragma
  • Implementation may need adjusting

► Private with_clauses
  • Not yet
Real time facilities

- Ravenscar
  - Fully supported
- Dynamic ceiling priorities
  - Implemented, re-implemented
- Execution-time clocks
  - No time budget yet
Interfacing

► Unchecked-union
  • Minimalist implementation available for years
  • Maximal version needs work
► Directory operations
  • gnat-directory_operations available
► Vector and Matrix operations
► Container libraries
  • Implementors welcome!