Introduction to CDT

Sebastien Marineau-Mes
CDT Project Lead
Outline

- What is CDT
- CDT project: goals and challenges
- CDT architecture
- Feature set – CDT 2.1
- Roadmap – CDT 3.0 and beyond
- Integrating with CDT
- Getting involved
CDT – A Bit of History

- CDT – C Development Tooling

- Project launched July 2002
  - Provide C/C++ development under Eclipse
  - Integrate with existing C/C++ command-line tools (compiler, debug etc)
  - Built as extensible and replaceable building blocks

- Milestones
  - CDT 1.0 Dec. 2002
  - CDT 1.1 (May 2003)
  - CDT 1.2 (Oct 2003)
  - CDT 2.0 (June 2004 - sync with Eclipse 3.0)
  - CDT 2.1 (Dec 2004)

- CDT committers
  - QNX, IBM/Rational, Intel
  - 35+ man-years of effort
CDT Momentum - Where we were at EclipseCon 2004

- Downloads
  - Enthusiasts
  - Mostly Windows and Linux
- Adoption in commercial products
  - QNX Momentics
  - IBM WSDD
  - Timesys Timestorm
  - Tensilica Xtensa Xplorer
  - Redhat Enterprise Linux
  - Montavista DevRocket
- IDE Prototypes
  - Altera
  - PalmSource
  - Intel
  - Rockwell Collins

- Participating companies
  - Timesys
  - Tensilica
  - Redhat
  - Montavista
  - Intel
  - Rockwell Collins
  - Real-time Innovations
  - Altera
  - PalmSource
  - Ericksson
  - Nortel
  - And other…
CDT Momentum - Where we are today

- Commercial products
  - QNX Momentics
  - IBM WSDD and Rational Sw Architect
  - Timesys Timestorm
  - Tensilica Xtensa Xplorer
  - Redhat Enterprise Linux
  - Montavista DevRocket
  - PalmOS Developer Suite
  - Intel C++ Compiler
  - TI Code Composer Essentials
  - HP Remote Development
  - Altera NIOS II IDE
  - Xilinx Platform Studio
  - ...

- Prototypes
  - Rockwell Collins
  - Symbian

- Participating companies
  - IBM
  - Intel
  - QNX
  - Timesys
  - Tensilica
  - Redhat
  - Montavista
  - WindRiver
  - Altera
  - Rockwell Collins
  - Altera
  - PalmSource
  - Ericksson
  - Nortel
  - Cisco
  - And others...

- Google Stat
  - Google “Eclipse CDT” return 20000+ hits
CDT project goals

- First-class framework for C/C++ tooling in Eclipse
  - Platform and purpose-neutral framework to support variety of development scenarios
  - Better JDT than JDT 😊
- Extensible and Interoperable
  - Provide powerful base functionality and allow extending/replacing features
  - Well-defined APIs for interoperable extensibility
  - CDT common integration point for all C/C++ tooling
- Cooperative
  - Pooling of resources for base C/C++ tools components
  - Well-defined “value-add” from contributing companies
CDT Platforms

- Desktop/Server
  - Linux, HP/UX, Windows, OS/X, Solaris, AIX
  - X86, PPC, PA-RISC, IA64, Sparc
  - Generally using a GNU toolchain
- Traditional Embedded
  - QNX, Linux, VxWorks
  - Host-target paradigm
  - CDT as integration point of embedded tooling
- Deeply Embedded
  - Variety of MCU, DSP and CPUs
  - SW/HW co-design, soft cores, FPGA
  - Interest in C, assembly; very simulator-centric

Off-the-shelf toolchain integration
- Default CDT implementations target GNU
- gcc most widely used for build/compile
- gdb debugger widely available
- Provide additional integration hooks for other toolchains
  - Optimized compilers, Win32 native debugger
Building a C/C++ IDE – it’s not easy!

- No “control” over back-end tools
  - Compiler, debugger, toolchains, build system all external
- Lots of C/C++ “legacy” to handle
  - Language extensions, build infrastructure
- C/C++ language challenges
  - Parsing challenges
  - Language variants
  - Complexity of C++
  - The Evil preprocessor
CDT Architecture

Eclipse platform

Workbench
  JFace
  SWT

Debug

Team

Workspace

Platform Runtime

C/ C++ Debug
  C/ C++ Debug Perspective
  Memory, Regs, Asm views
  CDI Interfaces, GDB/MI Plug-in

CDT Launch
  Launch Run/Debug session
  Glue between Core and Debug

CDT Build
  Project builds
  Make and Managed Make

CDT Core
  C/ C++ Editor
  Parsing, Outline, Indexing
  Code completion
  Search
CDT Core Architecture

```
org.eclipse.managedbuilder.ui
Managed Make Project Wizards
Preferences/Project Properties pages

org.eclipse.core
org.eclipse.make.core
Make Builder
Preferences/Project Properties pages
Make targets View

org.eclipse.managedbuilder.core
Managed Make Builder
Build Model

org.eclipse.cdt.ui
org.eclipse.cdt.core
C/C++ Project Navigator
C/C++ Editor
Build Model
Indexer
Output/Error Parsers
Binary Parsers

org.eclipse.make.ui
Standard Make Project Wizards
Preferences/Project Properties pages
Makefile Editor

org.eclipse.cdt.core
C/C++ Code Model
Standard C/C++ UI Pages

org.eclipse.make.core
Make Builder
Scanner Info Discovery

org.eclipse.make.ui
Standard Make Project Wizards
Preferences/Project Properties pages
Make targets View

org.eclipse.make.core
Make Builder
Scanner Info Discovery

org.eclipse.cdt.core
C/C++ Code Model
Standard C/C++ UI Pages

org.eclipse.cdt.ui
org.eclipse.core
C/C++ Project Navigator
C/C++ Editor
Build Model
Indexer
Output/Error Parsers
Binary Parsers

CDT Core/UI & CDT Build
```
CDT Debug Architecture

- `org.eclipse.cdt.launch`
  - Launch Tabs
  - Local Launch Delegates

- `org.eclipse.cdt.debug.ui`
  - CDT Debug Views
  - CDT Debug Actions

- `org.eclipse.cdt.debug.core`
  - CDT/Eclipse Core Debug Model

- `org.eclipse.cdt.debug.mi.core`
  - GDB Launching
  - GDB MI Interface

- `org.eclipse.cdt.debug.mi.core`
  - GDB Launch Tab
CDT Core Features

- Editor
  - C/C++ syntax highlighting
  - Code completion
  - Code folding
  - Hover help
- Search, outline, index
  - Parser extracts C/C++ elements from source files in project
  - Information used for search, outline, code completion, index, refactor
- Warning/error marker integration with build systems
  - Build tool output parsed to generate markers

Support for C++ Development

- C++ hierarchy
- C++ class browser
- Class creation wizards
- API and extension points to allow extensibility
CDT Build Features

- Standard Make
  - Re-uses existing makefiles
  - Simple integration with arbitrary build systems
  - Parsing of toolchain output to generate error markers

- Managed Make
  - Manages compiles and toolchain directly - No makefile editing
  - Fine control over compile, link settings
  - Compile options can be set per source file
  - Multiple toolchains can be defined per project
  - Some command expansion capabilities
  - Makefiles automatically generated
Introduction to CDT

CDT Debug Features

- Portable source level debugger
- Various views that extend Eclipse debug framework
  - Registers
  - Memory
  - Signals
  - Shared libraries
  - CDI (C Debugger Interface)
    - Allows targeting of a wide variety of CPU architectures
    - CDI APIs allow programmatic control over debugger
      - GDB-mi CDI implementation
      - MI plugin implementation (interface to GDB through machine-independent interface)
      - Support for MI level 1 and 2
      - Integrates with gdb version 5.2.1 and above
    - Win32 Native debugger (in-progress)
CDT Roadmap

2004:

- 2.0 GA

2005:

2.1

- Eclipse 3.0-based
- Integration API
- CDOM, Refactoring
- Managed build updates
- Class hierarchy
- I18N and accessibility

2.1.1

- New C++ class wizard
- Code Folding
- Makefile Editor
- MBS Improvements
- Debug Improvements

Maintenance Update

3.0

- Eclipse 3.1-based
- Faster Indexing
- AST/DOM
- Rename Refactoring
- MBS Improvements
- Debug Improvements

3.1
CDT 2.1 Release Highlights

CDT Managed Make Tutorial

View Executable

Navigate to the C/C++ Projects view and expand the Binary object, you will now see the executables listed.

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
    string yourName;
    cout << "Enter your name: ";
    cin >> yourName;
    cout << "Hello " + yourName << " 
    return 0;
```
CDT 3.0 Plan – Key Features and Themes

- Improved performance and scalability
  - Reduce indexing time and cost
  - Parser speed improvements

- AST/DOM

- Managed Build System enhancements

- Debugging enhancements

- UI enhancements

See [www.eclipse.org/cdt](http://www.eclipse.org/cdt) -> CDT Project Management -> 3.0 for the plan
CDT 3.0 AST/DOM and Parser

- CDT AST/DOM Completeness
  - Writable DOM (rename only)

- Port of DOM clients to use it
  - Content Assist
  - Search
  - Indexer
  - Will improve accuracy and performance of clients

- Parser Improvements
  - K&R C language support
  - Scanner Environment Discovery

- Indexer Improvements
  - Partial vs. full (accurate) index
  - Offline indexing
Managed Build System – 3.0 Plan

- Multi-version toolchain support
- Converter extension point
- Allow project conversion from one toolchain to another
- Build Macros
  - \$\{macro\} expansion in text configuration properties
- Environment variables
  - Environment or toolchain-defined variables
  - Passed to build system and used in the macro expansion
- Custom build steps
  - Command-line scripts
  - Pre- and Post- build scripts
- Custom file extensions
CDT 3.0 UI Improvements

- Have the Open Type, class browser and type hierarchy rely more on the indexer
  - Improve accuracy and speed of those views

- Enhanced C++ class wizards

- Update preference pages and templates to use Eclipse 3.1 functionality

- Definition/declaration view and open call hierarchy view

- Improved drag and drop support
CDT 3.0 Debug Improvements

- Replace CDT’s memory view with the Eclipse Debug Core view

- Source lookup improvements
  - Use platform’s source lookup framework
  - Improved search of external source

- New Modules view (replaced the shared library view)

- Eclipse 3.1 Debug Functionality
  - Limit display of call stack depth (using Eclipse 3.1 features)
  - Asynchronous implementation of step/resume/suspend/etc
Example Integration - Extending the Debugger

Debugger launch
- Standard debug launch for plain gdb, cygwin and gdb server
- May need to start session with own flavor of gdb, custom options
- May need to customize to perform additional steps, e.g.
  - Start simulator
  - Download code to target
  - Download additional files to target
  - Start extra tools
Example Integration – CDT Debugger

CDT Debug Core

Implementation of Eclipse Debug Core

Specific C / C++ Interface

CDI

CDT Launch

Extension point

GDB / MI

Cygwin / MI

My GDB flavor
CDT Debug Extension Points

What you need to do

- Implement 2 debugger extension points
  - point="org.eclipse.cdt.debug.core.CDebugger"
  - point="org.eclipse.cdt.debug.ui.CDebuggerPage">

Code starting points:
- org.eclipse.cdt.debug.mi.core -> GDBServerDebugger.java
- org.eclipse.cdt.debug.mi.ui -> GDBServerDebuggerPage.java
- about ~200 lines

Extension Points

<extension
  point="org.eclipse.cdt.debug.core.CDebugger">
  <debugger
    platform="native"
    name="%GDBDebugger.name"
    modes="run,core,attach"
    cpu="native"
    class="org.eclipse.cdt.debug.mi.core.GDBDebugger"
    id="org.eclipse.cdt.debug.mi.core.CDebugger">
  </debugger>
</extension>

<extension
  point="org.eclipse.cdt.debug.ui.CDebuggerPage">
  <debugPage
    class="org.eclipse.cdt.debug.mi.internal.ui.GDBDebuggerPage"
    id="org.eclipse.cdt.debug.mi.GDBDebuggerPage"
    debuggerID="org.eclipse.cdt.debug.mi.core.CDebugger">
  </debugPage>
</extension>

Debug configurations:

- Name: test.cpp
- Debugger: GDB Server
- GDB debugger: gdb
- GDB command file:
- Connection: Serial
- Device: /dev/tty30
- Speed: 115200
Example of integrating with CDT – Code Formatter

- CDT core provides an extension point for external code formatters:

  org.eclipse.cdt.core “CodeFormatter”

- Create your own plugin with “CodeFormatter” provider
  - Provides UI preferences content
  - Provide code to accept and format source stream
  - Pretty simple – see example code

- See [www.eclipse.org/cdt](http://www.eclipse.org/cdt) -> CDT Community Webpage
  -> GNU/Indent plugin
Example of CDT integration – MBS toolchain definition

Managed Build System uses toolchain definitions

- Toolchain contributed to extension point:
  - org.eclipse.cdt.managedbuilder.core

- Create your own plugin with "ManagedBuildInfo"
- Define your configurations (debug, release)
- Define your targets (executable, library, etc)
- Define tool names, command lines
- Define options

Extension Element Details
Set the properties of "enumeratedOptionValue"

id: gnu.c.optimization.level.more
name: %Option.Posix.Optimize More
isDefault: true
command: -O2
Contributed Plugins

- CPPUnit
- Integration of CPPUnit with CDT
- Eclipse-RPM
- Redhat package manager plugin
- Import RPM packages into CDT
- Export C/C++ projects as src and binary RPMs
- Eclipse-OProfile
- Integration of Linux oprofile (profiling) capabilities within Eclipse/CDT

See http://dev.eclipse.org/viewcvs/index.cgi/%7Echeckout%7E/cdt-home/community.html?cvsroot=Tools_Project
Long-term direction – What to expect from CDT

- Advanced features
  - DOM-enabled functionality
    - Refactoring
    - “quick fix”
  - Parallel and multi-core debugging
  - Project and code templates
  - Generic target model
  - Integrated Java and C/C++ development
- Increase adoption of CDT
  - Improve scalability and robustness
  - Mature ISV APIs to streamline integrations
  - Encourage additional integrations and contributions
- Extend CDT to further support embedded and enterprise needs
- Explore and participate in multi-language discussions
  - Can some elements of CDT be used as generic foundation for other compiled languages?
How to Contribute to CDT

- As a user of CDT
  - Download and use it for your C/C++ development
  - Provide feedback on features, usability
  - Suggest improvements
  - Report bugs
- As a developer of CDT
  - Provide patches and bugfixes
  - Implement features
  - CDT has no shortage of “hard” problems to solve
- Other areas
  - User documentation, HowTo’s, FAQ
  - Example integrations
  - Plugins that extend CDT

Want to get involved? We need your help! Visit www.eclipse.org/cdt.
Conclusion

- Interest in CDT keeps growing
- Many commercial products shipping with CDT
- CDT feature set/architecture still evolving based on feedback and needs
- CDT 3.0
  - Lots of work going on to bring CDT to the next level (refactoring, MBS)
  - Improved ISV APIs for integration
- We would love for you to get involved
  - As a user and developer
  - Many areas in need of contributions and leadership

These slides and the BOF slides will be posted on the CDT website