ides and eclipse

myths and facts about the worlds greatest ide

Greg Watson
IBM T.J. Watson Research Center

Contents

Why Use An IDE?

- Many published studies have shown that IDEs improve productivity
- Large number of software development organizations (e.g. Microsoft) have embraced the IDE as a central development platform
- As software and architectures become more complex, traditional tools will not be enough to achieve performance and quality requirements

IDE Benefits

- Ensures all tools required for development lifecycle are available
- Improves workflow
- Many activities can be automated
- Simplifies training and support
- Consistent tool set improves portability
- Helps avoid vendor lock-in

Eclipse Background

- Originally developed by Object Technology International (OTI) and purchased by IBM for use by internal developers
- Released to open-source community in 2001, managed by consortium
- Consortium reorganized into independent not-for-profit corporation, the Eclipse Foundation, in early 2004

Eclipse Foundation

- 157 member companies
- 20 strategic members
- 768 committers across 50+ organizations
- 700K downloads per month
  - 7M in 2006
- Number of developers using Eclipse will surpass Visual Studio in 2008

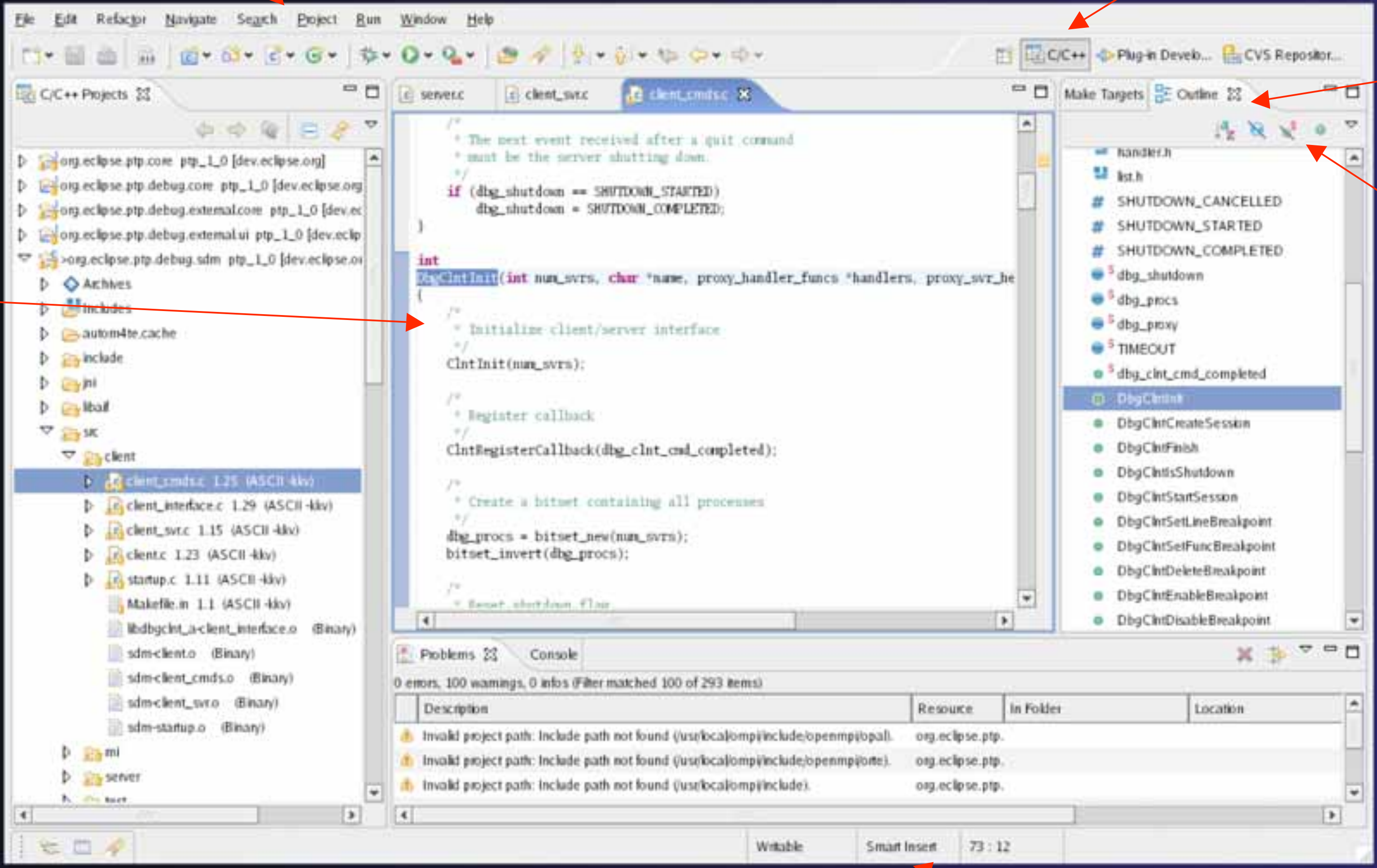
What is Eclipse?

- Cross-platform open source framework for highly integrated state-of-the-art tools
  - Designed to be robust, scalable, commercial quality
  - Highly extensible plugin architecture
- Wide array of integrated tools available
- Available for Linux, Unix and Windows
- Multi-language support for Java, C, C++, Fortran, Python, Perl, PHP, and others

Features

- Workbench
- Perspectives
- Project management
- Editors
- Build system
- Revision control
- Running & debugging
- Other tools

Workbench


menus
toolbar
perspective switcher
views
view menu
editor
fast views
status bar
File Edit Refactor Navigate Search Project Run Window Help
C/C++ Plug-in Develo... CVS Repositor...
C/C++ Projects
org.eclipse.ptp.core ptp_1_0 [dev.eclipse.org]
org.eclipse.ptp.debug.core ptp_1_0 [dev.eclipse.org
org.eclipse.ptp.debug.external.core ptp_1_0 [dev.ec
org.eclipse.ptp.debug.external.ui ptp_1_0 [dev.eclip
>org.eclipse.ptp.debug.sdm ptp_1_0 [dev.eclipse.or
Archives
Includes
autom4te.cache
include
jni
libaif
src
client
client_cmds.c 1.25 (ASCII -kkv)
client_interface.c 1.29 (ASCII -kkv)
client_svr.c 1.15 (ASCII -kkv)
client.c 1.23 (ASCII -kkv)
startup.c 1.11 (ASCII -kkv)
Makefile.in 1.1 (ASCII -kkv)
libdbgclnt_a-client_interface.o (Binary)
sdm-client.o (Binary)
sdm-client_cmds.o (Binary)
sdm-client_svr.o (Binary)
sdm-startup.o (Binary)
mi
server
server.c
client_svr.c
client_cmds.c
Make Targets
Outline
handler.h
list.h
SHUTDOWN_CANCELLED
SHUTDOWN_STARTED
SHUTDOWN_COMPLETED
dbg_shutdown
dbg_procs
dbg_proxy
TIMEOUT
dbg_clnt_cmd_completed
DbgClntInit
DbgClntCreateSession
DbgClntFinish
DbgClntIsShutdown
DbgClntStartSession
DbgClntSetLineBreakpoint
DbgClntSetFuncBreakpoint
DbgClntDeleteBreakpoint
DbgClntEnableBreakpoint
DbgClntDisableBreakpoint
Problems
Console
Writable
Smart Insert
73 : 12

```c
	/*
	 * The next event received after a quit command
	 * must be the server shutting down.
	 */
	if (dbg_shutdown == SHUTDOWN_STARTED)
		dbg_shutdown = SHUTDOWN_COMPLETED;
}

int
DbgClntInit(int num_svrs, char *name, proxy_handler_funcs *handlers, proxy_svr_he
{
	/*
	 * Initialize client/server interface
	 */
	ClntInit(num_svrs);

	/*
	 * Register callback
	 */
	ClntRegisterCallback(dbg_clnt_cmd_completed);

	/*
	 * Create a bitset containing all processes
	 */
	dbg_procs = bitset_new(num_svrs);
	bitset_invert(dbg_procs);

	/*
	 * Reset shutdown flag
```

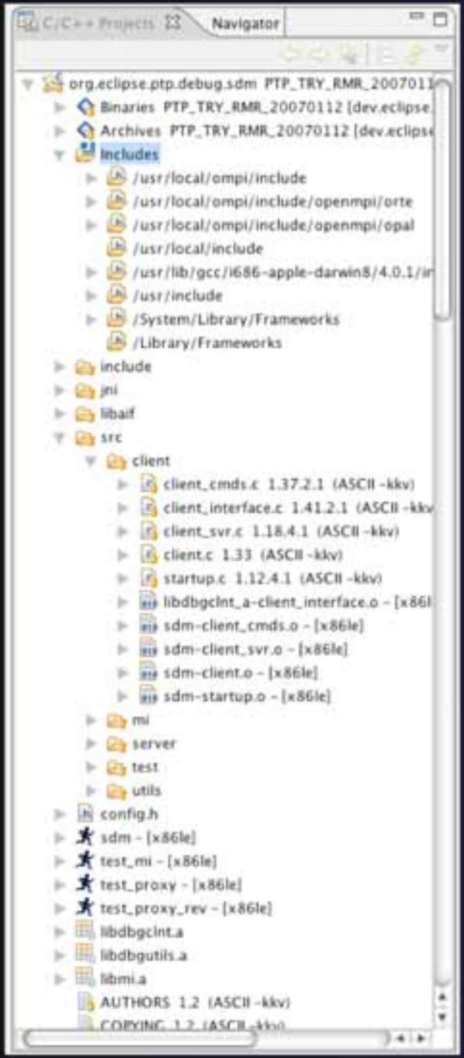
0 errors, 100 warnings, 0 infos (Filter matched 100 of 293 items)

Description	Resource	In Folder	Location
Invalid project path: Include path not found (/usr/local/ompi/include/openmpi/opal).	org.eclipse.ptp.		
Invalid project path: Include path not found (/usr/local/ompi/include/openmpi/orte).	org.eclipse.ptp.		
Invalid project path: Include path not found (/usr/local/ompi/include).	org.eclipse.ptp.		

Perspectives

- A collection of views is known as a *perspective*
- Allows views associated with an activity to be grouped together
  - E.g. C/C++ development, debugging, collaboration
- Switch perspectives by clicking a button
- Perspectives can be customized and saved

Project Management



```
org.eclipse.ptp.debug.sdm  PTP_TRY_RMR_2007011
  Binaries  PTP_TRY_RMR_20070112 [dev.eclipse.
  Archives  PTP_TRY_RMR_20070112 [dev.eclipse
  Includes
    /usr/local/ompi/include
    /usr/local/ompi/include/openmpi/orte
    /usr/local/ompi/include/openmpi/opal
    /usr/local/include
    /usr/lib/gcc/i686-apple-darwin8/4.0.1/in
    /usr/include
    /System/Library/Frameworks
    /Library/Frameworks
  include
  jni
  libaif
  src
    client
      client_cmds.c  1.37.2.1  (ASCII -kkv)
      client_interface.c  1.41.2.1  (ASCII -kkv
      client_svr.c  1.18.4.1  (ASCII -kkv)
      client.c  1.33  (ASCII -kkv)
      startup.c  1.12.4.1  (ASCII -kkv)
      libdbgclnt_a-client_interface.o - [x86l
      sdm-client_cmds.o - [x86le]
      sdm-client_svr.o - [x86le]
      sdm-client.o - [x86le]
      sdm-startup.o - [x86le]
    mi
    server
    test
    utils
  config.h
  sdm - [x86le]
  test_mi - [x86le]
  test_proxy - [x86le]
  test_proxy_rev - [x86le]
  libdbgclnt.a
  libdbgutils.a
  libmi.a
  AUTHORS  1.2  (ASCII -kkv)
  COPYING  1.2  (ASCII -kkv)
```

- Source code is organized into projects
  - Projects arrange files in hierarchical view
  - Icons distinguish file types
  - Maintains properties about the project
- Many different project types exist
  - Some allow mixed languages
- Easy to import existing source code
- Projects can be located in the *workspace* or linked to any location in the file system

Editors

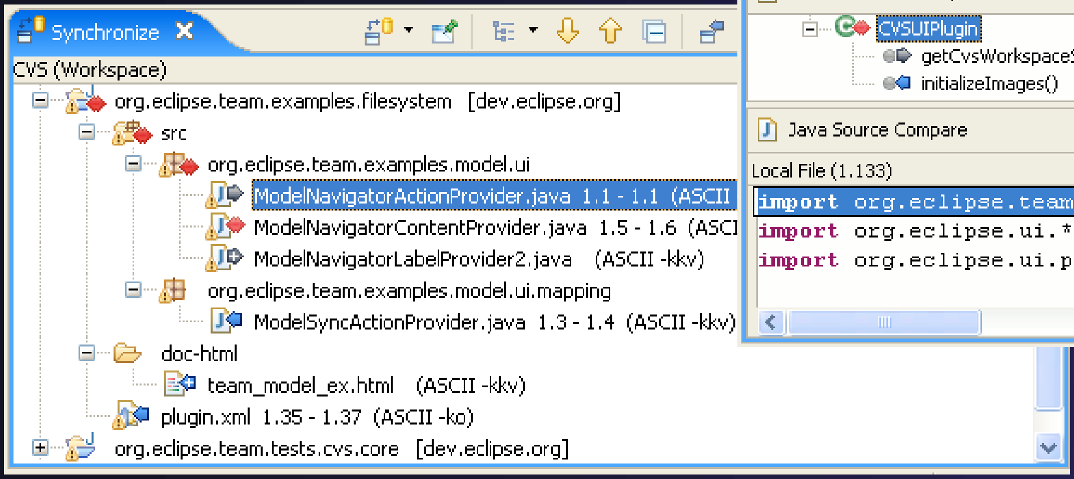
- Editors are language specific, but provide common features:
  - Syntax highlighting/coloring (doesn't everyone)
  - Outline view (high level view of source code)
  - Language specific error checking
  - Content assist
  - Search on language feature (type, constructor, etc.)
  - Refactoring
  - Context sensitive help

Build System

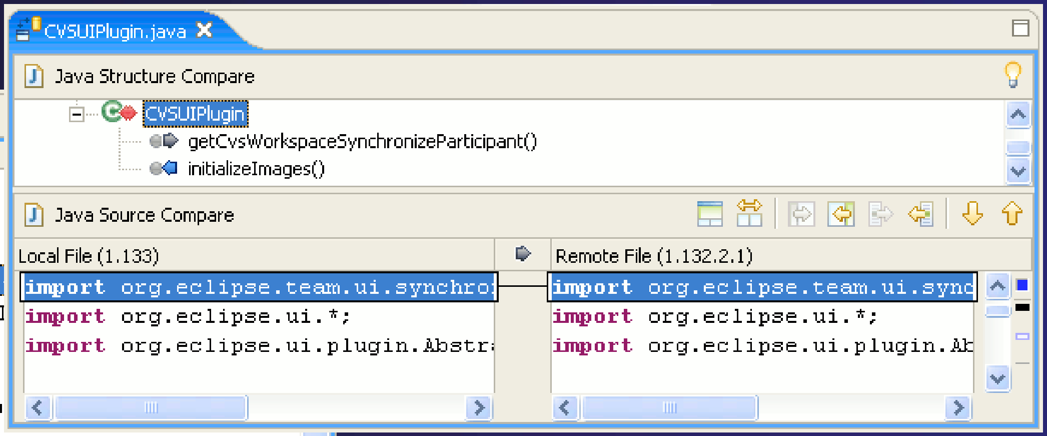
- Eclipse can manage building a project
  - Ant for Java projects
  - Automatic makefile generation for C/C++/Fortran
- Existing makefile-based build systems
  - Fully supported
  - Ability to seamlessly use make targets
- Autoconf can be used
  - Better integration is under development

Revision Control

- Full integration with CVS and Subversion
  - Team synchronization
  - Merging, versions, branches
  - Compare editor



CVS (Workspace)
- org.eclipse.team.examples.filesystem [dev.eclipse.org]
  - src
    - org.eclipse.team.examples.model.ui
      - ModelNavigatorActionProvider.java 1.1 - 1.1 (ASCII
      - ModelNavigatorContentProvider.java 1.5 - 1.6 (ASCI
      - ModelNavigatorLabelProvider2.java (ASCII -kkv)
    - org.eclipse.team.examples.model.ui.mapping
      - ModelSyncActionProvider.java 1.3 - 1.4 (ASCII -kkv)
  - doc-html
    - team_model_ex.html (ASCII -kkv)
  - plugin.xml 1.35 - 1.37 (ASCII -ko)
- org.eclipse.team.tests.cvs.core [dev.eclipse.org]



- CVSUIPlugin
  - getCvsWorkspaceSynchronizeParticipant()
  - initializeImages()

Local File (1.133)

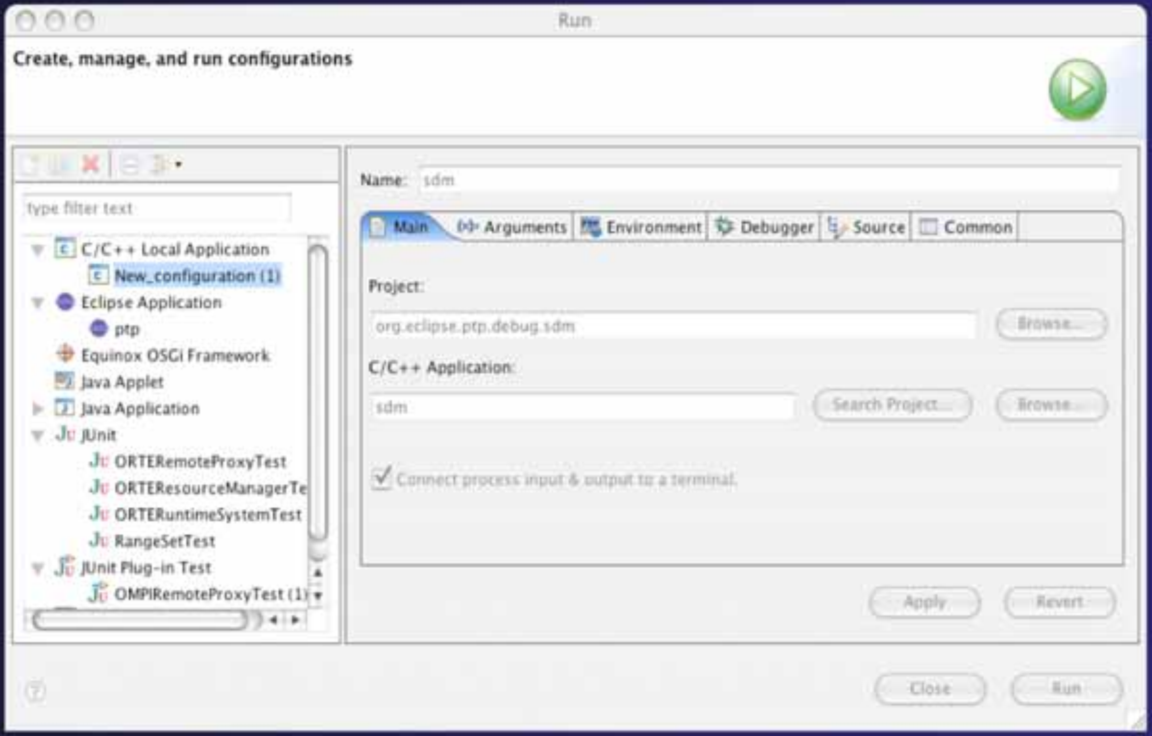
```
import org.eclipse.team.ui.synchro
import org.eclipse.ui.*;
import org.eclipse.ui.plugin.Abstr
```

Remote File (1.132.2.1)

```
import org.eclipse.team.ui.sync
import org.eclipse.ui.*;
import org.eclipse.ui.plugin.Ab
```

Running & Debugging

- Run configuration dialog
- Manages run and debug parameters, including program arguments and environment
- Supports a wide range of configuration types
- Single click launch from toolbar



Name: sdm

Project: org.eclipse.ptp.debug.sdm

C/C++ Application: sdm

✓ Connect process input & output to a terminal.

Running & Debugging (cont...)

- Integrated visual debugger
- Common interface across different languages
- Java debugging supports hot fixes
- C/C++ debugging supports full gdb command set and remote targets
- Parallel debugging with Parallel Tools Platform (PTP)

Other Tools

- Enterprise development
  - Business tools (BIRT), testing (TPTP), web tools (WTP)
- Embedded development
  - C/C++ tools (CDT), device software (DSDP), mobile tools for Java (MTJ), native application builder (NAB)
- Parallel development
  - Parallel Tools Platform (PTP), parallel debugger, MPI/OpenMP tools
- Application frameworks
  - Communication (ECF), modeling (EMF), graphical modeling (GMF), identity (Higgins), UML
- One third party site has over 800 plugins

Myths About Eclipse

Myth #1: Emacs Rules

- Emacs is the best editor ever, why would I want to use a big clunky IDE?

Myth #1: Emacs Rules

BUSTED

- The correct tool should be used for every job
  - Traditional editors may be appropriate for some tasks
- Eclipse provides functionality that is more appropriate for large team-based software development projects
- Sophisticated, language sensitive tools, achieve productivity improvements that are not possible with traditional environments

Myth #2: Java Only

- Eclipse is a Java development environment, so I have to develop my applications in Java

Myth #2: Java Only

BUSTED

- Eclipse was originally designed for Java development, but now provides a rich, multi-language, ecosystem
- Eclipse supports Java, C, C++, Fortran, COBOL, Python, Perl, PHP, JavaScript, Ruby, Ada, and more...
- Eclipse allows you to choose the most appropriate language for your application

Myth #3: Poor Performance

- Java is too slow, so Eclipse is sluggish

Myth #3: Poor Performance

BUSTED

- Early versions of Eclipse were slow
- Significant improvements have been achieved in:
  - Eclipse platform performance
  - JVM performance
  - Reducing memory usage
- Combined with improvements in hardware speed means there is no discernable difference between Eclipse and native applications

Myth #4: Clunky Look & Feel

- UI development in Java requires Swing, which has a clunky look and feel

Myth #4: Clunky Look & Feel

BUSTED

- Eclipse and Eclipse-based Java applications use the standard widget toolkit (SWT) not Swing
- SWT uses native widgets to achieve:
  - Native look and feel
  - Better performance

Myth #5: Large Project Problems

- I’ve heard that Eclipse has problems for projects with >500K lines of code and thousands of source files

Myth #5: Large Project Problems

BUSTED

- Early versions of the C/C++ tools did have some problems with large projects
- This is no longer the case
- Eclipse is regularly used to build the Linux kernel (5M lines of code) and other large projects such as Apache

Myth #6: Too Complicated

- The Eclipse interface is too complicated and difficult to learn

Myth #6: Too Complicated

BUSTED

- The Eclipse interface can be overwhelming for the novice user
- However, the interface is designed for consistency, so once a user becomes familiar with the concepts there are usually no problems
- Eclipse's plugin architecture also allows unneeded functionality to be removed if necessary

Myth #7: Hard to Install

- I downloaded Eclipse, but then I had to download a bunch of other stuff before I could use it

Myth #7: Hard to Install

PLAUSIBLE

- Eclipse is separated into different components corresponding to the top level projects
- Downloading the Eclipse SDK only provides Java functionality
- Other tools must be installed separately using the update manager
- EasyEclipse now provides pre-packaged Eclipse distributions

Conclusion

- Eclipse provides an incredibly rich set of development tools for virtually all software development activities
- Freely available, open-source, and backed by a very large global developer community
- Licensing model is designed to support commercial applications of Eclipse
- Trying Eclipse is the best way to understand the benefits

More Information

- http://eclipse.org
  - Main Eclipse web site
- http://planeteclipse.org
  - Eclipse blogs
- http://eclipsezone.org
  - Eclipse community site
- http://eclipseplugincentral.org
  - Third party Eclipse plugins
- http://easyeclipse.org
  - Pre-packaged Eclipse solutions