Agile C++ through Advanced IDE Features
Using Eclipse CDT for C++ Development

Prof. Peter Sommerlad
HSR - Hochschule für Technik Rapperswil
IFS Institute for Software
Oberseestraße 10, CH-8640 Rapperswil
peter.sommerlad@hsr.ch
http://ifs.hsr.ch
http://wiki.hsr.ch/PeterSommerlad
Outline

- Why an IDE for C++ development, why not
- Basic Eclipse CDT Usage
- Useful Plug-ins for C++ Development with CDT
- Code Analysis: CODAN
- Sconsolidator for Scons users
- Agility through CUTE Unit Testing, Refactoring and Feedback
- Outlook: what to expect from the future

- Most of the features are demoed
Quick Questionnaire
Quick Questionnaire

What are you using for C++ development:
Quick Questionnaire

- What are you using for C++ development:
  - vi(m)
Quick Questionnaire

- What are you using for C++ development:
  - vi(m)
  - emacs
Quick Questionnaire

- What are you using for C++ development:
  - vi(m)
  - emacs
  - some other editor
Quick Questionnaire

• What are you using for C++ development:
  • vi(m)
  • emacs
  • some other editor
Quick Questionnaire

• What are you using for C++ development:
  • vi(m)
  • emacs
  • some other editor

• Apple XCode
Quick Questionnaire

What are you using for C++ development:

- vi(m)
- emacs
- some other editor
- Apple XCode
- Visual Studio
Quick Questionnaire

What are you using for C++ development:

- vi(m)
- emacs
- some other editor
- Apple XCode
- Visual Studio
- Netbeans
Quick Questionnaire

What are you using for C++ development:

- vi(m)
- emacs
- some other editor
- Apple XCode
- Visual Studio
- Netbeans
- KDevelop / QTCreator
Quick Questionnaire

What are you using for C++ development:

- vi(m)
- emacs
- some other editor
- Apple XCode
- Visual Studio
- Netbeans
- KDevelop / QTCreator
- Eclipse CDT for C++
Quick Questionnaire

• What are you using for C++ development:
  • vi(m)
  • emacs
  • some other editor

• Apple XCode
• Visual Studio
• Netbeans
• KDevelop / QTCreator
• Eclipse CDT for C++
• some other C++ IDE
Brief Eclipse CDT Installation

  - current release Helios SR2, future: Indigo
  - For Windows users: look at Wascana
    - brings mingw and CDT on your machine
    - MS C++ support still nascent and not generally available.

- Install further useful plug-ins
  - e.g. for git or subversion integration
  - a must: http://cute-test.com
CDTs Bad Reputation

- The early versions (up to 4-5) where almost unusable, except for "hello world" projects.
  - CDT got much better over the years
- Focus of early main contributors was on debugging and embedded development.
- C-programmers writing Java code.
- Parser and internal model was incomplete.
  - Parser had been a Java translation of GCCs C code
  - C++ model was derived(copied) from Java.

- That is no longer the case: CDT of today works.
  - try it!
Brief Eclipse CDT
Introduction

• Key concepts (might be known from Java)
  o Workspace
    ➢ this is where your projects reside and are organized
    ➢ one Eclipse CDT instance can open one Workspace
    ➢ you can switch workspaces
    ➢ there is no cross-workspace relationship built in
  o Project
    ➢ organizes files around one build target (built-in builder)
    ➢ different "kinds" of projects (cannot change afterwards!)
    ➢ building results in either an executable or a library/DLL
    ➢ different build-configurations for Debug/Releases/etc.
    ➢ cross-project relationships are useful: exe -> libs
Key concepts (might be known from Java)

- Perspective = arrangement of views
  - usually for different tasks: Java, C++Dev, C++ Debugging
  - switch perspective - rearrange your own perspective
  - wrong perspective can give surprising behavior

- View = a window/tile/tab with specific functionality
  - some useful views are hidden in some perspectives
  - you need to open some explicitly, Windows->Show View
  - e.g., Error Log of Eclipse to figure out problems with installation
  - some views look very similar, that can be confusing because of different, e.g., Project Explorer vs. Navigator
  - different views interact, e.g., for navigation
Perspective and Views

Perspective Selector

Project Navigator View
Other UI elements

- **Standard Menus**

- **and buttons**

  - depend on perspective, view with focus,
  - and installed plug-ins
Real Benefits of CDT while Editing

• auto completion in the editor
  o CTRL-SPACE (CMD-SPACE)
• hovering over an identifier gives its definition or declaration (classes, functions, macros)
• quick navigation to an identifiers definition/declaration (CTRL-click)
• step-by-step analysis of macro expansions
• navigation from compile errors and warnings to the corresponding source position
• marking of problems within the editor
• search based on string search and on references
• some refactoring support available
Benefits of CDT for Novice Users

- no hassles with make
  - except with cygwin problems on windows (→ Wascana)

- (most) compiler settings just work
  - except for multi-project executables where manual adaption is still needed (except for CUTE test projects)

- code frames and #include guards are generated (New→Class)

- version management connection built-in (CVS) or available as plug-in (SVN, git, ...)
  - diff-viewer/editor, ... (not shown today)
Example: Macro Expansion Analysis
Example: Macro Expansion Analysis

```c++
#define ASSERT_EQUAL(expected, actual) ASSERT_EQUALM(expected, actual, (expected), (actual))

void testTheAnswer()
{
    ASSERT_EQUAL(42, theAnswer);
}
```

- Original: `ASSERT_EQUAL(42, theAnswer)`
- Fully Expanded: `cute::assert_equal((42), (theAnswer), "42" == "theAnswer", "/Users/sop/Desktop/word...`)
Example: Macro Expansion Analysis

```cpp
void testTheAnswer()
{
    ASSERT_EQUAL(42, theAnswer);
}
```

```
#define ASSERT_EQUAL(expected, actual) ASSERT_EQUALM(#expected " == " #actual, (expected), (actual))
```

Original:
```
ASSERT_EQUAL(42, theAnswer)
```

Fully Expanded:
```
cute::assert_equal(42, theAnswer), "42" == "theAnswer", 
```

Expand Macro Expansion - 4 step(s)
```cpp
#define ASSERT_EQUAL(expected, actual) ASSERT_EQUALM(#expected " == " #actual, (expected), (actual))
```

Original
```
ASSERT_EQUAL(42, theAnswer)
```

Expansion #1 of 4
```
ASSERT_EQUALM("42" " == " "theAnswer", 42, (theAnswer))
```
Example: Macro Expansion Analysis

```cpp
class Operate {
public:
    void testTheAnswer() {
        ASSERT_EQUAL(42, theAnswer);
    }
};
```

Original:
```
#define ASSERT_EQUAL(expected, actual) ASSERT_EQUALM(#expected " == " #actual, (expected), (actual))
```

Fully Expanded:
```
void testTheAnswer(){
    ASSERT_EQUAL(42, theAnswer);
}
```

Expanded:
```
#define ASSERT_EQUALM(msg, expected, actual) cute::assert_equal((expected), (actual), msg, __FILE__, __LINE__)
```

Expansion #1 of 4
```
ASSERT_EQUALM("42" " == " "the
```

Expansion #2 of 4
```
cute::assert_equal(((42)), ((theAnswer)), "42" " == " "theAnswer", __FILE__, __LINE__)
```
Example: Macro Expansion Analysis

```c++
void testTheAnswer() {
    ASSERT_EQUAL(42, theAnswer);
}

// Define the macro
#define ASSERT_EQUAL(expected, actual) ASSERT_EQUALM(#expected " == " #actual, (expected), (actual))

// Original
ASSERT_EQUAL(42, theAnswer)

// Fully Expanded
cute::assert_equal(((42),((theAnswer)),"42" " == " "theAnswer","/Users/sop/Desktop/work...)

// Define the macro
#define ASSERT_EQUALM(msg, expected, actual) cute::assert_equal((expected),(actual),msg,FILE,LINE)

// Expansion #1 of 4
ASSERT_EQUALM("42" " == " "theAnswer",FILE,LINE)

// Expansion #2 of 4
cute::assert_equal(((42),((theAnswer)),"42" " == " "theAnswer",FILE,LINE)

// Expansion #3 of 4
" " " == " "theAnswer",FILE

// Expansion #4 of 4
swer),"42" " == " "theAnswer","/Users/sop/Desktop/workspace_helios/demo/src/Test.cpp",
```
Example: Macro Expansion Analysis

```c
void testTheAnswer()
{
    ASSERT_EQUAL(42, theAnswer);
}
```

```c
#define ASSERT_EQUAL(expected, actual) ASSERT_EQUALM(#expected " == " #actual, (expected), (actual))
```

```c
void testTheAnswer()
{
    ASSERT_EQUAL(42, theAnswer);
}
```

```c
#define ASSERT_EQUALM(msg, expected, actual) cute::assert_equal((expected), (actual), msg, __FILE__, __LINE__)
```

```c
void testTheAnswer()
{
    ASSERT_EQUALM("42" == "theAnswer", __FILE__, __LINE__);
}
```

```c
#define __FILE__ "/Users/sop/Desktop/workspace_helios/demo/src/Test.cpp"
```

```c
void testTheAnswer()
{
    ASSERT_EQUALM("42" == "theAnswer", __FILE__, __LINE__);
}
```

```c
#define __LINE__ 10
```

```c
void testTheAnswer()
{
    ASSERT_EQUALM("42" == "theAnswer", __FILE__, __LINE__);
}
```
New interesting features of the CDT project

- **very advanced debugging support**
  - view threads, memory, variables etc.

- **beginning support for C++0x syntax**
  - lambdas, auto, (nullptr currently missing)

- **Codan - code analysis framework**
  - will mark problems while editing (or later)
  - still nascent, not all rules are perfect, can slow editing
  - building your own rules requires Java programming (as of today) and a lot of knowledge about CDT internals (AST)
  - we use it (and improve it) for some of our plug-ins
Codan checkers (Indigo)
Things to watch out

- Sometimes Eclipse CDT can behave confusingly
- Things that can happen:
  - using Java perspective while trying to edit C++ projects
  - opening a file in a wrong editor, e.g., system header files that do not have an extension
  - auto-completion not working because the Indexer is not done, or project settings are wrong, while the code still compiles perfectly
  - CODAN or the built-in spell checker complaining, while the code compiles perfectly and the spelling of your strings and comments just doesn't matter
Annoying thing for novices

- CDT's Console can lock Eclipse, when you enter EOF (CTRL-D) on some platforms
  - problem with exercises when reading from std::cin

Deselect this checkbox
Our (upcoming) CDT plug-in's features

- **CUTE C++ Unit Testing Easier**
  - TDD-Support (work-in-progress)
  - Test-Stub/Mock-Object support (WIP - not shown)

- **Refactoring Infrastructure**
  - Toggle Function Definition Position
  - Extract Typename Template Parameter
  - Transform loops to STL algorithms (research)

- **Concept Deduction and Check for Template Parameters (WIP - not shown)**

- **Include Optimization, Lint Integration**
  - (not shown, might become commercial, free beta test available)
CUTE plug-in

- integrates CUTE unit testing framework in CDT
- free

**features:**
- wizard creating test projects
- test function/registration generator with ...
- automatic detection of unregistered tests
- test navigator with green/red bar
- diff-viewer for failing tests
- code coverage view with gcov

**plus preview of refactorings:**
- toggle function definition position (CTRL-SHIFT-V)
- extract typename template parameter
CUTE plug-in

```c
#include "cute_runner.h"

struct ModulusTester
{
  bool divides(const unsigned & i)
  {
    return i%number == 0;
  }
  unsigned number;
  ModulusTester(unsigned int val):number(val){};
};
void thisIsATest()
{
  ModulusTester f(7);
  ASSERTMC("start writing tests", f.divides(43))
  ASSERT_EQUAL(true, f.divides(7));
}
void runSuite()
{
  cute::suite s;
  //TODO add your test here
  s.push_back(CUTE(thisIsATest));
  cute::idle_listener lis;
  cute::makeRunner(lis)(s, "The Suite");
}
int main()
{
  runSuite();
  return 0;
}
```

Test runner

Coverage

comparison view
CUTE support

- If you forget to register a test, you will be warned and registration happens on your behalf:

- It is even better to let the plug-in generate the test functions for you (CTRL-ALT-F/CMD-ALT-F)
CUTE plug-in stuff to come

- support for TDD
  - see demo
  - create local variable, function, class on a keystroke (aka quick-fix: CTRL-1/CMD-1)
    - work in progress, code written and compiled on Tuesday
  - move created types or function to a new header

- support for test-stubs and mock objects
  - dependency injection through template params
    - no "extract base class" and inheritance required
  - test-stub generation from template parameter "concept"
    - requires C++0x for local class as template argument
Refactoring

- We created (most of) the refactoring infrastructure within CDT
  - code (re-)generation from AST nodes
  - keeping comments when moving AST nodes
  - keeping macro calls when re-generating code from AST nodes (not the macro expansion)
- That meant hard work over several years
- Still a lot to do to improve the infrastructure
- Indexer is now a lot better than in the past
  - thanks to WindRiver's Markus Schorn
- Unfortunately many concrete refactorings didn't get maintained for newer releases
  - or were student projects lacking in quality
Interesting Refactorings (our inventions)

- **Toggle Function Definition**
  - we tried with "change function signature"
    - hard to use, clumsy dialog
Interesting Refactorings (our inventions)

- **Toggle Function Definition**
  - we tried with "change function signature"
    - hard to use, clumsy dialog
Interesting Refactorings (our inventions)

- **Toggle Function Definition**
  - we tried with "change function signature"
    - hard to use, clumsy dialog
Introduction

- Refactoring is a key element of modern software design
- In C++ refactoring demands multiple manipulations of source code
- Automated refactoring generates all changes out of one user input
- CDT lacks some important refactoring tools

Approach

- “Declare Function” as counterpart to “Implement Method”
- Main feature “Change Function Signature” with a lot of functionality

Result

- Ease the burden of a C++ developer to simultaneously change code in multiple files
- Better code with less errors

```
//Example.h
class Example {
    int foo(int i = 42);
}

//Example.cpp
#include "Example.h"
int Example::foo(int i){
    return i + 2;
}
```

```
//Example.h
class Example {
    int foo(Example &other, double number);
}

//Example.cpp
#include "Example.h"
int Example::foo(Example &other, double number){
    int i = 42
    return i + 2;
}
```
Interesting Refactorings (our inventions)

- **Toggle Function Definition**
  - we tried with "change function signature"
    - hard to use, clumsy dialog
  - therefore, we came up of the idea of collapsing function definition and declaration into a single place in the header file, let the user edit the single definition, and then toggle the definition back into the implementation file.

- **Extract Typename Template Parameter**
  - templates are often underused (e.g., for dependency injection), but it is tedious to make a concrete type used in a class or a function a template parameter, we automated that.
Extracting Typename Template Parameter
Extracting Typename
Template Parameter
Extracting Typename Template Parameter

The following changes are necessary to perform the refactoring.

Changes to be performed
- Changes
- `Test.cpp` in `demo1/src`

**Original Source**
```cpp
int square(int bar)
{
    return bar*bar;
}

void testTheAnswer()
{
    ASSERT_EQUAL(42, theAn
```

**Refactored Source**
```cpp
template<typename T1> T1 square(T1 bar)
{
    return bar * bar;
}
```
Extracting Typename
Template Parameter

```cpp
template<
typename T1>
T1 square(T1 bar)
{
    return bar * bar;
}
```
SConsolidator
Integrating SCons

- conversion of existing CDT "managed build" projects to SCons projects
- import of existing SCons projects into Eclipse with wizard support
- SCons projects can be managed either through CDT or SCons
- "interactive mode" to quickly build single source files speeding up round trip times
- a special view for a convenient build target management of all workspace projects
- graph visualization of build dependencies with numerous layout algorithms and search and filter functionality that enables debugging SCons scripts.
- quick navigation from build errors to source code locations
DEMO Time: TDD with CDT
More information on using CDT

- at least for German speaking developers
- "the missing manual"
Summary

- Eclipse CDT can and should be used for C++ development
  - especially if your target platform is not Windows
  - CDT runs well on Linux, Mac OS X and Windows
- More interesting features to come
  - improved refactoring and quick-fixes for typical coding problems
  - support for TDD and mock objects
  - our "to-be-commercial" but free beta testable plug-ins "Linticator" and "Includulator"
Further Information

- Eclipse für C/C++ Programmierer
  - German book (unfortunately no english translation available (yet))
- http://www.eclipse.org/cdt/
- http://cute-test.com
- http://www.ifs.hsr.ch
- http://includator.com (register for beta test)
- http://linticator.com (register for beta test)
Questions?

- contact me at peter.sommerlad@hsr.ch