Photran 6.0 Developer's Guide

Part I: General Information

N. Chen
J. Overbey

Contents

Checklist for New Developers

Revision: $Id: checklist.ltx-inc,v 1.1 2010/04/28 18:12:52 joverbey Exp

- ◯ **Install Photran, read its documentation, and become a proficient user.** Generally speaking, you cannot fix or enhance a program you don't know how to use.
- ◯ **Check out Photran's source code from CVS and run the test suite.** See Appendices A and B.
- ◯ **Join the photran-dev mailing list.** This is where discussions about Photran's development take place. Questions from developers (like you), release planning, API changes, etc. all get discussed here.
- ◯ **Join the photran users' mailing list.** It is good to keep track of what Photran's users are discussing, and as a Photran developer you will probably be able to answer many of their questions as well.
- ◯ **Create a Bugzilla account.** You will need to create an account at bugs.eclipse.org if you don't already have one. When you eventually want to contribute code to Photran, you will need to do it by attaching a patch to a Bugzilla bug.
- ◯ **Get a book or two on Eclipse plug-in development.** Eventually, you'll want to buy (or check out from the library) a book on Eclipse plug-in development. *The Java Developer's Guide to Eclipse* by D'Anjou et al. is a good choice.
- ◯ **Learn the basics of Eclipse plug-in development.** If you get D'Anjou's book, read Chapters 7-9 and 21. After you figure out the basics of Eclipse development (like how plugin.xml works), it's best to just start fixing bugs or adding features, returning to the book as a reference when you have something more specific you need to do.
- ◯ **Read the Photran Developer's Guide.** Chapters 1, 2, and 3 apply to everyone. The other chapters are more specialized; for example, the chapters on Parsing and Refactoring only apply to people developing refactorings, and the Photran Editors chapter applies mostly to people adding features to the Fortran editor.

Chapter 1

Introduction

Revision: $Id: intro.ltx-inc,v 1.9 2010/04/28 18:12:52 joverbey Exp - based on 2008/08/08 nchen

Photran is an IDE for Fortran 77–2008 that is built on top of Eclipse. It is structured as an Eclipse feature, in other words, as a set of plug-ins that are designed to be used together. Starting with version 3.0, it is an extension of C/C++ Development Tools (CDT), the Eclipse IDE for C/C++. The first version of Photran was created by hacking a copy of CDT to support Fortran instead of C/C++, but now we have developed a mechanism for adding new languages into CDT, allowing the Fortran support code to be in its own set of plug-ins.

Our purpose in writing Photran was to create a refactoring tool for Fortran. Thus, Photran has a complete parser and program representation. Photran also adds a Fortran editor and several preference pages to the CDT user interface, as well as Fortran-specific project wizards and support for several Fortran compilers.

Photran is part of the Parallel Tools Platform (PTP) project at the Eclipse Foundation, which provides the Web site, CVS repository, and Bugzilla repository. The Web site is http://www.eclipse.org/photran, and bugs can be submitted at https://bugs.eclipse.org under Tools > PTP by selecting one of the "Photran" components.

1.1 How To Read This Guide

This document explains the design of Photran so that interested contributors could fix a bug or add a refactoring. Contributors should know how to use Photran; user's guides and introductory tutorials are available from the *Documentation* section of Photran's website. Contributors also need to understand how to build Eclipse plug-ins before they read this document. Two good books on Eclipse plug-in development are *Building Commercial-Quality Plug-ins* and *The Java Developer's Guide to Eclipse.* Like almost all Eclipse plug-ins, Photran is written in Java, so knowing Java is also a prerequisite.

This is a *duplex* guide. The main chapters provide general descriptions of the various components and how they interact. The appendix describes concrete examples from Photran so that contributors can familiarize themselves with actual code and implementation details. This guide complements the source code in the repository; it is not a substitute for reading the actual source code.

1.2 Getting Started

If you have not already, work through the items on the "Checklist for New Developers" at the beginning of this manual.

In particular, *please* join the photran-dev mailing list, and use it to ask for help! Although we have attempted to document the most-used parts of Photran's codebase, either in JavaDoc or in this manual, our documentation is by no

means comprehensive. As you attempt to fix or extend parts of Photran, you will probably encounter problems that you cannot resolve simply by reading the documentation or reverse engineering the code. When this is the case, ask for help! It will make you more productive, and it will help us improve this documentation.

When your contribution is finished, follow the procedure in Appendix D to contribute your code to the "official" version of Photran.

Chapter 2

Plug-in Decomposition

Revision: $Id: plugins.ltx-inc,v 1.10 2010/04/28 18:12:52 joverbey Exp - based on 2008/08/08 nchen

2.1 Introduction

This chapter presents a high-level overview of the different projects and plug-ins in Photran. It serves as a guide for developers reverse-engineering Photran to *guess-and-locate* where certain components are. It also serves as a guide for contributers on *where* to put their contributions.

2.2 The Rephraser Engine

There are two major components in Photran's CVS repository: Photran itself, and another project called the Rephraser Engine. For the most part, you will probably be able to ignore the Rephraser Engine, but it is helpful to know what it is.

While developing Photran, one of our research objectives has been to create a common infrastructure that can be reused in refactoring tools for different languages. The Rephraser Engine is one part of this infrastructure: It contains base classes for refactorings, the virtual program graph (VPG), etc. *None of the classes in the Rephraser Engine know about Fortran, Photran, or CDT.* They are completely language-independent. However, many of the classes in Photran's refactoring engine inherit from base classes in the Rephraser Engine: They are "specialized" to work with Fortran.

In addition to Photran, Ralph Johnson's research group at UIUC is developing prototype refactoring tools for PHP, Lua, and BC; all of these use the Rephraser Engine. Therefore, if you ever need to change a class in the Rephraser Engine, you should be sure that the change applies to all of these tools, not just Photran.

2.3 Photran's Architecture: Overview and Dependencies

Figure 2.1 illustrates the plug-in decomposition of Photran and the Rephraser Engine, as well as their relationship with CDT and the Eclipse Platform. *Pay careful attention to the dependencies in this diagram.* If you are implementing a feature but need to introduce a dependency that is not present, you are probably implementing the feature in the wrong

component. It may be necessary to add an extension point to add your feature in the "correct" way. This should be discussed on the photran-dev mailing list.

The following sections are grouped by feature. A feature is a collection of related Eclipse plug-ins that the user can install as a whole.

2.4 Base Photran Feature: org.eclipse.photran-feature

The following projects comprise the "base" of Photran.

- **org.eclipse.photran.cdtinterface**

  This contains most of the components (core and user interface) related to integration with the CDT. It includes:

  - The FortranLanguage class, which adds Fortran to the list of languages recognized by CDT
  - Fortran model elements and icons for the Outline and Fortran Projects views
  - An extension point for contributing Fortran model builders, and a simple lexical analyzer-based model builder (which, generally speaking, is not used)
  - The Fortran perspective, Fortran Projects view, and other CDT-based parts of the user interface
  - New Project wizards and Fortran project templates

  For more information about CDT, see Chapter **??**.

- **org.eclipse.photran.core**

  This is the Photran Core plug-in. It contains much of the Fortran-specific "behind the scenes" functionality:

  - Utility classes
  - Error parsers for Fortran compilers
  - Fortran lexical analyzer
  - Workspace preferences

- **org.eclipse.photran.managedbuilder.core**,
  org.eclipse.photran.managedbuilder.gnu.ui,
  org.eclipse.photran.managedbuilder.ui

  Support for Managed Build projects using GNU Fortran (gfortran). Originally created by Craig Rasmussen at Los Alamos National Lab.

- **org.eclipse.photran.ui**

  This contains the Fortran-specific components of the user interface:

  - Fortran Editors
  - Preference pages

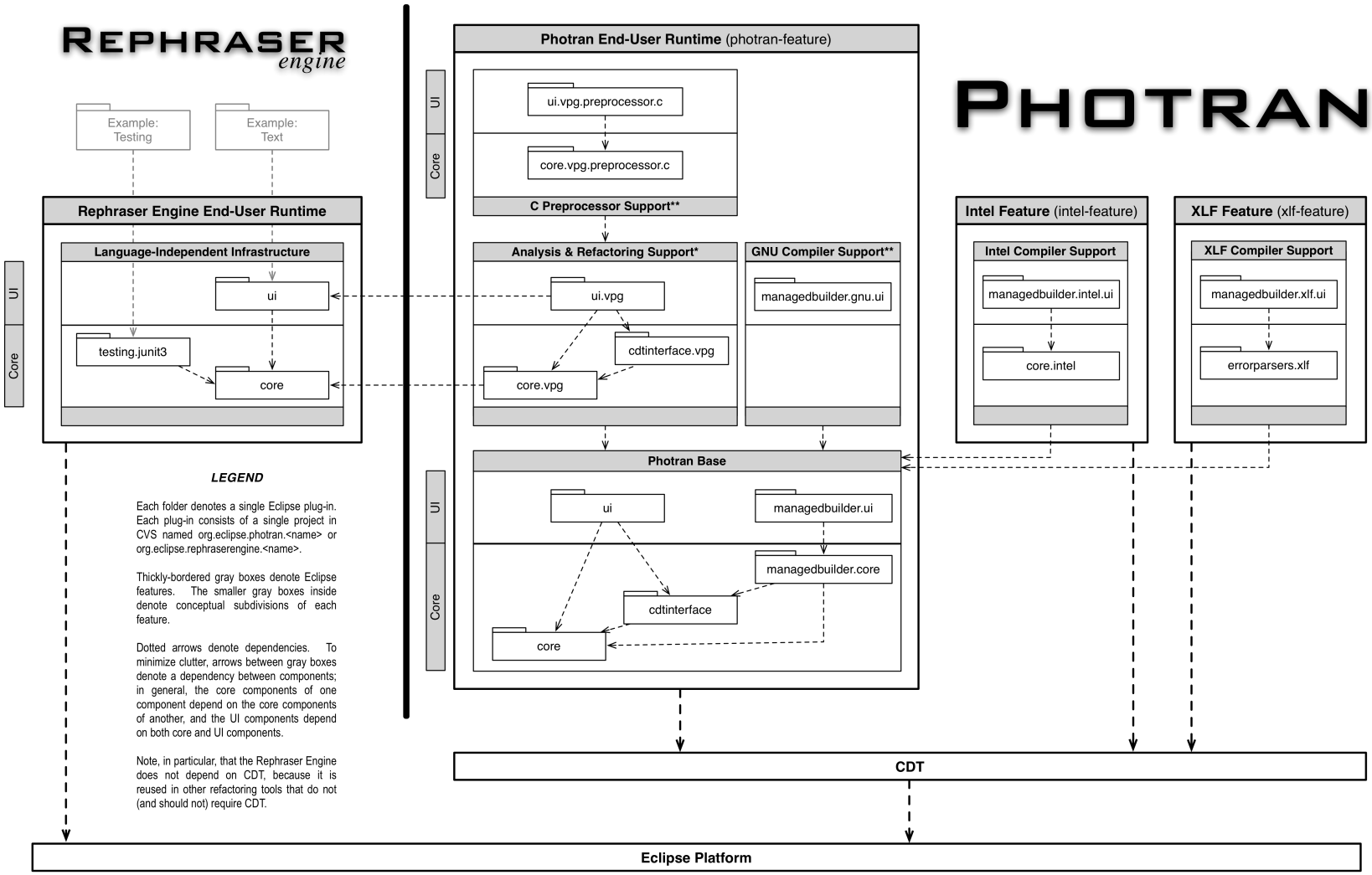

Figure 2.1: Plug-in decomposition of Photran and the Rephraser Engine

2.5 Virtual Program Graph (VPG) feature: org.eclipse.photran.vpg-feature

The following projects support parsing, analysis, and refactoring of Fortran sources. They are written in Java 5. The Virtual Program Graph is described in more detail in Chapter **??**.

- **org.eclipse.photran.core.vpg**

  This contains the parsing, analysis, and refactoring infrastructure.

  - Fortran parser and abstract syntax tree (AST)
  - Fortran preprocessor (to handle INCLUDE lines)
  - Parser-based model builder
  - Photran's Virtual Program Graph (VPG)
  - Utility classes (e.g., `SemanticError`, `LineCol`)
  - Project property pages
  - Name binding analysis (equivalent to symbol tables)
  - Refactoring/program transformation engine
  - Refactorings

- **org.eclipse.photran.core.vpg.tests**, **org.eclipse.photran.core.vpg.tests.failing**

  JUnit Plug-in tests for the VPG core plug-in.

  All tests in org.eclipse.photran.core.vpg.tests should pass. Test suites and test cases are placed in the "failing" plug-in project until all of their tests pass.

  These plug-ins *must* be run as a "JUnit Plug-in Test" (**not** a "JUnit Test"). In the launch configuration, the environment variable TESTING must be set to some non-empty value. (This ensures that the VPG will not try to run in the background and interfere with testing.)

- **org.eclipse.photran.ui.vpg**

  UI contributions that depend on the `org.eclipse.photran.core.vpg` plug-in. Currently, this includes the Open Declaration action, a project property page where the user can customize the search path for Fortran modules and include files, Fortran Search support, and all of the actions in the Refactor menu.

- **org.eclipse.photran.cdtinterface.vpg**

  This contributes the Fortran model builder which constructs the model seen in the Outline view and Fortran Projects view. It uses the Fortran parser contained in the org.eclipse.photran.core.vpg plug-in.

2.6 XL Fortran Compiler Feature: org.eclipse.photran.xlf-feature

The following are plug-ins to support the XL Fortran compiler.

- **org.eclipse.photran.core.errorparsers.xlf**, **org.eclipse.photran.managedbuilder.xlf.ui**

  Support for Managed Build projects using XL toolchains. Originally created by Craig Rasmussen at LANL.

2.7 Intel Fortran Compiler Feature: org.eclipse.photran.intel-feature

The following are plug-ins to support the Intel Fortran Compiler.

- **org.eclipse.photran.core.intel**, **org.eclipse.photran.managedbuilder.intel.ui**

  Support for Managed Build projects using Intel toolchains. Maintained by Bill Hilliard at Intel.

2.8 Rephraser Engine

The following plug-ins comprise the Rephraser Engine, a language-independent indexing and refactoring infrastructure that serves as the basis for Photran's refactoring engine. The Rephraser Engine will contribute a Rephraser Engine Plug-in Developer Guide to the runtime workbench (sourced in org.eclipse.rephraserengine.doc.isv); this provides more information about these plug-ins. Starred plug-ins are not distributed to end users.

- **org.eclipse.rephraserengine-feature**
- **org.eclipse.rephraserengine-support***
- **org.eclipse.rephraserengine.core**
- **org.eclipse.rephraserengine.core.refactoring**
- **org.eclipse.rephraserengine.core.vpg**
- **org.eclipse.rephraserengine.core.vpg.tests***
- **org.eclipse.rephraserengine.doc.isv***
- **org.eclipse.rephraserengine.examples.text***
- **org.eclipse.rephraserengine.examples.testing***
- **org.eclipse.rephraserengine.testing.junit3***
- **org.eclipse.rephraserengine.ui**
- **org.eclipse.rephraserengine.ui.refactoring**
- **org.eclipse.rephraserengine.ui.vpg**

2.9 Non-plug-in projects

The following projects are in CVS but are not distributed to users:

- **org.eclipse.photran-dev-docs**

  Developer documentation, including this document (`dev-guide/*`), CVS instructions (`dev-guide/cvs-instructions.`
  the materials from our presentation at EclipseCon 2006 on adding a new language to the CDT, and a spreadsheet
  mapping features in the Fortran language to JUnit tests (`language-coverage/*`).

- **org.eclipse.photran-samples**

  A Photran project containing an assortment of Fortran code.

Appendix A

Getting the Photran 5.0 Sources from CVS

*BEFORE YOU BEGIN: Make sure you are running **Eclipse 3.5** (Galileo) and a **Java 5** or later JVM. We recommend the Eclipse for RCP/Plug-in Developers Package.*

Revision: $Id: app-cvs.ltx-inc,v 1.9 2010/04/28 18:12:52 joverbey Exp

Part I. Check out the CDT 6.0 sources from CVS

1. In Eclipse, switch to the CVS Repository Exploring perspective.
2. Right-click the CVS Repositories view; choose New, Repository Location
3. In the dialog box, enter the following information, then click Finish.

Host name:	dev.eclipse.org
Repository path:	/cvsroot/tools
Username:	anonymous
Password:	(no password)
Connection type:	pserver

4. In the CVS Repositories view
   - Expand ":pserver:anonymous@dev.eclipse.org:/cvsroot/tools"
   - Then expand "HEAD"
5. Right-click on "org.eclipse.cdt"
6. Select "Configure Branches and Versions..."
7. Under "Browse files for tags", expand "all", then expand "org.eclipse.cdt", then click on the .project file
8. Under "New tags found in the selected files", click on the Deselect All button, then check cdt_6_0 in the list above it
9. Click Add Checked Tags
10. Click OK
11. Now, in the CVS Repositories view
    - Expand ":pserver:anonymous@dev.eclipse.org:/cvsroot/tools"
    - Then expand "Branches"
    - Then expand "cdt_6_0"

- Then expand “org.eclipse.cdt cdt_6_0”
- Then expand “all”

12. Click on the first entry under “all” (it should be org.eclipse.cdt), then shift-click on the last entry under “all” (it should be org.eclipse.cdt.ui.tests). All of the intervening plug-ins should now be selected. Right-click on any of the selected plug-ins, and select Check Out from the pop-up menu. (Check out will take several minutes.)
13. You now have the CDT source code. Make sure it compiles successfully (lots of warnings, but no errors).

Part II. Check out the Photran sources from CVS

14. Under “:pserver:anonymous@dev.eclipse.org:/cvsroot/tools,” expand HEAD, then expand org.eclipse.ptp, then expand photran
15. Click on the first entry under “photran” (it should be org.eclipse.photran-dev-docs), then shift-click on the last entry under “photran” (it should be org.eclipse.rephraserengine.ui.vpg). All of the intervening plug-ins should now be selected. Right-click on any of the selected plug-ins, and select Check Out from the pop-up menu. (Check out will take several minutes.)

The sources should all compile (albeit with lots of warnings).

Appendix B

Running Photran's Automated Test Suite

Revision: $Id: app-tests.ltx-inc,v 1.3 2010/04/28 18:12:52 joverbey Exp

1. In the Package Explorer view, select the `org.eclipse.photran.core.vpg.tests` project.
2. Right-click on that project and select Run As > Run Configurations.... A dialog will appear.
3. In that dialog, create a new **JUnit Plug-in Test** launch configuration. Call it "Photran-Tests".
4. For the configuration that you have just created, switch to the "Arguments" tab.
5. Change the "VM arguments" field to `-ea -Xms40m -Xmx512m`
6. Switch to the "Environment" tab.
7. Create a new environment variable called `TESTING` with a value of 1.
8. *(Optional)* If you are running Linux or Mac OS X and have gfortran installed:

   Some of Photran's refactoring unit tests can attempt to compile and run the Fortran test programs before and after the refactoring in order to ensure that the refactoring actually preserves behavior (and produces code that compiles). The following steps will enable this behavior. Note, however, that if the path to gfortran is incorrect, or if gfortran cannot be run successfully, it will cause the test suite to fail... so you might not want to do this the very first time you attempt to run the test suite.

   (a) Create a new environment variable called `COMPILER` with the full path to gfortran. This will be something like `/usr/local/bin/gfortran`

   (b) Create a new environment variable called `EXECUTABLE` with a path to some non-existent file in your home directory, e.g., `/Users/joverbey/a.out`. When gfortran is run, it will write the executable to this path.

   When both of these environment variables are set, you will be able to see the output from the compiler and the Fortran program in the Console view as the test suite runs. If compilation fails, or if the Fortran program exits with a nonzero status code, or if the Fortran program does not produce the same output before and after refactoring, the corresponding JUnit test will fail. See the JavaDoc for the method `RefactoringTestCase#compileAndRunFortranProgram` for more details.

9. Click the "Run" button to run the tests. It will take at least a minute to run the test suite. When it finishes, you should get a green bar in the JUnit view. If you get a red bar, some of the tests failed; the JUnit view will have details.

10. To run the tests again later, just launch the "Photran-Tests" configuration from the Eclipse Run menu.

    Note. *Some parser tests will attempt to look for closed-source code that you may not have. A warning will appear in the JUnit runner if this code is not available, but all tests should still pass. UIUC personnel: See the appendix Additional Information for UIUC Personnel in the Photran Developer's Guide for more information.*

Appendix C

Additional Information for UIUC Personnel

Revision: $Id: app-uiuc.ltx-inc,v 1.2 2010/04/28 18:12:52 joverbey Exp

The following information is only of interest to developers in the Department of Computer Science at the University of Illinois at Urbana-Champaign.

C.1 Photran Subversion Repository at UIUC

A private Subversion repository for internal testing and research work is provided by TSG at the following URL. *You will only be able to access this repository if you have specifically been granted access.* The repository requires authentication using your NetID and Kerberos password. To access a Subversion repository from Eclipse, you must install an additional Eclipse plug-in (either Subclipse or Subversive).

`https://subversion.cs.uiuc.edu/svn/Photran/`

Currently, this repository contains the following.

- **Large Fortran test codes (photran-projects).** Checking these out into your workspace will enable about 6,000 additional test cases in the JUnit test suite. *These applications are PRIVATE. They may NOT be distributed and are intended only for testing Photran.*
- **Sample Obfuscation refactoring (sample-refactoring).** The sample refactoring described in Appendix **??**.
- **Refactoring tools for PHP, Lua, and BC.** Prototype Eclipse-based refactoring tools based on the Rephraser Engine.

C.2 Photran Web Site at UIUC

TSG provides a Web hosting infrastructure based on the popular CPanel system and has set up a web hosting account for Photran. This hosting environment supports PHP, MySQL databases, CGI scripts, and a number of other popular web technologies. Documentation for these features is provided in the control panel. You can access your web content via the control panel, SSH, or WebDAV. Here are the account details:

```
      web URL: http://photran.cs.illinois.edu/
control panel: https://cpanel.cs.illinois.edu:2083/
   SSH access: cpanel.cs.illinois.edu
     web root: ~/public_html
```

Appendix D

Procedure for Contributing a New Feature or Bug Fix

Revision: $Id: app-contributing.ltx-inc,v 1.2 2010/04/28 18:12:52 joverbey Exp

1. Run Photran's automated test suite. All tests must pass.
2. If you are contributing a refactoring, program analysis, or similar complex feature, please include JUnit tests in your contribution. The parser and AST inevitably change over time, and this is how we will determine whether or not a change has broken your contribution.
3. Make sure you did not copy code from anywhere except Photran and CDT. If you copied code from books, mailing lists, Web sites, any other open source projects, other projects at your company, etc., **stop** and ask for further instructions. (If you do not know a Photran committer personally, ask on the photran-dev mailing list.) It may or may not be possible to contribute your code.
4. Determine who owns the copyright to your code. Generally, if you are an employee and you were paid to write the code, it is the property of your employer. If you are a student, the code may be your personal property, or it may be the property of your university, depending on several factors. Always check with your employer or university to determine who the copyright owner is. The following examples assume that the code was written by John Doe as an employee of the XYZ Corporation.
5. Your code must be contributed under the terms of the Eclipse Public License (EPL). If the copyright is owned by your employer or university, make sure they will permit you to contribute your code under the EPL. (They will probably be asked for verification by the Eclipse Legal team.)
6. Ensure that *every* Java file you created or modified has an accurate copyright header.
   - If you created the file from scratch (i.e., it is not a modification of someone else's code), the copyright header must name the copyright owner and list them as the initial contributor. For example:

```
/*******************************************************************************
 * Copyright (c) 2010 XYZ Corporation and others.
 * All rights reserved. This program and the accompanying materials
 * are made available under the terms of the Eclipse Public License v1.0
 * which accompanies this distribution, and is available at
 * http://www.eclipse.org/legal/epl-v10.html
 *
 * Contributors:
 *    John Doe (XYZ Corporation) - Initial API and implementation
 *******************************************************************************/
```

- If you modified an existing file from Photran or CDT, it must retain the *original* copyright notice, but you should add yourself as a contributor at the bottom. For example:

```
/*******************************************************************************
 * Copyright (c) 2004, 2008 IBM Corporation and others.
 * All rights reserved. This program and the accompanying materials
 * are made available under the terms of the Eclipse Public License v1.0
 * which accompanies this distribution, and is available at
 * http://www.eclipse.org/legal/epl-v10.html
 *
 * Contributors:
 *    IBM - Initial API and implementation
 *    John Doe (XYZ Corporation) - Photran modifications
 *******************************************************************************/
```

7. Every Java class should have a JavaDoc comment with a brief description of the class and an `@author` tag with your full name. Again, if you modified an existing class, add an `@author` tag with your name. For example:

```
/**
 * Refactoring which creates getter and setter subprograms for a module variable and replaces
 * variable accesses with calls to those subprograms.
 * <p>
 * This refactoring accepts a module variable declaration, makes that declaration PRIVATE, adds
 * getter and setter procedures to the module, and then replaces accesses to the variable outside
 * the module with calls to the getter and setter routines.
 *
 * @author Tim Yuvashev
 * @author Jeff Overbey
 */
public class EncapsulateVariableRefactoring
{ ...
```

8. Create a new bug in Bugzilla on one of the "Photran" components. You can do this by clicking the "Report a Bug" link on the Photran home page. If your code adds a new feature (rather than fixing a bug), set the severity of the bug to "Enhancement."

9. Create a patch with the files that you changed. (Highlight all of the projects you changed in the Package Explorer, right-click, and choose Team > Create Patch.) Attach the patch to the bug in Bugzilla. If you have any binary files (e.g., images), DO NOT include them in the patch; attach them to Bugzilla separately.

10. A Photran committer will review the code and may ask you to make changes. If so, you will need to create a new patch, attach it to the bug, and mark the old patch as "obsolete."

11. When the code review succeeds, the Photran committer will make the following comment on the bug:

    Please confirm that

    (a) you wrote 100% of the code without incorporating content from elsewhere or relying on the intellectual property of others,

    (b) you have the right to contribute the code to Eclipse, and

    (c) you have included the EPL license header in all source files?

    You can reply with something as simple as the following.
    I confirm that

    (a) I wrote 100% of the code without incorporating content from elsewhere or relying on the intellectual property of others,

    (b) I have the right to contribute the code to Eclipse, and

    (c) I have included the EPL license header in all source files.

    However, if you *did* incorporate content from elsewhere – e.g., if your contribution is based on code from CDT or elsewhere – DO NOT copy-and-paste this directly; change the first statement to note this explicitly.

12. Your code will be passed on to the intellectual property (IP) team at the Eclipse Foundation for a legal review. If there are any questions or concerns about the code, a member of the Eclipse IP team will contact you.

13. Once it passes IP review, the committer will commit your code to Photran's CVS repository.

14. Your name will be added on the "Contributors" page of the Photran Web site.

Eclipse Foundation References:
http://wiki.eclipse.org/Development_Resources#Everyone:_IP_Cleanliness
http://wiki.eclipse.org/Development_Conventions_and_Guidelines