

## **README file for Perfect Developer version 3.0**

### **CONTENTS**

0. Version Identification
1. Platform Requirements
2. Installation
3. Documentation
4. Editor customization
5. Major changes since version 2.10
6. Additional Changes since version 2.0
7. Reporting a problem
8. Target language compilers
9. Updates
10. Acknowledgements

### **0. VERSION IDENTIFICATION**

This Readme file is for *Perfect Developer* 3.0. The *PerfectDeveloper* executable, *PDTTool* executable and main documentation files all have version number 3.00 or 3.00.xx where xx is a minor revision number.

### **1. PLATFORM REQUIREMENTS**

*Perfect Developer* is intended to be installed on a PC with a 500MHz or faster processor, at least 256Mb of memory, and a Windows 2000 or Windows XP or Linux operating system. Operation under some other operating systems (e.g. Windows NT 4 with Service Pack 6, or Windows 98) or on less capable computers is possible but unsupported.

### **2. INSTALLATION**

The following notes refer to installation under Windows. For installation under Linux, see the separate *readme-linux.txt* file.

If you have a version of *Perfect Developer* prior to 2.08 installed, we recommend that you uninstall it (using the Windows Add/Remove programs function in the Control Panel) before installing this version. This is because the directory structure and registry entries have changed in this version. You may, however, carry your own project files and source files over from version 2.0.

To install *Perfect Developer* from a CD or other distribution media, insert the CD into the drive. If the install program does not start automatically, open My Computer or Windows Explorer, then browse to and run the file *Autorun.exe* on the CD.

### **3. DOCUMENTATION**

The following HTML documents are installed (some are supplied in PDF format as well):

- **User Guide** describes how to use the Project Manager, which is the main user interface to *Perfect Developer*..
- **Language Reference Manual** describes the *Perfect* language.

- **Development Overview** gives guidelines on how to develop software systems using *Perfect Developer*.
- **Language Introduction** introduces the *Perfect* language by way of examples. A more comprehensive introduction is to be found in the tutorial on our web site at [http://www.eschertech.com/support/perfect\\_developer\\_self\\_help.php](http://www.eschertech.com/support/perfect_developer_self_help.php).
- **Creating a Java application using Perfect Developer and JDK 1.4** describes how to build an application using *Perfect Developer* compiled to Java.
- **Interfacing a Java graphical user interface to a Perfect application** follows on from the previous document, explaining how to attach a Java graphical front-end to a *Perfect* application.

Other documents are available on our web site via the *Support* link.

#### 4. EDITOR CUSTOMIZATION

*Perfect Developer* does not have its own editor. Instead, we allow you to configure the Project Manager to invoke a third-party editor of your choice. This configuration mechanism is accessed via the *Options* menu of the Project Manager. If you do not configure an editor, *Perfect Developer* will try to use Notepad.

The Windows version of the *Perfect Developer* distribution disk includes evaluation copies of two editors (Crimson and TextPad). Linux users have access to a wide range of open-source editors.

We strongly recommend that you use an editor that supports syntax highlighting for the *Perfect* language. To this end, we provide editor customization files for a number of popular editors (XEmacs, Vim, Multi-Edit 9, TextPad 4, Crimson and Kate). For each editor, you will find the customization files in an appropriate subfolder of the *EditorCustomizations* subfolder of the *Perfect Developer* installation directory, together with a *Readme.txt* file explaining how to use them.

#### 5. MAJOR CHANGES SINCE VERSION 2.10

##### (a) Installation changes

[Windows only] The install program is now based on the Microsoft Installer, making the task of installation on a Windows network easier.

[Windows only] The default installation directory is now "C:\Program Files\Escher Technologies\Perfect Developer".

[Windows only] An evaluation copy of Multi Edit is no longer included on the CD. Instead, the installation CD includes an evaluation copy of Text Pad 4 and a copy of Crimson Editor.

[Linux only] C++ and Java runtime libraries are now included.

[Linux only] Installation directories have been changed to conform to the Linux Standards Base and the associated Filesystem Hierarchy Standard.

[Linux only] It is no longer necessary to edit the global configuration file after installation, provided that version 2.4.2 of wxGTK is used.

Syntax configuration files for the Crimson editor (Windows) and the Kate editor (Linux KDE) are

now provided.

The HelloWorld and Graphical example projects now use a common post build batch or script file to process the generated Java. The manifest files are no longer part of the installation but are generated on the fly.

### **(b) Functional changes to the Project Manager**

[Windows only] In the dialogs for selecting a directory, a *Create new* button is now provided.

[Linux only] In the dialogs for selecting a directory, the *Create new* button is no longer provided (because of limitations of wxGTK).

The *Add*, *Create* and *Remove* buttons in the *Files* pane have been replaced by tool buttons in the toolbar. An *Import UML* tool button has also been added.

The *Options...Editor* dialogue now allows you to automatically set the command line parameters needed by your editor by selecting the appropriate editor type. You can still enter the command line parameters manually by selecting *other*.

The proof- and unproven-output dialogs now allow the output detail to be set to *full* or *abbreviated*.

When performing a build, batch build or rebuild all, the Project Manager now checks for and reports non-zero exit codes from the pre- and post-build steps. If the pre-build step terminates with a non-zero exit code, the build is terminated.

The pre- and post-build commands are no longer restricted to being filenames, allowing them to refer to commands on the default search path and to include parameters.

Before running pre- and post-build commands, the Project Manager sets up some environment variables to allow the commands to determine the Java package name and the paths to the Java tools and the *Perfect Developer* libraries (see the User Guide for details). This allows a common post-build script to be used for multiple projects.

The *Options...Miscellaneous* dialog now allows the path to the Java tools to be configured, so that it can be made available to pre- and post-build commands.

### **(c) Functional changes to the compiler/verifier**

When running under Windows NT, Windows 2000 or Windows XP, the prover time limits now apply to the CPU time used, not elapsed wall-clock time. Under Windows 98 and Windows ME they continue to apply to elapsed wall-clock time.

A new benchmarking option *-mb* has been added to PDTool. It causes the times taken by the various phases to be reported to 0.001 second instead of in whole seconds.

Code can now be generated and/or verification attempted even if a source file contains loop variants or recursion variants with components of the form '?'. No run-time checks will be generated for such variants, and no proofs attempted; but a "Specification incomplete" warning message will be generated during verification.

### **(d) Language changes**

The following are now reserved words: **associative commutative idempotent identity pragma trace**.

When declaring a binary operator with no precondition, the operator can be declared any or all of **associative**, **commutative** and **idempotent**. In addition, a left **identity** expression can be defined (i.e. an expression  $E$  for which  $E \text{ op } X = X$  for all expressions  $X$ ). Verification conditions are generated to ensure that the properties declared are true. Example:

```
final class XYvector ^=
abstract
...
interface
  operator + (arg: XYvector): XYvector
    associative commutative identity XYvector{0, 0}
    ^= ...;
...
end;
```

The  $\text{op over collection}$  expression no longer has the precondition that *collection* is not empty if a left identity has been declared for *op*; the result when *collection* is empty is defined as the left identity of *op*. All library operators are declared with the appropriate set of properties. So you can now use e.g.  $+ \text{ over}$  to sum a possibly empty collection of integers, without having to handle the case of an empty collection specially. Likewise,  $++ \text{ over}$  a set of sets, bag of bags or sequence of sequences no longer requires a separate check for an empty collection.

In an  $\text{op over}$  expression over a set or bag, *op* must have been declared **associative** and **commutative**, otherwise an error is reported. Previously, verification conditions were generated at the point of the  $\text{op over}$  expression to assert associativity and commutativity.

*Perfect* now supports nondeterministic guarded choice in expressions and postconditions. This is expressed by using the normal syntax for a conditional expression or conditional postcondition except that the keyword **opaque** is inserted immediately before the first guard, e.g. "( **opaque** [guard1]: expression1, [guard2]: expression3)". An empty guard is not permitted when **opaque** is used. The semantics is that of nondeterministic choice between those expressions or postconditions whose guards are satisfied. One use of this facility is in translating Z specifications. For example, if P and Q are schemas that have been translated from Z to *Perfect*, the Z construct  $P \vee Q$  can be translated as "( **opaque** [p]: !P, [q]: !Q)" where p and q are the preconditions of P and Q respectively. **Note**: implementation of this feature is not yet complete! In particular, if an opaque conditional postcondition is used in parallel with another conditional postcondition (whether opaque or not), and the guard expressions allow either one (but not both) to modify a common variable (or part of a variable), then the conditions under which variables remain unchanged may be computed incorrectly, most likely rendering some verification conditions unprovable.

Multiple heaps may now be declared following the **heap** keyword. Example:

```
heap mainHeap, temporaryHeap;
```

### (e) Library changes

Class **bag of**  $X$  no longer requires the element type  $X$  to provide non-ghost equality (although some members such as the binary '#' operator do require non-ghost equality).

Methods *stdIn*, *stdOut* and *stdErr* of class *Environment* now return streams instead of file descriptors, and are available in the Java library as well as the C++ library. New stream classes *StandardInputStream* and *StandardOutputStream* have been introduced.

Class **seq of**  $X$  now provides methods for inserting into ordered sequences and merging ordered sequences.

Global function *intln* has been provided to return the logarithm to base 2 of its operand, rounded down to the nearest integer.

Methods *permndec* and *permninc* of classes **set of X**, **bag of X** and **seq of X** now require that the element type has a total '~~' operator. Alternative opaque methods *opermndec* and *opermninc*, which do not have this requirement, are also provided.

Methods *max* and *min* of classes **set of X** and **bag of X** now require that the element type has a total '~~' operator. Alternative opaque methods *omax* and *omin*, which do not have this requirement, are also provided.

When running under Windows NT, Windows 2000 or Windows XP, method *clock* of class *Environment* now returns the total CPU time used by all threads of the current process since the program started. Under Windows 98 and Windows ME it continues to return elapsed wall-clock time.

#### (f) Issued fixed

Refer to our web site for a list of issues fixed in this release.

#### (g) Other improvements

Many changes have been made to speed up the prover and to increase the range of problems that it can solve.

The time taken to initialize the compiler/verifier prior to parsing source files has been substantially reduced.

## 6. ADDITIONAL CHANGES SINCE VERSION 2.00

### (A) Incompatibilities with previous versions

#### i) *Perfect* language changes since v2.00

The *context* parameter of schema *main* is now a **limited** in-out parameter instead of a full in-out parameter. You will need to change your declaration of schema *main* accordingly (i.e. replace the parameter *context!*: *Environment* by *context!*: **limited** *Environment*).

A postcondition that involves the **then** keyword cannot be concurrently satisfied with another postcondition. The compiler will report an error if you try to do so. However, an operand of **then** may still comprise several concurrently-satisfied postconditions.

Previously, the **satisfy** part of a **change...satisfy** postcondition could not contain multiple comma-separated conditions. The syntax has now been changed so that this is allowed. However, when comma is used to separate a **change...satisfy** postcondition from another concurrently-satisfied postcondition, the **change...satisfy** postcondition must now be enclosed in brackets.

A new binary operator symbol '##' is now available, so if you previously used an expression like 'a ## b' (meaning 'a # (#b)') or '##a' (meaning '# (#a)'), you will now have to use the extra brackets. The symbol '%%' may also be defined as a binary operator.

The specification of the *append* members of the mapping class has been changed. Previously these members had a precondition requiring that if the domain-part of the appended pair was already in the mapping, the range-part had to be equal to the corresponding range-part in the

mapping (so that the append operation would do nothing). Now, the precondition is gone and the new pair of values overrides any existing pair with the same domain element.

## ii) Library changes since v2.00

A change has been made to the specification of the *split* member of class **seq of X**. Previously, zero-length segments (i.e. between two adjacent separator tokens) were not included in the result sequence; now they are included in the result. The old behavior may be obtained by replacing *x.split(y)* by **(those *r::x.split(y) :- ~r.empty*)**.

## iii) Other

The project file format has changed. This means that projects saved under version 2.10 of Perfect Developer cannot be opened using older versions of Perfect Developer.

## (B) New features and improvements

i) Project Manager facilities added to open proof, unproven and error files from the File menu or by right-clicking on a source file.

ii) The prover has been further improved in power, cutting the number of unproven but provable proof obligations.

iii) Proof output is now easier to read. Simplification steps are combined and summarized by default. Use option "-vpl=1" (that ends with "ell equals one") to include the detail.

iv) New library methods added:

- Binary operator **##** is predefined for sets, bags and mappings with meaning 'disjoint'.
- Boolean member function *empty* is defined for sets, bags, sequences and mappings such that *x.empty* means the same as *#x = 0*.
- Members *findFirst* and *findLast* are defined for sequences.
- Members *!sort* and *isOrdered* are available. Both these classes take a parameter of type **from Comparator of X** so that the ordering schema can be defined by the caller.
- *toString* is now defined for sets, bags, sequences and mappings.
- Stream classes have been extended and improved.
- Classes *FilePath* and *OsInfo* now redefine the *toString* method.

v) The C++ and Java code generators will now generate a 'switch' statement instead of a multi-way conditional, where appropriate.

vi) The Java package name may now be configured on the Code Generation tab of the project settings. It is no longer necessary to use the *-gk* directive to set the package name.

vii) A toolbar has been added to the Project Manager. The *Stop* and *Clear Window* buttons have been removed because these functions are now available via the menu or the toolbar.

viii) Appendices to the Language Reference Manual now cover the class library and provide a complete LALR(1) grammar for Perfect.

ix) A syntax definition file for the TextPad 4 editor is now provided.

x) Files in a project that are contained within the directory holding the .pdp project file or its subdirectories are now stored using relative path names. This means that a project directory can be moved to a different location without the filenames in the project becoming incorrect.

## 7. REPORTING A PROBLEM

If you encounter a problem with *Perfect Developer*, please check the following before reporting it or requesting support:

- You have registered your copy of *Perfect Developer* at <http://www.eschertech.com/register.htm> or by telephoning +44 (0) 1252 336565;
- You are using a compatible operating system and hardware platform;
- The problem is not already listed as a known limitation on our web site.

Report *Perfect Developer* problems using the problem report form on our web site <http://www.eschertech.com>.

## 8. TARGET LANGUAGE COMPILERS

If you are targeting the Gcc compiler, the *Target compiler* option must be set to *Gcc* when generating code (this is under the *Code generation* tab of the *Settings* property sheet in the Project Manager, or use option -gv=Gcc when running PDTool from the command line). Not using this option may cause the resulting executable to be unstable due to bugs in the Gcc compiler versions 3.01 and earlier. It is not yet known whether these have been fixed in later versions of Gcc.

See the support section of our web site (<http://www.eschertech.com>) for issues regarding other compilers.

## 9. UPDATES

From time to time we may make software updates available for download. Notice of availability of such updates will be posted on our web site and emailed to registered users.

## 10. ACKNOWLEDGMENTS

We wish to thank the following for their contributions to the *Perfect Developer* package:

- All those who contributed to wxWidgets (formerly wxWindows) - a professional, open source, cross-platform graphical user interface library;
- Tarma Software Pty Ltd. for providing the Autorun.exe file;
- Ingyu Kang for allowing us to distribute Crimson Editor;
- Helios Software Ltd. for allowing us to distribute an evaluation version of TextPad 4.

**Escher Technologies Limited, 01 December 2004.**