Predictor
Users manual
v1.0

Copyright © Henrik Bergström
DSV SU/KTH 2000

Table of Contents

1 INTRODUCTION 1

1.1. READING THIS MANUAL 1
1.2. CONTACT INFORMATION 2

2 SETTING UP PREDICTOR ON YOUR SYSTEM 3

2.1. SYSTEM REQUIREMENTS 3
2.2. DOWNLOADING AND INSTALLING PREDICTOR 3
2.3. REGISTERING PREDICTOR 4
2.4. USING PREDICTOR WITH A COMMON DATABASE 4

3 COST ESTIMATION TECHNIQUES SUPPORTED 5

3.1. EXPERT JUDGEMENT 5
3.2. ALGORITHMIC MODELS 5
3.3. CASE-BASED REASONING 6
3.4. ESTIMATION VS PREDICTION 6

4 THE MAIN WINDOW 7

4.1. THE TOOLBAR 8
4.2. THE MENUBAR 9

5 A GUIDED TOUR 11

5.1. CREATING A PROJECT DATABASE 11
5.2. RECORDING PROJECT DATA 13
5.3. USING PREDICTOR WITH VERY LITTLE DATA 14
5.4. MAKING A CBR PREDICTION 15
5.5. THE CBR CONTROL PARAMETERS 17
5.6. IMPORTING DATA 18
5.7. JACKKNIFING AND BATCH TESTS 19
5.8. CREATING AN ALGORITHMIC MODEL 20
5.9. CHECKING A RULE OF THUMB 21
5.10. GETTING FEEDBACK ON THE QUALITY OF ONES ESTIMATES 22
5.11. CREATING A NEW DATABASE TYPE 23

6 PRINTING REPORTS 25

6.1. DATABASE CONTENTS REPORT 25
6.2. PREDICTION ACCURACY REPORT 26
6.3. EARLIER PREDICTIONS REPORT 26
6.4. CBR PREDICTION REPORT 27
6.5. CBR JACKKNIFING RESULTS REPORT 27
6.6. CBR BATCH TEST RESULTS REPORT 28
6.7. WEIGHT FILE REPORT 29
6.8. ALGORITHMIC MODEL REPORT 29
6.9. ALGORITHMIC MODEL PREDICTION REPORT 30
6.10. ALGORITHMIC MODEL JACKKNIFING RESULTS REPORT 30

7 IMPORT FILTERS 31

7.1. DELPHI INTERFACE OF AN IMPORT FILTER 31
7.2. C/C++ EXAMPLE FILTER 34

8 REFERENCES 39

APPENDIX A - METRICS OF ESTIMATION ACCURACY 41

APPENDIX B - END USER LICENCE AGREEMENT 43

INDEX 45

Figures

FIGURE 1 PREDICTORS MAIN WINDOW 7
FIGURE 2 THE EMPTY MAIN WINDOW 11
FIGURE 3 THE NEW DATABASE WINDOW 12
FIGURE 4 PREDICTOR WITH AN EMPTY EXAMPLE DATABASE 12
FIGURE 5 PREDICTOR AFTER ADDING A FEW PROJECTS 13
FIGURE 6 MANUAL PREDICTION/ADJUSTMENT WINDOW 14
FIGURE 7 SET ATTRIBUTE WEIGHTS WINDOW 15
FIGURE 8 THE CBR PREDICTION TAB 16
FIGURE 9 THE ANALOGIES TAB 17
FIGURE 10 THE SET PREDICTION ATTRIBUTES WINDOW 18
FIGURE 11 THE IMPORT DIALOG 18
FIGURE 12 DEVELOP ALGORITHMIC MODEL WINDOW 20
FIGURE 13 THE ALGORITHMIC TAB 21
FIGURE 14 THE SEARCH DIALOG 22
FIGURE 15 DEVELOPING A RULE OF THUMB 22
FIGURE 16 THE PREDICTION ACCURACY WINDOW 23
FIGURE 17 DATABASE DESIGN WINDOW 24
FIGURE 18 NEW DATABASE FIELD DIALOG 24
FIGURE 19 THE PRINT DIALOG 25
FIGURE 20 DATABASE CONTENTS REPORT 25
FIGURE 21 PREDICTION ACCURACY REPORT 26
FIGURE 22 EARLIER PREDICTIONS REPORT 26
FIGURE 23 CBR PREDICTION REPORT 27
FIGURE 24 CBR JACKKNIFING RESULTS REPORT 28
FIGURE 25 CBR BATCH TEST RESULTS REPORT - DATA 28
FIGURE 26 CBR BATCH TEST RESULTS REPORT - GRAPHS 29
FIGURE 27 WEIGHT FILE REPORT 29
FIGURE 28 ALGORITHMIC MODEL REPORT 30
FIGURE 29 ALGORITHMIC MODEL PREDICTION REPORT 30
FIGURE 30 ALGORITHMIC MODEL JACKKNIFING RESULTS REPORT 30

1 Introduction

It is evidently difficult to develop large software systems in time, within budget, and with the agreed quality. The software engineering community seems today to commonly agree that the focus should be shifted from single methods, tools, and languages to efforts towards improving the entire development process including project management. This can be clearly seen in the interest that is given to process improvement programs such as ISO 9000 and CMM.

Many people also believes that such an organisation wide improvement program should be accompanied with a program on the individual level such as e.g. the Personal Software Process [Humphrey, 1995].

Regardless of the model for the improvement effort, however, it is necessary to establish a baseline in terms of quality attributes of the processes, products, and resources, and to compare new data with the baseline, i.e. an improvement program implies measurement as a normal activity in an organisation. As more and more companies undertake process improvement efforts, and quality programs, it is realistic to expect that in the future, more and more companies will record measurement data as part of their improvement efforts.

The measurement data collected and recorded for assessment could also facilitate another candidate area for improvement, namely project management. In project planning and monitoring, the project manager must estimate time, cost, and quality, and explore the relationship them between. Prediction models based on measurement data from past and current projects would increase the correctness of the estimates and serve as decision aid to make corrective actions.

The situation outlined above was part of the motivation for the project *Prediction Support for Project Management* undertaken by the software engineering research group at the Department of Computer and Systems Sciences, Stockholm University and the Royal Institute of Technology. The project is sponsored by the Swedish National Board for Industrial and Technical Development - NUTEK, and a group of industrial partners.

The goal of the project is to develop prediction models for project planning and monitoring based on measurement data, and to implement them in a project management tool set. The result of the work on this tool set is Predictor, a measurement database and prediction support tool intended to be used by project managers to keep track of projects. The information stored in the database can then be used to improve the estimation of various project features such as development time and fault density for new projects.

1.1. Reading this Manual

If Predictor is not yet installed on your system we recommend that you read Section 2.2 Downloading and Installing Predictor before you proceed.

After you have installed the program, run it and open the example database that is part of the distribution. It is located in the *tutorial* directory under the name *exampledb.pre*. Play around with this database while reading Section 4 that describes the functionality of the main window. This is the most important part of Predictor, and you need to be familiar with the functionality offered here to be able to use the program.

When you have gained some familiarity with the main window, you should proceed to Section 5, where you will be taken on a guided tour of Predictor. The tour introduces all the main topics of the program, and when you have gone through it you will be able to use the tool efficiently.

1.2. Contact Information

If you have any questions, comments, suggestions etc, do not hesitate to contact us. The address is:

Predictor Development Team
Henrik Bergström
Stockholm University/KTH
Department of Computer and Systems Sciences
Electrum 230
S-164 40 Kista, Sweden

Web: http://www.dsv.su.se/~henrikbe/Predictor
Email: predictor@dsv.su.se
Fax: +46 8 703 90 25

2 Setting up Predictor on your System

2.1. System Requirements

Predictor is a 32-bit application and runs under the following operating systems:

- Windows 95
- Windows 98
- Windows NT
- Windows 2000

Any machine with one of these operating systems and at least 10 MB of free hard drive space should be capable of running the program.

2.2. Downloading and Installing Predictor

The easiest way to get a copy of Predictor is to download it from the program's homepage: *http://www.dsv.su.se/~henrikbe/Predictor*. From the homepage you can also request a serial number which is necessary to be able to run the program. If you for some reason are unable to use this method do not hesitate to contact the Predictor Development Team at the address given in Section 1.2 to get a copy by other means. A nominal fee is charged for covering distribution media and postage of hardcopies, apart from that there are *no costs* for using Predictor.

There are two different distributions available of the program. The only difference between them is that one contains the Borland Database Engine (BDE), and the other does not. The BDE is necessary to run the program and most users should use the complete distribution. However, if you already are using one of Inprise programming environments you will probably not need the BDE and can thus save some download time by using the smaller distribution.

Both distributions comes in the form of a zip-file containing a standard Windows setup-progam and its datafiles. To get the program up and running follow these steps:

1. Unzip the distribution file. Exactly how this is done depends on how your system is configured. In most cases though you just has to double-click on the file to open the winzip-program. An alternative method, if this does not work, is to issue the command "unzip distribution_filename" or "pkunzip distribution_filename" from the command prompt. If neither of these works you have to contact your system administrator for information on how to unzip files on your system.
2. Now run the program *setup* that was part of the distribution. The easiest way to do this is to double click on the program's icon in the Program Manager or the Windows Explorer.
3. The setup-program will ask you where you want the program installed, and, if you are using the larger distribution, which parts of the distribution you want to install. When you are done with making your choices the program will be installed on your system. A group named Predictor will also be added to the start menu.
4. Run Predictor by clicking on the "PredictorProgram" alternative in the group on the start menu.
5. Since this is the first time you use the program you will be asked for your serial number. Make sure that you enter the serial number exactly as it is specified, otherwise the program will not work. If you do not have a serial number you can request one from the program's homepage. If for some reason this is not possible you can contact the Predictor Development team at the address given in Section 1.2. We would be grateful if you did not use somebody else's serial number since the serial numbers allow us to keep track of approximately how many people are using the program.

That is it! Predictor is now installed on your system.

If you ever want to remove the program you should do this by means of the "Add/Remove Program"-program that is available from the Control Panel to ensure that all parts of the program are removed. If you have made any changes to the files, e.g. created any new databases, these will not be automatically removed. To remove them simply delete the directory where Predictor is installed.

2.3. Registering Predictor

If you find Predictor useful and intend to continue using it you are obliged to register the program. Registration is free, but we ask you to give us some data on yourself and your usage of Predictor. This data will be used by us in our research and to improve Predictor, but *under no circumstances will any traceable information be given to any third party*.

When it is time for you to register the program the Registration Dialog will come up when you start the program. If you do not want to register at that particular time it is ok, but the dialog will continue to occur every time you start the program until you have registered. This registration procedure will happen twice with some time between them. This allows us to quickly get information, but also to get information from people who have used the program for some time.

When you choose to register the program, the Registration Dialog will provide a wizard interface where you will be asked several questions about your usage of Predictor. Examples of these questions are what kind of projects you are developing and how much data you have in your database. *Feel free to skip any questions that you do not wish to answer*, we have no use for fabricated data and even the single fact that you are using the program is of value to us.

2.4. Using Predictor with a Common Database

Predictor is primarily intended to be used as a *single user* measurement database and prediction support tool. However, the program works quite well when several people are using the same (physical) database. The only time there might be problems is when several people are trying to edit the same project. In this case one of them will get an error as soon as they start typing.

Worth to note though is that there is no network version of Predictor and that the database thus must be located on a common file server for this to work.

3 Cost Estimation Techniques Supported

Predictor can be used to predict almost anything, but to keep things simple the examples in this manual deal with estimating the effort (i.e. man-months) needed to complete a software project. We choose this example since effort estimation is an extremely important part of a project manager's job. Furthermore it is an area where many people experience problems as is evident from a long list of failures to keep software projects within schedule and budget.

While it by no means is enough, we believe that the usage of a tool such as Predictor will help an estimator to improve his or her skill. It can be used as a knowledge repository where data from projects are stored. It can also record estimations done at various stages of a project's lifetime so that it can be compared to what actually happens. Furthermore, Predictor can make estimations on its own which can be compared to the user's own views, or used to develop estimation rules of thumbs.

Before we go into how to use Predictor we would like to give a quick overview of the three types of estimation techniques that Predictor supports: Expert Judgement, Local Algorithmic Models, and Case-based Reasoning.

3.1. Expert Judgement

The normal situation in the industry today is that an estimation of the effort needed to complete a software project is made manually. It can be done either by a single person or by a team as in the Delphi method, but it is nevertheless done manually by humans.

These techniques which are commonly referred to as *expert judgement* actually work very well if the estimator is skilled. Unfortunately, the aforementioned list of failures show that such skill is not as common as one could wish, and the situation does not seem to improve.

But what then is needed for improvement? There is no definite answer to this question, but experience on doing estimates, and feedback on the results of estimates are definitely needed. Of these Predictor primarily supports the second since it allow you to store all your estimates so that you can compare them to the actual outcome of a project and to get alternative estimates directly from the program. It also helps with the comparison by means of various reports, and the ability to check the accuracy of the estimations done.

3.2. Algorithmic Models

An algorithmic cost model, or a parametric cost model as it is sometimes called, is a mathematical model which is used for cost estimation. In its simplest form it is a variant of the equation:

$$\text{Effort}=\text{base effort}+\text{effort per unit}\cdot\text{size}^{\text{effects of scale}}$$

During the years, several dozen algorithmic models have been introduced, many with tool support. The most wellknown of these models is COCOMO (now COCOMO II) developed by Dr Barry Boehm [Boehm, 1981, Boehm, 1998]. Unfortunately the predictive capability of these models when applied to other organisations than the one where they were developed has usually been bad, and off-the-shelf models thus need to be calibrated to the local environment. To avoid this problem Predictor supports the development of local algorithmic models, i.e. models developed from your data that do not need to be calibrated to your environment.

The algorithmic models that Predictor supports are on the form:

$$\text{effort}=c+\alpha_1\cdot A_1^{\beta 1}+\ldots+\alpha_n\cdot A_n^{\beta n}$$

Predictor's algorithmic models are (usually) developed by regression, and therefore to create a useful model you need access to relatively large amounts of data. If your project database is small, it is probably a better idea to use the Case-Based Reasoning estimation technique described below instead.

3.3. Case-Based Reasoning

Case-Based Reasoning (CBR) is a technique that we have borrowed from the Artificial Intelligence area that is based on the assumption that similar problems have similar solutions. In this setting this means that to make a prediction, Predictor searches the project database for projects that are similar to the one we are interested in and use their values to make a prediction.

This raises several questions such as:

- How do you measure similarity between projects?
- What is the relative importance of attributes? That the size of the project is more important than the colour of the buttons is pretty obvious, but how about size versus complexity?
- If a project lacks certain attributes, how is this handled? Do you remove all project that lacks any attribute, remove the attribute, assume that there is no useful information in the attribute, or use another alternative?
- When a set of projects has been considered similar enough, how do you make the actual prediction? Do you use the mean, the mode, or perhaps an algorithmic model to make the prediction? And how do you handle attributes that are not numbers?

Fortunately, as a user of Predictor, you do not have to worry too much about such things since Predictor will take care of most of them automatically. The only technical aspects you have to get acquainted with are the control parameters available in the Database Attributes Window and the concept of attribute weights. Both of these topics are covered in the guided tour of Predictor in Section 5.

To give you a general idea though, Predictor uses the Weighted Euclidean Distance (WED) to measure similarity between projects:

$$\text{WED}=\sqrt{\sum_{i=1}^{n}\left(D\left(p_i,q_i\right)\cdot W_i\right)^2}$$

This is a generalisation of Pythagoras' theorem to n dimensions. Basically you see each attribute as a separate dimension in a search space. Then, when a prediction is to be made for a project *p* the WED between *p* and every other project in the database is calculated. The projects with the lowest WED is then considered the most similar to *p*.

All projects that have a WED below a certain threshold are then used to make the prediction. How the prediction is made depends on what kind of attribute it is we want to predict as well as the settings of a number of control parameters discussed in Section 5.4.

If you want more details on how Predictor makes its CBR predictions, the components used for the task are described in [Bergström, 1999].

3.4. Estimation vs Prediction

The terms estimation and prediction has slightly different meanings, especially in the scientific community. Within Predictor, we don't make use of such distinctions, however, and the terms are used interchangeably.

4 The Main Window

The main window (Figure 1) is the heart of Predictor. It is here you as a user will spend most of the time, and to get the most from the program, you need to understand the functionality of this window.

The largest part of the window is taken up by the project database. In Figure 1 two projects are visible in the database, K1 and K2. The projects are described by nine attributes: ID (project identifier), Lang (programming language), Machine (the type of machine the product should run on), Months (how many calendar months the project took), MM (man-months), KSLOC (thousand lines of code), FP (function points), UA FP (unadjusted function points), and LOC per MM (lines of code per man-month).

Above the project database the current project is shown. The current project, K1 in this case, is the project we are interested in, and want to make a prediction for. To select a new current project, double click on any of the projects in the project database.

To the right of the project database and current project there are two areas: Prediction, and Earlier Predictions. The first of these, Prediction, is used to show information about the predictions made directly by Predictor, i.e. by an algorithmic model or by Case-Based Reasoning. The other, Earlier Predictions, is used to show information about recorded predictions. The information displayed in both these areas are partly controlled by the current project but in different ways. If you change the current project, the information in the Prediction part will stay the same, but the next prediction made by the system will be for this project. The Earlier Predictions part on the other hand will change directly to show any recorded predictions for the new current project.

Figure 1 Predictors Main Window

Tip!

If the name of an attribute is too long and partly hidden by the editbox (as LOC per MM in Figure 1) you can switch on the option ”Show hints”. If you do this and positions the mouse pointer above an attribute, you will see its full name.

4.1. The Toolbar

The toolbar contains 19 icons organised in eight groups:

The File Group

The New Database Button creates a new database.

The Open Database Button opens an existing database.

The Save Database As Button saves the database under another name.

The Print Button displays the Print Dialog from where you can print various reports.

The Database Group

The New Project Button inserts a new project into the database.

The Edit Project Button controls if the database is editable or not. If the button is down, you can edit the database, if it is up, you can not.

The Delete Project Button deletes a project from the database.

The Database Attributes Group

The Set Attributes Button changes the predicted attribute and other general properties of the database.

The Search Group

The Search Button allows you to search for projects with specific attributes values.

The Show Selected Only Button controls which projects that are visible in the project database. If it is down, then only projects that are not hidden are shown, if it is up, all projects are shown. You can hide projects by searching or by rightclicking on a project and selecting hide.

The Estimations Group

The Manual Estimation Button records information about a prediction for the current project.

The Manual Estimation Accuracy Button shows information about the quality of the recorded estimates. Despite the buttons name these estimations can be done directly by Predictor.

The CBR Group

The CBR Estimation Button makes a CBR estimation of the currently selected project. The results are written to the CBR Prediction tab in the Main Window.

The CBR Accuracy Button estimates the accuracy of the CBR approach on the current database. The information is written to the CBR Prediction tab in the Main Window.

The Set Weights Button opens the Set Attribute Weights Window where the weights of the attributes can be set.

The Algorithmic Model Group

The Algorithmic Estimation Button makes a prediction based on an algorithmic model. The results are written to the Algorithmic tab of the Main Window.

The Algorithmic Accuracy Button estimates the accuracy of the currently chosen algorithmic model. The information is written to the Algorithmic tab in the Main Window.

The Develop Model Button creates a new algorithmic model or calibrates an old.

The Help Group

The Help Button shows the help index.

4.2. The Menubar

The main menu contains all the functionality from the toolbar (Section 4.1). In addition it contains the following functions:

- File: Import Data
  Opens the Import Dialog where data in various formats can be imported into a Predictor database. The import functionality can be extended by means of import filters.

- File: Export Data
  Opens the Export dialog from which data can be exported to an ascii file.

- File: Close
  Closes the program.

- Search: Clear Search
  Clears the current search results and make all projects visible again.

- CBR: Batch Test
  Perform a series of tests to determine the best configuration for the control parameters that control how CBR predictions are made.

- Algorithmic: Choose Model
  Selects which algorithmic model to use through a standard file selector. Below this item the last four models are listed for direct access.

- Help: Index
  Opens the helpfile at the contents page. Corresponds to pressing the F1 key

- Help: About
  Shows the About Box.

- Help: Usage
  Shows information about your usage of Predictor.

- Help: Options
  - Add Predictions - Check one of these options to select when predictions made by CBR or algorithmic models should be added to the list of earlier predictions. By default predictions are not saved.
  - Show: Warnings - Check this to show warning messages. The default is to show warnings.
  - Show: Hints - Check this to show hints in the program. The default is to show hints.
  - Show: Project No. - If this is checked, the project's internal number in the database is displayed in the project database. This can be used to identify projects that do not have a real name. The default is to show the project number.
  - Show: Init Window - Check this to show the startup window when Predictor is started. The default is to show the window.
  - Visible Projects - Selects how many projects that are visible at the same time in the project database. The default is to show 2 projects.

5 A Guided Tour

Now that you have made yourself familiar with the main window, it is time to actually start using the program. To quickly get you started we have prepared a guided tour of the program for you. The tour takes you through all the basic steps of using Predictor, from setting up a project database to making a prediction.

The topics covered by the tour are:

- Creating a project database
- Recording project data
- Using Predictor with very little data
- Making a CBR prediction
- The CBR Control Parameters
- Importing data
- Jackknifing and Batch Runs
- Creating an algorithmic model
- Checking a rule of thumb
- Creating a new database type
- Getting feedback on the quality of ones estimates

Tip!

An electronic version of this tour is included in the helpfiles.

5.1. Creating a Project Database

The New Database Button

When you open Predictor, you are presented either with a ”blank” program (Figure 2), or the last database that was open. If all you have done so far is to have a look at the example database shipped with Predictor, then this is not particularly interesting. Therefore, lets start with setting up a new project database. To do this click on the New Database button, or select New from the File menu.

Figure 2 The empty Main Window

This takes you to the New Database Window (Figure 3). Here you can select what type of database you want to create. If none of the available database types suits you needs you can create new types by clicking on the New Type button, but for the moment lets assume you want a database of the type *Example*. Select it and click on the OK button. A file requester will come up and ask you to save your new database. It is a good idea to save it in an empty directory since it contains several files which might be hard to locate if saved together with other files.

Figure 3 The New Database Window

After you have saved the database the New Database Window will close, and you will return to the main window and presented with an empty database (Figure 4). As you can see, it is not the most interesting project database one could think of, but it will be sufficient for our needs during the tour.

Figure 4 Predictor with an empty example database

5.2. Recording Project Data

The New Project Button

The Edit Project Button

At this stage Predictor is only useful as a measurement database, so lets start with recording some data into the database. To do this click on the New Project Button. This will place the database in edit mode and insert a blank project in the database. The cursor will be placed on the first field of the new project. This field represents the name of the project. Enter ”Project 1” (without the quotation marks) and press tab once to get to the next field. This is the type of the project. In this example we will have two types of projects, named T1 and T2 for simplicity. Project 1 is a T1 project, so enter ”T1” in this field. The next field is the size of the project in thousand lines of code (kloc). Enter 130 here. The last field is the attribute we want to predict, the development time of the project in man-months (MM). Project 1 took 100 MM. When done press the Add Project Button again to add the next project to the database.

Now enter the data in the table below into Predictor:

Name	Type	Size (KLOC)	Dev. Time (MM)
Project 2	T2	50	20
Project 3	T1	120	95
Project 4	T2	75	

When you are finished entering data click on the Edit Project Button to set the database back into the normal browsing mode. Now the database will not allow you to edit the data stored in it, so if you made an error and want to change it, you have to press the Edit Database Button again to get back into the edit-mode. The main window will now look something like Figure 5.

Figure 5 Predictor after adding a few projects

Tip!

To move backward among the fields, use shift+tab. You can also click in a field to get there quickly.

5.3. Using Predictor with Very Little Data

The Manual Estimation Button

Now we have a project database. It does not contain much data yet, but at least it is something. All functionality of Predictor is now available to us although much of it is redundant at this stage. Two things that we can use now, however, are the ability to record predictions and to make CBR predictions. Lets start with recording a prediction.

To do this we need a new project to make the prediction for. To simplify things we will use one of the projects we recorded information on in the previous step. Make sure that the database is not in edit-mode, and double click on Project 4 in the Project Database part of the Main Window.

Project 4 will now be the Current Project and stay visible in the upper left part of the main window. This ensures that you always have access to all information on the project you are interested in.

Now, try to estimate how many man-months the project would need. The project database is available to help you. Look through the other projects and try to use the information available there (even though it is fabricated).

When you have estimated the time needed, click on the Manual Estimation button. This will open the Manual Prediction /Adjustment Window (Figure 6) where you can record your estimation. The name of the attribute, ”Dev. Time”, and the date should already be the correct ones. If not, you can correct them manually. Enter your prediction and your name in the appropriate fields. You can also enter a note on why you made the prediction, and how you reached your conclusions in the textarea to the right.

Figure 6 Manual Prediction/Adjustment Window

Tip!

There are a few things that might help you to get a clearer picture of the data:

If you have a large monitor you can change the number of visible projects under the menu “Help:Options:Visible Projects”. You can also change the size of the various parts of the main window by drag'n drop.

If some projects are clearly useless, you can hide them by right clicking on them and selecting "Hide" from the popup-menu that occurs. To make all projects visible again, either click on the Show Selected Only button or right click in the project database area and select "Show selected only".

When you are done, click on the OK button to close the window. You will now see your prediction in the lower right corner of the main window. Double clicking on it will open the Manual Prediction /Adjustment Window again and allow you to change it.

Close the Manual Prediction /Adjustment Window if it is open and double click on one of the other projects in the database. You will notice that the prediction we recorded disappears from the list. Do not worry though, it is simply because the recorded predictions are associated with the currently selected project. If you double-click on Project 4 again the prediction we recorded will reoccur in the list.

5.4. Making a CBR Prediction

The CBR Estimation Button

Our next step is to try out Predictor's CBR prediction capability. To make a CBR prediction, click on the CBR Estimation button. If you have followed this tutorial to the letter you will be presented with the No Weight File Dialog. This is because we have not created a weight file for the project database yet. This is a file that contains information on the relative importance of the attributes in the database, and how it should normalise the distance calculation that is a part of a CBR prediction. If this seems complicated, do not worry, Predictor will take care of most of it for you.

Select the "Create one manually" option and click on OK. Predictor will now create a simple specification file, and open the Set Attribute Weights Window (Figure 7). This window presents the information in the specification file to you in four columns: the name of the attribute, the weight, minimum value, and maximum value. The last two are only available for numerical attributes, that is why they are not shown for the Name and Type attributes.

Figure 7 Set Attribute Weights Window

Lets start by considering the weights. The weight of an attribute is a number that specifies its importance relative the other attributes when Predictor searches for similar projects. In a real project database, the setting of weights can be a very hard problem, but in this small example it is relative straightforward. Obviously the Name attribute is of no importance, so set its weight to 0. The Type and Size attributes on the other hand are both important, but we do not know how yet, so leave these at 1.

The last weight is for the Development Time attribute. Since we normally only do CBR predictions for projects that does not have this attribute we might believe that its weight does not matter, but it does. It must be set to 0, otherwise tests of CBR estimation accuracy will give too good values since they do a prediction for every project in the database that has this attribute.

Tip!

The actual values of the weights are not important. It is their relative values that matter.

There is more information about setting weights in the online help.

Now lets consider the minimum and maximum values. These are used to normalise the influence of attributes, and are by default set to the minimum an maximum values found in the database. The program will work with these default settings, but we recommend that you set them manually anyway.

Basically the values should correspond to the minimum and maximum values that the attribute ever can take. In this case we know that both the minimum values must be zero. Because of the scale used in the database the size of the projects seems to be approximately the same (within an order of magnitude at least) as the development time, so we can simplify things by setting the maximum values of both of these to the same value. Lets say that we don't believe that we will undertake any project that take more than 500 man months and set both maximum values to 500. If we ever do get involved in a larger project then the maximum values has to be updated of course.

When done editing the weights click on "Ok" to close the window. Since we now have a weight file Predictor will continue with the CBR prediction. To see the prediction, click on the CBR Prediction tab (Figure 8) in the Prediction part of the Main Window. The upper half of the CBR Prediction tab shows the latest CBR prediction. Does the value predicted, 30 man-months, correspond to the estimation you just did manually?

Figure 8 The CBR Prediction Tab

5.5. The CBR Control Parameters

The Set Attributes Button

Whether or not Predictor's value corresponds to your own you may want to examine how Predictor came up with it is prediction. First of all we have to check the settings of the prediction parameters. It is these together with the weight settings discussed in Section 5.4 that control how the prediction is made. The three most important parameters are shown directly on the CBR Prediction tab (Figure 8): Method, Threshold, and Size Adjusted.

The Method controls how the values of the analogies are used to make a prediction. There are three possible values of this attribute:

- Arithmetical mean - The predicted value is the normal arithmetical mean of the analogies values.
- Geometrical mean - The predicted value is the geometrical mean of the analogies values. See the online help for a definition of the geometrical mean.
- Voting - The value that most of the analogies has is used. This is normally only useful for boolean values.

The Threshold controls how similar a project must be to the one we are interested in to be used as an analogy. By default the Threshold is 50%, but it might be a good idea to change it to a higher value. During our own experiments has values between 70 and 80% usually given the best results.

Size Adjusted finally tells Predictor if you want the values of the analogies to be linearly adjusted to take into account difference in size between the target project and the analogy. To take an example, assume that we wish to predict the effort needed to complete a project P1 whose size we have estimated to 50 kloc and the only analogy we have is a project P2 who's size was 25 kloc. If Size Adjusted is set to true, the prediction will be that the effort needed for P1 is approximately twice the effort needed for P2. If, on the other hand, Size Adjusted is false, then the prediction will be that the effort is the same as for P2.

If you click on the Analogies tab you can see which projects were used as analogies. In this case it is only Project 2. The view gives you the name of the analogies, the calculated similarity between them and the project we are interested in, the size, and the actual value.

Figure 9 The Analogies Tab

To set the control parameters, click on the Set Attributes button to open the Set Prediction Attributes window (Figure 10). The first setting selects the attribute to be used as the name of the project. If it is not set, then the projects internal number will be used instead. The second setting selects the attribute we want to predict. If this is not set, then we will not be able to make a CBR prediction. The last three settings correspond to the control parameters discussed earlier.

When you are done setting the control parameters, click on the OK button to return to the Main Window where you can try out the new settings by clicking on the CBR Estimation button. Did the new settings change the prediction? Go ahead and try out different settings. Since the database is so

small, you can examine the results easily. Before you continue with the tour though, make sure that the settings are the same as in Figure 10.

Figure 10 The Set Prediction Attributes Window

5.6. Importing data

Before we can continue, we need more data in the database. It is tedious, however, to have to enter the data manually, and we have therefore prepared some example data for you. To import this data into Predictor, click on the File menu and choose Import Data. This will open the Import Dialog (Figure 11) from where you can import various types of data.

Figure 11 The Import Dialog

Ascii (text) files are the only format supported directly by Predictor. This is not such a big disadvantage as one might think since it is possible to import data from other formats by using import filters described in Section 7.

In this case the data is stored in an ascii file so select that option from the drop down list at the top of the screen and click on the Browse button. The data file is called “tutorialdata.txt” and is located in the tutorial directory.

Make sure that the Headings and Tab checkboxes are checked and look at the two lists of attribute names in the middle of the window. As you can see they contain the same names. Select the attribute “Type” in both lists and click on the Join button. This tells Predictor that you want the data stored in the column “Type” in the import file to be stored in the attribute with the same name in Predictor. If you make an error, you can correct it by right-clicking in the middle list and select Delete.

Manually selecting and joining attributes might be okay if it is just four attributes as in this example, but with databases with many attributes it is clearly inadequate. To speed up thing a little bit you can click on the All Equal button. This will perform a join of all attributes that have the same name in the file as in the database.

When you are done joining the attributes, click on the Ok button to import the data into Predictor and close the window.

5.7. Jackknifing and Batch Tests

The CBR Accuracy Button

Predictor has the capability to test the accuracy of the predictions made by the system. To perform an accuracy test click on the CBR Accuracy button.

The technique that Predictor uses to test the accuracy is called jackknifing. It is a simple technique where a prediction is made for each project in the database. The prediction is then checked against the real value and the difference is used to calculate various error metrics.

If you now look at the lower part of the CBR Prediction tab (Figure 8, page 7) you will see the results of the test. At the top, the values of the control parameters are listed. Below them comes the results of the accuracy test. The first two values tell how many predictions were actually made, and how many projects in the database the system tried to make predictions for. In this case 19 predictions were made out of a total of 20. The differences between the number of made predictions, and the total possible usually depends on that no suitable analogies were found for some projects or that they lacked the attribute that should be predicted. In this case it is Project 4 that Predictor couldn’t use since it doesn’t have the Dev Time attribute.

The last nine values are various error metrics. A complete description of all of these are beyond the scope of this tour and we will concentrate on the MRE and Pred(X) values. For information on the rest of the metrics see Appendix A.

The MRE, or Mean Relative Error, is defined as:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{|A_i - E_i|}{A_i}$$

Basically it is the mean of the absolute error adjusted for size. The idea is that a weeks work for one person means very little if the projects duration is several man years, but very much if it’s just one man-month.

The Pred(x) metric is defined as the percentage of the projects where the predicted value fall within x% of the actual value. Predictor gives four values for x: 5, 10, 25, and 50.

Neither of these metrics are perfect, the MRE is sensible to outliers, i.e. a few projects where the error is large, and Pred(x) can easily be “fooled” when most of the predictions falls just outside the limits of x, but together they usually give a good picture of the accuracy.

It is impossible to give general guidelines for what a good value of MRE or Pred(x) is since it depends too much on the domain. In some domains, an error of 50% might be considered a success, in others

an error of 5% might be totally unacceptable. The relative importance of the metrics also changes, e.g. if we can accept a few large errors as long as most predictions are good, then the importance of Pred(x) goes up, and MRE down. In the domain of software project cost estimation an MRE of less than 25% and Pred(25) of at least 70% is usually considered acceptable for a first estimation of projects.

If you're not sure about how to set the control parameters it might be a good idea to perform a Batch Test of all combinations. Be warned though that since this includes making a jackknifing test for all possible combinations of values (under the constraints that you specify) this might take a long time. To perform a Batch Test select the menu option CBR: Batch Test.

Tip!

You can copy the results of a prediction or an accuracy test to the clipboard by right clicking on it and select the alternative Copy.

5.8. Creating an Algorithmic Model

The Develop Model Button

The Algorithmic Estimation Button

The Algorithmic Accuracy Button

Now we have enough data to create an algorithmic model. Make sure that the Show Selected Only button is up, and click on the Develop Model button to bring up the Develop Algorithmic Model Window (Figure 12).

Figure 12 Develop Algorithmic Model Window

We want to create a simple model that tries to predict the development time from the size of the project. To do this select Dev Time from the drop down list under Predicted Attribute and double click on Size in the list of available attributes. To actually create the model, click on the Develop button. Predictor will now perform linear regression on the data in the database and create the model that best

fits the data. If you have followed the tutorial, the model will be the one presented in Figure 12, or in a more readable form:

$$DevTime = -7.721 + 0.83 \cdot size$$

If you wish you can test the model by clicking on the Test button. This performs a limited version of jackknifing which only tells you the MRE and Pred(25) metrics.

When you are satisfied with the model, save it and click on the Close button to return to the main window. Your newly created model is now ready to be used by clicking on the Algorithmic Estimation button. You can also perform a complete accuracy test similar to the one available for CBR predictions by clicking on the Algorithmic Accuracy button. The results of these actions are showed on the Algorithmic tab (Figure 13).

Figure 13 The Algorithmic Tab

5.9. Checking a Rule of Thumb

The Search Button

The Show Selected Only Button

Apart from creating new algorithmic models, the Develop Algorithmic Model window can also be used to test e.g. whether a rule of thumb is useful. Suppose that we believe that it takes approximately 0.9 man-months to develop a thousand lines of code in a system of type T1. To test if this is a reasonable model we need to do two things: first we have to create a view of the database that shows the relevant projects, then we create and test the algorithmic model mm=1.1·kloc. Remember 1/0.9≈1.1.

The easiest way to create a view of the database that shows only the T1 systems is to search for them. To do this click on the search button on the toolbar. It will open the Search Dialog (Figure 14). Select the attribute Type from the first checkbox and write T1 in the corresponding editbox. Then click Ok. Now if you look in the project database, only T1 projects are visible.

When you have created the view, open the Develop Algorithmic Model Window and create the model in shown in Figure 15. When your done, save the model, close the window and click on the Algorithmic Accuracy Button to test the accuracy of this rule of thumb. You will probably find that it gives quite good results.

Figure 14 The Search Dialog

Figure 15 Developing a rule of thumb

Now that we have seen that our rule of thumb corresponds quite well with the actual development time for T1 projects we might want to know if it is useful for other projects too. Since we only have two types of projects this is simple. Click once on the Show Selected Only button to show all projects in the database. Then click on the Algorithmic Accuracy button once more. The new test will be performed with all projects instead of just the ones that were visible. As you can see the results are less good this time. Apparently our rule of thumb does not correspond well with reality in this case.

5.10. Getting Feedback on the Quality of Ones Estimates

The Manual Estimation Accuracy Button

One of the more important features of Predictor is that you can use it to get feedback on the quality of your estimates. To take advantage of this feature, you have to save your estimations as was shown in Section 5.3. Do this for two or three projects, and make sure that you enter your name when you

record the estimations. When your done, click on the Manual Estimation Accuracy Button to open the Prediction Accuracy window (Figure 16).

To perform a test of your estimation accuracy, select your name from the drop-down list in the upper left part of the window, the attribute you have estimated from the list in the upper right part, and click on the Test button. Predictor will now perform a test similar to jackknifing for the estimations you have recorded and display the usual error metrics in the lower left part of the window. The lower right part is used for a textual description of the results. This however, is experimental, and only applicable for very limited domains yet.

Figure 16 The Prediction Accuracy Window

5.11. Creating a new Database Type

The example database type used in this tutorial is not particularly interesting. Therefore our last task will be to create a new database type.

To do this open the New Database Window (either by clicking on the New Button or by selecting "File:New" from the menu) and click on the New Type button. This will open the Database Design Window (Figure 17) where new databases are created.

Start by giving the new database type a name and an image. The name is written in the textbox just below the design area, and the image is set by clicking on the image in the lower left corner of the window. It is also a good idea to describe the database type so that you remember what it was supposed to be used for. Descriptions are written in the textarea that is showed if you click on the Description button.

Figure 17 Database Design Window

When you have set the name and the image its time to add the attributes to the model. To do this, click on the Add Attribute button. This will open the New Database Attribute Dialog (Figure 18). Which attributes you should include in the model is up to you, try to design a database that you would find useful. These things are definitely needed though:

- A project identifier: This is probably a string, i.e. the type should be discrete, and since it is the name of the project, it should have the name property set.
- The attribute you want to predict: This is normally a numerical or boolean value, and should have the predicted property set. In Software Engineering, this is usually the development time of a module or the number of faults expected to be found.
- Attributes that influence the attribute you want to predict. This is the hard part, if you can find a usable set of attributes here, then you have come a long way on your journey towards better estimations.

Figure 18 New Database Field Dialog

6 Printing Reports

Since the paperless office is still a utopia a tool such as Predictor must be able to produce reports. This section describes which reports that are available and how to produce them. The Print Dialog (Figure 19) is used to produce most of the reports available in Predictor. You can also use the dialog to set up which printer to use by clicking on the Setup button.

Figure 19 The Print Dialog

6.1. Database Contents Report

Description: The Database Contents Report is used to print the content of the database. The layout on the report is the same as in the project database. The number of projects printed depends on how the report is invoked.

How to print the report: The Database Contents Report can be printed from the Print Dialog or by right clicking in the project database.

Figure 20 Database Contents Report

6.2. Prediction Accuracy Report

Description: The Prediction Accuracy Report is used to print the results of an accuracy test done in the Prediction Accuracy Window.

How to print the report: In the Prediction Accuracy Window, click on the Print button.

Predictor Report
Prediction Accuracy
C:\Program Files\PPL\Predictor\tutorial\tour
2000-02-16

User: Henrik Bergström
Attribute: Dev Time
Made: 2
Total: 3
ME: 12.5
MAE: 12.5
MRE: 0.13
WMRE: 0.064
TS: 2
Pred(5): 0.5
Pred(10): 0.5
Pred(25): 1
Pred(50): 1
Textual Description: The average estimation acuracy is good.
All predictions fall within 25% of the actual value.
The predictions is biased.
Howevere these opinions is based on a VERY limited group of predictions, and should be interpreted in that light.

Figure 21 Prediction Accuracy Report

6.3. Earlier Predictions Report

Description: The Earlier Predictions Report is used to print recorded predictions. The number of recorded predictions that is printed depends on how the report is invoked.

Predictor Report
Earlier Predictions
C:\Program Files\PPL\Predictor\tutorial\tour
2000-02-16

Project: 1
Attribute: Dev Time
Date: 2000-02-16
Made By: Henrik Bergström
Prediction: 95
Motivation:

Original Prediction
Date: 2000-02-16
Made By:
Prediction:

Project: 3
Attribute: Dev Time
Date: 2000-02-16
Made By: Henrik Bergström
Prediction: 75
Motivation:

Original Prediction
Date: 2000-02-16
Made By:
Prediction:

Figure 22 Earlier Predictions Report

How to print the report: The Earlier Predictions Report can be printed from the Print Dialog in which case you can choose to print all recorded predictions or only the ones for the current project. It can also be printed by right clicking in the earlier predictions part of the main window (the lower right corner) and selecting the print option, or by clicking on the Print button in the Manual Prediction/Adjustment Window

6.4. CBR Prediction Report

Description: The CBR Prediction Report shows a single prediction made by CBR. All the information from the CBR Prediction tab in the main window is displayed as well as up to 25 of the projects that were used as analogies.

How to print the report: The CBR Prediction Report is printed from the Printer Dialog.

Predictor Report
CBR Prediction
C:\Program Files\PPL\Predictor\tutorial\tour
2000-02-16

Project: Project 4
Attribute: Dev Time
Method: Geometrical mean
Threshold: 50
Size Adj: Yes (Size)
Prediction: 41.85
Analogies:

Project	Similarity	Size
Project 4	100%	75
Project 10	100%	75
Project 19	99%	68
Project 18	98%	55
Project 9	97%	100
Project 2	97%	50
Project 12	97%	45
Project 17	96%	110
Project 11	96%	120
Project 6	50%	75

Figure 23 CBR Prediction Report

6.5. CBR Jackknifing Results Report

Description: The CBR Jackknifing Results Report shows the results of a test of estimation accuracy for CBR. All the information displayed on the CBR Prediction tab in the main window is showed.

How to print the report: The Report is printed from the Printer Dialog.

Predictor Report
CBR Jackknifing Results
C:\Program Files\PPL\Predictor\tutorial\tour
2000-02-16

Attribute: Dev Time
Method: Geometrical mean
Threshold: 50
Size Adj: Yes (Size)
Made: 19
Total: 20
ME: -0.55
MAE: 12.29
MRE: 0.19
WMRE: 0.0065
TS: -0.85
Pred(5): 0.26
Pred(10): 0.42
Pred(25): 0.74
Pred(50): 0.95

Figure 24 CBR Jackknifing Results Report

6.6. CBR Batch Test Results Report

Description: The CBR Batch Test Results Report is used to get information on a Batch Test, i.e. a series of tests used to find the best possible settings for the CBR control parameters. This report can easily be 50 pages or more long if you which to show as much detail as possible. First in the report comes the actual results of the tests in tabular form (Figure 25). This part is almost unreadable because of the wealth of information in it, and normally this information is used in some other program such as Excel or Minitab to analyse it more carefully. After this comes all the results in graphical form (Figure 26).

How to print the report: When a batch test is done the results can be printed directly from the dialog that occurs then. The results can also be saved, in the tabular form in Figure 25, so that it can be included into another program or printed later in which case it is done from the Print Dialog.

2000-02-16
Batch results for: tour

Attribu	Metho	Thresl	Size A	Made	Total	ME	MAE	MRE	WMRE	TS	Pred0	Pred1	Pred2	Pred5
Dev Time	Arithmeti	90	Yes	10	11	6.33	13.306	0.127	0.0119	4.757	0.2	0.3	1	1
Dev Time	Arithmeti	90	No	10	11	3.963	22.09	0.184	0.0197	1.794	0.4	0.4	0.7	1
Dev Time	Geometric	90	Yes	10	11	6.995	13.541	0.129	0.0121	5.166	0	0.3	1	1
Dev Time	Geometric	90	No	10	11	6.522	21.87	0.179	0.0195	2.982	0.3	0.4	0.7	1
Dev Time	Voting	90	-	10	11	7	30	0.244	0.0268	2.333	0.3	0.6	0.6	0.9
Dev Time	Arithmeti	95	Yes	10	11	5.757	13.037	0.123	0.0116	4.416	0.3	0.3	1	1
Dev Time	Arithmeti	95	No	10	11	-2.476	5.905	0.0712	0.00527	-4.194	0.8	0.8	0.9	1
Dev Time	Geometric	95	Yes	10	11	6.309	13.024	0.123	0.0116	4.844	0.3	0.3	1	1
Dev Time	Geometric	95	No	10	11	-1.753	6.065	0.0725	0.00542	-2.89	0.8	0.8	0.9	1
Dev Time	Voting	95	-	10	11	-0.5	6.5	0.0753	0.0058	-0.769	0.6	0.9	0.9	1

Figure 25 CBR Batch Test Results Report - Data

Figure 26 CBR Batch Test Results Report - Graphs

6.7. Weight File Report

Description: The Weight File Report prints the content of a weight file.

How to print the report: In the Set Attribute Weights window, click on the Print button.

Predictor Report

Weight File

C:\Program Files\PPL\Predictor\tutorial\tour

2000-02-16

Attribute	Weight	Min	Max
Name	0		
Type	1		
Size	1	0	500
Dev Time	0	0	500

Figure 27 Weight File Report

6.8. Algorithmic Model Report

Description: The Algorithmic Model Report shows an algorithmic model.

How to print the report: In the Develop Algorithmic Model window, click on the Print button.

Predictor Report
Algorithmic Model
C:\WINNT\bat71A.tmp
2000-02-16

dev time= -7.724

+0.83 * size [1]

Figure 28 Algorithmic Model Report

6.9. Algorithmic Model Prediction Report

Description: The Algorithmic Model Report shows a single prediction made by an Algorithmic Model. All the information from the Algorithmic tab in the main window is displayed.

How to print the report: The Algorithmic Model Prediction Report is printed from the Printer Dialog.

Predictor Report
Algorithmic Model Prediction
C:\Program Files\PPL\Predictor\tutorial\tour
2000-02-16

Project: Project 1
Attribute: dev time
Prediction: 100.18

Figure 29 Algorithmic Model Prediction Report

6.10. Algorithmic Model Jackknifing Results Report

Description: The Algorithmic Model Jackknifing Results Report shows the results of a test of estimation accuracy for an algorithmic model. All the information displayed on the Algorithmic tab in the main window is showed.

How to print the report: The Algorithmic Model Jacknifing Results Report is printed from the Printer Dialog.

Predictor Report
Algorithmic Model Jackknifing Results
C:\Program Files\PPL\Predictor\tutorial\tour
2000-02-16

Attribute: dev time
Made: 11
Total: 11
ME: -43.1
MAE: 43.1
MRE: 0.29
WMRE: 0.028
TS: -11
Pred(5): 0.091
Pred(10): 0.18
Pred(25): 0.27
Pred(50): 1

Figure 30 Algorithmic Model Jackknifing Results Report

7 Import Filters

To manually enter the measurement data might be possible if the database is used to record data only on whole projects, but if you want to keep track of e.g. the fault rates in every module in a large system this is an almost impossible task. To help ease the problem, Predictor supports user defined import filters that can be used to import data directly from many applications.

Writing an import filter is a relatively straightforward task of writing a Dynamic Link Library (DLL) that conforms to the interface given below. To install an import filter simply put the DLL in the directory *filter* that is located in the directory that Predictor is installed in.

7.1. Delphi Interface of an Import Filter

This section provides the interface of an Import Filter. The language used is Delphi, which in this case is almost the same as standard Pascal. A C/C++ implementation of the interface is given in Section 7.2.

The choice of language in this section was made because of the way that strings are passed from the filter to Predictor, which is made more explicit in Pascal than in C/C++. Basically the strings used are null terminated character arrays, with a fixed length of 256 characters, i.e. strings passed to and from the filter can be at most 255 characters long. To pass back strings from a filter we use var parameters, i.e. parameters that can pass back values to the calling method. For a Delphi programmer this will not create any real problems since character arrays are assignment compatible with strings, and for a C/C++ programmer they will act as C:s char * strings.

```
{
  This is the specification for the import filters
  that Predictor supports. To create a filter, implement
  all these methods in a DLL, then place the DLL
  together with any support files in the directory
  filter in the directory where Predictor is installed.
}
library DLLName;

uses ...

{
  The TStringArray type is used to pass string
  to and from the DLL.
}
type TStringArray=array[0..255] of char;

{
  The method getVersion returns the version of the
  import filter specification that the filter is
  implemented according to.

  For the current version the result should be 1.0.

  Example:
  version:='1.0';
}
procedure getVersion(var version:TStringArray); stdcall;

{
  The method getFileType returns a string that
  describes the type of files that the filter
  is used to import.

  Example:
  fileType:='Import Type';
}
procedure getFileType(var fileType:TStringArray); stdcall;
```

```
{
  The method getFileMask returns a list of file types
  that the filter accepts. This information is used
  to filter the files when the user selects a file
  for import.

  Each file type should contain a description followed
  by a vertical bar, followed by a list of file masks
  separated by semicolons. If several types are given
  should they be separated by vertical bars.

  Example:
  fileMask:='Text files|*.txt;*.asc|Any files|*.*';
}
procedure getFileMask(var fileMask:TStringArray); stdcall;

{
  The method canBeConfigured returns whether or not the
  filter can be configured.

  A value of 0 means that the filter can't be configured,
  any other value that it can.
}
function canBeConfigured: integer; stdcall;

{
  The method configure is called when the user clicks on
  the configure button in the import dialog. It is never
  called if canBeConfigured returns 0.

  This method should open a window where the user can
  configure the filter.
}
procedure configure; stdcall;

{
  The method preview returns whether or not the filter
  supports previews.

  A value of 0 means that it doesn't supports previews,
  any other value that it does.
}
function preview:integer; stdcall;

{
  The method setFileName sets the name of the file that
  the user wants to import.
}
procedure setFileName(filename:TStringArray); stdcall;

{
  The initializeFilter method is used to
  initialize the filter. Typical actions here
  are to open the file and place the position
  at the beginning of the file.

  If an error occures during the initialization
  phase, write an error message in the parameter
  error, otherwise, leave it blank.

  Note that this method can be called several
  times, e.g. once when the calling program
  wants a preview of the data, and once before
  it actually performs the import.
}
procedure initializeFilter(var error:TStringArray); stdcall;
```

```
{
  The closeFilter method is used close down the
  filter. Typical actions here are to close any
  open files.

  This method is called once, when the import is
  finished.
}
procedure closeFilter; stdcall;

{
  The method getNoOfAttributes returns the number of
  attributes in the file.

  initializeFilter has always been called before
  getNoOfAttributes is called, so you should not
  have to initialize the filter here.
}
function getNoOfAttributes: integer; stdcall;

{
  The method getAttributeName returns the name of the
  given attribute. Valid numbers for no is 0 to
  getNoOfAttributes-1.
}
procedure getAttributeName(no: integer; var attributeName:TStringArray); stdcall;

{
  The method getValue returns the value of the
  attribute with the given number. Valid numbers
  for no is 0 to getNoOfAttributes-1.

  If the attribute doesn't have a value should the
  function return the empty string.

  Note that getValue shouldn't "move around" in the
  file. It should only return the values for the current
  "line".
}
procedure getValue(no: integer; var value:TStringArray); stdcall;

{
  The method hasMoreData returns whether or not
  there are any more data to import. It also positions
  the filter on the next "line" in the file.

  A value of 0 means that there is no more data to import,
  any other value that it does.
}
function hasMoreData: integer; stdcall;

{
  A list of the functions that the filter exports.
}
exports
  getFileType, getFileMask, getNoOfAttributes,
  getAttributeName, getValue, getVersion, setFileName,
  canBeConfigured, configure, initializeFilter, closeFilter,
  hasMoreData, preview;
end.
```

7.2. C/C++ Example Filter

This section provides a C/C++ example implementation of the interface, The code was written in Borland C++ Builder, but should be easy to adapt to any other C/C++ dialect. Worth to note is that the code is intended only as an example and contains no real import functionality. No matter what the content of the file it is supposed to import it will return five projects with two attributes each with the values 1 and 2 respectively.

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------

/*
  The TStringArray type is used to pass string
  to and from the DLL.
*/
typedef char TStringArray[256];
//---------------------------------------------------------------------------

extern "C" __declspec(dllexport) void __stdcall
                        convertToArray(const char * str, char * arr);
extern "C" __declspec(dllexport) void __stdcall getVersion(TStringArray version);
extern "C" __declspec(dllexport) void __stdcall getFileType(TStringArray fileType);
extern "C" __declspec(dllexport) void __stdcall getFileMask(TStringArray fileMask);
extern "C" __declspec(dllexport) int __stdcall canBeConfigured();
extern "C" __declspec(dllexport) void __stdcall configure();
extern "C" __declspec(dllexport) int __stdcall preview();
extern "C" __declspec(dllexport) void __stdcall setFileName(TStringArray filename);
extern "C" __declspec(dllexport) void __stdcall
                        initializeFilter(TStringArray error);
extern "C" __declspec(dllexport) void __stdcall closeFilter();
extern "C" __declspec(dllexport) int __stdcall getNoOfAttributes();
extern "C" __declspec(dllexport) void __stdcall
                        getAttributeName(int no, TStringArray attributeName);
extern "C" __declspec(dllexport) void __stdcall
                        getValue(int no, TStringArray value);
extern "C" __declspec(dllexport) int __stdcall hasMoreData();

//---------------------------------------------------------------------------
int pos;    // Used to keep track of how man lines that has been read
//---------------------------------------------------------------------------

/*
   The method convertToArray is used to convert a
   char * to a TStringArray. This could of course
   be done with functions from the C standard libraries
   but this way is more explicit.
   The method is not a part of the filter interface
*/
void __stdcall convertToArray(const char * str, char * arr){
    int n;
    for(n=0; n<255 && str[n]; n++){
        arr[n]=str[n];
    }
    arr[n]=0;
}

/*
  The method getVersion returns the version of the
  import filter specification that the filter is
  implemented according to.

  For the current version the result should be 1.0.
*/
void __stdcall getVersion(TStringArray version){
    convertToArray("1.0",version);
}
```

```
/*
  The method getFileType returns a string that
  describes the type of files that the filter
  is used to import.
*/
void __stdcall getFileType(TStringArray fileType){
    convertToArray("C++ Example Type",fileType);
}

/*
  The method getFileMask returns a list of file types
  that the filter accepts. This information is used
  to filter the files when the user selects a file
  for import.

  Each file type should contain a description followed
  by a vertical bar, followed by a list of file masks
  separated by semicolons. If several types are given
  should they be separated by vertical bars.
*/
void __stdcall getFileMask(TStringArray fileMask){
    convertToArray("Text (*.txt)|*.txt",fileMask);
}

/*
  The method canBeConfigured returns whether or not the
  filter can be configured.

  A value of 0 means that the filter can't be configured,
  any other value that it can.
*/
int __stdcall canBeConfigured(){
    return 0; // This filter can't be configured
}

/*
  The method configure is called when the user clicks on
  the configure button in the import dialog. It is never
  called if canBeConfigured returns 0.

  This method should open a window where the user can
  configure the filter.
*/
void __stdcall configure(){
    // Since this method should never be called
    // so we don't have to do anything
}

/*
  The method preview returns whether or not the filter
  supports previews.

  A value of 0 means that it doesn't supports previews,
  any other value that it does.
*/
int __stdcall preview(){
    return 1; // We support previews
}

/*
  The method setFileName sets the name of the file that
  the user wants to import.
*/
void __stdcall setFileName(TStringArray filename){
    // We ignore the filename
    // since the data will be fabricated
}
```

```
/*
  The initializeFilter method is used to
  initialize the filter. Typical actions here
  are to open the file and place the position
  at the beginning of the file.

  If an error occures during the initialization
  phase, write an error message in the parameter
  error, otherwise, leave it blank.

  Note that this method can be called several
  times, e.g. once when the calling program
  wants a preview of the data, and once before
  it actually performs the import.
*/
void __stdcall initializeFilter(TStringArray error){
    // Since we don't read anything we don't have to
    // open any files, so we just set the position
    pos=0;
}

/*
  The closeFilter method is used close down the
  filter. Typical actions here are to close any
  open files.

  This method is called once, when the import is
  finished.
*/
void __stdcall closeFilter(){
    // Since we don't have any opened files we
    // don't have to close them either
}

/*
  The method getNoOfAttributes returns the number of
  attributes in the file.

  initializeFilter has always been called before
  getNoOfAttributes is called, so you should not
  have to initialize the filter here.
*/
int __stdcall getNoOfAttributes(){
    // There are two attributes
    return 2;
}

/*
  The method getAttributeName returns the name of the
  given attribute. Valid numbers for no is 0 to
  getNoOfAttributes-1.
*/
void __stdcall getAttributeName(int no, TStringArray attributeName){
    // In this case we give the attributes imaginative names :-)
    // Normally these would have been read from the
    // file during initialization of configuration
    if (no==0){
        convertToArray("Attribute 1",attributeName);
    } else {
        convertToArray("Attribute 2",attributeName);
    }
}
```

```
/*
  The method getValue returns the value of the
  attribute with the given number. Valid numbers
  for no is 0 to getNoOfAttributes-1.

  If the attribute doesn't have a value should the
  function return the empty string.

  Note that getValue shouldn't "move around" in the
  file. It should only return the values for the current
  "line".
*/
void __stdcall getValue(int no, TStringArray value){
    if (no==0){
        convertToArray("1",value);
    } else {
        convertToArray("2",value);
    }
}

/*
  The method hasMoreData returns whether or not
  there are any more data to import. It also positions
  the filter on the next "line" in the file.

  A value of 0 means that there is no more data to import,
  any other value that it does.
*/
int __stdcall hasMoreData(){
    pos++;
    return pos<=5;
}

//---------------------------------------------------------------------------
int WINAPI DllEntryPoint(HINSTANCE hinst, unsigned long reason, void*)
{
    return 1;
}
//---------------------------------------------------------------------------
```

8 References

H. Bergström, "A Family of Delphi Components for Case-Based Reasoning," in *Proceedings of ICTAI'99*, 1999.

B. W. Boehm, *Software Engineering Economics*: Prentice Hall, 1981.

B. W. Boehm, et al, *COCOMO II Model Definition Manual*: South Carolina University, 1998.

W. S. Humphrey, *A Discipline for Software Engineering*: Addison Wesley, 1995.

Appendix A - Metrics of Estimation Accuracy

The simplest error metric available is the Mean Error (ME) which is simply the average error.

$$\text{ME}=\frac{\sum_{i=1}^{n}(A_i-E_i)}{n}$$

The usability of ME is hampered by the fact that positive and negative errors cancel each other out. Therefore the ME is mostly useful to detect estimation bias. The Mean Absolute Error (MAE) avoids this by taking the mean of the absolute value of the errors.

$$\text{MAE}=\frac{\sum_{i=1}^{n}(|A_i-E_i|)}{n}$$

The Mean Relative Error (MRE) addresses another problem with both ME and MAE, namely that they do not take size into account. An estimation error of 100 person hours means little if the projects duration is several person years, but very much if it is only a few person months. Together with Pred(x) (discussed below) MRE are one of the most commonly used measurements of estimation accuracy in the cost estimation literature.

$$\text{MRE}=\frac{1}{n}\sum_{i=1}^{n}\frac{|A_i-E_i|}{A_i}$$

The Weighted Mean Relative Error is a relatively uncommon metric. Like MRE it takes size into account.

$$\text{WMRE}=\frac{\sum_{i=1}^{n}|A_i-E_i|}{\frac{1}{n}\sum_{i=1}^{n}A_i}$$

The Tracking Signal (TS) is a measurement of estimation bias defined as:

$$\text{TS}=\frac{\sum_{i=1}^{n}(A_i-E_i)}{MAE}$$

Pred(x) is defined as the percentage of the estimations which fall within x% of the actual value. It is more resistant to outliers than the other metrics used making it a good complement.

All these metrics can be calculated both on manual estimations, and on the estimations that Predictor does automatically.

Appendix B - End User Licence Agreement

End user license agreement: Predictor

Notice to user:

This is a contract. By installing this software you accept all the terms and conditions of this agreement.

The term software in this licence agreement refer to the program Predictor and all accompanying material such as manuals, helpfiles etc. The terms we and us refers to the development team of Predictor. Please read this agreement carefully. At the end, you will be asked to accept this agreement and continue to install or, if you do not wish to accept this agreement, to decline this agreement, in which case you will not be able to use the software. Failure to follow the agreement might result in us taking legal actions to protect our rights.

Upon your acceptance of this agreement, we grant to you a nonexclusive license to the use of one (1) copy of the software, provided that you agree to the following:

Registration

If you continue to use Predictor you are obliged to register it. Registration is free, but involves providing us with some information and feedback on Predictor and you usage of the tool, information which will be used by us in our research. However you may provide as much or as little information as you like. The program will indicate when it is time to provide this information, a procedure that will be undertaken two (2) times.

Distribution of the software

Predictor may be distributed freely as long as nothing is changed in it, and **nothing** except media costs is charged for it. However serial numbers and registration keys may **not** be distributed. These must be requested from us, either through email `predictor@dsv.su.se` or through the programs homepage `http://www.dsv.su.se/~henrikbe/Predictor` This restriction is because we need to keep track of how many people use the program.

Copyright

The software is owned by us, and its structure, organisation and code are our property. The software is also protected by Swedish Copyright Law and International Treaties.

Restrictions to usage

You agree not to modify, adapt, reverse engineer, decompile, disassemble or otherwise attempt to discover the source code of the software.

No Warranty

The software is being delivered to you AS IS and we make no warranty as to its use or performance. WE DO NOT AND CANNOT WARRANT THE PERFORMANCE OR RESULTS YOU MAY OBTAIN BY USING THE SOFTWARE OR DOCUMENTATION, AND WE MAKE NO WARRANTIES, EXPRESS OR IMPLIED, AS TO NONINFRINGEMENT OF THIRD PARTY RIGHTS, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL WE BE LIABLE TO YOU FOR ANY CONSEQUENTIAL, INCIDENTAL OR SPECIAL DAMAGES, INCLUDING ANY LOST PROFITS OR LOST SAVINGS, EVEN IF AN WE HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY THIRD PARTY.

Termination of agreement

This agreement shall automatically terminate upon failure by you to comply with its terms in which case you must remove the software from your system immediately.

Index

A

Algorithmic Accuracy Button 8, 20, 21
Algorithmic Estimation Button 8, 20
Algorithmic Model Jackknifing Results Report ...30
Algorithmic Model Prediction Report 30
Algorithmic Model Report.....29
Algorithmic Models.....5
Algorithmic Models Supported 5
Algorithmic tab 8, 21, 30
Analogies tab.....17
Arithmetical Mean.....17
Artificial Intelligence.....6
Ascii Files 18
Attribute Weights 15
Attribute Weights Window 8, 15

B

Batch Test 9, 19, 28
BDE ..... See Borland Database Engine
Borland Database Engine.....3

C

Capability Maturity Model.....1
Case-Based Reasoning.....6
CBR ..... See Case-Based Reasoning
CBR Accuracy Button 8, 19
CBR Batch Test Results Report 28
CBR Components.....6
CBR Estimation Button 8, 15, 17
CBR Jackknifing Results Report.....27
CBR Prediction Report 27
CBR Prediction tab.....8, 16, 17, 19, 27
Changing Size of Window Areas 14
CMM ..... See Capability Maturity Model
COCOMO.....5
Contact Information.....2
Cost Estimation Techniques Supported 5
Creating a new Database Type 23
Creating a Project Database 11

D

Database Contents Report.....25
Database Design Window.....23, 24
Database Types.....23
Delete Project Button.....8
Develop Algorithmic Model Window.....20, 21
Develop Model Button.....8, 20
DLL .....See Dynamic Link Library
Downloading Predictor.....1, 3
Dynamic Link Library 31

E

Earlier Predictions Report.....26
Edit Mode 13
Edit Project Button 8, 13
End User Licence Agreement.....43
Estimation vs Prediction 6
Example Database Type..... 12
Expert Judgement 5
Export Data 9

F

Feedback on the Quality Estimates..... 22

G

Geometrical Mean 17
Guided Tour 11

H

Help Button 8

I

Import Data 9, 18
Import Dialog 9, 18
Import Filters..... 31
  C/C++ Example..... 34
  Delphi Interface..... 31
  Installing 31
  Strings..... 31
Installing Import Filters..... 31
Installing Predictor..... 3
Introduction 1
ISO 9000 1

J

Jackknifing 19, 23, 27, 30

L

Licence Agreement..... 43
Local Algorithmic Models 5

M

Main Window.....7, 11, 14
Making a CBR Prediction 15
Manual Estimation..... 14
Manual Estimation Accuracy Button 8, 22, 23
Manual Estimation Button.....8, 14
Manual Prediction/Adjustment Window.....14, 27
Mean Error 41
Mean Relative Error.....19, 41
Menubar 9
Metrics of Estimation Accuracy 41
MRE.....See Mean Relative Error

N

New Database Button 8, 11
New Database Window.....12, 23
New Project Button.....8, 13
No Weight File Dialog 15

O

Open Database Button 8

Options 9

P

Personal Software Process 1
Pred(x) 19, 41
Prediction Accuracy Report 26
Print Button.....8
Print Dialog.....25
Printing Reports.....25
PSP .....See Personal Software Process

R

Recording Data.....13
References.....39
Registering Predictor 4
Registration Dialog.....4
Reports.....25

S

Save Database As Button.....8
Search Button.....8, 21
Search Dialog.....21, 22
Serial Number 3
Set Attributes Button 8, 17
Set Prediction Attributes Window.....17
Set Weights Button 8
Setting up Predictor 3
Setting Weights 15
Show Selected Only Button.....8, 21
Size Adjusted..... 17
System Requirements..... 3

T

Threshold 17
Toolbar..... 8
Tracking Signal 41
TS ..... See Tracking Signal

U

Using Predictor with Very Little Data 14

W

WED .....See Weighted Euclidean Distance
Weight File Report..... 29
Weighted Euclidean Distance..... 6
Weighted Mean Relative Error..... 41
Weights 15
Visible Projects.....9, 14
WMRE .....See Weighted Mean Relative Error
Voting 17