MAISA metrics tool: User manual

version 1.3

Juha Gustafsson and Lilli Nenonen

Helsinki December 17, 2001

UNIVERSITY OF HELSINKI
Department of Computer Science
MAISA project

Contact information

Jukka Paakki:	email Jukka.Paakki@cs.helsinki.fi
Inkeri Verkamo:	email Inkeri.Verkamo@cs.helsinki.fi
Juha Gustafsson:	email Juha.Gustafsson@cs.helsinki.fi
Lilli Nenonen:	email Lilli.Nenonen@cs.helsinki.fi

TEKES
Matti Sihto: email Matti.Sihto@tekes.fi

Kone
Salla Mannerjoki: email Salla.mannerjoki@kone.com

Nokia Mobile Phones
Markku Ilvesmäki: email Markku.Ilvesmaki@nmp.nokia.com

Nokia Research Center
Anssi Karhinen: email Anssi.Karhinen@nokia.com

Space Systems Finland
Niklas Holsti: email Niklas.Holsti@ssf.fi

Version history

Version	Date	Changes / Comments
0.0	10.10.2000	The first version.
0.1	02.11.2000	Patch version. Chapter "Main window": Saving results in HTML format and file format identification when saving results.
0.2	17.01.2001	Patch version, after getting user experiences Chapter "Installation": Some new points about the files and running the tool for the first time. Chapter "Main window": Two new user preferences and browse buttons in the preference window. All the preferences in use immediately. Chapter "Metric window": The not-yet-supported shown disabled in the metric window.
0.3	13.02.2001	Maisa Version 1.0. Chapter "Main window":Remembering program state between the sessions. Chapter "Pattern window": figures of pattern window changed.
0.4	13.02.2001	Maisa Version 1.0. Pictures of windows containing tree shaped lists updated (due to adding new, Maisa specific icons to the lists).
0.5	27.02.2001	Maisa Version 1.0. Command-line section updated
0.6	30.07.2001	Maisa Version 1.1. Chapter "Main window": Statistic features and pattern visualization added More pictures of the main window and its new tab sheet structure. Calculation threads added in the main program. Chapter "History data window" added.
0.7	05.09.2001	Maisa Version 1.2. Chapter "Main window/Statistics": Pattern distrubution added as a new diagram type. Chapter "Metric window": Pattern count and pattern instance count added as new pattern metrics. Chapter "History data window": Pattern distribution added as a new diagram type.
1.0	14.12.2001	Maisa Version 1.3. Chapter “Main window/Statistics”: support for comparison results added. Chapter "History data window": support for comparison added. Chapter “Main window”: history data file added as a new user preference. The document proofread.
1.1	17.12.2001	Maisa Version 1.3. User profile description moved from Chapter “Main window ” to Chapter “Introduction”. UML Resource Center link added to bibliography.

Contents

1 Introduction **1**

2 What MAISA is **2**

3 Installation **4**

4 Main window **5**

4.1 Architectures: projects and diagrams 8

4.2 Measuring 9

4.3 Pattern visualization 13

4.4 Statistics 14

4.5 Comparing 17

4.6 Preferences 17

5 Metrics window **19**

5.1 Size metrics 21

5.2 Performance metrics 21

5.3 Understandability metrics 22

5.4 Pattern recognition 22

6 Pattern window **24**

6.1 Pattern description 27

6.2 Edit mode 28

6.2.1 Prolog file 28

6.2.2 Index file 29

7 History data window **31**

8 Help window **34**

9 Using MAISA in command-line **36**

9.1 Pattern library directory 37

9.2 Help directory 42

A Current patterns in the library **i**

B File formats **vi**

B.1 Metrics result files vi
B.2 Comparison result files xii
B.3 History data files xv
B.4 Metric configuration files xvii

References **xix**

1 Introduction

The MAISA metrics tool is designed to measure object-oriented software architecture descriptions that are given in the UML language [UML]. The measurement can thus be done in early phases of the software development process and the software life cycle. The tool combines ordinary measures with design pattern and anti-pattern recognition and makes use of known design patterns while measuring the UML description.

The MAISA tool is intended to be used by software designers, test designers and quality managers. The MAISA user is expected to have basic knowledge on object-orientation, UML, design patterns, measurement and software quality, which helps her to understand the basic concepts of the tool. However, she needs not to be an expert on these.

In Chapter 2 we present general ideas behing MAISA. In Chapter 3 we explain how to install the MAISA tool in your personal computer.In Chapters 4 - 8 we tell how to use the windows of the MAISA tool. In Chapter 9 we tell how to use MAISA without the graphical user interface.

The MAISA metrics tool is implemented as a result of the MAISA research project in the University of Helsinki. The project is sponsored by TEKES, Kone Oyj, Nokia Mobile Phones, Nokia Research Center and Space Systems Finland.

This is the User manual for MAISA version 1.3 (manual version 1.0).

2 What MAISA is

The MAISA tool is intended for measuring UML diagrams, which have been previously constructed by another tool. It is not a CASE tool or a diagram drawing program. It reads UML descriptions and prints out the measurement results but it does not modify the diagrams.

In addition to measuring single UML diagrams, the user can compare two existing diagrams. The metrics results can be saved separately from the diagrams and later compared to each other.

The MAISA tool recognizes design patterns and anti-patterns that are used in the given software architecture. The tool has a pattern library, which contains information of both the structure of patterns and their influence on the properties of the architecture. The user can browse the pattern library. A professional user may also need to edit and update the library.

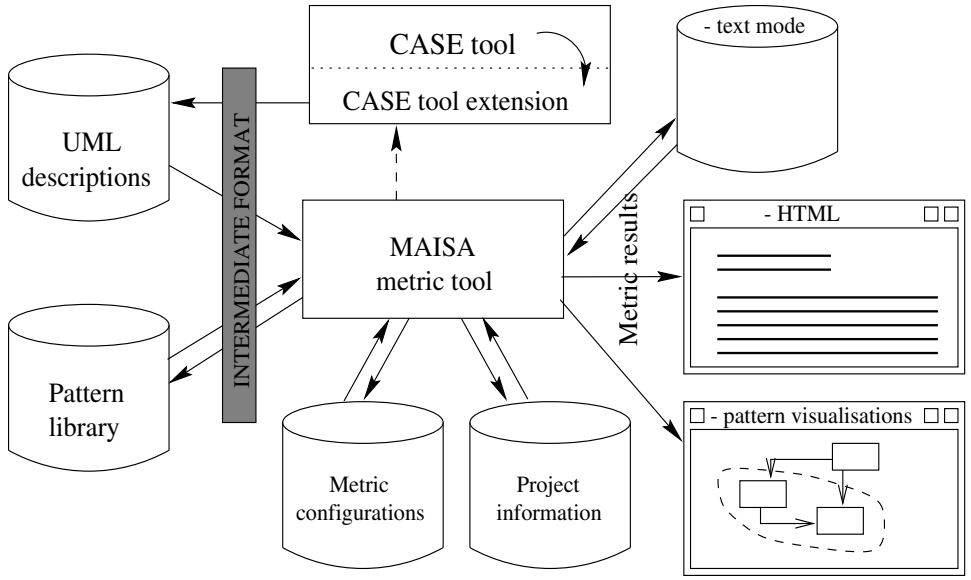

Figure 1: Overall architecture of MAISA.

Every measure consists of metrics. The user can browse the metrics and see what the measurement is based on. Because some of the metric results are estimates rather than accurate values, browsing the metrics is important especially for the professional user, who also may need to parametrize the metrics to tailor the tool for her needs. It must be noted that the MAISA tool calculates *accurate values of some metrics* and – in addition to that – *predicts the values of the other metrics*. In other words, the purpose of the MAISA tool is both calculating metrics values of the architecture and foretelling some properties of the to-be software product.

Figure 1 describes the architecture of the tool in a general level. The solid arrows present data flow and the dashed one presents control flow. Because UML is a visual language, the diagrams must be saved in an intermediate file format which is analyzable and measurable.

The UML diagrams are drawn with a separate CASE tool which generates the diagrams in the chosen intermediate format. In fact, an extension to the CASE tool produces the format, because at present no commercial CASE tool supports the intermediate file format used by the MAISA tool. The MAISA tool reads this representation and gives the results in textual form. The results can also be presented in HTML. The pattern library is separate from the UML descriptions. In addition to the metrics results, the MAISA tool prints out some additional files used by the tool itself, not directly by the user (namely project and metrics configuration files. They are discussed later). The control flow presents the possibility to run the CASE tool via the MAISA tool. Figure 1 lacks information of the additional files. They are used only by the tool itself, never by the user.

The MAISA tool is a single user program requiring no usernames or passwords. The pattern library is implemented as a group of text files, not a database. Also the measurement results are saved as fairly simple text files. The tool is implemented in Java^{TM}. It is designed to be run in Windows and Linux environments, but the implementation language does not prevent running the MAISA tool on other operating systems.

3 Installation

The MAISA system can be downloaded at the web site of the MAISA project, at `http://www.cs.helsinki.fi/group/maisa/`. Version 1.3 is delivered in two zip packages. The first package contains the actual program files and the other package contains the pattern library and on-line help files.

Windows:

1. Install the Java^{TM} 2 Runtime Environment if you don't have it already. It can be downloaded at `http://java.sun.com/products/jdk/1.2/jre/`.
2. Copy the zip packages into your computer and unzip them into `C:\maisa\`. This is called your MAISA directory.
3. Add the above-mentioned directory in your classpath (an environment variable used by the Java^{TM} system). In Windows NT, Start Control Panel, select System and click the Environment tab. Set the CLASSPATH variable in the User Variables section. In Windows 95 or 98, add the classpath to the autoexec.bat file.

Unix:

1. Install the Java^{TM} 2 Runtime Environment if you don't have it already. It can be downloaded at `http://java.sun.com/products/jdk/1.2/jre/`.
2. Copy the zip packages into your home directory as `~/maisa/`. This is called your MAISA directory.
3. Add the above-mentioned directory in your classpath (an environment variable used by the Java^{TM} system) with command `setenv CLASSPATH $CLASSPATH:$HOME/maisa/`.

The tool is started with command `java MAISA.MAISA_tool`. The tool can be used also with a command line interface (it is explained in Chapter 8).

Your MAISA directory now contains the Java^{TM} class files, the default metrics configuration file and the pattern library and help directories. When you use MAISA for the first time, MAISA generates the default metrics configuration file (`maisa/default.cfg`) and the pattern directory (`maisa/pl/`). The history data is by default saved in `maisa/history.data`. You can change the file names later by editing the user preferences (the contents of them are edited in the metrics window and in the pattern window). MAISA generates also two working files (for the use of merely the tool itself): `maisa/default.rsc` and `maisa/default.prp`. They contain the user preferences and the state of the MAISA program.

(A note to unix users: the tool can be started in any working directory. If it can not find your MAISA directory from the system properties, start the tool in your home directory.)

4 Main window

The MAISA tool is started with command `java MAISA.MAISA_tool` When the MAISA tool is started, the main window is opened. In this chapter we tell how to manage diagrams, projects, metrics results and comparison results in the main window. Initially the main window is empty. The tool contains four basic windows: the main window, the metrics window, the pattern (library) window and the history data window.

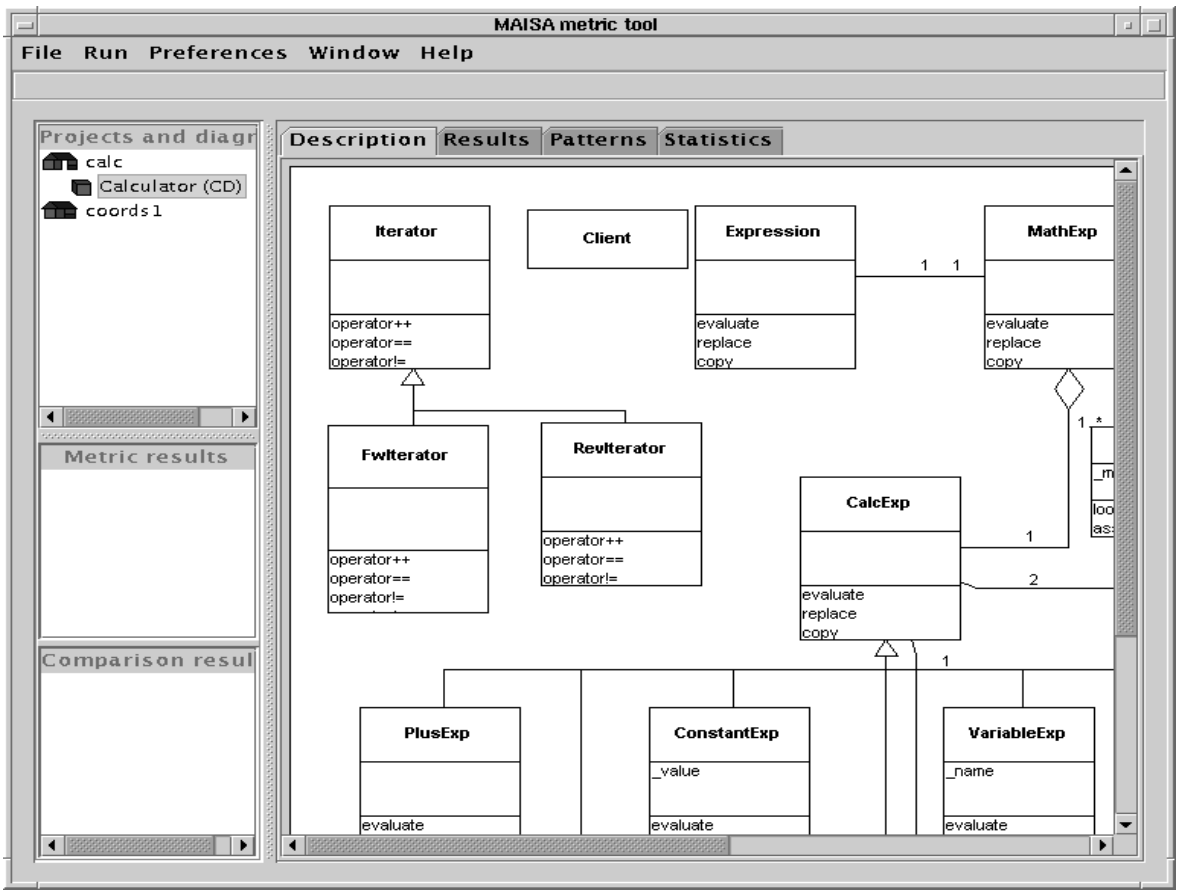

Figure 2: The main window.

Figure 2 presents the main window with some projects, diagrams and metrics results. The upper left part of the window contains a list of projects. A project contains one or more diagrams – the structure is shown in the tree-shaped list. Single diagrams (ie diagrams that do not belong to any project) are shown under the projects in the list. The concept of projects is explained in Chapter 4.1. The lower left part contains two lists: the metrics results and the comparison results (the concepts are explained in Chapters 4.2 and 4.5). The rest of the window contains four tab sheets: Description, Results, Patterns and Statistics.

The diagram selected in the project list is shown in the Description sheet. The selected metrics result or comparison result is shown in the Results sheet. One result (either metrics *or*

comparison) can be selected at a time. The lists and panels can be resized by dragging their borders. The design patterns of the selected metrics result are shown in the Patterns sheet and diagrams of the selected result can be drawn in the Statistics sheet. Note that the Patterns sheet is used for only metrics results, not comparison results.

The metrics and comparison results can be read and saved in MAISA's own formats (extension `.res` for metrics results and `.cmp` for comparison results). Metrics results can also be saved in a special read-and-write format to be used by unix tools like grep (extension `.unix`). Both metrics and comparison results can be saved also in html, but the tool does not read html. Metrics results can also be saved in a history data archive. The history data is collected to check out e.g. how the metrics values have changed in function of time and graphs of the history data can be drawn in the History data window. The history data is saved in unix-like format, so it can be handled with unix tools, too. The file formats are described id detail in Appendix B.

Under the main menu is the status bar for the messages and error messages given by MAISA. The main menu contains five menus: File, Run, Preferences, Window and Help.

The **File** menu contains common file operations. The selections that cannot be performed at the moment are disabled. The menus are classified according to the files handled (projects or results):

Open: Project / Diagram Opens a project or a diagram in the project list. Both directories and single files can be opened: if you open a directory, it is considered a project; if you open a prolog file that ends with `.pro`, it is considered a diagram. If a project is selected in the project list when you open a new diagram, the diagram belongs to (or becomes a property of) the selected project during the session. If nothing or a diagram is selected in the project list when you open a new diagram, the diagram is opened as a single diagram, not belonging to any project.

Open: Metrics / Comparison result Opens a metrics result in the metrics result list or a comparison result in the comparison list. Only single files can be opened. If the file ends `.res` or `.unix`, it is considered a metrics result; if the file ends `.cmp`, it is considered a comparison result. A project or a diagram must be selected in the project list before opening a result. The opened result belongs to (or becomes a property of) the selected project or diagram during the session.

Close: Project / Diagram Removes the selected project or diagram (with their results) from the project list.

Close: Metrics / Comparison result Removes the selected result from the metrics result or comparison result list.

Save: Metrics / Comparison result Saves the selected metrics or comparison result. Asks the user the file name where to copy. When saving a metrics result, use `.res` for the ordinary MAISA format, `.unix` for the unix format or `.html` for html. When saving a comparison result, use `.cmp` for the ordinary MAISA format or `.html` for html.

Save metrics to history data Copies the data of the selected metrics results into the history data archive.

Clear history data Empties the history data archive.

Exit Quits the program.

The measurements and comparisons are calculated in their own threads. The ongoing threads are shown in the thread dialog, where they can also be cancelled if necessary. When a thread is created, it appears in the thread dialog and when it is stopped or when it is cancelled, it disappears from the dialog. The **Run** menu contains the measurement commands:

Measure Measures the selected project or diagram using the current metrics configuration. A project or a diagram must be selected before performing the operation. The calculated metrics results appear in the metrics result list.

Compare Compares the two selected projects, diagrams, metrics results or the selected combination of them. The two items must be selected before performing the operation. The calculated comparison results appear in the comparison result list. If two architectures are selected, they are compared. If two metrics results are selected, they are compared. Otherwise, the selected architecture and the metrics result are compared.

Show thread dialog Opens the thread dialog.

The **Preferences** menu contains the user settings of the tool. The settings can be edited in the preference window. The boolean valued preferences can also be edited directly by selecting them in the menu. Note, that updating the preferences in the preference window saves the preferences in the disk, but setting the (boolean valued) preferences in the preference menu does not save them in the disk before you click the save item in the menu.

Edit... Opens the preference window where you can edit all the preferences (see Chapter 4.6).

Confirm unsaved files when closing A boolean preference. If it is selected, the program asks for confirmation before closing a result file that is not saved in the disk or closing an architecture containing result files that are not saved in the disk.

Confirm unsaved files on exit A boolean preference. If it is selected, the program asks for confirmation before exiting the program if there are open files that are not saved in the disk.

Confirm pattern / pattern group deletion A boolean preference. If it is selected, the program asks for confirmation before deleting a pattern or a pattern group in the pattern library window.

Confirm pattern recognition A boolean preference. If it is selected, the program asks for confirmation of the recognized pattern instances after running pattern recognition.

Save Saves the preferences in the disk.

The **Window** menu contains commands to open the two other basic windows of the tool:

Measures and metrics Opens the metrics window.

Pattern library Opens the pattern library window.

History data Opens the history data window.

The **Help** menu gives help to the user:

Help... Opens the help window.

About Gives information of the MAISA project.

4.1 Architectures: projects and diagrams

A project is a set of diagrams. Grouping diagrams into projects is a way of measuring more than one diagrams at the same time. A project can include different types of diagrams, one or more diagrams of each type. The supported UML diagram types are *class diagram, activity diagram, state transition diagram, sequence diagram* and *collaboration diagram*. MAISA supports also *extended activity diagrams* that contain information of performance requirements [Gus2000]. A project is also a directory, containing all the diagrams belonging to the project. The name of the directory is the name of the project. In practice, **a diagram** is a prolog file, presenting the information of a UML diagram. There can also be a figure file showing the visual diagram. Both projects and diagrams are called **architectures**. The project directory can contain following files read by MAISA, all the other files in the directory are ignored:

Diagram files in prolog format, named `xxx.pro` where xxx is the diagram name (generated by an external CASE tool). These are the only files in the project directory that are essential in metrics calculation.

Diagram figure files in gif format, named `xxx.gif` where xxx is the diagram name. Generated by an external drawing tool. These are used in the user interface.

Metrics result files with extension `.res`. Produced by MAISA.

Comparison result files with extension `.cmp`. Produced by MAISA.

You can open both projects and single diagrams in MAISA. The Open operation is located in the File menu. If you open a directory, MAISA considers it as a project – if you open a single file, it is considered as a diagram. You can also add new diagrams to a project or remove diagrams from a project. If a project is selected when opening a diagram, the opened

diagram is added to the project; if a diagram is selected when opening another diagram, the diagram is opened as a single diagram, not belonging to any project. Architectures can also be removed from the project list by selecting the architecture and clicking the Close item in the File menu. Note that these modifications are only temporary and they exist only during the session: opening or closing architectures does not modify the directory structure.

4.2 Measuring

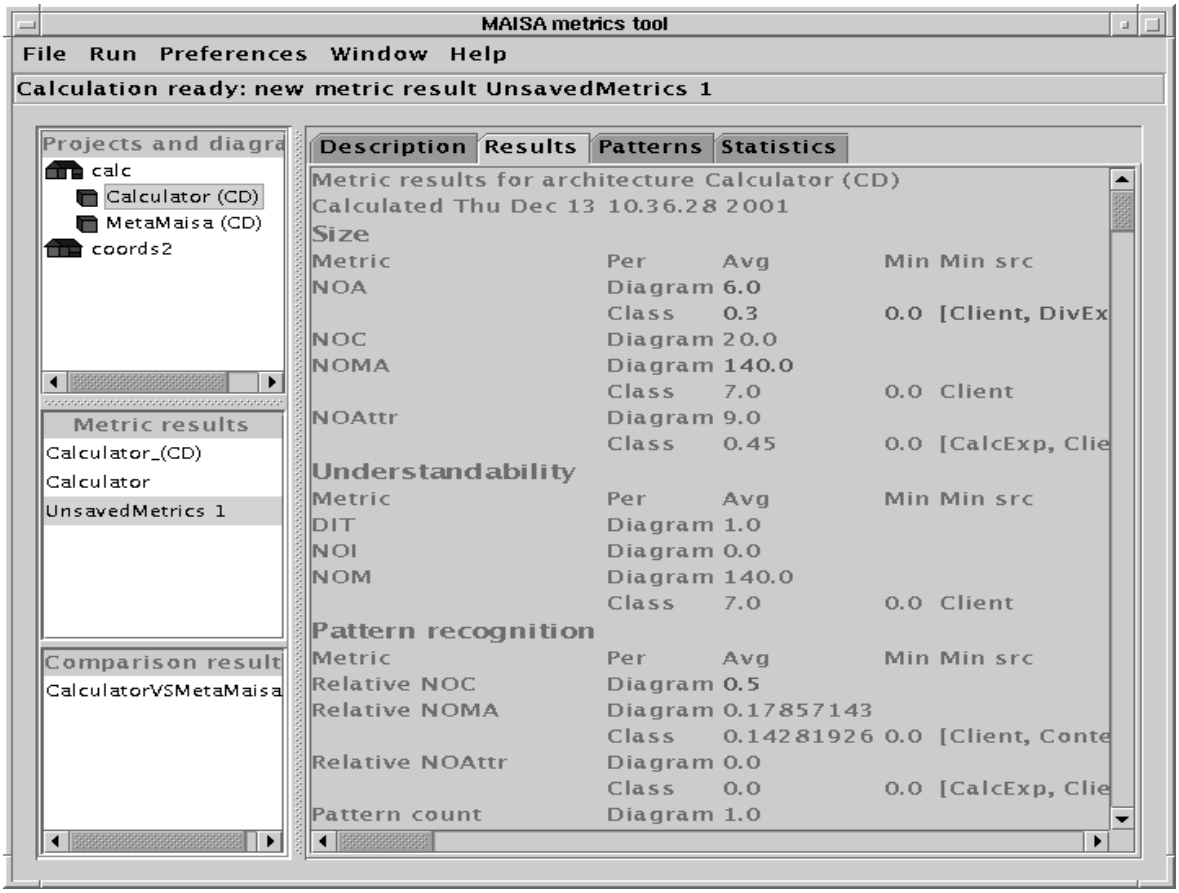

Figure 3: The main window showing metrics results.

To measure a diagram or a project, select the architecture and then click the Measure item in the Run menu. The calculates metrics results appear in the Results tab sheet. The **metrics configuration** that is calculated is set in the metrics window.

Figure 3 presents the main window showing calculated metrics results. Certain metrics cannot be measured from every kind of diagrams. E.g. if number of classes (NOC) is calculated from an activity diagram, the result is zero. In Chapter 5 we list the metrics and the diagrams they can be calculated from.

Diagrams are measured simply by calculating the metrics from the diagram (prolog) file. When a project is measured, the metrics are calculated from all its diagrams at the same time. The diagrams are unified and the metrics are calculated from these "unified diagrams", whenever possible. Class diagrams containing common classes can be unified using the common classes. Thus, the common classes are taken into account in metrics calculation only once. For example, a class diagram can present a class library, referenced from other class diagrams. The activity diagrams are unified in the same way. An activity diagram can present a procedure called from other activity diagrams.

Some metrics can be calculated with respect to project, diagram and/or diagram element (class, object, etc). A calculated metrics result (per an architectural element) is actually a set of results: minimum, maximum and average values and the element(s) that has been the source of those values. For example, the number of associations (NOA) can be calculated per project, per diagram and per class. The minimum value per class is e.g. 1, the maximum 9 and the average 4.8, with the minimun source class A and D (classes A and D both have only 1 association) and the maximum source class Z (class Z has 9 associations). In the Figure 3 you can see how the metrics values are listed. One row contains the name of the metrics (header Metric), the relation of which it is given (Per), average value (Avg), minumum value (Min), the source of the minimum value (Min src), the maximum value (Max) and the source of the maximum value (Max src).

The recognized patterns are listed below the other metrics results. A pattern instance is a set of bindings between the pattern roles and architectural elements (i.e. classes, associations, etc). The recognized instances are numbered and the number (header "Instance x", where x is the number) and the bindings in them are listed in the window under the name of the pattern.

If the measured values are not within the user given thresholds (how to set thresholds for a metrics is explained in Chapter 5) they are written in red colour in the result panel – in other case they are written with black colour. The thresholds and other metrics settings are given in the metrics window.

The recognized patterns are shown in Patterns tab sheet. It is discussed in Chapter 4.3.

Let's study a brief example. Figure 4 presents a class diagram of a company data system. The company is divided into departments and the employees and the managers who are persons, belong to a department. The departments use subcontractors, which are presented in Figure 5. The subcontractors produce products, which are composed of parts. The class Subcontractor in the latter class diagram is referenced from the former class diagram, and because it is common to both class diagrams, they can be unified. The unified class diagram is shown in Figure 6.

As an example, the Number Of Classes (NOC) per the example project is 8, and the average NOC per diagram is 4.5.

MAISA supports a set of software metrics, traditionally calculated from source code. In MAISA they are applied to the UML diagrams. The metrics provided by MAISA are explained in Chapter 5.

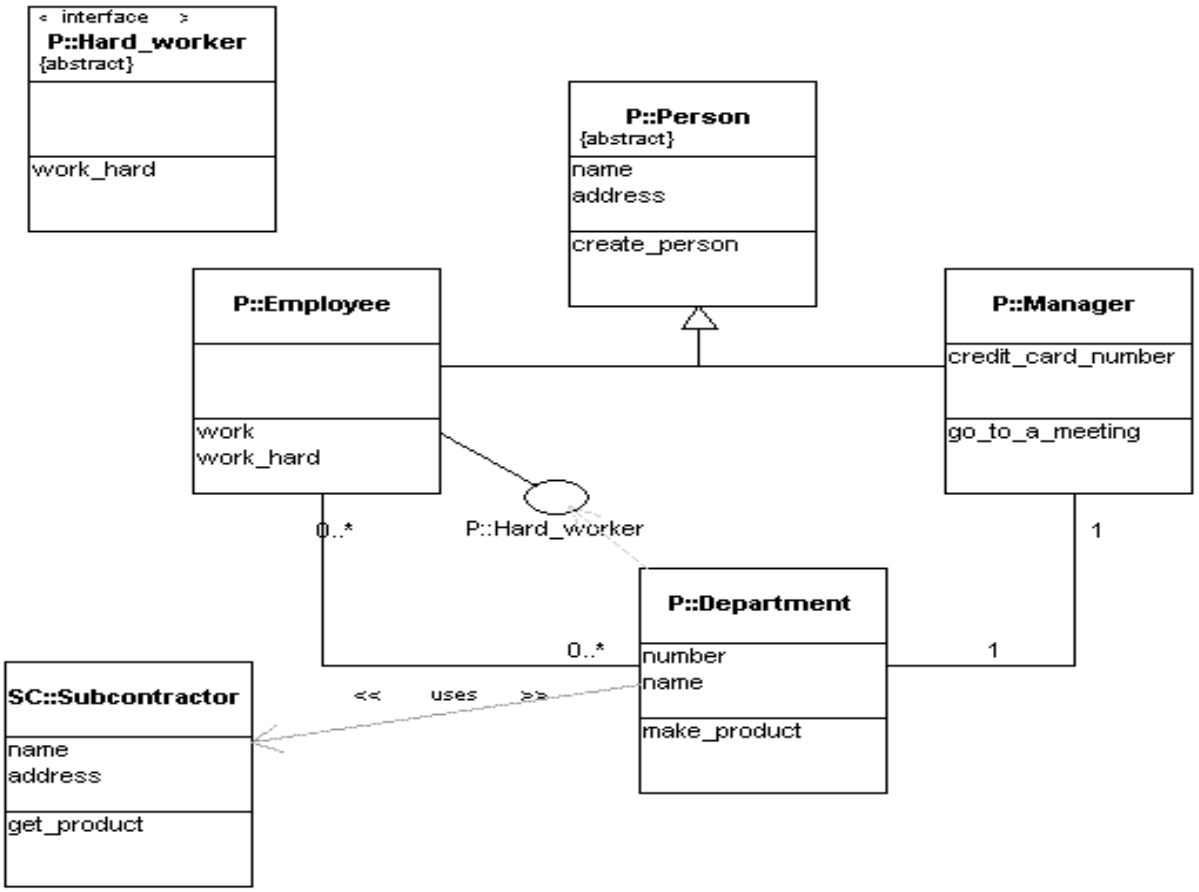

Figure 4: The company class diagram.

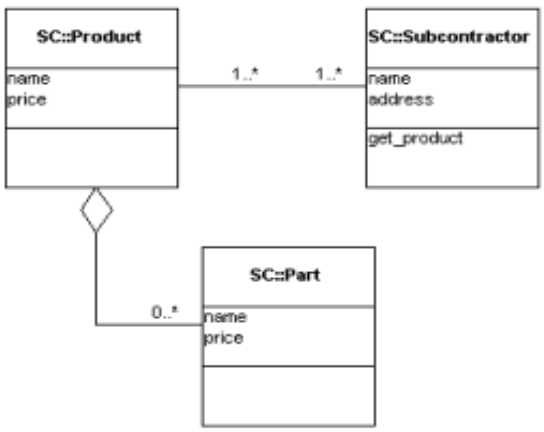

Figure 5: The subcontractor class diagram.

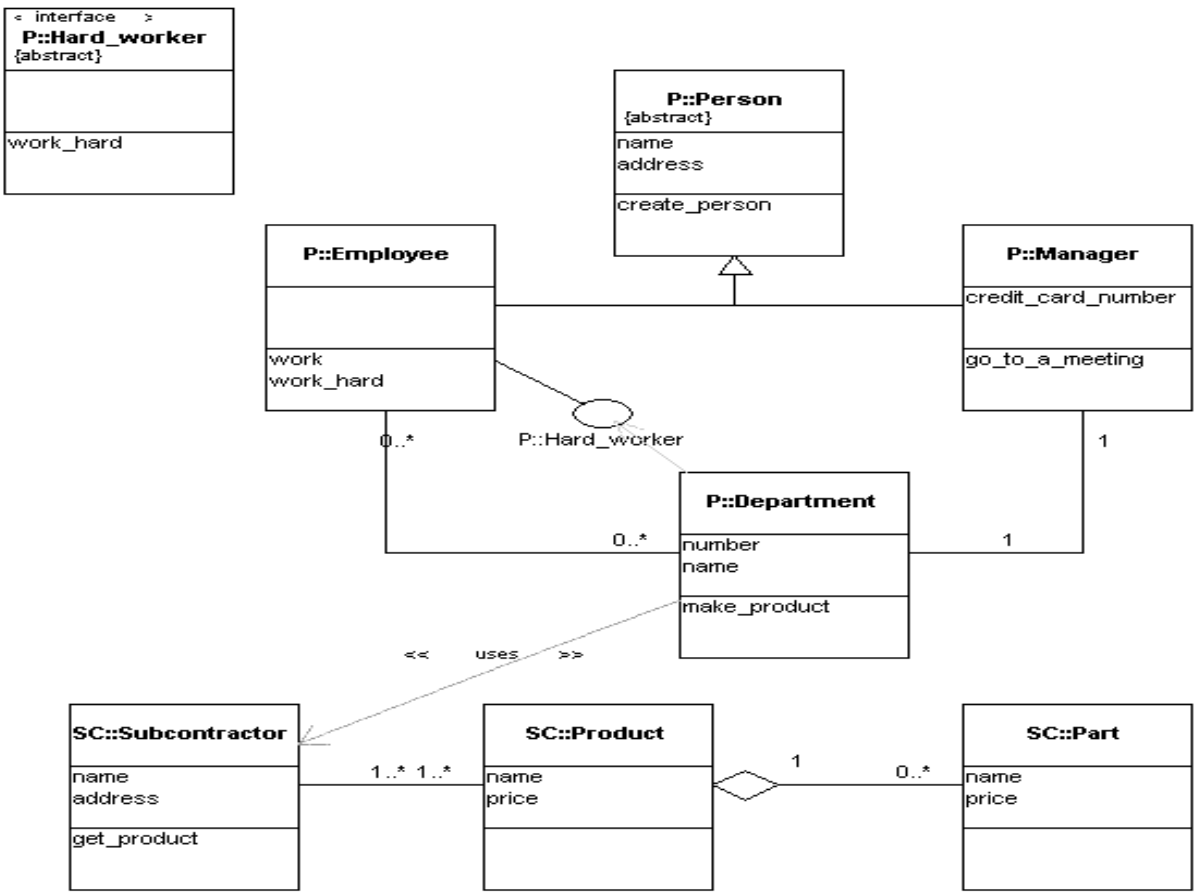

Figure 6: The unified class diagram.

4.3 Pattern visualization

The recognized patterns are visualized in the Patterns tab sheet. *Presently the visualization is available only for the metrics results calculated from single class diagrams.* Figure 7 presents the tabsheet showing some recognized patterns of the selected metrics result.

The list in the upper left part contains the recognized pattern instances, and the lower list contains the classes of the diagram in alphabetical order. On the right there is the so-called skeleton diagram.

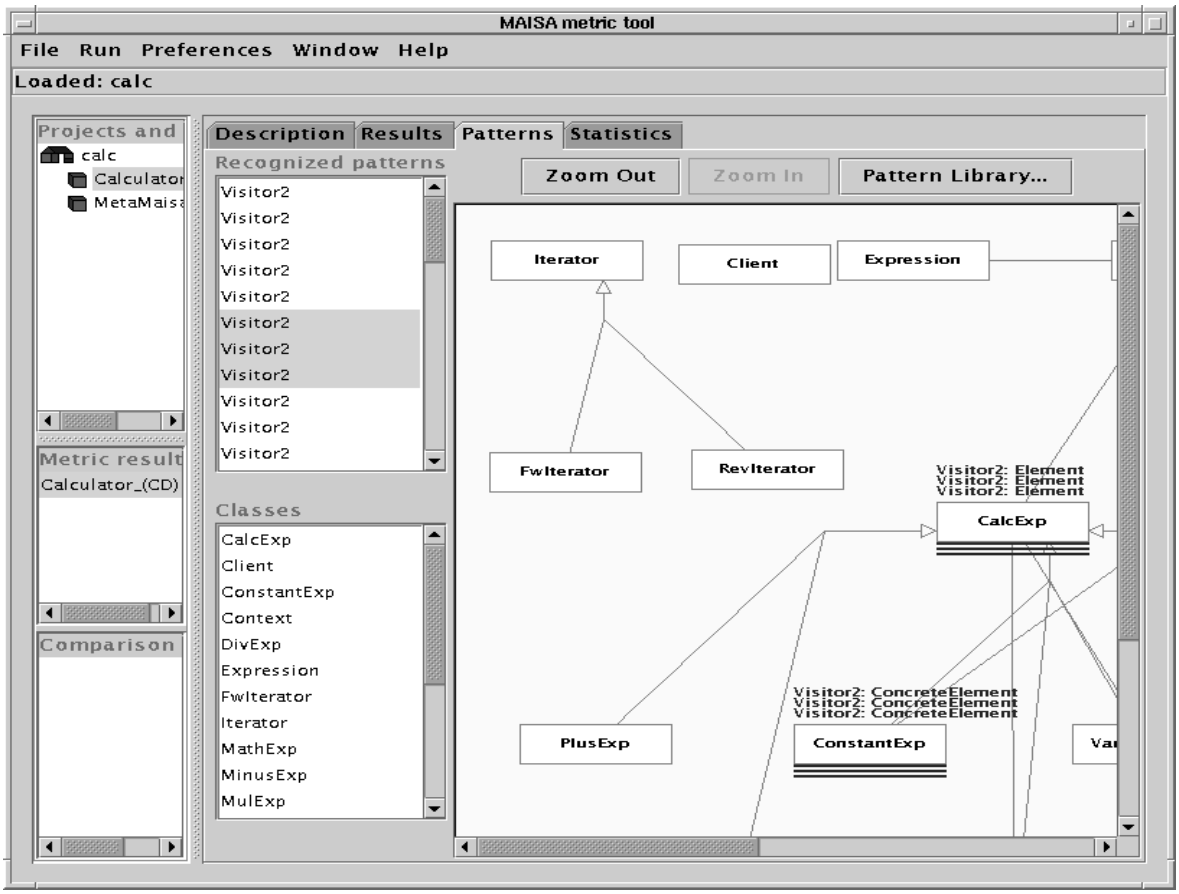

Figure 7: The main window showing recognized design patterns (zoomed in).

When an item of the recognized patterns list is selected, the pattern instance is emphasized in the skeleton diagram. Several instances can be emphasized at the same time. Separate colors are allocated to patterns and all the instances of a given pattern are emphasized with the same color. When an item of the class list is selected, all the patterns relating with the selected class are shown emphasized, different instances of patterns having separate colours.

Zoom in and Zoom out buttons are used for zooming the diagram skeleton. Zooming out simplifies the diagram by hiding such details as class or method names and pattern bindings

(roles of the classes in pattern instances). Zooming in brings out the details. Figure 7 presents the diagram zoomed in while Figure 8 shows the diagram zoomed out.

The Pattern library button opens the pattern library window, showing the pattern description of the selected instance.

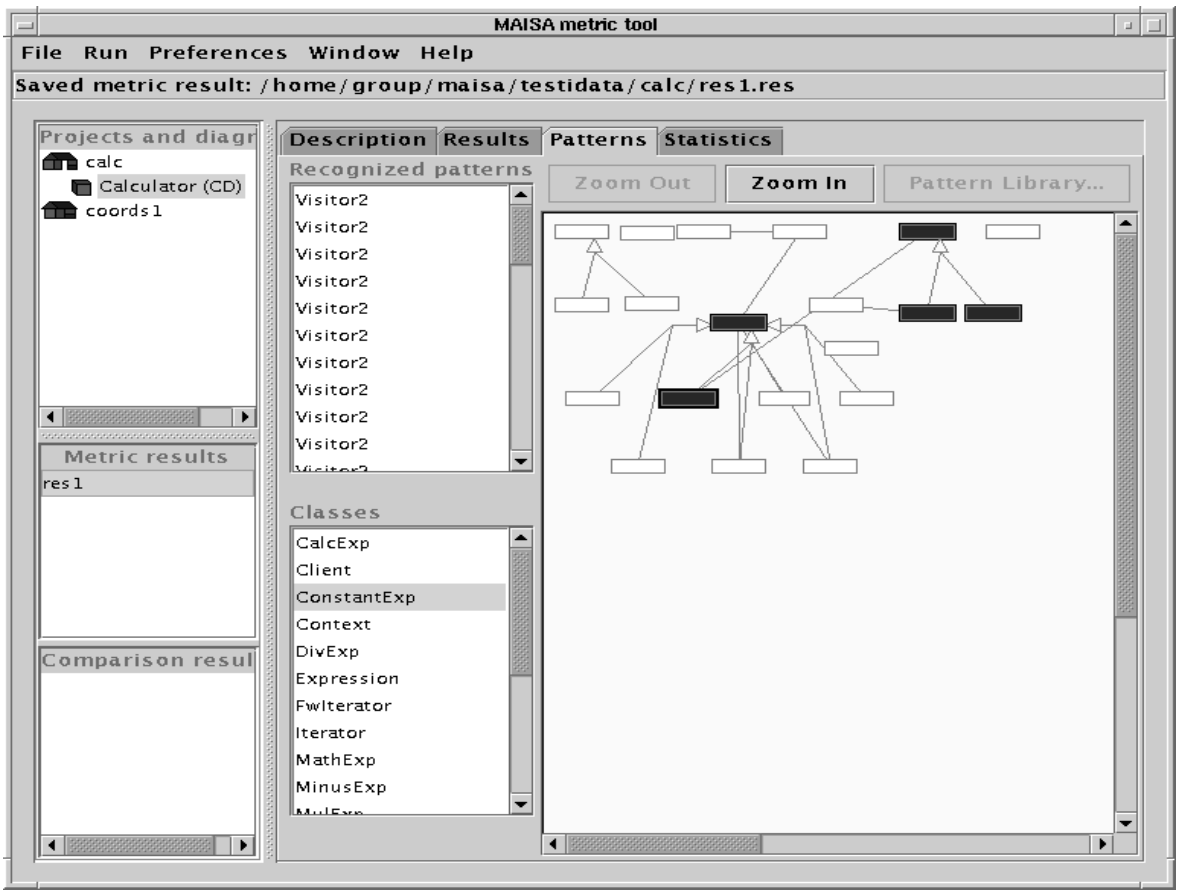

Figure 8: The main window showing recognized design patterns (zoomed out).

4.4 Statistics

In the Statistics tab sheet you can draw statistical diagrams and calculate key figures and statistical tests from the selected metrics result or comparison result.

Figure 9 presents the main window with the Statistics tab sheet. In a panel in the bottom of the sheet you define the diagram setup, or select the variable you want to examine. The upper left panel contains the key figures of the sample and the upper right panel contains the statistical diagram. The panels can be resized bu dragging their borders.

This is how you define the diagram setup. *First* select the diagram type. There is three types of diagrams:

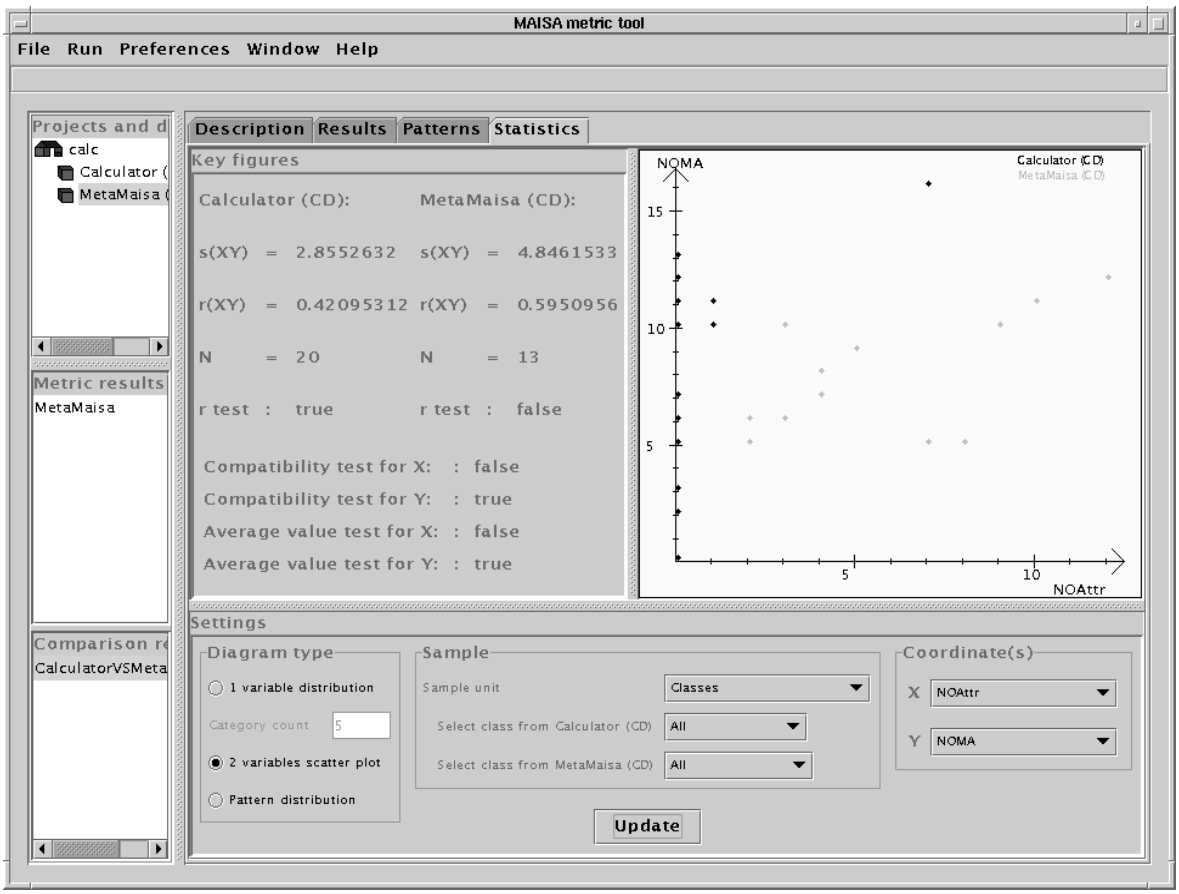

Figure 9: A two variable scatter plot drawn in the main window.

1 variable distribution Histogram from a classified sample. Remember to give the class count in the text field ("Category count").

2 variables scatter plot Scatter plot with one variable on the x axis and the other on the y axis.

Pattern distribution Histogram with one bar for the count of each pattern.

When handling a comparison result, the compared architectures are drawn in the same diagram, one in black and the other in gray. This makes it easy to visually compare the architectures. If there is both a black and a gray spot with the same coordinates in a scatter plot, the black spot is drawn bigger and the gray spot is drawn on the black one, so that it can be seen there are two overlapping spots.

Secondly, select the sample unit: per projects, per diagrams of certain type or per class. This affects the variables you can choose. You can also restrict the sample by selecting all or a certain project / diagram / class. If you are drawing a diagram from a comparison result, you can define the bounds for each two architectures, e.g. select one diagram from the first

architecture and another diagram from the second one.

Thirdly, choose the variable(s) to be examined. When drawing two variables scatter plot, choose both x and y variables, if you are drawing one variable histogram, choose only x (y is disabled) and if you are drawing pattern distribution histogram, you don't need to define either coordinate. If you selext metrics ICOUNT (Pattern instance count), you can also define the pattern whose instances are studied in the combo box that appears in the window. Note, that if you select "All" or study total pattern instance count, also the zero values are included in the diagram, but if you study ICOUNT of a certain pattern, the zero values are ignored.

Finally, push the Update button to draw the diagram. The key figures and the diagram appear on the left to the diagram. Note that they are not drawn until the button is pressed, not by merely changing the setup. For two first diagram types, the key figures and tests calculated by MAISA are:

- **Avg** Average value.
- **Mo** Mode.
- **E** Expected value.
- **D2** Variance.
- **s** Standard deviation.
- **Min** Minumum value.
- **Max** Maximum value.
- **N** Sample size.
- **Compatibility test** Calculated only when handling comparison results. A statistic test that tests if the samples are equally distributed with significance 0.05 (using Kolmogorov-Smirnov test).
- **Average value test** Calculated only when handling comparison results. A statistic test that tests if the average values are the same in significance 0.05 (using t test).

For scatter plots (or two dimensional distributions) the following correlation figures are also calculated:

- **s(XY)** Covariance.
- **r(XY)** Correlation coefficient.
- **R test** A statistic test that tests if r(XY) = 0, with significance 0.05.

The pattern distribution histogram is drawn from a nominal sample, which means that sensible dispersion values cannot be calculated. The key figures of the pattern distribution are:

- **Avg** Average value of category count.
- **MF** Name of the most frequent pattern (or mode).
- **LF** Name of the least frequent pattern.
- **Min** Minimum count value.
- **Max** Maximum count value.
- **N** Sample size.
- **CC** Category count.

4.5 Comparing

To measure a diagram or a project, select two architectures in the project list, two metrics results in the metrics result list or an architecture and a metrics result (comparison results cannot be compared). Then click "Compare" in the Run menu.

Inside the MAISA tool, comparison is performed by first measuring the possible selected architectures with the current metrics configuration, and then comparing the results. Only values of only the metrics common to both metrics results are compared. If the values are same they are written in blue in the result panel – otherwise they are written in black. Note, that comparing concerns only the ordinary metrics, not recognized design pattern instances and their role bindings.

4.6 Preferences

The user settings provided by MAISA are the following:

- **Confirm unsaved files when closing** Boolean. If true, the program asks for confirmation before closing a file that is not saved in the disk.
- **Confirm unsaved files on exit** Boolean. If true, the program asks for confirmation before exiting the program if files are not saved in the disk.
- **Confirm pattern / pattern group deletion** Boolean. If it is selected, the program asks for confirmation before deleting a pattern or a pattern group in the pattern library window.
- **Confirm pattern recognition** Boolean. If it is selected, the program asks for confirmation of the recognized pattern instances after running pattern recognition.
- **Remember work directory** Boolean. If it is selected, the file chooser dialogs remember in which directory you were last time.

- **Remember program state** Boolean. If it is selected, the program remembers its state between the sessions, ie when the program is started, it opens the windows that were open when the program was closed last time.
- **Pattern directory** String. The path of the pattern library directory.
- **Default metrics configuration** String. The path of the default metrics configuration file.
- **History data file** String. The file where the history data is saved.

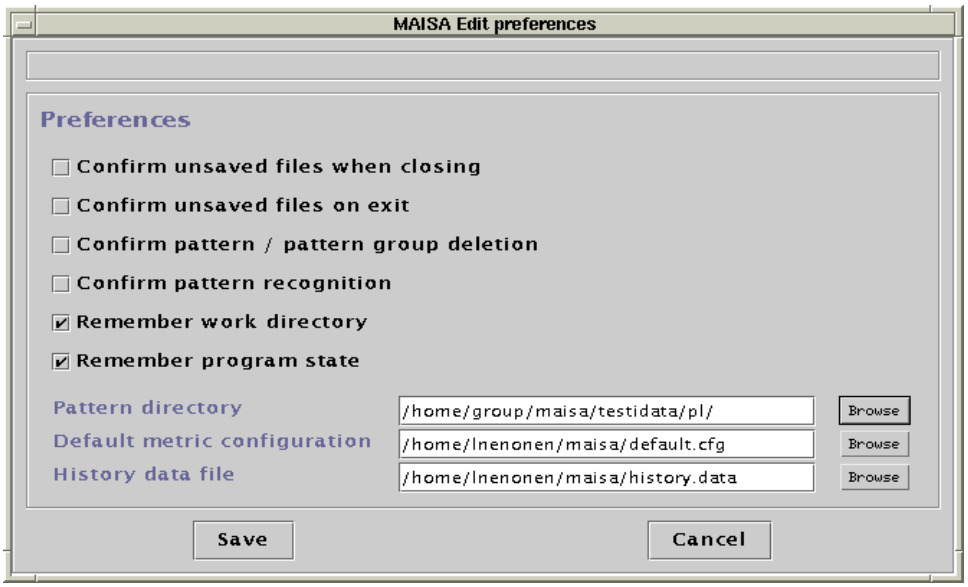

Figure 10: The preference dialog.

You can edit the preferences with the preference window which is opened by clicking the Edit item in the Preferences menu. The most frequently used boolean valued preferences, the first four ones in the list, can also be set directly in the Preference menu. Note that if "Remember program state" is selected, you lose the unsaved metrics result files when you quit the program (as you would do if "Remember program state" was not selected), even though the programs saves its state in every other part.

Figure 10 shows the preference window. On the bottom on the window there are two buttons: Cancel and Ok. The preference values are saved in the disk with pressing Ok and the setting modifications are cancelled with pressing Cancel. If a text field is left empty, no value is given to the preference and a default value is used. To select a directory or file, you can either write the file or directory name in the corresponding text field or select it in the file chooser dialog by pushing the corresponding Browse button.

You choose the file or directory names also in the the file chooser window that is opened by pressing the browse button right side of the test area you want to fill. You can also write the file or directory name in the text area yourself.

5 Metrics window

In the metrics window you can define the current metrics configuration. The default metrics configuration file is saved in your MAISA directory as `default.cfg`. Other configurations can also be used and edited in the metrics window. The metrics configuration contains up to three ordinary **measures**: Size, Performance and Understandability. Each measure consists of one or more **metrics**. Pattern Recognition is a special measure: on the other hand, some metrics belonging to other measures are based on pattern recognition, but on the other hand, there are some metrics connected to the pattern recognition. Thus, pattern recognition is a measure but at the same time it is an asset to another metrics, because the recognized patterns can be used as the basis of some metrics belonging to other measures.. In addition to the beforementioned measures, the configuration contains the recognized patterns and some metrics concerning them. *Note that the metrics that are not yet implemented are ahown disabled in the window. These kind of metrics are pattern based estimations and lines of code (LOC).*

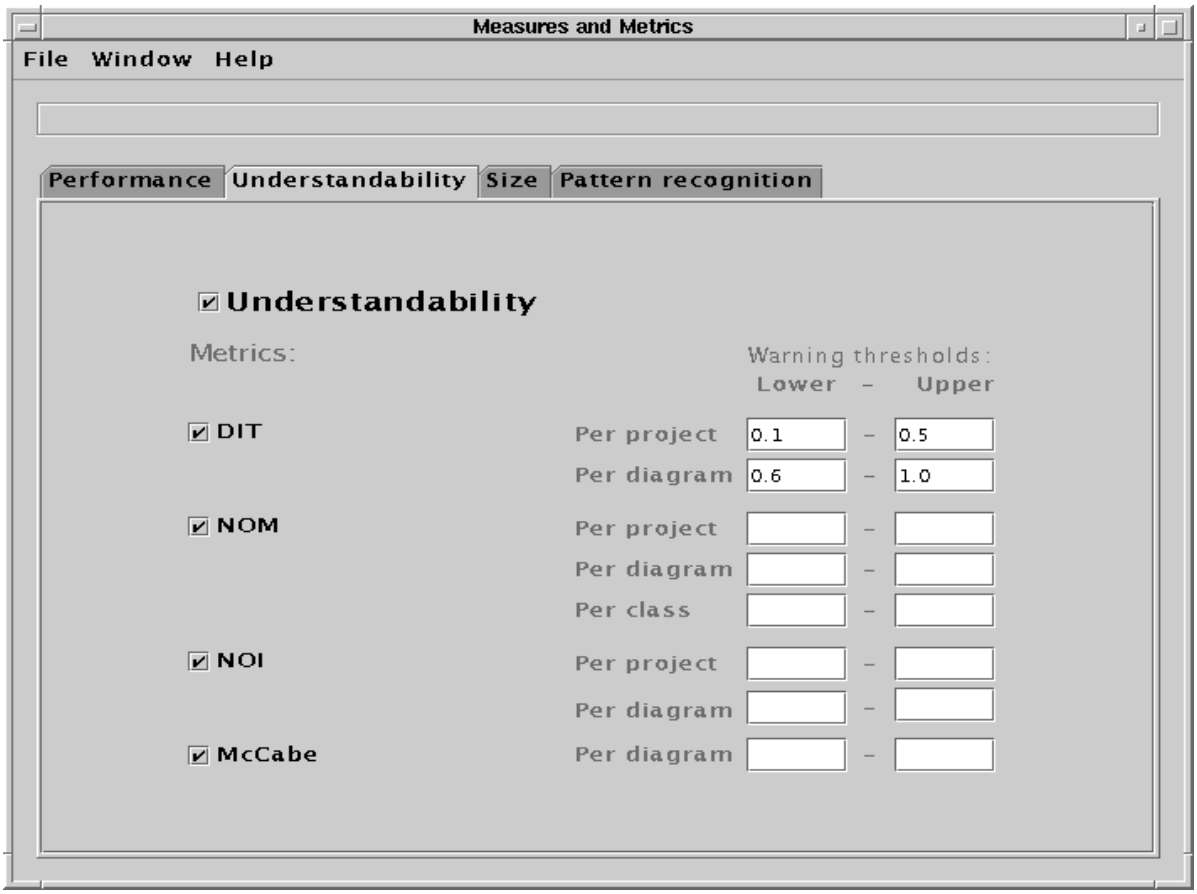

Figure 11: The metrics window.

Figure 11 presents the metrics window with an example metrics configuration. Each measure (and pattern recognition) is presented as a tab sheet. A tab sheet contains a checkbox for

each metrics belonging to the measure and text fields for upper and lower thresholds for each metric. Warning thresholds – minimun or maximum or both – can be given for each metrics in a certain relation. The user is informed if a calculated metrics value is not between the thresholds. The metrics belonging to a measure cannot be selected before the measure is selected. The metrics are given in certain relations, ie per diagram, per project and/or per a diagram element: per class, per object and so on. The metrics are selected to be calculated by checking the corresponding checkboxes, e.g. number of associations (NOA) per diagram in the metrics (tab sheet) "Size". Any number of checkboxes can be selected at a time. Pattern recognition metrics are shown in one of the tab sheets. Some metrics shown in the other tab sheets are based on pattern recognition. The recognized patterns (used by those metrics) and the metrics related to the pattern recognition are selected in the pattern recognition sheet.

The status bar is under the main menu. The messages and error messages given by MAISA are written in the status bar. The main menu contains three menus: File, Window and Help. The file menu contains the file operations concerning the metrics configuration files:

Open loads a new metrics configuration in the window.

Save saves the current metrics configuration in the disk with the current file name.

Save as... saves the current metrics configuration in the disk with a name given by the user.

Set as default saves the current metrics configuration as the default metrics configuration.

Reset to default loads the default metrics configuration.

Close window closes the window.

The Window menu contains commands to open the three other basic windows of the tool:

Main opens the main window.

Pattern library opens the pattern library window.

History data opens the history data window.

The Help menu contains the help command that opens the help window, specific to the metrics window.

Note, that certain metrics cannot be calculated from every kind of diagrams. e.g. if number of classes (NOC) is calculated from an activity diagram, the result is zero. In the following chapters we list the supported metrics and their diagram recommendations. The metrics algorithms are discussed in detail in our report [NeG2000].

5.1 Size metrics

Most of the metrics belonging to this category measure numbers of certain architectural elements, such as classes, associations and methods. The values of these metrics can be given in accurate numbers. Only one of the size metrics, lines of code (LOC) is an estimation. It is not yet implemented in version 1.3.

- **LOC** Lines Of Code. An estimated value. Given per project, diagram or class. *Not yet implemented.*
- **Pattern based LOC** As LOC, but the calculation based on the recognized patterns. *Not yet implemented.*
- **NOA** Number Of Associations (cf. NOAttr). Calculated from class diagrams. Given per project, diagram and class.
- **NOC** Number Of Classes. Calculated from class diagrams. Given per project and diagram.
- **NOMA** Number Of All Methods (cf. NOM, an understandability metric). Calculated from class diagrams. Given per project, diagram and class.
- **NOMsg** Number Of Messages. Calculated from sequence and collaboration diagrams. Given per diagram and object.
- **NOOb** Number Of Objects. Calculated from sequence and collaboration diagrams. Given per diagram.
- **NOOp** Number Of Operations. Calculated from state diagrams. Given per diagram.
- **NOS** Number Of States. Calculated from state diagrams. Given per diagram.
- **NOT** Number Of Transitions. Calculated from state diagrams. Given per diagram and state.
- **NOAttr** Number Of Attributes (cf. NOA). Calculated from class diagrams. Given per project, diagram and class.

5.2 Performance metrics

The unit of these metrics is the one in the diagrams from which the metrics are calculated. The pattern based and memory estimations are not yet implemented in version 1.3.

- **Pattern based estimation** performance estimation, based on patterns. *Not yet implemented.*
- **Time requirement** Calculated from extended activity diagrams. Given as best, worst or typical case.

- **Memory requirement** Calculated from extended activity diagrams. Given as best, worst or typical case. *Not yet implemented.*

The extended activity diagrams contain extra information used in performance estimation by MAISA. Note, that the typical case is not (necessary) the average case. The activity diagram extension is discussed in our report [NeG2000].

5.3 Understandability metrics

Understandability is not a clear concept to define. The following metrics are widely accepted as predictors of understandability of the source code and the architecture.

- **Design patterns** The recognized patterns that are printed to the user. Every pattern instance contain information on which architectural elements (classes, objects) have roles in the pattern. These bindings are given as role-element pairs.
- **DIT** Depth of Inheritance Tree. Calculated from class diagrams. Given per project and diagram.
- **NOM** Number Of public Methods (cf. NOMA in the measure Size). Calculated from class diagrams. Given per project, diagram and class.
- **NOI** Number Of Interfaces. Calculated from class diagrams. Given per project and diagram.
- **McCabe** The McCabe cyclomatic complexity measure. Calculated from state diagrams and activity diagrams. Given per project and diagram.

5.4 Pattern recognition

The metrics belonging to this category indicate pattern coverage, i.e. they measure the propotion of architectural elements that are bound to the (recognized) design patterns. A corresponding size metrics can be found to these relative metrics. Also two non-relative metrics belong to this category.

- **RNOA** Relative Number Of Associations. Calculated from class diagrams. Given per project, diagram and class.
- **RNOC** Relative Number Of Classes. Calculated from class diagrams. Given per project and diagram.
- **RNOOb** Relative Number Of Objects. Calculated from sequence and collaboration diagrams. Given per project and diagram.

- **RNOMA** Relative Number of All Methods. Calculated from class diagrams. Given per project, diagram and class.
- **RNOMsg** Relative Number Of Messages. Calculated from sequence and collaboration diagrams. Given per diagram and object.
- **RNOAttr** Relative Number Of Attributes. Calculated from class diagrams. Given per project, diagram and class.
- **PCOUNT** Pattern count. Calculated from class diagrams. Given per project, diagram and class.
- **ICOUNT** Pattern instance count. Calculated from class diagrams. Given per project, diagram and class.

6 Pattern window

The pattern window provides the user interface to the pattern library, partly outside of the metrics system. The patterns library contain textual descriptions and user instructions for each pattern, which can be browsed in the pattern window and which are intended to be used by human users of the library, not the metrics algorithms. The pattern library also contains formal descriptions of the patterns, written in Prolog language and an index file common to the whole library. These files are used by the pattern recognition algorithm and unless add your own patterns in the library, you do not need to handle these formal descriptions at all. The patterns are grouped into pattern groups by their nature. The patterns can be updated, removed and added to the library and the groups can be added and removed. Initially the pattern library contains the design patterns presented in the Design patterns book and the so called (Pree's) meta patterns. See appendix A to see the full list of these patterns with their brief descriptions.

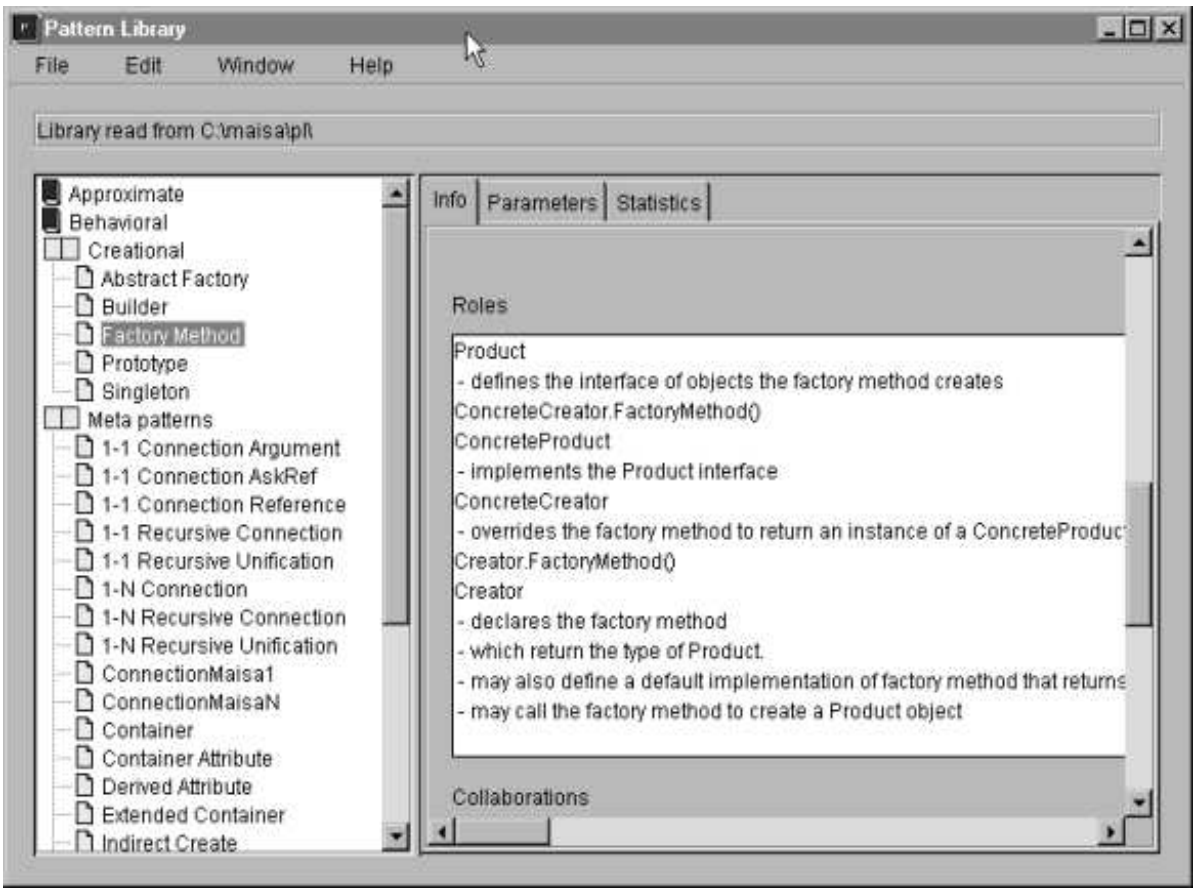

Figure 12: The pattern library window.

Figure 12 presents the pattern library window. The window works like a HTML frame. On the left is a tree-shaped list of patterns and pattern groups. The data of the selected pattern

is shown on the right part of the window, which is divided into three tab sheets. The border between the tab sheets and the pattern list can be dragged. The Info sheet contains the textual pattern description. The Parameters sheet contains the metrics parameters of the pattern (they are not implemented in version 1.3) and the Statistics sheet contains the statistical values of the pattern: how many times it has been recognized and how many times the recognition has been confirmed or wrong. The pattern description part follows the documentation model used in the Design Patterns book [Gam1995]. The design pattern documentation is discussed in detail in Chapter 6.1.

Most of the operations in the main menu are for administrative use of the pattern library. If you just use the existing patterns in the directory, without updating the library, you have little use to the main menu. The file menu contains operations that do not modify the structure of the pattern library:

Copy copies the selected pattern of pattern group to another directory selected by the user. Note that only directories can be selected.

Close window closes the pattern library window.

The edit menu contains operations that modify the pattern library in the disk:

Edit mode Sets edit mode on / off.

Save Saves the current pattern in the disk.

Import: Pattern Copies a new pattern to the selected pattern group.

Import: Pattern group Copies a new pattern group to the library.

New pattern Creates a new pattern to the selected pattern group with a name provided by the user.

New pattern group Creates a new pattern group to the library with a name provided by the user.

Delete Deletes the selected pattern group or pattern from the library.

Setting the preference "Confirm pattern / pattern group deletion" in the main window prevents you from deleting a pattern group or a pattern by accident. To avoid updating pattern information by accident, the window must set into edit mode before updating the pattern description text. Figure 13 presents the pattern library window in edit mode. In edit mode the text areas become editable, the figure file and the prolog file can be set to the pattern, the index file of the pattern library appear in the window and you can reset the statistic information by pressing the Reset Statistics button that appear on the right side of the statistic values.

To administrate the pattern library, you should be familiar with the structure of the physical pattern library. The pattern library in the disk is a directory (it can be set as a preference,

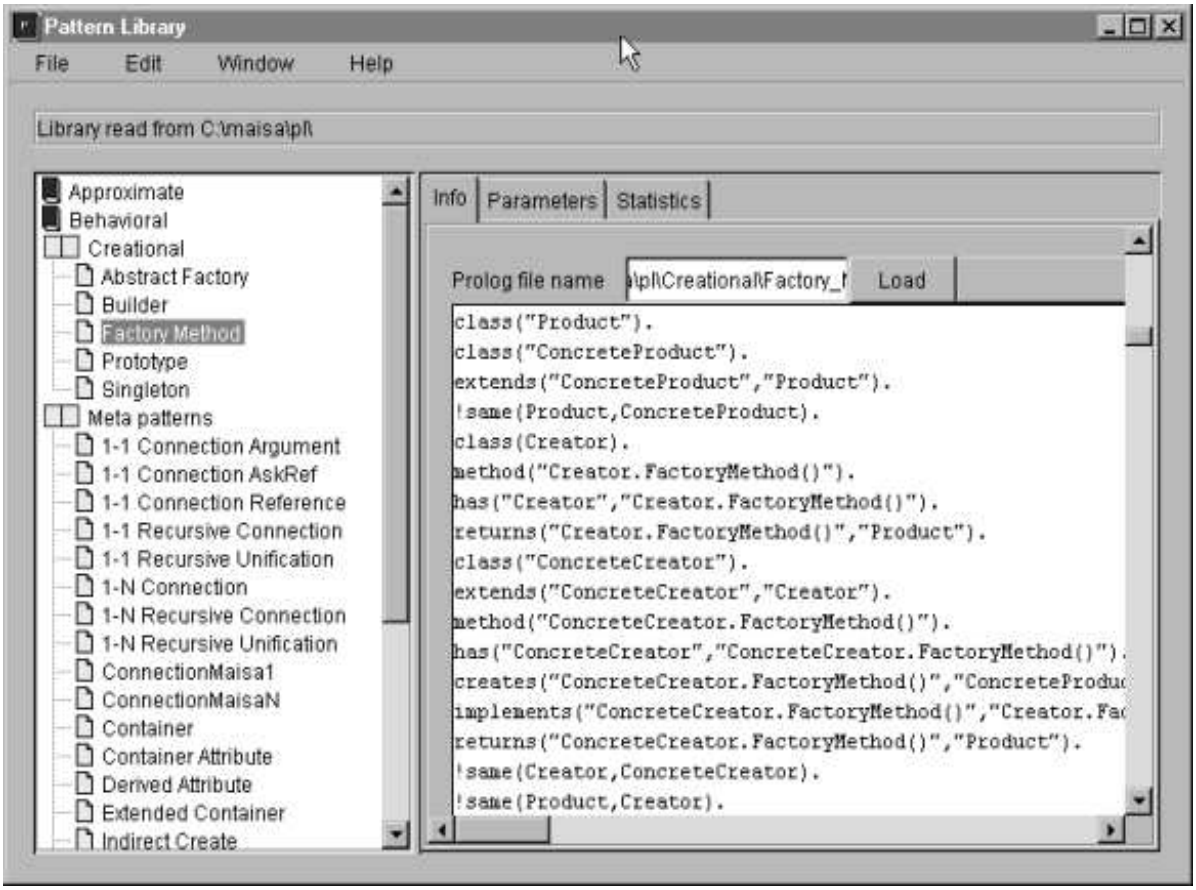

Figure 13: The pattern window in edit mode.

otherwise a default value for the directory path is used). The pattern library directory consists of pattern group directories and the pattern group directories consist of pattern directories. A pattern directory contains all the necessary files describing the pattern. When importing a new pattern (group) to the library you select the corresponding pattern (group) directory in the file chooser dialog. The tool copies all the necessary files in the directory. The name of a pattern (group) directory is the name of the pattern (group), with the space charecters replaced with underline characters.

The Window menu contains commands to open the three other basic windows of the tool:

Main opens the main window.

Measures and metrics opens the metrics window.

History data opens the history data window.

The Help menu contains the help command that opens the help window.

6.1 Pattern description

The textual descriptions of the patterns are divided into sections, following the documentation model presented in the Design Patterns book [Gam1995]. The text in the chapters is given in a certain formula. The following list describes the content of the sections and the formula of the text areas of the pattern window. Every section is optional and can be left empty.

- **INTENT** The meaning of the pattern – a brief description of what it does. Any text except empty rows.
- **ALSO KNOWN AS** A comma-separated list of other names of the pattern, if there is any. For example:

  ```
  Name, Another name, Yet another name, The last name
  ```

- **MOTIVATION** A description of what design problem the pattern solves. Perhaps an example of the problem. Any text except empty rows.
- **APPLICABILITY** A list of points where the pattern can be used. The first row of each item begins with a hyphen. For example:

  ```
  - You can use the pattern if...
  Especially if ...
  In a situation...
  - Another place to use the pattern is...
  - The pattern is extremely efficient when...
  Btw, there is...
  ```

- **PARTICIPANTS** A list of the roles of the pattern. The role names are written one name per a row. A role name row can be followed with a list of the tasks belonging to the role, each task beginning with a hyphen. For example:

  ```
  Role A
  - Task A1
  - Task A2
  Role B
  Role C
  - Task C1
  ```

- **COLLABORATIONS** A list of points where the elements of the pattern co-operate. The first row of each item begins with a hyphen.
- **CONSEQUENCES** A list of consequences of using the pattern. The first row of each item begins with a hyphen.
- **IMPLEMENTATION** Implementation specific tips on using the pattern. Any text except empty rows.

- **SAMPLE CODE** Sample code fragments, with possibly some ordinary text. Any text except empty rows.
- **KNOWN USES** Examples of the pattern found in real systems. Any text except empty rows.
- **RELATED PATTERNS** Pattern's relations to other patterns. Which other patterns should be used with this one. Any text except empty rows.

6.2 Edit mode

When the pattern window is set to edit mode, the text areas and text fields and the pattern name header become editable. When updating the pattern description text, remember to save the pattern. The modifications are not written to the disk automatically and they are cancelled if you select another pattern without saving the modifications on the former one.

The textual descriptions of the pattern are written in the text areas as described in the previous chapter. In addition, you can give a figure file in gif format, describing the structure of the pattern and you must give the prolog file, describing the architectural constraints that a pattern instance must fulfil. The pattern library also has an index file that is common to all patterns and describes the order in which the patterns are recognized. You can set the figure files and set and edit the two textual files in edit mode. There are two buttons and a text field for each of the three files. The directory path of the file is written in the text field or selected by browsing the file system by pressing the Browse button. After choosing the directory, you can load the file in the text area (or load the figure in the window) by pressing the load button. The textual files can be edited in the text areas. If you load and save an external file e.g. the prolog file of the pattern, the external file is copied in the pattern directory. Note that the text in the text areas are written to the prolog and index files. So you must load the text in the window if you want to e.g. load the prolog code from an external file - it is not enough to just write the directory path in the text area.

6.2.1 Prolog file

The heart of the pattern library is a collection of Prolog files (one for each pattern). These files are named **constraints.pro** and they are located in the directories of their respective patterns. Each file contains the facts that define the particular pattern. They can be edited while in the Edit mode of the Pattern library window or with any text editor.

The facts used in the Prolog-files are as follows[1]:

[1]Note that the role names must be in quotes.

Fact	Description
class(X)	X is a class
method(X)	X is a method
attribute(X)	X is an attribute
public(X)	X is public
protected(X)	X is protected
private(X)	X is private
abstract(X)	X is abstract
final(X)	X cannot be redefined
static(X)	X is a class member
has(C,X)	class C has a member X
extends(X,Y)	Class X is a subclass of Y
typeof(A,C)	Attribute A is of type (class) C
contains(A,C)	Attribute A contains reference(s) to class C
implements(X,Y)	Method X implements the (virtual) method Y
returns(M,C)	Method M returns (a reference to the) class C
argument(M,C)	Method M gets (a reference to the) class C as a parameter
reads(M,A)	Method M reads the attribute A
calls(X,Y)	Method X calls method Y
creates(M,C)	Method M creates an instance of class C
same(X,Y)	Components X and Y are the same
binded(X)	Role is binded to a particular component X

Negation is represented with the !-symbol. It can be used with facts: !public("X"). or components: !"X".

Example: The Prolog description of the *Singleton* pattern is

```
class("Singleton").
attribute("Singleton.instance").
has("Singleton","Singleton.instance").
typeof("Singleton.instance","Singleton").
static("Singleton.instance").
```

6.2.2 Index file

The index file, located at the root directory of the pattern library, is named **common.txt**. It is used to locate the Prolog files of the patterns and to guide the search process.

The index file contains an entry for each pattern in the library. Each entry has four main parts:

- Pattern name: The line begins with the character #, followed by the name of the pattern.
- Constraint file: The line begins with *file:* followed by the path name of the constraint

file (relative to the root directory of the library).

- Pattern requirements: The names of those patterns that are prerequisites to this one. This line begins with the keyword *requires:*.
- Generated facts: The descriptions of the facts that are produced as a by-product while searching for this pattern. These will be used when more patterns are searched. The keyword is *generates:* followed by the description of the generated facts (one per line). The names of the roles as well as that of the fact must be enclosed in quotes. For example, the line: "myfact"("Role"), would generate the fact: myfact(<binding>), where <binding> is the binding *Role* had when this directive was triggered.

Example: The *Indirect Write* entry in the index file

```
#Indirect Write
file: Meta_patterns/Indirect_Write/constraints.pro
requires: Derived Attribute
generates:
        "writes"("First","Attribute").
        "indirectwrite"("First","Attribute").
```

7 History data window

In the history data window you can draw statistical diagrams and calculate key figures and statistical tests from the history data that you have collected while using the MAISA tool. The window works like the Statistic tab sheet in the main window, but uses the history data instead of metrics results and offers one extra diagram type to draw. The selected metrics results are added in the history data archive by selecting "Save metrics to history data" in the file menu of the main window. The whole history data is emptied by selecting the "Clear history data". In the history data window it is possible to compare two projects / diagrams / classes whose metrics results are saved to the history data.

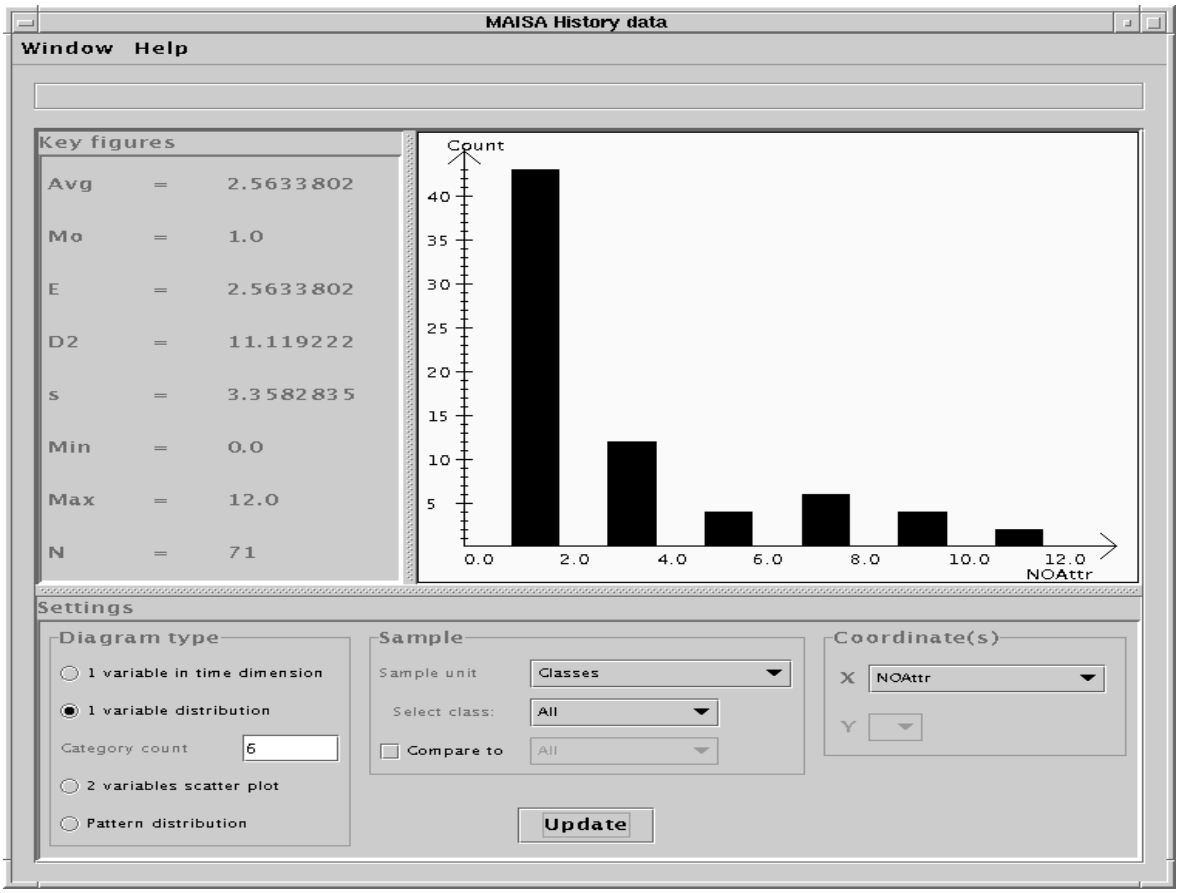

Figure 14: A histogram drawn in the history data window.

Figure 14 presents the history data window. In a panel in the bottom of the window you can do the diagram setup, or select the variable you want to examine and so on. The upper left panel is for key figures of the sample and the upper right pannel is for a diagram itself. You can drag the borders between the panels according to taste. Initially, the diagram and key figure panels are empty.

There is two menu in the menubar: Window and Help. The help menu has only one item,

"Help..." and it opens the help window. The window menu contains items to open the other basic windows:

Measures and metrics opens the metrics window

Pattern library opens the pattern window

Main window opens the main window

Close closes the history data window

Let's see how you define the setup. *First* select the diagram type. There is four typess of diagrams:

1 variable in time dimension Scatter plot with time on the x axis and variable values on the y axis.

1 variable distribution Histogram from a classified sample. Remember to give the class count in the text field.

2 variables scatter plot Scatter plot with one variable in x axis and the other in y axis.

Pattern distribution Histogram with one bar for the count of each pattern.

Secondly, you select the sample unit: per projects, per diagrams of certain type or per class. This affects the variables you can choose. You can also restrict the sample by selecting all or a certain project / diagram / class. If you want to compare the selected project / diagram / class to another one, you click the "Compare to" checkbox so that you can define the sample (i.e. select project / diagram / class) where to compare.

If you do compare, the two samples are drawn in the same diagram, one in black colour and the other in gray. This makes it easy to visually compare the architectures. If there is both a black and a gray spot with the same coordinates in a scatter plot, the black spot is drawn bigger and the gray spot is drawn on the black one, so that it can be seen there are two overlapping spots.

So *third*, you choose the variable(s) to be examined. When drawing two variables scatter plot, choose both x and y variables, if one variable histogram, choose only x and if one variable in time dimension, choose y (the other variables are disabled). If you are drawing pattern distribution histogram, you don't need to define either coordinate. If you selext metrics ICOUNT (Pattern instance count) as a variable, you can also define the pattern whose instances are studied in the combo box that appears in the window. Note, that if you select "All" or study total pattern instance count, also the zero values are included in the diagram, but if you study ICOUNT of a certain pattern, the zero values are ignored.

Finally, push the you draw Update button to draw the diagram. The key figures and the diagram appears on the left. Note that they are not updated before pressing the button, not by merely changing the setup. When drawing three first diagram types, the key figures and tests calculated by MAISA are:

- **Avg** Average value.
- **Mo** Mode.
- **E** Expected value.
- **D2** Variance.
- **s** Standard deviation.
- **Min** Minumum value.
- **Max** Maximum value.
- **N** Sample size.
- **Compatibility test** Calculated only when comparing. A statistic test that tests if the samples are equally distributed with significance 0.05 (using Kolmogorov-Smirnov test).
- **Average value test** Calculated only when comparing. A statistic test that tests if the average values are the same with significance 0.05 (using t test).

For scatter plots (or two dimensional samples) the following key figures and tests are also calculated.

- **s(XY)** Covariance.
- **r(XY)** Correlation coefficient.
- **R test** A statistic test that tests if r(XY) = 0, in significance 0.05.

The pattern distribution histogram is drawn from a nominal sample, which means that sensible dispersion values cannot be calculated. The key figures of the pattern distribution are:

- **Avg** Average value of category count.
- **MF** Name of the most frequent pattern (or mode).
- **LF** Name of the least frequent pattern.
- **Min** Minimum count value.
- **Max** Maximum count value.
- **N** Sample size.
- **CC** Category count.

8 Help window

The help window shows this user manual and it can be opened from the four basic windows of the tool (main, metric, pattern library and history data). It works like a HTML frame. On the left side of the window there is a tree-shaped list of the chapters of the manual. The selected chapter is shown on the left, with the text on the top and the figures on the bottom. The text and the figures are shown in separate, scrollable panels. You can drag the borders between the chapter list and between the text and figure panels according to your taste. Note, that – unlike in this manual – the figures are numbered according to the chapters. If you select a main chapter in the list, also the text and figures are shown on the right. If you select a subchapter, only the text and figures of the subchapter are shown.

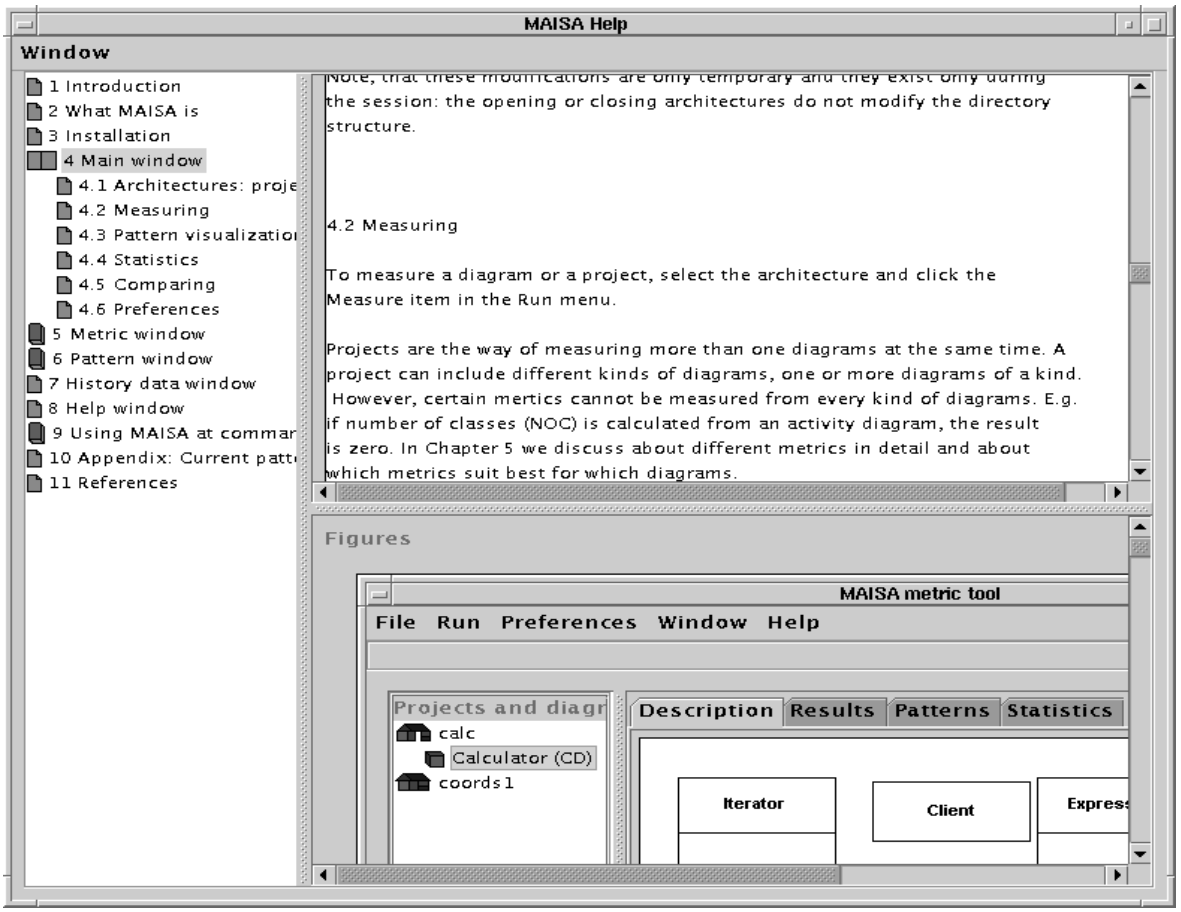

Figure 15: The help window.

Figure 15 presents the help window. There is only one menu in the menubar: Window. The window menu contains items to open the four basic windows:

Main window... opens the main window

Measures and metrics... opens the metrics window

Pattern library... opens the pattern window

History data... opens the history data window

Close closes the help window

9 Using MAISA in command-line

The command-line interface of the MAISA tool can be used with or without command-line options. Naturally, the graphical features of the MAISA tool, such as the visualization of patterns, cannot be used. Thus, this feature serves best as a quick testing tool or to provide an overview of the architecture.

Without options, the program is started simply with the command[2]:

```
java MAISA.datastruct.MaisaStructure
```

The program asks the user for the project to be measured. Presently, all simple (i.e. not pattern based) metrics are calculated for each diagram in the selected project. The result is written (into the project directory in question) into the file *result.txt*. See [Gus2000] for an example of the result file.

Next the program asks if the user wishes to continue. By answering *Yes*, the user can proceed to a new measurement, with a *No*, the user will exit the program.

By using command-line options, the user may take advantage of the pattern searching features of the MAISA tool as well as get the result e.g. in HTML format. The project directory, metric configuration and output files (i.e. the d, c and o options) must be specified. The pattern library can be omitted if no pattern based metrics are needed. For a more detailed description of the pattern library, see Chapters 6.2.1 and 6.2.2. Setting the metric configuration and different metrics are discussed in Chapter 5.

The program is started with the command:

```
java MAISA.datastruct.MaisaStructure <options>
```

The following options are available:

`-d <project directory>`
the directory can contain one or more diagram files

`-p <pattern library>`
the root directory of the pattern library

`-c <configuration file>`
the file must be a MAISA metric configuration file

`-o <output file>`
the path of the desired output file

One of the following may also be used:

[2]Naturally, the classpath needs to be correctly set.

`-m`
the result is presented in MAISA result format
`-u`
the result is presented in Unix format
`-h`
the result is presented in HTML format (default)

9.1 Pattern library directory

The pattern library can be updated in the pattern window of the MAISA tool or with the common file handling tools of the operating system (the former way is more user friendly to novice users, while the latter is quicker and more efficient). Patterns or pattern groups can be added, edited and removed from the library. These operations are intended for administrative use. A pattern is identified with pattern name and group name. There can be patterns with the same name belonging to different pattern groups, for example versions of the same pattern.

The physical pattern library is a directory, located in `maisa/pl`. The root directory contains one subdirectory for each pattern group. The group name is the name of the subdirectory, with underline characters replaced with spaces. Each group directory contains one subdirectory for each pattern belonging to the group. The name of the pattern directory is the name of the pattern, with spaces replaced with underline characters. The group names and the pattern names do not contain any underline charecters and the directory names do not contain any spaces, in order that the directory structure works in the environments where file names with spaces are not supported. The root directory also contains one prolog file,**common.txt**, which is common to all the patterns and which is called. It defines the order in which the patterns are recognized, as explained in the previous chapter. The user can edit the file with the MAISA tool but it is not generated by the tool. Each pattern directory contains five files used by the MAISA tool: prolog, description, statistics, metadata and figure file (naturally it can also contain other files connected to the pattern, for example sample source code and additional figures and diagrams).

Figure 16 shows the structure of the pattern library directory. Note that there is one group directory also for the metapatterns [Pre1995], because an instance of a metapattern can be constructed as more than one design patterns presented by Gamma et. al. [Gam1995], depending on the semantics. Metapatterns themselves can also be recognized, without mapping them into design patterns. The following list explains how the pattern files are used.

constraints.pro The Prolog file that contains the structural requirements (or definition) of the pattern. The pattern recognition is based on these files. The file is not generated by the MAISA tool but the user can edit or set it with the tool.

pattern.description Text file that contains the same information of the pattern as the Design Patterns book (see pages 39-41) [Gam1995]. The information of the file is for the human user. The file is generated by the MAISA tool based on the data provided by the user.

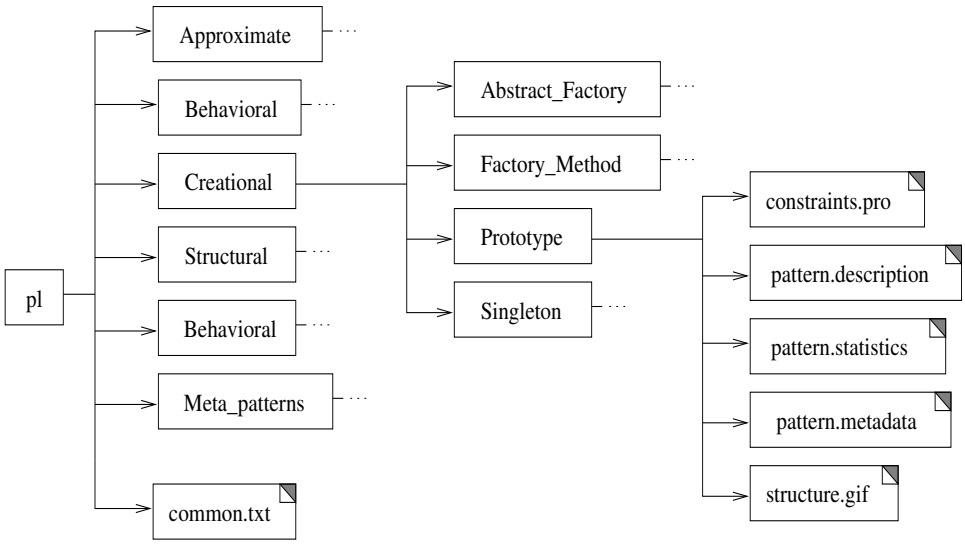

Figure 16: Structure of the pattern library directory.

pattern.statistics Text file that contains statistical information of the pattern: how many times it has been recognized, how many of these have been confirmed and how many of these have been false recognitions. The statistics file is automatically generated and updated by the MAISA tool and the user can reset the statistical information.

pattern.metadata Text file that contains metadata of the pattern that is used by pattern based estimation algorithms: LOC per pattern instance, LOC per client of the pattern and the performance factor of the pattern. The metadata file is generated by the MAISA tool based on the data provided by the user.

structure.gif Figure file that describes the structure of the pattern. Optional. The file can not be edited or generated with the MAISA tool, but it can be set with the tool.

The formats of the text files are simple. The statistics and metadata files contain name-value pairs, one pair in a row. The name and the value are separated by a colon. The values are typically numeric, while the names are special keywords. The keyword sets can be easily extended in the future if needed. The keywords of the statistics file are:

- **WRONG** false recognitions
- **CONFIRMED** confirmed recognitions
- **TOTAL** all recognitions

A sample statistics file:

```
WRONG: 5
CONFIRMED: 12
TOTAL: 50
```

Metadata files are not yet in use, because they are only used by the pattern based estimation algorithms, which are not yet implemented. The exact types of the values are not yet fixed. The keywords of the metadata files are (the set can be extended in the future):

- **PERFORMANCE** performance factor
- **LOCPI** LOC per a pattern instance
- **LOCPC** LOC per a pattern client

The description file is divided into paragraphs, according to the documentation technique presented in the Design Patterns book [Gam1995]. The paragraphs are separated by an empty row and they can be in any order. The first line of the paragraph contains the mere paragraph name. The paragraphs are identified by their names, which are keywords. The structure of the paragraph depends on its name, as described in the following:

- **INTENT** What the pattern does. Any text except empty rows (the paragraph separator).
- **ALSO KNOWN AS** A comma-separated list of other names of the pattern.
- **MOTIVATION** What design problem the pattern solves. Any text except empty rows (the paragraph separator).
- **APPLICABILITY** A list of points where the pattern can be used. The first row of each list item begins with an asterisk.
- **PARTICIPANTS** A recursive list of the roles of the pattern. Each list item presents a role and contains a list of the tasks belonging to the role. The first row of a role item begins with an asterisk that is followed by the role name. The following rows of the role item begin with "$>$" that is followed by a task description.
- **COLLABORATIONS** A list of points where the elements of the pattern co-operate. The first row of each list item begins with an asterisk.
- **CONSEQUENCES** A list of consequences of using the pattern. The first row of each list item begins with an asterisk.
- **IMPLEMENTATION** Implementation specific tips on using the pattern. Any text except empty rows (the paragraph separator).

- **SAMPLE CODE** Sample code fragments, with possibly some ordinary text. Any text except empty rows (the paragraph separator).
- **KNOWN USES** Examples of the pattern found in real systems. Any text except empty rows (the paragraph separator).
- **RELATED PATTERNS** Pattern's relations to other patterns. Which other patterns should be used with this one. Any text except empty rows (the paragraph separator).

A sample description file:

```
INTENT
The pattern is used to...

ALSO KNOWN AS
Name 1, Name 2, Name 3

MOTIVATION
The pattern is useful, because...

APPLICABILITY
* Point 1
* Point 2
* Point 3

PARTICIPANTS
* Role A
> task 1
> task 2
* Role B
> task 3
> task 4
> task 5

COLLABORATIONS
* Point A
* Point B

CONSEQUENCES
* Point X
* Point Y
* Point Z

IMPLEMENTATION
The pattern can be implemented in this way...

SAMPLE CODE
Class MyClass{
...
}

KNOWN USES
The pattern has been used in...

RELATED PATTERNS
The pattern can be used with pattern X and...
```

9.2 Help directory

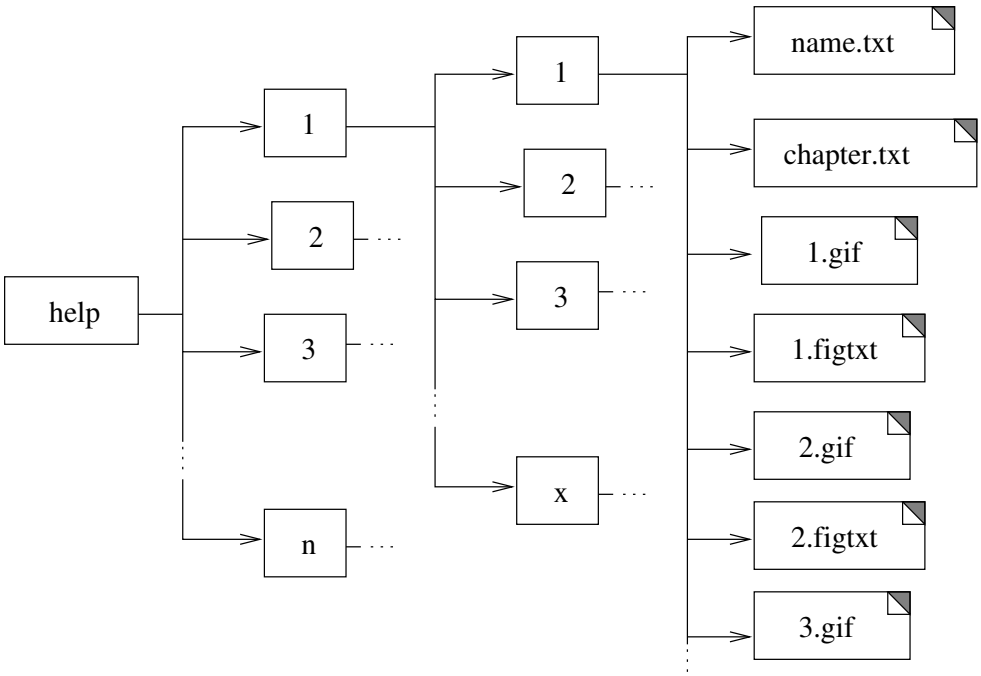

Figure 17: Structure of the help directory.

As the pattern library, also the online manual is a directory. It is located in `maisa/help` and it consists of directories corresponding the chapters of the manual. The name of a chapter directory is the number of the chapter. Figure 17 shows the structure of the help directory. As shown in the figure, a chapter directory contains the following files:

name.txt Contains the name of the chapter – only one row.

chapter.txt The chapter itself, or the text of the chapter – plain text.

x.gif The figure files, where x is an integer, presenting the figure number.

x.figtxt The figure text files, where x is an integer, presenting the figure number.

As in the graphical user interface, the numbering of the figures is specific to the chapter, not to the whole manual. The figure numbering begins from 1. Also the names of the chapter directories are integers. There are no subdirectories for the subchapters.

A Current patterns in the library

The search for patterns in MAISA is hierarchical. Each pattern may have one or more prerequisite patterns and for each pattern to be searched, these prerequisite patterns are searched first.
As the metapatterns given below are very general (and common), they take more time to find than the more specific patterns. Because the more specific patterns are defined using these metapatterns, searching for a small number of specific patterns can still lead to an extensive search. More detailed description of the following patterns can be found in [Gam1995, Pre1995]. As the design level diagrams generally contain much less information than is needed for exact identification of many design patterns, we are implementing some patterns in the library using somewhat relaxed rules. These can be found under the *Approximate* pattern group. They are named using a number after the name of the original pattern (e.g. *Factory Method2*). The only difference between the two versions of a pattern is that the approximate one is more suitable for design level pattern searching and thus they should be preferred in using the MAISA tool.

Auxiliary structures:

These pattern-like elements are used to indicate some key structures found in object-oriented programs. Among such structures are inherited attributes and class references (i.e. associations). While they are not design patterns, they are treated in a similar fashion during the search. As the actual patterns are based on these structures, they are calculated in most cases even if they are not explicitly selected.

Name	**Description**	**Requirements**
Derived Attribute	Inherited attributes.	-
Indirect Create	Indirect creation of instances	-
Indirect Read	Indirect read-access of an attribute.	Derived Attribute
Indirect Write	Indirect write-access to an attribute.	Derived Attribute
Single Reference	A class has a reference to (an instance of) another class.	Derived Attribute
Container	A class that contains a collection of objects.	Iterator
Extended Container	The subclasses of a container are also containers.	Container
Container Attribute	Attributes whose type is a container.	Extended Container
Multiple Reference	A class has a reference to (multiple instances of) another class.	Container Attribute

Metapatterns:

Several of the Gang-of-Four's pattern are equivalent to these patterns. The *Template Method*, for example, matches *Unification*. Note that the *1:1 Connection* pattern consists of three separate cases.

Pattern	**Description**	**Requirements**
1:1 Connection Reference	An *1:1 Connection*, where the association exists through a reference attribute.	Indirect Read, Single Reference
1:1 Connection Argument	An *1:1 Connection*, where the association exists through a method parameter.	1:N Connection
1:1 Connection AskRef	An *1:1 Connection*, where the association exists through the return value of a method.	1:N Connection
1:N Connection	As 1:1 Connection Reference, but there can be references to several instances.	1:1 Connection Reference, Multiple Reference, Indirect Read
Unification	A method calls a virtual method (of same class). The virtual method gets defined in subclasses.	-
1:1 Recursive Unification	A request is passed along a chain of instances (of same class) until the request is captured.	Indirect Read, Single Reference
1:N Recursive Unification	As above, but the request can be passed on to several objects	Indirect Read, 1:1 Recursive Unification, Multiple Reference
1:1 Recursive Connection	An object has a reference to its own superclass.	Indirect Read
1:N Recursive Connection	As above, but there can be several references.	1:1 Recursive Connection, Container Attribute

Gang-of-Four's patterns:

An asterisk (*) after the name of the pattern indicates that the pattern is currently not supported by the MAISA tool. There are two reasons for this. The pattern may be only a variation of one of the metapatterns. On the other hand, the pattern may not be easily expressed using purely syntactic rules.

Creational patterns:

Pattern	**Description**	**Requirements**
Abstract Factory	An interface for creating related objects without specifying the concrete class.	Factory Method
Builder	Separates the construction process of an object from the representation.	Indirect Create, Indirect Read, Indirect Write
Factory Method	An interface for creating an object without knowing its concrete class.	Indirect Create
Prototype	New instances are created by copying a prototype.	Indirect Create
Singleton	Ensures that only a single instance of a class exists at a time.	-

Structural patterns:

Pattern	**Description**	**Requirements**
Class Adapter	The interface of a class is converted into another via multiple inheritance.	-
Object Adapter	The interface of a class is converted into another via object composition.	Indirect Read, Single Reference
Bridge*	Decouples an abstraction from its implementation.	
Composite*	Hides part-whole hierarchies into composite objects. Individual objects and compositions can then be treated uniformly.	
Decorator*	Extends the functionality of an object dynamically.	
Facade*	A simple interface for a complex (sub)system.	
Flyweight	Supports a number of fine-grained objects efficiently by sharing (and thereby reducing the number of objects actually needed.	Indirect Create, Indirect Read, Indirect Write
Proxy	A proxy acts as a surrogate for another object.	Indirect Read, Single Reference, Virtual Proxy
Virtual Proxy	The creation of expensive objects is deferred until needed (a special case of *Proxy*).	Indirect Create, Indirect Read, Indirect Write, Single Reference Factory Method

Behavioral patterns:

Pattern	**Description**	**Requirements**
Chain of Responsibility*	Requests are passes along a chain of objects until an object handles it.	
Command*	Requests are encapsulated as objects.	
Interpreter*	The grammar of a programming language is represented as a class structure. This representation can then be used to interpret the language	
Iterator	Access to an aggregate object is separated from the underlying representation.	Indirect Create, Indirect Read, Indirect Write, Single Reference
Mediator	Loose coupling is promoted by centralizing requests to go via a mediator object.	Single Reference, Multiple Reference
Memento	Captures the internal state of an object without violating encapsulation.	Indirect Create Indirect Read, Indirect Write
Observer	Any changes in the state of an object are automatically relayed to its dependents.	1:1 Connection Reference, 1:N Connection, Indirect Create, Indirect Read, Indirect Write
State*	The behavior of an object changes according to changes in its internal state.	
Strategy*	A family of algorithms is defined as a set of interchangeable classes.	
Template Method*	The constant part of an operation is separated from the variable parts. The variable parts can then be implemented as needed by subclassing.	
Visitor	New operations can be defined without the need to change the classes those operations affect.	-

B File formats

Each file format is expressed as a grammar. The tokens between "<>" are nonterminal symbols and the tokens between "[]" are terminal symbols, as well as the quoted tokens. ":–" is the production symbol. The nonterminal symbols can be defined in a previous grammar, e.g the comparison result file grammar uses some nonterminal symbols from the metric result file grammar.

After each grammar is a sample file. Over 80 characters long rows continue in the next row. The sample files are truncated from real files.

There is some information that cannot (at last easily) be expressed by a grammar. The matrixes in the details part of the metrics result file contain only those metrics that are calculated, while the matrixes in the history data file contain all the available metrics. The last (rightmost) column of each history data matrix is "Time".

B.1 Metrics result files

:– <HEADER PART> <EMPTY ROW> <SUMMARY PART> <EMPTY ROW> <DETAILS PART>

<HEADER PART> :– <DATE ROW> <LINEBREAK> <ARCHITECTURE ROW> <LINEBREAK> <CONTENTS ROW>

<LINEBREAK> **<LINEBREAK>** :– <LINEBREAK>

<LINEBREAK> :– [linebreak]

<EMPTY ROW> **<EMPTY ROW>** :– <EMPTY ROW>

<EMPTY ROW> :– [empty row]

<DATE ROW> :– "Date: " <DATE>

<DATE> :– <WEEKDAY> " " <MONTH> " " <DAY> " " <HOURS> "." <MINUTES> "." <SECONDS> " " <YEAR>

<ARCHITECTURE ROW> :– "Architecture " <ARCHITECTURE NAME>

<CONTENTS ROW> :– "CONTENTS " <DIAGRAM NAME SET>

<DIAGRAM NAME SET> :– [empty] | <DIAGRAM NAME> <DIAGRAM NAME SET>

<DIAGRAM NAME> :– [string]

<ARCHITECTURE NAME> :– [string]

<WEEKDAY> :– "Mon" | "Tue" | "Wed" | "Thu" | "Fri" | "Sat" | "Sun"

<**MONTH**> :– “Jan” | “Feb” | “Mar” | “Apr” | “May” | “Jun” | “Jul” | “Aug” | “Sep” | “Oct” | “Nov” | “Dec”

<**DAY**> :– [1 .. 31]

<**HOURS**> :– [0 .. 23]

<**MINUTES**> :– [0 .. 59]

<**SUMMARY PART**> :– <RESULTS PART> <EMPTY ROW> <PATTERNS PART> <EMPTY ROW> <PARAMETERS PART>

<**RESULTS PART**> :– “Results” <LINEBREAK> <RESULT ROW SET>

<**PATTERNS PART**> :– “Patterns ” <LINEBREAK> <PATTERN ROW SET>

<**PARAMETERS PART**> :– “Parameters” <LINEBREAK> <PARAMETER SET>

<**RESULT ROW SET**> :– <RESULT ROW> <LINEBREAK> | <RESULT ROW> <LINEBREAK> <RESULT ROW SET>

<**RESULT ROW**> :– <METRIC NAME> “:” <LINEBREAK> <SPP> <SPD> <SPC>

<**SPP**> :– [empty] | “Per project - ” <SUMMARY ROW> <LINEBREAK>

<**SPD**> :– [empty] | “Per diagram - ” <SUMMARY ROW> <LINEBREAK>

<**SPC**> :– [empty] | “Per class - ” <SUMMARY ROW> <LINEBREAK>

<**METRIC NAME**> :– [string]

<**SUMMARY ROW**> :– “Min = ” [float] “; Max = “ [float] “; Avg = ” [float] “; Min source = “ [string] “; Max source = “ [string]

<**PATTERN ROW SET**> :– <PATTERN ROW> <LINEBREAK> | <PATTERN ROW> <LINEBREAK> <PATTERN ROW SET>

<**PATTERN ROW**> :– <PATTERN NAME> “:” <LINEBREAK> <INSTANCE ROW SET>

<**INSTANCE ROW SET**> :– <INSTANCE ROW> <LINEBREAK> | <INSTANCE ROW> <LINEBREAK> <INSTANCE ROW SET>

<**INSTANCE ROW**> :– “Instance - ” <ROLE BINDING SET>

<**ROLE BINDING SET**> :– <ROLE BINDING> | <ROLE BINDING> “, “ <ROLE BINDING SET>

<**ROLE BINDING**> :– <ROLE NAME> “=” <BINDED NAME>

<**ROLE NAME**> :– [string]

<**BINDED NAME**> :– [string]

<**PATTERN NAME**> :– [string]

<**PARAMETER SET**> :– <PARAMETER ROW SET> <LINEBREAK> <PATTERN PARAMETERS>

<**PATTERN PARAMETERS**> :– <RECOGNITION> <LINEBREAK> <SHOWN> <LINEBREAK> <SC> <LINEBREAK> <P LOC>

<**RECOGNITION**> :– "Pattern recognition: " <LINEBREAK> <PATTERN SET>

<**SHOWN**> :– "Pattern shown: " <LINEBREAK> <PATTERN SET>

<**SC**> :– "Suspicious contracts: " <LINEBREAK> <PATTERN SET>

<**P LOC**> :– "Pattern based LOC: " <LINEBREAK> <PATTERN SET>

<**PATTERN SET**> :– empty | <PATTERN LIST>

<**PATTERN LIST**> :– <PATTERN NAME> | <PATTERN NAME> "; " <PATTERN LIST>

<**PARAMETER ROW SET**> :– [empty] | <PARAMETER ROW> <LINEBREAK> <PARAMETER ROW SET>

<**PARAMETER ROW**> :– <METRIC NAME> ":" <LINEBREAK> <PPP> <PPD> <PPC>

<**PPP**> :– [empty] | "Per project - " <THRESHOLDS>

<**PPD**> :– [empty] | "Per diagram - " <THRESHOLDS>

<**PPC**> :– [empty] | "Per class - " <THRESHOLDS>

<**THRESHOLDS**> :– "Lower threshold = " [float] ", Upper threshold = " [float] | "Lower threshold = " [float] | "Upper threshold = " [float]

<**DETAILS PART**> :– "Details" <EMPTY ROW> <PER PROJECT> <PER CLASS> <PER DIAGRAM> <PATTERN DATA>

<**PER PROJECT**> :– [empty] | "Per project" <LINEBREAK> <MATRIX> <EMPTY ROW>

<**PER CLASS**> :– [empty] | "Per class" <LINEBREAK> <MATRIX> <EMPTY ROW>

<**PER DIAGRAM**> :– [empty] | "Per diagram" <LINEBREAK> <MATRIX> <EMPTY ROW>

<**PATTERN DATA**> :– [empty] | "Pattern data" <LINEBREAK> <PATTERN MATRIX> <EMPTY ROW>

<**MATRIX**> :– <MATRIX HEADER> <LINEBREAK> <MATRIX ROW SET>

<**HEADER**> :– “Project name” | “Diagram name” | “Diagram type” | “Class name” | <METRIC NAME>

<**MATRIX HEADER**> :– <HEADER> | <HEADER> “:” <MATRIX HEADER>

<**MATRIX ROW SET**> :– <MATRIX ROW> | <MATRIX ROW> <LINEBREAK> <MATRIX ROW SET>

<**MATRIX ROW**> :– <VALUE> | <VALUE> “:” <MATRIX ROW>

<**VALUE**> :– [float] | [string]

<**PATTERN MATRIX**> :– <PATTERN MATRIX HEADER> <PATTERN MATRIX ROW SET>

<**PATTERN MATRIX HEADER**> :– “Architecture name:Pattern name:Pattern count”

<**PATTERN MATRIX ROW SET**> :– <PATTERN MATRIX ROW> | <PATTERN MATRIX ROW> <PATTERN MATRIX ROW SET>

<**PATTERN MATRIX ROW**> :– <ARCHITECTURE NAME> ”:” <PATTERN NAME> “:” [float]

A sample metric result file:

```
Date: Thu Aug 30 13.9.32 2001
Architecture calc
Contents Calculator (CD), MetaMaisa (CD)

Results
NOA:
Per project - Avg = 7.0
Per diagram - Min = 1.0;  Max = 6.0;  Avg = 3.5;  Min source = MetaMaisa (CD);
Max source = Calculator (CD)
Per class - Min = 0.0;  Max = 3.0;  Avg = 0.21212122;  Min source = [Client, DivE
xp, FwIterator, Iterator, MinusExp, NodeCounter, Parser, PlusExp, RevIterator, St
atistics, VariableExp, CActivityDiagram, CAssociation, CAttribute, CClass, CClass
Diagram, CElement, CMethod, CPackage, CParameter, CState, CStateDiagram];  Max so
urce = CalcExp
NOC:
Per project - Avg = 33.0
Per diagram - Min = 13.0;  Max = 20.0;  Avg = 16.5;  Min source = MetaMaisa (CD);
  Max source = Calculator (CD)

Patterns
Derived Attribute:
Instance - Sub = CAttribute; Super.attr = CElement.name; Super = CElement
Instance - Sub = CAttribute; Super.attr = CElement.oid; Super = CElement
Instance - Sub = CClass; Super.attr = CElement.name; Super = CElement
Single Reference:
Instance - ref = CPackage.belongsTo; Target = CPackage; Client = CPackage


Parameters
NOA:
Per project - Lower threshold = 6,  Upper threshold = 8
Per diagram - Lower threshold = 40,  Upper threshold = 60
Per class - Lower threshold = 6,  Upper threshold = 8
NOC:
Per diagram - Lower threshold = 10,  Upper threshold = 12
Pattern recognition:
Observer; Visitor; Singleton; Builder; Factory Method; Visitor2; Derived Attribute
Patterns shown:
Observer; Visitor; Singleton; Builder; Factory Method; Visitor2; Derived Attribute
Suspicious constructs:
Observer; Visitor; Singleton; Builder; Factory Method; Visitor2; Derived Attribute
Pattern based LOC:
Observer; Visitor; Singleton; Builder; Factory Method; Visitor2; Derived Attribute
```

```
Details

Per project
Project name:NOA:NOC:NOMA:NOAttr:DIT:NOI:NOM:Relative NOC:Relative NOA:Relative N
OMA:Relative NOAttr:Pattern count:Pattern instance count
calc:7.0:33.0:193.0:54.0:3.0:0.0:187.0:0.0:0.0:0.0:0.0:0.0:2.0:75.0

Per class
Project name:Diagram name:Class name:NOA:NOMA:NOAttr:NOM:Relative NOA:Relative NO
MA:Relative NOAttr:Pattern count:Pattern instance count
calc:Calculator (CD):CalcExp:3.0:10.0:0.0:10.0:0.0:0.0:0.0:1.0:18.0
calc:Calculator (CD):Client:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0
calc:Calculator (CD):ConstantExp:1.0:10.0:1.0:10.0:0.0:0.0:0.0:1.0:7.0

Per diagram
Project name:Diagram name:Diagram type:NOA:NOC:NOMA:NOOB:NOAttr:Time requirement,
 worst case:Time requirement, best case:Time requirement, typical case:DIT:NOI:NO
M:McCabe:Relative NOC:Relative NOA:Relative NOMA:Relative NOOb:Relative NOMsg:Rel
ative NOAttr:Pattern count:Pattern instance count
calc:Calculator (CD):1.0:6.0:20.0:140.0:0.0:9.0:0.0:0.0:0.0:1.0:0.0:140.0:0.0:0.0
:0.0:0.0:0.0:0.0:0.0:0.0:1.0:42.0
calc:MetaMaisa (CD):1.0:1.0:13.0:53.0:0.0:45.0:0.0:0.0:0.0:2.0:0.0:47.0:0.0:0.0:0
.0:0.0:0.0:0.0:0.0:0.0:1.0:33.0

Pattern data
Architecture name:Pattern name:Pattern count
calc:Derived Attribute:33.0
calc:Single Reference:42.0
```

B.2 Comparison result files

:– <C HEADER PART> <EMPTY ROW> <C SUMMARY PART> <EMPTY ROW> <C DETAILS PART>

<C HEADER PART> :– <DATE ROW> <LINEBREAK> <ARCHITECTURE ROW 1> <LINEBREAK> <CONTENTS ROW> <LINEBREAK> <ARCHITECTURE ROW 2> <LINEBREAK> <CONTENTS PART>

<ARCHITECTURE ROW 1> :– "Architecture 1 " <ARCHITECTURE NAME>

<ARCHITECTURE ROW 2> :– "Architecture 2 " <ARCHITECTURE NAME>

<C SUMMARY PART> :– <C RESULTS PART> <EMPTY ROW> <C PATTERNS PART> <EMPTY ROW> <C PARAMETERS PART>

<C PARAMETERS PART> :– "Parameters" <PARAMETER ROW SET>

<C RESULTS PART> :– "Results" <LINEBREAK> <C RESULT ROW SET>

<C RESULT ROW SET> :– <C RESULT ROW> | <C RESULT ROW> <LINEBREAK> <C RESULT ROW SET>

<C RESULT ROW> :– <METRIC NAME> ":" <LINEBREAK> <C SPP> <C SPD> <C SPC>

<C SPP> :– [empty] | "Per project - " <C SUMMARY ROW> <LINEBREAK>

<C SPD> :– [empty] | "Per diagram - " <C SUMMARY ROW> <LINEBREAK>

<C SPC> :– [empty] | "Per class - " <C SUMMARY ROW> <LINEBREAK>

<C SUMMARY ROW> :– "Min = " [float] "&" [float] "; Max = " [float] "&" [float] "; Avg = " [float] "&" [float] "; Min source = " [string] "&" [string] "; Max source = " [string] "&" [string]

<C DETAILS PART> :– <DETAILS PART 1> <EMPTY ROW> <DETAILS PART 2>

<DETAILS PART 1> :– "First details" <EMPTY ROW> <PER PROJECT> <PER CLASS> <PER DIAGRAM> <PATTERN DATA>

<DETAILS PART 2> :– "Second details" <EMPTY ROW> <PER PROJECT> <PER CLASS> <PER DIAGRAM> <PATTERN DATA>

A sample comparison result file:

```
Date: Fri Nov 9 16.44.5 2001
Architecture 1 calc
Contents  Calculator (CD), MetaMaisa (CD)
Architecture 2 cdtesti
Contents  Calculator (CD), Gradulaisen_tilat (STD), Laajennustesti3 (STD), act090
500 (AC), act250400 (AC), col110500 (CollD), demo2 (CD), MetaMaisa (CD), act1409
(AC)

Comparison
NOA:
Per project - Avg = 7.0 & 11.0
Per diagram - Min = 1.0 & 1.0;  Max = 6.0 & 6.0;  Avg = 3.5 & 3.6666667;  Min sou
rce = MetaMaisa (CD) & MetaMaisa (CD);  Max source = Calculator (CD) & Calculator
 (CD)
Per class - Min = 0.0 & 0.0;  Max = 3.0 & 3.0;  Avg = 0.21212122 & 0.28947368;  M
in source = [Client, DivExp, FwIterator, Iterator, MinusExp, NodeCounter, Parser,
 PlusExp, RevIterator, Statistics, VariableExp, CActivityDiagram, CAssociation, C
Attribute, CClass, CClassDiagram, CElement, CMethod, CPackage, CParameter, CState
, CStateDiagram] & [Client, DivExp, FwIterator, Iterator, MinusExp, NodeCounter,
Parser, PlusExp, RevIterator, Statistics, VariableExp, Henkilö, CActivityDiagram,
 CAssociation, CAttribute, CClass, CClassDiagram, CElement, CMethod, CPackage, CP
arameter, CState, CStateDiagram];  Max source = CalcExp & [CalcExp, Gradu]
NOC:
Per project - Avg = 33.0 & 38.0
Per diagram - Min = 13.0 & 5.0;  Max = 20.0 & 20.0;  Avg = 16.5 & 12.666667;  Min
 source = MetaMaisa (CD) & demo2 (CD);  Max source = Calculator (CD) & Calculator
 (CD)

First Details

Per project
Project name:NOA:NOC:NOMA:NOAttr:DIT:NOI:NOM:Relative NOC:Relative NOA:Relative N
OMA:Relative NOAttr:Pattern count:Pattern instance count
calc:7.0:33.0:193.0:54.0:3.0:0.0:187.0:0.0:0.0:0.0:0.0:0.0:3.0:79.0

Per class
Project name:Diagram name:Class name:NOA:NOMA:NOAttr:NOM:Relative NOA:Relative NO
MA:Relative NOAttr:Pattern count:Pattern instance count
calc:Calculator (CD):CalcExp:3.0:10.0:0.0:10.0:0.0:0.0:0.0:1.0:18.0
calc:Calculator (CD):Client:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0
calc:Calculator (CD):ConstantExp:1.0:10.0:1.0:10.0:0.0:0.0:0.0:1.0:7.0

Per diagram
Project name:Diagram name:Diagram type:NOA:NOC:NOMA:NOOB:NOAttr:Time requirement,
```

```
 worst case:Time requirement, best case:Time requirement, typical case:DIT:NOI:NO
M:McCabe:Relative NOC:Relative NOA:Relative NOMA:Relative NOOb:Relative NOMsg:Rel
ative NOAttr:Pattern count:Pattern instance count
calc:Calculator (CD):1.0:6.0:20.0:140.0:0.0:9.0:0.0:0.0:0.0:0.0:1.0:0.0:140.0:0.0:0.0:0.0
:0.0:0.0:0.0:0.0:0.0:0.0:1.0:42.0
calc:MetaMaisa (CD):1.0:1.0:13.0:53.0:0.0:45.0:0.0:0.0:0.0:0.0:2.0:0.0:47.0:0.0:0.0:0.0:0
.0:0.0:0.0:0.0:0.0:0.0:2.0:37.0

Pattern data
Architecture name:Pattern name:Pattern count
calc:Derived Attribute:33.0
calc:Single Reference:4.0
calc:Visitor2:42.0

Second Details

Per project
Project name:NOA:NOC:NOMA:NOAttr:DIT:NOI:NOM:Relative NOC:Relative NOA:Relative N
OMA:Relative NOAttr:Pattern count:Pattern instance count
cdtesti:11.0:38.0:194.0:68.0:4.0:0.0:188.0:0.0:0.0:0.0:0.0:0.0:3.0:85.0

Per class
Project name:Diagram name:Class name:NOA:NOMA:NOAttr:NOM:Relative NOA:Relative NO
MA:Relative NOAttr:Pattern count:Pattern instance count
cdtesti:Calculator (CD):CalcExp:3.0:10.0:0.0:10.0:0.0:0.0:0.0:1.0:18.0
cdtesti:Calculator (CD):Client:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0
cdtesti:Calculator (CD):ConstantExp:1.0:10.0:1.0:10.0:0.0:0.0:0.0:1.0:7.0

Per diagram
Project name:Diagram name:Diagram type:NOA:NOC:NOMA:NOOB:NOAttr:Time requirement,
 worst case:Time requirement, best case:Time requirement, typical case:DIT:NOI:NO
M:McCabe:Relative NOC:Relative NOA:Relative NOMA:Relative NOOb:Relative NOMsg:Rel
ative NOAttr:Pattern count:Pattern instance count
cdtesti:Calculator (CD):1.0:6.0:20.0:140.0:0.0:9.0:0.0:0.0:0.0:0.0:1.0:0.0:140.0:0.0:0.0:
0.0:0.0:0.0:0.0:0.0:0.0:0.0:1.0:42.0
cdtesti:Gradulaisen_tilat (StateD):4.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.
0:3.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0

Pattern data
Architecture name:Pattern name:Pattern count
cdtesti:Derived Attribute:39.0
```

B.3 History data files

:– <H PER PROJECT> <H PER CLASS> <H PER DIAGRAM> <H PATTERN DATA>

<H PER PROJECT> :– [empty] | “Per project” <LINEBREAK > <H MATRIX> <EMPTY ROW>

<H PER CLASS> :– [empty] | “Per class” <LINEBREAK > <H MATRIX> <EMPTY ROW>

<H PER DIAGRAM> :– [empty] | “Per diagram” <LINEBREAK > <H MATRIX> <EMPTY ROW>

<H PATTERN DATA> :– [empty] | “Pattern data” <LINEBREAK > <H PATTERN MATRIX> <EMPTY ROW>

<H MATRIX> :– <H MATRIX HEADER> <LINEBREAK> <H MATRIX ROW SET>

<H HEADER> :– <HEADER> | “Time”

<H MATRIX HEADER> :– <H HEADER> | <H HEADER> “:” <H MATRIX HEADER>

<H MATRIX ROW SET> :– <H MATRIX ROW> | <H MATRIX ROW> <LINEBREAK> <MATRIX ROW SET>

<H MATRIX ROW> :– <H VALUE> | <H VALUE> “:” <H MATRIX ROW>

<H VALUE> :– <VALUE> | <DATE>

<H PATTERN MATRIX> :– <PATTERN MATRIX HEADER> “:Time” <H PATTERN MATRIX ROW SET>

<H PATTERN MATRIX ROW SET> :– <H PATTERN MATRIX ROW> | <H PATTERN MATRIX ROW> <H PATTERN MATRIX ROW SET>

<H PATTERN MATRIX ROW> :– <ARCHITECTURE NAME> ”:” <PATTERN NAME> “:” [float] “:” <DATE>

A sample history data file:

```
Per project
Project name:NOA:NOC:NOMA:NOAttr:DIT:NOI:NOM:Relative NOC:Relative NOA:Relative N
OMA:Relative NOAttr:Pattern count:Pattern instance count:Time
cdtesti:11.0:38.0:194.0:68.0:4.0:0.0:188.0:0.0:0.0:0.0:0.0:0.0:3.0:85.0:Thu Nov 29 13
.8.15 2001
calc:7.0:33.0:193.0:54.0:3.0:0.0:187.0:0.0:0.0:0.0:0.0:0.0:3.0:79.0:Thu Nov 29 13.9.2
7 2001
coords1:4.0:5.0:1.0:14.0:1.0:0.0:1.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:Thu Nov 29 12.31.59
2001

Per class
Project name:Diagram name:Class name:NOA:NOMA:NOAttr:NOM:Relative NOA:Relative NO
MA:Relative NOAttr:Pattern count:Pattern instance count:Time
cdtesti:Calculator (CD):CalcExp:3.0:10.0:0.0:10.0:0.0:0.0:0.0:1.0:18.0:Thu Nov 29
 13.8.15 2001
cdtesti:Calculator (CD):Client:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:Thu Nov 29 13.
8.15 2001
cdtesti:Calculator (CD):ConstantExp:1.0:10.0:1.0:10.0:0.0:0.0:0.0:1.0:7.0:Thu Nov
 29 13.8.15 2001

Per diagram
Project name:Diagram name:Diagram type:NOA:NOC:NOMA:NOS:NOT:NOOp:NOMsg:NOOB:NOAtt
r:Time requirement, worst case:Time requirement, best case:Time requirement, typi
cal case:DIT:NOI:NOM:McCabe:Relative NOC:Relative NOA:Relative NOMA:Relative NOOb
:Relative NOMsg:Relative NOAttr:Pattern count:Pattern instance count:Time
cdtesti:Calculator (CD):1.0:6.0:20.0:140.0:0.0:0.0:0.0:0.0:0.0:0.0:9.0:0.0:0.0:0.0:1.
0:0.0:140.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:1.0:42.0:Thu Nov 29 13.8.15 2001
cdtesti:Gradulaisen_tilat (StateD):4.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.
0:0.0:0.0:0.0:0.0:3.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:Thu Nov 29 13.8.15 2001
cdtesti:Laajennustesti3 (StateD):4.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:
0.0:0.0:0.0:0.0:0.0:4.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:0.0:Thu Nov 29 13.8.15 2001

Pattern data
Architecture name:Pattern name:Pattern count:Time
cdtesti:Derived Attribute:39.0:Thu Nov 29 13.8.15 2001
cdtesti:Single Reference:4.0:Thu Nov 29 13.8.15 2001
cdtesti:Visitor2:42.0:Thu Nov 29 13.8.15 2001
Visitor:Visitor2:42.0:Thu Nov 29 13.8.15 2001
CPackage:Derived Attribute:2.0:Thu Nov 29 13.8.15 2001
CPackage:Single Reference:2.0:Thu Nov 29 13.8.15 2001
CMethod:Derived Attribute:2.0:Thu Nov 29 13.8.15 2001
```

B.4 Metric configuration files

:– <PARAMETER ROW SET> <LINEBREAK> <M PATTERN PARAMETERS>

<M PATTERN PARAMETERS> :– <M RECOGNITION> <LINEBREAK> <M SHOWN> <LINEBREAK> <M SC> <LINEBREAK> <M P LOC>

<M RECOGNITION> :– "Pattern recognition: " <LINEBREAK> <M PATTERN SET>

<M SHOWN> :– "Pattern shown: " <LINEBREAK> <M PATTERN SET>

<M SC> :– "Suspicious contracts: " <LINEBREAK> <M PATTERN SET>

<M P LOC> :– "Pattern based LOC: " <LINEBREAK> <M PATTERN SET>

<M PATTERN SET> :– empty | <M PATTERN LIST>

<M PATTERN LIST> :– <PATTERN NAME> | <PATTERN NAME> ", " <M PATTERN LIST>

A sample metric configuration file:

```
NOA:
Per project - Lower=6.0, Upper=8.0
Per diagram - Lower=40.0, Upper=60.0
Per class - Lower=6.0, Upper=8.0
NOC:
NOMA:
Per diagram - Lower=10.0, Upper=12.0
NOOB:
NOAttr:
Time requirement, worst case:
Time requirement, best case:
Time requirement, typical case:
DIT:
Per project - Lower=0.1, Upper=0.5
Per diagram - Lower=0.6, Upper=1.0
Pattern recognition:
Singleton, Builder, Factory Method, Visitor2, Derived Attribute, Observer, Visito
r, Single Reference
Patterns shown:
Singleton, Builder, Factory Method, Visitor2, Derived Attribute, Observer, Visito
r, Single Reference
Suspicious constructs:
Singleton, Builder, Factory Method, Visitor2, Derived Attribute, Observer, Visito
r, Single Reference
Pattern based LOC:
Singleton, Builder, Factory Method, Visitor2, Derived Attribute, Observer, Visito
r, Single Reference
```

References

[Gam1995] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley 1995.

[Gus2000] Juha Gustafsson: *Metrics calculation in MAISA*, University of Helsinki, Department of Computer Science, August 2000.

[Mis1997] Pauli Misikangas: *Suunnittelumallien automaattinen tunnistaminen olio-ohjelmista*, Master's Thesis, University of Helsinki, Department of Computer Science, C-1998-1, 1997.

[NeG2000] Lilli Nenonen, Juha Gustafsson: *Pattern recognition in the MAISA tool*, University of Helsinki, Department of Computer Science, August 2000.

[Pre1995] Wolfgang Pree: *Design Patterns for Object-Oriented Software Development*, Addison-Wesley, 1995.

[UML] Unified Modeling Language Resource Center
`http://www.rational.com/uml/`