1. Working with CREAM v.3.0.

Here is the user guide for CREAM v.3.0. The process of installation and configuration is described in chapter 1.1. The following sections show how you can use the features of CREAM v.3.0.

1.1. Installing and configuring CREAM

CREAM v.3.0. is integrated with external applications that make it possible to use the full functionality of the tool. It is recommended to install them prior to the installation of CREAM. These include:

- **.NET Framework 4.0** [1]
- **Visual Studio** in one of the following versions:
  - Visual Studio 2010 Premium.
  - Visual Studio 2010 Ultimate.
  - Visual Studio 2008 Team System.
- **NUnit** v.2.5.7. [2]
- **NCover** v.3.4.14 [3]
- **Tester** [4]
- **SVN Client** [5]
- **Microsoft Excel** 2007 or higher

The first two items in the list are required, while the remaining are optional, depending on the CREAM functionality and user's choice.

A user should install additional software if they want to implement tools for unit tests different than the ones shipped with Visual Studio. The tool supports unit tests prepared with the use of MSTest and NUnit

In addition, if you want to use another tool (than the one built in Visual Studio) to test code coverage, you can choose NCover or Tester.

SVN Client is required if a user wants to store mutant files on the SVN server.

It is also recommended to install Microsoft Excel 2003 or higher since the libraries supplied with this program are used when saving reports to .xls files.

In order to install CREAM v.3.0. download setup.msi file from [6]. Once you run the installer, follow the on-screen prompts.

When the installation is successful, you can run CREAM v.3.0 using the shortcut on the desktop or selecting the appropriate item from the Windows Start menu *(Programs-> EiTIPW-> CREAM)*. At this point you may be asked to install .NET Framework 4.0. as it is required by the application.

The tool is supplied with sample files with saved mutants and the results of the tests conducted on them (the files with the .mut extension). Therefore, a user can test all the available tool features without the need to generate and test mutants.

Once you run CREAM v.3.0. the main application window appears (Fig. 1).

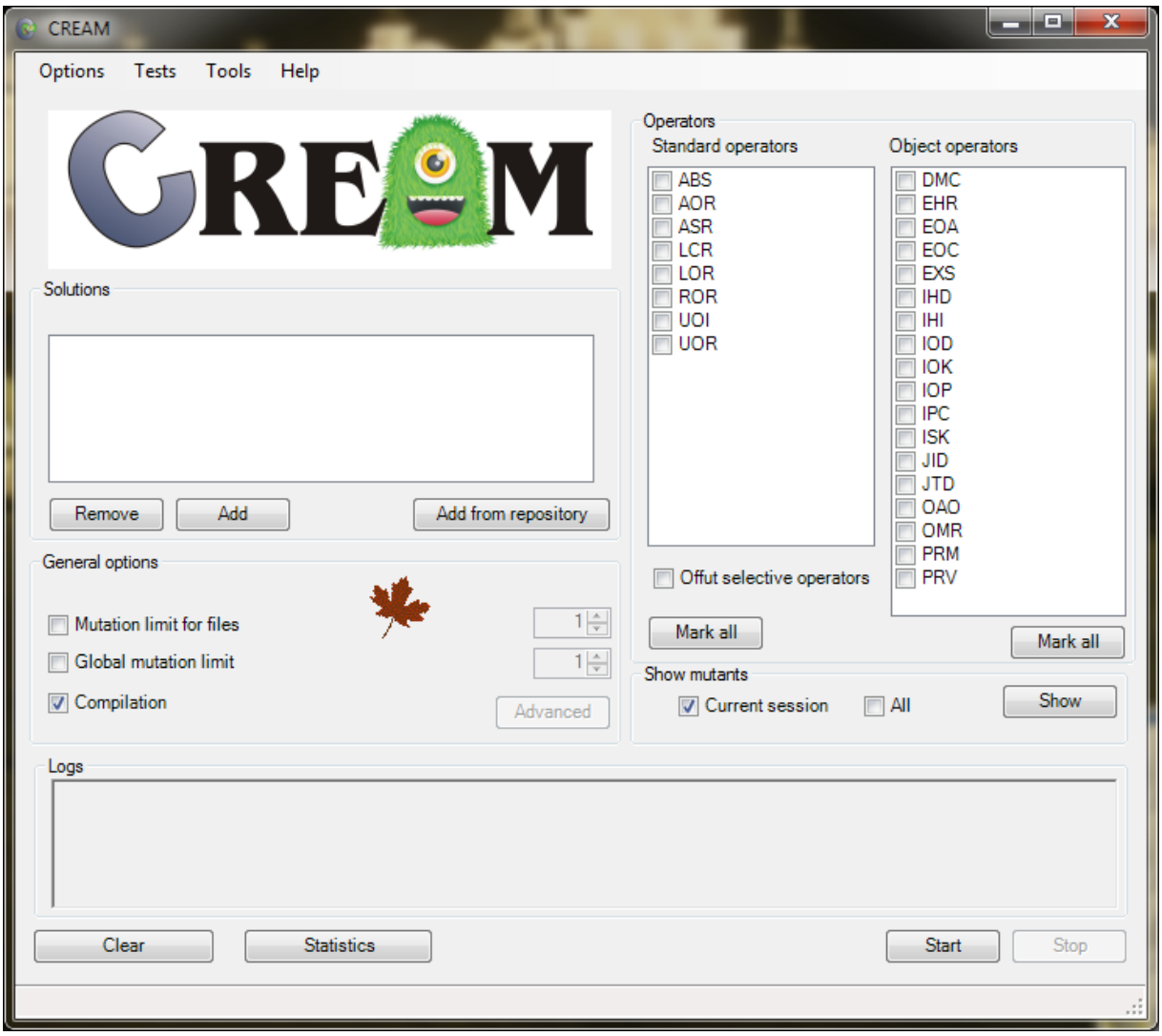

Figure 1 CREAM v.3.0. - Main application window

First of all, the application should be configured. To do this, select *Options->CREAM Settings* from the menu at the top of the main application window. The Settings window will appear (Figure 2). To configure the application you should fill in the following data:

- *Input source directory* - the default directory to search for programs
- *Svn.exe file path* - the directory with the svn.exe file (optional, required only when using the function of storing mutants in the svn repository)
- *Output local directory* - the working directory of CREAM v.3.0. (a user must have permission to write in this directory)
- *Default repository* - the SVN repository default address (optional, needed only if a user wants to save mutants on the SVN server.
- *devenv.exe localization* - the path to devenv.exe (compiler shipped with Visual Studio).
- *Unit testing tool localization* - the path to the unit test tool. It will be either the path to the directory with the nunit-console.exe (unit tests prepared with NUnit) or mstest.exe (unit tests prepared with the tool built in Visual Studio)

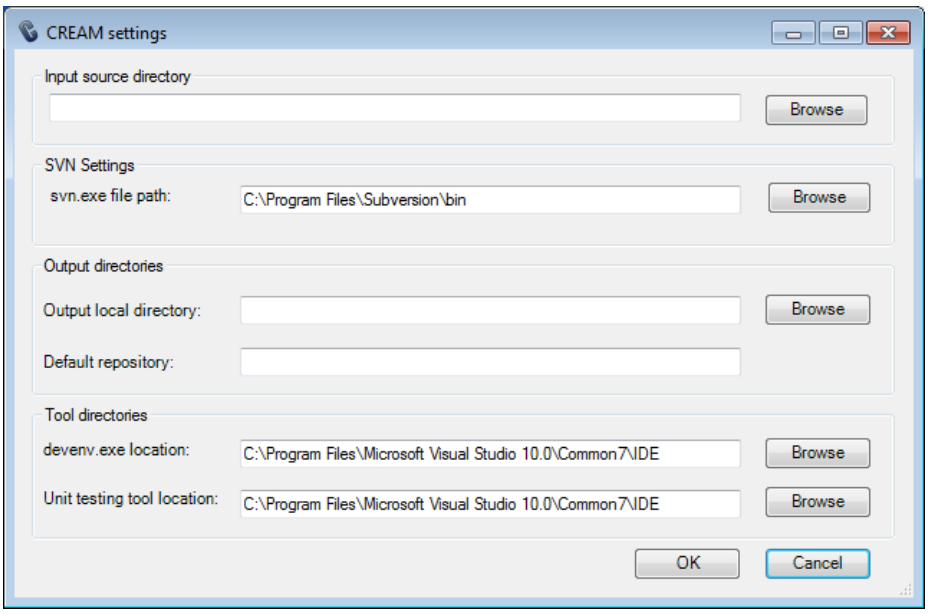

Figure 2 CREAM v.3.0. - The application settings window

After you have filled in the form, the tool is configured and ready to work. The next step is to generate and test mutants (see sections 1.4-1.9).

1.2. Loading the project

In order to load the project you have to:
1) Run and configure CREAM v.3.0. as described in chapter 1.1.
2) Load a solution with a program written in C# with the use of Visual Studio by indicating the location of the .sln file (item 1a or 1b in Figure 3).

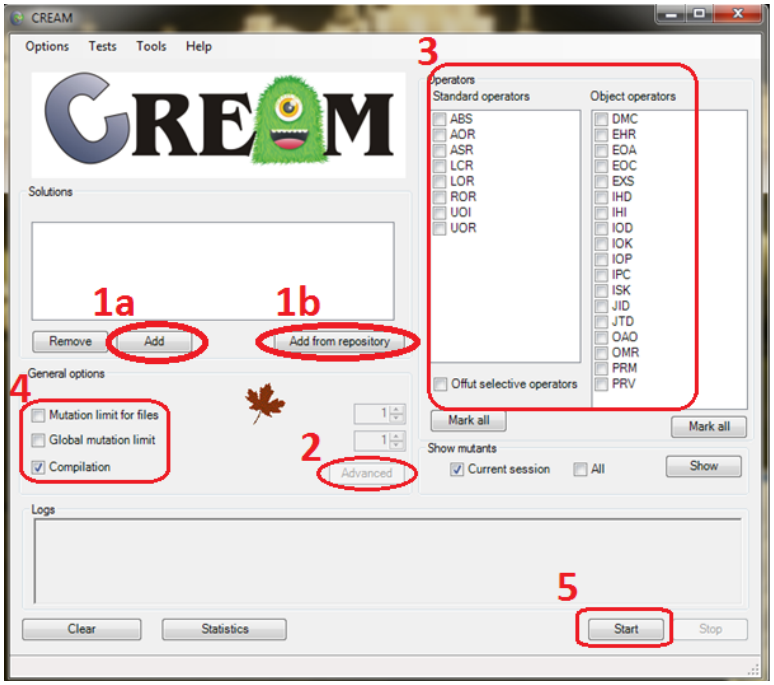

Figure 3 CREAM v.3.0 main application window - the process of generating mutants

a) loading the solution stored on the local disk: in order to load the solution stored on the local disk press the *Add* button in the *Solutions* section of the main application window (1a in Figure 3), and then indicate the location of the .sln file.
b) loading the solution stored in the SVN repository: to load the solution stored in the SVN repository press the *Add from repository* button in the *Solutions* section of the main application window (1b in Figure 3). The window will appear (Fig. 4) and you can browse repositories. To select the solution file in a given repository you should:
- Enter a full repository address in the *Reposiory root* box and press the *Connect* button
- Select a revision number from the *Revision* list where the non-mutated project is stored
- In the tree in the *Contents* section, select a file (under the given revision) with the .sln extension and press *OK*.

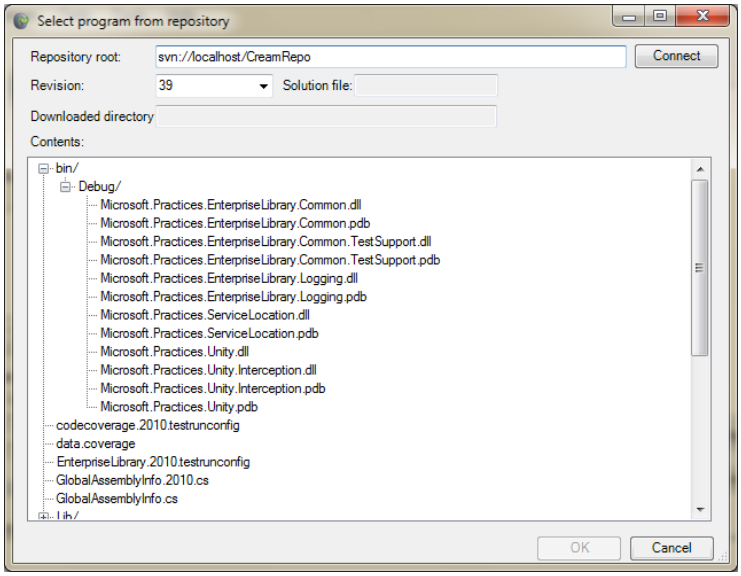

Figure 4 Browsing SVN repository

1.3. Using the SVN repository wizard

CREAM v.3.0. offers the SVN repository wizard. You can run it by selecting *SVN Repository Creator* in the Tools menu of the main application window (Fig. 5).

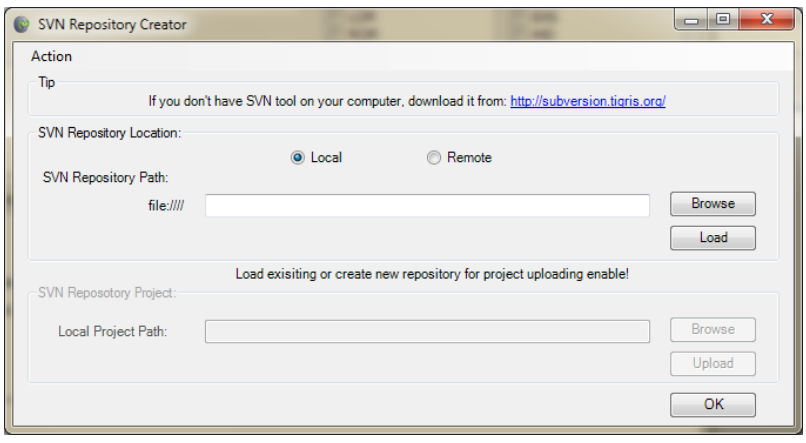

Figure 5 SVN Repository Creator

First you have to load the SVN repository. To do this: in *SVN Repository Location* choose whether the repository is local or remote and fill the address box. When you have provided the correct data and pressed the *Load* button the repository is ready. If it is empty, a new project can be loaded and then used as the original project for the mutation process.

If you select the local repository and provide the address referring to an empty directory you will be asked if you want to create an empty repository. After selecting 'yes' the wizard creates a new repository where we can load the project.

1.4. Generating mutants (base scenario)

In order to generate mutants, perform the following steps:

1. Run, configure and load the project as described in chapter 1.2.
2. Select the operators with the use of which the mutants are to be created (point 3 in Figure 3). It should be noted that under the list of structural operators a user can select *Offutt selective operator*. By selecting it a subset of selective structural operators is loaded (according to the research described in [7])
3. In the *General Options* section of the main window set additional parameters involved in the mutation process (point 4 in Figure 3):
   - Define the maximum number of mutation that can be implemented using different operators in each file (*Mutation limit for files* box)
   - Define the maximum number of mutation that can be implemented by each of the operators *(Global mutation limit* box).
   - Determine whether implementing the mutation will trigger the compilation process (*Compilation* box)
4. Start the process of generating mutants by pressing the *Start* button (point 4 in Figure 3).

1.5. Generating mutants (advanced scenario)

You can use additional features of CREAM v.3.0. by the following steps:

1. Follow the steps: 1, 2 and 3 from the base scenario (Section 1.4)
2. Select *Advance* in *General options* section (point 2 in Figure 3) or *Advanced* in the *Options* menu of the main window and the advanced options window will appear (Fig. 6).

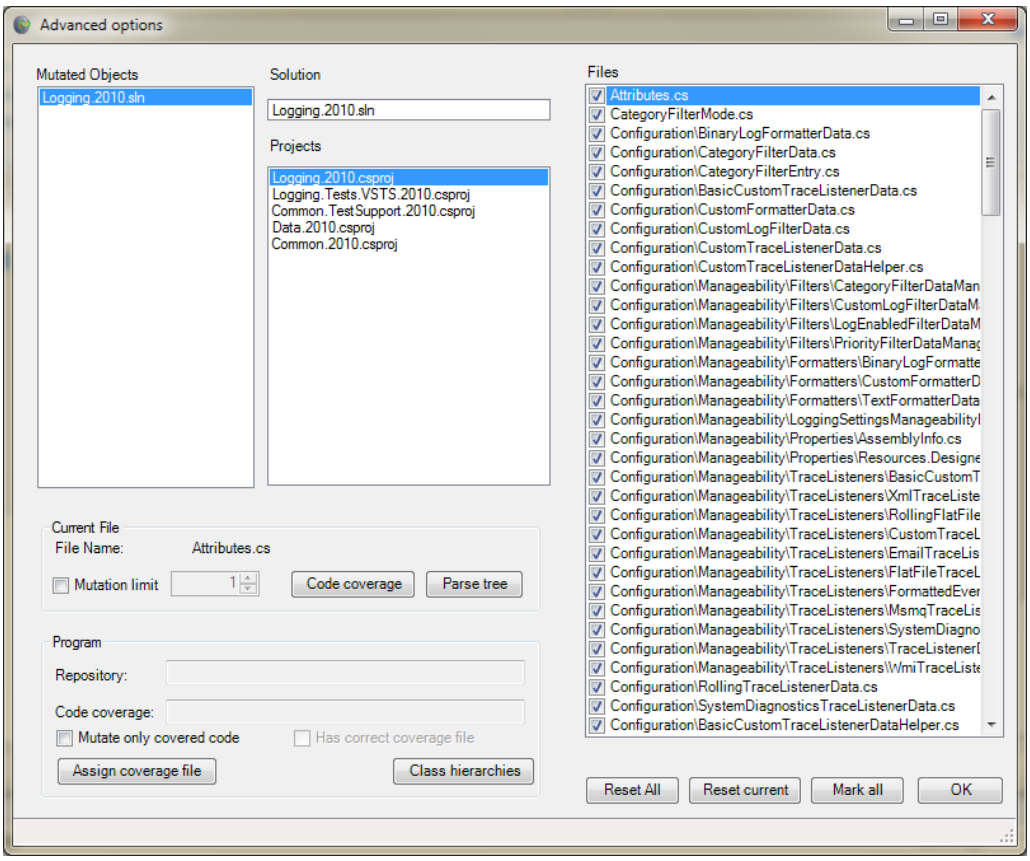

Figure 6 CREAM v.3.0. - Advanced options for mutations

In this window you can see the directory structure and files of the loaded project. By checking / unchecking the files in the *Files* section, you can decide which of them will participate in the process of mutation.

In the *Advanced Options* window, you can also assign a code coverage file to the loaded project (by pressing *Assign coverage file* and selecting the previously generated file) and define that only covered pieces of code should be mutated (by selecting *Mutate only covered code* in the *Program* section). CREAM v.3.0. recognizes files with information about the code coverage generated by: NCover [3] (.xml files), Visual Studio (.coverage files), Tester [4] (.txt files). After selecting the file with the information about the coverage you will be asked how to treat the lines, of which no information is available in the coverage files. It is recommended to treat them as not covered lines.

In order to verify the accuracy of the loaded data the user can select any file from the list and, by pressing the *Code coverage* button (under *Current File)*, display its code with coverage highlighted (Fig. 7). In this window, each line of the code represents different information, depending on its color:

- Black - no file with coverage information
- Green - code covered
- Light green - no information on a given line of code treated as covered
- Red - code uncovered
- Bright red - no information on a given line of code treated as uncovered

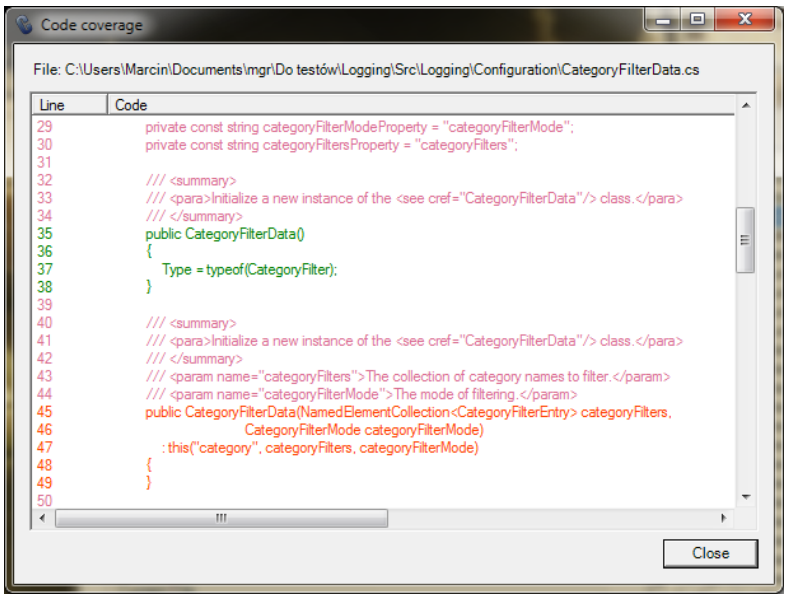

Figure 7 CREAM v.3.0. - Window with information about the code coverage

The *Advanced Options* dialog provides the possibility to limit the number of mutations which will be applied to individual files in the mutation process.

In this window you can also view the parse tree of the selected file (Fig. 8) by choosing the file in the *Files* section and click on *Parse tree* in the *Current File* section.

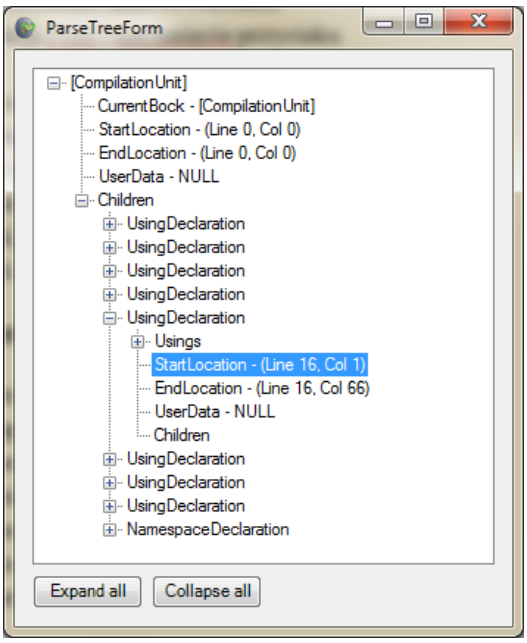

Figure 8: Parse tree of a file

1.6. Reviewing generated mutants

To review generated mutants you should:

1. Generate mutants as described in section 1.4 or 1.5.
2. Select the *Show* button in the *Show mutants* section of the main application window. A dialog with mutants appears (Fig. 9). In this window, you can choose previously generated mutants from the list on the left side. Next, a window with the original and mutated code

appears, along with the lines where the changes were made. In this window you can view both mutants stored locally as well as those from the SVN repository.

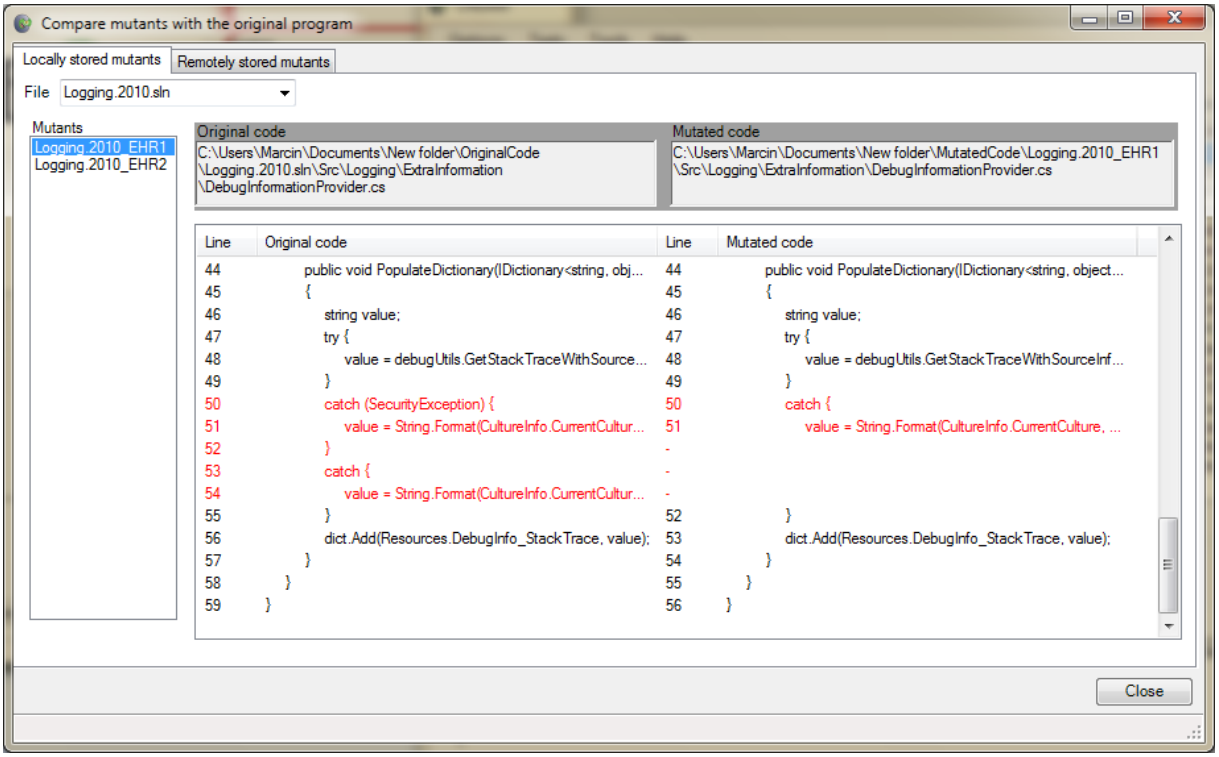

Figure 9 Reviewing mutants

1.7. Testing mutants

CREAM v.3.0. (when compared to v.2.0.) provides a new way of testing mutants with the possibility to save the results to a file. Thereby, the processes of generating and testing mutants can be isolated from further analyses.

To perform tests on the mutants and generate a file with the results follow these steps:

1. Generate mutants in accordance with the instructions in sections 1.4 or 1.5.
2. In the main application window select *Run tests* from the *Tests* menu
3. In the first wizard window, read the information on how to carry out the testing process and click *Next.*
4. In the second step, choose where the mutants are stored: local disk or SVN repository.
5. In the next step of the wizard, select the appropriate data to load the project, mutants and tests. Depending on whether the mutants and the project are stored on your local disk or in the SVN repository you will be asked for additional data. The difference lies in how to indicate a place where the original project is stored (see Section 2, Chapter 1.2). Other information include: the relative path to the test files within the project (.dll files) and the choice of external tools to perform unit testing (NUnit or MSTest). After providing the above data, press *Next.*
6. The last step is to choose whether we want to test all of the mutants or just the selected ones (*Prepare data from* in the wizard), generate results (by pressing the *Run* button*)* and save the file with data with .mut extension on the local disk *(Save* button).
7. Test results saved in the generated file can be viewed in the *Data Viewer* (described in Chapter 1.8).

1.8. Data Viewer

The basic feature of the *Data Viewer* is to display data stored in .mut files. The main application window is shown in Figure 10.

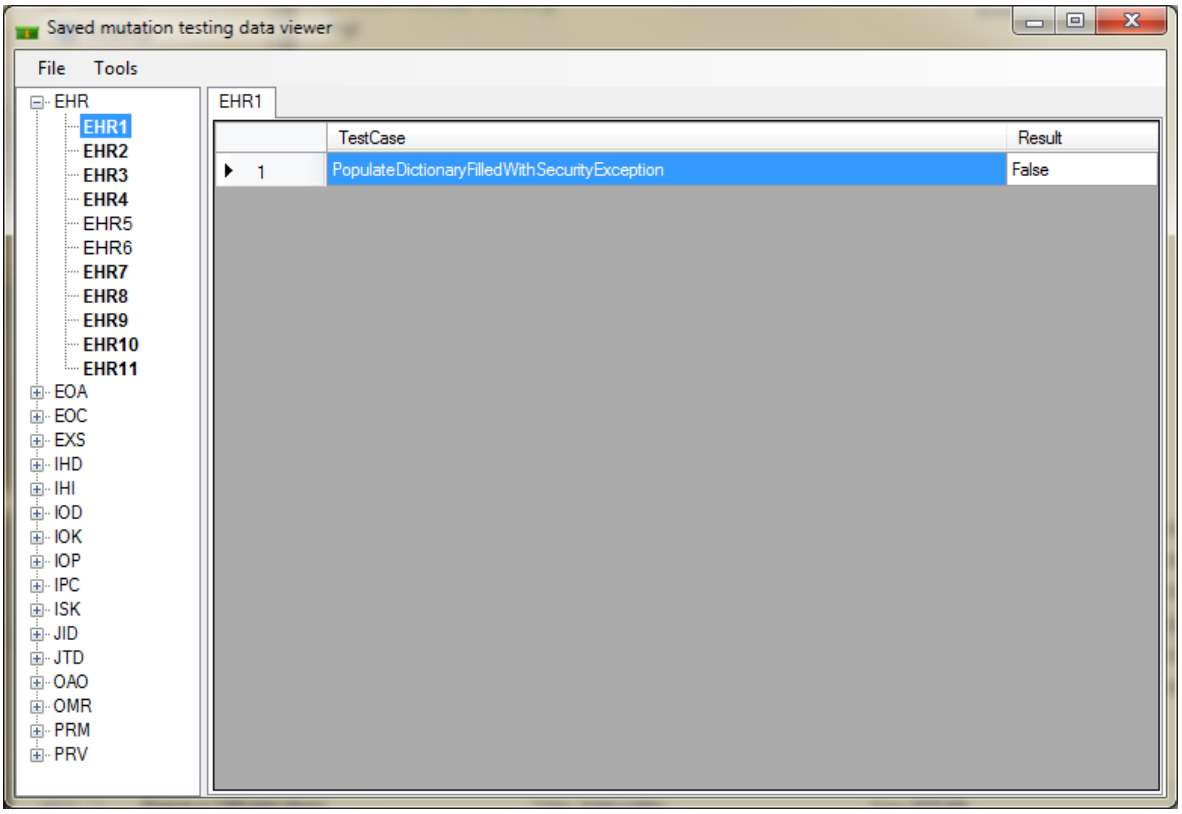

Figure 10: Data Viewer main window

The application can be launched in different ways:

- By double-clicking on the file with the .mut extension. *Data Viewer* is automatically associated with .mut files during the installation. When you start the application you will immediately see the data stored in the selected file.
- By selecting *Data Viewer* from the Windows Start menu (by default in the EiTIPW directory). You can load the file by selecting *Load* from the *Options* menu.
- By selecting *Data Viewer* in Tools menu in the main CREAM window.

If you run the application in a different way than double-clicking on the .mut file, you have to load the data file (generated by CREAM v.3.0.). After loading the file, a tree with mutants (grouped by operators) appears on the left side of the main application window. Selecting one of the available mutants on the right will open the tab with information about the tests that kill it.

The more detailed information about the selected mutant may be obtained by right-clicking on it and selecting *Properties* from the context menu (Figure 11). Among the information presented in the window, some of the boxes are grayed out (you cannot edit them), and some are editable. Remember that when you make changes, they will be saved only if you click on *Submit*.

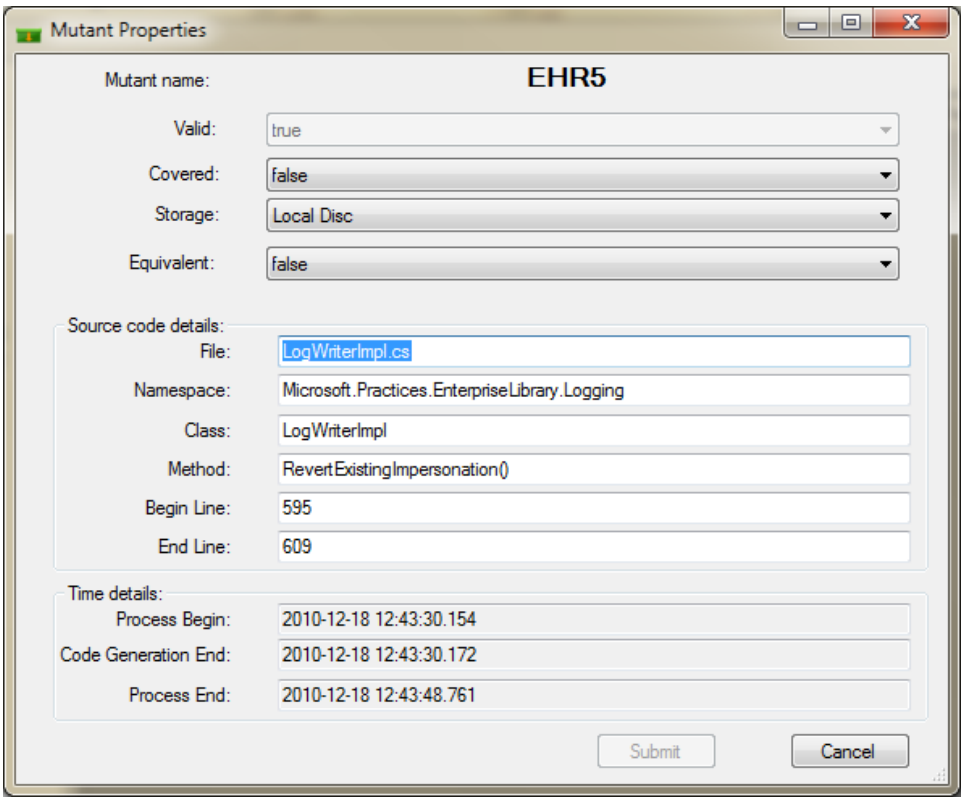

Figure 11: Mutant Properties window

Additionally, the application allows you to perform several operations on the files. These features are accessible from the *Tools* menu in the main window:

- *Merge Files* - opens a dialog where a user can select the original file and additional files, which can then be combined into one output file. Mutants stored in the additional files are only attached if there isn't any mutant of the same name in the original file, or when they haven't been attached previously. Having done this, save the generated file (select *File -> Save As* from the menu in the main window).
- *Statistics* - summary of the number of loaded mutants divided into types. Additionally, you will find the number of killed mutants and the Mutation Score here. You can also set your own criteria taken into account in calculating the statistics (live/killed, covered/uncovered, equivalent/non-equivalent mutants).
- *Check for uncovered and killed* - find uncovered mutants that can be killed by appropriate tests. This information is used to verify the accuracy of the loaded data

1.9. Using the feature of testing cost reduction methods

Files with tests results (with *.mut* extension) generated as described in sections 1.5 or 1.6. can provide input data for further analyses examining cost reduction methods.

The following steps of the analyses are visualized in the form of wizards. All of them have some common parts, regardless of the chosen analysis (as described in Section 1.9.1) as well as analysis-specific parts (as described in Sections 1.9.2, 1.9.3, 1.9.4).

1.9.1. Common parts of the visualizations

The first step of each visualization is a brief description of the selected analysis.

The next common step is to load the input data (Fig. 12). In this step the user can load the data from an existing file *(Load saved analysis input data* position in the *Input data* section*),* or open the wizard to generate the input data (*Generate new analysis input data* position in the *Input data* section*),* which was described in section 1.7.

In this window a filter (see *Mutants filter)* selects mutants for further analysis:

- Covered by the tests or all regardless of coverage (*Covered mutants* option)
- Compilable or all (*Valid* option)
- Marked as non-equivalent or all (*Nonequivalent* option)

In addition, in this step the user can select the type of analysis (*Testing type* section):

- Single test - the course of the analysis is presented step by step, the user can manually set the parameters and observe the results of the succeeding stages.
- Overall test - the course of the analysis is presented in the form of a report. It contains the complete set of input parameters, and the generation takes place without a user interaction. From the user's perspective the report is generated in the same manner, regardless of the selected analysis. The differences are visible only in the final report.

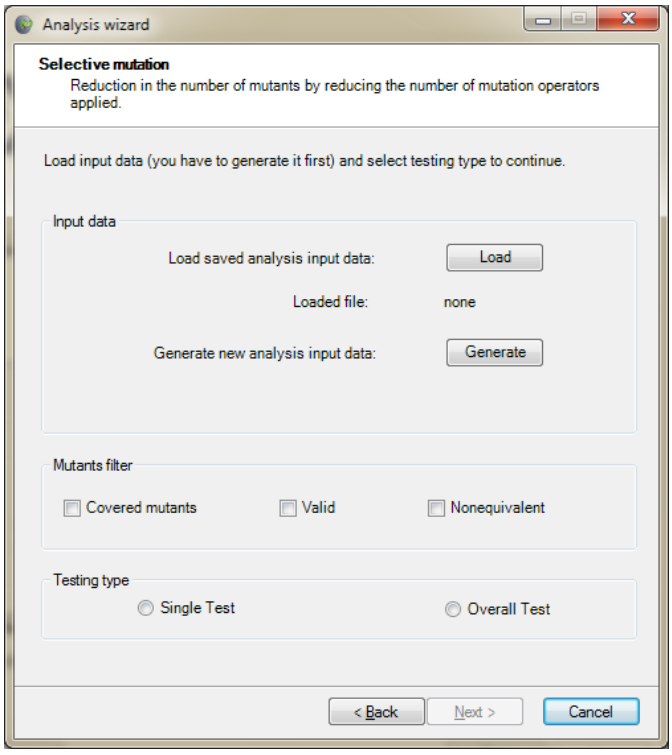

Figure 12 Loading the input data for analysis

After loading the data the user can select a subset of mutants. It depends on the chosen analysis and it is described in following sections.

Once we have selected a subset of mutants the process of generating minimal sets of the test cases begins (Fig. 13). You can choose one of the available algorithms and, by pressing the *Run* button, start the process of generating. When it is completed, its results will be presented in a table. Depending on the algorithm and its parameters the process can generate more than one set. The user selects one of them using the drop-down list above the table with the results.

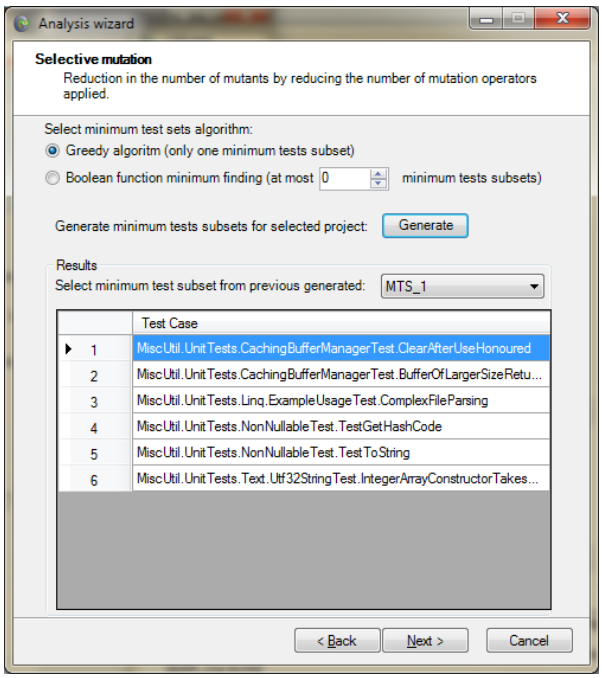

Figure 13 Generating a minimal set of test cases

Once you have selected a subset of test cases, the next step is to calculate the statistics and the corresponding MS values, which are the results of the given analysis (Fig. 14).

The meaning of particular MS values is as follows. The first MS value specifies the percentage of killed mutants (out of all generated ones) with the use of all supplied test cases. Whereas the MS values presented below are calculated using the subset of test cases from the previous step of test case analysis and all the generated mutants.

After viewing the results, the user can close the wizard *(Finish* button), or go back and perform another analysis with different parameters *(Back* button).

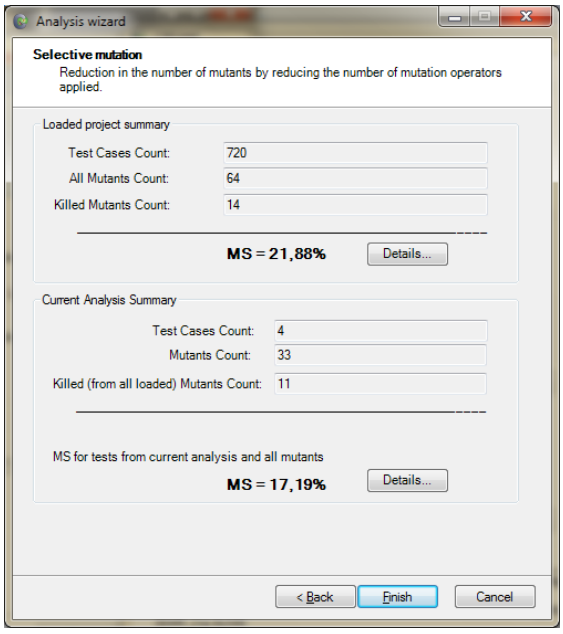

Figure 14 Results of the analysis

1.9.2. Selective mutations

A subset of mutants for the selective mutation may be selected in two ways:

- by defining the number of ignored operators (Fig. 15). In this approach we successively eliminate the operators according to their popularity (the number of generated mutants). To do this you have to choose a value from the *Most popular operator count ommited in analysis* list in the wizard.
- by manually selecting the operators that we want to take into consideration in the further analysis. It can be done by selecting *User operators select* option.

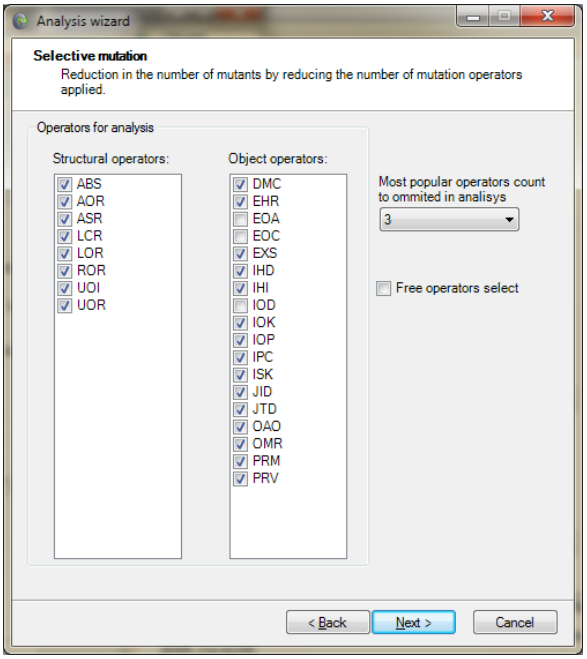

Figure 15 Selecting the operators to ignore in the selective mutation

1.9.3. Sampling mutants

In this approach, a subset of mutants is chosen at random. As parameters you can provide the type of randomness (in *Random Type* list) and the percentage of mutants to be selected (choosing a value from *Percent of randomly selected mutants)* (Fig. 16)

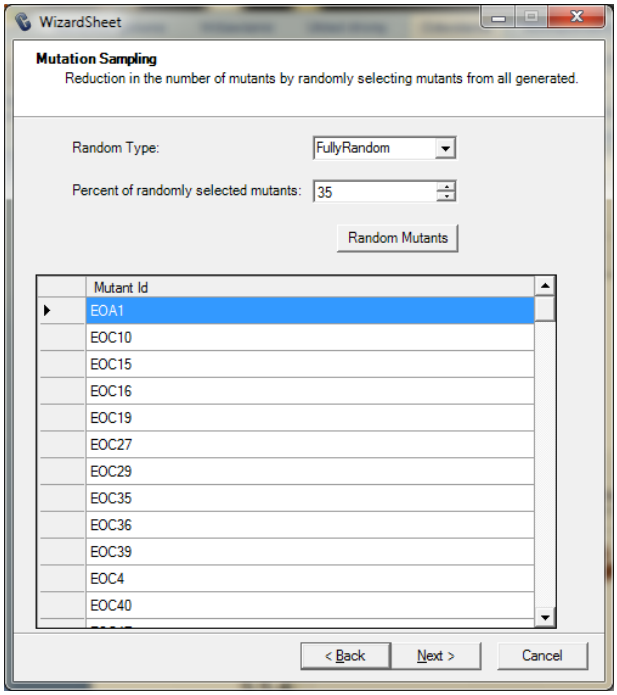

Figure 16 Selection of mutants - sampling mutants

1.9.4. Clustering mutants

In the clustering method a subset of mutants is selected as representatives of generated clusters (Fig. 17). As a parameter you specify the similarity ratio between members of the clusters (in *Threshold for clustering algorithm)*.

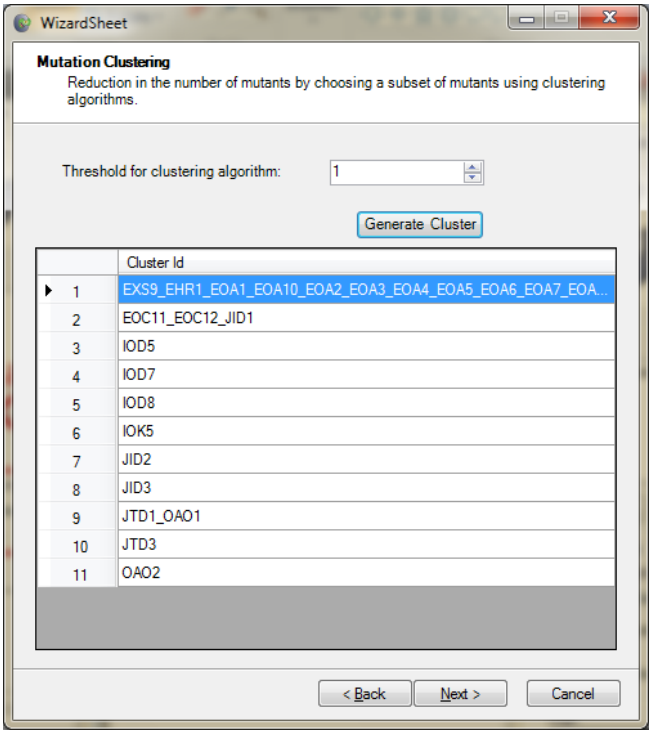

Figure 17 Generating a subset of mutants in the process of clustering

1.9.5. Overall reports for analysis of cost reduction methods

While working with the cost reduction analysis wizard, you can choose an Overall Test. Figure 18 shows exactly where you can find this option.

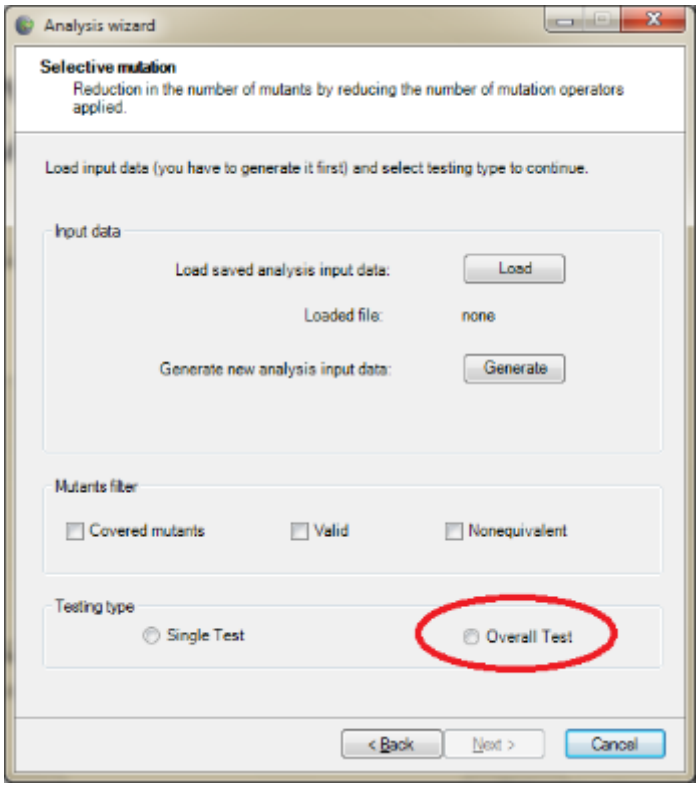

Figure 18. Selecting Overall Test in the analysis wizard

This option generates the report in .xlsx file. (you can open it using the Microsoft Excel 2007 or higher).The structure of the generated reports is very similar, regardless of the cost reduction method in use.A report consists of:

1. A sheet with general information about the tested project and the values of the parameters used in the calculation of analysis quality
2. Sheets with numerical data
3. Sheets with visual charts of collected data

The following chapters describe in detail the contents of each sheet.

1.9.5.1. Sheet with general information about the tested project and the values of the parameters used in the calculation of analysis quality

This sheet is independent of the cost reduction method in use. It contains the following information (Fig. 19):

1) Project name
2) Date of report generation
3) The total number of test cases provided in the analyzed project
4) The number of all mutants generated for the analyzed project
5) The mutation result value calculated for all mutants with the use of all test cases
6) All mutants creation time (with compilation)
7) All mutants creation time (without compilation)
8) All test cases execution time
9) The weighting factors used for calculating analysis quality

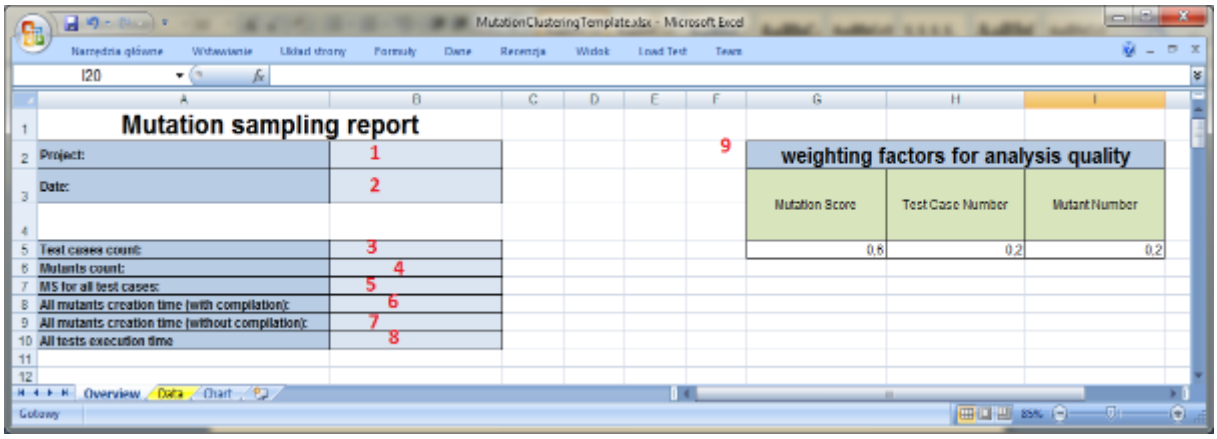

Figure 19. Overall Report sheet with general information

1.9.5.2. Sheet with numerical data

A sheet with numerical data consists of two parts.The initial columns depend on the current analysis.The following columns are independent of the prior choices.

The columns with data specific to selective mutations are shown in Figure 20.Their meaning is as follows:

1) The number of mutation operators ommited or a specific operator name (if there is only one)

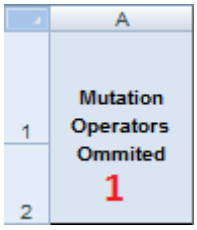

Figure 20. Spreadsheet columns with data specific to the selective mutations

The columns specific to the mutant sampling are shown in Figure 21. Their meaning is as follows:

1) A value that specifies the percentage of mutants that are examined in further analysis

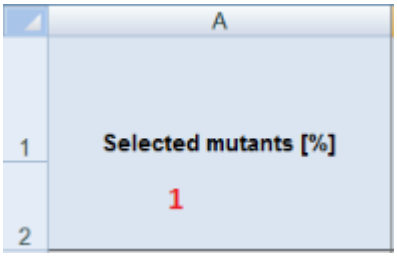

Figure 21 Spreadsheet columns with data specific to mutant sampling

The columns specific to the mutant clustering are shown in Figure 22. Their meaning is as follows:

1) The value of the clustering algorithm parameter (determines the degree of similarity between clusters)
2) The number of clusters formed by the algorithm

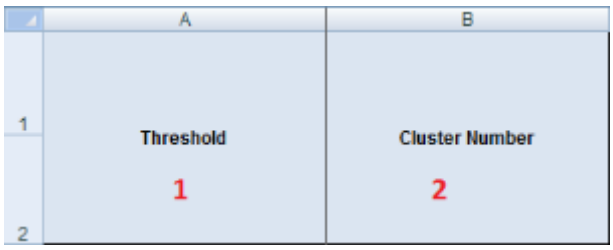

Figure 22. Spreadsheet columns with data specific to the mutant clustering

The spreadsheet columns with data independent of the chosen analysis are shown in Figure 23. Their meaning is as follows:

1) The number of minimum test cases sets for an analyzed subset of mutants.
2) The minimum, average and maximum Mutation Score calculated for the examined subset of mutants with each of the minimum sets of test cases.
3) The minimum, average and maximum test case number for each of the sets of minimum test cases
4) The minimum, average and maximum execution time of all tests for each of the sets of minimum test cases
5) The number of mutants examined
6) Creation time of examined mutants (without compilation)
7) Creation time of examined mutants (with compilation)
8) Values for the ratio of measurement 2, 3, 4, 5, 6, 7 to the original one (calculated for all mutants and the whole set of test cases)
9) Normalized values of selected measures of 8
10) The calculated (with weighting factors from Fig. 19 p.9) value of the quality of the analysis
11) Normalized value of the quality of the analysis 10

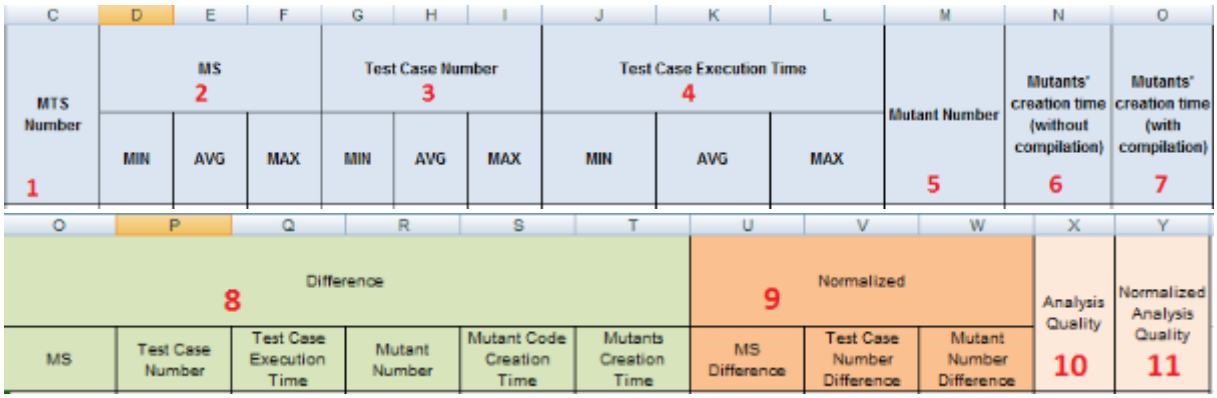

Figure 23. Spreadsheet columns with data independent of the chosen analysis

1.9.5.3. Sheet with charts of collected data

For each of the overall reports there are spreadsheets with charts. They visualize the data collected in the previous sheets. Figure 20 presents a sample chart.

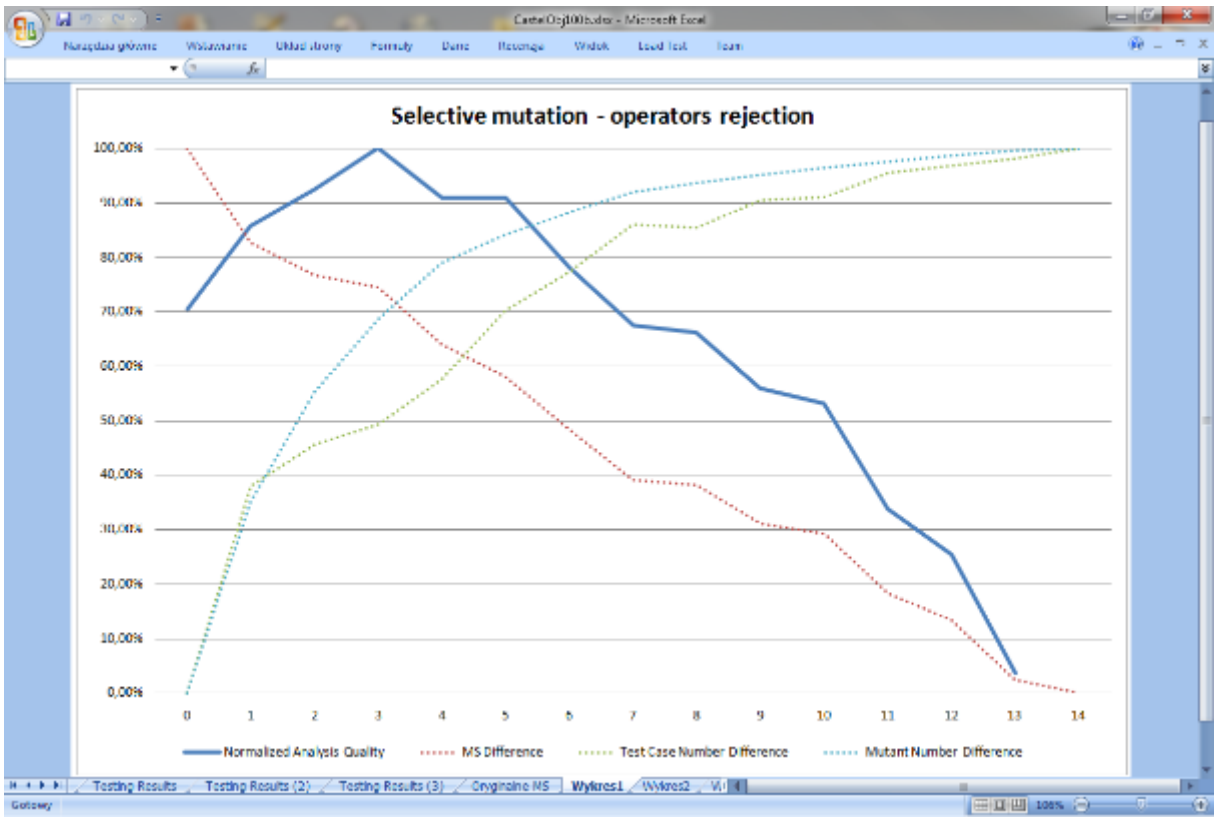

Figure 24. Sample chart from the overall report

1.10. Guidelines for CREAM users

- If you want to generate a lot of mutants, you have to use the SVN repository (not a local drive).
- Use "Mutate only covered code" option to generate only useful mutants.
- Do not generate mutants from unit testing projects.
- Choose correct unit testing tool that you want to use in mutant testing process.
- Long mutation testing process can be split to several smaller. Then you can use *Data Viewer* tool to merge previous generated data.
- If you want to use .coverage files to import coverage information, first you have to generate them on your own computer.

[1] . NET Framework v.4.0., http://www.microsoft.com/downloads/en/details.aspx?FamilyID=9cfb2d51-5ff4-4491-b0e5-b386f32c0992&displaylang=en , 2010-11-23

[2] NUnit, http:// www.nunit.org , 2010-10-09

[3] NCover, www.ncover.com , 2010-11-23

[4] K. Sarba - "Automating the process of testing programs and statistical analysis", MA thesis, Warsaw University of Technology, Institute of Computer Science, 2007

[5] SVN http://subversion.tigris.org/ , 2011-01-27

[6] CREAM Website, http://galera.ii.pw.edu.pl/~adr/CREAM

[7] AJ Offutt, G. Rothermel, and C. Zapf, "An Experimental Evaluation of Selective Mutation," in *Proceedings of the 15th International Conference on Software Engineering (ICSE'93).* Baltimore, Maryland: IEEE Computer Society Press, May 1993, pp. 100-107.