

**Wi!Mi 2.1**  
**User manual**  
Edition 1.0

Moscow 2015

## Contents

<b>Introduction.....</b>	<b>3</b>
<b>2 Application area.....</b>	<b>3</b>
2.1 Brief capability description... ..	3
2.2 User competence level.....	3
2.3 Set of operation documentation.....	3
3. The purpose and application conditions.....	3
3.1 The hardware requirements.....	3
3.2 The software requirements.....	3
<b>4 Getting Started.....</b>	<b>4</b>
4.1 The structure and the content of the software package.....	4
4.2 The launch of the system.....	4
4.3 The setting.....	4
4.4. The steps of model development and modeling.....	4
<b>5 Work with the program.....</b>	<b>4</b>
5.1 Model development.....	4
<b>5.2 Objects .....</b>	<b>6</b>
5.2.1 Classes .....	6
5.2.2 Parameters .....	7
<b>5.3 Relations .....</b>	<b>9</b>
5.3.1 Rules .....	14
5.3.2 Constraints.....	16
<b>5.4 Model testing.....</b>	<b>18</b>
<b>5.5. Visual representation .....</b>	<b>21</b>
5.5.1 Solution graph .....	21
5.6 Program operation description and application area.....	22
<b>6. Possible errors.....</b>	<b>25</b>
<b>7. Guidelines for program use.....</b>	<b>26</b>

## **1. Introduction**

The software program provides the performance of the following functions:

- The development and editing of qualitative situation models/subject domains;
- Structural analysis of the models, acquisition of solution logical inference and its interpreting in the form of a series of implemented operations.

## **2. Application area**

### **2.1. Brief capability description**

The program provides the performance of the following functions:

1. The development and editing of subject domain model description in the form of an array/a list and a graph:
  - creating and editing objects of the subject domain (parameters and classes);
  - creating and editing relations and rules connecting these objects.
2. Structural analysis including the analysis of correctness and completeness of the entry data.
3. Data analysis.
4. The development and output of the acquired logical inference algorithm of situation resolution, the calculation of the required values.

### **2.2. User competence level**

To work with the program it is sufficient to have knowledge about the subject domain for which the model is developed. To develop a complex relation (not often used) in JavaScript the user should have programming skills.

### **2.3. Set of operation documentation**

- User manual;
- Examples of the models;

## **3. The purpose and application conditions**

### **3.1. The hardware requirements**

The program can be installed on a personal computer with processor 2 GHz or faster, with at least 4GB of RAM and at least 60 MB of free storage space on the hard drive.

### 3.2 The software requirements

The program works with the following operating systems: Microsoft Windows 7, Microsoft Windows 8, Mac OS 10.9 and higher (not included in the installation package).

To work with the help, Adobe reader is required (not included in the installation package) or another program allowing PDF files to be read.

## 4. Getting Started

### 4.1 The structure and the content of the software package.

*In case you use the software package on operating systems Windows.*

The situation modeling system is supplied in the form of a file "WiMi\_[revision].exe", running installation procedure of the program, user manual and description techniques of subject areas.

*In case you use the software package on operating systems Mac OS.*

The situation modeling system is supplied in the form of a file " WiMi\_[revision].dmg", running installation procedure of the program, user manual and description techniques of subject areas.

### 4.2 The launch of the system

The launch of the system is implemented by the executing file WiMi.app or WiMi.exe. For user convenience the icon for the indicated file can be created on the desktop.

### 4.3 The setting

All the models developed in the program are stored in the .xml. file extension.

When downloading the program, the application opens that is ready to start working. To download the existing model you need to select "Open" in the menu "File", select the corresponding .xml file and click on«OK» button. It is possible to open only one model in the same program simultaneously.

### 4.4 The steps of model development and modeling.

The model development and modeling is implemented in several steps:

1. The development and parameterization of the model:
  - selection of objects (see 5.2)

2. The development and binding of the model:

- assigning relations (see 5.3);
- assigning rules (see 5.3.1);
- assigning correctness conditions for input data (constraints) (see 5.3.2).

3. Solution of tasks, text description of the inferences with the indication of the rule used, acquired or found data.

## 5. Work with the program

### 5.1 Model development

After opening the application, a new model is created automatically. If it was closed, the user can create a new model. To create a model the user should select “File” in the menu bar and select “Create project” from the File menu (fig. 5.1). If there are unsaved data before creating a new model, a dialogue box will inform you about it.

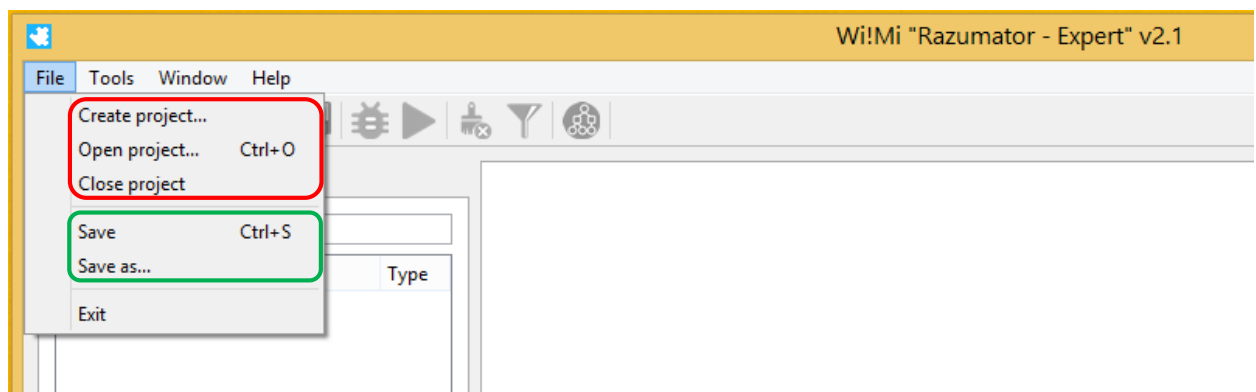


Figure 5.1. – File menu

The user can continue working with the existing model as well. To do this, you should select “Open project” from the “File” menu. Select the necessary model in the dialog box that appears (fig. 5.2). Files of the model are stored in XML format.

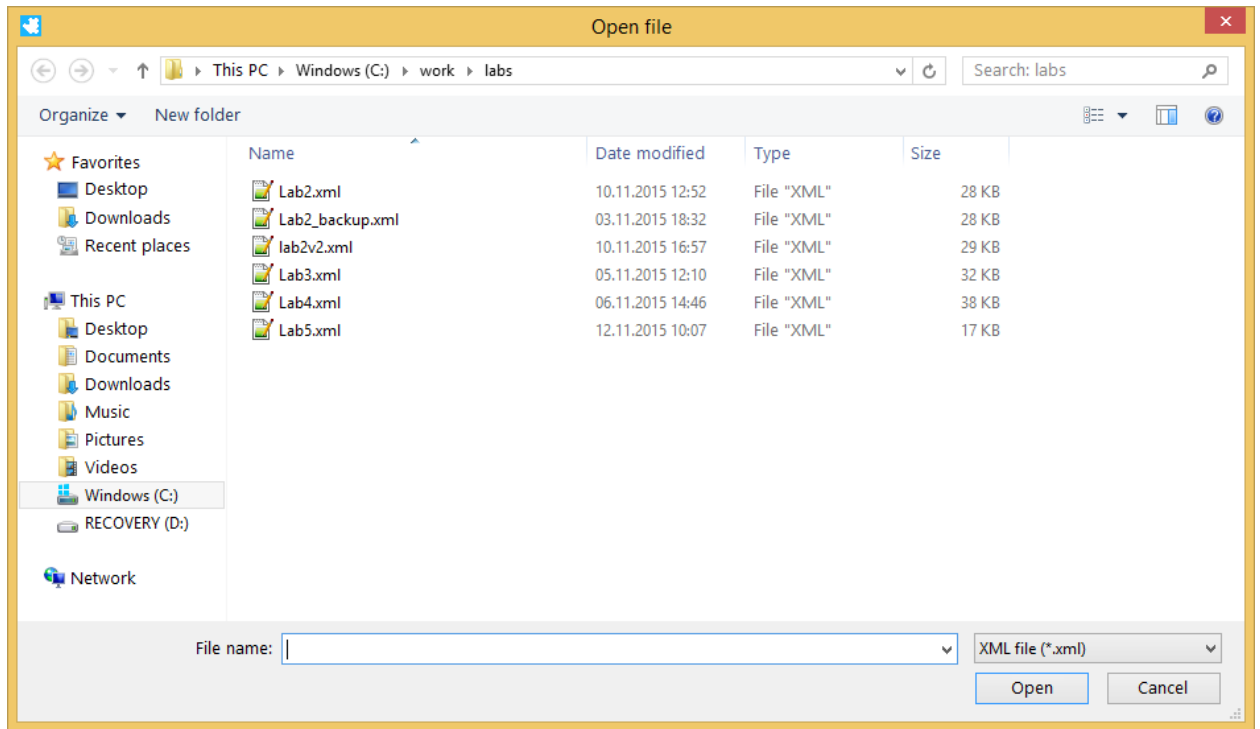


Figure 5.2 – Model opening dialogue box

To close the current model without closing the application, click on the “Close project” button from the “File” menu.

If it is necessary to save changes made in the current model, select “Save” from the “File” menu to save changes in the active tab or the option “Save all” to save changes in all the open tabs to the current file of the model or “Save as” if it is necessary to save the model to a new file.

The operations described above can be executed by clicking on the corresponding buttons on the toolbar (fig. 5.3).

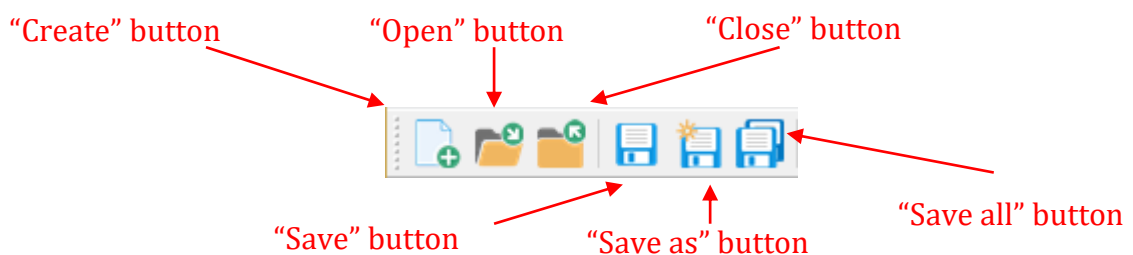


Fig.5.3 Toolbar buttons for model management

After downloading the model, all the objects are displayed in the form of a tree list on the left side of the application and all the relations are displayed on the right side. Editing corresponding elements of the model is implemented in the tabs located in the center of the application. If errors

detected in the process of developing the model, corresponding records are displayed in the list at the bottom of the application (fig. 5.4).

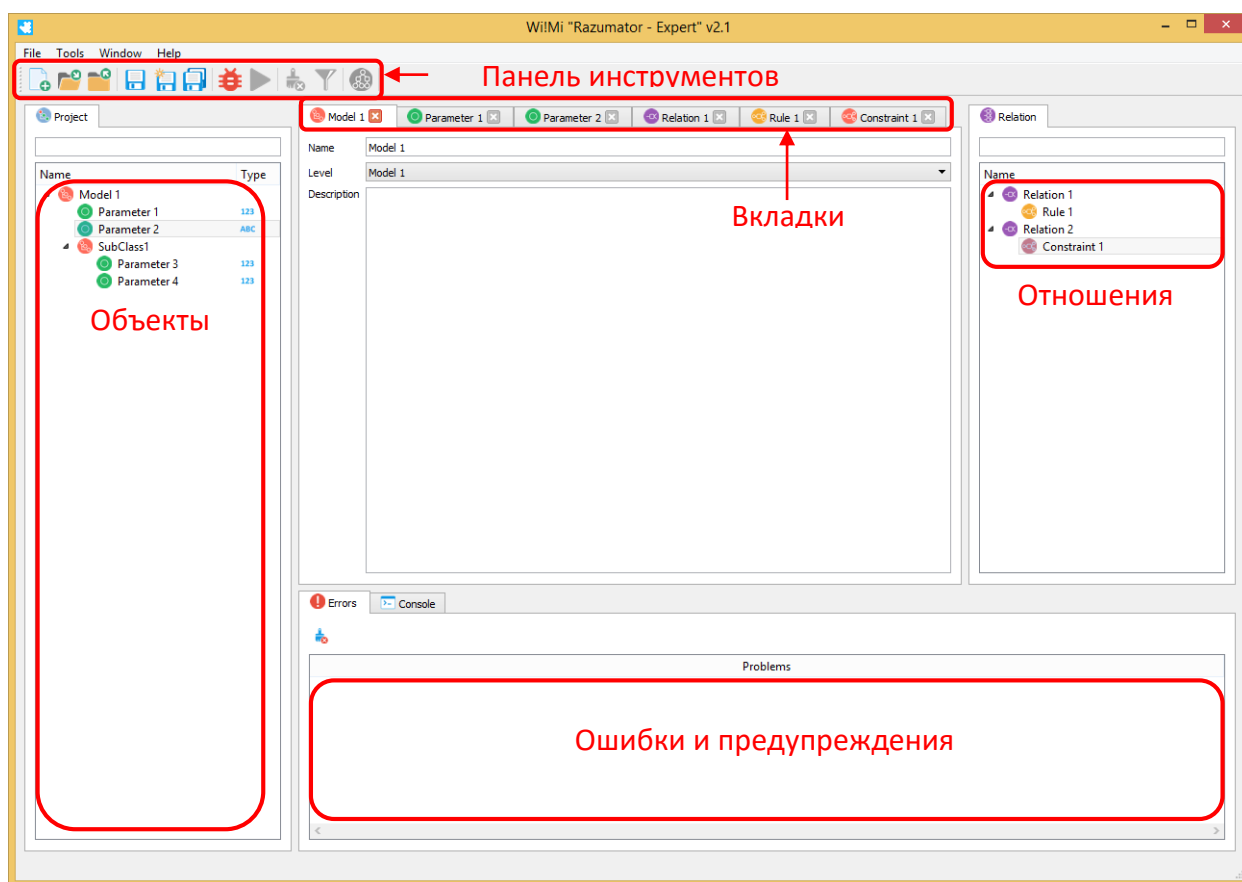


Fig. 5.4 – The main application window

## 5.2. Objects

There are two types of objects: classes and parameters.

### 5.2.1. Classes

Class is an abstract entity, collective concept. Class can contain parameters and other classes. Class has its name, hierarchy level and description. There must be at least one class in any model. The class with the highest hierarchy level is called root class. When developing a new model, a root class is created by default.

To create, change or delete the class you should use the context menu (fig.5.5) called up by clicking on the right mouse button on the corresponding class from the tree list on the left side of the screen.

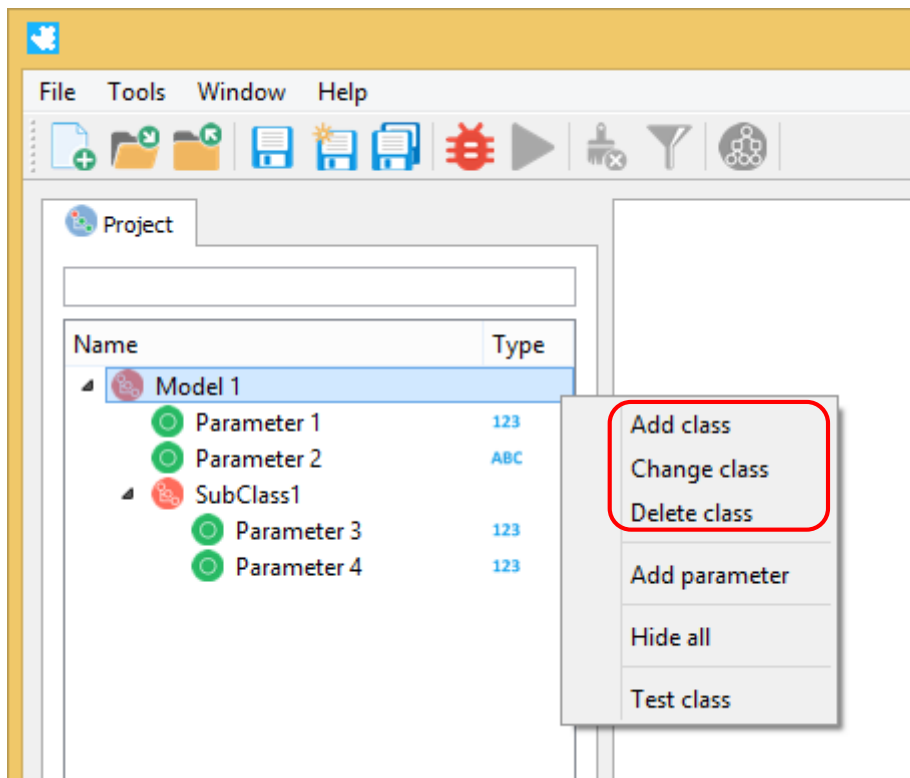


Figure 5.5 – Context menu called up by clicking on the class

After clicking on “Add class” or “Change class” buttons, a new tab opens in the center of the window allowing you to implement appropriate operations.

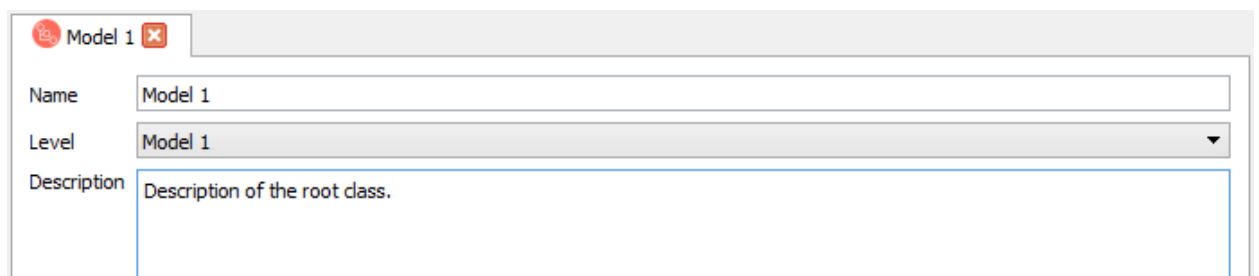


Figure 5.6 – The tab designed to change the class

The user can change the name, the hierarchy level selecting a new parent class from the drop-down list as well as add description in the tab that appears (figure 5.6).

To apply changes or finish the process of designing a new class, save the model. A new or changed class is displayed in the list on the left side of the screen.

### 5.2.2. Parameters

Parameter is an object containing the value of a particular type: numerical value or text value. Similar to the class, parameter has its name, hierarchy level and description. Moreover, parameter can have some default value.

To create a parameter you should choose the class, inside of which it will be created, bring up the context menu by right-clicking on this class and select “Add parameter” (figure 5.7). To



change or delete the existing parameter you should use the context menu called up by right-clicking on a particular parameter from a tree list on the left side of the window (figure 5.8).

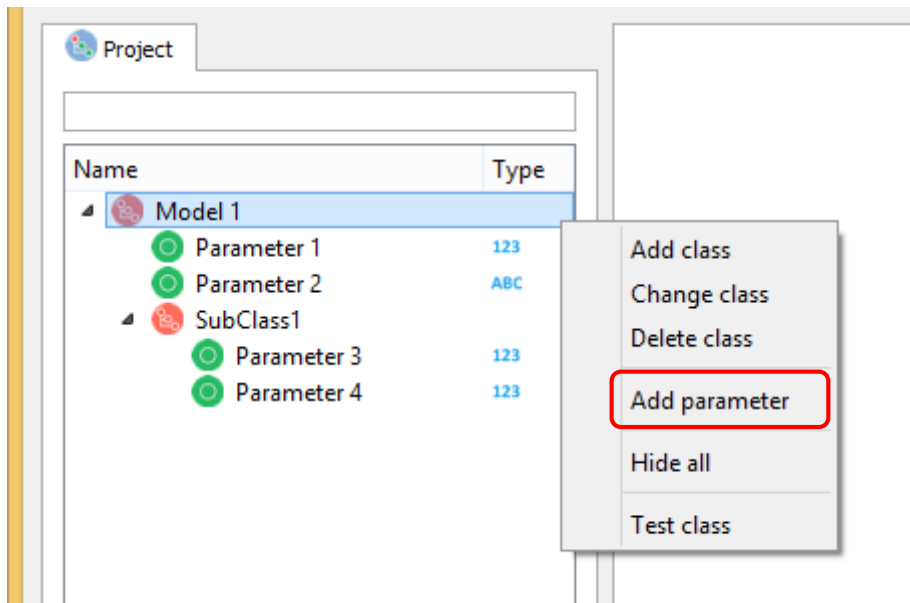


Figure 5.7 – Context menu called up by right-clicking on the class

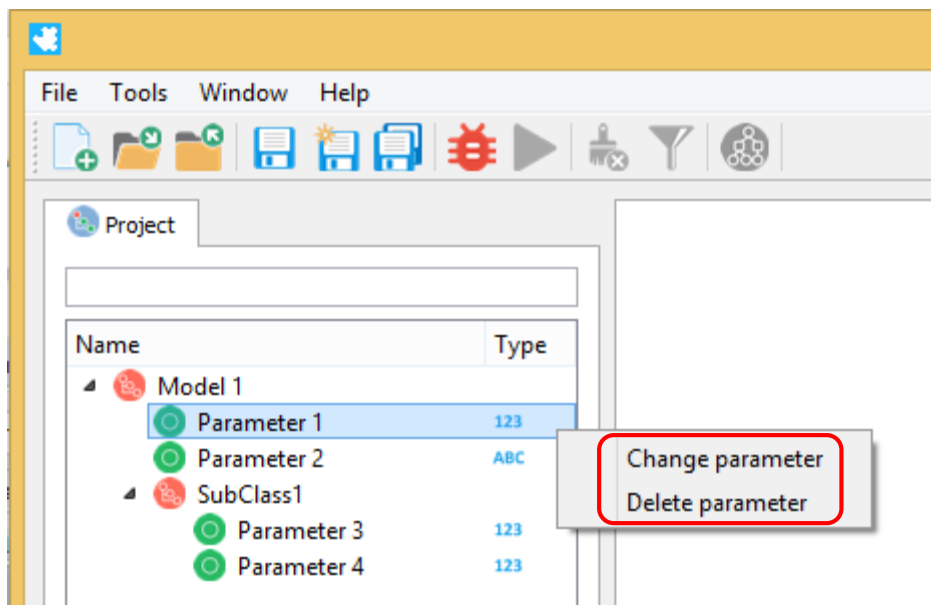


Figure 5.8 – Context menu called up by right-clicking on the parameter

After clicking on “Add parameter” or “Change parameter” buttons, a new tab opens in the center of the window allowing you to change parameter (figure 5.9).

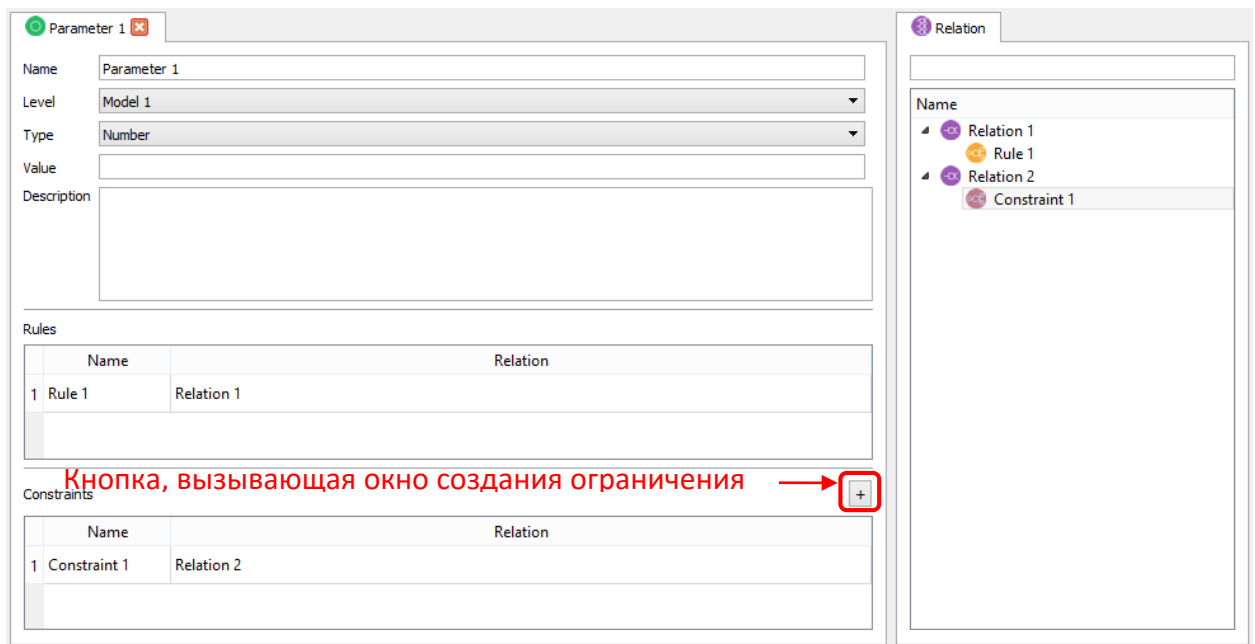


Figure 5.9 – The tab designed to change parameter

In the appearing tab the user can change parameter name, hierarchy level and data type. In addition to required properties, the user can add default value and description as well. Moreover, in this tab there are two tables representing rules connected with a particular parameter and constraints. Each table consists of two columns: the first one displays the name, the second one displays relation, which the corresponding rule or constraint is connected with. To apply changes or finish the process of developing a new parameter, save the model. A new or changed parameter is displayed in the list on the left side of the window.

### 5.3. Relations

We will define the following concepts:

- 1) Connection – is association between objects significant for the considered subject area. Random amount of input objects is converted in a single output object or a set of output objects by means of connection.
- 2) Relation – is a type of connection using abstract variables and describing their interaction.
- 3) Rule - is type of connection binding relation with certain objects.
- 4) Constraint– is a type of rule controlling the correctness of input data. For example, the side of a triangle can't be negative.

To add, change or delete a relation, call up the context menu by clicking on the corresponding relation from the list on the right side of the application window and select the corresponding option (figure 5.10).

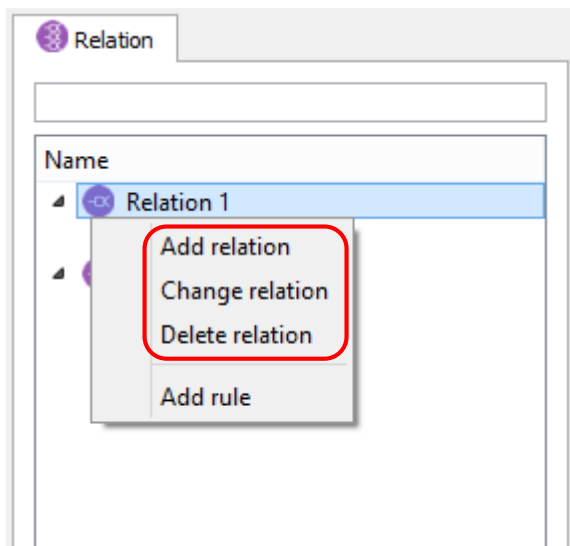


Figure 5.10 – Context menu called up by clicking on the relation.

After clicking on the “Add relation” and “Change relation” buttons a new tab opens in the center of the window allowing you to execute appropriate operations (figure 5.11).

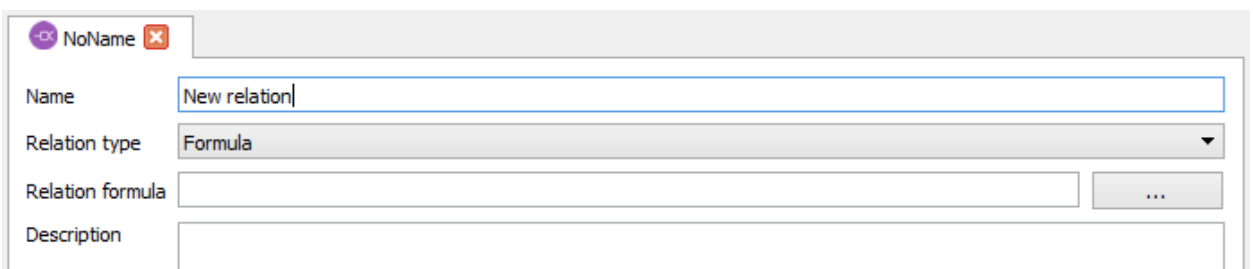


Figure 5.11 – The tab designed to change relations

In the tab that appears the user can change the name, choose the type and assign the appropriate relation body. Moreover, you can add a description to the relation.

There are four types of relations:

- 1) formula – mathematical formula consisting of variables and operators;
- 2) conditional relation – production relation “if..., then...else...”;
- 3) constraint – production relation imposing constraint on parameter value;
- 4) complex relation – programming relation for an advanced user. Relation body corresponds to each relation type. To change relation body you should click on “...” button next to the corresponding field.

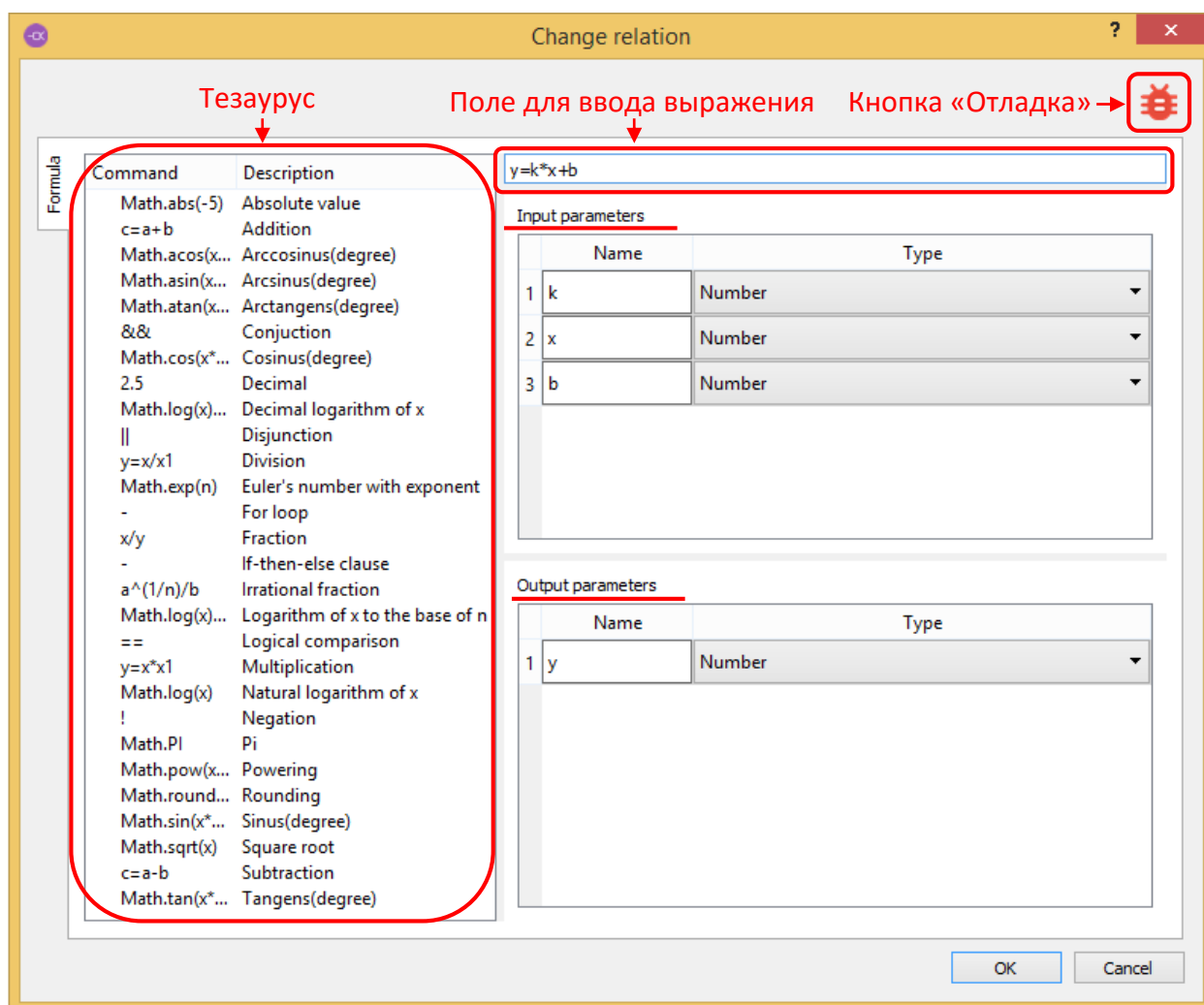


Figure 5.12 – Dialog box “Change relation” for relation type “Formula”

When choosing any type of relation, examples of the correct thesaurus entry are displayed on the left side of the window. You can choose a relation from there, clicking on it two times or create a new one. The rules of creating relations:

- 1) Relation is written in a way that output variables are on the left hand side of the formula and input variables are on the right hand side;
- 2) it is possible to use any Latin letters and operator symbols in the relations;
- 3) “string” type variables acquire values in the following way:  $y = \text{“String Value”}$ ;
- 4) variables in different relations can be identical and abstract;
- 5) attention! There can’t be the same input and output variables in the relation. For example, the type of the relation “ $y = y + 1$ ” is not allowed!

After creating a relation, it should be analyzed clicking on the appropriate button at the top of the dialog box. If relation body is correct, then the program will identify input and output variables of the relation in the corresponding tables of the dialogue. Numerical type is assigned to the

variables by default. It can be changed to the text type if necessary. To finish creating the relation, click on the “OK” button. The new relation is displayed in the tree list on the right side of the application window.

Dialog box “Change relation” has different form depending on the chosen relation type indicated on the left side of the window (figure 5.12).

**IF-clause relation**

Command	Description
Math.abs(-5)	Absolute value
c=a+b	Addition
Math.acos(x...	Arccosinus(degree)
Math.asin(x...	Arcsinus(degree)
Math.atan(x...	Arctangens(degree)
&&	Conjunction
Math.cos(x...	Cosinus(degree)
2.5	Decimal
Math.log(x)...	Decimal logarithm of x
	Disjunction
y=x/x1	Division
Math.exp(n)	Euler's number with exponent
-	For loop
x/y	Fraction
-	If-then-else clause
a^(1/n)/b	Irrational fraction
Math.log(x)...	Logarithm of x to the base of n
==	Logical comparison
y=x*x1	Multiplication
Math.log(x)	Natural logarithm of x
!	Negation
Math.PI	Pi
Math.pow(x...	Powering
Math.round...	Rounding
Math.sin(x*...	Sinus(degree)
Math.sqrt(x)	Square root
c=a-b	Subtraction
Math.tan(x*...	Tangens(degree)

**IF**

**THEN**

**ELSE**

**Input parameters**

	Name	Type
1	a	Number

**Output parameters**

	Name	Type
1	y	Number

OK Cancel

Figure 5.13 - The change of the relation type “Conditional relation”

If you chose conditional relation type (figure 5.13), three fields are used to set it up. Condition of relation implementation is entered in the “IF” field, in the fields “THEN” and “ELSE” expressions are entered executed if the condition is performed or not performed correspondingly. Analogous fields are used to set up relation body of the “Constraint” type (figure 5.14). To set up the constraint, it is necessary and sufficient to indicate the condition of constraint implementation in the field “IF”.

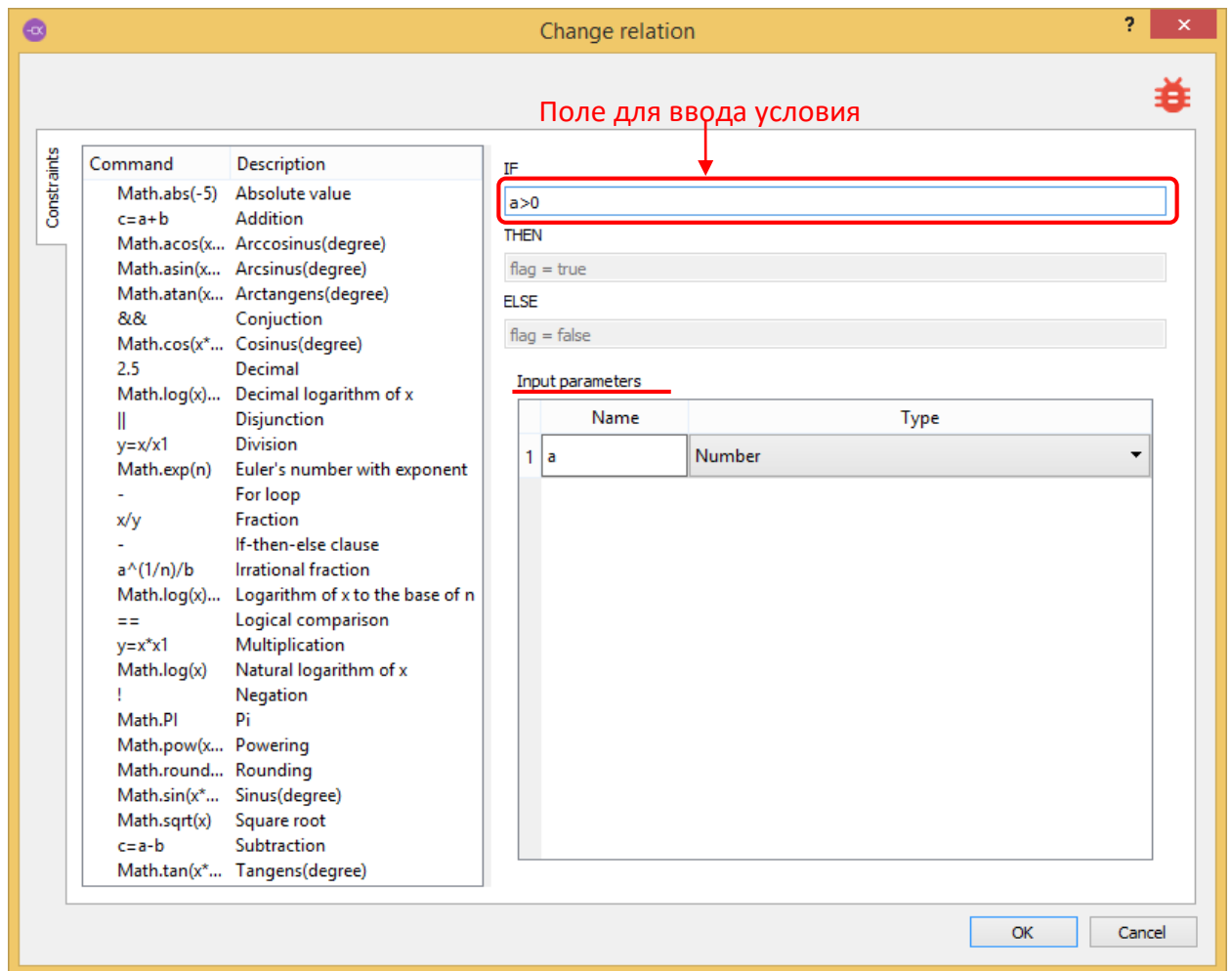


Figure 5.14 - The change of relation type “Constraint”

To create a complex relation, programming skills are required. To create a complex relation (figure 5.15) the user should write JavaScript code in the field available, after that perform the analysis of the expression clicking on the appropriate button. If no errors are detected, you should add input and output parameters clicking on the appropriate button. If unnecessary parameters were created, they can be deleted. To delete parameters select any field of table row related to the parameter being deleted and click on the appropriate button located above the table. Input parameters are described in the table above, output parameters are described in the table below. To define the parameter, enter corresponding name of the variable from the code in the field “Name”.

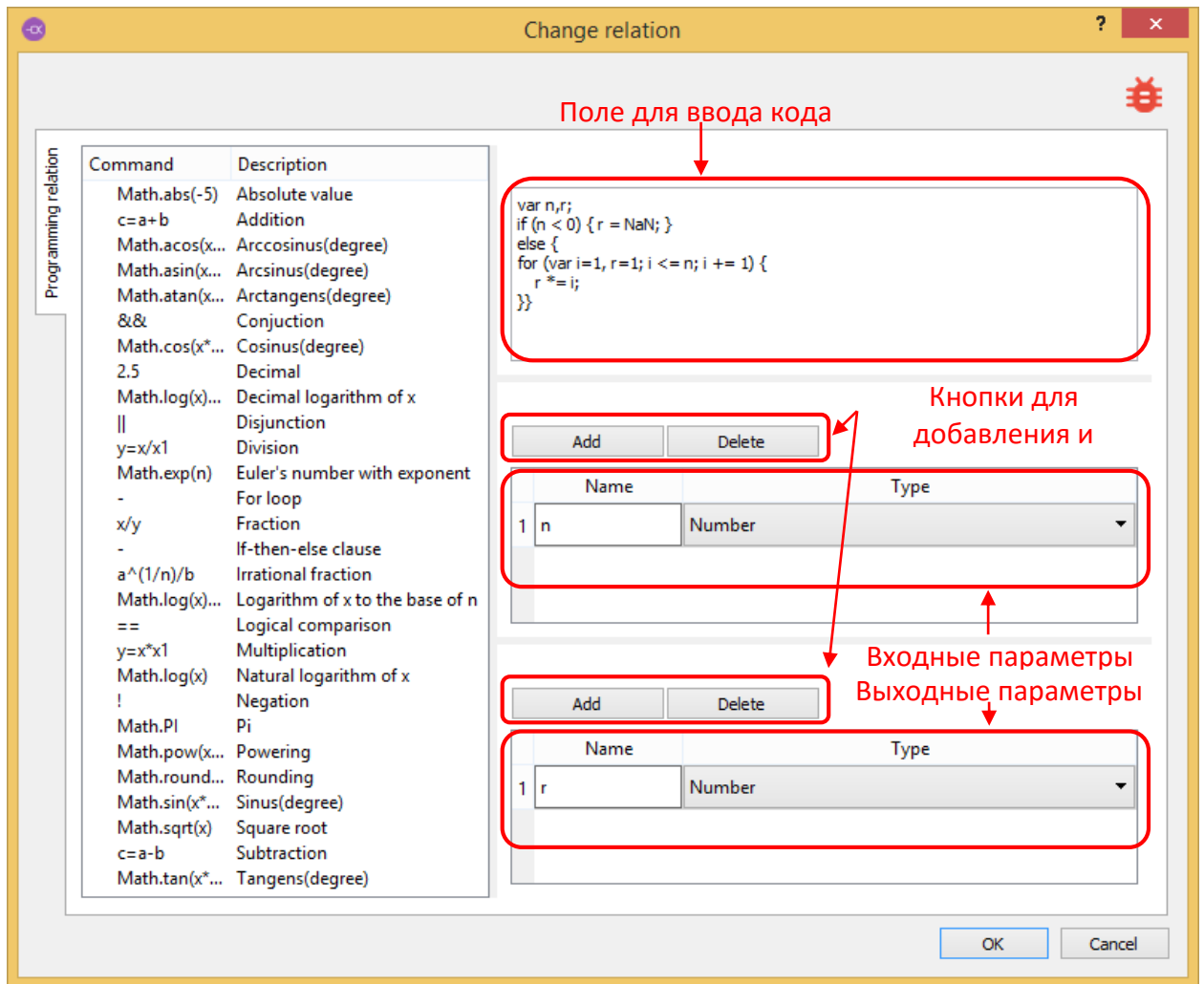


Figure 5.15 –Design of complex relation on the example of factorial function

Created relations open up an opportunity for developing rules and constrains based on them.

### 5.3.1. Rules

To create a rule for a given relation you should right-click on this relation from the list on the right side of the application window and choose “Add rule” (figure 5.16). Note: a rule can be created only if the relation has the type “formula”, “conditional relation” or “complex relation”.

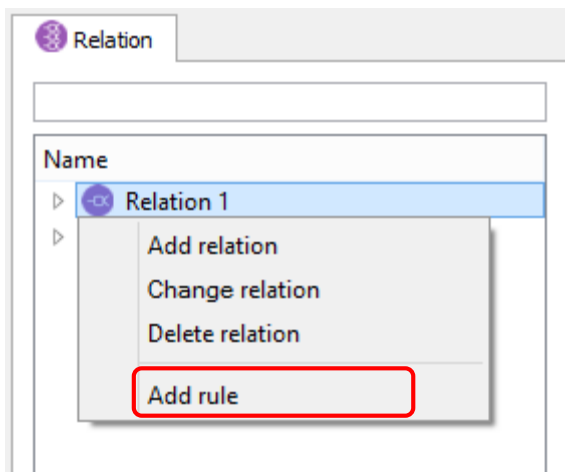


Figure 5.16 – Context menu called up to add a rule

Relation: fact

Name: factorial

Description: The rule for calculating factorial.

Input parameters

	Name	Parameter	
1	n		... [Reset]

Output parameters

	Name	Parameter	
1	r		... [Reset]

Кнопка для выбора параметра  
Кнопка для сброса параметра

Figure 5.17 – The tab designed to change the rule

In the appearing tab (figure 5.17) the user can change the name, add description and choose parameters for which the rule is true. To define parameters click on “...” button in the corresponding row of the table. In the dialog box that appears select a parameter from the list of existing parameters and click on “OK” button. Using the field above the list, you can search for parameters by name. If parameter is chosen incorrectly, you can clear the field of the table clicking on the appropriate button.



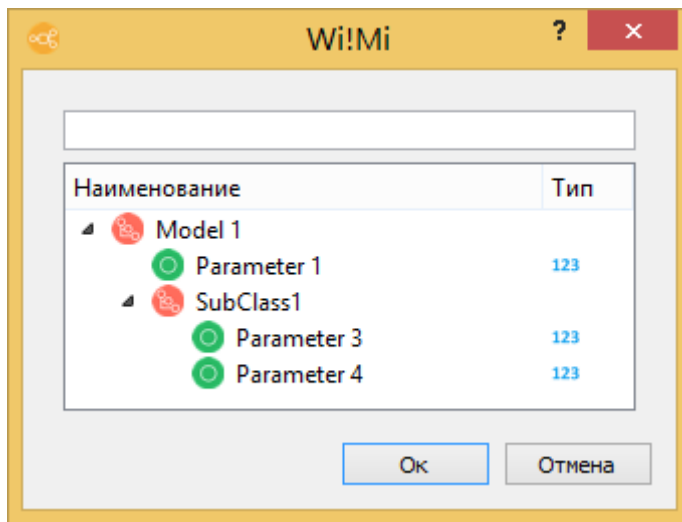


Figure 5.18 – Dialog box for choosing parameters

After filling in the necessary fields, save the model. The new rule is displayed in the list of relation's child elements, which the rule is based on at the right side of the application. To change or delete the existing rule call up the context menu by right-clicking (figure 5.19) and select the action you need.

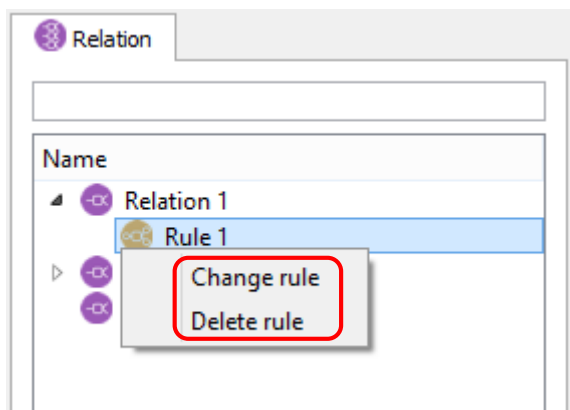


Figure 5.19 –Context menu called up by clicking on the rule

### 5.3.2. Constraints

Constraint can be created only on the basis of relation of corresponding type. To create constraint you should right-click on the relation from the list on the right side of the application window and choose “Add constraint” (figure 5.20).

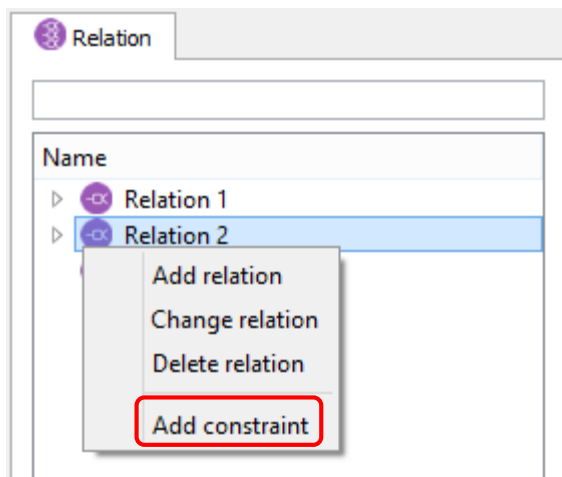


Figure 5.20 – Context menu, called up by clicking on the relation with the type “constraint”

	Name	Parameter		
1	a		...	

Figure 5.21 – The tab designed to change constraint

In the appearing tab (figure 5.21) the user can change the name, add description and choose parameters for which the constraint is true. The process of choosing parameters is analogous to corresponding procedure for rules.

After filling in the necessary fields, save the model. The new constraint is displayed in the list of relation’s child elements, which the rule is based on at the right side of the application window.

To change or delete the existing constraint call up the context menu (figure 5.22) by right-clicking and select the option you need.

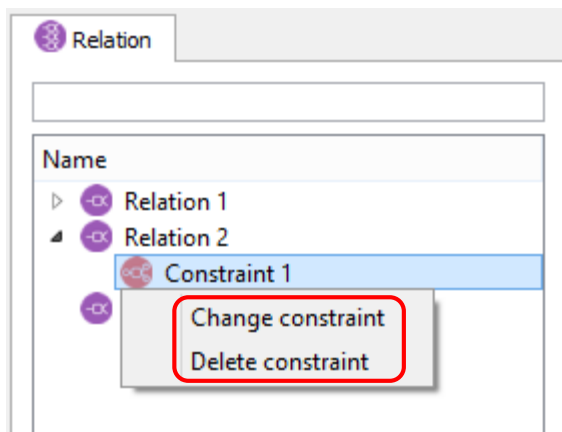


Figure 5.22 – Context menu called up by clicking on the constraint

You can create constraint for a particular parameter clicking on “+” button in the parameter change tab (figure 5.9). In the appearing dialog box (figure 5.23) you should specify constraint conditions. It is possible to add several conditions. Moreover, you can specify between conditions (“logical AND” or “logical OR”).

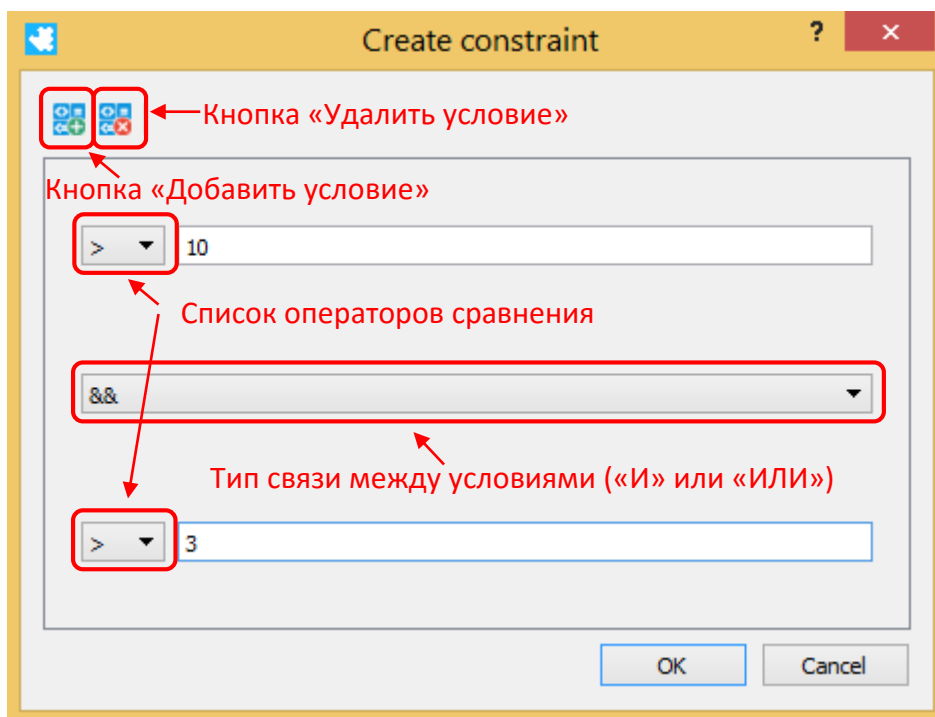


Figure 5.23 – Dialog box “Create constraint”

Constraints are used for data verification. When testing the model (see para. 5.4) if any constraint is not executed the user gets the appropriate message in the console, and further calculation is stopped.

## 5.4. Model testing

Model testing is designed to infer logical path from input data to output data, verify data correctness and calculate required values. To implement testing, right-click on the corresponding class of the model and select “Test model” in the context menu. You can also open this tab using the menu “Tools” (figure 5.24).

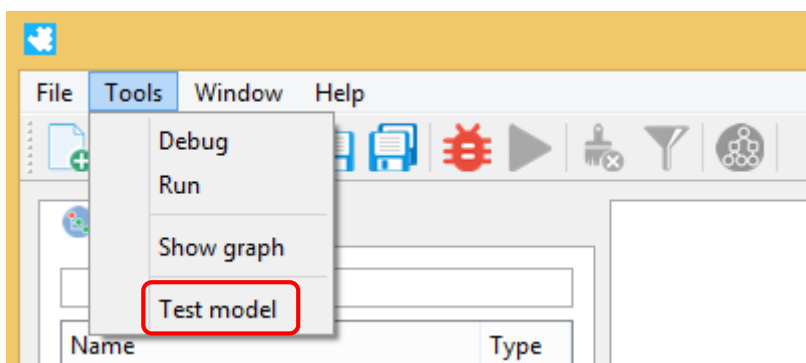


Figure 5.24 – Menu “Tools”

In the displayed tab there is a table that contains all the parameters belonging to the chosen class and its children. Each parameter has a field for entering input data (default values) and checkbox “Find”.

Before testing, it is appropriate to check the model for errors. To do this click on the “Errors” button on the toolbar. You can also implement this operation selecting “Debug” from the menu “Tools”. If problems are detected in the model, appropriate messages are displayed in the tab “Errors” (figure 5.25). There are two types of messages: error messages and warning messages. If an error occurs, saving and testing the model is impossible until the problem is corrected. When warning message comes up the model is working, however, it is recommended to correct problems that the warning message indicates to enhance model stability and reduce the probability of new error occurrence in the process of further working with the model.

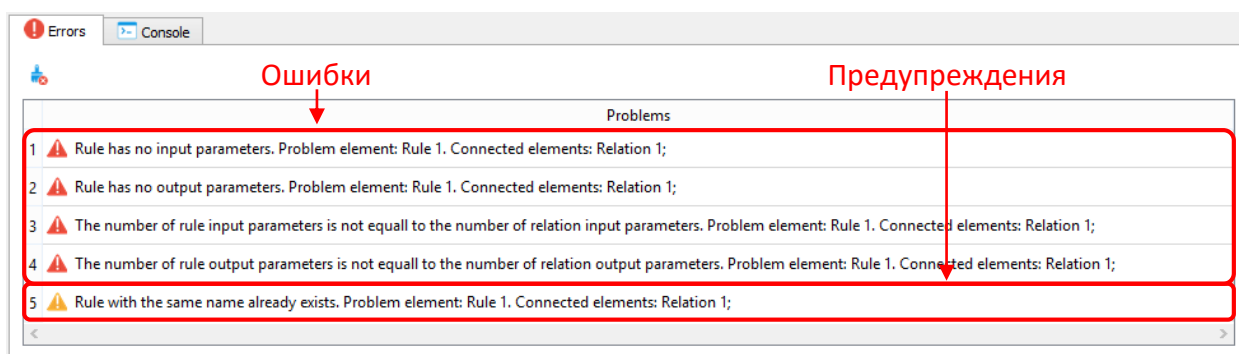


Figure 5.25 – The tab “Errors”

To test the model (figure 5.26) you need to enter required input data and select checkboxes for parameters that are necessary to calculate. After that you should click on the “Run” button on the toolbar. You can also execute this operation selecting “Run” from the menu “Tools”. If successful, calculated parameters will be highlighted in red, all the current information will be

displayed in the console: triggered constraints (figure 5.27), found and not found values and logical inference with the indication of the rule used, its description, input and output values as well as found values.

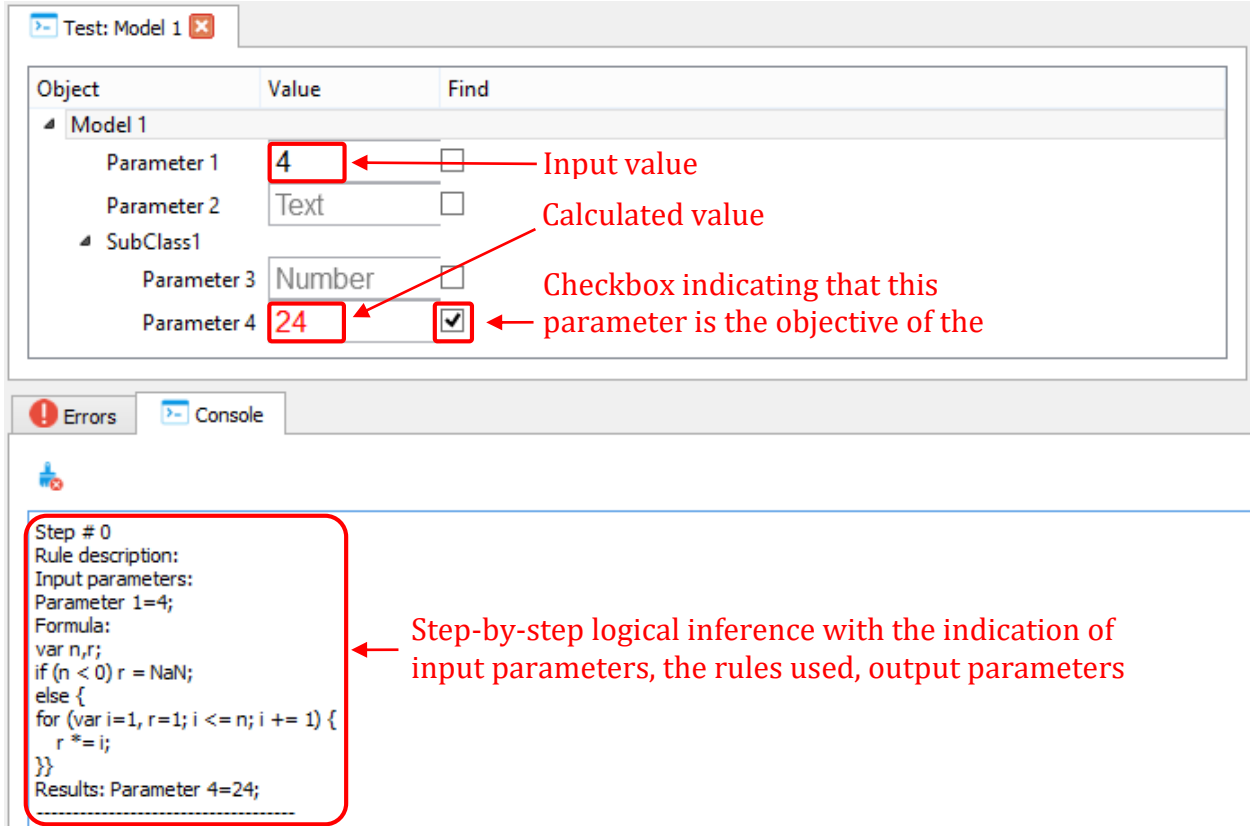


Figure 5.26 - Model testing

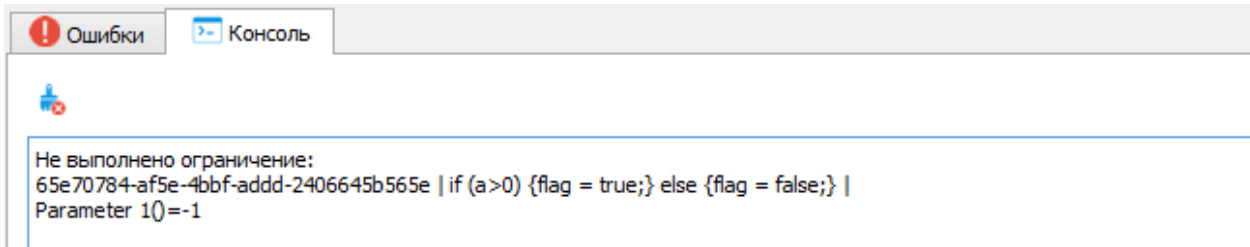


Figure 5.27 – Console when constrain isn't executed

Before every new testing, clear the results of previous tests!

Otherwise, new calculation will not be implemented and the application will give a message that the path is not found when searching for set parameter. You can clear the results of tests manually or you can use the appropriate button on the toolbar.

After performing testing successfully, you can disable the display of input data. To do this, use the button “Show results only” on the toolbar (figure 5.28).

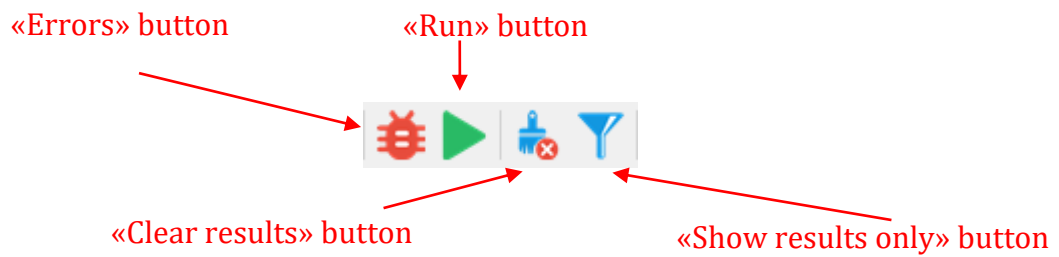


Figure 5.28 Toolbar buttons relating to model testing

### 5.5.1. Visual representation

Mivar model is visually represented in graph form.

#### 5.5.1.1. Solution graph

To show solution graph (figure 5.30), you should perform model testing. If testing is successful open the menu “Tools” and select “Show graph”. You can also do it by clicking on the appropriate button on the toolbar (figure 5.29).



Figure 5.29 – The button “Show graph”

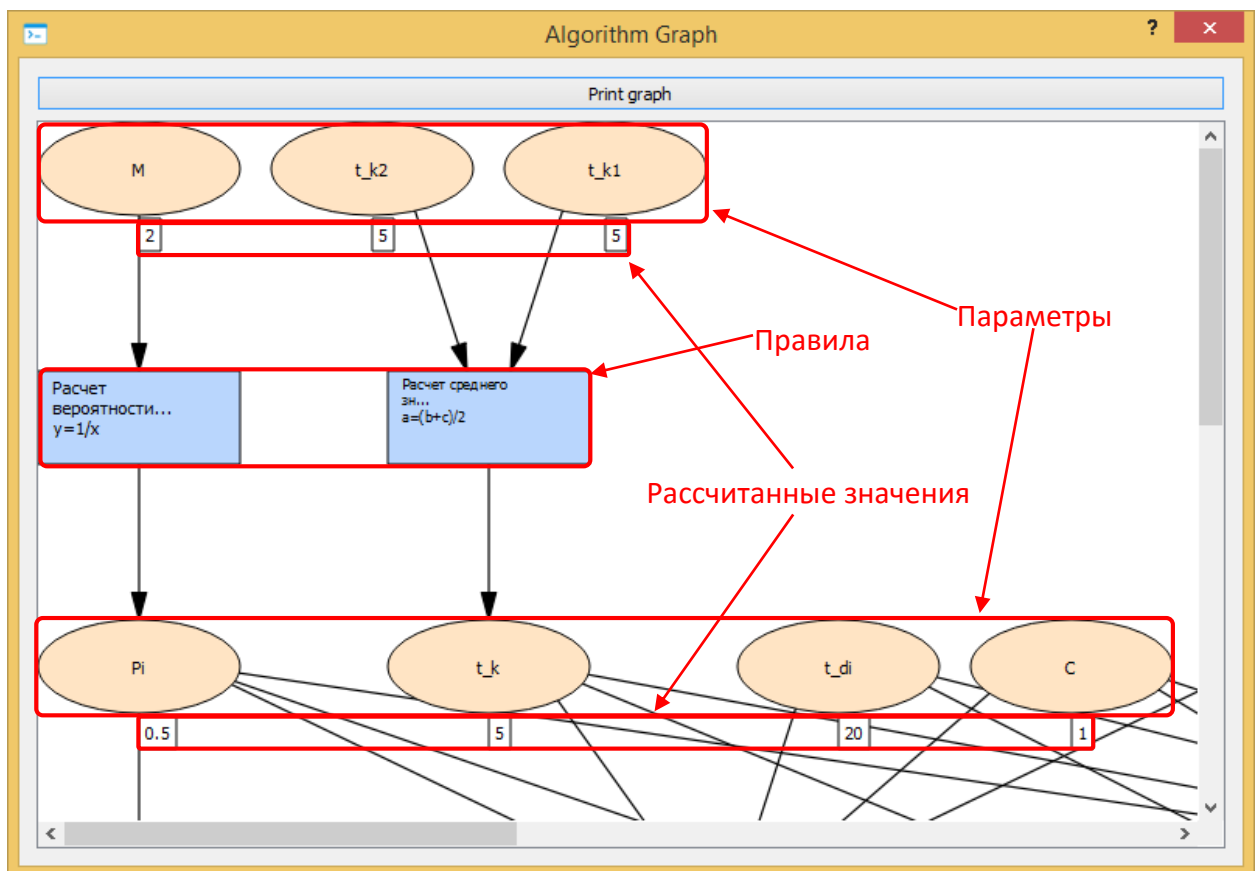


Figure 5.30 – Solution graph

Rules are denoted by rectangles and parameters are denoted by ellipses. When clicking on the rule, all the input and output parameters are displayed which are connected with the rule by green or red arrows correspondingly. Clicking on the parameter displays all the rules and parameters connected with it. Current value is displayed opposite each parameter. You can move within the graph using scroll bars and the mouse. To move using the mouse, move the mouse pointer over empty space on the graph, press the left mouse button and move the pointer holding the button. Graph will move along with the movement of the pointer. If you double-click on a rule, partition of the graph will be displayed consisting of the rule and input and output parameters connected with it (figure 5.31). To return to the whole graph representation you need to double-click on the rule.

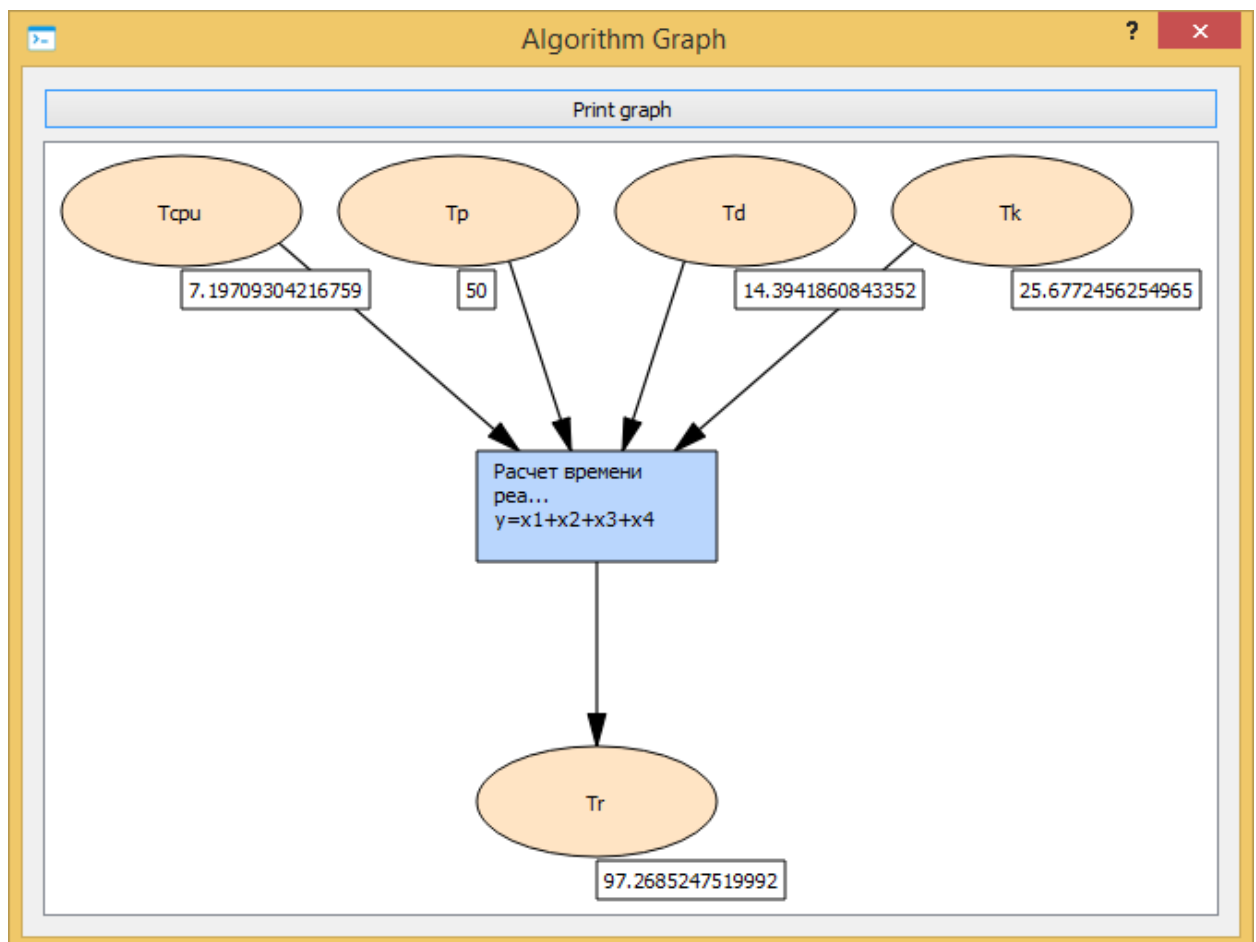


Figure 5.31 Enlarged graph partition consisting of the chosen rule and parameters connected with it

The graph can be printed. To print the graph, open the graph window and click on the “Print graph” button. Choose appropriate parameters in the dialog box that opens. When settings have been completed, click on the “Print” button.

## 5.6. Program operation description and application area.

WiMi is designed to develop specialized information systems capable of solving complex logical tasks associated with processing large amounts of information in real time. WiMi allows us to calculate required parameters from input parameters by finding computation algorithms.

Calculation of all required intermediate parameters is implemented as well.

In computation process WiMi calculates only required parameters connected to algorithmic calculation chains rather than all possible parameters. Such an approach ensures high program performance with little resources used.

Runtime technology embedded in the second version of WiMi considers the values of calculated intermediate parameters when designing task solution algorithm. Thus, if there are several alternative chains, the system's choice is based on the value of calculated parameter (see figure 5.32). Consequently, if rules have several output parameters but only some of them will be calculated, further algorithm can be reconstructed so that only calculated parameters are involved in calculation.

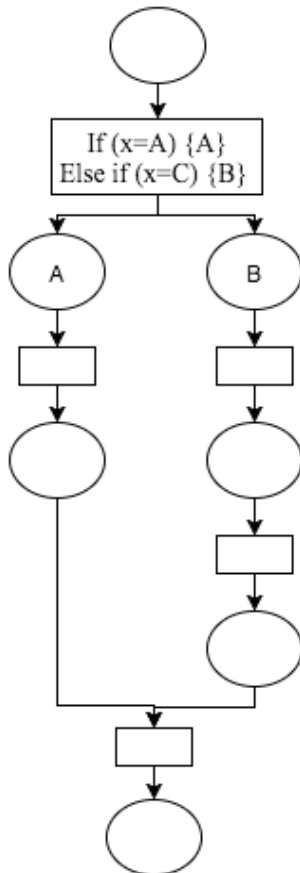
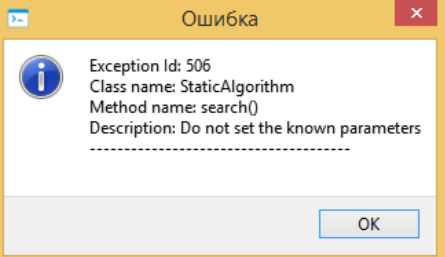
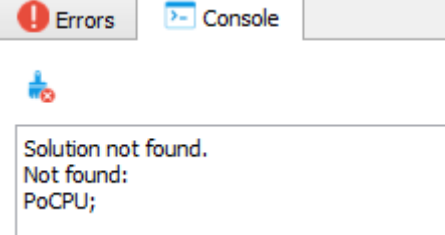
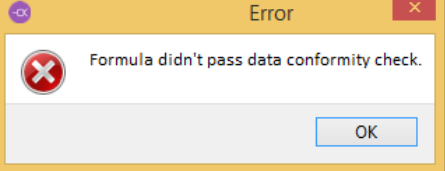
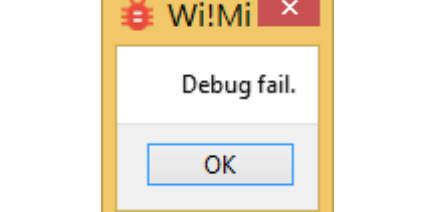
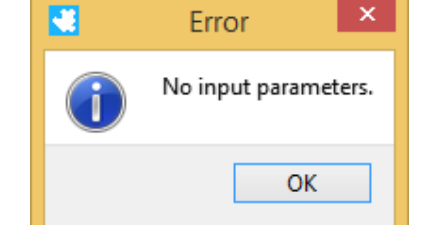
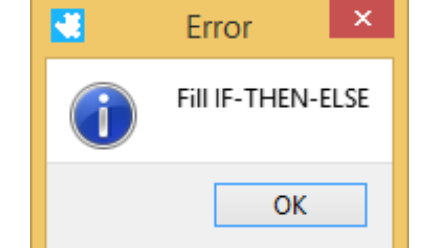


Figure 5.32

Illustration of runtime technology



## 6. Possible errors

Problem description	Troubleshooting steps	Screenshot of an error message
Input parameters are not set	Check if input parameters are set	
The message «Solution not found»	Check the existing rules for errors; clear the results of previous testing	
The message «Data inconsistency in formula relation »	Check the formula using the “Debug” button	
The Message «Debug error »	Check the code for syntactic errors; make sure that all the variables being used are declared- input and output parameters must be declared in the code	
The message «No input/output parameters»	Add input/output parameters corresponding to the relation	
The message «Fill in IF-THEN-ELSE»	Add condition to “IF” block, check expressions in “THEN”, “ELSE”	

## **7. Guidelines for program use**

WiMi training courses have been organized to help you to master the program completely. You can learn more about the courses and register for them on the official site of the company Mivar.