

Ferda FrontEnd - User environment for the Ferda DataMiner datamining system

Martin Ralbovský

[<martin.ralbovsky@gmail.com>](mailto:martin.ralbovsky@gmail.com)

Tomáš Kuchař

Michal Kováč

2006-01-24

Table of Contents

[Dockable working environment](#)

[Box - module from the view of the user](#)

[Parts of the working environment](#)

[Menu](#)

[Toolbar](#)

[Archive](#)

[User note](#)

[Property grid](#)

[New Box](#)

[Context help](#)

[Desktop](#)

Abstract

This document describes the user interface of the data mining system Ferda DataMiner. It defines the parts of working environment and describes how user works with these parts.

Dockable working environment

All the components (parts of the working environment) except of the main application menu and the Toolbar are movable across the working space by the means of the docking mechanism. The program has a preset setting of the components (Archive and New Box on the left, Properties and Context Help on the right). However, user can dock the components freely. The docking mechanism is a standard used mainly in the user environment of the development tools (MS Visual Studio etc.)

Box - module from the view of the user

The project team wanted originally to use the word "module" for a graphical element of the user environment. However the word is used in many different context, so we decided to name the graphical element of the visual programming otherwise - a box.

We distinguish between 4 terms in the box terminology: box category, box type, box name and box. We use an example to illustrate the differences of the terminology. We have the 4ft task box. A box is one particular instance of the box type 4ft task. It is defined by its parameters (antecedent, succedent, quantifiers...). User can name this box or he can use the standard name provided by the system. The box

type 4ft task is an identifier of all instances of 4ft tasks. Finally, the box category in this case would be 4ft-Miner category, which besides the task contains a set of quantifiers specific for the 4ft task.

For the user, box is a graphical object, that can be used on the desktop. Box is represented by icon, name and set of properties, that the user can change. Furthermore, it contains sockets, where other boxes can be connected. This connection will be represented on the desktop by an arrow from the connected box into the socket. Each box type has a predefined list of actions, which can be used with the box instances of this type.

A box type can also contain Modules for interaction. Module for interaction means executing an action of the box, which is provided by other module that the box itself. Modules for interaction can be dialogs, tables, graphs...

Boxes asking for creation is a concept, that should simplify the visual data mining in Ferda. A box can offer a list of boxes in its context menu or in the main application menu. When a user clicks on the box (in the menu), the clicked box is created, and the properties and connections are properly set. We should create a demonstrative example to show how it works. Let us have a *Database* box. This box offers to create a *Data matrix* box. There is a list of data matrices names in the database in the context menu of the *Database* box under the *Boxes asking for creation* item (this works only when there the *Database* box has a valid ODBC connection string set). When the user clicks on one of the names of data matrices, a new *Data matrix* box will be created besides the *Database*. Moreover, the *Database* box will be connected to the newly created *Data Matrix* box.

A box contains a full information what is connected inside the box, even if the user does not see it on the desktop (in Ferda's visual data mining it means the part of the graph that leads to the box "from the left"). User can know this information with the aid of packing and unpacking mechanism, which is provided by the desktop (more in the desktop section).

Parts of the working environment

Menu

Application has a menu that provides the most important functionality. The menu consists of a static part and a dynamic part. The static part calls functions that manage the whole application. The dynamic part is dependent on the box (boxes) selected on the desktop or in the archive.

Application menu functions

File

The File group of the menu contains actions for managing the project. Project for the user means all the boxes in the archive and in all the desktops.

- **New project.** System creates a blank new project.
- **Open project.** Opens project on a defined location. It uses the standard .NET OpenFileDialog.
- **Recent projects.** In the submenu of this item, there is a list of ten projects user recently opened in Ferda.
- **Save project.** Saves the project on the disc. If there wasn't an open project, it behaves exactly like *Save project as*.
- **Save project as.** Saves the current project into a location specified by the standard .NET

SaveFileDialog.

- **Close.** Ask the user if he wants to save the changes in current project. Then optionally saves the project, releases all the resources and closes the application.

Edit

The Edit group of the menu contains actions for controlling the boxes. Menu is created dynamically depending on the selected boxes in the archive or on the desktop. If there is only one box selected, submenus *Actions*, *Modules for interaction* and *Boxes asking for creation* are added to the menu.

View

The View file group is another group of actions common to the applications with more user components. User can freely move and dock various components as the Archive or Property Grid. If the user closes one of the components, it can be opened again by clicking on its name in the *View* menu group.

- Archive
- New box
- Context help
- Property Grid
- User note

Desktop

The Desktop group of the menu is designed for a more comfortable work with the desktops. It provides the functionality for opening and closing the desktops (maybe more in the future).

- **New desktop.** Creates a new blank desktop
- **Open desktop.** There is a list of names of desktops that are in the project, but are not opened at the moment in the submenu of this item. When the user clicks at one of the names, the desktop will open.
- **Remove desktop.** By clicking at one of the desktop names in the submenu of this item, user deletes that desktop from the project. Boxes that were on the desktop remain in the Archive.

Tools

The Tools menu group contains the tools to configure the application. We do not know yet which kinds of setting will be accessible to the user. There will be the localization of the application, some advanced settings of the environment, maybe the ICE Pack Server address or dynamic modules loading into the application present in this section. For now, it contains the item *Preferences* that displays the incomplete *Preferences* dialog. In this dialog, user can change the localization of the application.

Help

This group contains the help of the system.

- **Ferda's tutoria.** This item opens a pdf document that helps a beginner in Ferda and guides him

through his first data mining project.

- **Ferda help.** It shows the user the help about FrontEnd user environment and how to use it.
- **Theoretical help.** This item shows a pdf-style document that will contain the theoretical background behind the Ferda system (the LISp-Miner data mining procedures).
- **About Ferda.** Shows the standard About dialog.

Toolbar

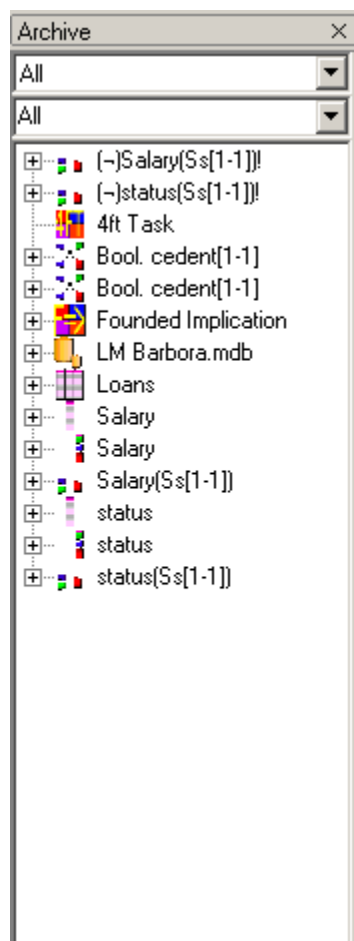
Toolbar is an alternative for the main application menu and the context menu of the boxes. Each item in the toolbar has its own icon and when user holds the mouse on the icon, it shows the name of the action represented by the icon. By clicking on the icon, the same action is executed as it would be in the corresponding item in the menu.

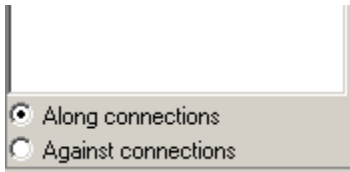
Archive

The aim of archive is to simplify the process of data mining for the user. All boxes that have been created by the user are stored in the archive. User can thus easily return to previous practices and reuse them. With initial setting, the archive is located on the left side of user environment.

The archive component consists of four parts. The first part is a combo-box with all the box categories that the user used in the project. The second part is also a combo-box, but contains the box types that user used in the project. The third part is a .NET TreeView component for showing the archive. Each item in the archive is represented by an icon and a name. The fourth part contains 2 buttons for interpretation of unpacking of the archive - along or against the direction of connections.

Visual representation of the archive





User work with the archive

- **Switching through the archive parts.** User can switch through the archive parts by using tab.
- **Vertical archive browsing.** Vertical browsing: user selects a box type that he has in his archive. He is doing it by selecting a box category in the first combo-box or (and) selecting the box type in the second combo-box. System shows in the TreeView all boxes of that type that are in the archive.
- **Change of archive browsing direction.** The archive supports browsing against and along the connection direction. The 2 buttons at the bottom of the archive aid to distinguish this direction. The first button represents browsing along the connection direction, the other one against. When user selects one of the buttons, the other one deselects automatically. Therefore it is clear in which direction to unpack. After a change of direction archive is reset into the initial position: all boxes of type selected in the combo-box are displayed.
- **Horizontal archive browsing.** Horizontal archive browsing means, that user examines the archive in the direction of incoming or outgoing arrows on the desktop - along and against the connection direction. In case of outgoing arrows (along the connection direction) user clicks on the cross(on the left) by the box in the TreeView. Afterwards the component unpacks all the boxes that are connected by an outgoing arrow with the working (clicked) box.

In case of incoming arrows (against the connection direction) user also uses click on the cross beside the box. Then, all the boxes connected to some socket of the box are displayed. One should remark that even if in the TreeView the unpacking goes to the right, on the desktop it would be to the left.

- **Showing the box in the Property Grid, Context help and on the Desktop.** If user left-clicks on a box in the TreeView part of the archive, the properties of the box are showed in the Property Grid and the boxes context help is shown in the Context Help. If the box is visible in any of the desktops, it is highlighted red.
- **Renaming the box.** User can rename any box with the aid of the context menu, item *Rename*. After clicking on *Rename* in the context menu, user inserts a new name and clicks *Enter*. The action can be done also by a standard key *F2*.
- **Copying the box into the clipboard.** After selecting the *Copy* item in the context menu, the box is copied into the clipboard. This can be done also by pressing *Ctrl+C* key.
- **Box cloning.** Cloning of the box means creating a new instance of the box type in the archive. The new box inherits all the properties and input connections of the cloned old box. However, it will be a new independent object in the archive (and in the whole project). User clones the box by clicking on the *Clone* item in the context menu. System then creates and displays the new box in the archive. This can be done also by pressing *Ctrl+E* key.
- **Deleting the box from archive.** Deleting box from the archive means for the user a complete removal of the box from the whole project and loss of all information connected to the box. User deletes the box by clicking on the *Delete from archive* item in the context menu. System then asks the user if he wants really to delete the box and deletes it on approval. This can be done also by pressing *Shift+Del* key.

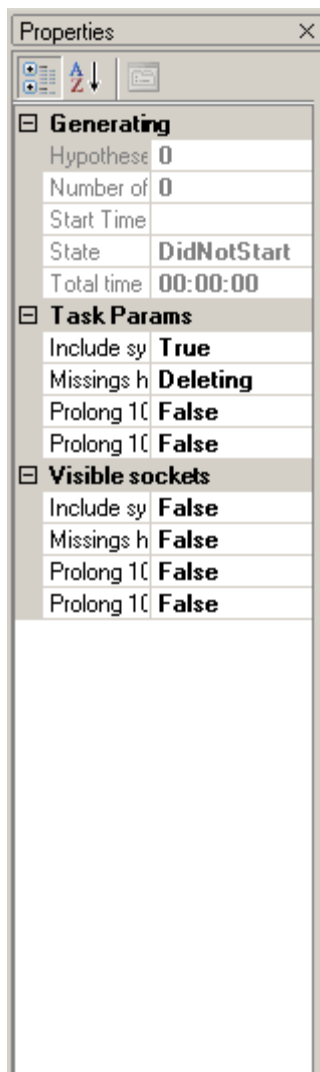
- **Box validation.** Validation of a box means checking the availability of all the actions that a box can do with its current settings in a current context. If there are other boxes connected into this box, it also means checking of semantic constraints of these boxes. User validates the box by clicking on the *Validate* item of the context menu. The action can be done also by the *Ctrl+Q* shortcut.
- **Modules for interaction.** The *Modules for interaction* submenu offers the user all the modules for interaction that the given box provides. More information in the *Box - module from the users view* section.
- **Executing the box actions.** Each box type can have actions that can be run in the user environment. All actions are to be executed by the context menu *Actions* submenu, each action having there its own item with the same name.

User note

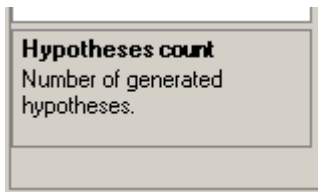
User can add some additional information to any box that he has in the project. User note is a component located in the right lower part of the docking environment. It is hidden behind the context help defaultly. If user selects a box in the archive or on the desktop, he can add a note to the box by simply adding a text in the User note component.

Property grid

Property grid is a control, where user can set the properties of all boxes. Properties of simple types can be set directly in the property grid, for more complex properties the property grid runs dialogs with the ability to set the property.



Generating	
Hypotheses	0
Number of	0
Start Time	
State	DidNotStart
Total time	00:00:00
Task Params	
Include sy	True
Missings h	Deleting
Prolong 1C	False
Prolong 1C	False
Visible sockets	
Include sy	False
Missings h	False
Prolong 1C	False
Prolong 1C	False



A box can have many properties and they can belong to several groups. The property grid component shows all the visible properties of the box and sorts them into property groups.

If the user selects more than one box on the desktop, there are properties displayed in the property grid that have all the boxes in common. If the properties share also the same value, the value is displayed (otherwise a default value for the type of the property is used). If user set some of the common properties, the value is automatically set into all the selected boxes.

There is a special group of properties in the property grid called *Visible sockets*. For each normal writable property of the socket, there is one Boolean property in the *Visible sockets* group, implicitly set to *false*. If the user sets this property to *true*, a new socket is created on the desktop for the selected box. This socket represents this property. Afterwards, the property can be set only at the desktop by connecting a box into the socket and it is read-only in the property grid. Socket can be removed after resetting the value of the property in the *Visible sockets* to *false*.

User work with the Property grid

The standard .NET component PropertyGrid is used for the implementation of Ferda's Property Grid. All the user work with this component is specified in the .NET library.

New Box

The New box component is used for creation of new boxes in the Ferda application. It contains boxes divided thematically into a tree structure. User creates new boxes by dragging a selected box type from the New box component and dropping it into the desktop.

Context help

The aim of the context help is to introduce the user with the functionality of the box currently selected in the desktop (archive). In the recent implementation it contains the name of the selected box, type of the box and a short hint. In the future, it will also contain help for individual sockets and properties and links to relevant files with theoretical help.

Desktop

Desktop is the main component for visual data mining. User inserts new boxes into the desktop, connects them, packs or unpacks them, and executes them (calls their actions).

User is not limited by only one desktop, he can open, show and hide as many desktops as he wants.

Abilities of the desktop

Actions of the desktop

- Selecting boxes
- Showing the box in the Property grid, Context help and the User note

- Moving the boxes
- Drag&drop support
- Creating a connection

The context menu actions above the blank desktop

- Pasting a box
- Graph layout
- Making a group box

The context menu actions above a box

- Renaming a box
- Cloning a box
- Copying the box to the clipboard
- Localizing the box in archive
- Deleting the boxes from the desktop
- Deleting the boxes from the archive
- Validating a box
- Modules for interaction
- Boxes asking for creation
- Executing actions of the boxes
- Packing all sockets of the box
- Unpacking of one layer of all sockets of the box
- Unpacking of all layers of all sockets of the box

The context menu actions above a socket

- Packing of a socket
- Unpacking of one layer of a socket
- Unpacking of all layers of a socket

User work with the desktop

- **Selecting boxes.** User can select boxes on the desktop by clicking the left mouse button and creating a rectangle around the boxes. Selecting boxes can be used for moving the boxes, deleting the boxes or setting the common properties.

- **New box.** User creates new boxes on the desktop by dragging and dropping them from the *New box* component.
- **Showing the box in the Property grid and the Context help.** If the user left-clicks on the box on the desktop or selects more boxes on the desktop, properties of the box(es) are showed in the Property grid and context help. More information in the *Property grid* section.
- **Moving the boxes.** User moves the boxes around when he selects one or more boxes on the desktop. The by pressing the left mouse button and moving the mouse user can move the boxes into the desired location. The connections to the box also move with the box.
- **Drag&drop support.** The desktop supports drag&drop from the Archive and New box components. For placing a box existing in the archive user can use drag&drop. User selects the box in the archive by left clicking on it and the drags it into the desktop. System checks if there is not already this box somewhere on the desktop. Afterwards, it shows the box on the desktop and connects it correctly into the remaining boxes on the desktop.

For creating a new box, user can drag&drop from the New box component. For more information, see the *New box* section.

- **Creating a connection.** Visual connecting of the boxes on the desktop is the most common way to crate connection between the boxes. First of all, user places the mouse on the output connector of the box he wishes to connect. Than the system shows him the handle on the socket and then the user left clicks on the handle, with the left mouse button clicked he moves with the mouse onto the socket of the box, he wishes to connect. System checks the type consistency of the connection. If it is not correct, it generates an error message, otherwise it creates a new connection.
- **Pasting a box.** User can paste a box into the desktop by clicking on the *Paste* item of the context menu. In comparison with cloning the box, no new instance of the box is created while pasting. So it is the same practice as using drag&drop from the archive. System again checks, if there is already that box in the desktop and adds the box to the desktop, if not. This can be done also by pressing *Ctrl+V* key.
- **Graph layout.** A layout algorithm is being initiated after the *Graph layout* context menu item click. This algorithm will redistribute the boxes on the desktop. It can be useful, when the boxes overlay each other.
- **Renaming the box.** User can rename any box with the aid of the context menu, item *Rename*. After clicking on *Rename* in the context menu, user inserts a new name and clicks *Enter*. The action can be done also by a standard key *F2*.
- **Box cloning.** Cloning of the box means creating a new instance of the box type in the archive. The new box inherits all the properties and input connections of the cloned old box. However, it will be a new independent object in the archive (and in the whole project). User clones the box by clicking on the *Clone* item in the context menu. System then creates and displays the new box in the archive. This can be done also by pressing *Ctrl+E* key.
- **Copying the box into the clipboard.** After selecting the *Copy* item in the context menu, the box is copied into the clipboard. This can be done also by pressing *Ctrl+C* key.
- **Localizing the box in archive.** Sometimes it is important for the user to get a quick overview of the box and its neighborhood, that cannot be gotten from the desktop or property grid. For these purposes, there is a *Localize in archive* context menu action. System reacts by setting the TreeView archive part into the basic position that corresponds to the box type of the selected box. (see Archive, Vertical Archive browsing). The system also selects the box in the archive. User can evoke

this action also with the *Ctrl+H* key.

- **Making a group box.** When user selects more boxes on the desktop, he can place them into a *group* box by clicking on the *Make a group box* context menu item. System then creates a group box, connects all the selected boxes into the group box and hides the selected boxes. The keyboard shortcut for this action is *Ctrl+G*.
- **Box validation.** Validation of a box means checking the availability of all the actions that a box can do with its current settings in a current context. If there are other boxes connected into this box, it also means checking of semantic constraints of these boxes. User validates the box by clicking on the *Validate* item of the context menu. The action can be done also by the *Ctrl+Q* shortcut.
- **Deleting the boxes from the desktop.** Deleting the boxes from the desktop means (in comparison to deleting boxes from the archive) deleting the visual representation from the focused desktop. This can be used when the desktop is overfilled with boxes and the user can erase some boxes. User can delete the boxes from desktop by clicking on *Delete from desktop* context menu item or keyboard shortcut *Ctrl+D*.
- **Deleting the box from archive.** Deleting box from the archive means for the user a complete removal of the box from the whole project and loss of all information connected to the box. User deletes the box by clicking on the *Delete from archive* item in the context menu. System then asks the user if he wants really to delete the box and deletes it on approval. This can be done also by pressing *Shift+Del* key.
- **Modules for interaction.** The *Modules for interaction* submenu offers the user all the modules for interaction that the given box provides. More information in the *Box - module from the users view* section.
- **Executing the box actions.** Each box type can have actions that can be run in the user environment. All actions are to be executed by the context menu *Actions* submenu, each action having there its own item with the same name.
- **Boxes asking for creation.** There are all the boxes that can be created from this box by the mechanism or Boxes asking for creation. User can select and create a box by clicking on its name in the submenu of this item.
- **Packing all sockets of the box.** When the user wants to pack all the ingoing connected boxes, he uses the context menu item *Pack all sockets* or the keyboard shortcut *Ctrl+P*
- **Unpacking of one layer of all sockets of the box.** This context menu item unpacks one layer of boxes that are connected to all the sockets of the box. There is also a keyboard shortcut *Ctrl+S*
- **Unpacking of all layers of all sockets of the box.** This context menu item unpacks all layers of boxes that are connected to all the sockets of the box. There is also a keyboard shortcut *Ctrl+U*

User desktop work with the keyboard

User can apply following keyboard actions on the desktop:

- *Esc* key cancels the box selection.
- The *F8* key pressed with selected connection deletes this connection.
- The *Ctrl+A* selects all boxes on the desktop.

- The *Ctrl+arrow* key moves the selected boxes in a direction of the arrow.