ABng Repository Manager

Institute of Computer Science
University of Mining and Metallurgy
Cracow, Poland

Krzysztof Dobranowski & Wojciech Gosztyła

7th July 1999

Contents

Chapter 1

Introduction

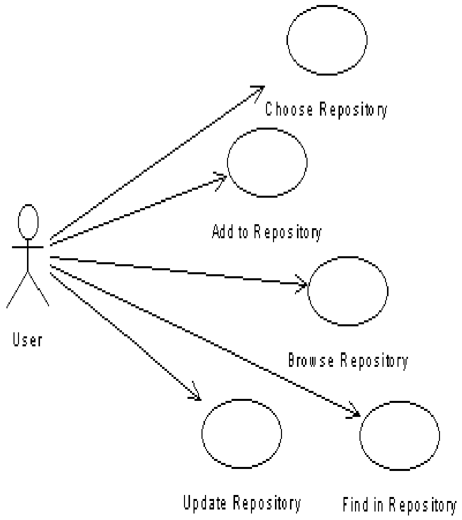
Repository manager is an application for managing and handling information in ABng system. It simplifies operations available for repositories such as browsing, adding, updating, removing and finding. Our application supports also callback mechanism which will notify the user every time any data was changed in repositories. Because of the fact that most of the repositories are very similar, our goal was to create a pattern which will simplify in future a process of adding an interface for a new repository. Further, this solution can be used for developing interfaces for newly created repositories.
Application was written in Java 1.1.4, Swing 1.0.1 and OrbixWeb 3.0.

Chapter 2

General structure of the system

2.1 Use Cases

Figure 1 shows how the system functions from it's user point of view. User has access to Repository Manager which gives him a list of available repositories and allows him to choose a certain repository. After choosing a repository user has access to repository managing functions (browsing, adding, updating, finding).



2.2 Class diagram.

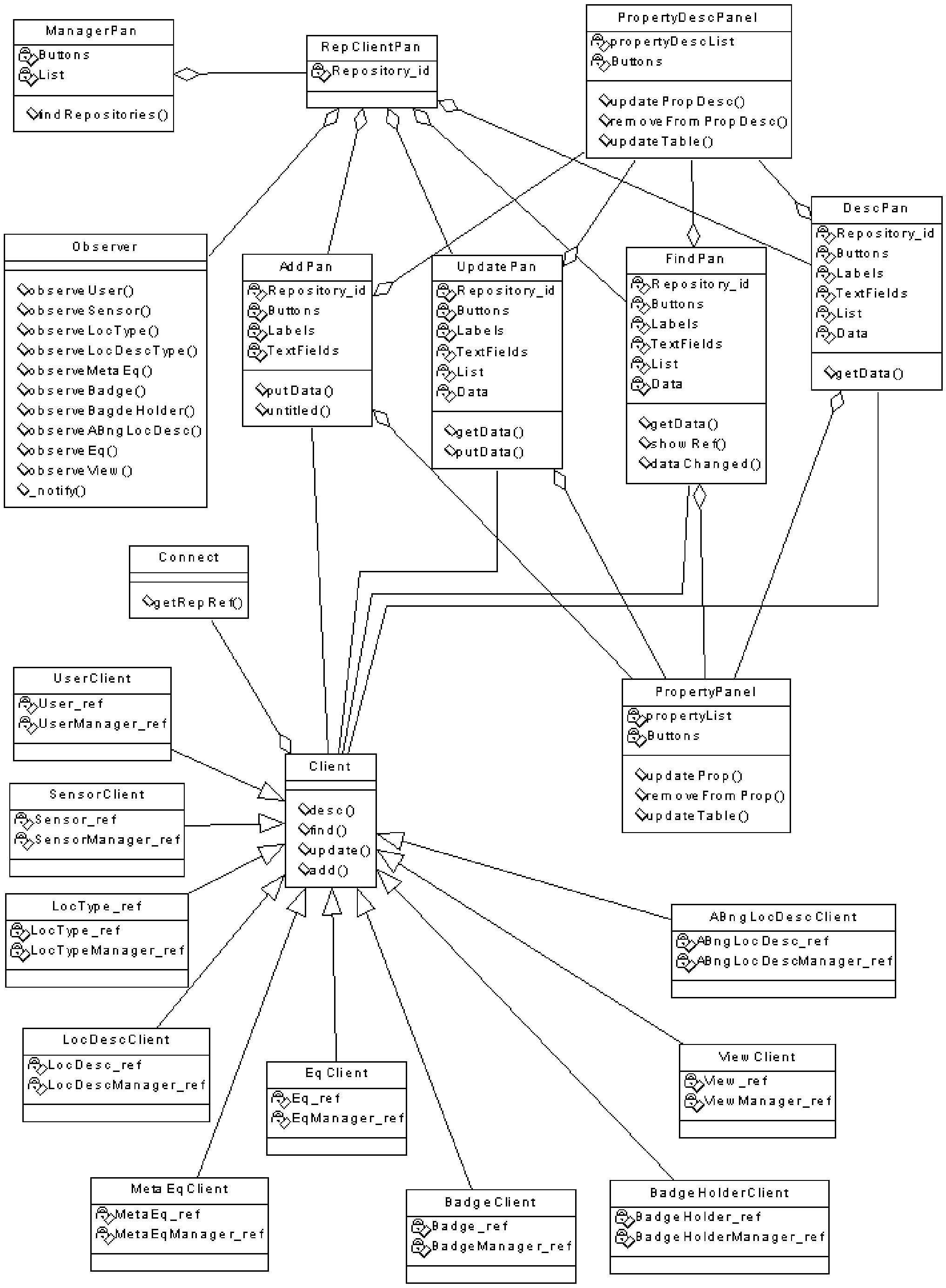


Fig. 2. Class structure of an application.

2.3 Class description

ManagerPan - class representing panel with the list of available repositories. From this panel user chooses repository to work with.

Attributes:
Buttons - an array of panel buttons.
List - a list which consists available repository names.

Functions:
findRepositories - finds the list of all available repositories and displays it in the List.

RepClientPan - class representing the main application panel.

Attributes:
Repository_id - an identificator of a chosen repository.

Functions: none.

DescPan - class representing panel from which user can browse repository data.

Attributes:
Buttons - an array of panel buttons.
Labels - an array of labels connected with TextFields
TextFields - an array of text fields on which repository data are displayed.
List - list of all repository data records.
Data - a two dimensional array of Strings which consists all repository data.
Repository_id - an identificator of a chosen repository.
PropertyPanel - panel on which property list is displayed
PropertyDescPanel - panel on which property description list is displayed

Functions:

getData - this function obtains a repository data from <RepName>Client class and displays it on an appropriate text fields, reference buttons, property and property description lists.

FindPan - class representing panel from which user can find certain data in repository.

Attributes:
Buttons - an array of panel buttons.
Labels - an array of labels connected with TextFields
TextFields - an array of text fields on which repository data are displayed.
List - list of found repository data records.
Data - a two dimensional array of Strings which consists found repository data.
Repository_id - an identificator of a chosen repository.
PropertyPanel - panel on which property list is displayed
PropertyDescPanel - panel on which property description list is displayed

Functions:
getData - this function obtains a repository data from <RepName>Client
class and displays it on an appropriate text fields, reference buttons, property and property description lists.
dataChanged - this function is called by _notify from Observer class every time data are changed in repository, It updates currently displayed data.
showRef - function is called every time user presses reference button. This function is responsible for displaying the data of this reference on the Find Panel.

UpdatePan - class representing panel from which user can update data in repository.

Attributes:
Buttons - an array of panel buttons.
Labels - an array of labels connected with TextFields
TextFields - an array of text fields on which repository data are displayed.
List - list of repository data records.
Data - a two dimensional array of Strings which consists repository
data.
Repository_id - an identificator of a chosen repository.

PropertyPanel - panel on which property list is displayed
PropertyDescPanel - panel on which property description list is displayed

Functions:
getData - this function obtains a repository data from FindPan (it displays and allows update of data found in Find Panel).

putData - this function obtains data entered by user from the text fields , reference buttons, property and property description lists and sends it to <RepName>Client for purpose of writing data to repository.

AddPan - class representing panel from which user can add new data to repository.

Attributes:
Buttons - an array of panel buttons.
Labels - an array of labels connected with TextFields
TextFields - an array of text fields on which user can enter a new data.
Repository_id - an identificator of a chosen repository.
PropertyPanel - panel on which property list is displayed
PropertyDescPanel - panel on which property description list is displayed

Functions:
putData - this function obtains data entered by user from the text fields , reference buttons, property and property description lists and sends it to <RepName>Client class for purpose of writing data to repository.

PropertyDescPanel - class which represents panel responsible for displaying and managing property description list. This panel is present only for two repositories (Location Type Repository, Meta Equipment Repository)

Attributes:
Buttons - set of control buttons
propertyDescList - array which consists of property description data

Functions:

updatePropDesc - this function adds or updates data in property description list

removeFromPropDesc - this function removes chosen data from property description list

updateTable - converts property description list to a table in which the data are displayed

PropertyPanel - class which represents panel responsible for displaying and managing property list. This panel is present only for two repositories (Location Description Repository, Equipment Repository)

Attributes:

Buttons - set of control buttons

propertyList - array which consists of property data

Functions:

updateProp - this function adds or updates data in property list

removeFromProp - this function removes chosen data from property list

updateTable - converts property list to a table in which the data are displayed

Client - abstract class created for purpose of easy adding an interface to new repositories.

Attributes: none.

Abstract Functions:

desc - this function is supposed to obtain data from the repository and return them to the DescPan where the data are displayed.

find - this function is supposed to find a certain data in the repository and return them to the FIndPan where the data are displayed.

update - this function is supposed to update record in repository.

add - this functions is supposed to add new record to repository.

<RepName>Client - ancestor classes for class Client. Each class is connected with one repository. Those classes are responsible for writing or reading data directly from certain repository and sending those data to <Pan> classes.

Functions are inherited from Client class. In case of adding an interface to a new repository new <RepName>Class must be added and all those functions must be written from the beginning for purpose of handling a new repository (see 4. for more details).

Connect - this class handles communication with repositories.

Attributes: none.

Functions:
getRepRef - this functions returns reference to certain repository.
resolve <RepName> - set of functions which obtains reference to a specified repository or repository manager

Observer - this class is responsible for handling callback mechanism
Functions:
observe<RepName> - this function register callbacks for certain repository.
_notify - this function is called every time something has changed in observed repository

2.4 Data flow in the system.

Below we show two examples of interaction diagrams. The first one describes the behavior of the application when a new user is added to the system. The second one shows a sample of callback invocation during update process.

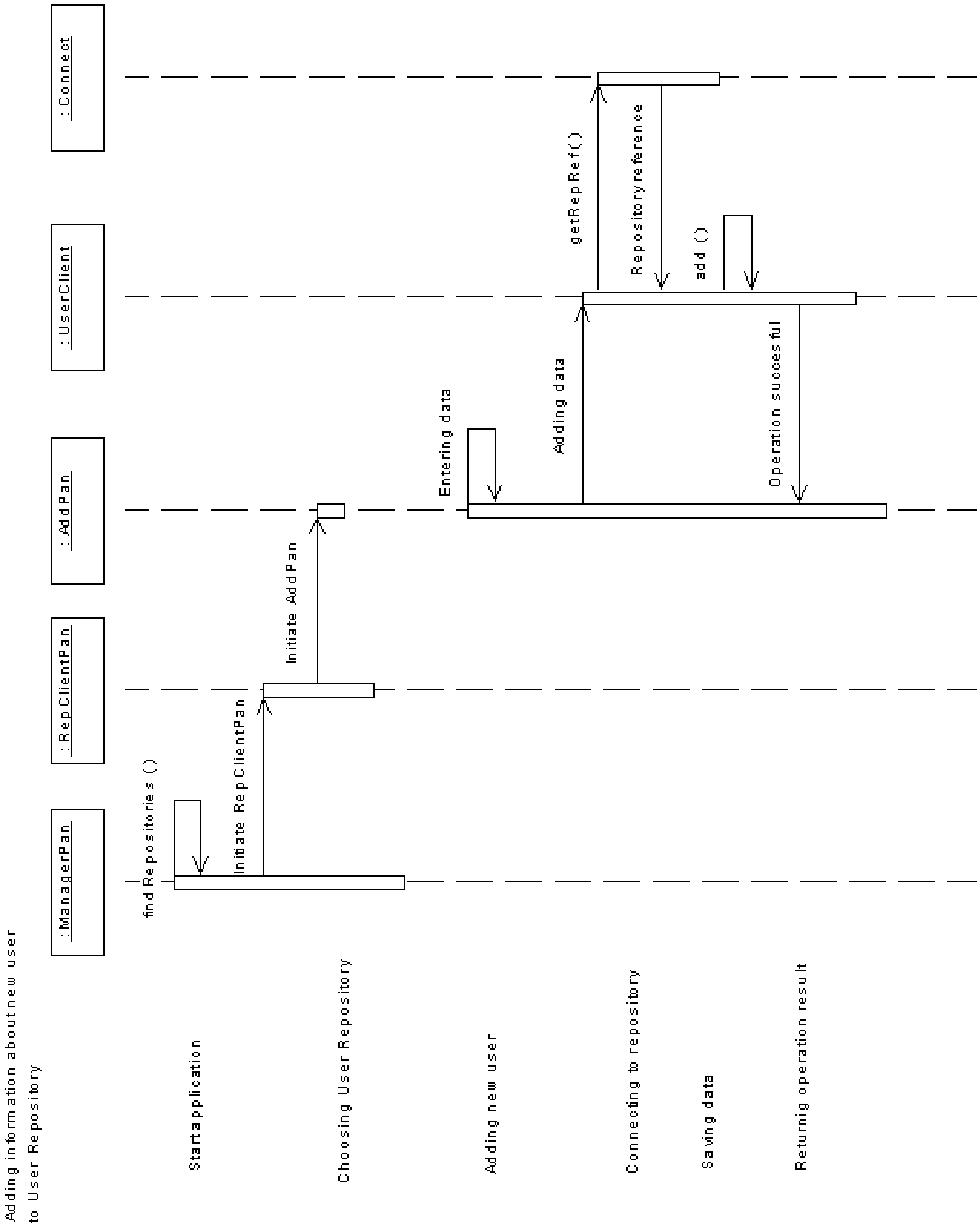


Fig. 3. Adding information about new user to the User Repository.

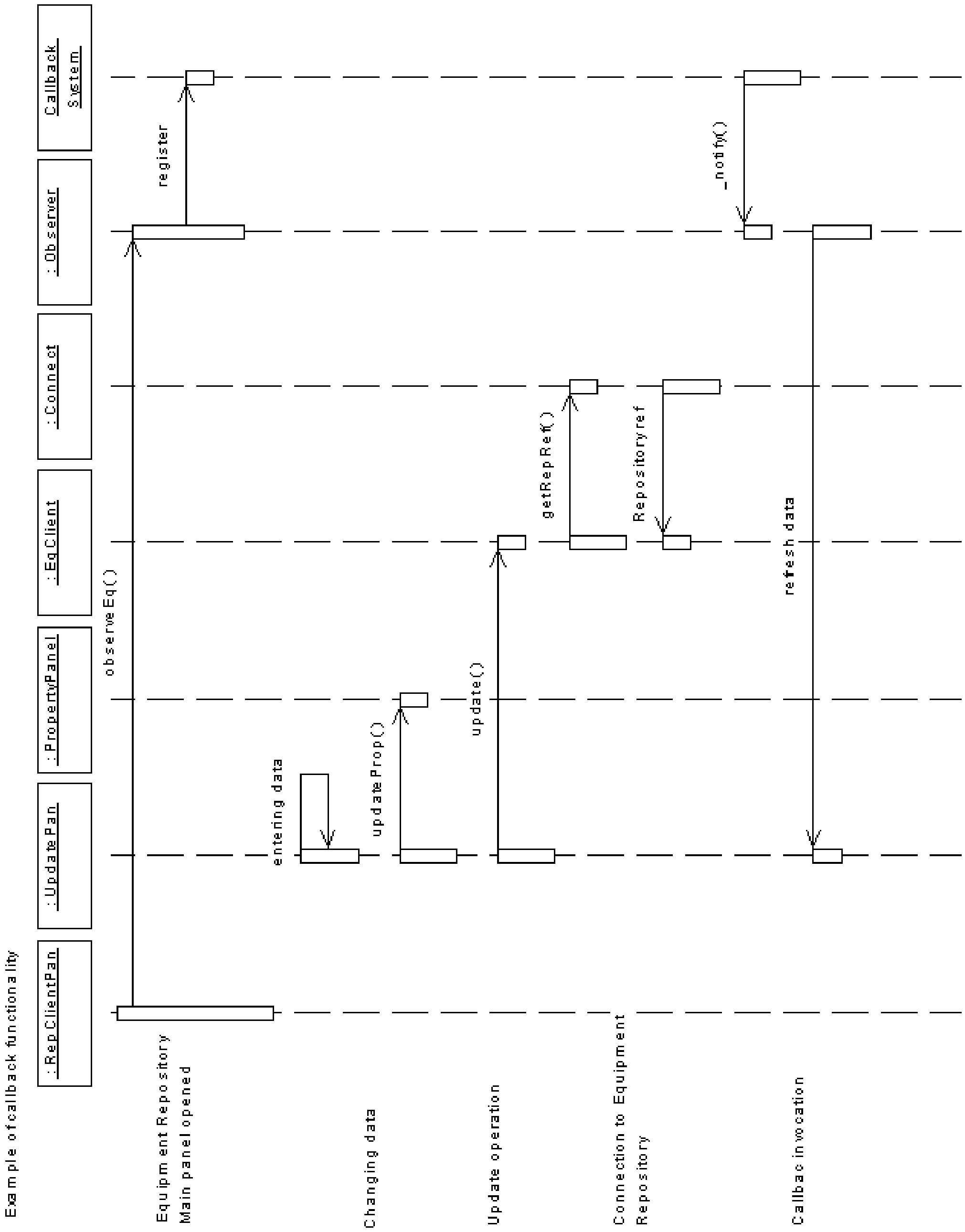


Fig. 4.Callback invocation during update process.

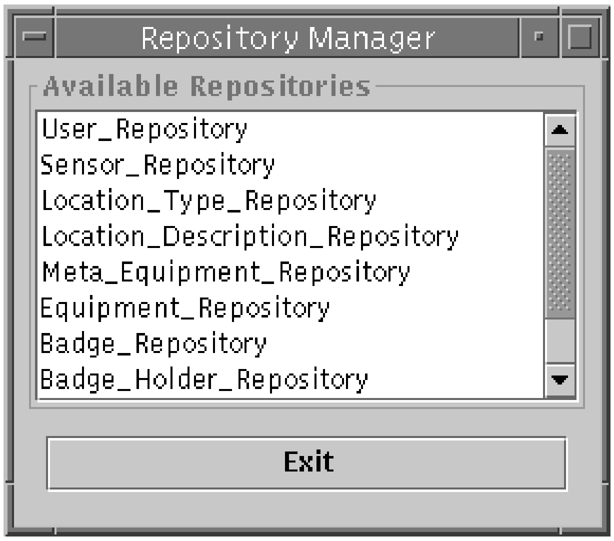
Chapter 3

System functionality

3.1 How does it work?

3.1.1 Manager Panel

After starting Repository Manager a ManagerPanel window is displayed.



From this window user can choose one of the available repositories (those which repository servers are alive). After choosing certain repository from a list a Repository Panel is displayed.

This panel consists of four bookmarks (Description, Find, Update, Add) from which user has access to repository managing functions. Every repository consists of different data types, that's why bookmarks can consist of different set of components such as:

list - shows repository data records

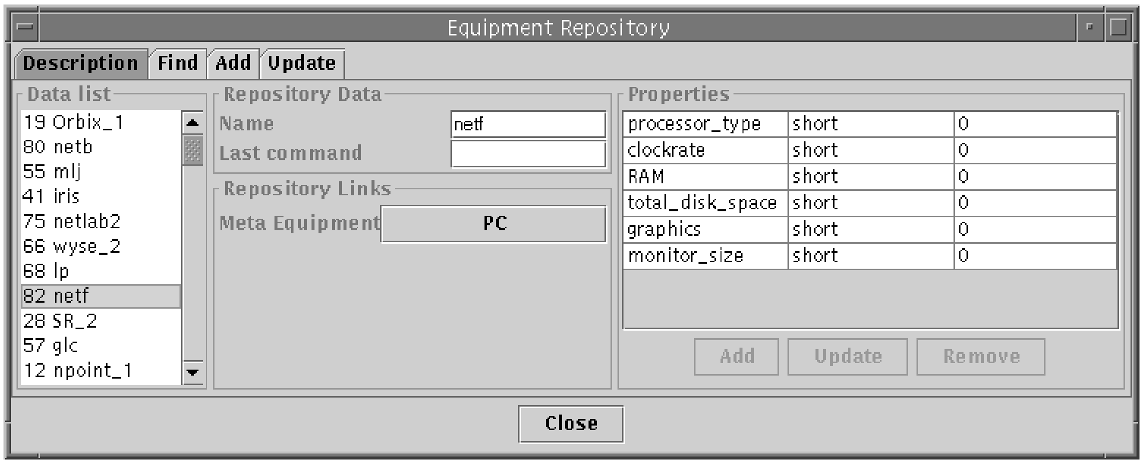
text fields - on which repository data are displayed or in which user can enter appropriate data,

buttons - which are used to reach other repositories (some repositories have references to another one).

property or property description panel - showing property or property description data

Below we show an example of four bookmarks for the most complex type of repository panel which consists of every components mentioned above. It is based on Equipment Repository panel.

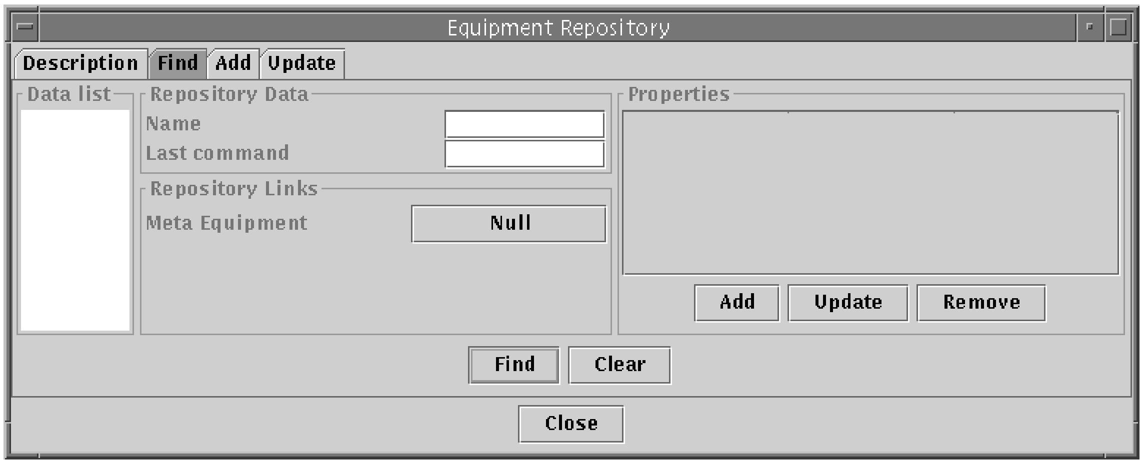
3.1.2 Description Panel



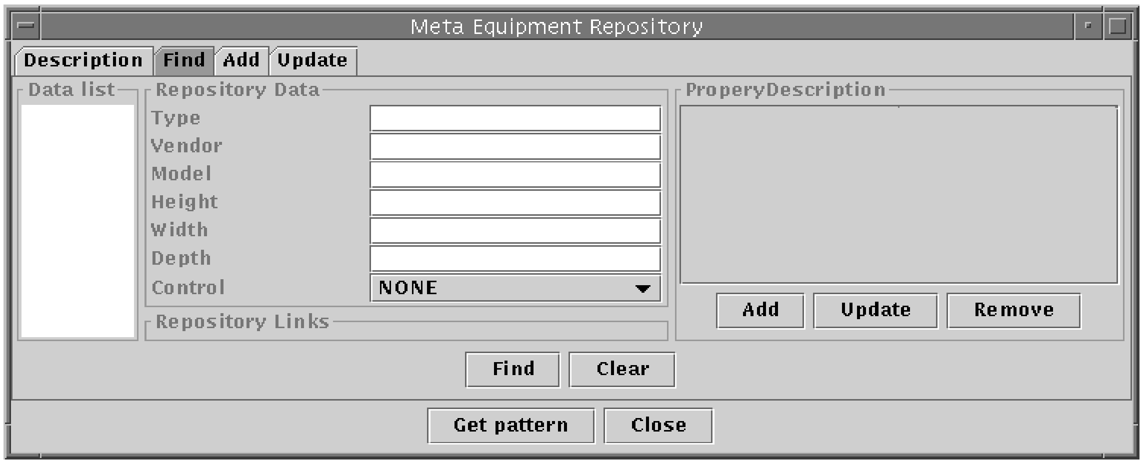
On the Description Panel user can browse all data included in repository. On the left side there is a list of all repository records. Clicking on the list user can choose which record will be displayed. In upper part of panel there are text fields which are filled with basic repository data. Below them there are reference buttons. Labels on the buttons display representative data of the reference. If label is set to "Null" that means no reference is present. Clicking on reference button opens the other repository panel which displays the data of the certain reference. In this case pressing Meta Equipment button (PC labeled) opens a Meta Equipment Repository panel and displays PC data.

On the right side there is a Property or Property Description Table. In this table all data from Property or Property Description List are displayed. In the very bottom part of this panel there is Close button which closes this panel.

3.1.3 Find Panel



The next panel is Find Panel. From this panel user can find certain data in repository. It consists of the same components as the Description Panel but all text fields are empty, all reference button labels are set to Null and there is no Property or Property Description table. To find data in repository user have to fill text fields, property or property description table and reference buttons with searched data. Not every data must be set, it depends on the user. Clicking on the add, update or remove button allows user to insert Property or Property Description data. Clicking on the reference button displays the repository panel to which button refers.

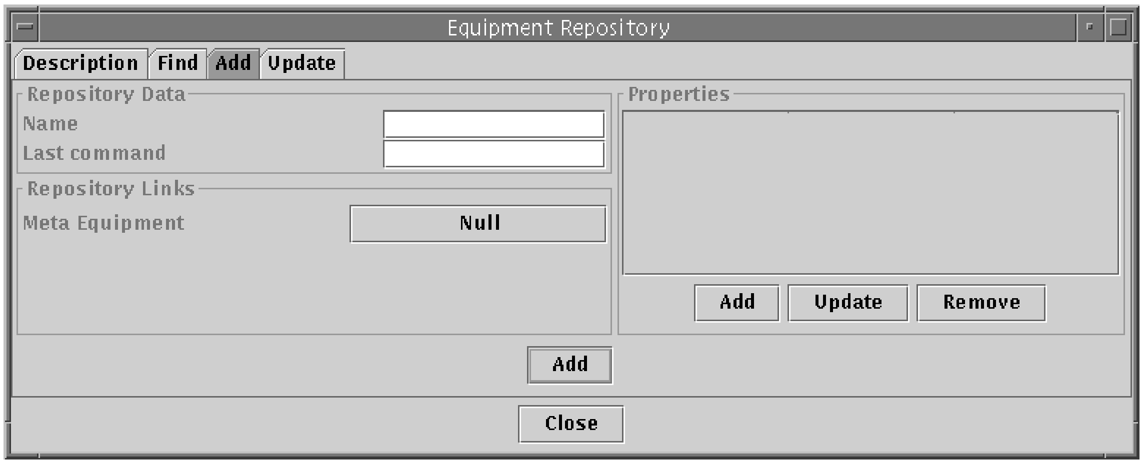


This panel differs a little bit from the “normal” Find Panel because of the “Get Pattern” button placed in the bottom part of the panel. On this panel user can also enter searched data. Pressing the “Get Pattern” button sends data to the “main searching panel” and sets appropriate reference button label to “pattern set”. Clicking once again on this button opens the repository panel to which button refers but with last entered data.

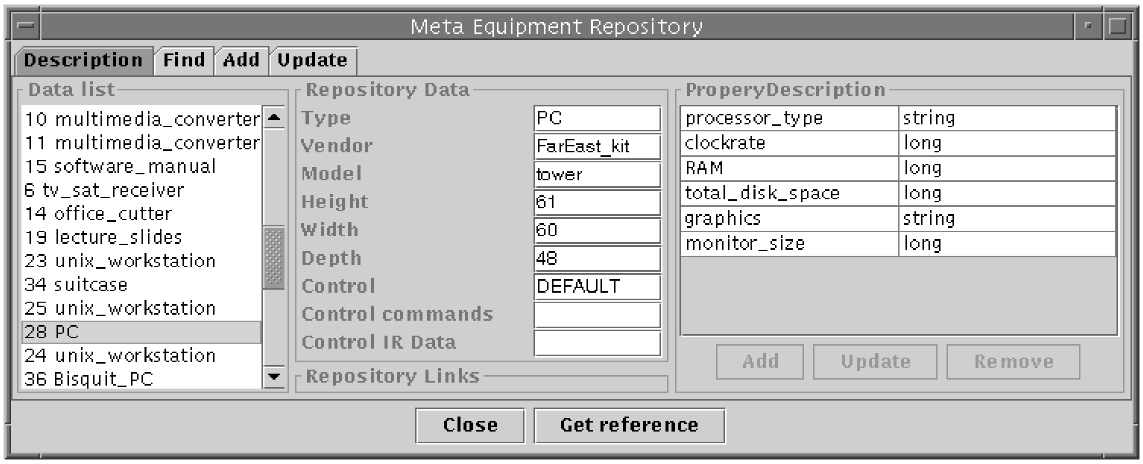
After entering all searched data user can press Find button from the bottom part of the Find Panel. If searching succeeds found records will occur in the list. In this moment Find Panel works like Description Panel but only for found data. If searching fails (no data are found) “No match” will be displayed in the list on the left side. To search the data once again user have to clear panel with “Clear” button and enter new searched data.

3.1.4 Add Panel

The add panel is used to add new data to repository. This panel slightly differs from previous two panels. It has no list on the left side.



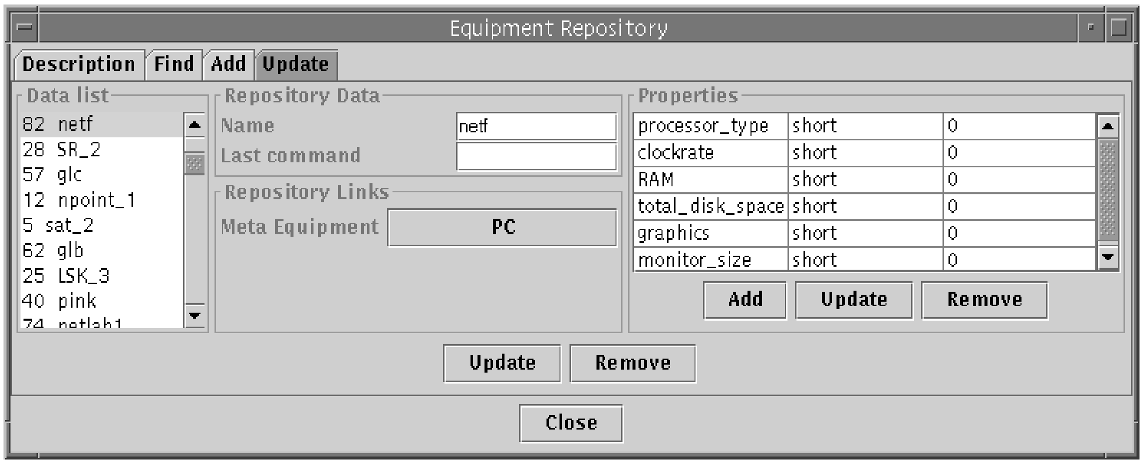
To add new record user have to fill text fields and property or property description table with appropriate data. To add a reference user have to click on the one of the reference buttons. After that repository panel to which button refers is opened.



User has to choose record from the list and press "Get reference" button which will send reference of the chosen record to "main add panel" and fill appropriate button label with representative data of the reference. Pressing the "Add" button will add a new record to repository. It is very important to enter all required data otherwise add will fail and user will be prompted with appropriate error message.

3.1.5 Update Panel

Update Panel is very similar to Find Panel. It uses the same data as Find Panel. User cannot update data unless he finds it using Find Panel.



After finding data user can browse it like on the Find Panel but he can also change them. Changing a reference works like in the Add Panel. After changing data user can update them pressing "Update" button. From this panel user can also remove data from repository. Choosing the record from the list and clicking the "Remove" button will remove chosen record from repository.

3.2 Callback

Our application supports callback mechanism which is implemented in ABng system. Every time user chooses certain repository from the list on Manager Panel, program starts observing this repository. Whenever anything will change in observed repository our application will be notified about that and will be able to refresh displayed data. Thanks to this mechanism a lot of Repository Manger applications can be started in the same time and still data coherency will be provided. Besides this mechanism assures that user has access to the newest data.

Data on Description Panel are refreshed each time callback message is received. Data on Find Panel and Update Panel are refreshed only when changed record is on the list of found records.

Chapter 4

Creating interface to a new repository.

To simplify adding an interface to a new repository we have created an abstract class **Client**. To add a new interface a new **<RepName>Client** class must be created. This class must inherit from Client class. Client class includes four abstract functions (desc, add, find, update) which are also inherited. Those functions are supposed to handle a communication with repository, by using an implemented Repository Server functions (such as find, add etc), and with graphical panels. To use Server functions user need to get a repository reference. In order to obtain repository reference function **getRepRef in Connect** class must be updated. Also new **resolve<RepName>** function must be added (and **resolve<RepName>Manager** function if needed).
Next step is to modify a **findRepositories** function from ManagerPan class to allow choosing this new repository from Manager Panel. Then **constructors** of RepClientPan, DescPan, FindPan, UpdatePan and AddPan class must be modified. For RepClientPan only a new **Repository_id** number must be chosen (important: this **number must differ** from other repositories numbers !!!) for other panels also the number of text fields, labels and buttons must be chosen (it depends on how much data one repository record consists of and how many references to other repositories has this repository). Also user have to specify if new repository has Property or Property Description Table . If new repository has reference to other repositories function **showRef** from Find Panel must be updated in order to allow user to browse reference data.
Finally the **Observer** class must be changed in order to support Callback mechanism for a new repository. New function **observe<RepName>** must be added, appropriate **_notify** function must be updated and constructor of Observer class must be updated. Also **dataChanged** function from Find Panel must be updated.

Despite the fact that repositories are usually similar most of them have some individual features so it is usually necessary to write a completely new functions or even classes to make a new interface fully functional.