Mobius Forensic Toolkit

version 0.5

Tutorial

December 9, 2009
by Eduardo Aguiar
eduardoaguiar@ig.com.br

Contents

1 Introduction

Nowadays, open source forensic tools are domain specific. Each tool tries to grab a little of the investigation scope, and some do it very well. Unfortunately, they lack integration, and their development is made harder because of the absence of common code, and therefore of code reuse. Their outputs are not standardized, and most of them use command line interface.

Mobius Forensic Toolkit is a framework to develop forensic tools. It is written in Python, using PYGTK and PyCairo. It is very extensible through extensions, whose are programs that share services, program environment and have access to a case model.

This tutorial is not intend to be a complete guide to the tools built so far, but it is simply a hands-on guide and may grow further with the releases to come.

2

Setting up your case

2.1 Creating case

The first step to use Mobius is to create a case. A case is an abstraction and might be anything you call a case, such as an investigation case, a part of an investigation case. It is basically a container of evidences.

To create a case, hit new case button at toolbar, or hit `File→New menu` option (see figure 2.1).

Figure 2.1: Mobius main window

A new case named `Untitled Case #01` was created (see figure 2.2).

To set up this case, hit the `properties` button. It will open the Case Properties dialog (figure 2.3), where you can edit your case properties. `Base`

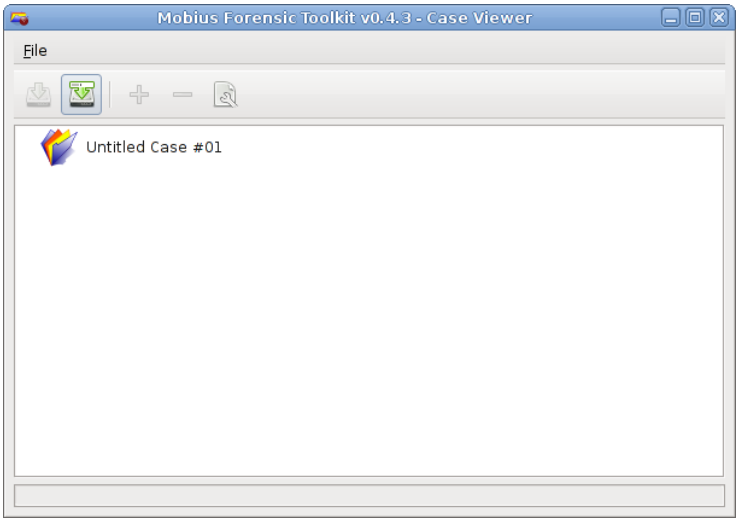

Figure 2.2: new case window

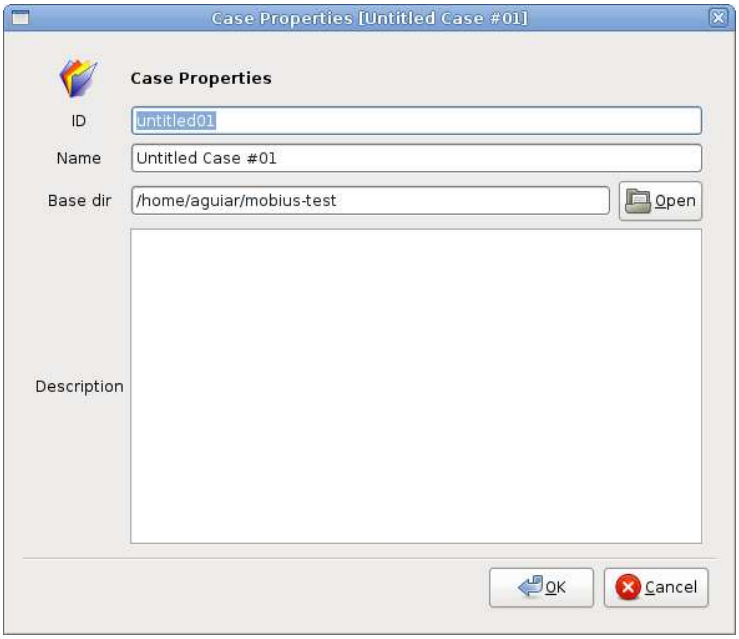

Figure 2.3: new case properties dialog

`dir` is where Mobius and its extensions will save information about your case, so choose a suitable folder.

You can save your case by pressing `save` button. It will open a file chooser dialog where you can enter your case. Mobius case files have extension `.case`, which is added by default.

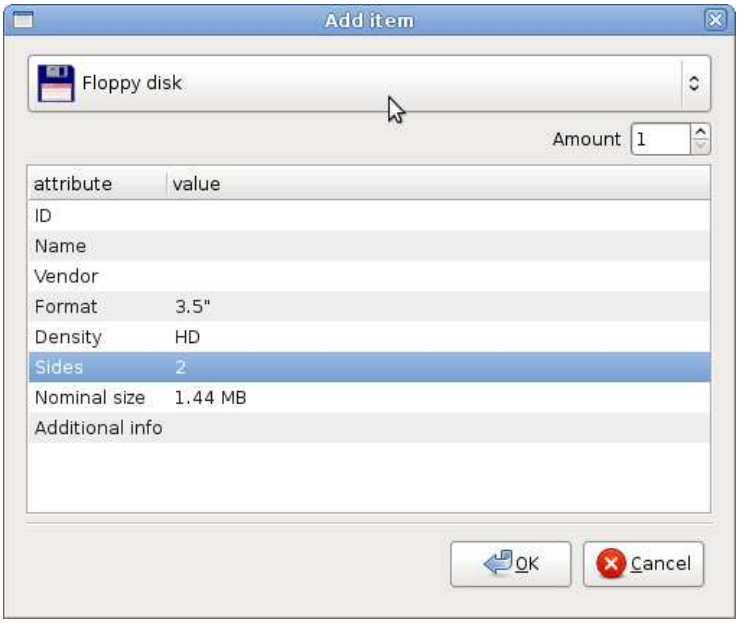

Figure 2.4: add item dialog

2.2 Adding items to case

Once your case has been created successfully, you can start adding evidences to it. Evidences are divided in categories, such as `harddisk`, `floppy` and so on. In section 4.1 we will see how to create new categories on the fly.

To add an item, you have to select its container. Click on case item at Case Viewer dialog. Now click on `add` icon. It will open "Add Item" dialog.

Choose a category and optionally the amount of items to be created. You can also set the attributes that are common to the items being inserted. The `Generic item` can be used to represent anything without having to create a new category. Usually it is used as a container, and may represent a place (John Doe's), a document (Investigation Request 055). To group items you can also use `Set` item, which is a generic set. That way, to group 154 floppies, you can create a set and 154 floppies under it.

In this example, we have created 5 floppies (figure 2.5).

Alternatively, since release 0.5 you can drag and drop a file directly into case or any item, to create a file object (figure 2.6). Some files have special meaning when dragged. For example, the `.log` file generated by Logicube Talon^{TM} is parsed and instead of a file item, a harddisk item is created, with some attributes filled, and associated to a datasource of type 'talon'.

2.3 Browsing item attributes

Now that we have a case and some items, let us browse item attributes. Double click on `Attribute Viewer` icon at Mobius main window, to open Attribute

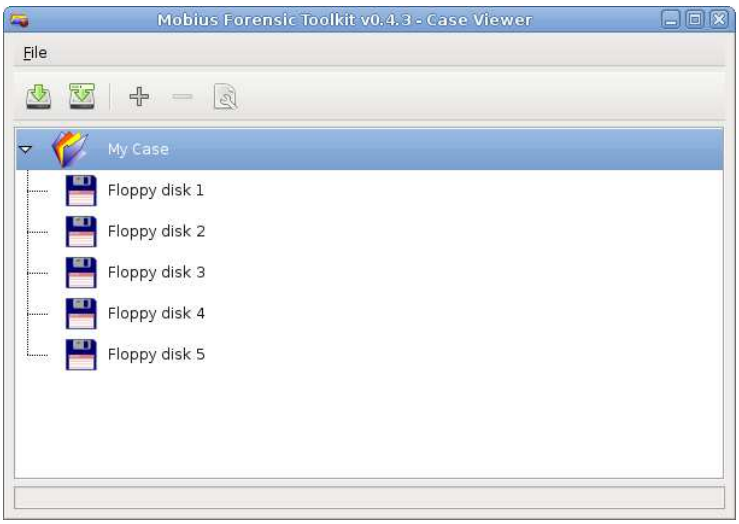

Figure 2.5: case viewer with 5 floppies

Figure 2.6: add item through drag and drop

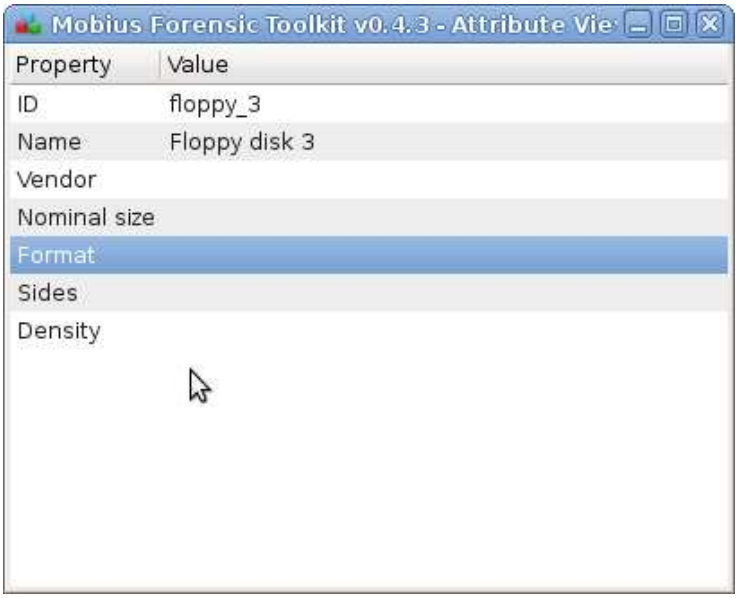

Figure 2.7: item attributes

Viewer. After that, click on any item to see its attributes (figure 2.7).

Clicking at any attribute starts its edition. After editing attributes, save case to persist changes.

3

Managing datasources

Each case item can have an associated datasource. A datasource is an object that represents a stream of bytes.

In section 2.2 we saw how to drag and drop a file to create an item. In fact, two objects are created: an item and an associated `raw` datasource, pointing to the dragged file path.

To manage datasources, click on `Data Sourcerer` icon. A window like the one shown in figure 3.1 will be opened. The Data Sourcerer extension can be used to associate and disassociate a datasource to an item, change datasource attributes, and export datasources.

Click on an item to see the datasource and its attributes. You can change the datasource associated to an item by dragging and dropping a new file directly into Data Sourcerer window.

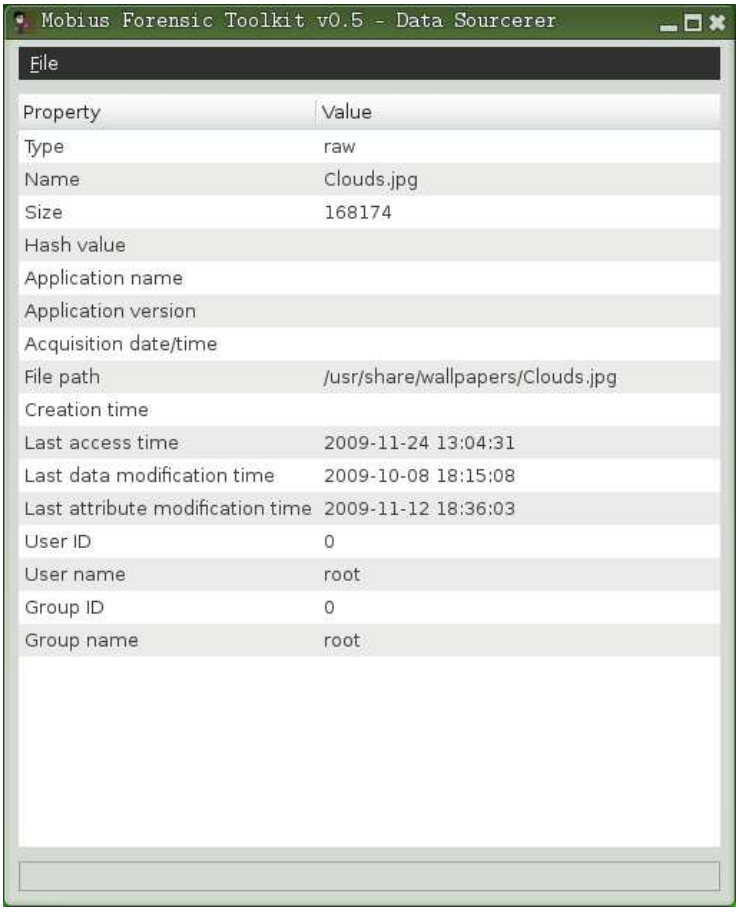

Figure 3.1: Data Sourcerer window

4

Managing categories

The Category Manager extension allows creation, modification and deletion of categories and their attributes on the fly (figure 4.1).

4.1 Creating categories

To create a category, hit `add` button below category listview (leftmost button). A new category named `<NEW CATEGORY>` is created. Now click on it to edit its icon, ID and name.

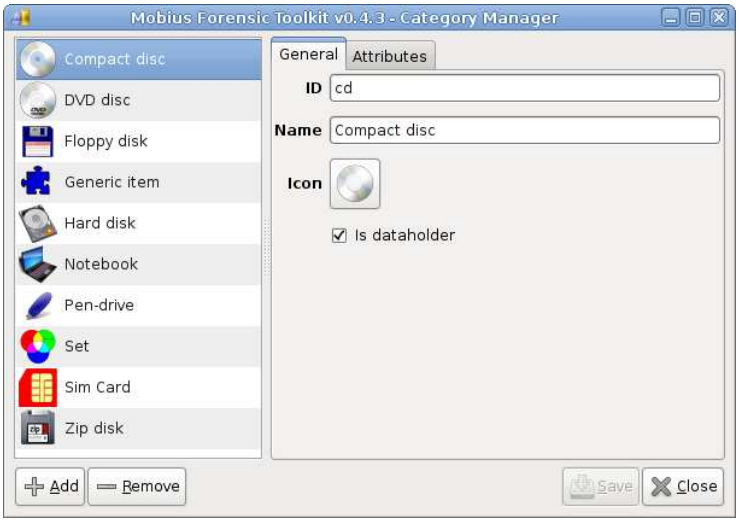

Figure 4.1: Category Manager extension

To edit its attributes, click on `Attributes` tab folder.

After modifications, click `save` button. Now this category will be available for all cases, and items of that type can be added to the current case.

You can also use Category Manager to modify existing categories and its attributes, and even to translate attribute's description to your language, as long as you keep its IDs from changing. You can add attributes to an existing category or even remove some attributes.

5 Managing parts

The Part Catalogue extension was created to fulfill attributes of common parts. If you have harddisks with part-number `ABC-123`, you can fill the attributes which are common to this kind of harddisk, lefting those which are device dependent in blank.

5.1 Automatic startup

Part Catalogue is started everytime you fill an attribute whose ID is `part_id`. If this part-id is already recorded in Part Catalogue database, it will fulfill item attributes with those attributes you have set to this part. If not, it will open a window to register this new part and its attributes.

To test this, add a harddisk to current case, open Attribute Viewer if it is not opened, click on harddisk item and change Part ID to **ABC-123**. Part Catalogue will open a window like the one shown in figure 5.1.

Enter attributes which are common to this part number and hit `save` button. The next time you enter a harddisk with part id **ABC-123** in Attribute Viewer, Part Catalogue will automatically fill its attributes.

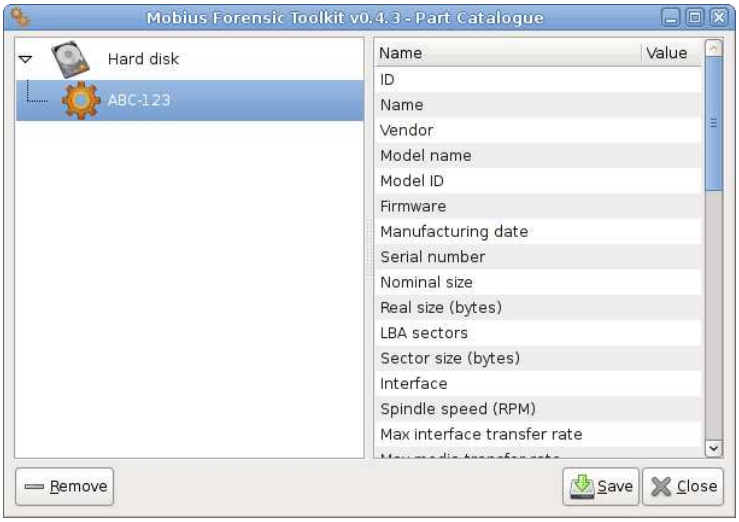

Figure 5.1: Part Catalogue extension

6 Imaging floppy disks

The Floppy Imager extension was designed to image floppy disks and collect its metadata as well. It runs only in Linux systems. To run, `/dev/fd0` must have permission **0666**.

6.1 Running Floppy Imager

To start Floppy Imager, click on `Floppy Imager` icon at Mobius main window. A window like one shown in figure 6.1 will be opened. Floppy Imager is only active when you select a floppy case item. Any other item will handle Floppy Imager inactive.

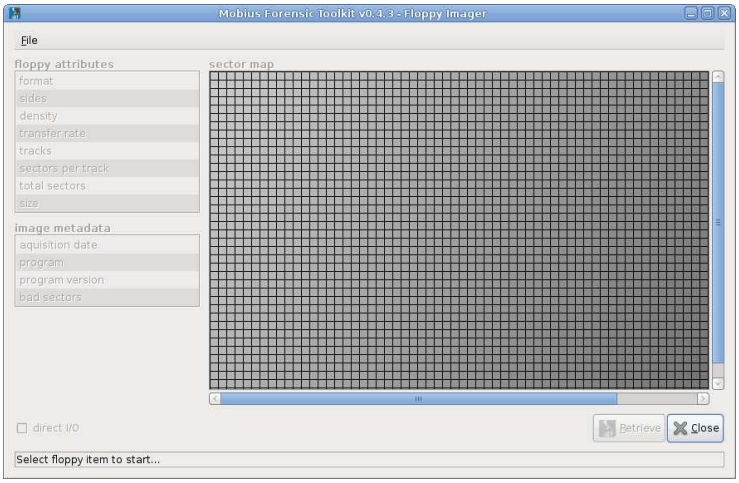

Figure 6.1: Floppy Imager extension

Click on any floppy item, insert a floppy into device `/dev/fd0` and hit `retrieve` button. Floppy Imager will collect and show floppy metadata. Each block on sector map represents a sector. Gray blocks are undefined, blue ones are sectors successfully read and red are bad sectors (figure 6.2).

Any floppy disk can be imaged more than once. If you select an already imaged floppy disk and hit `retrieve`, Floppy Imager will try to retry only bad sectors. Usually, if you eject and re-insert floppy disk, Floppy Imager will recover some bad sectors.

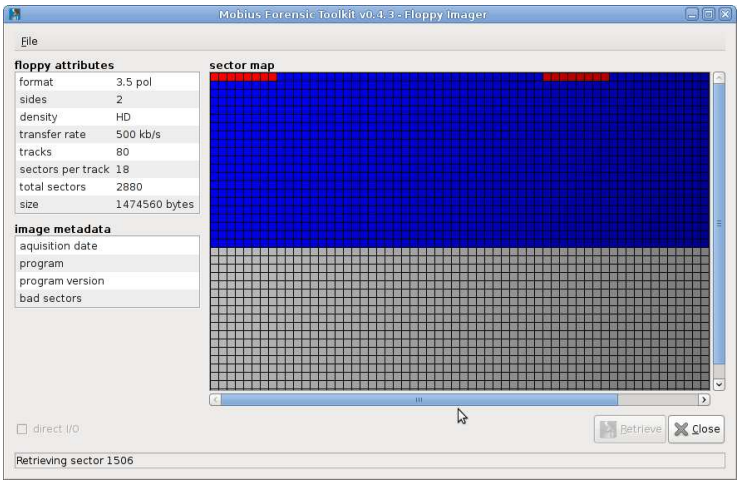

Figure 6.2: Floppy Imager running

When you have clusters of 8 bad blocks, usually only one of them is really bad, but as Linux read 4k at once, all eight are tagged bad. To recover most of this, mark option `Direct I/O`, which sets direct access to floppy driver. It is slower, but in most cases it recovers $\frac{7}{8}$ of bad sectors.

Floppy images are saved at folder *`casedir`*`/image`, where `casedir` is case basedir.

7 Developing extensions

The Mobius Forensic Toolkit is implemented through extensions. Each extension is a separated program that runs in its own independent namespace. Each one is coded in XML file, and could be developed solely using a text or XML editor. Nevertheless, there is an extension that was specifically made to that job: Extension Builder. It is a complete IDE that handles the underlying extensions and services structure, with code editing capabilities.

To start Extension Builder, click on `Extension Builder` icon in Mobius main window. A window like one shown in figure 7.1 will be opened.

7.1 Opening an extension

After you have started Extension Builder, click on `Open` menu option or on the corresponding icon in the toolbar, to open an extension.

Mobius Forensic Toolkit distribution files (`.tar.gz`, `.tar.bz2`, or `.zip`) have a directory named `extensions` where you can find all extensions that are distributed inside those packages. Feel free to open those extensions, and even to create new ones based upon their source codes. In this example, we have selected all extensions from `extensions` directory (figure 7.2).

To use an extension you have modified, you must install it using Mobius main window `tools` option.

7.2 Creating a new extension

As told before, you can open an existing extension, modify its source codes and save it as a new extension. But you can also start with a fresh new one. Click on `New` menu option or on the corresponding icon at toolbar, to create an extension.

Change your extension properties using `properties` option, and it will open up a dialog (figure 7.3).

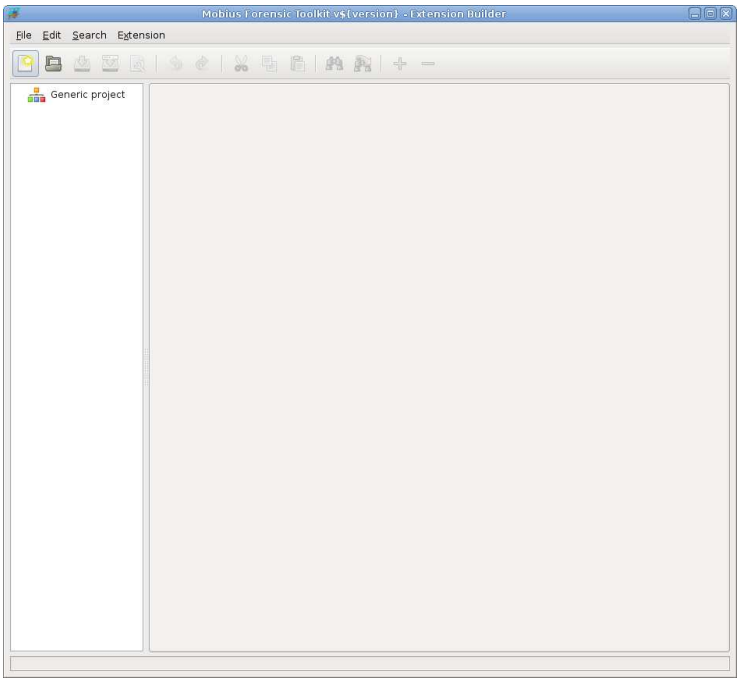

Figure 7.1: Extension Builder running

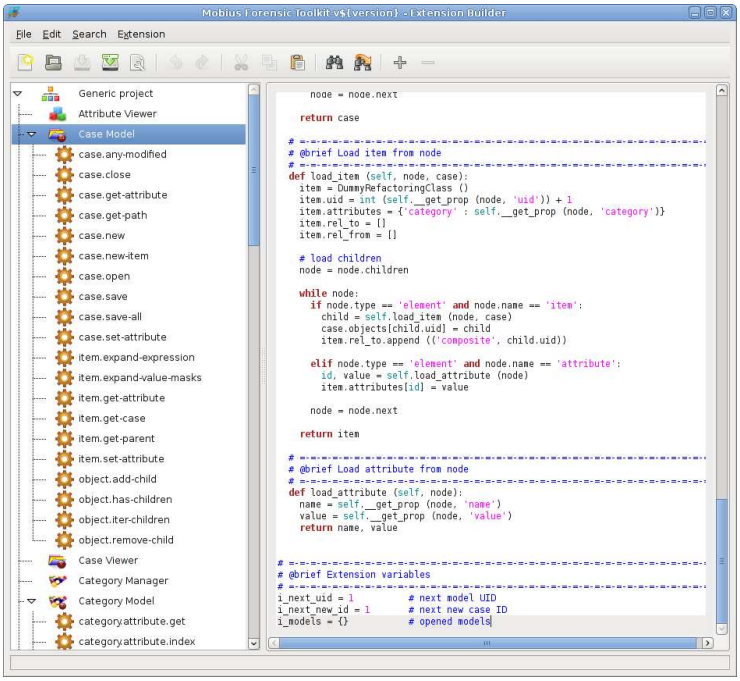

Figure 7.2: Extension Builder showing extensions

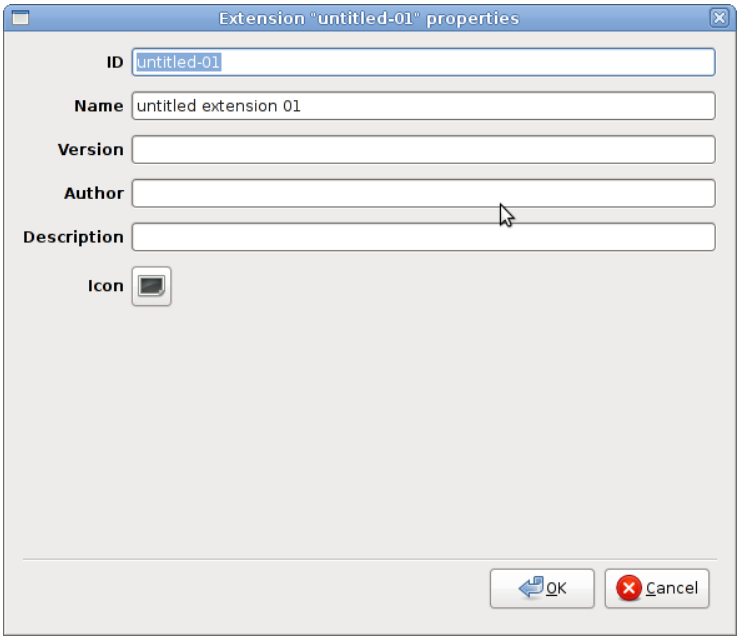

Figure 7.3: Extension Builder properties dialog