

OwlWatcher User Manual

Peter E. Midford

May 2, 2008

Contents

1	Introduction and Motivation	3
1.1	Ontologies and Communication	3
1.2	Getting Help	4
2	Getting Started	5
2.1	Installing OwlWatcher	5
2.1.1	Windows	5
2.1.2	Mac OSX	5
2.1.3	*nix	5
2.2	First time configuration	6
2.3	QuickStart with the Demo Package	6
2.4	Creating a project	8
2.5	Loading a base ontology	8
2.6	Loading a video clip	9
2.7	Scoring	10
2.8	Saving your work	11
3	Working with Projects	12
3.1	Project Tab Commands	12
3.1.1	Menu Commands	12
4	Observing with OwlWatcher	14
4.1	Watch Tab Commands	14
4.1.1	Menu Commands	14
4.1.2	Watch Tab Buttons	15
4.2	Working with the class display area	16
5	Working with Ontologies	18
5.1	Ontology Tab Commands	18
6	Working with Timelines	19
6.1	Timeline Tab Commands	19
7	Using Markov Models for larger data sets	20
7.1	Why Markov models?	20
7.2	Markov Tab Commands	20
8	Plugins	21
8.1	CSV Export	21

1 Introduction and Motivation

OwlWatcher is designed to help people observing behavior to record their observations using an ontology. If you are reading this, you most likely have an interest or reason for observing and recording behavior, but the ontology part is probably less clear. This section will try to explain why an ontology is a useful framework for recording behavior and what sorts of projects are most likely to benefit from using an ontology. Although OwlWatcher provides a number of tools that will be useful to anyone scoring behavior, there are a number of other free and commercial applications that may be more appropriate to behavior studies that wouldn't benefit from using an ontology.

An ontology starts with a collection of terms. For the moment think of a term as a word or phrase, that for our purposes refers to a type of behavior or a concept you would use in describing behavior. For example, `raise head`, `walk`, or `sidle display` could appear in a behavior ontology. So could terms like `head`, `repetition frequency`, or even `reproductive function`.

1.1 Ontologies and Communication

Ontologies, in the sense used by computer scientists and bioinformatics researchers, are a way to share information: between humans and computers as well as between different software applications. OwlWatcher is designed to work with several ontologies simultaneously. Most frequently, you will start with an ontology of behavior terms and augment with additional ontologies that would supply terms for anatomy, for time, or for representing biomechanics. OwlWatcher maintains its own ontology of concepts that it understands what to do with. These concepts include other ontologies, both user created and imported, data sources and viewers (e.g., video files and players), and some temporal concepts. This list of internal concepts will grow as OwlWatcher becomes more capable.

OwlWatcher will also have a mechanism to allow you to map concepts from external ontologies to terms in OwlWatcher's native vocabulary - this will allow OwlWatcher to work with and reason with the other terms when it encounters them in ontologies you construct.

1.2 Getting Help

If you have any difficulties with OwlWatcher, please do not hesitate to contact me: Peter.Midford@gmail.com

2 Getting Started

2.1 Installing OwlWatcher

Getting started with OwlWatcher requires three things:

- A copy of the OwlWatcher software
- A source of behavior (usually video, but could be a live animal or written notes)
- Connection to the internet, but only for setting up the application for the first time

If you are planning to work with video, you will also need an appropriate video player library. This could be QuicktimeTM for MacIntosh or Windows, or, in the future, another supported library¹.

2.1.1 Windows

Installing OwlWatcher on Windows requires Java 1.5 or better. The only supported video at this time is QuicktimeTM which you will need to download separately. There is now a simple windows release `OwlWatcher.zip`. To install, just unzip the zip file and run the `Owlwatcher.bat` file in the top level folder. Note: if you run the version in the embedded `bin` folder, plugins will not load and the application will probably crash.

2.1.2 Mac OSX

Installing OwlWatcher on Mac OSX requires OS 10.4, and Java 1.5. It has been tested with Quicktime 7, though it might work with Quicktime 6. Simply download the `OwlWatcher.dmg` file, open it, and drag the OwlWatcher application to a suitable folder (e.g., the Applications folder in the system area or under your home folder). Simply double click the application to get started.

2.1.3 *nix

Installing OwlWatcher on a *nix type system requires a Java JRE running 1.4.2 or better. Currently there are no supported video players, though this will change. To install, download the `OwlWatcher.tgz` file, and from a

¹Suggestions for such players, particularly for *nix systems are welcome in the project's forum on [sourceforge](#).

console window, unpack it with `tar -xzf OwlWatcher.tgz`. In the resulting directory, you can start OwlWatcher by executing the `OwlWatcher.sh` script.

2.2 First time configuration

The first time you start OwlWatcher, the application will configure itself and will display the preferences window. Currently the only options on the preferences window are the location where OwlWatcher will keep local copies of ontologies you read across the net, and a display option for classes. In most cases you can just leave the defaults as they are, and choose the OK button on the window.

2.3 QuickStart with the Demo Package

For a quick introduction to what OwlWatcher can do, download the Demo Package from the link on the OwlWatcher homepage (<http://ethotools.sourceforge.net/owlwatcher>). When you open the archive, you will find three files:

- `readme.txt` A summary of these instructions
- `jays.mov` a QuicktimeTM movie file with 20 seconds of Florida scrub-jay behavior
- `abocore.owl` The preliminary (no definitions) version of the ABOCore ontology, translated into OWL

Here's how to use them.

Download and install the latest OwlWatcher release and download and unpack the Demo Package archive into a convenient location. Then start up OwlWatcher. In the file menu choose “New:Project...” and specify a project file, which should use the extension `.rdf` (e.g., `test1.rdf`). Next choose “Open:Ontology from File...” and in the file dialog that appears, locate the `abocore.owl` ontology file and click the open button. Now the name of the project and ontology files appear in the main pane.

As you may have noticed, OwlWatcher starts with the “Project” tab selected. Now that the project and ontology files are selected, switch to the “Watch” tab. When you first enter the tab, all you will see are a couple of buttons. Ignore them for now, instead go to the File menu and chose “Open:Video from File...” and in the file dialog locate the `jays.mov` file and click the open button.

After a moment, the window should change and show two panes. On the right you will see the first frame of the video clip. Below the clip are some

motion control buttons. On the left side, you will see an empty white pane with the word “Root” at the top left.

Click once on the word “Root” to select the pane then double-click to open the tree (you will see the word “Model” appear below “Root”). If the double click causes a text edit box to appear around root (or any other tree node) press the escape (‘esc’) key to cancel the edit.

Click on the triangle next to “Model” to open up the tree further. The tree contains all the terms in the ABOCore ontology. For more information about the ABOCore ontology itself, refer to the website www.ethodata.org. For the purposes of this demo, locate the concept “Move head” in the ontology (its ‘path’ is Root:Model:root of:Behavioral Acts:Body part movement:Move head).

The video player currently has only four controls: back one frame, stop, forward one frame, and run. Start the movie and let it run until the first jay is in the ring and starts digging (around 5.44 seconds into the clip).

Suppose you wanted to record the bill sweep that occurs in the following two frames. First you should create a subclass of the appropriate existing term. I would suggest “Move head” as the most appropriate superclass, since the bill is not opening or moving relative to the head. So, open the Move head concept and see the three concepts below it. “Shake Head” is the most appropriate of the three subclasses. To create a new subclass of “Shake Head”, start by selecting the add subclass button. This is the left most of the three buttons above the pane with the tree, the one with the green plus symbol in the box. You can also use the “Add subclass...” command in the ontology menu. Once you have selected the command, click once on the “Shake Head” concept. A little dialog will appear and ask you to name the new subclass. Call it Bill Sweep (no quotes) and click the OK button. Now a triangle appears next to “Shake Head”, which, when selected will show the new subclass. Now go back to the video and step the clip forward and backward until you find a frame with a bill sweep. Now just click the “Bill sweep” concept to record an instance of this type of event. You can step forward and find further examples of this class to record.

Now that you have recorded some behavior events, save your project and data, using File:Save. Now if you look in the location where you saved the project file, you will find two new files, one with the extension `.rdf` and the other with the extension `.owl`. The first of these contains project settings, and the second one contains your new class and records of whatever events you recorded. If you open up the OWL file in an editor, you may be able to find the definition of Bill sweep as a subclass of Shake head and a number of time stamped instances of Bill sweep. If you are familiar with OWL editors

such as the Protege OWL plugin or SWOOP, you can look at the file there as well. In Protege, use the triples view to see the time stamps for the individuals.

You can also export a list of events scored in the project to a “comma-separated value” (.csv) file. Files in this format can be loaded into Microsoft Excel™ or most statistical programs. To do this, chose “CSV Export” from the Export submenu of the File menu. After showing you a file dialog so you can specify the location of the CSV file, the program will show you a small option box with two options. The ‘Sort by time’ option should be self explanatory, especially since the event time will be listed in the first column of the data file. The second column will contain the type of event that was recorded as having happened at that time. If the ‘List full hierarchy’ option is selected, then the classes above the event’s class will be recorded in columns extending to the right in the CSV file, as many as are needed. For example, the columns for a Shake Head event event would contain (left to right) the time, Shake Head, Move Head, Body part movement, Behavioral Acts, root of, Model, and Root. If the ‘List full hierarchy’ option is not selected, only two columns will be listed. Choose the OK button to save the CSV file and look at it, either in a spreadsheet or text editor to see how it works.

To reload a previously saved project, choose “Open:Project From File...” on the File menu. The command currently reloads the project and ontology file(s), leaving you ready to select the Watch tab and load the video. The project file will save more editor state (e.g., the video file and what tab you were last using) in a later release.

2.4 Creating a project

After the configuration process is over, the main OwlWatcher window will look like one of the following screens, depending on your computer’s operating system. Now chose new project from the file menu. In the file chooser dialog that appears, enter a file name and include the extension “.rdf”. For example, demo.rdf.

2.5 Loading a base ontology

Now you need to tell Owlwatcher what ontologies you will be using to define the behavior terms you create as you observe. For this example, you can just use the ABOCore ontology, for which an OWL translation has been developed as part of the Ethotools project. There are two ways to tell Owl-

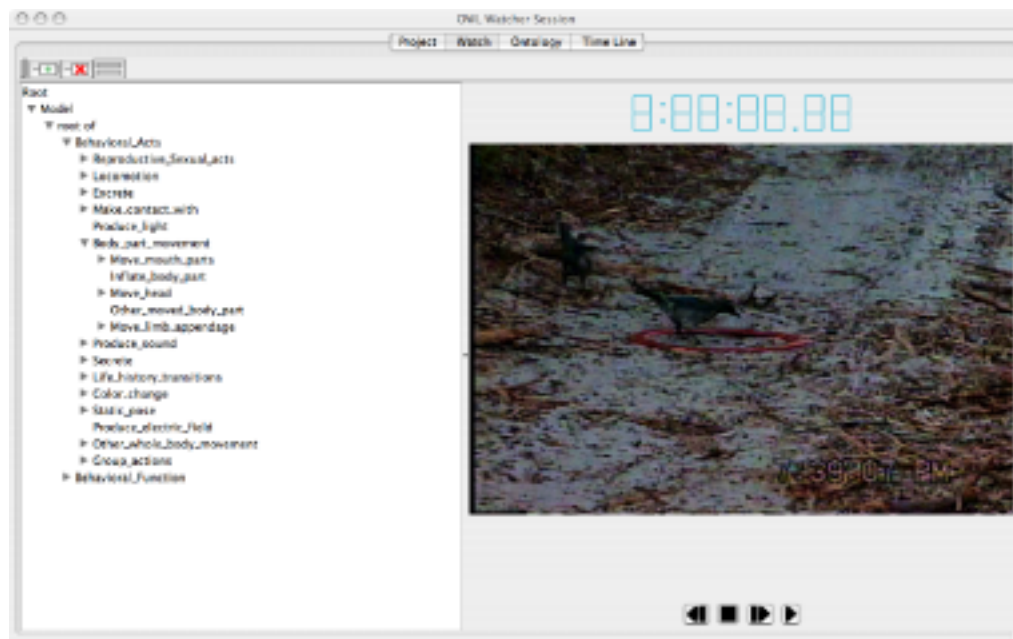
Watcher where to find an ontology: either on the web using a URL, or in a local file. You can get a local copy of the OWL translation of ABO-Core in the demonstration package, available from the OwlWatcher home page, or you can just specify the URL of a copy on the OwlWatcher website: <http://ethotools.sourceforge.net/owlwatcher/ontology/abocore.owl>.

If you are wondering about whether to use a local copy or web-based copy, the most important consideration is whether you will have an active internet connection available while observing. If you, you need to have a local copy. It might also be preferable if your connection is unreliable or slow. On the other hand, using a web-based copy allows for possible updates - which may or may not be a good thing.

In OwlWatcher 0.35 and above, when a web-based ontology is first loaded, the program will save a copy in a special 'cache' location. It will use this saved copy in subsequent sessions if it encounters problems with connecting to the internet. The current version of OwlWatcher doesn't check for updates to web-based ontologies. If you wish to update, the easiest thing is to use the preferences menu to find the cache location and then walk down the directory (folder) structure to find the cached copy and delete it. The program will then load the newer version and save a copy when it next loads an ontology that uses it.

2.6 Loading a video clip

Once OwlWatcher has a project and an ontology, it will list the names in the *Project Pane*. If, as in this example, the session will use only one base ontology, you can proceed to the *Watch Tab* by selecting it. The watch tab contains a number of panes. To see them fully, you will first need to load a video clip. For this example, you can either use a QuickTime video of your own, use the 20 second clip of Florida scrub jay behavior (Midford et al. 2000). This clip is available in the demonstration package, available from the OwlWatcher home page.



Once the view clip is loaded, the Watch Tab will fill in its two panes. In the left pane, trees for each ontology you loaded into the project will appear. Currently, only the root of each ontology will appear. Next to the root will be an icon (varies based on what operating system and computer you're using). Click the icon to expand the tree below the root. You can continue expanding using the icons next to subnodes in turn. Above the panel showing the trees, there is a button menu.

The right pane will have three panels. The middle panel will show first frame of the video you loaded. The top panel will show the elapsed time, which should be 0. The bottom panel will contain buttons to control video playback.

2.7 Scoring

Scoring with OwlWatcher consists of playing the video and recording events by clicking on their types in the tree. For example, if you saw a scrub jay taking a step, you could score a `Move limb appendage` event by clicking on the term in the tree. Not fast enough to keep up with the video? You can stop the video at any point and step through frame by frame using the buttons on either side of the square stop button below the video. Step to where you want to record the event and then click on the term.

What if you want to define a new type of action, one that refines a term

in the tree? That's what the left-most button, the one with a green "plus sign" is for. For the system to understand you, you must choose the button first. When you press the button, it changes color to show that it's active. If you change your mind, just press it again. Otherwise, just chose the term you want to extend and click on it. When you do so, a small window will appear to allow you to enter an id for the new term. Press the window's Ok button and the term will be added to the tree.

You can use the second button, the one with the red X to delete terms you've added in the same way. Again, select the command first, then the term.

The third button will, in the future, allow switching between the tree view and a view that only shows buttons for each term at the tip of one of the trees. This should support faster scoring, which could be especially useful for scoring real time events.

2.8 Saving your work

When you are done scoring the video, use the save command from the file menu. This will save *two* files: the first, with an `rdf` extension, contains project information; the second, with a `owl` extension, contains your data, including all events and any classes you defined. It won't contain all the terms you saw on the tree, those are defined in the ontology files you loaded at the start. If you are familiar with an ontology editor, such as Protege or Swoop, you can use it to look at your data file. If you are using Protege, you need to have the owl plugin installed.

You can also export a list of events scored in the project to a "comma-separated value" (`.csv`) file. Files in this format can be loaded into Microsoft ExcelTM or most statistical programs. To do this, chose "CSV Export" from the Export submenu of the File menu. After showing you a file dialog so you can specify the location of the CSV file, the program will show you a small option box with two options. The 'Sort by time' option should be self explanatory, especially since the event time will be listed in the first column of the data file. The second column will contain the type of event that was recorded as having happened at that time. If the 'List full hierarchy' option is selected, then the classes above the event's class will be recorded in columns extending to the right in the CSV file, as many as are needed. For example, the columns for a `Shake Head` event event would contain (left to right) the time, `Shake Head`, `Move Head`, `Body part movement`, `Behavioral Acts`, `root of`, `Model`, and `Root`. If the 'List full hierarchy' option is not selected, only two columns will be listed.

3 Working with Projects

This section provides a detailed discussion of the commands and operations available in the Projects page.

Note that the current version of OwlWatcher does not distinguish between events and states. It seems that the proper ontological implementation of that will require the use of restrictions. Eventually there will be a 'built-in' ontology that will provide terms for **event** and **state**. You will then specify whether a particular action term is an action or a state by forming a restricted subclass an action term. The restriction will specify that the subclass is temporally either an event or a state. If you choose to use a different ontology for time, you will be able to form a mapping between the imported temporal ontology and the built-in one, so the OwlWatcher will know whether the restriction defines a state or an event.

3.1 Project Tab Commands

3.1.1 Menu Commands

- File
 - New
 - * **Project...** This command allows you to specify the location of a file that will contain the project information. The command operates by bringing up a file chooser window. Select the appropriate location, and specify a file name that uses the file extension **.rdf**. For example **jays.rdf**.
 - Open
 - * **Project From File** This command allows you to specify the location of an existing file that contains project information. The command operates by bringing up a file chooser window. By default, it filters the file display to only show files with the file extension **.rdf**. Select the file and use either the open or cancel button. The open button will cause OwlWatcher to attempt to load the project, its associated data file, and any associated ontology files. If successful, the large display pane will list the name of the project file and those of any associated ontology files that were loaded.
 - * **Project From URL** This command allows you to specify the URL of a file that contains project information. The com-

mand operates by bringing up a single line text box for entering the URL. After entering the URL, use either the open or cancel button. The open button will cause OwlWatcher to attempt to load the project from the location specified by the URL. It will also attempt to a data file (same URL except for changing the file extension) and any associated ontology files. If successful, the large display pane will list the name of the project and those of any associated ontology files that were loaded.

* **Ontology From File**

* **Ontology From URL**

- **Plugins** This will display a window listing the loaded plugins. Next to each plugin is a checkbox allowing it to be disabled. Available plugins are described in Chapter 8.
- **Save** This command saves the project and its associated data into their default locations: `projectname.rdf` and `projectname.owl` respectively.
- **Save As...** This command allows you to save the project and its associated data to a new location. The command operates by bringing up a file choose window. Navigate to the desired location and specify the new name, which, as above, should end with the extension `.rdf`. The data file will be saved to the same name and location, but with the extension `.owl`.

- **Edit**

- **Window**

- **Help**

- **Project Pane Help**

- **Help Contents**

- **Show License** This command displays the open source license(s) applicable to OwlWatcher.

- **About OwlWatcher** Displays an about window.

4 Observing with OwlWatcher

This section provides a detailed discussion of the commands and operations available on the Watch page. The most important thing to understand about the operation of the watch page is that its purpose is to record events. This is not the place to edit the event hierarchy, with the exception of adding or deleting a new class. Filling in details of the ontology occurs in the ontology page, or optionally in an external editor.

4.1 Watch Tab Commands

4.1.1 Menu Commands

- **File**
 - **Open**
 - * **Video From File...** This command allows you to select a file containing a video clip you wish to annotate. At least one of the video players installed in OwlWatcher must be able to read, and hopefully play, the file you select. As of version 0.03, this means a format that the Quicktime plugin is capable of playing. The command will bring up a file dialog that allows you to select the file containing the video. After the file is selected, OwlWatcher will load the file and, if successful, the contents of the pane will change. The left side will contain a white window containing the concept hierarchies in the ontologies you specified in the project page. The right side will show the video player and the first frame of the video.
 - * **Video From URL...** This command operates in the same way as **Video From File** except that the file's location is specified by typing a URL into a text box, rather than choosing a file from a dialog.
 - **Start Video** This command runs the video and is equivalent to the play button in the viewer pane.
 - **Save** This command saves the current ontology and the associated project file back to the files they were loaded from. This will not save any of the imported ontologies, since OwlWatcher will not have modified them.

- **Save As** This command puts up a dialog to allow you to specify a file where the project file will be saved. This is will be a .rdf file. It will then ask whether you wish to save the ontology as well - in almost all cases the answer to this question will be yes. Do not save the project as a video file, though that may come up as a suggested default.
- **Export** If any export plugins, such as the CSV format exporter, are available, they will appear as submenus of this command. Their specific behavior will vary, but in most cases they will prompt both for a file to export to and for options specific to the particular format. For example, the CSV format exporter will prompt whether to sort the rows by time and whether to list the entire concept hierarchy to the right of the instance that heads each row.
- **Ontology** Currently this menu provides items to add subclasses or delete terms. At present they operate equivalently to the buttons in the button row.

4.1.2 Watch Tab Buttons

- **Ontology Pane Buttons**
 - **Add Term Button** This button, with an icon consisting of a green plus symbol in a box, is used to add a subclass to an existing term. The button must be selected first, followed by the term in the hierarchy to receive the subclass. The new term will then appear as a subclass of the parent. Note that only terms created in this way are defined within the project and can be subsequently deleted.
 - **Delete Term Button** This button, with an icon consisting of a box overlaid with a red x, is used to remove an existing term. Term deletion is now fully implemented, but OwlWatcher will not let you delete terms that have either subclasses or instances defined, or that were imported from another ontology (since you likely wouldn't be able to save that change to the imported ontology).
 - **Toggle Button View Button** The button will, in future versions, allow toggling between the current tree hierarchy view and an 'flat' view, consisting of an array of rectangular buttons, limited to those terms visible in the tree when the view is toggled.

- Viewer Pane Buttons
 - Step back This button, when the player is stopped, causes it to back up approximately one frame. The exact behavior will be dependent on your choice of operating system and installed video player, as some players offer finer control over playback.
 - Stop This button stops the playback of the clip.
 - Step forward This button causes the player to move forward approximately one frame. It suffers the same limitations as the step back button.
 - Play normal rate This plays the clip at the standard rate until either the end of the clip is reached or the stop button is pressed.

4.2 Working with the class display area

The class display area displays the available set of terms. By default, the display consists of one or more hierarchical trees. Each tree corresponds to an imported ontology - terms in the current ontology will appear as leaves from one or more of the trees of imported terms. Imported terms appear in grey, whereas the terms you add to the current ontology will be in black. There are four basic operations you can perform with the class display:

- Use the display widgets on each branch of the tree to expand or collapse branches in the tree.
- Click a class term with the mouse. If you do this without previously selected a button in the bar above the pane, then OwlWatcher will create an instance of the term specified and timestamp it with the current time available in the viewer. This is how you tell OwlWatcher that something of this type occurred at this time. OwlWatcher will use italic script to indicate terms that have one or more instances.
- Select an button on the pane, then select a term with the mouse to perform the action. In the current release, this means you can select to either add a subclass or delete a class term. Although you can add subclasses just about anywhere on the trees, OwlWatcher does not allow you to delete classes that have either subclasses or instances. Delete these first if necessary.
- You can use the right mouse button (or control-click on OSX) to bring up a set of menu commands to immediately operate on the term (e.g.,

add subclass or delete). The menu will also allow you to view instances of the class if any are available. If none are available a message dialog will appear, otherwise, a dialog with a drop-down list of instances (displayed as termname@time) will appear. Currently the time is specified in units specific to the media file. You can either just view the instances or choose to delete them - currently only one at a time deletion is supported.

5 Working with Ontologies

The ontologies tab will allow you to relate your behavior instances to other ontologies.

5.1 Ontology Tab Commands

6 Working with Timelines

6.1 Timeline Tab Commands

7 Using Markov Models for larger data sets

7.1 Why Markov models?

7.2 Markov Tab Commands

8 Plugins

Currently OwlWatcher supports one plugin, the CSV exporter.

8.1 CSV Export

The CSV Export plugin allows you to export behavior instances you have recorded to a “comma-separated value” (.csv) file. Files in this format can be loaded into Microsoft Excel™ or most statistical programs. The file will contain one row for each instance, but the number of columns may vary. The first column is always the time, which is current displayed whatever units the video clip and player use internally. For Quicktime™ this is 3000 units per second or 100 units per frame. The second column is the behavior class you scored this event as. Third and later columns may contain the class hierarchy above the selected class if selected as an option. The file currently does not include headers. When the exporter is selected, a dialog appears allowing you to specify where to save the file. After specifying the file’s location, the plugin puts up a small dialog with two checkbox options. To do this, chose “CSV Export” from the Export submenu of the File menu. After showing you a file dialog so you can specify the location of the CSV file, the program will show you a small option box with two options. Sort by time sorts recorded actions by time. List full hierarchy includes the class hierarchy of the scored behavior class in the third and subsequent columns of the .csv file. For example, the columns for a **Shake Head** event event would contain (left to right) the time, **Shake Head**, **Move Head**, **Body part movement**, **Behavioral Acts**, **root of**, **Model**, and **Root**. If the ‘List full hierarchy’ option is not selected, only two columns will be listed. .