

STARgui User's Guide

Jonathan Frain
Andrew Foulks
R. Daniel Bergeron
April 5, 2008

1. Introduction

1.1. STARgui overview

The *STARgui* tool allows a user to generate a multiresolution hierarchy from an array based data set or a time series of array based data sets. The gui is actually a front end to a command line program called *makear*. *STARgui* leads the user through the process of defining the desired combination and ordering of both spatial and temporal resolutions that the user wishes to include in the output hierarchy. *STARgui* generates appropriate scripts that are used as input to the *makear* utility. For very large data sets, *makear* can take a significant amount of time (sometimes it will be hours). Consequently, *STARgui* starts up the *makear* scripts in the background. In principal, the user doesn't really need to know about *makear*, except to know that it exists and that it is actually generating the data hierarchy, rather than *STARgui*.

1.2. The STAR data model

STARgui is one component of a suite of software tools aimed at supporting the generation and use of multiresolution data hierarchies for time series data sets. STAR is an acronym for **S**patio-**T**emporal **A**daptive **R**esolution data. The STAR data model incorporates the idea that a time series data set can be represented at many different resolution levels using arbitrary combinations of spatial and temporal decompositions. We assume that the time series data is stored with one data file per time step per data attribute. From this data we can generate a lower resolution representation of that data by applying either a spatial or temporal data reduction.

Given a time series of spatial data, we perform a *temporal* data reduction by applying a one-dimensional time series data reduction for each spatial position. In other words, for each spatial position and for each attribute at that position, we create a 1D time series for the attribute and apply a wavelet transformation (or some other data reduction technique). All the resulting time series are recombined into a single time series of spatial data of the same dimensionality and size as the original. This process is depicted in Figure 1(a), which shows how the wavelet transformation generates time steps at new time step positions.

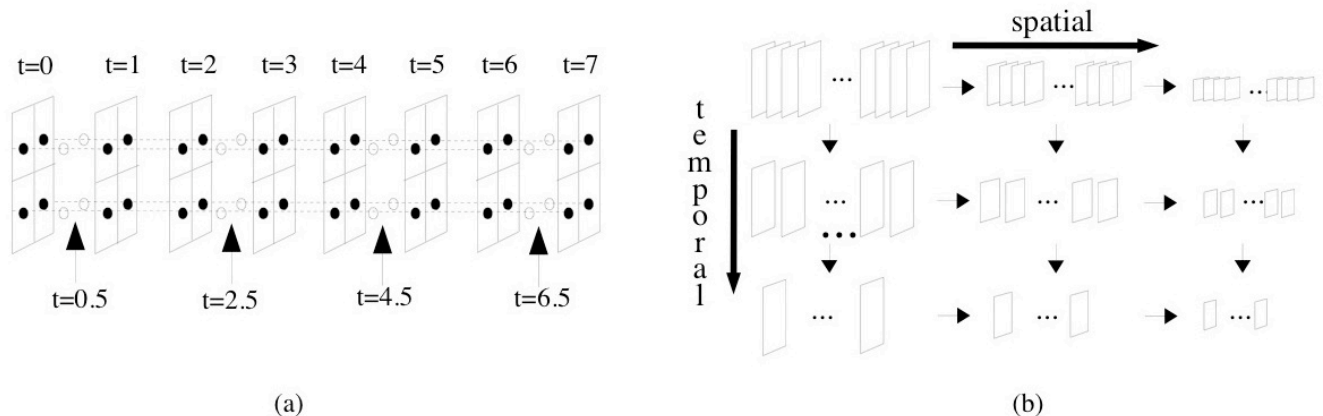


Figure 1: The STAR data model overview

To create a spatial data reduction, we apply a wavelet transformation to the data at each time step of the time series. A spatial wavelet transformation reduces the data size to $1/2^d$ of the original, where d is the dimensionality of the data. This size reduction is represented as smaller boxes in Figure 1b.

1.3. File organization conventions

Although the low level *makear* tool allows the user to generate lower resolution data representations using any desired file organization or naming convention, this flexibility has significant disadvantages in terms of creating a compatible set of tools to utilize the multiresolution hierarchy that gets generated. Consequently, the *STARgui* tool imposes a specific file organization convention upon the user. The original data (time series) must be stored in a directory named *data* inside a directory that is called the “root” directory of the multiresolution data hierarchy. If the user applies a spatial wavelet to the original data, it will be generated in a child directory of the “root” whose name starts with the string “space”, or the abbreviation “s”. If the user applies a temporal wavelet, it will be generated in a child directory of the “root” with a name that starts with “time” or “t”. The actual data files will be in a subdirectory of the space or time directories, named *data*.

1.4. Input data and metadata files

The *STAR* data generation utilities can handle any binary data files that are organized as 2- or 3-dimensional arrays. The data files themselves should contain only the actual data. Consequently, it is necessary to describe the characteristics of those files (data type, dimensionality, and dimension sizes, number of time steps, etc.) in an associated *metadata* file located with the actual data files in the *data* directory. If the metadata file already exists, *STARgui* will open it automatically. If it does not exist, *STARgui* will prompt the user to specify the required information and then save that information in one or more appropriate metadata files. The software actually uses slightly different metadata file formats, but *STARgui* hides that from the user.

2. Installation

Open the STAR website and navigate to the “DOWNLOADS” page. (Note: At the time this document was written, the STAR website was located at: <http://www.cs.unh.edu/sdb/star/>)

For questions related to installing STARgui with sample data, download and read the README.stargui document.

It is assumed that the user has access to the *STARgui* distribution release which is also found on the “DOWNLOADS” page as stargui.tar.gz. The major contents of the distribution are:

```
/stargui
  / sample_data
    /data
      TSD.tsd
      <data files>
  / makear
  STARgui.jar
  makefile
```

Simply follow the instructions in the readme to compile the C code needed for the application.

3. Using STARgui

1.5. Invoking the tool

To invoke the tool from the command line simply enter:

```
java -jar <pathToJar>/STARgui.jar
```

or, you can add <pathToJar > to your CLASSPATH environment variable and type:

```
java -jar STARgui.jar
```

If STARgui.jar is invoked in a directory that has a /data directory with a .tsd file inside the data directory, a window should pop up that looks like the image in Figure 2. (If the user invokes the executable from a location where a .tsd file is not found, a popup message will appear and after clicking 'OK' the user will see a window like that in Figure 3, except with the metadata fields all empty.) The key features to notice, in figure 2, are the Root Directory at the top of the GUI and the four tabbed panes titled *Tree View*, *Metadata*, *Script Debug* and *Multiple Build*. The user may change the location of the Root directory by typing in the absolute path into the text field, or by clicking the *Change Root Directory* button and navigating to the desired directory.

The Root directory setting depends on the operating mode described in the next section. Note: changing the Root directory will cause the program to automatically search for a 'data' directory and a .tsd file within that directory.

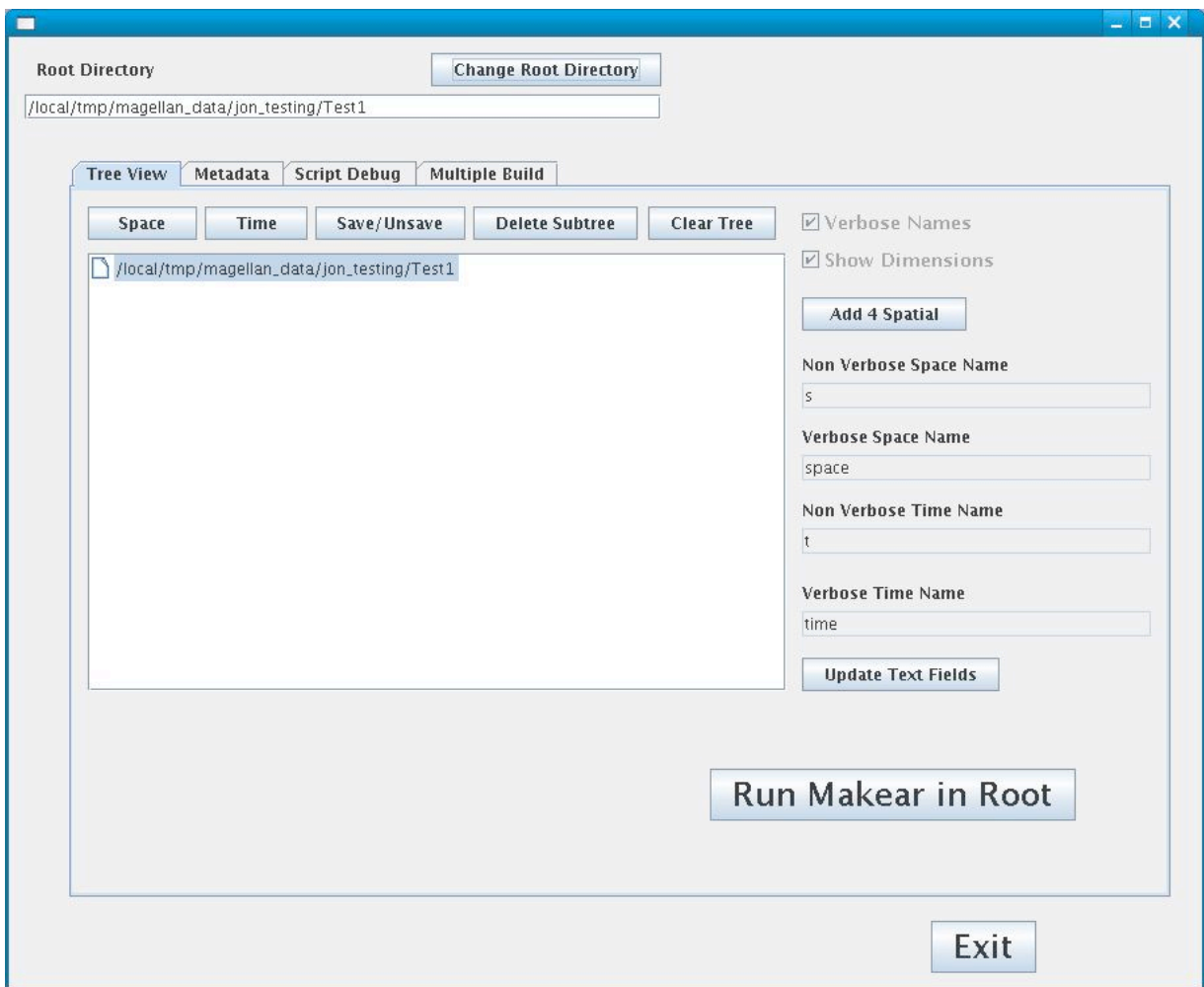


Figure 2: STARgui Appearance

1.6. STARgui modes

The program allows a user to create a multiresolution hierarchy for a time series defining a single attribute, or for a multi-attribute time series.

1.6.1. Single attribute mode

The single attribute mode runs the *makear* application on a single set of data, and thus in a single directory. The application assumes this mode of operation when invoked and looks for any files ending with “.tsd” (the extension that identifies a metadata file for the *makear* utility) in the *Root/data* directory. If the tool finds multiple files with that extension, it will force you to select the .tsd file you want to use. If no file is available, the program will prompt the user to specify the needed metadata information as described later.

Once the user has selected a .tsd file, the application opens it and loads the attributes from the file. Any attributes that it recognizes, and is looking for, will be appended to the text fields of the GUI.

Figure 3 shows the results after loading a .tsd file. The text fields are all editable, but those with white backgrounds are required values. Attempting to run *makear* with any of those fields blank will cause errors, and thus this application will not even attempt to do so. It will popup an error message informing the user of this.

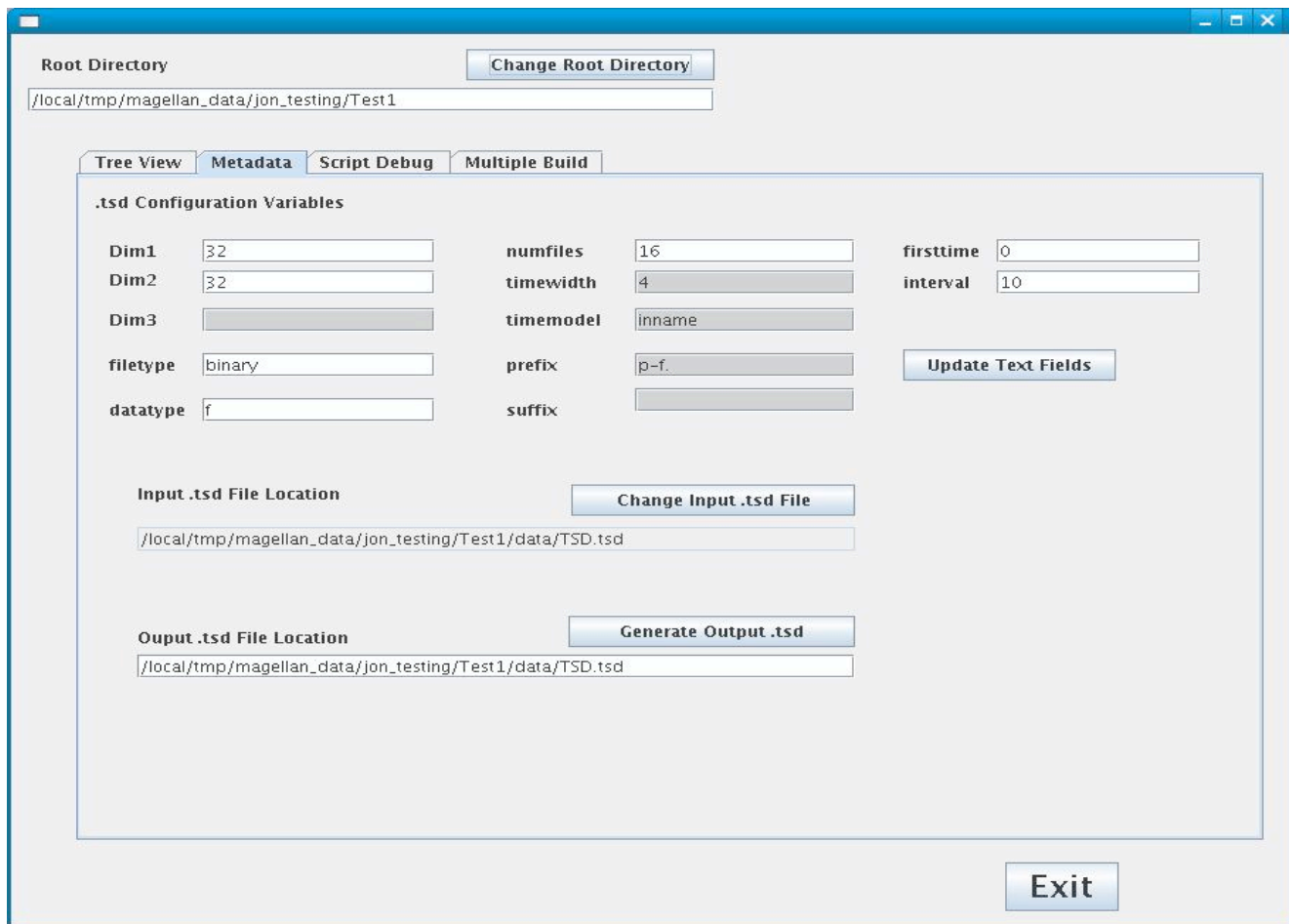


Figure 3: Metadata Tab

The user can define where and when to apply spatial or time wavelets inside the *Tree View* tab as seen in Figure 4. The application selects the Root Node by default, and you can recognize which Node is selected by the blue background color. Generating the tree will not cause *makear* to be invoked. It is not until the user hits the *Run Makear in Root* button that actual wavelets will be applied.

The GUI has been designed to only allow one node to be selected at any time. You can generate a script that will apply a time or spatial wavelet to any node, and the application will not check for errors. It is the users responsibility to make sure they generate a valid tree of wavelets for their given data.

If you look at the tree in Figure 4, you will note that it has properly used the data contained within the *.tsd* of Figure 3 to generate the spatial and time wavelets. The names of each Node are the exact names that will be used for the directories generated by the *makear* save command. By default, these directory names have verbose names and include the spatial and time dimensions. The user can turn off verbosity and the inclusion of dimensions by deselecting the check boxes. The user can also specify how the spatial and time directories will be labeled in both verbose and non-verbose modes.

The purpose of generating this tree is for the creation of a script file. This script file is piped to

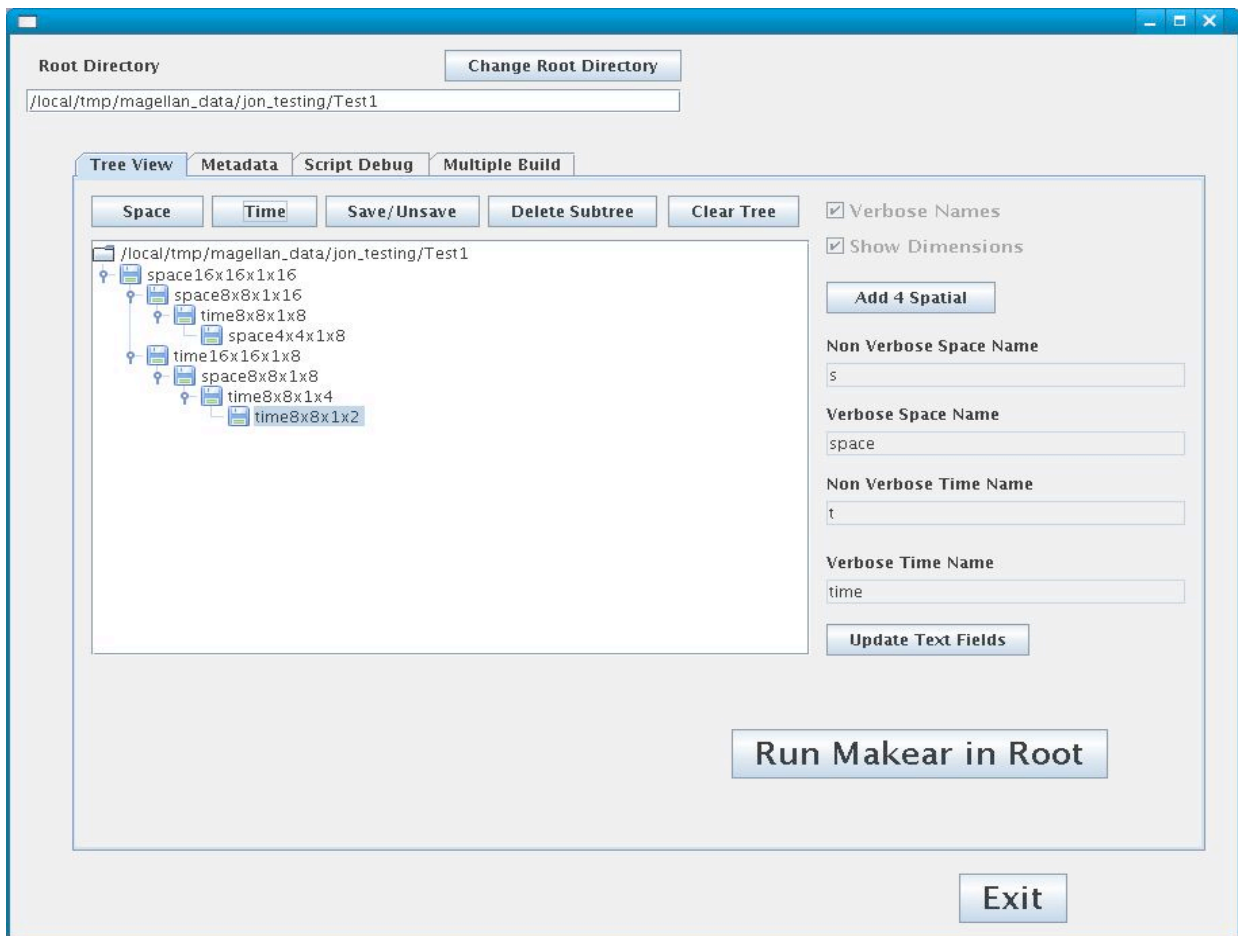


Figure 4: Tree View Tab

makear as input. Figure 5 shows the contents of the script file that will be generated from the user's tree. To simply view the script, without writing it to a file in the Root directory, the user can select the *only place script in text area* button to the right. The larger button at the bottom will place the script contents into the text area and also generate a new *makear.script* file in the Root directory.

When the user is satisfied that they have generated the proper script and also have the correct information for their *.tsd* file, they can go to the "Tree View" tab and hit the *Run Makear in Root* button (see Figure 2).

The tab titled *Multiple Build* has no real purpose when running *makear* in single attribute mode. If you have multiple attributes in multiple directories, it is normally most convenient to use the multiple attribute mode described in the next section. However, if you want to generate different hierarchies for different attributes, you can do that by defining each attribute separately using single attributed mode.

1.6.2. Multi-attribute mode

For time series data sets with multiple attributes, it is usually convenient to generate the same multiresolution hierarchies for all attributes or for selected subsets of the attributes. For instance, if the velocity information is stored in three separate dimensions, x, y and z, it is highly likely that all these attributes should have the same resolution hierarchies. The proper way to use this tool in that scenario is to set the Root to the directory that contains the directories that hold data.

Figure 6 shows the *Multiple Build* tab contents when the application is run from a directory with 5 subdirectories, dir1-dir4 and dir90. The only way to change the directories seen in the List on the left

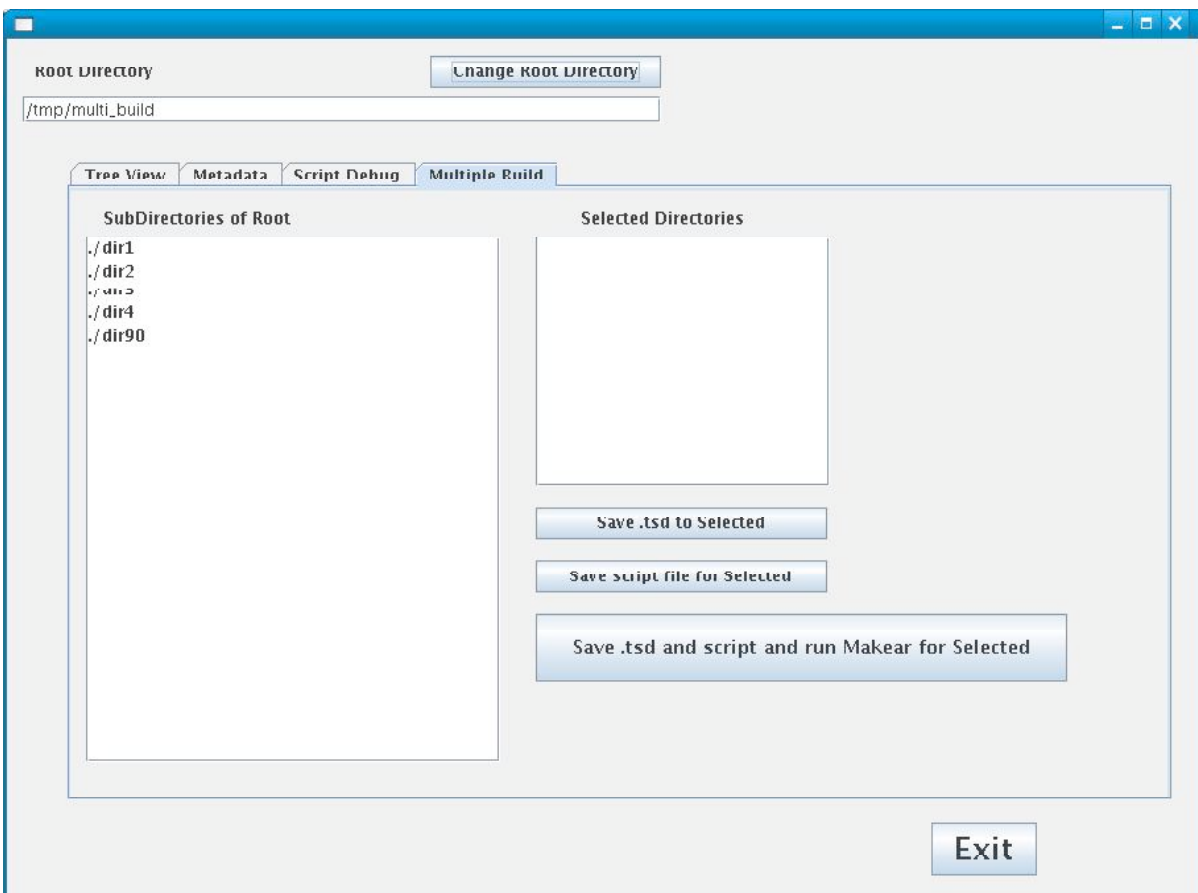


Figure 6: Multiple Build Tab

is to change the Root directory, as it is only looking at the subdirectories of the Root directory.

Upon invocation or a change in the Root directory setting, the application will look for a *.tsd* file in the Root directory or the Root/data directory. If no *.tsd* is found in either of those directories, it assumes that the user wishes to enter multi-attribute mode. In this situation, there will probably be *.tsd* metadata files in all the subdirectories and these need not be identical. The application requires the user to choose the desired *.tsd* file.

The user will still use the first three tabs as they had in single attribute mode. The difference is that the user must select the subdirectories on which to run the *makear.script* (see Figure 7). The following steps must be completed (in any order) to apply the same tree specification to multiple attributes

- 1) Generate the tree in the *Tree View* Tab
- 2) Open the proper *.tsd* file or manually enter the proper information in the fields
- 3) Select the directories on which *makear* will be run, with the specified tree and *.tsd* information

You will notice that there are three buttons in the *Multiple Build* Tab. The two smaller buttons are designed to allow the user to just generate either the *.tsd* or the *makear.script* files. This might be desirable if the user wants to run *makear* later, but generate the input files now. The large button generates each of these files for each selected directory and invokes the *makear* script in each of the selected directories.

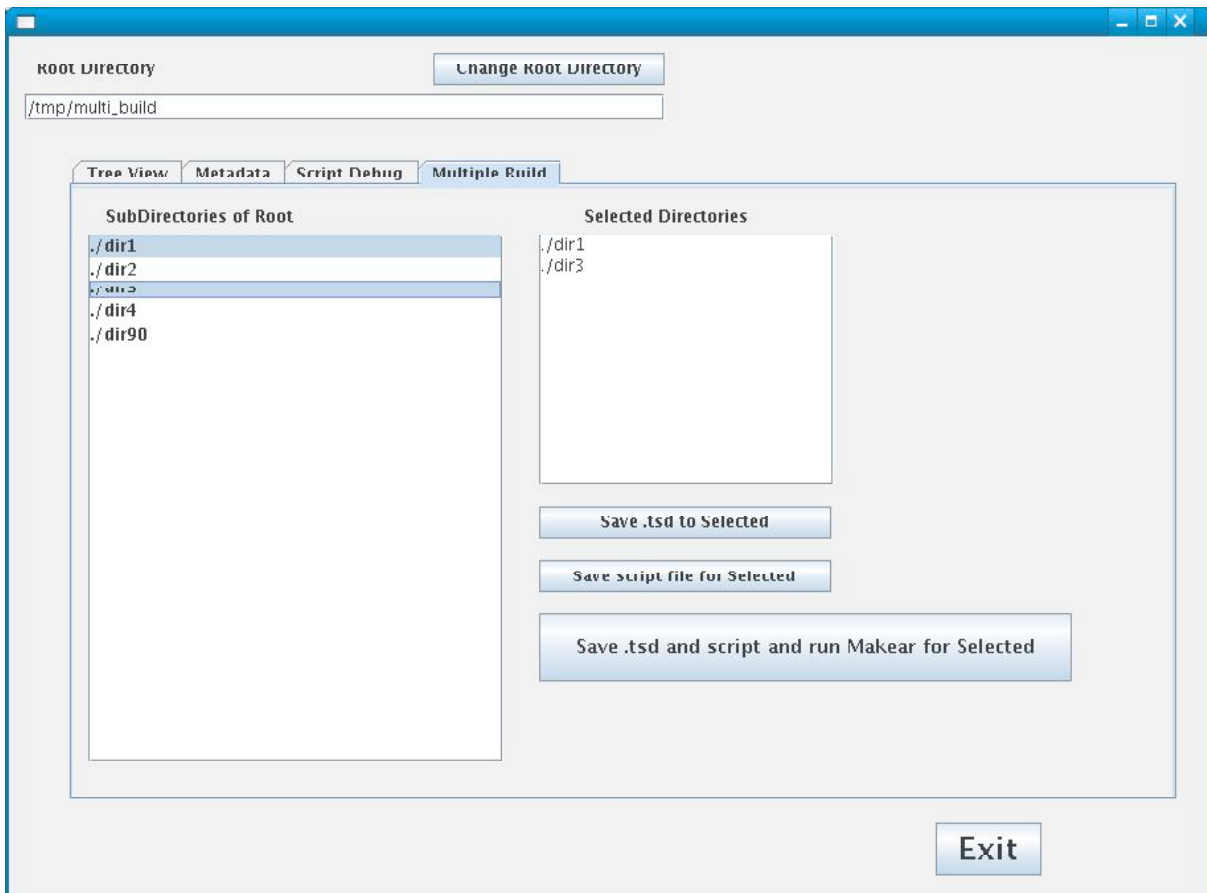


Figure 7: Multiple Build with selected directories

Notes:

1. The application will not run *makear* if the required TSD information is missing or incorrect.
2. The application will pop up a warning message if the user is about to overwrite an existing file. The user can turn off this functionality by adding *nowarnings* as an argument to the application:
\$java -jar <pathToJar>/STARgui.jar nowarnings.
3. The *nowarnings* option only suppresses the generation of the interactive popup warning dialog box. This is desirable when the user is running *makear* on large data sets in multiple directories. Since the program runs *makear* in a serial fashion one directory at a time, the popup may only

appear after after the user has gone home for the night and then execution will pause until someone closes the dialog box! The warnings will still be generated to the terminal

4. The application specifically looks for and expects to see a data subdirectory for every selected directory. If the selected directory doesn't have a data subdirectory, it will not copy the .tsd and script files and it will not invoke makeear. Instead, it will move on to the next selected directory, assuming there is one.

4. Installing and running STARgui with sample data

This section simply gives a quick overview on how to install and run the STARgui tool with the provided sample data. For a more in depth discussion of how to use the Tool, please refer to the previous sections.

- 1) Extract all of the files from the STARgui.tar.gz file as detailed in the Installation section.
- 2) Invoke the jar as explained in section 1.5
 - For example:
 1. Let's assume that "stargui.tar.gz" has been downloaded to the following directory on your machine:
`/MyDir/`
 2. The next step is to change to that directory by typing the following at the command prompt(%):
`% cd /MyDir`
 3. If you type the following at the command prompt, you should see 'stargui.tar.gz' in the directory:
`% ls`
 4. Extract the distribution by typing the following:
`% tar xvfz stargui.tar.gz`
 5. If you type the following at the command prompt, you should see a new directory called 'stargui'
`% ls`
 6. Type the following to enter the newly created directory:
`% cd /stargui`
 7. You now need to compile the C code that is contained within the distribution. To do this simply type. (It is assumed that the user has GCC installed on their machine):
`% make`
 8. If everything compiles fine you are now ready to run with the test data.
 9. You can invoke STARgui from any location on your machine. If you don't know how to set your classpath variable, you can simply invoke it by giving the exact location where it was extracted to. In the case of this example it would be invoked by:
`% java -jar /MyDir/stargui/STARgui.java`
 10. Depending on where you entered the previous command, you may get a popup window stating that a .tsd file could not be found. To make the tool see the sample data and it's accompanying .tsd file, simply change the Root Directory at the top of the GUI to be the location of the sample data. In the case of this example, the Root directory would be changed to:
`/MyDir/stargui/sample_data`
 11. Upon changing the Root Directory to point to the sample data, the tool should recognize the .tsd file that was provided and fill in the metadata fields as found in Figure 3.

12. Navigate back to Tree View tab and apply some spatial or temporal wavelets by using the Space and Time buttons.
13. When you have created the tree as desired, hit the “Run Makear in Root” button at the bottom right of this tab. You should see the results of makear applying wavelets to the sample data displayed in the console where you invoked the GUI.