Scenario Manager Enhancements
Requirements

CADSWES

# Introduction

The Scenario Manager allows editing of certain RiverWare model data without affecting the integrity of the baseline model. With the Scenario Manager, the values of a subset of slots can be altered independently of the baseline model. The Scenario Manager provides an interface to load, edit, and save values for these slots, run the model using these different slot values, and compare the baseline values with the scenario values.

[insert brief description of sandbox here, what it is and perhaps what we learned from it]

A shortcoming of the existing Scenario Manager is that the input options are either too limited or too difficult. When using the Scenario Sandbox, the options are limited to scaling or offset of up to three slots; this is too limiting. When using the full Scenario Manager, the Stakeholder can change any scenario slot, but the interface requires RiverWare experience to make the changes.

This document presents requirements and preliminary design ideas for improving the interface and functionality of the Scenario Manager. The description of this task from the contract is:

Requirements and Design/Estimates for RiverWare Scenario Manager for Stakeholder Use: Extend RiverWare's Scenario Manager so that Stakeholders can create, run compare and save scenarios, with permitted input variations on a provided baseline model, without a full RiverWare.

To achieve this goal, we design a scenario exploration tool with which Stakeholders will interact to change the baseline model, conduct scenario runs, view the results, export a description of the scenario out of RiverWare if desired, and provide the developed scenario back to the Developer with comments.

# Requirements

We first summarize the required Scenario Manager enhancements, focusing on the Stakeholder experience, and then provide more details in the remainder of the section.

Note that in the context of scenario management, it is useful to differentiate between two types of RiverWare users, the Developer and the Stakeholder[[1]](#footnote-1):

**Developer**: The Developer creates the baseline model with which the Stakeholder interacts, specifying and constraining the ways in which the Stakeholder can interact with it. We assume that the Developer has taken the relevant training classes and has experience building and running RiverWare models.

**Stakeholder**: The Stakeholder is the person who will be creating and running scenarios. They may have little or no experience with RiverWare or the baseline model. Often the Stakeholder will care more about certain aspects of the model than others (e.g., their interests might focus on their own state or irrigation district).

## Summary of Requirements

* Provide a tool with which the Stakeholder can explore multiple model scenarios. That is, the Stakeholder can repeatedly accomplish the following sequence of actions: change aspects of the baseline model, execute a simulation, and view results.
* To conduct scenario exploration, the Stakeholder requires a baseline model provided by the Developer and a RiverWare executable licensed for scenario exploration.
* The RiverWare license for scenario exploration is free or low cost.
* Stakeholders can change the baseline model by changing numeric values in scalar, series and table slots. The Developer specifies which slots can be changed and provides a valid range for each value that can be changed.
* Support for viewing results should be flexible, but focus on comparing results from a scenario run with the baseline run and with other scenario runs.
* The Stakeholder can save a description of a scenario run, including their changes and the associated results, and this description can be communicated to the Developer.
* Stakeholder interaction with RiverWare is constrained to running scenarios as described above; no other functionality is available to Stakeholders.
* No training is required of the Stakeholder. The user interface presented to the Stakeholder is limited to that necessary to accomplish their objectives and is easy to understand and use.
* The Stakeholder can easily access on-line help specific to the scenario exploration.

## Access to RiverWare

A primary requirement is that a RiverWare executable for scenario exploration be available to the Stakeholder for free or low cost. As a result a new type of license will be required. It should be:

* Free or low cost
* Easy to obtain
* Obtainable without much staff time from CADSWES (this requirement may be met by other tasks to make the licensing more streamlined.)

## Restrictions on functionality

RiverWare licensed for scenario exploration has a strictly limited functionality. In particular, the following behaviors are not supported with this license:

* Load non-baseline models
* Execute input DMIs
* Execute arbitrary output DMIs (the Developer controls which output DMIs are available).
* Create objects, slots, or rules.
* Save the model (they can save scenarios).
* Execute MRM run

## Modifying Inputs

In this section we describe the ways in which the Stakeholder can change inputs. Following are the types of inputs that should be editable for a scenario:

* Scalar slot values (within limits)
	+ Direct edit
	+ Slider
	+ Select value from a list of numeric values
	+ Select value from an enumerated list. For example, Wet, Dry, Average (menus or radio buttons). In this case, the hydrology type corresponds to a scalar value that is referenced by rules configured by the Developer. The rules use the scalar value to influence other slots within the run.
* Series slot values (within limits).
	+ Direct edit
	+ Copy/paste
	+ Scale
	+ Adjust (offset)
	+ Select an alternative series to use, by name. For example, High, Medium, Low. (This requirement could be met by the forcing the Developer to configure a scalar slot and rules.)
* Table slot values (including periodic slot)
	+ Direct edit
	+ Copy/paste (?)

Some operations should be supported for sets of slots. In particular, the Stakeholder should be able to conveniently scale a set of slots up or down (E.g., adjust all demands up 20%). While this functionality could be met by allowing the Stakeholder to change a scalar slot value which is then used by a rule that does the scaling of the slot values, this process is not transparent to the Stakeholder. The logic of the rule and even knowledge about what a rule is are not available to the Stakeholder, so it is preferable to provide a mechanism which is explicit about what slots are being changed and how.

The Developer controls which slots may be edited, which types of editing are appropriate for each input, and, where appropriate, what types of optional interface controls should be provided (e.g., a slider, menu, or set of radio buttons).

We do not allow the Stakeholder to run input DMIs because this could lead to undesired slots getting set or would require significant enhancements to prevent those slots from being set. Thus all of the scenario data that is not entered by the Stakeholder should be slot data contained in the model file.

## Scenario Execution

Users will be provided with a button to initiate the execution of a scenario. A progress bar indicates progress, and a Stop button allows the run to be cancelled.

## User Interface

For the Stakeholder, the scenario exploration tool should be the main interface they see and the only interface in which changes are made. As a result, the Scenario Manager should have a friendly and easy to use interface.

## Viewing Results

The Stakeholder must have an easy to use interface to view the scenario results and compare them to baseline results and other scenarios. Plots should be readily available and easy to generate. In general, a Stakeholder may want to see different results than another Stakeholder. For example, for a river with multiple water users, each user might only care about their demands. There must be a way for the Developer to choose the general type of output that the Stakeholders can view and then allow the Stakeholder to choose the specific information.

We are considering two possible approaches:

The Developer could choose the main system outputs, like Pool Elevation and Storage. The Developer also chooses the general slots that Stakeholder can see, like “Shortage”. Stakeholders could then choose which instances of the slots (i.e., for which objects) they wish to compare against the baseline (this might require the Stakeholder to re-run the baseline model). For example, the Stakeholder might choose to see their irrigation district’s Shortage.

The Developer chooses a complete list of scenario output slots. The Stakeholder then chooses the information they wish to see.

## Exporting Information

The Stakeholder should be able to generate output devices, if enabled by the Developer. For example, the Developer may create an Output DMI to Excel or a Model Report that displays an HTML representation of the scenario.

## Saving Scenarios

The Stakeholder must be able to save the scenario inputs and outputs and be able to provide the information back to the Developer. For example, the user should be able to save a scenario file that contains all the scenario information; everything necessary to reproduce the scenario run. This scenario file along with a model report that shows the information in a readable format should provide the information for a Stakeholder to say “Here is my scenario that I like.”

# Design Ideas

Following are design ideas to meet the above requirements:

Licensing: A new license type will be created. This type of license will behave as required when the Stakeholder opens a baseline model. If they open a non-baseline model, perhaps RiverWare could behave as though it is a viewer license.

Interface and Input Options: The Script Manager provides much of the configuration of the user controls that are desired. Could we somehow blend the Scenario Sandbox with a Script Dashboard? Here is how it could work from the view of a Stakeholder. They open the Scenario Sandbox and get the following type of layout (we would likely need to put these on tabs to conserve real estate):

Create new scenario or select scenario:



Dashboard with user control for modifying specific values



 

Run scenario: 

View results compared to basiline



Generate output devices: 

For this design, the Developer would need to set up a script that sets values in the model. Perhaps only a subset of the actions available is allowed for a “Scenario Dashboard”. This would prevent them from doing too much with the script. The main actions needed are the Set Values on the various types of slots.

Output: The Developer would also need to select the list of output slots and create output devices that could be generated by the Stakeholder. Perhaps the Developer selects a small set of slots that are displayed in the output dashboard and selects all the slots they could see in other output devices. The main outputs would be viewable in the scenario manager while less commonly used outputs could be viewed in a different dialog, like a generate plot or report.

# Notes

* To what extent should we support editing values on slot sets as opposed to individual slots.
* Are there additional controls needed, e.g., ability to change output DMI output file name.
* Make explicit requirement that should be able to reproduce scenario later.
* Software mechanism to limit Stakeholder interaction – we only need to limit the behavior of the windows that we present to the user, so the more we limit that, the more feasible this becomes. For example, if we limit interaction to the scenario exploration tool, the slot value changing/viewing dialogs, and the output dialogs, this becomes a manageable problem.
* The Developer needs to provide the Stakeholder with a description of the baseline model and instructions for what to do. RiverWare online help is of limited use in this context as it is broad and uses special RiverWare terminology. Maybe this dialog needs a special help mechanism, as well as possibility of presenting a configured custom description.
* Analyzing outputs: we need support for comparing to other scenario runs, not just baseline.
* The proposal for licensing might be problematic for the situation in which both a Developer and Stakeholder use the same machine, in which case they might both get the same license. RiverWare staff and the Developer will want to test the baseline model and so run RiverWare in scenario exploration mode.
1. We refer to the Developer and Stakeholder as if each was a single user, but each role could be played by several individuals or could in some cases be the same person. [↑](#footnote-ref-1)