**JSL\_DB Issues**

Issue #1: The model element ID can change from simulation to simulation based on model structure.

* The model element ID is assigned automatically based on a static variable within ModelElement. The number assigned is based on the number of model elements created within the model. Each model element is assigned its ID based on the total number of elements previously created. Thus, the number assigned can vary based on changes to the model creation logic from simulation instance to simulation instance.

Conclusions

* There is no way to guarantee that the same ID is assigned in two different simulation instances to an element that is conceptually the same.
* One mitigation strategy is to make the counter of model elements based on the containing simulation/model. The current method guarantees that the identifiers will be different if multiple simulations are created within the same program invocation. Making it relative to the containing simulation maintains identity per model instance but allows two simulations that have the same order creation for conceptually similar model elements to get the same identifiers.
* Bottomline, we cannot guarantee the IDs will be the same.

Issue #2: The model element name can change from simulation to simulation based on model structure.

* The name can be directly assigned by the user when creating a model element.
* A model element in one simulation that is conceptually the same as a model element in another simulation may be assigned different names. There is no way to guarantee that they get the same name because the user can provide the name. Thus, this has to be controlled via practices/guidelines.
* When a user does not provide a name, a default name is provided. The default name is based on the model element name concatenated with the model element ID. Thus, for the same reasons as in Issue #1, the model elements may get different names.
* The recommended idiom of assigning the model element name based on the parent name allows model element names to be unique. That is, getName() + “: element’s name”. While better, this method also cannot guarantee that elements that are conceptually the same get the same name in different simulation instances because the order of creation can affect the path implied by this naming convention.

Conclusions

* There is no way to guarantee that the same ID is assigned in two different simulation instances to an element that is conceptually the same.
* Control problem through education and practices.
* Use mitigation strategy for Issue #1.

Issue #3: Statistic name and model element name may be different for statistics derived from ResponseVariable, TimeWeighted, Counter from simulation to simulation.

* The underlying statistic name is by default set equal to the model element name. Since the model element name can vary from simulation to simulation, so can the statistic name.
* The API allows the statistic name to be directly set by the user. The user could change the statistic name at any time. Thus, the statistic name may even be different than the response model element name within the same simulation.
* Many statistical analysis tables depend on the names being the same. For example, pairwise comparisons assume some name correspondence between simulation instances.

Conclusions

* Because of issues #1 and #2, there is no way to guarantee that the statistic name for some conceptually the same response has the same name from simulation instance to simulation instance.
* Pairwise views will only show differences when the names are the same and this should be documented.
* Because the model element name and the corresponding statistic name may be different within the same simulation SQL based on the assumption that they are the same is fragile.
* The API should be changed to guarantee that the names are the same within a simulation instance.
* Consider re-defining the statistical tables so that both the response model element name and the statistic name are stored in the same table.