1 Requirements

System Performance

(1) The System should support construction of a preliminary model in less than five days. A preliminary model is the initial testable implementation of a modeling hypothesis. It has 5% of the maximum supported model size in terms of reactions, and less than ten each of runs, parameter sets, initial condition sets, experiments, transforms, and objective functions. The time required to implement the transforms and objective functions for the preliminary model is not included. Parameter estimation is not supported.

System Functionality

(2) The System should support organization of a working model over three or more years. A working model has 20% of the maximum supported model size in terms of reactions, runs, parameter sets, initial condition sets, experiments, transforms, and objectives. Parameter estimation is supported. Organization of a model includes maintaining all of the associated data files, historical model development information, and the developed software for handling model data.

Model Building Performance

(3) The Model Builder should support a minimum of 1000 reactions in a model. A reaction consists of a name, kinetic equation, parameters needed for the kinetic equation, and a description. Each reaction only involves a small of number of species, and the total number of species and reactions is approximately equal.

(4) A reaction should take less than two minutes to enter. The time required to enter a reaction includes the time required to load a working model, specify a new reaction including additional species, parameters, and kinetic laws required by the reaction, and save the model.

Model Building Functionality

(5) The Model Builder should support a standard interchange language such as SBML Level 2. Support of a standard interchange language includes the ability to read, but not necessarily display, any valid model using the language and the ability to reject invalid models. The interchange language should be the native language used by the Model Builder for saving models.

(6) The Model Builder should support the display of a wiring diagram for the model. Some conventionally accepted format must be used for the wiring diagram. The wiring diagram need only be displayed on demand.

(7) The Model Builder should highlight errors in the model while working. An error includes any definition that will not persist in the interchange language and any definition that will definitely prevent simulation of the model.

(8) The Model Builder should verify the model structure and check that the model is consistent. Verification involves checking that all references to model elements refer to an element that exists in the model. The references that need to be checked include references to compartments, species, rules, and parameters in SBML. Additionally, verification involves checking for invalidly structured models, such as a cycle of compartment containment.

(9) The Model Builder should support the specification of ordinary and stochastic differential equations. Specification of ordinary differential equations requires constructing the right-hand side of the equation. There should be no unreasonable limits on the terms or number of terms that can appear in the right-hand side. Specification of stochastsic differential equations additionally requires constructing a noise function for the equation.

(10) The Model Builder should create differential equations from the reactions. Entering the reaction network and kinetics for the reaction is sufficient for generating the right-hand sides of the equations. No further user input should be required for generating ordinary differential equations.

(11) The Model Builder should support the display of the automatically generated differential equations. The equations should be displayable as plain text mathematics, rich text mathematics such as TEX, or program code as needed.

(12) The Model Builder should detect conservation relations in the reactions. Conservation relations are sets of linearly dependent equations in the model specification. An optimization can be made to the model by eliminating one of the differential equations and replacing it with an algebraic relation. Opportunities for this optimization should be detected and presented to the user.

(13) The Model Builder should support the import of model fragments into the current model. Model fragments are packaged models or components of models. Importation of model fragments requires selecting the model fragment and specifying how the current model should incorporate the model fragment.

(14) The Model Builder should partially automate the input of similar types of reactions. A model typically has many reactions whose kinetics are the same. The time to input these reactions should be reduced by allowing the reuse of a kinetic law defined for previous reactions. Additionally, the Model Builder should support basic editing functionality such as copy and paste.

(15) The Model Builder should support the creation of multiple views to hide levels of detail. A component of the model is a user-defined collection of species, reactions, or compartments. When some of these components are collapsed to a single point, a simplified view of the model is obtained. Each component should be independently collapsible. References to model elements within a collapsed component should have a notation that indicates where the element is located.

Model Execution Performance

(16) The Model Executor should support a minimum of 10000 runs. The number of runs is typically larger than the number of reactions in the model. A run selects a model, a parameter set for the model, an initial condition set for the model, a simulator, and tuning parameters for the simulator.

(17) The Model Executor should support a minimum of 500 parameter sets. The number of parameter sets is typically a fraction of the number of runs in the model. A parameter set consists of a value for each undefined, named parameter in the model. Parameters are defined as in SBML.

(18) The Model Executor should support a minimum of 500 initial condition sets. The number of initial condition sets is typically a fraction of the number of runs in the model. An initial condition set consists of a value for each species in the model. Species are defined as in SBML.

(19) A run should take less than five minutes to enter. The time required to enter a run includes the time required to load the run file for a working model, enter a new run in the program interface, and save the run file. Entering a run requires giving the run a name, selecting a model to run, selecting parameters and initial conditions for the run, selecting a simulator for the run, and configuring the simulator.

(20) A single run for a preliminary model should take less than one minute to execute.

(21) Making a single parameter and initial condition change should take less than one minute. The time required to make a parameter and initial condition change includes the time required to load the run file for a working model, enter a new parameter value, enter a new initial condition value, and save the run file.

Model Execution Functionality

(22) The Model Executor should support multiple simulators. Simulators should have a mechanism for registering with the Model Executor to indicate availability. All tasks required for performing a simulation should be accessible through an interface that is independent of how the simulator is defined.

(23) The Model Executor should support configuration of simulator parameters. Each simulator should provide a list of configurable parameters to the Model Executor when registering. These parameters should be configurable through the Model Executor interface and be persisted with a run definition.

(24) The Model Executor should validate that the model and configuration is suitable for simulation. Simulators cannot accurately execute every model. Additionally, some simulator configuration options are only suitable for a limited set of models. Prior to execution, the model and configuration should be checked against the simulator capabilities.

(25) The Model Executor should support inheritance of parameter and initial condition sets. Parameter and initial condition sets commonly share values with other

sets. These related sets should derive from a single source so that the duplicated entries only need be entered once. Updates to a set should propagate to all derived sets that do not explicitly define that value.

(26) The Model Executor should support both mutant and experimental variants. Three ways to describe variation in a run are by the model definition, parameter set, and initial condition set. A collection of runs should support entries with any of these three run elements changed.

(27) The Model Executor should highlight errors in the configuration while working. An error includes a missing or illegal value for the model, simulator, simulator settings, parameters, or initial conditions of a run.

(28) The Model Executor should update the model definition in a run when it changes. Each run specifies a model definition for execution. Modifying a model definition should automatically propagate changes to all runs using that model.

(29) The Model Executor should support the import of parameter and initial condition sets from the model. Some model interchange languages, such as SBML, support the definition of parameter and initial condition values in the model. When models provide parameter or initial condition values, the Model Executor should assist in transferring the defined values to a run.

(30) The Model Executor should support the execution of a single simulation run and display its output. A validated run definition should be eligible for execution in the Model Executor, with the model results displayed in some suitable form.

Model Analysis Performance

(31) The Model Analyzer should support a minimum of 10000 experiments. The number of experiments is typically the same as the number of runs. An experiment consists of an identifier, experimental observations, format information for the experimental observations, and a description of the experiment.

(32) The Model Analyzer should support a mimimum of 500 distinct transforms. The number of transforms is typically the same as the number of experimental protocols defined. A transform consists of a series of steps that define how model data is captured by the Model Analyzer. Additionally, there should be a mapping from each experimental identifier to the transform that should be run.

(33) The Model Analyzer should support a minimum of 50 distinct objective functions. An objective function measures similarity between the experimental data and model data, and indicates when the model acceptably reproduces the experimental results. Additionally, there should be a mapping from each experimental identifier to the objective function that should be used.

(34) Each experimental observation should take less than two minutes to enter. The time required to enter an experimental observation includes the time required to load the set of experimental results for a working model, define a new experimental observation, and save the set of experimental results.

(35) Each experiment should take less than five minutes to configure. The configuration time does not include programming transforms, objective functions, or types. The time required to configure an experiment includes the time required to load the set of experimental results for a working model, load a transform and objective function for each experimental result, select a transform and objective function for the new experiment, and save all modified data files.

Model Analysis Functionality

(36) The Model Analyzer should support the division of experimental observations into multiple pieces. An experimental observation frequently consists of a collection of data, such as a time series or multiple related qualitative or quantitative observations. These complex data types consisting of multiple values should be decomposable so that each scalar value can be referenced.

(37) The Model Analyzer should support the use of both owned and external experimental observations. Owned experimental observations are those which the Model Analyzer has control over storage and definition. External experimental observations are those which some program independent of the Model Analyzer has control over storage or definition. The use of either type of experimental observation should be transparent to consumers of the experimental observation.

(38) The Model Analyzer should support references back to source information for an experimental observation. An experiment should aggregate the experimental observation along with references to the origin of the experimental observation.

(39) The Model Analyzer should support the assignment of types to experimental observations, transform inputs and results, and objective function inputs. The type of each experimental observation should be specifiable by the user. The type of each transform input and output, and the type of each objective function input, should be specifiable by the code author. If there are multiple addressable values in a collection, each addressable value should be described by the type.

(40) The Model Analyzer should highlight type errors while working. Experiments in which the structure of the experimental observation, transform, or objective function does not match the defined type should be flagged.

(41) The Model Analyzer should support independent definition of the experimental observations, transforms, and objective functions. The definition and persistence of each should be physically and logically separable. Code defining a transform or objective function should not be mixed with the storage of a set of experiments. Connections between experimental observations, transforms, and objective functions should be made using identifiers that are independent of user-controllable properties.

(42) The Model Analyzer should update the run definition in an experiment when it changes. Transforms in the Model Analyzer frequently include a step using the

Model Executor to perform a run. Modifying a run definition should automatically propagate changes to all experiments using that run for the next time the transform is evaluated.

(43) The Model Analyzer should support unattended execution. A batch mode allows for sequential execution of many experiments without human intervention. Non-extraordinary errors should not prevent the further execution of experiments. Progress should be monitored and be available to the user during execution. The accumulated results and list of errors should be made available for examination after the batch is completed.

(44) The Model Analyzer should support the definition of subgroups of experiments. An experiment subgroup is a collection of experiments that are typically run at the same time. Important subgroups may be used as a quick check of model fitness before running the entire experiment suite. A defined subgroup should persist with the experiment definition for easy access during future runs.

(45) The Model Analyzer should support user defined transforms, objective functions, and data types. It is not expected that a comprehensive set of transforms, objective functions, and data types will ever be created. New experimental observations will have novel structure and organization, and new models will require novel analysis methods. Integration with the Model Analyzer should not be required to add this new code.

Model Tuning Performance

(46) The Model Tuner should take less than a day to configure. A working and well-tested configuration for the Model Builder, Model Executor, and Model Analyzer may be used to assist the configuration.

Model Tuning Functionality

(47) The Model Tuner should support unattended execution. A long-running mode allows for continued trial of many parameter sets without human intervention. Progress should be monitored and be available to the user during execution. The final result and list of errors should be made available for examination after the run is complete.

(48) The Model Tuner should return found parameters to other tools. After a successful parameter estimation run, new values are obtained for the free parameters. These optimized values should automatically be inserted into a Model Executor run file so that other tools can use the parameters found.

(49) The Model Tuner should support weighting of the experiments. The modeler associates each experimental observation with a certain amount of confidence about the accuracy of the observation. These confidence measures should be explicitly accounted for during parameter estimation by weighting each experiment according to confidence when finding an optimum. If the Model Analyzer is using a compatible objective function that defines these weights, the Model Tuner should support importing the weights from the Model Analyzer.

2 Current Support for Requirements

As of May 1st, one of the System requirements is supported by JigCell. This is unchanged from January 1st.

#	Supports	< 1 month	< 3 months	Fails
1	•			
2			•	

Table 1: JigCell Support for Requirements (System)

As of May 1st, seven of the Model Builder requirements are supported by JigCell. Two requirements are close to being supported. Since January 1st, requirement 14 was completed and requirement 3 has regressed.

#	Supports	< 1 month	< 3 months	Fails
3		•		
4	•			
5	•			
6				•
7	•			
8	•			
9			•	
10	•			
11		•		
12	•			
13				•
14	•			
15				•

Table 2: JigCell Support for Requirements (Building)

As of May 1st, ten of the Model Execution requirements are supported by JigCell. Four requirements are close to being supported. This is unchanged from Janaury 1st.

#	Supports	< 1 month	< 3 months	Fails
16	•			
17	•			
18	•			
19		•		
20	•			
21		•		
22	•			
23	•			
24				•
25	•			
26	•			
27		•		
28		•		
29	•			
30	•			

Table 3: JigCell Support for Requirements (Execution)

As of May 1st, thirteen of the Model Analysis requirements are supported by JigCell. Two requirements are close to being supported. Since January 1st, requirements 35, 39, 40, and 43 were completed.

#	Supports	<1 month	<3 months	Fails
31	•			
32	•			
33	•			
34	•			
35	•			
36	•			
37	•			
38	•			
39	•			
40	•			
41	•			
42		•		
43	•			
44		•		
45	•			

Table 4: JigCell Support for Requirements (Analysis)

As of May 1st, none of the Model Tuning requirements are supported by JigCell. Two requirements are close to being supported. This is unchanged from Janaury 1st.

#	Supports	<1 month	<3 months	Fails
46				•
47		•		
48		•		
49			•	

Table 5: JigCell Support for Requirements (Tuning)