A WEB-BASED REPOSITORY OF REPRODUCIBLE SIMULATION EXPERIMENTS FOR SYSTEMS BIOLOGY

Michael A. Guravage and Roeland M. H. Merks
Centrum Wiskunde & Informatica (CWI)
Netherlands Consortium for Systems Biology & Netherlands Institute for Systems Biology (NCSB-NISB)
Science Park 123, 1098 XG Amsterdam, The Netherlands

Keywords: SED-ML, SBML, MIASE, Simulation, Modeling, Plone, CMS.

Abstract: Systems Biology requires increasingly complex simulation models. Effectively interpreting and building upon previous simulation results is both difficult and time consuming. Thus, simulation results often cannot be reproduced exactly; making it difficult for other modellers to validate results and take the next step in a simulation study.
The Simulation Experiment Description Mark-up Language (SED-ML), a subset of the Minimum Information About a Simulation Experiment (MIASE) guidelines, promises to solve this problem by prescribing the form and content of the information required to reproduce simulation experiments. SED-ML is detailed enough to enable automatic rerunning of simulation experiments.
Here, we present a web-based simulation-experiment repository that lets modellers develop SED-ML compliant simulation-experiment descriptions The system encourages modellers to annotate their experiments with text and images, experimental data and domain meta-information. These informal annotations aid organisation and classification of the simulations and provide rich search criteria. They complement SED-ML's formal precision to produce simulation-experiment descriptions that can be understood by both men and machines. The system combines both human-readable and formal machine-readable content, thus ensuring exact reproducibility of the simulation results of a modelling study.

1 INTRODUCTION

Systems biologists unravel the inner workings of biological systems in a continuous cross-talk between predictive computer simulation models, and biological experiments to test the model predictions. To build models efficiently and compare model predictions with experiments, effectively interpreting simulation results and building upon previous work is crucial. Unfortunately, it is both difficult and time consuming, because, 1) publishing exact model definitions and working simulation codes is not (yet) common practice, and 2) the exact steps taken and the information required for reproducing the simulation (e.g. model parameters) are rarely recorded exactly. As a result, trying to reproduce published simulation results can become a research project in itself. This hampers progress in systems biology. The difficulty in validating results impedes the progress of simulation studies.

A range of public databases exists for depositing computational models in a systematic way, including the BioModels database (Le Novère et al., 2006), the modelDB (Hines et al., 2004), the Database of Quantitative Cellular Signalling (DOQCS) (Sivakumaran et al., 2003) and Physiome (Yu et al., 2009; Yu et al., 2011). Although these databases make it much easier to retrieve the definitions and codes of published simulation models, reproducing an observation in a simulation model also requires exact knowledge of many additional factors including the initial conditions, the parameters used, the integration times, etc. In other words, the model databases contain the ingredients for a simulation experiment, but the recipe is lacking.

The Simulation Experiment Description Markup Language (SED-ML) (Köhn and Le Novère, 2008) implements a subset of the *Minimum Information About a Simulation Experiment* (MIASE) guidelines. MIASE is an emerging standard that promises to solve this problem by prescribing the form and content of the information required to reproduce simulation experiments. SED-ML is a recipe template, an instance of which include a description of and reference to existing, preferably curated, models, inputs,

A. Guravage M. and M. H. Merks R. (2011).
A WEB-BASED REPOSITORY OF REPRODUCIBLE SIMULATION EXPERIMENTS FOR SYSTEMS BIOLOGY.
In *Proceedings of 1st International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, pages 134-141
DOI: 10.5220/0003598001340141

procedures and outputs that, together, describe of how to use the models in simulations that constitute an experiment. While SED-ML is detailed enough to enable automatic rerunning of simulation experiments, its XML-based format makes it difficult for modellers to interpret. Also, SED-ML focuses on computational models of biological processes written in, e.g., SBML format; whereas the simulations most difficult to reproduce are typically written in general-purpose languages. Our repository accommodates models written in both XML and general-purpose languages like C++ by allowing model references to include both model databases and source code repositories.

SED-ML is an grammar for describing simulation experiments. Its specificity allows one to describe a simulation experiment in sufficient detail to reproduce the simulation results automatically. SED-ML eventually can act as a scripting language to describe the steps leading to a published simulation result. While SED-ML ensures that the simulation-experiment description is unambiguous, it is difficult to interpret the intention of the modeller by reading only his SED-ML code. Combining complimentary informal descriptive information with SED-ML's formal specification ensures that simulation experiments descriptions can be interpreted by both humans and machines. Adherents of Literate Programming (Knuth, 1984) have long argued that the complementary nature of informal descriptions and formal specifications facilitates the retention and transfer of information.

For the modeller, creating simulation-experiment descriptions in our repository is analogous to keeping a laboratory notebook. Models can be described in the context of real simulations at an appropriate level of abstraction and detail, and the complementary nature of informal and formal specifications encourages correctness and completeness. As a result, simulation experiments are both more easily reproduced and more readily understood. Using the repository thus facilitates collaboration between modellers. The repository can be tailored to realise any collaboration strategy, e.g., a repository's contents can be accessible to the public or restricted to an enumerated list of individuals. Modelers can invite colleagues to collaborate on specific simulation-experiment descriptions. Workflows shepherd simulation-experiment descriptions through a series of states from inception to completion; so a modeller can easily see which simulation-experiment descriptions are in development, which have been submitted for approval and which have been tested and validated. Experimental collaborators can evaluate a repository of qualified simulation-experiment descriptions that describe models in the context of a real simulation.

In this paper, we argue that placing simulation-experiment descriptions in a repository that allows annotating them with prose descriptions, domain meta-information and experimental data, and that guides their progress through a publication workflow, will significantly improve the usability of the SED-ML recipes. Our final goal is to see published simulation experiments refer to simulation-experiment descriptions in our repository. Those descriptions will contain both machine-executable and human-readable recipes that allow modellers to interpret, evaluate and reproduce the simulations and build upon the published results.

2 THE MODEL REPOSITORY

2.1 Requirements

We aimed to construct a system that allows modellers to create simulation-experiment descriptions conforming to the MIASE guidelines, according to the following system requirements:

2.1.1 Ease of Use, Encourage Annotation

The system should help modellers along the process of creating and managing their simulation-experiment in ways that are natural and easy for them. The top level of a simulation-experiment description hierarchy is comprised of five classes:

- Models
- Simulations
- Tasks
- Outputs
- Data Generators

The system assists the modeller in creating the various elements in the correct order and with valid data. In addition to the minimum data prescribed by the MIASE guidelines, the system should encourage the modeller to annotate each part of a simulation-experiment description with descriptive information at an appropriate level of abstraction. The editing environment should be familiar to anyone who has worked with office suite software. The system should also allow the user to include experimental data such as images and animations, and generic and domain-specific meta-information. This data should be indexed automatically and be made available as search criteria.

2.1.2 Data Security

The system should have several levels of security, such that users can choose to keep a project from themselves, share it with coworkers or reviewers, or make it public after the results have been published in a scientific journal. Modelers own their simulation-experiment description, and they cannot be viewed or altered without their owner's consent. The owner of a simulation-experiment description is able to invite fellow modellers to collaborate on specific simulation-experiment descriptions. Thus, the system should make a distinction between anonymous and authenticated users. Permissions based on roles, e.g., anonymous visitor, authenticated member, modeller and curator can be used to delegate responsibilities and precisely control access and visibility.

2.1.3 Data Accessibility

Someone using the system should easily be able to search through the collection of simulation-experiment descriptions using a variety of search criteria. Finding the the simulation-experiment description he or she wanted, a modeller should be able to browse through it and, if appropriate, to export the entire simulation-experiment description to a file conforming to the SED-ML standard. Over time, simulation-experiment descriptions should resemble 'laboratory notebooks' that contain their entire histories from inception through development to publication. Additional tools should make it possible to reproduce automatically simulations based on the exported SED-ML files.

2.2 Architecture

Based on these system requirements, we implemented a prototype of the system based on the content management system Plone[1] (Pastore, 2006). We choose Plone because its concept of 'roles' clarifies responsibilities: this allows modellers to concentrate on creating their content. Plone's development tools allowed us to turn Unified Modeling Language (UML) [2] models into running Plone code.

Plone is a free and open source content management system written in Python[3] and built on top of the Zope[4] application server. It stores all its information in Zope's built-in transactional object database (ZODB) (Fulton, 2000). We have developed our simulation-experiment description repository by modelling a set of SED-ML content types in UML. Our models are input to ArchGenXML (Auersperg et al., 2007) - a code generating tool will turn our UML models into a Plone add-on product that we install in a Plone portal running on the Zope server.

2.2.1 Performance

The number of SEDs a repository can hold is limited only by the physical capacity of the server on which repository runs.

The Oscillation to Chaos (O2C) SED example (Köhn and Le Novère, 2008) is a hierarchy of 18 SED objects in our repository that occupy 30K in the Zope database. O2C's exported SED-ML representation, stripped of all its annotations, is a mere 3.5k.

To evaluate the effect of the Varnish cache we used ApacheBench[5] to collect median performance data from both the default Apache/Varnish/Plone route, and directly from Plone via the zeo-clients. Figure 1 shows how efficacious a caching server becomes as the number of concurrent requests increase.

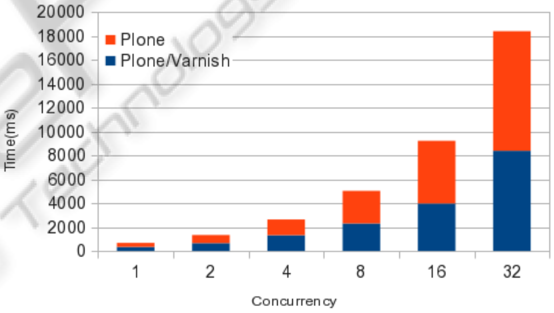


Figure 1: Median Total Request Time.

2.2.2 Deployment

The SED repository can be installed on any Zope application server running Plone 3. We deploy Plone on a shared virtual server with two quad-core Xeon E5405 processors and 4GB of RAM.

Plone is configured in a standard production environment (Aspeli, 2007) with an Apache web server acting as a reverse proxy for a Varnish caching server that balances requests over five zeo clients connected to a single Zope application server.

Our memory footprint is less than 100MB for the Zope application server and an average of 300MB for each zeo-client.

[1] www.plone.org

[2] www.omg.org/spec/UML/2.1.2

[3] www.python.org

[4] www.zope.org

[5] httpd.apache.org/docs/2.0/programs/ab.html

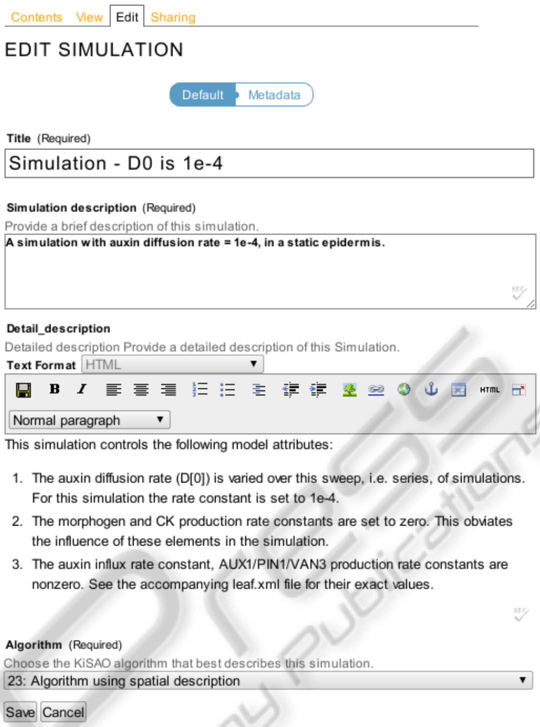
2.3 Object Model

As a formal representation of the simulation experiments, we made use of the Simulation Experiment Description Object Model (SED-OM). The SED-OM is a formal representation of the MIASE guidelines expressed using the Unified Modeling Language (UML). From the object model we derive the Simulation Experiment Description Meta Language (SED-ML) schema that represents the grammar for SED-ML compliant documents. The SED-ML XML schema is distributed in XSD[6] format - a W3C[7] consortium standard for defining an XML Document. Beginning with the SED-OM schema we reverse-engineered our own object-model - annotated with Plone supplied tagged-values that specify how data members look and behave in the Plone UI. For this purpose, we chose to use an XML editor Named Oxygen[8] and a UML editor named Poseidon[9]. Our repository and its custom content types are based on the Level 1 Version 1 (Draft) version of the SED-ML schema (Bergmann, 2010).

Casting the SED-ML XML schema as a set of custom content types that represent the objects in the SED-OM allows nontechnical users to use the content types as building blocks to create and manage Simulation and Experiment Descriptions. Modelers are not confronted with complex XML syntax, but create and manage simulation-experiment descriptions through a set of clear and concise forms. Figure 2 shows the Simulation edit form. Its data fields conform to the SED-ML XML schema simulation element. The detailed description field uses Plone's own powerful rich editor with text formatting and image and link insertion abilities. A simulation-experiment description's data are stored in such a way that it can be exported in SED-ML compliant XML.

2.3.1 Code Generation

To generate Plone add-on modules from the UML class model, we used ArchGenXML[10]. ArchGenXML is a code-generator for Plone. Archetypes is a framework for building content objects based on schemas and provides features such as automatically generating editing and presentation views, assigning unique content Ids, registering content types with various Plone management tools and storage transparency. Our newly generated content types behave like native content types in that they adhere to Plone's document workflows and security model.



EDIT SIMULATION

Title (Required)

Simulation - D0 is 1e-4

Simulation description (Required)

Provide a brief description of this simulation.

A simulation with auxin diffusion rate = 1e-4, in a static epidermis.

Detail_description

Detailed description Provide a detailed description of this Simulation.

Text Format HTML

Normal paragraph

This simulation controls the following model attributes:

1. The auxin diffusion rate (D[0]) is varied over this sweep, i.e. series, of simulations. For this simulation the rate constant is set to 1e-4.
2. The morphogen and CK production rate constants are set to zero. This obviates the influence of these elements in the simulation.
3. The auxin influx rate constant, AUX1/PIN1/VAN3 production rate constants are nonzero. See the accompanying leaf.xml file for their exact values.

Algorithm (Required)

Choose the KiSAO algorithm that best describes this simulation.

23: Algorithm using spatial description

Figure 2: Simulation Edit Form.

ArchGenXML automatically generates the editing and presentation views for each new content type. In the UML model, we annotate class member data with tagged-values that describe how it should look and behave. For example, the standard stipulates that the top-level simulation-experiment description class must have an integer version number. The tagged-values assigned to it indicate that it is a required field, that it must be an integer, what its label and description text are and its default value.

Tagging simulation-experiment descriptions with standard and domain meta-information will facilitate their organisation and classification while simultaneously providing rich search criteria. Plone implements the Dublin Core (Weibel et al., 1998) metadata - a small set of text elements through which Plone content types can be described and catalogued. Their values are indexed in Plone's internal catalogue and can be used in searches. In addition to the Dublin Core stipulated 'description' element, each simulation-experiment description content type has a detailed description field in the form of a rich text editor, that the modeller can use for writing prose descriptions of the simulation experiments.

The simulation-experiment description object model specific two vocabularies: a model's encod-

[6] www.w3.org/XML/Schema

[7] www.w3c.org

[8] www.oxygenxml.com

[9] www.gentleware.com

[10] plone.org/products/archgenxml

ing formalism and a simulation's algorithm designation. The former is a simple list including e.g., SBML, CellML and C++. The latter describes the Kinetic Simulation Algorithm Ontology (KiSAO) [11] encoded in the vocabulary description markup language (VDEX) [12]. Both vocabularies, while used by the repository, are managed separately, which allows the site administrators to modify the contents of the vocabularies without adversely effecting the existing models and simulations that use them.

2.3.2 Collaboration Strategies

Plone's workflows control the progress a modelling project through a series of states from inception to publication. Figure 3 shows the states and transitions in a restricted Plone workflow. By customising these workflows, we can realise a variety of collaboration strategies. For example, the current workflow for our several simulation-experiment description repositories makes them accessible only to members of their respective modelling groups. The workflow progressively expands the audience as individual simulation-experiment descriptions proceed through a series of approvals, first to other members of the research group, then to portal members, and finally to the general public.

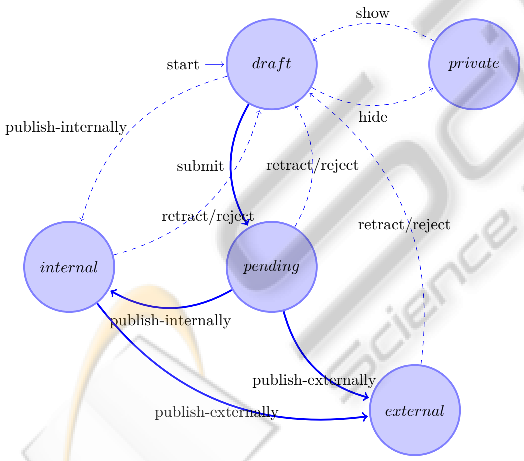


Figure 3: A Restricted Plone Workflow.

2.3.3 Exporting SED-ML Compliant XML

In order to use the repository for automatically reproducing simulation experiments, the system must have facilities to export the simulation-experiment description to SED-ML compliant XML. We have achieved this by customising the code generated by ArchGenXML.

The simulation-experiment description class is the root of a hierarchy that, taken in its entirety, represents a complete simulation-experiment description. We added custom python code to each class in this hierarchy to enable their instances to create an SED-ML representation of themselves and request their children to do likewise. The code was written with Python's light-weight implementation of the Document Object Model interface. A document tab named 'Export this' appears at the bottom of every simulation-experiment description. It is linked to a browser-view - a special class that links URLs with specific bits of code in Zope's component architecture. The tab, when clicked, sets a simulation-experiment description's export process in motion.

Concluding, we have developed a set of custom content types that faithfully represent the SED-ML XML schema, which can be installed into a Plone portal running on a Zope application server customised to local needs.

2.4 Usage

2.4.1 Data Entry

The user interacts with our system via a Plone-based web interface. Each SED-ML content type has its own presentation and editing forms. Creating a simulation-experiment description consists of navigating to the repository and clicking on the 'Add SED' menu button. A new simulation-experiment description edit form appears and solicits the required information, and an appropriate amount of descriptive information. As specified in the schema, the SED-ML name space and level fields have default values. Since the simulation-experiment description object is the root of a new simulation experiment hierarchy, after clicking the simulation-experiment description's 'Save' button, the add items menu presents a new list of content items the modeller can insert at this point. This process continues until the simulation-experiment description is complete.

2.4.2 Searching

Since we specified which information would be indexed while we modelled our SED-ML content types in UML, searching through the repository is now a straightforward matter of typing our search criteria into the search box.

[11] www.ebi.ac.uk/compneur-srv/kisao

[12] www.imsglobal.org/vdex

2.4.3 Reproducing Simulation Experiments

When presented with a complete simulation-experiment description, especially one written in a general purpose programming language and not (yet) reproducible automatically, the modeller can simply retrieve the recipe from the repository. Section 3 is a case study describing how to build a simulation-experiment description whose model is written in C++.

If a modeller wishes to reproduce the results of a simulation-experiment description whose models are written in a dialect of XML, e.g., SBML, and whose simulations are, e.g., uniform time course, he or she can export the simulation-experiment description in SED-ML compliant XML directly by clicking on the simulation-experiment description's 'Export this' tab. The repository will return a file whose name is derived from the simulation-experiment description's Id. The modeller can use this file as input to one or another simulation system such as COPASI (Hoops et al., 2006).

2.4.4 In Conclusion

The repository is a resource with which a modeller can fully describe a simulation-experiment description recipe. The SED-ML compliant content types assist in this by soliciting and validating information as it is entered. Content and meta-data are indexed automatically and available as search criteria. A completed simulation-experiment description should contain the descriptive information to allow the simulation experiment to be repeated manually, and if appropriate its contents can be exported in SED-ML compliant XML to be rerun automatically.

3 EXAMPLE SED COMPOSITION

To demonstrate how to compose a SED, we have chosen an experiment that uses the VirtualLeaf (Merks et al., 2011), an open-source framework for cell-based modelling of plant tissue growth and development. This particular experiment shows how auxins, a family of plant growth hormones, can accumulate in growing plant tissue to form bulbous patterns.

The VirtualLeaf distribution contains the VirtualLeaf framework software, various dynamical models and sample simulation data; so acquiring, installing and running the experiment is simple and comprises only a few steps.

1. Download the VirtualLeaf Framework source bundle from its Google Code repository.
2. Compile and install the VirtualLeaf software.
3. Choose a dynamical model to use, i.e. auxin growth.
4. Choose the simulation data describing the experiments initial conditions.
5. Run the experiment.
6. Aggregate the resulting experiment snapshots into a single animation.

In a simulation experiment description the information in these various steps is distributed over several SED-ML elements as follows.

- The SED contains two *Model* elements. The first model describes the VirtualLeaf framework. The second model describes the auxin growth model plug-in that describes the dynamic character of growing cells. Both models contain unique identifiers naming the particular version of the VirtualLeaf software used in the simulation experiment and its location in a googlecode repository from where the source code can be downloaded.
- A single *Simulation* element contains a leaf description file that describes the initial configuration of cells and their various properties; it also contains a KiSAO designation of the type of algorithm used.
- A single *Task* element associates the models with the simulation as a single experiment.
- A single *DataGenerator* element describes how to aggregate the experiment's results, in this case screen shots, into an animation; it also includes a shell script that performs the aggregation.
- A single *Output* element describes the experiments results, and points to a the dataGenerator that tell how to reproduce them.

Figure 4 shows the repository's output element view of the animation generated by concatenating the VirtualLeaf's multi-screenshot output.

A screencast demonstrating how to build these elements is available on our Repository's public website: sed.project.cwi.nl.

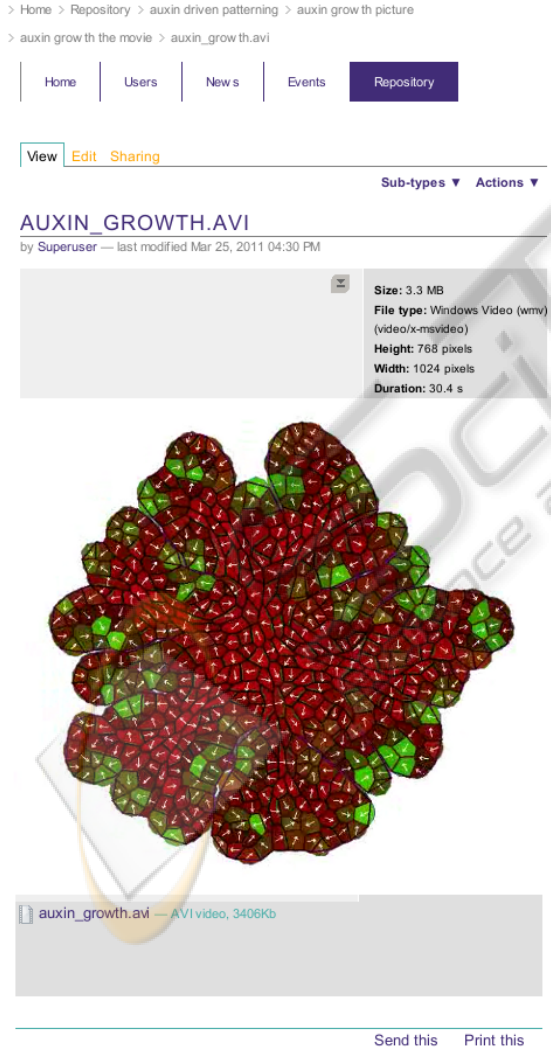
3.1 Run Simulations Automatically

Currently, only XML-based, e.g., SBML, simulations can be reproduced automatically; using tools like COPASI. For the moment, our SEDs whose models are written in general-purpose languages contain descriptive information in each SED element to help the reader understand the simulation and then run it manually.

Future versions of the repository will read a SED's resulting SED-ML file and generate scripts to reproduce the simulation (semi-) automatically using an approach similar to Python's buildout (Aspeli, 2007) system where configuration recipes preform common tasks such as downloading, compiling and installing a software source distribution or executing a program together with its inputs.

4 DISCUSSION

In this paper, we have described a web-based repository that assists modellers in creating simulation-experiment descriptions that conform to the Simulation Experiment Description Markup Language (SED-ML) - a format for the implementation of the MIASE guidelines. Modelers can build simulation-experiment descriptions in an intuitive way, and annotate them with descriptions, experimental data and domain meta-information. This information is indexed automatically and made available as search criteria. Finally, the contents of simulation-experiment descriptions can be exported in SED-ML compliant XML.


> Home > Repository > auxin driven patterning > auxin grow th picture
> auxin grow th the movie > auxin_grow th.avi
Home
Users
New s
Events
Repository
View
Edit
Sharing
Sub-types ▼
Actions ▼
Send this
Print this

AUXIN_GROWTH.AVI

by Superuser — last modified Mar 25, 2011 04:30 PM

Size: 3.3 MB
File type: Windows Video (wmv) (video/x-msvideo)
Height: 768 pixels
Width: 1024 pixels
Duration: 30.4 s

auxin_growth.avi — AVI video, 3406Kb

Figure 4: Repository Output Element View of a VirtualLeaf Animation.

Our web-based simulation experiment repository allows modellers to create and manage simulation-experiment descriptions in a way analogous to keeping a laboratory notebook. Models can be described in the context of real simulations at an appropriate level of abstraction and detail. The entire provenance of a simulation experiment can be preserved and retraced. Ideally, published simulation experiment will refer to simulation-experiment descriptions in our repository that are machine-executable and human-readable recipes that allow modellers to reproduce simulations and build upon the published results.

The benefit to the modellers that they have at their disposal a repository of qualified simulation-experiment descriptions that describe models in the context of real simulations. Their detailed technical specification and associated descriptive information will help reduce the time and effort needed to reproduce simulation results, and provide an excellent start for reusing and extending models.

Maintaining a proper laboratory notebook requires a great deal of discipline. Adequately annotating a simulation-experiment description is equally difficult, if not more. For while a laboratory notebook may be a private work record, it is our intention that once a simulation study has been published its simulation-experiment descriptions in our repository will be read and used by many. The danger for simulation-experiment descriptions, like any documentation, is that they will be left incomplete.

While our repository can export SED-ML compliant XML, it cannot (yet) convert it back into a SED-ML object hierarchy. We are contemplating writing an import facility to read SED-ML files into our repository. We would also like to update our Simulation Experiment Description Object Model to conform to the SED-ML Level 1 Version 1 (Final). Currently, only XML-based, e.g., SBML, simulations can be reproduced automatically; future versions will also implement scripts that (semi-)automatically reproduce simulations implemented in general-purpose languages, e.g., using an approach similar to Python's buildout system.

We hope that modellers will be motivated to write reproducible simulation-experiment descriptions using our repository when they experience the benefits of having full and immediate on-line access to their work.

ACKNOWLEDGEMENTS

This work was (co)financed by the Netherlands Consortium for Systems Biology (NCSB) which is part of the Netherlands Genomics Initiative / Netherlands Organisation for Scientific Research.

We would also like to thank our colleagues in the NCSB-NISB Biomodeling and Biosystems Analysis - Life Sciences Group at CWI for their cooperation in developing and testing the repository.

REFERENCES

Aspeli, M. (2007). *Professional Plone Development*. Packt.

Auersperg, P., Klein, J., and van Rees, R. (2007). ArchGenXML code generator.

Bergmann, F. (2010). Simulation Experiment Description Markup Language (SED-ML): Level 1 Version 1 (Draft).

Fulton, J. (2000). Introduction to the Zope Object Database. In *Proceedings of the 8th International Python Conference*.

Hines, M., Morse, T., Migliore, M., Carnevale, N., and Shepherd, G. (2004). ModelDB: a database to support computational neuroscience. *Journal of Computational Neuroscience*, 17(1):7–11.

Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P., and Kummer, U. (2006). COPASI–a COmplex PAthway SImulator. *Bioinformatics*, 22(24):3067.

Knuth, D. (1984). Literate programming. *The Computer Journal*, 27(2):97.

Köhn, D. and Le Novère, N. (2008). SED-ML–An XML Format for the Implementation of the MIASE Guidelines. In *Computational Methods in Systems Biology*, pages 176–190. Springer.

Le Novère, N., Bornstein, B., Broicher, A., Courtot, M., Donizelli, M., Dharuri, H., Li, L., Sauro, H., Schilstra, M., Shapiro, B., et al. (2006). BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Research*, 34(suppl 1):D689.

Merks, R. M. H., Guravage, M., Inzé, D., and Beemster, G. T. S. (2011). Virtualleaf: an open source framework for cell-based modeling of plant tissue growth and development. *Plant Physiology*, 155(656):666.

Pastore, S. (2006). Web content management systems: using plone open source software to build a website for research institute needs. In *Digital Telecommunications, , 2006. ICDT '06. International Conference on*, page 24.

Sivakumaran, S., Hariharaputran, S., Mishra, J., and Bhalla, U. (2003). The Database of Quantitative Cellular Signaling: management and analysis of chemical kinetic models of signaling networks. *Bioinformatics*, 19(3):408.

Weibel, S., Kunze, J., Lagoze, C., and Wolf, M. (1998). Dublin core metadata for resource discovery. *Internet Engineering Task Force RFC*, 2413.

Yu, T., Lawson, J. R., and Britten, R. D. (2009). A distributed revision control system for collaborative development of quantitative biological models. In Magjarevic, R., Lim, C. T., and Goh, J. C. H., editors, *13th International Conference on Biomedical Engineering*, volume 23 of *IFMBE Proceedings*, pages 1908–1911. Springer Berlin Heidelberg.

Yu, T., Lloyd, C., Nickerson, D., Cooling, M., Miller, A., Garny, A., Terkildsen, J., Lawson, J., Britten, R., Hunter, P., et al. (2011). The Physiome Model Repository 2. *Bioinformatics*.