Background

-History

People have been discussing the possibility of adding modularity to SBML since the very beginning of the language, ten years ago. In an ‘internal discussion document’ entitled “Possible extensions to the Systems Biology Markup Language” and principally authored by Andrew Finney (<http://sbml.svn.sourceforge.net/viewvc/sbml/trunk/specifications/sbml-level-3/old/sbml-team-proposals/original/sbmlext.pdf>), written even before the final version of SBML Level 1 Version 1 was released (in March of 2001), the first of the four titular possible extensions is for ‘submodels’. In it, the main ‘model’ object contains a list of submodels which are model definitions only, and a list of submodel instantiations, which are references to elements of the submodel list. Variable IDs use the ‘X.y’ structure to refer to element y of submodel instance X, and a form of linking model elements through ‘substitutions’ is present. It also introduced the concept of validation through what it called the ‘expanded’ version of the model (now commonly referred to as the ‘flattened’ form): if the flat version of the model would be valid, the model as a whole is valid.

In June of 2001, at the Third Workshop on Software Platforms for Systems Biology, Martin Ginkel and Jörg Stelling presented “XML Notation for Modularity”, with a writeup (<http://sbml.org/images/3/3d/Joerg-sbml-proposals.pdf>), a presentation (<http://sbml.org/images/7/73/Vortrag.pdf>), and some sample XML (<http://sbml.org/images/1/18/Sbml2.txt>), partially in response to deficiencies or missing elements they saw in the ‘submodel’ Finney extension. In it, a ‘classic view’ of modularity is presented, where models are packaged as black boxes with interfaces, with one design goal of being able to substitute one module for another with the same interface, so as to be able to easily simplify or complicate models as needed. Model re-use was emphasized, along with the possibility of developing model libraries that could be used in more complicated or extensive models.

Martin Ginkel again presented in the July 2002 Fifth Workshop on Software Platforms for Systems Biology, with an expanded proposal with the same design goals, in the hopes that it could be incorporated into SBML Level 2 (<http://sbml.org/images/9/90/Sbml-modular.pdf>, with the presentation at <http://sbml.org/images/0/06/Sbml-modular-talk-2.pdf>). This proposal clarified the need to separate model definitions from model instantiations, and, further, the need to designate one model per document as the ‘main’ model.

In March of 2003, an independent proposal by Jonathan Webb was posted to the sbml-discuss mailing list (<http://sbml.org/Forums/index.php?t=msg&th=67&rid=0#msg_111>). The proposal itself has not survived intact on the web (the host site disallowed indexing of their site, preventing even archive.org from saving it), but from the follow-up discussion between him, Andrew Finney, and Martin Ginkel, it seems that this proposal included a generic way to link and reference elements from submodels using XPath (previous proposals had separate mechanisms for species, parameters, compartments, and reactions), brought up the issues of how to successfully resolve conflicting attributes of linked elements, debated whether formal interfaces were necessary or even preferable to direct access to model elements, talked about type checking for linkages, and discussed issues with unit incompatibility. Around this time, Martin formed the Model Composition Special Interest Group (<http://www.mpi-magdeburg.mpg.de/zlocal/martins/sbml-comp>), a group that eventually reached 18 members, including Jonathan Webb.

Model composition did not make it into SBML Level 2 when that was released in June of 2003, being deemed [er, what? Too complex? Proposal not mature?] , and so Andrew Finney (now the co-chair of the Model Composition SIG), presented another proposal in May of 2003, even before Level 2 Version 1 was finalized, that aimed to add model composition to SBML Level 3. With only two years having passed between SBML Level 1 and Level 2, the feeling was that Level 3 would perhaps be released in 2005 or 2006, and the model composition proposal would be ready when it was. This turned out to be a strategic error, as the Level 3 Version 1 release candidate 1 wasn’t released until the last day of 2009, and is only now (at the end of 2010) that Level 3 is about to be formally released with two working implementations of its specification. We’re getting ahead of the story, but it is important to realize that this and all subsequent development of this proposal on the SBML front has remained strictly theoretical, and each software tool that has dealt with modularity in some fashion has done so on its own.

Andrew’s 2003 proposal (<http://www.mpi-magdeburg.mpg.de/zlocal/martins/sbml-comp/model-composition.pdf>; revised in October at <http://sbml.org/images/7/73/Model-composition.pdf>) was an attempt to synthesize Martin and Jonathan’s earlier proposals with his own original submodel ideas, and was envisioned in parallel to another proposal of his regarding arrays and sets of submodels that could be created and destroyed during model simulation (posted to sbml-discuss in September of 2003; <http://sbml.org/Forums/index.php?t=msg&th=234&rid=0#msg_683>). Andrew attempted to resolve the differences in the two basic philosophies (essentially ‘black box’ vs ‘open access’) by introducing optional ‘Ports’ as interfaces between a submodel and its containing model, but also including an XPath-based method to allow the same thing. In this way, it was hoped, a modeler desiring to follow the classic modularity approach could do so, but a second modeler could still use models in ways not envisioned by the original modeler by having access to the model’s core elements. In both schemes, elements in the submodels were replaced by corresponding elements of the containing model.

This proposal provided a ‘direct link’ construct to allow containing models to refer directly to submodel elements without providing a top-level analogue. For example, a containing model could have a reaction that converted a species in one module to a species in a different module, and only create the reaction itself, providing links to the species instead of local linked species.

After this, effort in the SBML community moved to updates to SBML Level 2, and since model composition was slated for Level 3, not much progress was made for a few years, apart from Andrew including a summary of his 2003 proposal and of some of the unresolved issues in a poster at the 2004 Intelligent Systems for Molecular Biology conference in Glasgow (<http://sbml.org/images/9/9c/Ismb-2004-sbml-level-3-poster.pdf>).

Finally, in June of 2007, unplanned discussions at the [Fifth SBML Hackathon](http://sbml.org/Events/Hackathons/The_5th_SBML_Hackathon) prompted the convening of a workshop specifically to revitalize the model composition package, and in September of 2007, the [SBML Composition Workshop](http://sbml.org/Events/Other_Events/SBML_Composition_Workshop_2007) was held in Connecticut, hosted by the Virtual Cell group, and run by Ion Moraru and Michael Blinov. The flurry of activity surrounding this event produced several artifacts, still available online:

* Ranjit Randhawa presented his work on model fusion, aggregation, and composition at <http://sbml.org/documents/proposals/CCB2007DemoPresentation.pdf>, which he had done for his PhD work with Dr. John Tyson at Virginia Tech, as part of the JigCell group, and published at <http://en.scientificcommons.org/53559395>. Among the highlights of this presentation and work:
  + A description of different methods which all need some form of model composition, along with the realization that model fusion and model composition, though philosophically different, entail exactly the same processes and require the same information.
  + Conversion between types: The JigCell wizard described can, for example, promote a parameter to a species, a concept which had been assumed to be impossible and undesirable in previous proposals.
  + For implementation, merging SBML models should be done in the order Compartments -> Species -> Function Definitions -> Rules -> Events -> Units -> Reactions -> Parameters. If done in this order, potential conflicts are resolved as you go.
* Martin Ginkel provided a list of goals for model composition: <http://sbml.org/Events/Other_Events/SBML_Composition_Workshop_2007/Martin_Goals>. This presented some use cases, and summarized many of the issues discussed in this document thus far, including definition vs. instantiation, linking, referencing elements with no SBML IDs, and optional interfaces. It also mentioned the need to allow parameterization of instances (i.e. setting new numerical values that override the defaults), and the need to be able to delete elements of submodels. (He also provided a summary of ProMoT’s model composition approach and a summary of other approaches at <http://ntcnp.org/twiki/bin/view/VCell/OoModelingPromot>).
* Andrew Finney also brought up several comments, listed at <http://sbml.org/Andrew_2007_Comments_about_Model_Composition>. This brought up some old issues as well as some new ones, including:
  + A flag for ports indicating whether they must be overloaded
  + N to N links: when a whole swath of elements in one model are replaced as a group, conceptually, with one or more elements from a different model.
  + Relationship with other L3 packages: this proposal should be generic enough to accommodate future updates and other packages.
* Wolfram Liebermeister presented his group’s experience with SBMLMerge (<http://sbml.org/images/c/c1/SemanticSBML_SBMLcomposition.pdf>), dealing with the pragmatics of merging multiple models. As far as this proposal goes, he noted that the annotations in a composed model need to be considered, particularly since they can be crucial to successfully merging models in the first place.
* Nicolas le Novère created a proposal for modularity in Core at <http://sbml.org/Events/Other_Events/SBML_Composition_Workshop_2007/Modularity_In_Core>. This was unrelated to the above efforts, and is an attempt to modularize a ‘normal’ SBML model. It is more about divvying up the information in an SBML model into chunks than it is about composing a model from different chunks. It was agreed at the conference that this is a completely separate idea, and should be handled on its own. (With Nicolas working on other elements of SBML, further development of this idea has stalled, and should not be expected soon.)
* As a collective, the group produced an ‘Issues to Address’ document at <http://sbml.org/Events/Other_Events/SBML_Composition_Workshop_2007/Issues_To_Address>, with several conclusions:
  + A composed model should be flattenable to a valid ‘core’ model, and all questions of validity can then be simply applied to the flattened model: if the core version is valid, the composed model is valid.
  + The proposal should cover both designed and ad-hoc composition.
  + Elements probably need to be able to be deleted. It is preferable that the deletion syntax be explicit, instead of implied only by the lack of anything replacing the element.
  + Anything may be linked.
  + Overloading defined at <http://sbml.org/Events/Other_Events/SBML_Composition_Workshop_2007/Overloading_Semantics>, and is generally true for all SBML objects. (Contrary to the JigCell Wizard, no type changing is allowed in this scheme.)
  + It is proposed that elements in the outer model always override elements in the submodels, and perhaps that sibling linking be disallowed. This was hotly debated.
  + Interfaces (ports) are indeed helpful, but should be optional. They should not be directional in the electrical engineering ‘input’ and ‘output’ sense—the outer element always overrides the inner element, but apart from that, biology doesn’t tend to work that way.
  + File import might need some sort of check to see if what was imported was the same as it was when the modeler created the model (i.e. MD5 hash).
  + Probably want to allow only whole-model imports, not importing of individual SBML elements.
  + Unit conversion will be an issue.

From all of this, Stefan Hoops was put in charge of creating a first draft of a Model Composition package, now renamed ‘Hierarchical Model Composition’: <http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Hierarchical_Model_Composition_(Hoops_2007)>. Dated a mere one day after the end of the workshop, the proposal was an attempt to summarize the workshop as a whole, and provide a coherent whole, suitable as a Level 3 extension. It provided a brief overview of the history and goals of the proposal, as well as several UML diagrams of the proposed additions. Stefan presented the proposal in August of 2008 at the 13th SBML Forum <http://sbml.org/images/e/e9/HierarchicalModelGothenburg.pdf>, and showed the same slides at the 7th SBML Hackathon in March of 2009 and at the 14th SBML Forum in September of 2009. In order to proceed, Level 3 had to be finalized, and that was yet to come.

In the meantime, Herbert Sauro, one of the founding members of the SBML team, received a grant to develop a modular human-readable model definition language, and hired me in November of 2007 to do so. Herbert and Frank Bergmann, his graduate student, had previously written up a proposal for this at <http://www.sys-bio.org/sbwWiki/_media/sbw/standards/2006-12-17_humanreadable_mdl.pdf>, and this was the design document I initially used to create what I eventually called Antimony. Through a few iterations, the design I eventually settled on (<http://antimony.sourceforge.net/Tutorial.pdf>) was very similar in concept (largely by coincidence) to that developed by the group in Connecticut: model definitions existed separately from their instantiations in other models, and elements of models could be linked (or ‘synchronized’, in Antimony terminology) to each other. Because Antimony was designed to be more ‘quick and dirty’, it allowed type conversions much like the JigCell wizard, where a parameter could become a species, compartment, or even reaction. Synchronized elements could end up with aspects of both parent elements in their final definitions: if one element defined a starting condition and the other how it changed in time, the final element would have both. If both elements defined the same aspect (like starting condition), the one designated the ‘default’ would be used in the final version. I developed methods to import other Antimony files and even SBML models, which could then be used as submodels of other models and exported as ‘flattened’ SBML.

So it was from that perspective that I first heard about the plan for SBML modularity from a presentation by Michael Hucka at the ‘Standards and Specifications in Synthetic Biology’ workshop we hosted in Seattle in April of 2008 (though this presentation is not available online). In March of 2009, I heard Stefan’s presentation for the first time at the 7th SBML Hackathon, and then again in September at the 14th SBML Forum. In January of 2010, I became an SBML Editor, just as the SBML Level 3 Version 1 release candidate 1 came out, and in May of 2010, we hosted the SBML Hackathon here in Seattle. Stefan wasn’t present to talk about the proposal, so after a discussion with some others there, I threw together a few slides and talked about some of the issues I saw with the 2007 proposal (<http://sbml.org/images/b/b6/Smith-Hierarchical_Model_Composition-2010-05-03.pdf>). In particular, I proposed a separation of the ‘replace’ concept (where old references to replaced values are still valid) from the ‘delete’ concept (where old references to replaced values are no longer valid). I wrote up a summary of that discussion, added some more of my thoughts in the interim, and posted it to sbml-discuss (<http://www.sbml.org/Forums/index.php?t=tree&goto=6124>). In it, I proposed and/or reported eight possible modifications to the Hoops 2007 proposal, including:

* Separate ‘replace’ from ‘delete’.
* Separate model definitions from instantiations.
* Get rid of ports, and use annotation instead.
* Annotate the ‘N to N’ replacements, instead of giving them their own construct.

There was but a single response to that post; however, having since gone through the past proposals in more detail, several of the issues in that post turned out to be issues that had indeed been brought up at the 2007 meeting, and had simply gotten missed in the conversion of those discussions to a full proposal. The separation of definitions from instantiations, for example, had been a part of every proposal up until 2007, and indeed was mentioned in the notes for that meeting, so its omission was likely merely an oversight. Similarly, the group had indeed preferred to differentiate deletions from replacements more strongly than by simply having an empty list of replacements.

The current proposal was written from scratch, but draws strongly from the Hoops 2007 and Finney 2003 proposals, as well as every one of the above sources to one degree or another. Some practical decisions are new to this proposal, often due to additional design constraints present in the final incarnation of Level 3, but all of them draw from a wealth of history and fiddling by many different people over the last decade. Where I have differed from the historical consensus, I have attempted to explain my reasoning, but for the most part I took the road most traveled, and have tried, as have my predecessors, to be clear, as simple as possible and as complex as necessary, and applicable to the largest number of situations.

-Lucian Smith, September 13th, 2010