Towards Reusing Model Components in Systems Biology

Adelinde M. Uhrmacher, Daniela Degenring, Jens Lemcke, and Mario Krahmer

Department of Computer Science,
University of Rostock, D-18051 Rostock, Germany
{lin, daniela.degenring, jens.lemcke, mario.krahmer}
@informatik.uni-rostock.de

Abstract. For reusing model components, it is crucial to understand what information is needed and how it should be presented. The centrality of abstraction being inherent in the modelling process distinguishes model components from software components and makes their reuse even more difficult. Objectives and assumptions which are often difficult to explicitate become an important aspect in describing model components. Following the argumentation line of the Web Service ontology OWL-S, we propose a set of metadata which is structured into profile, process model, and grounding to describe model components. On the basis of the specific model component Tryptophan Synthase, its metadata is refined in XML. The reuse of the described model component is illustrated by integrating it into a model of the Tryptophan operon.

1 Introduction

Modelling in general requires a lot of effort, thus the question how models can be reused is a major research effort in modelling and simulation, e.g. [5,10,24,26,30]. Particularly, the availability of software methods that support a modular design of models and their widespread application has revitalised research on reusability of models. Most of the work has concentrated on the technical interoperation of simulation systems, e.g. [13], or how to build simulation systems that support a hierarchical, modular composition of models, e.g. [11,26]. However, progress on developing model components that can be reused by third parties for different objectives has been slow. One of the central reasons might be that capturing the semantics of a model component in an unambiguous way has so far been elusive.

Much of the subsequent is based on the following assertion: A model is an abstraction of a system to support some concrete objective. Thus, we follow the definition of Minsky [27] that "A Model (M) for a system (S) and an experiment (E) is anything to which E can be applied in order to answer questions about S.". As Cellier [9] points out, this defintion does not describe "models for systems" per se. A model is always related to the tuple system and experiment. A model of a system might therefore be valid for one experiment and invalid for another. In consequence of this definition, it is very unlikely to derive a model being valid for all possible experiments, unless it is an identical copy of the system and thus no longer a model. Modelling is a process of abstraction. It involves simplification and omission of details. Which simplifications are permissible and what details can be omitted, depends on simulation objectives. Therefore, all valid reuse must take the objectives of developing this specific model component into account.

In the following, we will explore possibilities to capture the meaning of a model component to support its reuse in Systems Biology. The paper will be structured as follows. First we will explain the concept of model components and take a look at related efforts in Computer Science. Afterwards, we will suggest a set of metadata to describe the syntax and semantics of model components. Within a modular, hierarchical model of the Tryptophan Synthase at cell level, we will identify possible model components for reuse. An enzyme responsible for the last reaction step of the Tryptophan synthesis will be singled out as a possible candidate. We will fill in the metadata for the identified model component and show the reuse of the model component in a different context. Afterwards, we will conclude by discussing related work in modelling and simulation and in Systems Biology.

2 Model Components

Model components and software components have much in common. Both are units of independent deployment and third-party composition. They should come with clear specifications of what they require and provide. A component should be able to plug and play with other components [20]. Software components offer a service to an unknown environment. Thus, approaches developed to facilitate the identification and discovery of suitable Web Services could possibly be exploited for our purpose. For example, the OWL-based Web Service ontology OWL-S (Ontology Web Language for Services) supplies Web Service providers with a core set of markup language constructs to describe the properties and capabilities of their Web Services in an unambiguous, computer-intepretable way [1]. Based on the Resource Description Framework (RDF) expressing simple semantic relations, DAML+OIL (DARPA[1] Agent Markup Language, Ontology Interface Layer) provides mechanisms for describing complex class and property hierarchies. The Web Services classification OWL-S (formerly DAML-S) utilises this framework to unambiguously classify software components in an ontology while including semantic information. The broad aim is to build up a so-called Semantic Web linking specifications of multiple distributed services together to be used in many applications. Central issue of this approach is to provide a precise description expressed in a formal language to support

- Automatic discovery,
- Automatic invocation,
- Automatic composition and
- Automatic monitoring of Web Services.

[1] Defense Advanced Research Projects Agency

To accomplish these goals, OWL-S is structured into three parts:

1. The service profile for advertising and discovering services. It answers the question: What does the service require of the user(s) or other agents and what does it provide for them?
2. The process model, which gives a detailed description of a service's operation and answers the question: How does it work?
3. The grounding, which provides details on how to interoperate with a service via messages and answers the question: How is it used?

Although having much in common, model components also differ from software components, because they always represent an abstraction of reality. The problem of reusing model components thus seems to be even more difficult than the reuse of software components. Due to different objectives in the modelling process, alternative models of the same physical system may equally use different abstractions. Explorative modelling, multi-resolution and multifaceted modelling emphasise the importance of developing families of models and recognising the existence of multiplicities of objectives and models as a fact of life [14,35]. Selecting a model for reuse requires an understanding of its meaning, part of which refers to its objectives, constraints as well as underlying assumptions. Those are difficult to capture, because they often are only implicitly included, and modellers might not even be aware of many of them.

Components as "a self-contained, interoperable, reusable and replaceable unit that encapsulates its internal structure and provides useful services to its environment through precisely defined interfaces" facilitate the development of models by composition as could be shown in empirical studies [33]. Thus, the number of component-based approaches and component libraries offered by commercial simulation systems is steadily increasing [6]. However, component libraries have so far been restricted to well established, highly specialised and well understood application areas of modelling and simulation offering mostly low-level, prefabricated components whose underlying assumptions and constraints are commonly known. Improving model reuse for areas like Systems Biology requires a significant development in several areas [29]. This includes

– understanding what information is needed to support reuse and how it should be presented,
– developing mechanisms to collect and record this information,
– understanding how to design for reuse,
– developing advanced search tools to locate model components and
– developing criteria to decide when model reuse is desirable.

In the following, we will focus on the questions: What information about model components is needed to support reuse not only syntactically but also semantically, and how can this information be represented? Finding answers seems more tractable if the questions are restricted to a specific domain of interest, as in our case the area of Systems Biology.

3 Defining Metadata for Model Components

The information that we distinguished as crucial in describing a model component will be stored in a set of metadata whose structure resembles OWL-S [1] and the description structure proposed in [19].

The **model component profile** contains the information for advertising and searching a model component. It should contain the overall application domain, the name of the model component, the name of the entity or process that it models, the objective of the model, a short textual description and the simulation environment or simulation formalisms it has been designed for. In Systems Biology, a link to general taxonomies like the gene ontology and the enzyme ontology can be established by the corresponding indices. In [19], the validation, assumptions and the non-domain-dependent classification is included in the profile. In contrast, we moved the former two into the "grounding" and assigned the latter to the "process model".

The **component's process model** describes how the model component works. It thus contains the non-domain-dependent classification of the model's type and a kind of abstract description of the internal processing. If a service is described, the process model is used to specify the coordination strategies based on which the service interacts with other services. Generally, there is typically no interest to reveal the "internals" of software components or services. However, as discussions among modellers showed [2], knowledge about its internals increases the trust into a model component.

The **model component's grounding** contains the interface of a model component. In some areas like embedded systems, models contain interfaces to externally running software. This is hardly the case in Systems Biology. Anyway, answering the question how a model component is used implies to answer the question how it interacts with other models, and whether and how it interacts with the user. Furthermore, before reusing a component, it needs to be parametrised. How to use a model component is also restricted by the underlying assumptions and constraints. They therefore have to be described as well.

4 Tryptophan Synthase — Selection of a Model Component

The Tryptophan Synthase model proposed by [16] has been developed in JAMES [32], which supports modular, hierarchical modelling. Thousands of enzyme models are responsible for converting the incoming metabolites IGP, G3P, Indole and Serine to their products including Tryptophan. Their inputs and outputs are served by a single bulk solution model managing the current amount of freely floating metabolites.

Figure 1 shows part of the GUI of JAMES, which is currently under development [7]. In the upper left corner, the overall hierarchical structure of the model is shown. In the lower left corner, the behavior of an α-subunit is specified as a

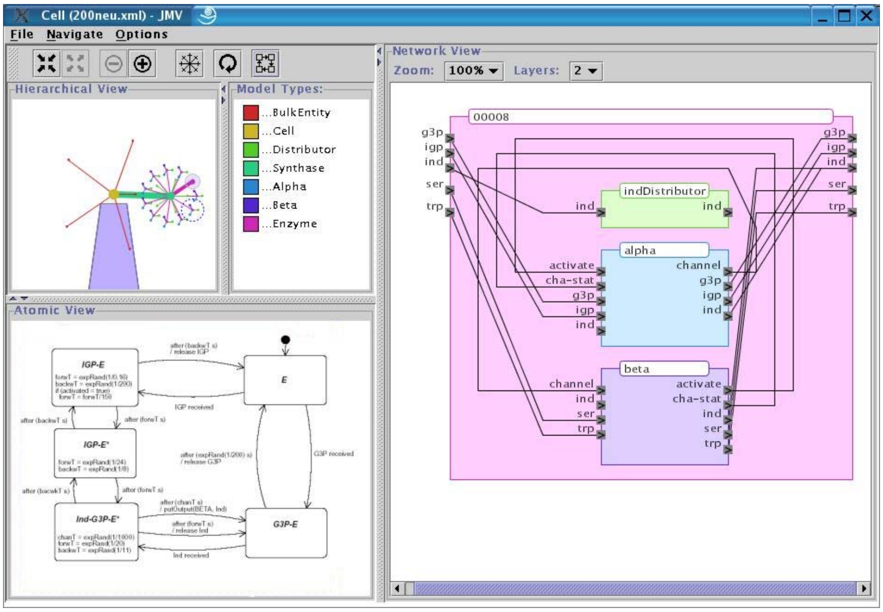

Fig. 1. Screen shot showing the three different perspectives [7]

statechart. At the right hand side, we see the structure of an enzyme with its subunits and their interaction via a static channel.

When identifying model components being eligible for reuse, we face three potential candidates: the population of enzymes, the bulk solution and single enzymes. The bulk solution serves as an experimental frame to the thousands of enzymes. Thus, candidates for reuse are either the population of homogeneous enzymes or one single enzyme. This leads us to the problem of selecting a suitable granularity of components to be stored in a reuse library. If components are at too low a level, the modelling process resembles coding from scratch. If model components are high-level aggregates, then their reuse is limited. Independent of the application domain, the simulation system already offers specialised, coupled model components that can be parametrised to contain an arbitrary number of model components of certain classes interacting in a homogeneous manner. In addition, the properties of the population are entirely specified by its members, the single enzymes, which comprise two subunits, the α- and the β-subunit interacting with each other via the static channel. Thus, from their complexity, they seem to form suitable building blocks for models.

5 Metadata for the Model Component Tryptophan Synthase

Based on the example of the Tryptophan Synthase, the metadata of this model component, whose overall structure has been discussed in Sect. 3, will be refined and filled in an XML format. XML is widely used for storing and exchanging models. For a general discussion see [17] and for Systems Biology see Cell Markup Language (CellML) [12] and Systems Biology Markup Language (SBML) [3]. In addition, an XML specification will allow to exploit multimedia databases for an efficient storage and retrieval of model components [23].

The current name in the profile (Fig. 2) reflects the fact that several models of Tryptophan Synthase might exist, which might vary referring to the model formalims used or with respect to the model's abstraction level. For example, `AAA` refers to the pathway, i.e. aromatic amino acid biosynthesis, `Trp` is a shortform for the Tryptophan Synthase, `Ecoli` is the organism — as the properties of enzymes differ between different organisms. To keep the figure simple, the `name` of the

```
<profile>
  <name>Enzyme.AAA.Trp.Ecoli.DEVS.Model_1</name>
  <application_domain>Enzymology;Systems Biology</application_domain>
  <domain_dependent_classification>
    GO:0004834 ; EC:4.2.1.20
  </domain_dependent_classification>
  <text_description>
    Tryptophan Synthase is an enzyme classified by the E.C. number:
    EC 4.2.1.20. The described enzyme stems from the organism E.coli.
    It produces Tryptophan and Glycerole-3-phosphate and
    consumes Serine, Indole-glycerole-3-phosphate.
    The model component is developed using the DEVS formalism.
  </text_description>
  <objective>
   Analysing the static channeling effect; Analysing the behavior of
   single enzymes.
  </objective>
  <responsible_persons>
    <person function="developer">
      <name>Daniela Degenring</name>
      <e-mail>daniela.degenring@informatik.uni-rostock.de</e-mail>
    </person>
  </responsible_persons>
  <simulation_environment>
    <simulation_system>
      <name>James</name>
      <version type="direct_match">CoSA 1.0</version>
      <subclass_of>
         <name>DEVS-Simulators</name>
         <information type="URI">
           http://www.sce.carleton.ca/faculty/wainer/standard/
         </information>
      </subclass_of>
    </simulation_system>
    <platform>
      <name>Java</name>
      <version type="same_or_above">1.4.1_03</version>
    </platform>
  </simulation_environment>
  <executable type="uri">
    http://www.informatik.uni-rostock.de/~dd012/models/trpsynth.jar
  </executable>
  <references type="uri">
    http://www.informatik.uni-rostock.de/~dd012/models/trpsynth.bib
  </references>
</profile>
```

Fig. 2. The profile of the model component

```
<process>
  <modelling_and_simulation_classification>
    <separated_model_and_simulation>yes</separated_model_and_simulation>
    <model>
      <class>Discrete-Event</class>
      <formalism>DEVS</formalism>
      <topology>Coupled model</topology>
      <scale_of_variables>Qualitative</scale_of_variables>
      <scale_of_variables>Semi-quantitative</scale_of_variables>
      <type_of_events>Situation-triggered</type_of_events>
      <type_of_events>Time-triggered</type_of_events>
      <stochastics>
        <applied>yes</applied>
        <distribution>Exponential</distribution>
      </stochastics>
      <world_view>Process-based</world_view>
      <world_view>Statecharts</world_view>
    </model>
  </modelling_and_simulation_classification>
  <metamodel>
    <type>Statechart</type>
    <figure type="uri">
      http://www.informatik.uni-rostock.de/~dd012/models/trpsynth/doc/chart
    </figure>
    <animation type="uri">
      http://www.informatik.uni-rostock.de/~dd012/models/trpsynth/doc/chart.avi
    </animation>
  </metamodel>
  <expected_effects_of_parameter_change>
    <text>
      Effects of changing tunnel-capacity see Degenring (2003)
      For other effects see Anderson (1995)
    </text>
  </expected_effects_of_parameter_change>
</process>
```

Fig. 3. The process model of the model component

```
...
<xsd:element name="model">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="class" minOccurs="1" maxOccurs="unbounded">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Continuous"/>
            <xsd:enumeration value="Discrete-event"/>
            <xsd:enumeration value="Discrete-stepwise"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      ...
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
...
```

Fig. 4. Exemplary XML Schema fragment for defining the type of model

model component is represented by a plain string of characters. However, to enable an automatic model retrieval, a more structured way of defining the `name`'s constituents is needed, e.g. by a hierarchy of XML tags. In the same way, the `domain_dependent_classification` has to be refined, e.g. by using XML namespaces.

The model's classification is part of the process model (Fig. 3). It refers to the type of model, e.g. whether it is a continuous, discrete-event or discrete-

```
<grounding>
  <model_component_to_model_component>
    <input type="materialistic" unit="1">IGP</input>
    ...
    <input type="materialistic" unit="1">Serine</input>
    <output type="materialistic" unit="1">Tryptophan</output>
    ...
    <output type="materialistic" unit="1">Indole</output>
    <output type="informational">Enzyme.dead</output>
  </model_component_to_model_component>
  <model_component_to_model_user>
    <output type="model_state">Internal phase</output>
    <output type="statistical">Tunnel occupation</output>
  </model_component_to_model_user>
  <invariant type="m_c_to_m_c">
    <sbml xmlns="http://www.sbml.org/sbml/level1" level="1" version="1">
      <model name="Tryptophan Synthesis">
        <reaction name="IGP + Ser -> G3P + Trp">
          <listOfReactants>
              <specieReference specie="IGP" stoichiometry="1"/>
              <specieReference specie="Ser" stoichiometry="1"/>
          </listOfReactants>
          <listOfProducts>
              <specieReference specie="Trp" stoichiometry="1"/>
              <specieReference specie="G3P" stoichiometry="1"/>
          </listOfProducts>
        </reaction>
      </model>
    </sbml>
  </invariant>
  <integration_with_other_models>
    <text>
      as part of a population within a multi-level, or micro
      model for analysing purposes,
      single enzyme combinations for education purposes
    </text>
    <documented_use_of_model_component>
      <text>
        Degenring (2003); Degenring (2004)
      </text>
  </integration_with_other_models>
   .
   .
   .
```

Fig. 5. The interface specification as part of the grounding the model component

stepwise model. Again, the `model`-tag contains just a plain string of characters to define the classification. In order to give an unambiguous description, it would be desirable for simulationists to agree upon some common and precise terms to identify and specify these properties (Fig. 4).

In the process model (Fig. 3), the tag `metamodel` refers to a metamodel, which forms a model of the executable model component. The metamodel shall provide a more abstract, often visual representation to the user, thus facilitating the understanding of a components' potential dynamics. In JAMES, if the model component is sufficiently simple, it can directly be specified as a statechart and no further abstraction via a separate metamodel is needed. However, often the statechart will form an abstraction and thus can be interpreted as a metamodel of the underlying atomic model in JAMES. In this context, statecharts are only one possibility to

```
     .
     .
     .
    <parametrisation>
      <parameter minValue="gt0" maxValue="not_yet_explored"
                                default="0.02222" unit="sec">
        E-Ser_to_E~AA
      </parameter>
      ...
      <parameter minValue="gt0" maxValue="not_yet_explored"
                                default="0.001" unit="sec">
        Ind-G3P-E*_to_Channel
      </parameter>
      <parameter minValue="gt0" maxValue="not_yet_explored"
                                default="150" unit="1">
        IGP-E_to_IGP-E*_Speed-up
      </parameter>
    </parametrisation>
   <underlying_assumptions>
      <text>
        The model deals with molecule numbers instead of concentrations.
        Due to the transformation from a differential equation to a
        stochastic
        discrete-event model according to Gillespie (1976), more than 100
        model components of this type should possibly be included for
        analysing purposes.
      </text>
    </underlying_assumptions>
    <validation>
      <text>
        Reproduction of original experimental and simulation results
        Anderson (1995).
        Additionally, the effect of the tunnel capacity was investigated
        with plausible results Degenring (2003).
        Ongoing work: implementation of the model into a Trp-operon model
        according to Santillan (2001).
      </text>
    </validation>
  </grounding>
```

Fig. 6. The parametrisation, assumptions, and validation as part of grounding the model component

define a metamodel: e.g. live sequence charts, e.g. [21], Petri Nets, e.g. [28], graphs, e.g. [22], and even pseudocode are alternatives. Regarding the metamodel, it is important that it contains a declarative and easy to understand description of the underlying model. However, what is easier to understand depends on the context of the user. Alternative descriptions should therefore be included. Developing different metamodels of a single model component requires a lot of effort and has to be balanced to the effort of directly inspecting its source code.

The main part of the grounding is concerned with the specification of the interface (Fig. 5). Most inputs and outputs are of type `materialistic` which means that they are not multiplied if they are sent to multiple addressees. In contrast, information is a non-consumable ressource and thus can be multiplied. Single molecules form the inputs and outputs of our model component. Between those inputs and outputs of a model component, invariants are defined. In the case of our enzyme, the mass conservation law holds between consumed and produced species, which has been specified according to the SBML [3] suggestion for chemical reactions.

Model components include a notion of time. In simulation, we typically distinguish between three types of time: the physical time, which refers to the time of the modelled physical system; the simulation time, which refers to the representation of the physical time within the simulation; and the wall-clock time, which refers to the time that progresses during executing the simulation. In the above example, one unit of simulation time corresponds to one second in physical time. The model component gives a stochastic, discrete-event description of the behaviour of an enzyme. Thus, methods to estimate parameters and to analyse the output of stochastic, discrete-event simulation have to be considered. For a discussion of implications of using stochastic simulations in Systems Biology see e.g. [25].

6 Reuse of the Model Component

The major role of the Tryptophan Synthase lies in the enzyme cascade for the production of Tryptophan. The Tryptophan Synthase is produced by transcription and translation of the Tryptophan operon. The process of transcription and translation is regulated by the amount of Tryptophan. So we could imagine to combine our model component with other enzymes to capture the entire aromatic amino-acid biosynthesis, or we could integrate it with a model of the Tryptophan operon.

As Trp-enzyme catalyses the last reaction step in the Tryptophan synthesis, the model component can be combined with other enzyme models, which proliferate the substrates of the reaction, i.e. mainly the IGP Synthase or rather the Serine Synthase, and corresponding enzymes. A coupling with enzymes, which consume the built products, G3P and Tryptophan, like the G3P Dehydrogenase catalysed reaction (EC 1.1.99.5) or the Trp-tRNA-synthesising enzyme, is also possible and would direct the focus to their products. In addition, the products of the Tryptophan model component can interact with other model components as effectors. The inhibiting influence of Tryptophan onto the transcription of the Trp-operon via the process of attenuation or gene repression is e.g. a known fact. Since all metabolites participate at the same time in other reactions, no direct coupling between various enzymes is proposed but an indirect interaction via a model like our bulk solution might prove beneficial. The model of the bulk solution records the amount of available metabolites, enzymes, genes, and mRNA, and calculates the frequency of collissions.

Whereas the above combinations act on more or less the same time scale abstraction, analysing the regeneration of the Trp Synthase requires to combine the model component Trp Synthase with other models that describe the operon and thus act at a different time scale. Here the model component Trp-operon produces the model component Trp Synthase during the process of transciption and subsequent translation. This type of model requires that the simulation system supports variable structure models, i.e. models that change their composition and interaction during simulation as e.g. James does.

We integrated several hundreds of the model component Tryptophan Synthase into a model that describes the transcription dynamics of the Tryptophan operon focussing on gene regulation and monitoring switch activities. First experiments executed in James were able to reproduce the results documented in [31], which was no big surprise considering the description of the model component Tryptophan Synthase. The model component Tryptophan Synthase had been validated under similar assumptions, i.e. assuming comparatively high numbers of the Tryptophan model component, and the integration with the operon has even been foreseen by the model component's description. Dealing with different time scales is the virtue of discrete-event simulation, so no errors should have been induced by the simulation engine. The only question was, whether a simpler model of the enzyme would not have sufficed, as the role of the channel was not the objective of the current simulation study. However, this question has still to be explored. The integration of the current model component would allow to test more sophisticated hypotheses about structural interdependencies within the overall gene expression process. For example, if the objective had been to test the hypothesis how a damaged gene affects the capacity of the channel, the model component Tryptophan Synthase with its explicit subunits and the channel would have appeared most suitable.

7 Related Work in Systems Biology

Major efforts in modelling and simulation are aimed at supporting the reuse of models and entire simulations. The High Level Architechture (HLA) [13] is a general purpose architecture for simulation reuse and interoperability. Developed under the leadership of the Defense Modelling and Simulation Office (DMSO), it provides a standard means (IEEE 1516) for individual simulations or federates to interoperate in a federation since 1996. HLA is mostly concerned with synchronising different simulations via the Run-Time Infrastructure (RTI) and thus with the question of interoperation [8]. However, for HLA to work, the interoperable federates share a common Federation Object Model (FOM) document that describes the types of information they are exchanging. This metadata not only includes information needed for synchronisation and coordination by the RTI software such as classes, attributes, and interactions, but also includes descriptive information such as the objective and sponsor of the federation or federate. HLA focuses on the concrete synchronisation of different simulation systems rather than on the retrieval of models.

Other major research efforts are directed towards improving the exchange of models between simulation systems in Systems Biology. Representatives are the Systems Biology Markup Language (SBML) and the Cell Markup Language (CellML) [12]. Whereas SBML is meant to support basic biochemical network models, CellML covers a more general field of application including electrophysiological and mechanical models as well as biochemical pathway models. Both provide in addition to parameter definitions information about the equations of the underlying biological processes such as reaction mechanisms. Those can be executed by simulation systems supporting an SBML or CellML interface. Thus, a complete, unambiguous description of the model is required to enable these different simulation systems to execute it.

Additionally, CellML also supports the assignment of metadata to facilitate the reuse of model components by providing background information. Similarly to our approach, e.g. information on the species is given by referring to established ontologies and making use of the biological databases on the web. The underlying assumptions are covered in the limitations and validation slots. The type of model is characterised by referring to an ontology of mathematical problems as CellML and SBML focus on continuous system models.

In contrast, our approach is not restricted to the continuous realm. It therefore becomes important to explicitly represent what model formalism is used. Whereas continuous models can typically be described by differential or mathematical equations, no such general exchange format does exist for models belonging to different modelling formalisms. To execute and thus eventually reuse a model, a link to its specifation and simulation engine or its entire implementation is often unavoidable. Thus at this point, the reuse of general models reveals a Web Service characteristic. Current efforts like multi-formalism modelling are directed towards facilitating the reuse of models of different formalisms [34,15]. The idea is to provide a meta description of these formalisms. So in the future, it might be possible to include the specifiation of the model and a meta description of the formalism used.

8 Conclusion

The reuse of components promises to facilitate the design of models. We suggested a set of metadata to describe model components and illustrated its use with the model component Tryptophan Synthase. The overall structure of the metadata was influenced by the ontology for Web Services OWL-S, whose features profile, process model and grounding have been redefined in this context. The metadata are structured to distinguish between information used for retrieving a component, i.e. profile, information about how the component works, i.e. the process model, and the specification of its interface, i.e. grounding. As do [29], we believe that a key part of a model description must involve capturing the objectives, assumptions and constraints under which the original models were developed in a form that can be searched and analysed. Currently, our proposed metadata contains still many text slots. For an advanced search that would be able to identify model components of interest, for consistency checks to recognise incompatibilities among model components, and to help determining the fidelity of the entire model, the proposed metadata present only a first step.

Compared to approaches like SBML and CellML, our approach focuses on the retrieval of components rather than on supporting the exchange of models between simulation systems. Our long term goal is similar to that of OWL-S: providing meta information about a model in a formal, rigorous manner to support the automatic discovery and composition of model components. For the final execution of models, we will depend on efforts like SBML and CellML or, as our approach is not restricted to a particular modelling and simulation paradigm, on efforts like multi-formalism modelling and simulation.

As in the case of modelling in general, we are confronted with the problem to provide as little information as needed but not less. Many modellers might find the inspection of the source code more informative. Of course, this would not allow to manage model component repositories in an effective and efficient manner. Still the question is, whether researchers are likely to use and maintain these repositories, and whether we now understand what information is needed to support reuse of model components. The answer can not be given by us but can only found in discussions with third-party users. Therefore, an integration into efforts like CellML or SBML is mandatory.

Acknowledgements

We would like to thank the anonymous and non-anonymous referees Jan Himmelspach and Mathias Röhl for their comments on an earlier version of this paper. Part of the research presented in this paper has been funded by the DFG.

References

1. http://www.daml.org/services/owl-s/1.0/.
2. http://www.dagstuhl.de/04041/.
3. http://www.sbml.org/docs/.
4. K.S. Anderson, A.Y. Kim, J.M. Quillen, E. Sayers, X.J. Yand, and E.W. Miles. Kinetic characterization of channel impaired mutants of tryptophan synthase. *The Journal of Biological Chemistry*, 270(50):29936–29944, 1995.
5. J. Aronson and P. Bose. A model-based aproach to simulation composition. In *Proceeding of the Fifth Symposium on Software Reusability*, pages 73–82, 1999.
6. F. Barros and H. Sarjoughian, editors. *Special Issue on Component-Based Modelling and Simulation.* Simulation - Transactions of the SCS Simulation. Sage, 2004.
7. S. Biermann, A.M. Uhrmacher, and H. Schumann. Supporting multi-level models in systems biology by visual methods. In *Proceedings of European Multi-Simulation Conference*, page submitted, 2004.
8. A. Buss and L. Jackson. A comparison of hla, corba, and rmi. In *Proceedings of the 1998 Winter Simulation Conference*, pages 819–825, 1998.
9. F. E. Cellier. *Continuous System Modeling.* Springer, New York, 1992.
10. G. Chen and B. K. Szymanski. Object-oriented paradigm: component-oriented simulation architecture: toward interoperability and interchangeability. In *Proceedings of the 2001 Winter Simulation Conference*, pages 495–501, 2001.
11. G. Chen and B. K. Szymanski. Cost: A component-oriented discrete event simulator. In *Proceedings of the 2002 Winter Simulation Conference*, pages 776–782, 2002.
12. A.A. Cuellar, C.M. Lloyd, P.F. Nielsen, D.P. Bullivant, D.P. Nickerson, and P.J. Hunter. An overview of cellml 1.1, a biological model description language. *Simulation - Transactions of the SCS*, 79(12):740–747, 2003.

13. J. Dahmann, R. Fujimoto, and R. Weatherly. The dod high level architecture: An update. In *Proceedings of the 1998 Winter Simulation Conference*, pages 797–804, 1998.
14. P.K. Davis, J.H. Bigelow, and J. McEver. Exploratory analysis and a case history of multi-resolution, multiperspective modeling. Technical Report RP-925, RAND, 2000.
15. J. de Lara and H. Vangheluwe. AToM3: a tool for multi-formalism and meta-modelling. In *European Joint Conference on Theory And Practice of Software (ETAPS), Fundamental Approaches to Software Engineering (FASE)*, pages 174 – 188, 2002.
16. D. Degenring, M. Röhl, and A. M. Uhrmacher. Discrete event simulation for a better understanding of metabolite channeling- a system-theoretic approach. In Priami C., editor, *Lecture Notes in Computer Sciences*, volume 2602, pages 114–126. Springer Verlag Heidelberg, 2003.
17. P.A. Fishwick. Using xml for simulation modeling. In *Proceedings of the 2002 Winter Simulation Conference*, 2002.
18. D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22:403–434, 1976.
19. M. Heisel, J. Luethi, A.M. Uhrmacher, and E. Valentin. A description structure for simulation model components. In *Proceedings of the Summer Simulation Conference*, page submitted, 2004.
20. B. Hnich, T. Jonsson, and Z. Kiziltan. On the definition of concepts in component based software development. Technical report, University Department of Information Science, Uppsala, Sweden, 2000.
21. N. Kam, D. Harel, H. Kugler, R. Marelly, A. Pnueli, E.J. Hubbard, and M.J. Stern. Formal modelling of c. elegans development: A scenario based approach. In Priami C., editor, *Lecture Notes in Computer Sciences*, volume 2602, pages 4–20. Springer Verlag Heidelberg, 2003.
22. H. Kitano. A graphical notation for biochemical networks. *Biosilico*, 1(5):169–176, 2003.
23. M. Klettke and H. Meyer. *XML & Datenbanken - Konzepte, Sprachen und Systeme.* DPunkt Verlag, 2003.
24. G. Mackulak and F. Lawrence. Effective simulation model reuse; a case study for amhs modeling. In *Proceedings of the 1998 Winter Simulation Conference*, pages 979–984, 1998.
25. M. Marhl. Transition from stochastic to deterministic behaviour in dependence on the divergence of systems. In *3rd Workshop on Computaiton of Beiochemical Pathways and Genetic Networks*, pages 49–58. Logos Verlag, 2003.
26. Y. Miller, J. A. Ge and J. Tao. Component-based simulation environments: Jsim as a case study using java beans. In *Proceedings of the 1998 Winter Simulation Conference*, pages 373–381, 1998.
27. M. Minsky. Models, Minds, Machines. In *Proc. IFIP Congress*, pages 45–49, 1965.
28. M. Nagasaki, A. Doi, H. Matsuno, and S. Miyano. Genomic object net: a platform for modeling and simulating biopathways. *Applied Bioinformatics*, 2003.
29. C. M. Overstreet, R.E. Nance, and O. Balci. Issues in enhancing model reuse. In *First International Conference on Grand Challenges for Modeling and Simulation*, 2002.

30. E. Page and J. Opper. Observations on the complexity of composable simulation. In *Proceedings of the 1999 Winter Simulation Conference*, pages 553–560, 1999.
31. M. Santill'an and M. C. Mackey. Dynamic regulation of the tryptophan operon: A modeling study and comparison with experimental data. *Proceeding of the National Academy of Sciences of the USA*, 98(4):1364–1369, 2001.
32. A. M. Uhrmacher, P. Tyschler, and D. Tyschler. Modeling Mobile Agents. *Future Generation Computer System*, 17:107–118, 2000.
33. E. C. Valentin, A. Verbraeck, and H. G. Sol. Effect of simulation building blocks on simulation model development. In *Proceedings of the International Conference of Technology, Policy and Innovation*, pages CD–ROM proceedings, 2003.
34. H. Vangheluwe and J. de Lara. Meta-models are models too. In *Proc. of the Winter Simulation Conference*, pages 597 – 605, 2002.
35. B.P. Zeigler. *Multifacetted Modelling and Discrete Event Simulation.* Academic Press, London, 1984.