Using Mathematical and Scientific Markup as an Approach to Model Specification

Joseph B. Collins
Naval Research Laboratory, Code 5583
4555 Overlook Ave, SW
Washington, DC 20375-5337
joseph.collins@nrl.navy.mil

Keywords: physics, modeling, mathematics, markup, system specification

Abstract

Complex systems need to be modeled in a way that supports validation, flexible composition of component models and component model re-use. A methodology that supports these requirements must: support conceptual modeling, i.e., mathematical description, and; document the semantics embodied in a model. The expression of mathematical semantics alone will not suffice. For physics-based models, the semantics must also identify those physical laws that were applied by the modeler. Only in this way can one adequately document the particular modeling decisions made in the model's construction.

The development of Markup Languages has enabled the Semantic Web, with the universal document type being expressed in the Extensible Markup Language (XML)[1]. Higher level semantic layers may be defined in XML applications such as is done with the Mathematical Markup Language (MathML). The mathematical semantics embodied within Content MathML are currently being expanded beyond the expression of "grade school mathematics" to enable the expression of higher level mathematics by the use of *content dictionaries*. These mathematical content dictionaries contain the definitions of semantic XML tags so that the use of well-defined mathematical concepts in a document may be unambiguously expressed by application of these tags. Building on top of this base, scientific markup languages are beginning to be developed.

In particular, we have identified several requirements and some solutions towards the development of a physics markup language. For example, it must be embedded within a documentation standard that supports an accompanying natural language description. The physics markup language itself must have the ability to support the expression of physical *models*. This includes the concepts of: *physical objects*, or physical bodies, which have *physical observables* as properties; and *physical laws*, which when applied to physical objects results in model equations, or other relations. Physical objects have spatio-temporal extents which are usually functions of time. These concepts must be translated into XML *elements* so that the elements may be used by modelers to express their models in a straightforward way using semantic tagging. In this paper we will discuss progress to date, including: the identification of Open Mathematical Documents (OMDoc) [2] as a candidate documentation system; the definition of an abstract data type for the expression of physical variables; and the definition of a physics-based model as a set of constraint relations defined on a set of physical objects.

1. INTRODUCTION

Increasingly, computer models are used not only to design, develop, or analyze just a single aspect of a new product, but to co-design and co-develop multiple aspects and to analyze their interactions. Analyzing the complex interactions of geometry with physical properties such as thermal and mechanical transport, vibration, electrical behavior, etc., requires integrated computational support. We believe that it is possible, and desirable, to provide a documentation framework to support interchange of model specifications between computational applications for design and analysis of those models. To support this goal and others, we are developing a physics markup language.

While there are many aspects to system representation, we envision a physics markup language to be used to represent the physics-based aspects of systems. When large physical systems are modeled it may not be appropriate to just compose a model of independent, individually modeled components. In order to correctly model physical systems, we must often have models that are integrated at the conceptual, or mathematical level. Only in this way can we correctly model the behavior of a whole system with strongly interacting parts. We describe here some of the mathematical representation issues encountered when doing this.

A basic choice we have made in this endeavor is to make use of the expressiveness offered by the Semantic Web technologies, i.e., XML. XML may be thought of as a universal data representation language. This is relevant because we include in our thinking the idea that model specification requires that we remove the commonly assumed boundary between the idea that models are distinct from data: we must treat models as a kind of data. To do this, we focus particularly on those technologies dealing with the representation of mathematical semantics.

2. THE PHYSICAL OBJECT CONCEPT

Our starting point is to first discuss how to represent the things or nouns of the proposed language. A *physical object* is a unification of the concept of a *body* in physics (as used in Newton's Definition I and elsewhere [3] , also Einstein's *Körper* [4, 5]) with the computer science concept of an *object* (see, for example, [6]), represented as a (mathematical) relation of attributes and behaviors. The term, physical object, refers to the class of objects that represents tangible, concrete entities that are presumed to be governed by the laws of physics; those objects with attributes that represent the physical observable properties of such concrete entities. With the exception of an identity attribute, other attributes that do not represent observable properties governed by the laws of physics are extraneous to our treatment here.

The computer science concept of an object enables one to couple state, or attribute values, together and also to associate those linked attributes with behaviors that depend on that state. There are some concepts, such as inheritance, embodied in the object concept that may not be necessary to representing physical objects but the object concept does seem sufficient for representing physical objects, which require part-of hierarchy for representing aggregate objects, and, as we will see, abstract data types. The concept of a physical object narrows the computer science concept to be restricted to only those properties necessary for representing concrete physical objects that have observable and measurable attributes. This should not impede in any way the use of the physical object concept within disciplines that make use of the broader properties of general objects.

The attributes of a physical object are generally those properties that can be measured, such as mass, length, time, current, temperature, etc., and assigned physical units, such as kilogram, meter, second, ampere, Kelvin, etc. These are generally regarded as fundamental, being ensconced in international standards for many years. We would also include state representations such as the quantum wave function, even though strictly it is the expectation value of its Hermitian operators that are measurable: the wave function is generally needed for the representation of those observables.

One might question whether the concept of a physical object is as fundamental as, say, the concept of a measurable observable. A physical object is more abstract since it cannot itself be directly measured – only its attributes may be measured. When we model physical systems we often use the idea of macroscopic primitive physical objects, which are not aggregate objects: a perfectly conducting sphere, a rigid cube, etc. But, we do not believe these are fundamental: we don't necessarily believe that such perfect objects exist. They really represent a rather ad hoc way of dividing up the perceptible world. The only instances of primitive physical objects to represent anything fundamental are, perhaps, the various elementary particles. We cannot, however, mistake how fundamental the *concept* of a physical object is with how fundamental, or not, are most instances of that concept that we use. We must remember that the concept of a physical object provides the context, and hence, an essential part of the meaning of the measurements of the attributes. One thing that the object concept provides is the concept of a reference frame. Since Einstein, it is generally agreed that the notion of an absolute reference frame is physically meaningless. In physical reality, reference frames must have a basis defined by physical objects.

3. AN ABSTRACT DATA TYPE FOR PHYSICAL OBSERVABLES

Earlier [7] we posed the question: "Is there a mathematical type for physical variables?", meaning by the term physical variable the symbols that we use to represent the values of the observable properties of physical objects. Many physicists and engineers may be unconcerned with this question, acting as specialists in a well defined specialty and comfortable with the standard notation of that specialty. The problem of having communicating models where classical, relativistic, and quantum physics models interact, however, requires that we have a mechanism to translate the representation of variables between these models. Mathematical manipulations could be more efficient and clear if they were in a uniform notation. Nevertheless, the desirability of a single mathematical, or abstract, data type for physical variables does not guarantee its existence, nor would its apparent existence guarantee that it would always be so. The future could bring new physical phenomena that would break the old notation.

The answer we arrived at was that, indeed, there appears to be a single abstract data type that can account for most, if not all, of the mathematical objects generally used by physicists and engineers for representing physical variables. Briefly stated, that type is formed as the commutative mathematical product of a physical unit, such as meter or second, and an element of a geometric algebra, which belongs to the family of Clifford algebras. The elements of Clifford algebras are vector-like objects. More interestingly, Clifford algebras possess a vector product that is associative and, with respect to which, each non-zero element has an inverse. Real and complex valued scalars, vectors and tensors may all be represented using geometric algebras. Geometric algebras also contain operators of physical invariance, such as the Euclidean transformations (spatial rotations, translations, and reflections) and Lorentz transformations (space-time rotations, boosts, and reflections). Even the more esoteric Lie group elements such as spinors and quaternions are elements of geometric algebras.

With this understanding we can now formalize the representation of physical objects, which have physical quantities as attributes, as a mathematical relation of elements of a geometric algebra. We would also note that representing a geometric algebra would appear to require polymorphism and inheritance: the elements are sometimes scalars, sometimes vectors, and sometimes multi-vectors, and; because the elements of a geometric algebra have an associative product and an inverse, they inherit group properties. So, while we observed that there appears to be no inheritance hierarchy for physical objects based on physical principles, there appears to be one for their attributes.

4. DEFINING MODELS

What is a model? This is a question whose answer defines the scope of what we may represent. Part of what must be considered is how what we call models are created and used. Physics-based computational models that are used as components in software systems are typically formulated as functional blocks: once values are bound to all of the inputs, a unique output result may then be computed. This functional block form works particularly well for causal predictive models where past states serve as inputs and the model produces output states that represent the future, relative to the inputs. While the functional block form of a model may be a frequently used form, it does not necessarily capture the full meaning of what may be meant by a physics-based model. To understand what a fuller meaning of a physics-based model may be, we review the creative process that gives rise to models. We will arrive at two forms for a model: a specification-form, represented as a constraint satisfaction problem, and a solution-form, represented as a functional-block form. We note that this development is a synthesis of previous work in describing semantics of physics-based models [7-9] and some work that has informed the development of the Systems Modeling Language (SysML) [10-11].

Developing a physics-based model begins with a decision to describe the behavior of one or more physical objects, their behavior, and their interactions with each other. After deciding on the particular behavior and interactions of interest, as well as the level of detail desired, a representation of the attributes of the physical objects are created in terms of spatio-temporal variables, and the physical laws governing the behavior of those objects and their interactions is applied. The physical laws typically are formulated in terms of differential equations involving the variables representing the attributes of the physical objects. Variables may also be assigned to represent the initial conditions for a meaningful representation of a solution to the differential equations. Once the problem is so formulated, one may say "all of the physics are represented". This means that all of the constraints on the behavior of the modeled objects that are imposed by physical principles, at least those considered by the modeler to be relevant, have been applied. It is quite usual for some parameter values to be left unspecified, in particular scenario specific conditions, such as spatio-temporal boundary conditions, or specific material properties such as conductivity, hardness, etc. Once all of the physics have been represented parametrically, what remains for a complete understanding of the model is to explore the solution space of the problem, for example, exploring the range of allowed values of any parameters whose values are unspecified. There may be undiscovered phenomena hiding in the solution space even though the individual model equations are well known.

4.1. Constraint Satisfaction Problems

The result of making the modeling decisions that results in a model is commonly represented by a set of model equations. A complete summary of the model may be phrased as a *constraint satisfaction problem* (see, for example, [12]). Constraint satisfaction problems, or CSPs, are well known in the computer science community, though very often being defined over discrete domains while physics-based models are usually defined over real or complex domains.

Formally, a constraint satisfaction problem is defined by a triple, {X, D, C}, where X is a tuple of all the variables in the problem, D is a tuple of corresponding domains for those variables, and C is a set of constraints over the variables in X. Each constraint may be represented as a pair, {t, R}, where t is an n-tuple of some variables in X and R is an n-ary relation, defining a set of tuples of allowed variable values. For a CSP defined over finite, discrete domains, the relation may take the form of an enumeration of its elements. For infinite domains, such as real-valued physical variables, the relation must be stated more succinctly, such as an equality or inequality.

What makes a CSP most interesting, of course, are the solutions. An evaluation of the variables is a function assigning values to the variables consistent with the variable domains. A solution to a CSP is an evaluation of all the variables that satisfies all of the constraints. In other words, a solution is a way for assigning a value to each variable in such a way that all constraints are satisfied by these values. A CSP is said to be *satisfiable* if solutions exist.

A CSP may be represented as an undirected graph, a constraint graph, where the nodes are the variables and edges represent binary constraints, or constraints between pairs of variables. Higher order, or n-ary, constraints may be represented with a second type of node having edges connected to all of the variable nodes that it affects. The application of a typical physical law gives rise to an n-ary constraint for each pair of physical objects, coupling n physical variables in an equation or inequality.

Because the form of the CSP captures the physics-based requirements, we also refer to it as the *specification-form* of a model.

4.2. Functional Block Models

Many if not most of the computational models resulting from the mathematical modeling process are more specialized than the conceptual model that they are drawn from. After transforming models into a discretized version, they are commonly implemented in an imperative programming language, and are characterized by sequences of variable bindings. While side-effects are not unknown, in the best of cases these models, when implemented for operational use, retain a functional form, i.e., when all of the inputs are known or bound, then the outputs may be explicitly computed and bound. For this reason, we also refer to the functional block model as the *solution-form*, since that is the form that is operationally useful.

It is important to note how the functional block representation of a model is different from the constraint satisfaction problem representation of a model. Take, for instance, a model given by:

F = m*a

This is a statement of a constraint relation, of equality in particular, between the three variables, F, m, and a, and from which we can form a CSP. Taking this particularly simple CSP we can easily derive three different functional block computational models, one for each of the variables, i.e.,

"F := m*a", "m := F/a", and "a := F/m"

Here the variables on the right-hand sides of the assignment statements are inputs and the variables on the left-hand sides of the assignment statements are outputs. These statements not only reflect the equality constraints, but also explicitly specify the sequence of bindings. Because these three variables, F, m, and a, each depend on the other two, their three corresponding function-block models have mutually incompatible precedence relationships: it does not make sense to have any pair of these three functional blocks simultaneously operable. To understand this we need to recall that each input and each output represents the value of a specific observable property of a specific physical object. To compose functional block forms in such a way as to have the binding of a particular observable precede itself doesn't make sense.

From this analysis we see that a significant difference in these representations of models is the binding sequence: it is an essential part of the function-block representation of a model, but not always present in the CSP form. One could conceivably add precedence constraints to the constraint list and even attach physical interpretation to them, such as causality, where initial conditions precede later states. Adding precedence constraints to the constraint list does not appear to be part of the CSP formalism, nor does it appear to be necessary since it does not affect the nature of the solutions to the CSP with respect to the resulting functional equations.

Considering the above example, the function-block representation may be viewed as originating from a CSP solution with the subsequent addition of binding sequence constraints. In this way, one CSP can give rise to multiple function-block models where the CSP represents a set of functional requirements and the function-block representation provides an effective implementation satisfying the precedence requirements in a more specific context. Another significant difference between the CSP representation of models and the function-block representation is that the constraints expressed in the CSP typically hold physical semantic value as representing properties of the solution. The function-block representation, by itself, when it is disconnected from a corresponding CSP description to which it provides a solution, suffers a loss of the information that is embodied in the constraints.

5. COMPOSITION OF MODELS

The composition of models is important to consider, particularly from the objectives of re-use and systems design and analysis. If we want to analyze how two objects will interact, we will want to meaningfully combine, or compose, their respective models. Also, it is quite possible, if not likely, that a given model is actually comprised of multiple sub-models. Understanding how sub-models may be combined to form a given model, or be removed from or be substituted into that given model, also comes under the subject of model composition.

When a model is posited as a functional block, composition of models is most naturally given by function composition. Composition of models in functional block form is very straightforward: bind the outputs of one block to the inputs of a second. This type of composition is used when the first of the two component models provides state values that are used by, but not affected by, the second model.

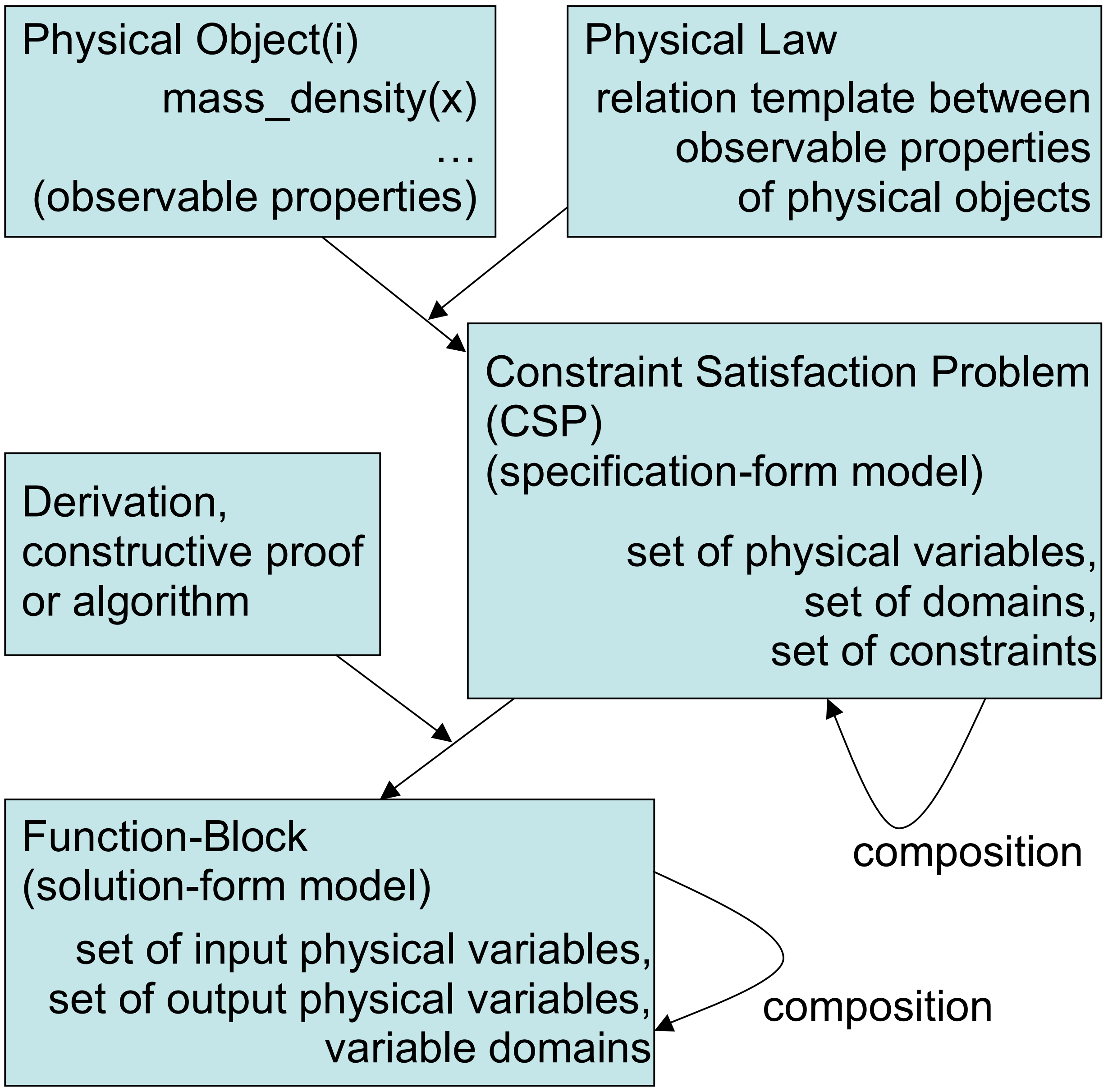
When a model is posited as a constraint satisfaction problem, then composition of two problems is reasonably defined by the union of the variables and the union of the sets of constraint relations, i.e., for

```
{X, D, C} =
   Composition({X1, D1, C1},{X2, D2, C2})
```

Where in the resulting CSP, X is the union of the tuples X1 and X2: variables that occur in either X1 or X2 will be represented once in X and variables that occur in both X1 and X2 are only represented once in X. Since the variable domains, D, D1, and D2, essentially represent unary constraints on the variables, if $X_i = X1_k = X2_m$ then D_i is given by the union of the constraints, which is just the intersection of $D1_k$ and $D2_m$. Domains of variables occurring in only one of the tuples X1 or X2 are the same in D as they are in D1 or D2. Finally, the set of constraints, C, is given by the union of C1 and C2.

Composition of physical objects is achieved by aggregation. A primitive physical object is simply one that is not an aggregated object. An aggregate physical object is comprised of primitive and aggregate physical objects. Aggregating physical objects is largely a modeling choice, but should be motivated by the existence of interactions between the constituent physical objects. Aggregating physical objects must obey the necessary physical conservation laws, i.e., of mass, energy, charge, momentum, etc.

Figure 1. Some semantic concept classes for physics markup


Physical Object(i)
mass_density(x)
…
(observable properties)
Physical Law
relation template between observable properties of physical objects
Constraint Satisfaction Problem (CSP)
(specification-form model)
set of physical variables,
set of domains,
set of constraints
Derivation, constructive proof or algorithm
Function-Block
(solution-form model)
set of input physical variables,
set of output physical variables,
variable domains
composition
composition

6. FULL PICTURE OF THE MODELING PROCESS

Considering what we have discussed, we now put together a full picture of a regular, formal structure to the process of mathematical modeling in physics. This results in semantic concept classes, in Figure 1, which will help us define semantic XML tags. (The discussion of this section refers to Figure 1).

First we identify the physical objects and representations of their measurable properties. The class of physical objects may be described as a mathematical relation of these properties. The modeler may create aggregate objects or primitive (non-aggregate) objects.

The modeler then chooses how to apply different physical laws to the objects. The allowed operations are defined by the algebra of the variables representing the physical observables: in general these are elements of a specified geometric algebra.

The application of physical laws results in equations and inequalities which constitute a set of constraints phrased as a Constraint Satisfaction Problem, or CSP. The CSP form is a well established form for encapsulating a problem definition. The constraints may be annotated with the names of physical laws used to define them.

Even with soundly defined problems, i.e., satisfiable CSPs, there is no guarantee of a path to a solution. Sometimes solutions may be found by the modeler, with or without the aid of computer algebra solvers, or in the literature. The documentation form of the finding of a solution is that of a derivation, constructive proof, or algorithm.

The final solution-form is a function-block. Note that the equality relation, $F=ma$, can result in three distinct function-block forms: $F:=ma$; $m:=F/a$; $a:=F/m$. The difference between these is that assignments impose evaluation precedence, i.e., inputs must be known before outputs may be computed. Computer models are frequently provided as a single function-block form.

Meaningful composition is feasible for both the function-block version of a model and the CSP version. Function-block composition may only result in an acyclic graph. Composing such “strongly-interacting” sub-models, where attempting to do so at the function-block level it would appear that a variable is both an input and an output, may only be done properly at a higher level, by composing CSP forms or by additional modeling constraints provided by the modeler.

7. OMDOC AND PHYSML

Developing a physics markup language is an attempt to document certain types of abstract knowledge. As such, we must consider not only the particular elements of physics-like knowledge but we must also consider the framework in which such knowledge is embedded. We must, for example, have the concept of a *document*. The concept of a document is essential for encapsulating particulars regarding the intellectual product contained within. For example, identifying the subject matter, the author, the date produced, etc., are important. An important feature of much of the effective knowledge in physics is its mathematical nature, therefore having a means of expressing the mathematical semantics of the knowledge is very important. After surveying the literature we identified OMDoc as having many desirable attributes to further our objectives. OMDoc is open and extensible. OMDoc uses OpenMath and Content-MathML for expressing mathematical formulae and also provides other modules, which support, for example, axiom, theorem and proof structures. These shall prove useful for documentation of the formal aspects of physics knowledge.

From the OMDoc documentation[2], OMDoc is a markup format and data model for Open Mathematical Documents. It serves as semantics-oriented representation format and ontology language for mathematical knowledge. OMDoc concentrates on representing the meaning of mathematical formulae instead of their appearance. OMDoc is an extension of the OpenMath and (content) MathML standards. It extends these formats by markup for the document and theory level of mathematical documents, so that the document author can specify them and the consumer (an OMDoc reader or a mathematical software system) can take advantage of them. The value of having a document marked in this way should provide at least two benefits: a means of providing unambiguous documentation, valuable for validation, and; a form that is readily interpreted by computer for testing by model execution.

OMDoc is an XML application, organized into modules. While it includes support for expressing mathematical objects, abstract data types, and proofs, necessary to supporting the bulk of the above discussed concepts, It also supports documentation tagging, such as provided by Dublin Core[13] and Creative Commons[14] metadata, necessary for supporting documentation of textual intellectual products.

Recently, a new collaborative project, PhysML[15], was initiated as part of the OMDoc project to create a markup language for documenting physics knowledge, not restricted to purely formal knowledge. The representation of physics-based models is only a subset of the more formal aspects of physics knowledge. Since this and prior work share some of the same goals and can benefit from joint work, we expect in the future to participate in that collaboration. In particular, we are embarking on the creation of content dictionaries to capture the physical semantics, including the concepts expressed in this paper.

8. CONCLUSIONS

We have attempted to identify as abstractly as possible the form that the physics-based modeling process takes. Our purpose in doing this is to identify the concepts, particularly the mathematical concepts, that may be represented using existing and emerging mathematical content representation formats. The purpose for doing this is to provide a mechanism for unambiguous documentation of the physics-based models that lie at the core of the modeling and simulation of many systems. While the focus of this work is on the representation of physical semantics, we have tried, principally by relating it back to the object model and synthesizing with other systems specification work[], to phrase it in such a way as to be a semantic layer that is compatible with other types of information necessary for documenting the design and analysis of systems.

References

[1] World Wide Web Consortium (W3C), "Extensible Markup Language (XML)" http://www.w3.org/XML/

[2] M. Kohlhase, "OMDoc: An Open Markup Format for Mathematical Documents (Version 1.2)", number 4180 in Lecture Notes in Artificial Intelligence (LNAI), Springer Verlag, 2006

[3] F. Cajori, "Sir Isaac Newton's Mathematical Principles of Natural Philosophy And His System Of The World" University of California Press (1947).

[4] A. Einstein, "The Meaning of Relativity", Princeton University Press, expanded edition (2005).

[5] A. Einstein, "Zur Elektrodynamik bewegter Körper", ("On the Electrodynamics of Moving Bodies"), Annalen der Physik vol XVII, 1905 p. 891-921,

[6] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, "Object-Oriented Modeling and Design", Prentice Hall, (1991).

[7] J. B. Collins, "A Mathematical type for Physical Variables", submitted, Mathematical Knowledge Management 2008, for publication in the Lecture Notes in Artificial Intelligence (LNAI) series, Springer-Verlag.

[8] J. B. Collins, and D. Clark, "Towards an Ontology of Physics", Proceedings of the European Simulation Interoperability Workshop, July 2004, 04E-SIW-044.

[9] J. B. Collins, "Standardizing an Ontology of Physics for Modeling and Simulation", Proceedings of the Fall Simulation Interoperability Workshop, September 2004 04F-SIW-096.

[10] M.W. Wilson, R.S. Peak, R.E. Fulton, "Enhancing Engineering Design and Analysis Interoperability - Part 1: Constrained Objects", First MIT Conference Computational Fluid and Structural Mechanics (CFSM), Boston (June, 2001).

[11] R.S. Peak, and M.W. Wilson, "Enhancing Engineering Design and Analysis Interoperability - Part 2: A High Diversity Example", First MIT Conference Computational Fluid and Structural Mechanics (CFSM), Boston (June, 2001).

[12] E. Tsang, "Foundations of Constraint Satisfaction", Academic Press (1995)

[13] http://dublincore.org/
[
[14] http://creativecommons.org/

[15] E. R. Hilf, M. Kohlhase and H. Stamerjohanns, "Capturing the Content of Physics: Systems, Observables, and Experiments" MKM 2006, Lecture Notes in Computer Science, Vol 4108 pp165-178, Springer (2006)

Biography

Joseph Collins has been a Research Physicist at the Naval Research Laboratory for eighteen years, working largely on modeling and simulation of physical systems.