On enabling a model-based systems engineering discipline

Peter Denno
National Institute of Standards and Technology
Gaithersburg, Maryland, USA
peter.denno@nist.gov

Thomas Thurman and
John Mettenburg
Rockwell Collins, Inc.
Cedar Rapids, IA, USA

Dwayne Hardy
American Systems
Chantilley, VA, USA

Published and used by INCOSE with permission.

Abstract. This paper considers the requirements of a model-based systems engineering (MBSE) discipline and the benefits that would be realized from it. A premise of MBSE is that the technical environment supporting systems engineering has evolved, and is still evolving. We analyze the basis of systems engineering decision making in conjunction with the technical environment in which it may soon be performed. The analysis provides insight into the requirements that, when met, enable a model-based systems engineering discipline. We report on our practical experience toward meeting the requirements using UML, SysML, AADL, AP233, and a systems engineering tool interoperability testbed.

Introduction

This paper describes requirements to enable a MBSE discipline. The use of "model" in the term "model-based systems engineering" may appear vacuous since models have always played a large role in engineering and systems engineering specifically. However, it is not the use of models that distinguishes MBSE but rather *how* models may be used. In MBSE, the models must enable a family of more sophisticated tools, and thereby, new processes. Among these tools are parametric modeling and decision-support technology. These two technologies could play a larger role in future MBSE practices – whatever challenge the systems engineer might face, these illuminate the space of potential solutions. And it is the ability to understand and navigate this space that is the essence of systems engineering. Focus on agile application of these technologies is also consistent with the future described by the INCOSE 2020 Vision (Sage, 2006).

Yet, in practice, the use of parametric modeling and decision-support technology is throttled by the high cost of transforming information to a form that these tools can use. MBSE techniques can lower this cost. Models, and the formal description that they provide, can be paired with information mapping engines to lower the cost of information transformation. Further, the

integrity of models can be verified by tools mechanizing the constraints on usage found in the formal description.

Enabling agile use of parametric models and decision-support technology places certain requirements on the models used. We call these the requirements of MBSE. They are the distinguishing characteristics of MBSE as we define the term.

1. Formal specifications of viewpoint are employed.
2. Specification of the correspondence of information across viewpoints is possible.
3. A characterization of the validity of actual instances of this correspondence is possible.
4. Traceability is a mechanistic outcome of the design process.

This paper argues that a new, model-based discipline of systems engineering will emerge as the four characteristics are realized. Progress toward an MBSE discipline will remain evolutionary. The characteristics are interrelated: partial achievement of (1) and (2) above enable solution to (3) and (4). (1) and (2) can be considered near-term goals; (3) and (4) long-term goals.

On an infrastructure realizing these characteristics, it will be possible to provide the systems engineer with a stratum of sophisticated tools, including agile parametric design technology and evidence-based decision support. These tools will provide the principal benefits of MBSE.

The remainder of this paper elaborates these points and reports on our experience. Section 2 of the paper describes the systems engineering (SE) problem space in terms of its technical environment and the basis of its decision making. This leads to an understanding of the requirements that should direct work toward a MBSE infrastructure. Section 3 discusses each of the challenges in turn and describes the role that various technologies play in a MBSE infrastructure. Section 4 concludes the paper.

Although this paper reports on our experience with elements that may lead to MBSE, it is important to keep in mind that this paper is about *requirements* of MBSE. Our work may illuminate a path to the solution, but it is not the solution.

The SE Problem Space – technology perspective

For the purpose of this paper, we define systems engineering broadly, and without regard to whether it is "model-based." It is any methodical approach to the synthesis of an entire system that (1) defines views of that system that help elicit and elaborate requirements, and (2) manages the relationship of requirements to performance measures, design constraints, system components, and discipline-specific system views. The technical environment supporting systems engineering includes a design process, product data management (PDM) systems, discipline-specific analytical tools, modeling tools, (for (UML 2007) and (SysML 2006), for example)[1] engineering ontologies and their schemas (*e.g.* SysML, (MARTE, 2005), (AP233, 2004) information mapping engines (*e.g.* (QVT, 2005), ATL (Jouault, 2006), (Express-X, 2003)), catalogs of parts and their characteristics, and authoritative references.

The variety of engineering ontologies used in a typical SE environment and differences in the

[1] An index of the abbreviations and acronyms used is provided in section *Acronyms*.

technologies employed in expressing them present technical challenges that will be discussed in this paper. Figure 1, adapted from Sudarsan (Sudarsan, 2006), illustrates the scope of various standards relevant to systems engineering and the product lifecycle. The vertical axis represents the specificity of scope, *e.g.,* UML and OWL have wide applicability, so they are low on the vertical axis; OAGIS BODs are specifically for messaging so it is high.). The height of the ovals has no meaning.

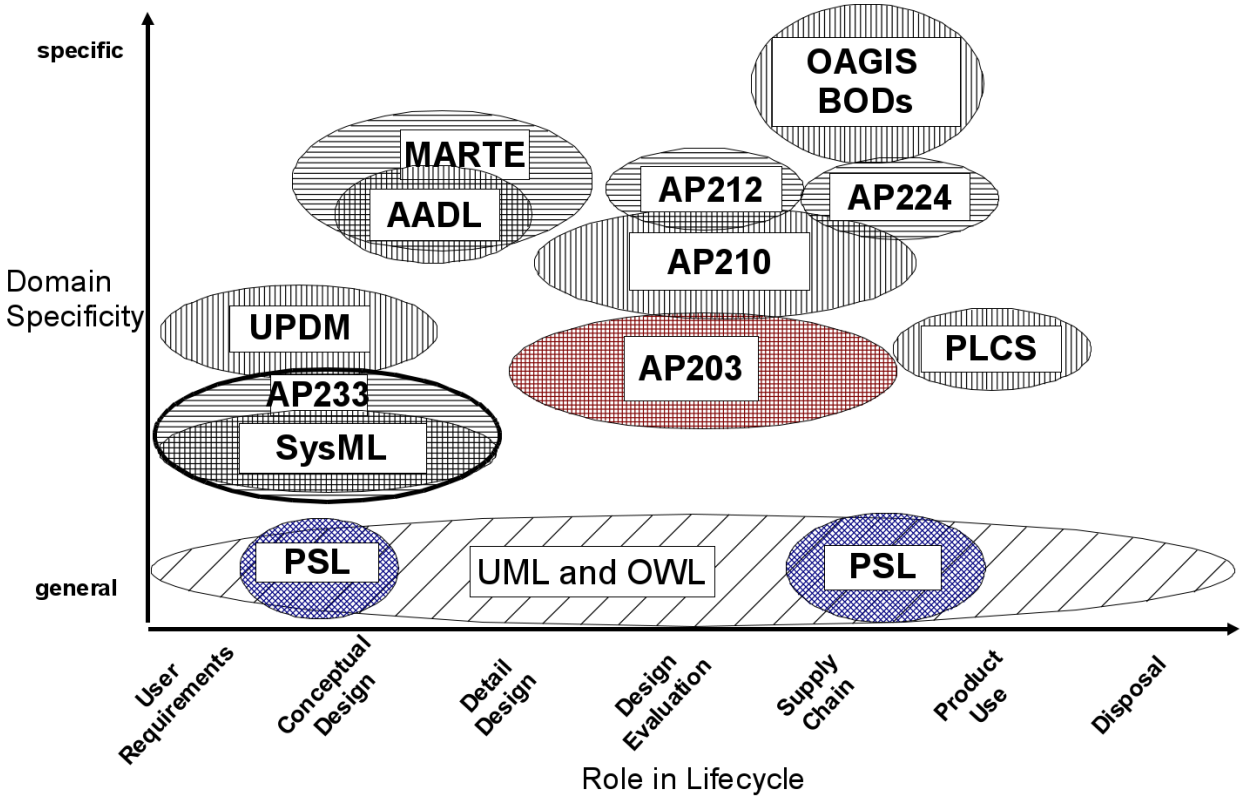


Figure 1. Lifecycle role of various standards, specificity of domain

Basis of SE decision making: In the following, adapted from an earlier work, (Denno and Thurman, 2005) we analyze the basis of systems engineering decision making. This is done mindful of the technical environment in which decisions are made. In the process, we identify the differing notions of traceability that are employed in systems engineering.

The constituents of the basis are

- **Change process:** Knowledge of the existence of precursors and the history of the properties that distinguish them.
- **Origin in requirements:** Relationships that indicate how the data are related to requirements (such as design rationale and trace relationships). This is the sense of traceability usually intended when the term is used by systems engineers.

- **V&V process:** Knowledge of the deliberate steps taken to ensure that data is consistent with other beliefs and with requirements. Typically these "steps" are defined in the design guidelines of the enterprise, in regulations, and in quality standards.
- **Origin in other belief:** Relationships that indicate the logical basis for the belief and the history of its logical support. Examples include correspondence of the data to an idealized model, or an entailment from that model made available from a validation process (but not the process itself, which is the subject of the previous constituent). The logical support of a belief may change over time. The assumptions on which it rests may no longer be true, or may be found to have been false all along. This is a logical sense of traceability.
- **Authority:** The power that data has, due to an approval it is granted or an estimate of its maturity. Approvals and estimates of maturity ultimately are underwritten by a role in the organization. Lesser notions of authority are applied when, for example, a decision is supported by component characteristics from a supplier.
- **Origin in media:** Relationships that indicate where the data reside. ``Media'' might not be as concrete as "on the disk of computer xyz" but rather "in the CAD repository of designs approved for manufacture." This is a logistical sense of traceability. It is closely related to the constituent *authority* above.
- **Logical consistency:** This includes type awareness, interpretation constraints, and well-formedness conditions. Type awareness concerns knowledge of the type of thing to which the data refer. Type reference data may include units of measure. Interpretation constraints are assertions about what roles a particular type can serve. For example, a "thrust-specific fuel consumption" value can only refer to an object that consumes fuel and produces thrust. Well-formedness conditions constrain the structure of the data to be consistent with the system of expression (syntax, domain of values) in which it is defined.
- **Measurement Conditions:** This is knowledge of the process by which a datum was measured or computed, an expression of confidence in the value, or a statement of accuracy.
- **Associativity across views:** Knowledge of relationships among properties conceived in differing viewpoints. There are two forms of ignorance of these relationships
  - A *conceptualization gap* is the absence of the knowledge that two conceptualizations can be used for the same purpose. For example, a time period could be specified by either a start and stop time-point, or a start time-point and a duration. Bridging a conceptualization gap (which can often be done *a priori*) may make existing data governed by the schemas more useful, without explicitly referencing any of the data.
  - A *recognition gap* is the absence of the knowledge that two references refer to the same thing. For example, that a geometric feature in a tolerance stack-up analysis is the same feature in a CAD model. Bridging a recognition gaps makes available additional information about the things being referenced.

The notion of conceptualization gap emphasizes an analytic and intensional viewpoint on associativity. Recognition gap emphasizes a synthetic and extensional viewpoint. Collectively they are referred to as *associativity gaps* (Peak, 2003). Associativity gaps are accidents of nature when they occur across disparate views. They are modeling errors when they occur within a single view.

When a conceptualization gap concerning *identity conditions* (Guarino and Welty, 2001) is resolved (by equating identity conditions across viewpoints), it may be possible to resolve recognition gaps among the individuals identified by those conditions.

A *proposition* is a complete though that is expressed by an indicative sentence. It is whatever can be asserted, denied, contended, maintained, assumed, supported, implied, or presupposed. The same proposition may be expressed by different sentences. (Johansson, 2007) Any proposition relevant to the system under study may be qualified by, or related to, other propositions about the system that fit the nine categories above. For example, if for some hypothetical property P, the proposition "The value of P is 0.05" is asserted, the following propositions might also apply:

- "Our ability to achieve requirement *x* diminishes as P exceeds 0.07." (origin in requirements)
- "The value of P we calculated for this design is close to that found in earlier designs." (change process)
- "The value of P is confirmed through an engineering simulation that is routinely performed in the validation of this product line." (V&V process)
- "A value of 0.05 when used in equation Q, provides a value of R that is consistent with observations." (origin in other belief)
- "Supplier provided characteristics also suggest that P=0.05 is obtainable." (authority)
- "This value of P was obtained from the aero model in the preliminary design library." (origin in media)
- "The units of P are m/sec, which are valid units of measure for observations of P." (logical consistency)
- "This value of P was observed under 1 ATM pressure, 20% humidity and 20 degrees Celsius." (measurement conditions)
- "If P=0.05 in the static analysis, it can be expected that P=0.055 will be found in the dynamic analysis." (associativity across views)

The nine constituents above overlap and interrelate in complex ways. For example, the constituent *measurement conditions* includes an assessment of ``the degree of confidence."
Each of the eight other constituents also contributes to an assessment of a degree of confidence, however, these other constituents interrelate propositions. *Measurement conditions* only concern the proposition asserting the measured value, and the degree of confidence expressed is abstracted to an elementary measure (such as an ordering) of an undifferentiated notion of confidence.

The Roles of Models

Achieving the aim of the four challenges listed in the introduction provides two positive results. First, it becomes possible to communicate essential SE information to the decision support and parametric design tools that can automate SE processes. Second, the investment in models with these four properties makes clear the relationships inherent in the information and processes employed. This may have far-reaching consequences in the ability to manage the enterprise and its projects. Models provide an enduring repository for institutional knowledge. Models can be improved iteratively. They seldom become wholly obsolete.

Systems engineering currently makes little use of evidence-based decision-support tools. It is difficult to integrate them into a process that lacks a methodical means to draw information from multiple views. The need to do so is critical to decision support.

Systems engineering makes better use of parametric design tools, and the value of such tools is well known to most systems engineers. But again, the difficulty of obtaining information that would drive the process is an obstacle. Without the benefits of a MBSE environment, it will remain necessary to (1) intervene manually to resolve associations across views, and (2) translate analytic results to implications on requirements. This situation prevails despite the fact that technical frameworks for more agile application of parametric design already exist (Peak, 2000).

The critical difference that has made the use of parametric design technology a mixed success, and the application of decision-support tools largely a failure, is that the former can be applied in routine design scenarios where the cost of implementation can be amortized over the long period in which those design scenarios persist. In contrast, decision support is used in more *ad hoc* situations and toward more elementary decisions. While this ought to be an advantage, the setup cost makes it infeasible.

A meta-object framework is a key enabler of a MBSE environment. Figure 2 illustrates relationships among elements of a meta-object framework. *N.B.*: It depicts an abstract conceptualization, not an architecture diagram. The figure depicts elements from two disparate technologies, UML and STEP. The inclusion of two or more technologies not sharing a common metamodel may be typical of actual implementations.

The technologies are integrated through the use of a Meta Object Framework (MOF) based metamodel for EXPRESS-based data. This metamodel is being developed as part of the MOF2 EXPRESS Interoperability and Coexistence (MEXICO) project (Barkmeyer, 2007) (Krause & Kaufmann, 2007). Using the "MEXICO Injector" an MOF-based translation of an EXPRESS schema (in this case for AP233) is produced. This schema conforms to the EXPRESS metamodel. The schema is used as source objects in a mapping of EXPRESS to UML, (producing the "UML-based AP233 Schema" depicted). The UML-based AP233 schema is used to map AP233 objects to a repository in a manner similar to how SysML or MARTE objects would be mapped. A similar procedure could be used to map other STEP application protocols that might be relevant to the domain. One such protocol is AP210, the electro-mechanical design application protocol.

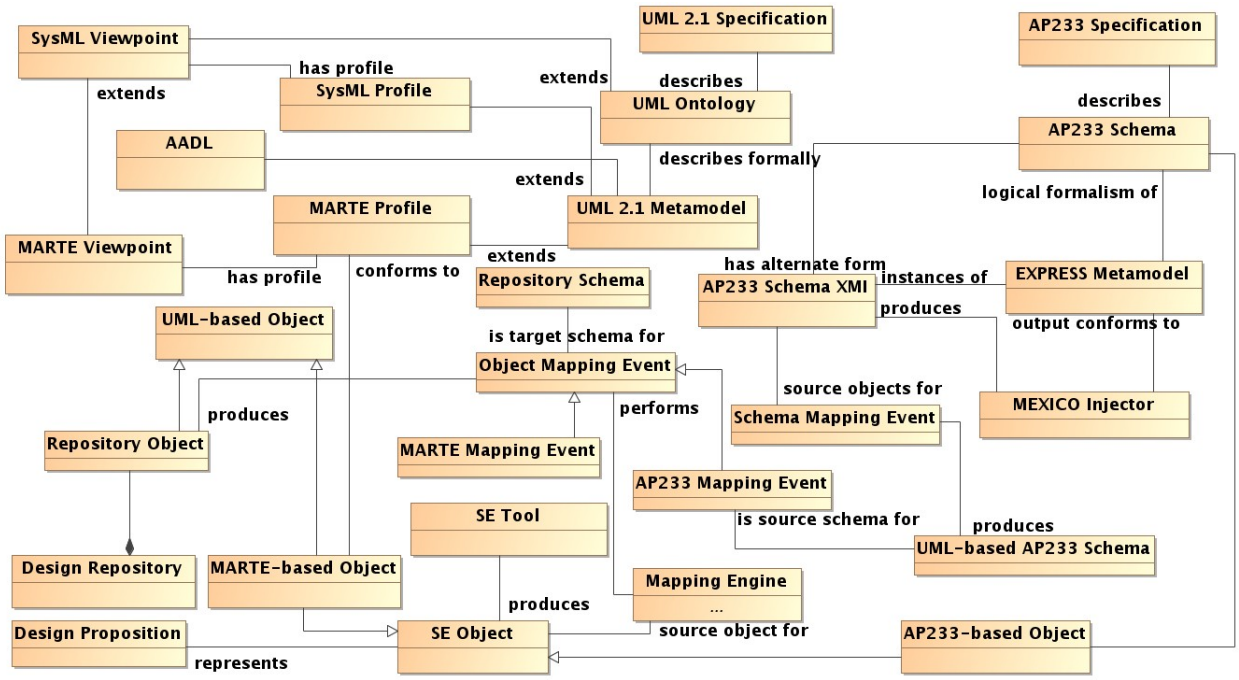


Figure 2. Components of a MBSE Architecture

The next section describes the four challenges from the Introduction in turn, with reference to the decision-making constituents, the metaobject framework, and their implication on enabling the technologies.

Challenge 1: Formal Specification of Viewpoint

A viewpoint is expressed as the selection of concepts, relationships, and constraints on interpretation identified in some universe of discourse. A viewpoint may be expressed in a technology such as the UML, EXPRESS, and OWL (W3C, 2004). In order to understand the value that formality brings to the specification of viewpoints, it is useful to view these technologies as *knowledge representation languages* (KRLs). Fikes (Fikes, 2004) defines a KRL as consisting of a logical formalism, an ontology, and a proof theory. By reference to proof theory, this definition might seem ill-fit to UML, and EXPRESS,[2] which were not conceived having one. Yet, some aspects of proof theory apply to these languages and, as described below, provides the principal benefit of the formalization of the language.

Proof theory (Buss, 1998) seeks to characterize mathematically the soundness and completeness of the rules of inference of a logical formalism. Typically, proof theories may be mechanized, thereby providing a tool implementing the inference rules of the logical formalism. Resolution-based reasoners (Robinson, 2001) exemplify tools in which a proof theory (of first-order logic in this case) describes a means of mechanization. The Process Specification Language (PSL, 2005) exemplifies an ontology germane to SE that is specified in a logical formalism that has a proof theory.

[2] EXPRESS is the information modeling language ("logical formalism") of several application protocols ("ontologies") including AP233.

Mechanization of proof theories provides two principal benefits. First, it provides a software tool to check the consistency of the theory (by identifying contradictions). Secondly, the same tool can be used to derive a proof of any statement that follows from the theory. (This is true of logical formalisms that are said to be "complete" – a property typically demonstrated through the proof theory). "The theory" involved in the consistency checking and proof above is, to return to the definition of Fikes, the ontology. An *ontology* in the framework of Fikes consists of

1. a set of non-logical symbols defined or restricted
2. definitions of non-logical, non-primitive symbols
3. a set of axioms restricting the interpretation of primitive non-logical symbols

In (1) and (2) above, non-logical symbols are distinguished from logical symbols in that the former are elements of the vocabulary of the universe of discourse and the latter are elements of the logical formalism. This is analogous to the difference between the name of a class in a UML class diagram (non-logical) and the notation for generalization in UML (logical). In (2) above, non-primitive symbols are distinguished from primitive symbols in that necessary and sufficient conditions are provided for the former (they are "defined") but not for the latter. In (3) "axioms" refers to statements in the ontology that express a truth concerning a concept represented by a primitive symbol, but do so with less than necessary and sufficient conditions.

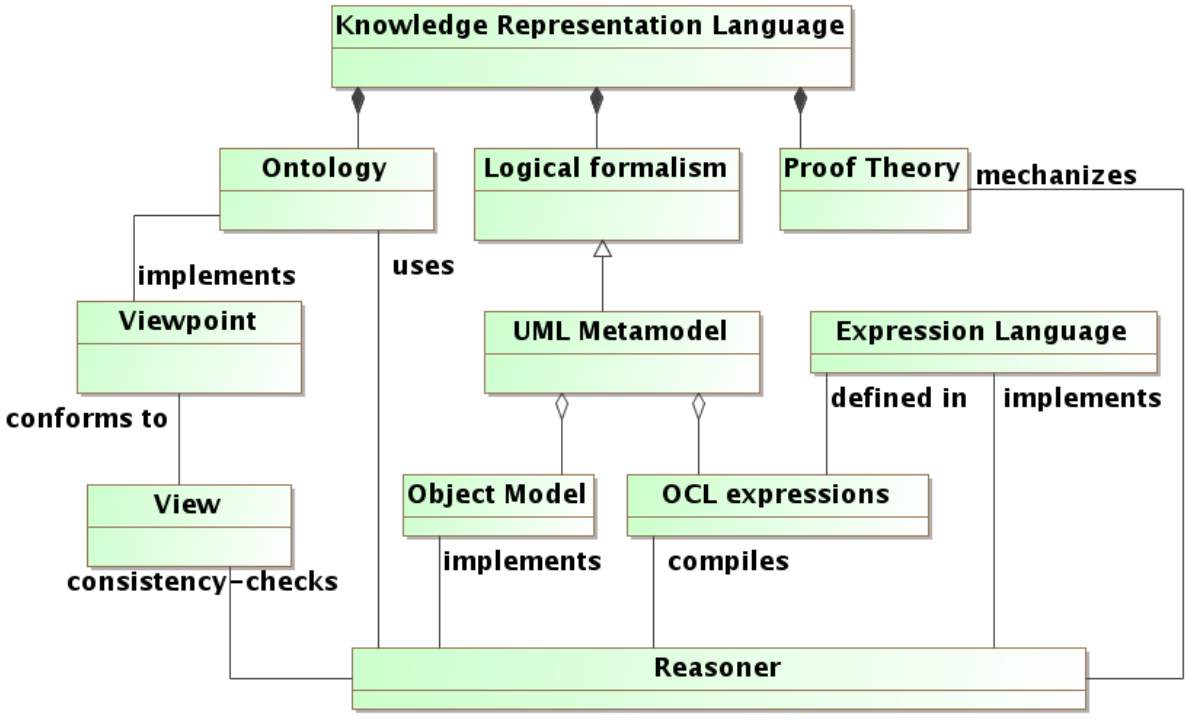
To summarize the above in the context of expressing a viewpoint by means of a KRL, a proof theory of a logical formalism provides tools to test the consistency of the ontology and, for semantically complete proof theories, to provide proofs of any true statement about the ontology. Further, the distinction between definitions and axioms provides an epistemic property of statements made in the ontology (an aspect of the constituent *origin in other belief*). In fact, there is another distinction that can be made here – that between definitions that introduce new primitives and those that do not. (The latter are called *conservative* definitions.)

Mechanized proof theories theoretically provide the capability to test the validity of information intended to conform to an ontology. Such information may be provided by an exchange file from a tool that operates in the context of the viewpoint. In earlier, but incomplete work (Denno, 2006), we implemented this approach using a first-order logic (FOL) reasoner, Vampire (Raizanov, 2003) and an ontology for the management of intercontinental supply chain transport. The ontology was defined as an extension of the Suggested Upper Merged Ontology (SUMO) (Niles, 2001) and information from exchange files (ebXML (OASIS, 2007) messages) was extracted to ground facts. In a FOL KRL, it is not necessary (nor easy) to make a distinction between the information originating from the ontology and that originating from the exchange file – it is all uniformly represented in the logical formalism of the KRL.

A significant advantage of this FOL approach is that it is not necessary to define specific tests of consistency – it is enough to query the reasoner for a proof of *any* statement known to be false with respect to the ontology. If the combined ontology plus exchange file content is inconsistent, the proof returned will make use of a contradiction in those statements. A disadvantage of this approach, however, is that it can be difficult to automate the process that determines *what* inconsistency in the ontology and exchange file enabled the proof.

Important differences exist between KRL technology possessing a proof theory and technologies

such as UML, EXPESSS, SysML, and AP233. However, these latter technologies are specified sufficiently to enable mechanized *conformance checking tools*, that assess whether populations conform to their respective ontologies. For example, a SysML conformance-checking tool may confirm that a population obeys the stipulation that an "ItemFlow" must be associated with a "Connector" or "Association." Unlike first-order logic, where a formal proof theory specifies a mechanization of the rules of inference, the implementation of tools for these languages is guided largely by an understanding of its object model and expression language, which together are used to specify an ontology. An *object model* concerns the notions of type composition, inheritance, properties, relationships, and mechanisms for abstraction of an object-oriented formalism. An *expression language* is a subset of the logical formalism consisting of operators composable to expressions that can be used to further constrain the ontology.[3] Implementing the object model and an evaluation engine for the expression language enables the generation of conformance tools. They can be created by reading the metamodel of the target ontologies and generating from this (1) class definitions conforming to the implemented object model (the classes are instances of the object model) and, (2) a translation of constraints provided in the metamodel to executable code that runs against the evaluation engine. We did exactly this to generate conformance tools for UML and SysML, using the UML metamodel and SysML Profile, (Denno, 2007) and for AP233, using MEXICO and the AP233 schema. Figure 3, illustrates the relationships among concepts involved.



[3]An object model for SysML is, roughly speaking, provided by MOF. The expression language of SysML is the Object Constraint Language (OCL). MEXICO provides a metamodel of EXPRESS (the logical formalism of AP233) as a UML model. The MEXICO metamodel covers both the object model and expression language of EXPRESS.

Figure 3. KRLs, proof theory and consistency checking

Further discussion of the implementation of conformance-checking tools is provided in our earlier work (Denno, 1996).

Conformance and interoperability validation tools: Conformance-checking tools serve the important role of identifying where software that exchanges information by means of a shared interface specification (an exchange file specification) does not conform to the normative stipulations of that specification. Conformance to a shared exchange specification is a key enabler of *interoperability*, the ability of parties to work jointly toward a shared goal. In the case of SysML and AP233, at least, the great majority of these stipulations are the constraints of the ontology, and a small minority concern the serialization of the content in the exchange file (*e.g.* XMI (XMI, 2005) for SysML, and ISO 10303-28 (10303-28, 2006) for AP233).

We have developed a testbed that can be used to assess the interoperability of systems engineering tools. (Denno, 2007) The near-term goal of the testbed is to help SE tool developers make this assessment using its conformance-checking tools. The testbed provides conformance-checking tools for information provided in AP233, UML, and SysML exchange files.

Because exchange files for UML, SysML and AP233 can encompass the complete ontology of their respective viewpoints, the conformance-checking tools can also serve the purpose of identifying errors and shortcomings in the specifications themselves. In this regard, the testbed tools have identified many errors in the OCL of UML, and a few situations where SysML would benefit by the specification of additional constraints.

The mid-term goal of the interoperability testbed is to provide an environment for tool developers to assess the ability of tools to share information in situations where those tools implement a common exchange file specification (*e.g.* they all implement XMI for SysML exchange). At the time of this writing, that work, called "The SE Tool Interoperability Plugfest," is just starting.

The long-term goal of the testbed is to provide a facility to improve the ability of SE tools to share information across closely related viewpoints. For example, requirements encoded in the SysML viewpoint could be shared with those encoded in AP233.

To summarize the discussion of the role of formality in specification, formality enables automated means to assess the logical consistency of expression. When the ontology is stated in a logical formalism for which there is a proof theory, it may be possible to check the consistency of the ontology itself. As the viewpoints that SE standards developers seek to express become increasingly complex, and as these viewpoints become grounded on a mathematical foundation, the value of consistency checking will become increasingly obvious.[4] Yet we cannot rule out the possibility that for the foreseeable future, technologies such as UML and OCL will adequately serve the requirements of MBSE. In cases where a proof theory of the logical formalism is not

[4]This trend in the development of consensus specifications might be called ``model-based standards development."

known, formality might still enable the development of conformance-checking tools.

Relation to decision-making constituents: Formality in specification strongly relates to the constituent *logical consistency*. Type awareness, interpretation constraints, and well-formedness conditions are precisely what formality provides. The value of formality in *associativity across views* is discussed in section *associativity,* below. Whereas the above two constituents have a general information-technology flavor, the remaining seven constituents are more specifically SE domain concerns. The benefit of formal specifications is not limited to engineering processes. The enterprise benefits in areas of program/project management, training, service, manufacturing, and certification as its processes and viewpoints become more widely understood and integrated.

Challenge 2: Associativity across viewpoints

Engineering enterprises are comprised of collaborating units, typically departments, each working under the assumptions and viewpoints appropriate to their discipline-specific tasks. Knowledge of how information from these various viewpoints interrelates is essential to many system engineering tasks. This is apparent when one considers how much of systems engineering concerns the decomposition of information from requirements and the reintegration of components to a complete system.

In consideration of the technical environments in which the work of the discipline and of systems engineering is performed, there are three principal means by which associativity across viewpoints is acquired

1. through interrelations inherent in a schema [5] that encompasses multiple viewpoints
2. through explicit mappings that reference objects in disparate ontologies
3. through skilled, domain knowledge-intensive, consideration of the data in the context of information that affects its interpretation

Sources of associativity (1) and (2) are discussed in this section, and (3) in section *Extent of validity*

A distinguishing characteristic of MBSE when compared to traditional processes is that, in MBSE, viewpoints are aggregated and interrelated under a schema. In a multi-viewpoint framework, tools that manipulate the viewpoints bear much of the responsibility for reusing and interrelating objects. A differentiating factor among these schemas is the degree of inherent cohesion among the viewpoints that they aggregate. Rather "generic" integration schema such as GEIA-927 (GEIA, 2006) and the EPISTLE core model (EPISTLE, 2003) can correlate data through basic ontological primitives such as class relationships and process decomposition. Schemas that integrate a more closely related collection of viewpoints, such as SysML and AP233, additionally track the identity of objects across views. For example, a hydraulic pump in a structural view has a port that has properties attached that are referenced in a parametrics view.

[5] The word *schema* (and not *ontology*) is used here to distinguish the technical aggregation of the content from its conceptual scope. A schema in this context is a collection of viewpoints under a common system of expression. AP233 and SysML are representative – they contain viewpoints for requirements, system breakdown, *etc.*

Extended provisions for cohesion are provided in schemas such as AP210 (AP210, 2001) and AADL that, because their scope is more focused, may include more domain-specific facilities for integration. The AADL notion of sub-program exemplifies this distinction: AADL has provisions to describe precisely the relationships among sub-programs, processes, and processors. In SysML, this relationship might be abstracted to associations among blocks. In SysML, no language provisions anticipate the notion of sub-program.

It is conceivable in a tightly integrated collection of viewpoints that the "roll up" of allocated properties (*e.g.* weight, power use) could be expressed as a concept in the overarching ontology. Likewise, behavioral properties that in some way "subsume" other system properties could be expressed. An example of subsumption in this sense is a throughput property of a system consisting of components in serial connection – the throughput of the system is the throughput of the slowest component.

By *explicit mappings* in (2) above, we mean that associativity across disparate views is identified in *mapping declarations* defined in a structural mapping language such as ATL, QVT or Express-X. Mapping languages such as these are a distinguishing characteristic of MBSE, enabled by the use of a shared metamodel architecture. Traditional methods of translator development require implementation of serialization functions (potentially reading and writing very different serializations) as well as structure mapping functions. The use of mapping languages has isolated the mapping function into the implementation of the mapping engine. The separate serialization function is dependent only on a meta-meta-model, such as MOF, which the source and target models share. Though the efficiency gained through this is obvious, perhaps more important is the value declarative mappings provide to the enterprise – with these, it is possible to study the conceptual mapping without viewing irrelevant implementation details.

Our experience suggests, however, that mapping engine implementations are relatively immature. Performance on very large datasets is not good and provisions for incremental mapping (*i.e.* mapping a increment of source data into an existing body of target data) are not well established.

Challenge 3: Extent of validity across viewpoints

Requirement (3) from the introduction is a long-term goal of MBSE. It enables the expression of the nature of, and confidence in, the relationship between properties expressed in disparate viewpoints.

Where well-understood relationships across viewpoints exists these relationships can be aggregated into a multi-viewpoint integration schema,[6] removing conceptualization gaps. This was noted above. When, instead, an enterprise uses multiple disparate ontologies, similar results can be achieved using mapping schema. This was also discussed above. In both of these cases, conceptualization gaps are bridged as analytic judgments (*i.e.* a deductive process). What remains is the recognition of relationships that cannot be resolved *a priori* because either (1) some properties involved in the (synthetic) judgment are in a "gray area" where their interpretation is unclear, or (2) the ontologies which they span are pair-wise inconsistent or

[6]UML, SysML, AP210, AP233, AADL and MARTE could all be described as "multi-viewpoint integration schema" in that they assert relations across the viewpoints they aggregate.

incomplete.[7]

Having applied mapping and multi-viewpoint integration to all concerns that can be addressed *a priori,* what remains is an area requiring human engineering interpretation. There will be, for the foreseeable future, situations where the conceptualization of one engineering discipline cannot be restated easily in the conceptualizations of another. In these situations, an interpretation of one discipline's data might only serve as *evidence* of some property defined from the viewpoint of the receiving discipline.

We have said that the development of MBSE is an evolutionary process and that the partial achievement of the requirements for formal specification of viewpoint, and correspondence across viewpoints (requirements (1) and (2) from the Introduction) enable characterization of the validity of instances of the correspondence (requirement (3)). In this regard, decision support technology is both enabling MBSE and its goal. It is enabling because the rules provided to decision support systems. For example, Bayesian knowledge bases, (Laskey, 2007) provide a characterization of validity across viewpoints. Indeed, if the validity could be established deductively there would be no point in using this technology. The relationship to the goal only appears convoluted when the role of characterizing validity across viewpoints is conflated with the larger role of providing decision support to the systems engineer.[8]

Probabilistic techniques such as (Laskey, 2007) can be used to express confidence in a relationship that spans viewpoints. More explicit expression of the relation may be possible, but supporting technology in this area is less mature. A KRL that appear to possess the expressive power to relate propositions across multiple viewpoints is the IKRIS Knowledge Language (IKL) (Hayes, 2006, Hayes, 2007). However, as of this writing, there are no reasoners for IKL.

Challenge 4: Traceability is a mechanistic outcome

The fourth challenge enumerated in the Introduction is to ensure that the design system has, built-in, the ability to describe the motivation of the design commitments or "refinements" that lead to the current design. This should be viewed as a long-term goal. The near-term benefits of MBSE can be realized without this ability. The work toward this goal is generally immature and exploratory.

A *refinement* is, roughly speaking, a commitment made along the path from a requirements specification to a system specification satisfying those requirements. In a sense, a refinement is like a step in a proof. However, it is unlikely that automated reasoners of the sort described in section *Formal Specification* could be employed effectively to perform design (*i.e.* "advance" a design by making sound inferences that are refinements). Among the difficulties to such an approach are the lack of goal-directness of the reasoner and the weakness of the rules of inference as a means of design ideation. As an improvement over these general-purpose tools, term rewriting has been applied as a means to automate design. (Zave, 1997) Term rewriting is analogous to theorem proving where, in this analogy, the inference rules (*e.g.* resolution) are

[7] It is common that the steps of the engineering process are gated by these quintessential engineering judgments.

[8] Though it could be argued that characterizing the validity of information across viewpoints is the principal role of a systems engineer!

replaced by another set of syntactic transformation rules, the rewrite rules. The rewrite rules reflect engineering commitments, in the sense that they rewrite something that has the form of a requirement (or derived requirement) into something that is closer to a design specification.

The notion of refinement employed in term-rewriting-based design methods is quite specific. The method has the advantage of obviating some design validation work (provided the rewrite rules are sound). However, it has severe limitations. First, it requires a formal and correct specification of the requirements. Second, it is problematic in domains where the relationships between the intended function of elementary components and their behavior is not clear and deliberate. This limitation has excluded nearly all domains of application but software and some aspects of integrated circuit design. More promising for these domains may be hybrid methods that transform and compose "building blocks" rather than the simpler syntactic structures of term rewriting. Examples of this are found in the developing research area called *proof-based systems engineering (Biely, 2005)* and the ASSERT project (Morisse, 2005). It may also be possible to augment automatic reasoners with human assistance. (Kirchner, 2000)

A more promising near-term goal of MBSE is to leverage knowledge of the relationships among properties in order to document design commitment rationale. This can be viewed as a modest increment in the practice of product data management (PDM). PDM systems can keep annotations on whole designs and simulation results. In MBSE, reference can be made to the ontologies and mapping specifications where some of the information supporting the commitment is stated, and an engineering decision-support system (Ullman, 2007) can be used to record the support. Support could include reference to the various notions of traceability (*e.g.* reference to requirements, change process, and V&V process) as noted in the discussion of the constituents.

This same approach can be extended to leverage knowledge of mathematical relationships of design parameters to behavior. The engineering decision-support system could be used to record the sensitivity of behaviors to perturbation of design parameters. This information is perhaps already known through engineering simulations and parametric design studies, but repeating it in an engineering decision-support system in a MBSE environment may expose it for more general use across the enterprise.

Conclusion

Model-based Systems Engineering is possible in an infrastructure that supports formal specification of viewpoints, rigorous specification of the correspondence of information across viewpoints, knowledge of the extent of validity of this correspondence, and built-in traceability. On an infrastructure possessing these properties, it will be possible to implement agile parametric design technology and evidence-based decision support systems. These are the principal benefits realized by MBSE.

Formal specification of viewpoint enables automated consistency checking of ontologies and tools for conformance checking of standards-based exchange forms. The importance of the former increases as the SE discipline strives to codify its practices. The latter is essential to tool interoperability. The development of the SE tool interoperability testbed is a means toward

achieving these goals.

Formality of specification also leads to better means to express relations among information in related viewpoints. Expression of this information has "cultural" benefits to the enterprise, as details of the relationships among disciplines are revealed. By enabling structural information mapping, the metamodel architecture also enables more agile application of parametric design technology: knowledge of the mapping can be used to transform data into forms needed by analytical tools, and to transform results from these tools back to statements about requirements and design feasibility constraints.

Assessing the extent of validity of the correspondence of properties across viewpoints is a quintessential systems engineering task. This is an area where the investment in formal ontologies and careful modeling returns the most value. The near-term goal, to codify this knowledge for manual analysis, is in itself a significant challenge. The long-term goal is to make the knowledge available to evidence-based decision-support tools.

Finally, by means of an analysis of the basis for SE decision making, the paper illustrates some of the notions of traceability at work in SE, and sources of them. Recording, with decision-support tools, the traces and refinements made in engineering design commitments provides a repository of institutional knowledge and a platform for agile family-of-products development.[9]

Index of Acronyms and Abbreviations

AADL – Avionics Architecture Description Language, an SAE standard.
ATL – Atlas Transformation Language, a mapping language similar to QVT.
AP210 – ISO 10303-210, electromechanical design application protocol.
AP233 – ISO 10303-233, system engineering application protocol.
ebXML – Electronic Business using eXtensible Markup Language, an OASIS and UN/CEFACT standard.
Express – ISO 10303-11, information modeling language used in ISO 10303 (STEP) suite of standards.
Express-X – ISO 10303 Structural Mapping Language.
EPISTLE – ISO 15926, an integration schema
INCOSE – International Council on Systems Engineering
ISO 10303-28 – XML-based serialization for STEP-based information.
QVT – Queries, Views, Transformations, an OMG specification.
MARTE – Modeling and Analysis of Real-Time Embedded Systems
MOF – Metaobject Framework, an OMG specification.
OWL – Ontology Language for the Web, a W3C specification.
SysML – Systems Modeling Language, an OMG specification.
UML – Unified Modeling Language, an OMG specification.
V&V – validation & verification.

[9]References to proprietary products are included in this paper solely to identify the tools actually used in the industrial applications. This identification does not imply any recommendation or endorsement by NIST as to the suitability of the product for the purpose.

XMI – XML-based serialization for MOF-based information, an OMG specification.

References

AADL: SAE International: Architecture Analysis & Design Language (AADL), AS5506, November 2004, (2004)

AP210: International Organization for Standards (ISO): Application Protocol: Electronic Assembly, Interconnect and Packaging Design, ISO 10303-210:2001 (2001)
AP233, International Organization for Standards (ISO): ISO/CD 10303-233, Part 233: Systems engineering data representation, Interim Release, 2004-01-26, http://ap233.eurostep.com/ (2004)

Barkmeyer, E. J.: The EXPRESS metamodel and mapping project (aka MEXICO), Object Management Group presentation, http://www.omg.org/cgi-bin/doc?mantis/2008-02-05, (2008)

Biely, M, Le Lann, G., Ulrich S.: Proof-Based Systems Engineering Using a Virtual System Model, In: *LNCS 3694, Proceedings 2nd International Service Availability Symposium (ISAS'05), Springer*, Berlin, Germany, pp 164-179 (2005)

Buss, S. R.: "An Introduction to Proof Theory" in Handbook of Proof Theory, Samual R. Buss (ed), Elsevier, Amsterdam, (1998)

Cantrell, J. B.: SADASAE, http://www.sadasae.com/Pages/meaning.htm. (2007)

Denno, P., Thurman, T: Requirements on information technology for product lifecycle management. In: International Journal of Product Development, Vol 2, Nos. 1/2, (2005)

Denno, P., Iviezic, N.: Message Validation with a Semantic Reasoning Tool, NIST Internal Report, NISTIR 7347, (2006)

Denno, P: MOF2 / EXPRESS Integration and Coexistence (MEXICO), http://syseng.nist.gov/se-interop/mexico (2007)

Denno, P.: Dynamic Objects and Meta-level Programming of an EXPRESS Language Environment, Dynamic Objects Workshop; Object World, Boston, MA, (1996)

Denno, P.: Systems Engineering Tool Interoperability Plugfest,
http://syseng.nist.gov/se-interop/sysml-tools-overview (2007)

EPISTLE, International Organization for Standards (ISO): EPISTLE Core Model, ISO 15926-2 (2003)

EXPRESS: International Organization for Standards (ISO): The EXPRESS language reference manual, ISO 10303-11:2004 Ed. 2, (2004)

Express-X: International Organization for Standards (ISO): The Express-X Language Reference Manual, ISO 10303-14, International Standard (2003)

Fikes, R.: lecture slides on knowledge representation. "Multi-use Ontologies" for CS222 Winter 2004, Stanford University (2004)

GEIA-927: Government Electronics & Information Technology Association (GEIA): Handbook and Guide for GEIA-927 Common Data Schema for Complex Systems, GEIA-HB-927 (2006)

Guarino, N. and Welty C.: Identity and Subsumption, LADSEB-CNR Internal Report, 01/ (2001).

Hayes, P: "A logic for ontology interoperation" Ontolog Forum invited presentation, http://ontolog.cim3.net/forum/ontolog-forum/2006-10/msg00082.html, October 26, 2006.

Hayes, P. et al.: "IKL Specification Document"
http://www.ihmc.us/users/phayes/IKL/SPEC/SPEC.html

ISO 10303-28: International Organization for Standards (ISO): XML representation of EXPRESS schemas, using XML schemas, Draft International Standard, ISO/DIS 10303-28, 2006-01-19, (2006)

Jouault, F., Kurtev, I.: On the Architectural Alignment of ATL and QVT In: Proceedings of the 2006 ACM Symposium on Applied Computing (SAC 06). ACM Press, Dijon, France, chapter Model transformation, pages 1188–1195 (2006)

Kirchner, H.: Combining assisted and automated deduction, In Annals of Mathematics and Artificial Intelligence, Springer Netherlands, Volume 28, Numbers 1-4/October, 2000

Krause, F. L., Kuafmann, U.: Metamodeling for Interoperability in Product Design, CIRP Annals, Manufacturing Technolgy, Volume 56, Issue 1, 159-162, 2007.

Laskey, B.: MEBN: A Language for First-Order Bayesian Knowledge Bases, George Mason University, Department of Systems Engineering and Operations Research, 2007.

MARTE, OMG: Joint UML Profile for MARTE Initial Submission, realtime/2005-11-01 (2005)

Morisse, J., Martelli, A., David, P.: The highly reliable infrastructure system family for ASSERT, Proceedings of the DASIA 2005 - DAta Systems In Aerospace - Conference, 30 May - 2 June 2005, Edinburgh, UK (ESA SP-602, August 2005)

Niles, I., and Pease, A.: Towards a Standard Upper Ontology. In: Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001), Chris Welty and Barry Smith, eds, Ogunquit, Maine, October 17-19, 2001

OASIS: OASIS ebXML Messaging Service, http://www.oasis-open.org/committees/ (2007).

OWL: OWL Web Ontology Language Reference, W3C Recommendation 10 February 2004, http://www.w3.org/TR/owl-ref/ (2004)

Peak, R. S.: Characterizing fine-grained associativity gaps: a preliminary study of CAD-CAE model interoperability. In: **Proceedings of DETC'03, ASME Design Engineering Technical Conferences**, Chicago, Illinois, USA (September 2–6, 2003)

Peak, R. S.: X-Analysis Integration (XAI) Technology, Georgia Tech Engineering Information Systems Lab Technical Report EL002-2000A (2000)

PSL: International Organization for Standards (ISO): Process Specification Language – Part 11: PSL Core, ISO 18629-11 (2005)

QVT: MOF QVT Final Adopted Specification, ptc/05-11-01.

Raizanov, A.: Implementing an Efficient Theorem Prover, PhD Dissertation, University of Manchester, Manchester, United Kingdom, (2003)

Robinson, J., A., and Voronkov, A., (editors): Handbook of Automated Reasoning, MIT Press, (2001)

Sage, A.: INCOSE 2020 Vision for SE education & research, Academic Forum, INCOSE 2006 Symposium, Orlando, Florida, July 9-13, 2006

Sudarsan, R., Foufou, S., Kemmerer, S.: Analysis of Standards for Lifecycle Management of Systems for US Army – a preliminary investigation; NIST Internal Report, NISTIR 7339, (2006)

SysML: OMG SysML Specification, ad/06-05-04

Ullman, D.: "The Ideal Engineering Decision Support System" http://www.robustdecisions.com/theidealenginsyste1.pdf (2007)

UML: UML 2.1.1 Superstructure Specification, formal/07-02-03 (2007)

XMI: MOF 2.0/XMI Mapping Specification, v2.1, formal/05-09-01, (2005)

Zave, P., Jackson, M.: Four Dark Corners Of Requirements Engineering In: ACM Transactions on Software Engineering and Methodology (1997)

Biography

Peter Denno is a Computer Scientist at the National Institute of Standards and Technology. He has 24 years experience in software solutions to engineering design and systems engineering problems. He has contributed to AP233 and SysML and was the technical editor of

the ISO STEP information mapping language, EXPRESS-X. His current work focuses on the Systems Engineering Interoperability Testbed, http://syseng.nist.gov/se-interop, a project associated with INCOSE's MBSE initiatives. Peter received a BS in mathematics from the University of Connecticut in 1983.

Thomas Thurman is a Principal Engineer at Rockwell Collins in Cedar Rapids, Iowa, USA. He received a BS in Electrical Engineering from St. Louis University, USA. He has 39 years of experience in the Avionics sector. For 18 years he developed instruments and instrumentation systems for engineering, manufacturing, and field support applications in the analog, digital, video, rf and microwave domains. For 5 years he served as technical lead in the development and implementation of an integrated CAD/CAE system for printed wiring board and assembly design. This included development and application of the first commercial integrated analog/digital system simulation environment. For the past 13 years he has been a technical expert assigned to the PDES Inc. consortium in the area of electronics. He serves as the project leader for the ISO Application Protocol for electronics design. His research interests include multi-disciplinary model mapping, systems integration, and novel applications of AP 210.

John Mettenburg is a Principle Systems Engineer in Commercial Systems at Rockwell Collins. He currently works on the application of Model Based approaches to reduce the cost and schedule of complex system integration. Previously, he worked in the Advanced Technology Center at Rockwell Collins on projects aimed at establishing the practice (and value) of rigorous architecture specifications for the purpose of advanced analysis, including formal methods. John's primary interest is embedded systems software development with an emphasis on increasing productivity. He started at Rockwell in 1993 working in the Government System GPS business, left in 1997 and returned to Cedar Rapids in 2001 to re-join the GPS business working on the MUE program. John transferred to ATC in 2005 and to Commercial Systems in 2008 and has been promoting the AADL as a foundation for software systems architecture modeling and analysis. John received a BSEC from George Mason University in 1986.