A Strategy for Improving Dynamic Composability: Ontology-driven Introspective Agent Architectures

Levent YILMAZ
Department of Computer Science and Software Engineering
Auburn University
Auburn, AL 36849, USA
yilmaz@auburn.edu

ABSTRACT

Seamless composability of disparate simulations within systems of systems context is challenging. Large complex simulations must respond to changing technology, environments, and objectives. The problem exacerbates when dynamic extensibility and adaptability are significant concerns in an application domain. Simulations that are dynamically extensible, while being composable require principled designs that facilitate engineering interoperability and composability in the first place. Basic concepts for composability are delineated to set the stage for emergent challenges. Specifically, the issue of sharing and exchange of simulations through advanced model bases that enable intelligent brokering and matchmaking is raised to frame the dynamic composability problem. The role of ontologies and their axiomatization is discussed as a potential strategy to improve dynamic composability.

Keywords: interoperability, composability, ontology, simulation, intelligent agents.

1. INTRODUCTION

An increasingly important trend in the engineering of complex systems is the design of component integration standards. Such standards define rules of interaction and shared communication infrastructures that permit composition of systems out of independently developed components. One problem with these standards is that it is often difficult to understand exactly what they require and provide (i.e., import and export features), and to recognize their substantive properties. To reduce the cost of developing complex simulations and to facilitate the process of building complex collaborative simulation systems, interoperability between disparate models is of paramount importance. Moreover, to make such a system highly extensible, individual federates, which could reside on the same or distributed hosts, should be able to freely join and leave a federation without full knowledge of its peer federates. Simply put, an ideal simulation system should allow for quick and flexible assembly of a complex simulation out of independently developed simulations on demand and at the same time allow the participant simulations to have maximum independence

Dynamic extensibility of simulations is needed at least for the following reasons:

1. For most realistic scientific problems, the nature of the problem changes as the simulation unfolds. Initial parameters, as well as models, can be irrelevant under emergent conditions. Relevant models need to be identified and instantiated to continue exploration. Manual exploration is not cost effective and realistic within a large problem state space.
2. Dealing with uncertainty is paramount in analyzing complex evolving phenomena. Adaptivity in simulations and scenarios is necessary to deal with emergent conditions for evolving systems in a flexible manner.
3. As simulations of complex phenomena are more and more used to aid intuition, dynamic run-time simulation composition with exploratory simulation will help identify (re)solution strategies that are flexible, adaptive, and robust.

In this work the separation of meta-level run-time composition mechanisms from the simulation infrastructure is suggested to facilitate evolution of composition strategies independent of the actual simulation infrastructure. We also discuss that by altering the way the meta-level operates one can dynamically evolve and extend the actual simulation system without ad-hoc interventions. Hence, we need simulation infrastructures that support change and extension while addressing the composability challenges between existing and new model components inserted at run-time. Large complex simulation systems must respond to changes in environment, technology, and requirements. The position advocated in this paper is that introspective agent architectures that support seamless dynamic evolution and extension in conjunction with formal axiomatized ontologies have the potential to address dynamic composability challenges raised above.

The rest of the paper is organized as follows. Section 2 provides a brief overview of existing related work on composability and interoperability in relation to the proposed strategy, which is outlined in section 3. Section 4 argues how formalized ontologies can aid reasoning about composability of models at run-time performed by agents within the introspective meta-level simulation. Finally, section 5 concludes by discussion of potential avenues of further research.

2. INTEGRATABILITY, INTEROPERABILITY AND COMPOSABILITY

Composability is defined as the capability to select and assemble components in various combinations to satisfy user requirements meaningfully [1]. Tolk and Muguira [3, 4, 5] suggests a model called Levels of Conceptual Interoperability (LCIM), the key point of which is the need for unambiguous interpretation of shared data at multiple levels of abstraction. In LCIM, technical interoperability refers to integratability via physical connectivity and shared protocols with standardized interfaces. Technical interoperability [2] via shared data structures and representation languages are viewed as necessary, but insufficient to achieve semantic interoperability. Harmonized alignment and interpretation of data and services exchanged between simulation services is a prerequisite for semantic interoperability. Finally, LCIM suggests the alignment of the conceptualization space as well as behavioral assumptions and obligations of models for meaningful composability of models.

Concomitantly, recent initiatives such as Extensible Modeling and Simulation Framework (XMSF) are targeting the development of web-scale simulation technology [6]. XMSF is based on a set of Web-based technologies that can be applied within an extensible framework to enable new generation of modeling & simulation (M&S) applications to emerge and interoperate [7, 8]. Concomitantly, Agent-Based Environment for Linking Simulations (ABELS) is another federated infrastructure that uses software agents to allow simulations to enter and exit a virtual simulation "cloud" of heterogeneous resources [9].

The framework uses a limited form of brokering and service matchmaking to facilitate loosely-coupled interactions among disparate simulations. However, none of these infrastructures currently has the capability to mediate incompatible interactions, improve composability, and support transparent simulation updating.

In engineering systems, hardware assembly (composability) is paramount but not universally realizable. The non universality is typical in systems approach. For example, the assembly of the "best" engine, the "best" body, the "best" wheels, and the "best" brake system not only does not end up with the "best" car, but components may be completely incompatible. Hence, the assembly may not be realizable at all. And if by some coincidence, the assembly is physically realized, the performance of the assembly may be far from being acceptable with respect to the requirements of intended users. In engineering applications, the selection of a hardware component cannot be done by functionality alone. There are compatibility standards and each component is labeled accordingly. This type of labeling (or documentation) can be named semantic labeling and has a cardinal role in selecting a hardware component. Furthermore, a given hardware component may be interchangeable by a set of other components.

This type of knowledge is also well documented for hardware interchangeability (substitutability). Hence, semantic labeling is necessary for pertinence (applicability) as well as interchangeability. Hence, the success of some engineering fields, such as mechanical and electrical, rely on composability and interchangeability (substitutability) of components into workable systems and by nesting, to the realization of systems of systems where components are also systems. However, hardware composability and interchangeability require disciplined approach in developing hardware components and labeling (documenting) their characteristics with great care. A warehouse of hardware components without any proper documentation about their usability and compatibility may not be sufficient for successful practice of component-based engineering. Similar considerations should be taken into account for successful practice of model composability.

Nayak's 1995 ACM Distinguished Dissertation showed that the general model selection problem for application composition is NP-hard [10] Others have shown that deciding whether an identified collection of submodels meet a stated set of objectives is an NP-complete problem [11, 12]. Currently faced difficulties of simulation model composability as well as worst-case theoretical limitations on automated model selection [13] should not be deterrent factors for model composability. Rather, necessary studies such as found in [14] should be conducted to overcome the apparent difficulties. At one stage of the maturity of modeling and simulation field, some systems were (erroneously) labeled as ill-defined systems. However, relentless studies have been influential in the advancement of for example, human behavior modeling and simulation.

Basic Concepts

In general, the term "composability" is the quality of being composable and means to be capable or worthy of being composed. Similar to other terms ending with –ability, for example acceptability, it refers to the object to which it applies and not to the agent (a model composer –human or software) which performs necessary acts to realize the composition of models and/or model components. In simulation, three aspects of "model composability" need to be elaborated on. These aspects are: related entities, related processes, and related characteristics (see Figure 1).

Entities - Model composability *is* related with the following *entities*:

- *e1* - A model composed from other models or model components (This model can be called a composed model(a synthesized model, an assembled model), or model, for short).
- *e2* - Models or model components from which one can compose other models (they are elements of a model base for composable models).
- *e3* - A model-base for models or model components from which one can compose other models.
- *e4* - An entity (human or preferably a software system) that composes (synthesizes) models from other models or model components. This entity can be called a model composer or composer, for short.

Processes - Model composability is related with the following processes:

- *p1* - Labeling of the models and model components in the model base prior to any search. Semantic labeling would entail, among other things, specification of the intention (or goal, or aim) for the use of the model, applicable assumptions, constraints, etc. For a model component, semantic labeling may necessitate its nature (e.g., variable, constant, parameter, state transition function, output function, etc.); for a variable, one can specify its type (input, output, auxiliary variable; if applicable, physical units, upper and lower acceptable values; for state variables, default initial conditions, etc.)
- *p2* - The process of formulation of a set of search criteria –based on the intention or the goal of the user– to detect relevant models and/or model components in the model base.

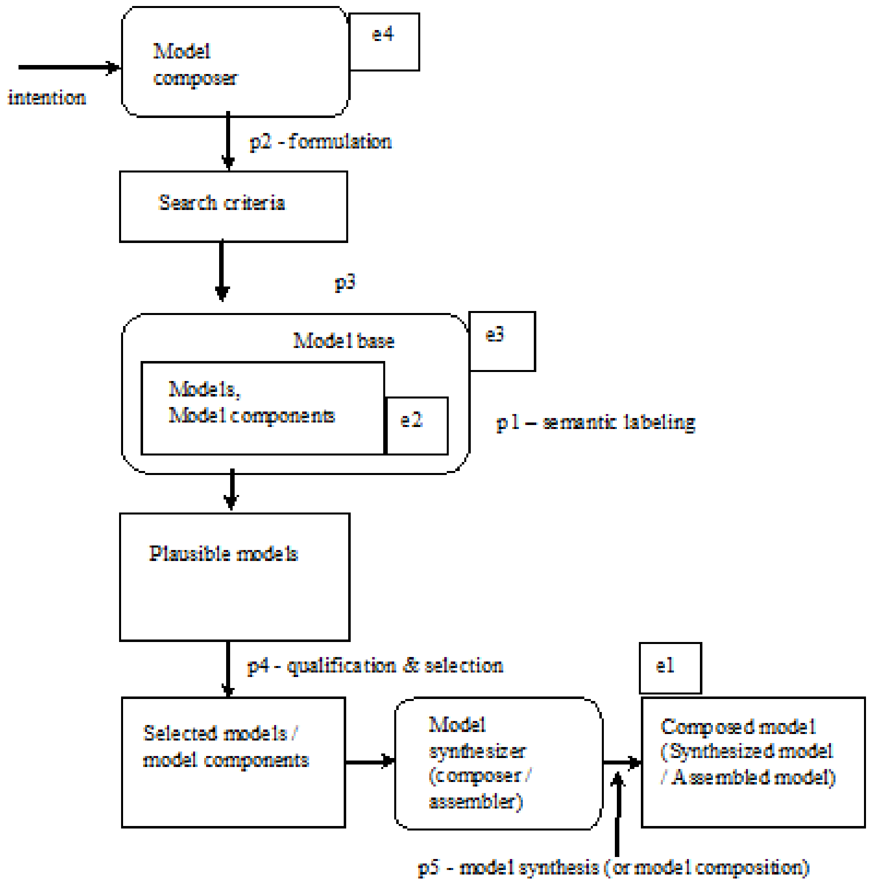
intention
Model composer
e4
p2 - formulation
Search criteria
p3
Model base
e3
Models, Model components
e2
p1 – semantic labeling
Plausible models
p4 - qualification & selection
e1
Selected models / model components
Model synthesizer (composer / assembler)
Composed model (Synthesized model / Assembled model)
p5 - model synthesis (or model composition)

Figure 1: Entities and Processes in Model Composability

- *p3* - Searching the model base according to the search criteria. (This may require a semantic search engine to be developed for the model base.) The result of the search may be some plausible models and/or model components.
- *p4* - Selection of relevant models and/or model components after screening plausible models or model components for relevancy. This is qualification and selection.
- *p5* - Synthesizing a model from selected model(s) and/or model component(s) (This process can also be called model composition or model assembly).

Characteristics - Model composability entails characteristics of the following entities:

- *c1* - Characteristic of the composed model: Within this perspective, model composability is the characteristics of a model to be synthesized (or composed, or assembled) from other models and/or model components into computationally (syntactically) and logically (semantically) coherent combinations that work together within a simulation system to satisfy user's intentions.
- *c2* - Characteristics of models or model components from which one aims to compose other models: From this perspective, models and model components need to be annotated to be analyzable for the determination of possible detection, selection, and relevance assurance for model synthesis. Hence, crude legacy models may need to be preprocessed for model composability. High-level specification languages may be useful in alleviating the need of semantic labeling.
- *c3* - Characteristics of model bases: A model base can be used for model composability, if the models and model components it contains are annotated to be analyzable for the determination of possible detection, selection, and relevance assurance for model synthesis.
- *c4* - Characteristics of model composer: A model composer needs: (1) the ability to process intention of model composition, (2) the ability to formulate a set of search criteria, (3) to access to a model base of properly annotated models and model components, (4) to perform relevance assessment of plausible models and model components, and (5) the ability to synthesize (or compose, or assemble) models from selected other models and/or model components into computationally (syntactically)

and logically (semantically) coherent combinations that work together within a simulation system to satisfy user's intentions.

While engineering disciplines successfully apply component-based approach to build systems, it has proven significantly difficult to apply in simulation model development. As such, advancements in the theory, methodology, and infrastructure of simulation modeling are needed to facilitate compositional development of components of simulation studies, such as simulation models, experimental frames as well as model behavior generators and processors.

Challenges

Improving composability through the realization of the characteristics of the entities and processes identified in this section require advancing the theory, methodology, and technology of simulation modeling. Figure 2 depicts these three elements of the composability infrastructure. In particular, the following prospective issues emerge as the challenges that need to be addressed to facilitate satisfaction of with the desiderata listed in section 1.

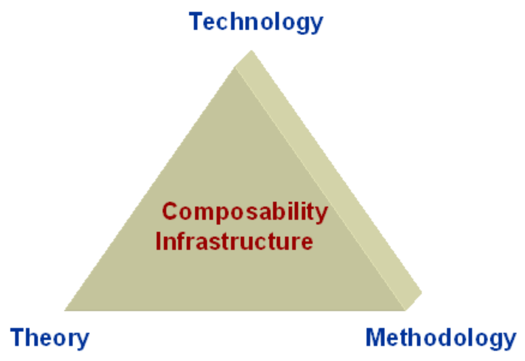

Figure 2: Elements of the Composability Infrastructure

- How can we improve the technology of sharing and exchange of simulations through advanced model bases that enable intelligent brokering and matchmaking between simulation goals (intentions) and contextual (i.e., experiential, conceptual, realization) assumptions of available models?
- From a methodology point of view, what are the components of conceptual models of composable and reusable simulation models? How can contextual assumptions of components can be packaged and distributed with simulation models to facilitate high precision context-sensitive search over model bases?
- With regard to theory, are there novel design constructs (other than popular but intractable component-connector strategy) that can facilitate development of a practical and sound model of composition. What would be the proper underlying unified theory with uniform syntax and semantics for composition rules that can take contextual assumptions into account?

3. IMPROVING DYNAMIC COMPOSABILITY

To address the above issues we need simulation infrastructures that support change and extension without causing interoperation and composability problems between the existing and new model components that are inserted at run-time. Large complex simulation systems must respond to changes in environment, technology, and requirements.

Requirements
To satisfy the desiderata listed in section 1 and to address the issues raised in section 2, we need to develop simulation infrastructures that support their own evolution. There are several forces associated with this problem:

- The simulation system needs to be updated without changing the underlying simulation software program. This is mainly due to at least two reasons. First, in real-time training simulations, emergent unforeseen actions (e.g., course of actions that are inconsistent with the learning objectives) may require updating the simulation to bring the trainee back to the scenario to achieve objectives. Second, simulation system may require performance tuning.
- Integrating changes and extensions should be uniform and easy.
- The proposed solution should facilitate not only structural, but also behavioral changes.
- It should be possible to incorporate certain changes that are not foreseen earlier at the design time.

Furthermore, the lack of machine processable formal annotations describing the behavior, assumptions and obligations of federates is a fundamental roadblock, as such information pertains to (1) finding and matching candidate models, (2) infer limits on the use and interpretation of federates, and (3) perform run-time mediation and facilitation (i.e., translation) among disparate federates. To facilitate formal composability as envisioned here, advances in the following areas are needed:

1. **Formalisms:** There is a need for formalisms that form the basis of annotating models with profiles that include assumptions, objectives, and constraints. Such information should be amenable to inference needed to identify and qualify models. HLA FOM and BOM models fall short such inference since specified interactions simply denote the syntax and type information. Furthermore, the type information is not available at run-time to facilitate analysis for composition even at the syntactical level.
2. **Ontologies:** Ontologies based on a specification formalism need to be utilized to capture various facets of models and simulators to describe the kinds of composability information needed in a given problem domain. Yet, the underlying formalism must be general enough to accommodate various problem domains.
3. **Profiles:** Each model needs to be annotated with a schema that describes the services they provide in terms of domain-specific ontologies. Such service ontologies may include (1) declarative advertisements of model properties and capabilities, in the form to be used for automatic federate discovery, qualification, and instantiation (2) declarative APIs of federates for execution, and (3) declarative specification of the assumptions and obligations of federates and their capabilities to infer the consequences of their use during automatic run-time federate composition.
4. **Tools:** To aid the instantiation and configuration of simulations, tools are necessary to perform inference and

make run-time decisions about composing candidate models.

Strategy
We propose a meta-level introspective agent architecture that comprises various agents that coordinate and orchestrate seamless information, data, and service exchange among conceptually interoperable simulations. Figure 1 depicts an agent organization that constitutes mediator, facilitator, broker, and matchmaker agents that are proposed to perform the necessary data and service management, alignment, and transformation functions.

Furthermore, the agent organization aims to decouple the simulation from the intricate details of instantiating and interoperation of a family of models to avoid explicit concrete assumptions and facilitate seamless reconfiguration with alternative ensembles. This way, the agent organization abstracts the simulation instantiation and interoperation process. It helps make a simulation system independent of how its models are created, composed, and represented. The organizational domain encapsulates the knowledge about which models the simulation uses. Furthermore, the concrete organization hides the details about how simulation programs for these models are created and composed together. Therefore, the decoupling of the instantiation and interoperation processes from the simulation infrastructure gives significant flexibility in terms of what concrete components get instantiated and exchanged, who instantiates them, how they get created and transformed, and when.

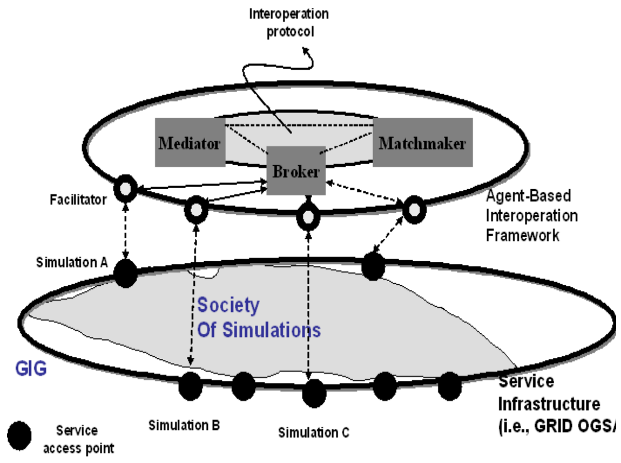

Figure 3: Meta-simulation Layer

Figure 3 illustrates how the interoperation framework and its components are positioned with respect to the service infrastructure. An infrastructure that facilitates dynamic composability and interoperability needs to be aware of its evolution. MSF is provided by specific functions, by which simulations can alter meta-simulations (facilitator agents) to influence the subsequent behavior of the simulation. More specifically, the MSF aims to provide the facilities that

1. establish a self-representation of each simulation,
2. provide means by which this representation can be manipulated , and
3. assure that the manipulations to the self-representation immediately affect the behavior of the simulation system.

In effect, the simulation system's self-representation is causally connected to the behavior of the actual simulation. The structure of a simulation application is divided into two components: (1) Simulation level and (2) Meta-simulation level. The simulation level includes the stable components of the model, simulation application level software objects, and the structural and behavioral dependencies between the components it includes. The meta level includes components that are subject to change, MSF agents (i.e., meta-simulation entities), each capturing a particular aspect of the structure and behavior of the simulation level. The MSF Façade object provides an interface to facilitate configuring or updating meta-simulations. The Façade object provides three categories of functions:

- **Reflection***:* Simulation level can access information about the simulation (itself) via facilitator agents associated with the simulation. This information can then be used to guide the behavior of the simulation.
- **Introspection:** Simulation level can access and update the parameters of existing meta-simulation entities (e.g., facilitator agents). This enables seamless and transparent update of the behavior of the simulation system, since the behavior of the simulation level is influenced by the meta-simulation entities.
- **Intercession:** Simulation level can change, exchange, insert, or remove meta-simulation entities and their connections to the simulation level. This feature enables dynamically including or inserting new simulations into the society of simulations at run-time.

The *facilitator agent* in Figure 1 acts as a gateway between the simulation infrastructure and the agent organization that orchestrates the simulation interoperation. Simulations join a society of simulations by registering their facilitator with the meta-level interoperation protocol. As a controller, the facilitator agent is aware of the capabilities and needs of the simulation service that it is associated with. The requests coming from the simulation domain will be delegated to brokering, matchmaking, and mediation agents in accordance with the embedded interoperation protocol that facilitate seamless data discovery, location, retrieval, and transformation. The mediator agent is responsible for converting simulation content to/from a common reference model (i.e., C2IEDM). To facilitate mediation, conflicts between the assumptions and obligations of simulations need to be resolved.

The interaction between content (i.e., data, model) requesting facilitators (consumers) and potential service providers (producers) are achieved via flexible mechanisms that can vary depending on the characteristics of the application domain. The brokering protocols include recommendation, recruiting, and notification [15]. Qualifying content specification objects and then ranking them requires interpretation of specifications to compute relevance metrics. The matchmaker agent measures the distance between the requested and target objects to qualify concepts within a common domain ontology used by the mediation component.

4. ONTOLOGY-DRIVEN COMPOSABILITY

The strategy presented above facilitates seamless evolution of the simulation. Yet, to improve composability for run-time extension requires mechanisms that address the requirements that pertain to ontologies, profiles, their underlying formalism, and associated tools that enable inference.

Conceptual alignment between a model and its new context needs to be established for meaningful composition. Unless the composition of two models is expected to generate the desired outcome and satisfy the requisite objectives, assuring their interoperability may not be of value for the simulation study. Ontologies (i.e., SPEM – Software Process Engineering Metamodel) in the MSF framework, if engineered with composability in mind, could improve dynamic composability. As a principle, the tasks for which the ontology will be used need to impose requirements on the ontology. The *aptitude* of an ontology is defined as its capability to respond to a set of questions and evaluations with respect to a specific requirement. Specifically, we define *composability aptitude* of an ontology in the context of Rational Unified Process (RUP) as the capabilities of the ontology to facilitate querying and performing inference pertaining to composability. This raises the issue of the extent of inferencing and deductive capabilities that is to be assumed by an ontology.

One possible strategy is to define the ontology as a specification of conceptualization that includes objects, attributes, and their relations. The properties of the objects and the relations over them can be defined in terms of predicates. Finally, a set of axioms can be defined as the constraints over the objects and relations. The axiomatization of the ontology provides a declarative specification of various definitions and constraints on the domain of discourse. The consistency of the constraints of the ontology and the results of the queries imposed on the identified models provides a basis to evaluate the composability of the context and the new model. More specifically, if the structural and behavioral capabilities of the model defined in first-order logic satisfy the constraints required by the ontology on that model, we can safely substitute the identified model for composition. However, this strategy requires associating metadata with models so that structural (e.g., what role does a model play?) and behavioral (e.g.,what are the activities available for a role to achieve its goal?) aptitude queries can be applied. Introspective models [15] that enable access to their own specification could be useful to serve that purpose. The results of these queries can then be used to evaluate the model against the axioms of the ontology to determine if it is consistent with the domain invariants.

An Ontology for Software Process Simulation

To illustrate the utility of the proposed strategy we provide an example that pertains to software process simulation. The metamodel depicted in Figure 4 presents the concepts and relations captured in the simulation. According to the ontology, an organization aims to achieve a number of performance *objectives* in terms of a *strategy*. The organization is specified by *organization design*, the layout of which is defined by the *structure* component. The function of the organization is defined by the *behavior* class that constitutes primitive or composite task objects. Tasks are performed by *agents* playing *roles* that exhibit *skills* related to solving the *tasks*. The behavior of agents is moderated by behavior moderators such as *technology, turnover,* and *deadlines*. Agents interact with each other, act on objects in the environment, and use *resources* to complete their task.

The conceptual domain model presented in an ontology provides a common shared vocabulary. However, existing ontology specification languages are limited in describing the constraints on properties of concepts as well as relations. On the other hand, composability analysis requires clear and precise description of the assumptions and obligations of models with respect to each other within the compositional structure. Furthermore, domain constraints impose restrictions on properties of individual models (i.e., a team should include at least two agents). Axiomatization of an ontology facilitates specification of such constraint so that metasimulation level agents can make inference about the composability of models.

Formalization of Ontologies using Axioms

The constraints over a domain can be classified in three categories: (1) domain invariants, (2) model assumptions, and (3) model obligations. Domain invariants refer to constraints that must hold true for all models and their relations that are instances of the concepts and dependencies represented in the metamodel. For instance, the constraint that requires every team in the simulation to contain at least two agents as team members is a domain invariant.

Domain Invariants: Domain invariants may not only constrain the properties of a single model, but may also pertain to constraints over a set of models and their dependencies. According to the ontology shown in Figure 4, an organization is specified by organization design and realizes a strategy (i.e., innovation, market expansion, risk reduction).

specified_by (o, od)
realizes (s, o)

The predicates *specified_by* and *realizes* with parameters *o* (organization), *od* (organization design), and *s* (strategy) denote the instances of associated concepts in the ontology. While each instance may be a model on its own, a combination of entities may be aggregated to constitute a composite model. In that case the entities are considered to be the components of a model.

Agents play roles that require skills needed for a given task.

plays(a, r) and *has(r, S),* where *a*, *r*, and *S* denote the specific agent, its role, and the set of skills played by the agent, respectively

Similarly, the agent under consideration exhibits a specific behavior (e.g., defined by an algorithm encapsulated in an instance of the behavior class) to fulfill a task that requires a skill set. Note that this skill set should be the proper subset of *S*.

Formally,

exhibits(a,b) and *implements(b,t)* and *requires(t, S')* and $S' \subseteq S$

Another domain constraint is that only two or more agents can form a team; hence, we have:

For all t, where team(t), there exist two agents a_1 *and* a_2*, s.t* $a_1 \neq a_2$ *and has (t,* a_1*) and has (t,* a_2*).*

Domain constraints can be imposed on the properties of individual concepts and models. For instance, the simulation may require specific types of agents (i.e., *subsumption reactive* architecture) that use a *contract net task allocation* in conjunction with coordination by synchronization protocol. Such properties can be encoded in the domain metamodel as well as the metadata associated with individual models to facilitate matching domain constraints to model properties.

Assumptions: Assumptions specify the expectations of a model (i.e., a concept or collections of concepts that constitute a model) from external entities that depend on it. In our example, a task model in a simulation requires the existence of an instance of corresponding behavior component that implements the task specification. The metadata for a behavior component includes its precondition (*pre(b)*) and postcondition (*post(b)*). A precondition is a predicate that must be true before enacting the behavior, while a postcondition is the predicate that must be true after the completion of the behavior. Similarly, a task is associated with a precondition (*pre(t)*) and postcondition *(post(t))*.

The relation between the task and its associated behavior can be characterized as follows:

$$Implements(b,t) = pre(b) \Rightarrow pre(t)\ and\ post(t) \Rightarrow post(b)$$

Obligations: Obligation of a model refers to the formalization of the relation(s) between the source model (concept under consideration) and a target model (concept). Note that in the metamodel the relations are binary; therefore each relation involves two components. In our example, a task model requires a skill model. One possible interpretation of this relation is in terms of a domain constraint. That is, the behavior that implements the task is performed by an agent that plays a role, which requires a set of skills. The skills needed for this task need to be a proper subset of the skills of the role that performs the task.

$$\forall a, \exists r\ plays(a, r) \text{ and } has(r, S) = \underline{requires(t, S')}\ and\ S' \subseteq S$$

Analyzing the composability of a model involves (1) assuring its alignment with the domain constraints and (2) satisfiability of metamodel relations that it participates by taking into account the local assumptions and obligations.

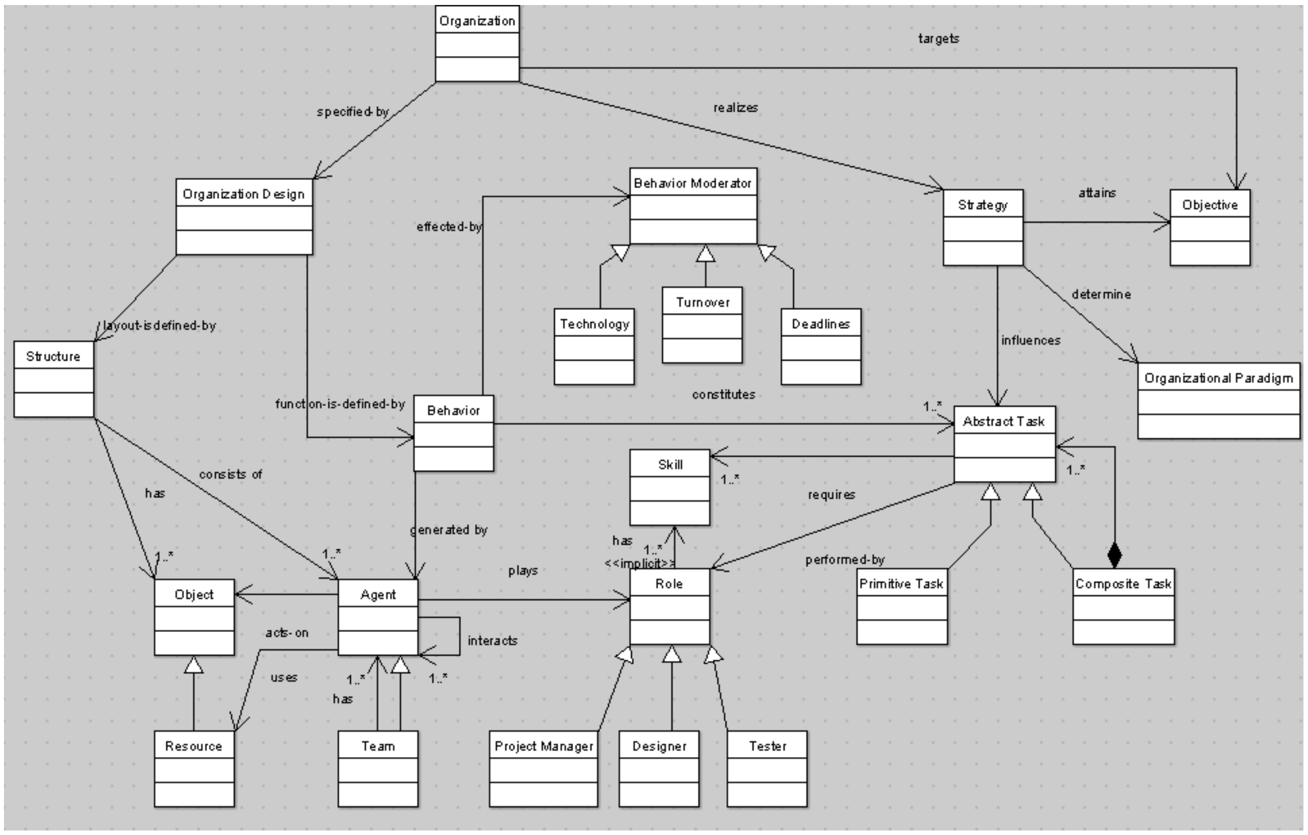

Figure 4: A Partial Domain Ontology

On the Need for Introspective Models

To evaluate such queries to decide the composability of models depicted by (possibly collection of) concepts shown in Figure 4, an inference engine needs to decide if

- the properties (predicates) of the model under consideration are consistent with the domain constraints,
- the assumptions of the model are not violated with respect to other entities that it is related to according to the metamodel, and
- the obligations of the model do not violate the assumptions of the entities that it is associated with.

To facilitate inference, a composability analyzer at the metasimulation level needs to have access to the metadata associated with each potential model before it is qualified for composition. Therefore, similar to the introspective access from the simulation level to metasimulation, a reflective access from simulation to metasimulation level would enable accessing specification information. As discussed in [16] predicates that depict the constraints of simulation models hold out the potential to improve ability to understand and reason about the fitness of a simulation model in a new context. Achieving these benefits, however, involves effective communication and distribution of such constraints. This requires delivering contextual assumptions along with the simulation model itself. The significance of this issue is apparent: a client of a simulation model can not harvest the benefits of the specification of the concept and contextual assumptions, unless the specification is delivered along with the simulation model. While conventional methods used in distributing specifications and documentation in printed form, as plain text or HTML, or in a platform specific help file can be utilized, such an approach misses the opportunity to appropriately use such contextual dependency information by leveraging them in development and integration tools. The ability of a system to respond to inquiries about its structure and ideally its behavior is a well-studied concept and is the cornerstone of computational reflection [17,18]. Computational reflection is defined as an activity of a system to query its structure and behavior to guide its own computation by accessing and potentially updating its own state. Many OO languages provide facilities for accessing interface information (i.e., Java), yet the packaging of concept specification and contextual information requires embedding coarse-grain specification objects accessible through the interface of simulation models.

While the proposed strategy discussed in [16] enables seamless access to specification objects that facilitate awareness about the specification of the model, there are a number of drawbacks. An immediate issue is the augmentation of the simulation models with such objects and associated services that may lead to code inflation. This strategy naturally requires development of drivers to deploy reused simulation model components and retrieve their specification objects. Development of such drivers needs to be cost effective. Another concern entails the representation and communication of specification objects. First and foremost, the representation must be interpretable and be manipulated through program level accesses to the contextualized simulation models. The use of semi-structured data representation mechanisms such as XML is a plausible strategy. Committing to a specific format and having a generally acceptable standard structure that defines the conceptual constraints of a simulation model are challenging tasks.

5. CONCLUSIONS

This paper lays out basic concepts to advance composability through progress in theory, methodology, and technology. While simulation science is founded on powerful foundations, there is still need for improvement to facilitate addressing emergent challenges of reuse and composability. As such, we delineate the requirements and characteristics of a composability infrastructure. We argue that, unlike ad hoc solutions to composability, advancements in simulation theory and methodology along with their support in the development of next generation infrastructures could provide a sound basis.

The separation of composition protocols from the simulation infrastructure constitutes the primary contribution of the proposed strategy. The proposed level of indirection via an agent organization aims to decouple the simulation system from the intricate details of instantiation and interoperation of a family of models to avoid explicit assumptions and to facilitate seamless (run-time) reconfiguration with alternative ensembles. This way, the agent organization abstracts the simulation instantiation and interoperation process. There are two major principles underlying the proposed strategy that makes it useful for the improvement of composability. First, the agent organization encapsulates the knowledge about the interoperation administration, alignment, transformation, and management functions. Second, it hides the details about which simulation services are discovered, located, and instantiated as the simulation unfolds. As such, it has the potential to make a federated simulation system independent of how federates are created, composed, and represented. While many of the conflicts that exist between disparate simulations can be resolved via man-in-the-loop simulations, providing such an agent technology can help operators focus on mission-critical activities as opposed to routine interoperability problems.

6. REFERENCES

[1] P. K. Davis and R. H. Anderson, Improving the Composability of Department of Defense Models and Simulations, **RAND Technical Report**, 2003, http://www.rand.org/publications/MG/MG101/[last visited June 2005].

[2] E. H. Page, R. Briggs, and J. A. Tufarolo, Toward a Family of Maturity Models for the Simulation Interconnection Problem, Paper 04S-SIW-145 **in Proceedings of the Spring Interoperability Workshop**. 2004.

[3] A Tolk, Composable Mission Spaces and M&S Repositories - Applicability of Open Standards, Paper 04S-SIW-009 in **Proceedings of the Spring Simulation Interoperability Workshop 2004.**

[4] A. Tolk and S. Y. Diallo, Model Based Data Engineering for Web Services," in **IEEE Internet Computing.** 2005

[5] A. Tolk and J. A. Muguira, The Levels of Conceptual Interoperability Model. **2003 Fall Simulation Interoperability Workshop** Orlando, Florida, September 2003.

[6] D. Brutzman., K. J. Morse, M. Pullen, and M. Zyda, Extensible Modeling and Simulation Framework (XMSF): Challenges for Web-Based Modeling and Simulation, **Interim Technical Report**. Naval Postgraduate School,

Monterey, California. Available online via <http://www.movesinstitute.org/xmsf/workshop/XmsfWorkshopReportDraft.pdf> [accessed August 1, 2004].

[7] Mikalsen T. and I. Rouvellou, Transactional attitudes: Reliable composition of autonomous Web services, **IBM Watson Research Center Technical Report, Available online** via <http://xml.coverpages.org/ni2002-04-03-a.html> [accessed February 10, 2004].

[8] White L. E and J. M. Pullen, Adapting Legacy Computational Software for XMSF, **Simulation Interoperability Workshop (SIW),** Fall 2003, Orlando, Florida.

[9] Murphy P. J., A. Mills-Tettey, L. F. Wilson, Greg Johnston, B. Xie, Demonstrating the ABELS System Using Real-World Scenarios, In **Proceedings of the SAINT.** 2003.

[10] P. P. Nayak, *Automated Modeling of Physical Systems*. **PhD thesis**, Stanford University. 1992.

[11] E. Page and J. Opper, Observation on the Complexity of Composable Simulation, **Proceedings of the 1999 Winter Simulation Conference** , pp. 553-560. 1999.

[12] M. D. Petty, E. W. Weisel, R. R. Mielke, Computational Complexity of Selecting Components for Composition, **Proceedings of the Software Interoperability Conference**, 03F-SIW-073, Fall 2003.

[13] A. Y. Levy, Y. Iwasaki, and R. Fikes, Automated Model Selection for Simulation Based on Relevance Reasoning," **Artificial Intelligence**, Vol. 96, 1997, pp. 351–394.

[14] P. K. Davis and A. R. Anderson, Improving the Composability of Department of Defense Models and Simulations, **RAND Technical Report,** 2003.

[15] L. Yilmaz and S. Paspuletti, Toward a Meta-Level Framework for Agent-supported Interoperation of Defense Simulations, **Journal of Defense Modeling and Simulation.** vol 2, no 3, pp. 161-175. 2005.

[16] L. Yilmaz, On the Need for Contextualized Introspective Simulation Models to Improve Reuse and Composability of Defense Simulations, **Journal of Defense Modeling and Simulation**, vol. 1, no. 3. pp. 135-145. 2004

[17] G. Kiczales, J. Des Rivieres, and D G. Bobrow, **The Art of the Metaobject Protocol.** MIT Press. 1992.

[18] M. Saeki, T. Hiroi, and T. Ugai, Reflective Specification: Applying a Reflective Language to Formal Specification," In **Proceedings of the 7th International Workshop on Software Specification and Design,** IEEE CS Press, pp. 204-213. 1993.