Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

ENGINEERING AB INITIO DYNAMIC INTEROPERABILITY AND COMPOSABILITY VIA AGENT-MEDIATED INTROSPECTIVE SIMULATION

Levent Yilmaz

Computer Science and Software Engineering
Auburn University
Auburn, AL, U.S.A.

Andreas Tolk

Virginia Modeling Analysis and Simulation Center
Old Dominion University
Norfolk, VA, U.S.A.

ABSTRACT

Complex software intensive simulation systems must respond to changing technology, environments, and requirements. Hence, dynamic extensibility and adaptability is a significant concern in an application domain. However, existing interoperability and composability solutions are limited in dealing with (1) dynamically evolving content needs of existing simulations and (2) run-time inclusion of new components into a simulation system. Simulations that are dynamically extensible, while being interoperable require principled designs that facilitate engineering extensibility, interoperability, and composability in the first place. We propose and examine the utility of a strategy based on an agent-mediated meta-simulation architecture.

1 INTRODUCTION

The Modeling & Simulation (M&S) community is taking steps to facilitate the improvement of integratability (Page 2004), interoperability (Turnitsa 2005), and composability (Davis and Anderson 2003) of simulation models. For instance, Tolk (2004) suggests the use of open standards along with explicit delineation of model interdependencies as a prerequisite for a practical solution to composability. Web services and the management of their composition (Tosic et al. 2001) via rich metadata are suggested as an enabling technology to improve interoperability and composability. Within the interoperability domain, various models of interoperability (Morris et al. 2004) are suggested to facilitate the improvement of different dimensions (i.e., programmatic, constructive, and operational) of interoperability.

The premise of this paper is based on the observation that existing interoperability and composability solutions are limited in dealing with the dynamically evolving content needs of existing simulations and the run-time inclusion of new components into a simulation system. Hence, we advocate that *simulations that are dynamically extensible, while being interoperable and composable, require principled designs that facilitate engineering extensibility, interoperability, and composability in the first place.*

1.1 The Need for Dynamically Extensible Simulations

The composability of simulations during dynamic updating and extension is needed at least for the following reasons:

- 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.
- Dealing with uncertainty is paramount to analyzing complex evolving phenomena. Adaptability in simulations and scenarios is necessary to deal with emergent conditions for evolving systems in a flexible manner.

1.2 Interoperation in Extensible Simulations

The work described in this paper involves the introduction of the notion of an agent-based meta-level, called Meta Simulation Framework (MSF) to improve interoperability and composability over the simulation infrastructure via explicit separation of run-time interoperation and composition mechanisms from the simulation environment. We do not claim to solve all the composability challenges (Davis and Anderson 2003), but rather suggest a strategy and framework within which such challenges can be addressed in a tractable and effective manner. In particular, agent-based mediation, brokering, matchmaking, and facilitation services are suggested as critical components for seamless and transparent interoperation and composition.

The proposed approach is based on the following premises: (1) deployment of various forms of brokering among interaction producers and consumers within a simulation system improves transparency and balance of workload in interoperation; (2) ontology-driven matchmaking mechanisms provide flexible (ability to adapt to changing schemas), efficient, and effective retrieval than conventional keyword-based discovery and matching mechanisms; (3) mediation facilities decoupled from the simulation infrastructure improve transparency and run-time extensibility. Note that run-time extensibility and dynamic composability (Davis and Anderson 2003) implies situations where data, scenario, and service needs of potential simulations, which may join to and detach from the society of simulations, may not be foreseen in advance at the design time.

This requires mechanisms that facilitate run-time recommendations and alignment of used models, along with mediation facilities that support seamless exchange of data and services within the conceptualization (composability) and realization (interoperability) space. The proposed strategy is based on the observation that for transparency and improvement of the reuse of simulations such interoperation protocols should be decoupled from the simulation infrastructure (Yilmaz 2004).

2 CONCEPTUAL INTEROPERABILITY LEVELS

Current research results led to the development of the Levels of Conceptual Interoperability Model (LCIM), which was presented in several papers (Tolk and Muguira 2003). LCIM copes with different layers of interoperation from technical aspects to conceptual ideas, which are the basis for the purposeful abstraction of reality underlying an M&S application. Similar ideas are found in Page et al. (2004) who introduced the idea of using three dimensions of integratability, interoperability, and composability in their paper.

The LCIM introduces seven layers to cope with different aspects of interoperation. These aspects are characterized as follows:

- Stand-alone systems have *No Interoperability*.
- On the level of *Technical Interoperability*, a communication protocol exists for exchanging data between participating systems.
- The *Syntactic Interoperability* level introduces a common structure to exchange information, i.e., a common data format is applied.
- If a common information exchange reference model is used, the level of *Semantic Interoperability* is reached. On this level, the meaning of the data is shared.
- *Pragmatic Interoperability* is reached when the interoperating systems are aware of the methods and procedures that each are employing. In other words, the use of the data – or the context of its application – is understood by the participating systems.
- If systems have attained *Dynamic Interoperability,* then they are able to comprehend the state changes that occur in the assumptions and constraints that each is making over time, and are able to take advantage of those changes.
- Finally, if the conceptual models – i.e. the assumptions and constraints of the meaningful abstraction of reality – are aligned, the highest level of interoperability is reached: *Conceptual Interoperability*.

The first category of layers copes with hardware and firmware requirement and lower communication protocols, referred to as means of integration enabling *integratability* of components. This also includes problems of the underlying network including network connectivity. The second category copes with implementation issues of interoperation, supporting *interoperability* of components: syntactic and semantic interoperability, which is the simulation side of M&S. Finally, the third category targets the *composability*, the modeling side of M&S. Hoffmann calls these layers the "Model Based Information Processing System" specific issues (Hoffmann 2004), which is the modeling part of M&S.

3 PRINCIPLED DESIGN OF SIMULATION SYSTEMS FOR DYNAMIC INTEROPERATION

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.

3.1 Requirements

To satisfy such needs 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 certain training simulations the unforeseen interactions may require updating the simulation to bring the trainee back to the scenario to achieve pedagogical objectives. Second, simulation system may require performance tuning at run-time.
- Integrating changes and extensions should be performed in a uniform manner.

- The proposed solution should facilitate structural and behavioral changes.
- It should be possible to incorporate changes that are not foreseen earlier at the design time.

3.2 Strategy

Recent initiatives such as Extensible Modeling and Simulation Framework (XMSF) are targeting the development of web-scale federated simulation technology (Brutzman et al. 2002). 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 (Mikalsen and Rouvellou 2002, White and Pullen 2003). 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 (Murphy et al. 2003). 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, take composability challenge into account, and support transparent simulation updating.

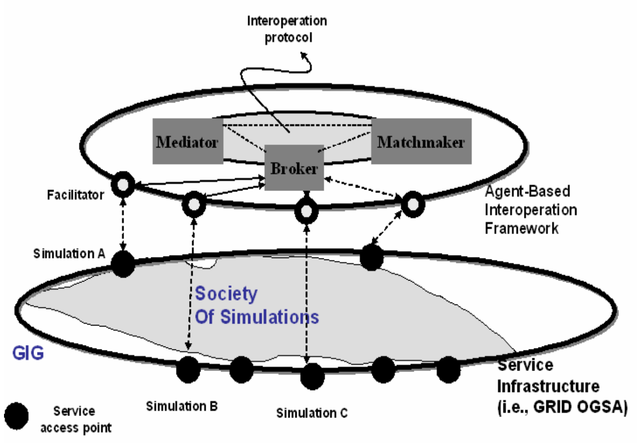

Figure 1: Agent-Mediated MSF

We propose a meta-level introspective agent architecture that constitutes various agents that coordinate and orchestrate seamless information, data, and service exchange among conceptually congruent simulations. To support the requirements listed in section 3.1, an agent-based strategy is suggested. 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 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* gets instantiated and exchanged and *how* it gets created and transformed, and *when*. Figure 1 depicts how the interoperation framework and its components are positioned with respect to the infrastructure. Section 5 provides a synopsis of the functionality of each one of these components.

Infrastructures that facilitate dynamic composability and interoperability need to be aware of its evolution. The components of the Meta Simulation Framework (MSF) in conjunction with meta-simulations (i.e., facilitator agents) enable altering the behavior and structure of the simulation. MSF should include selected aspects of the simulation application that are subject to change (i.e., algorithms). 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 the simulation,
2. Offer means by which this representation can be updated , and
3. Assure that the manipulations to the self-representation influence the behavior of the simulation system.

In effect, the simulation system's self-representation is connected to the behavior of the actual simulation. Hence, the structure of a simulation application is divided into two components: the Simulation Level and the 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, and three categories of functions:

- **Reflection***:* Simulation level can access information about the simulation 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 enable dynamically including or inserting new simulations into the simulation system at runtime.

4 SIMULATION MODELING OF RUP

To illustrate the utility of the proposed strategy we provide an example that pertains to the simulation of the software process, called Rational Unified Process (RUP) (Larman 2005). We examine two problems in the context of the simulation RUP: (1) transparent update and seamless dynamic evolution of models within the simulation and (2) dynamic interoperability and composability of models during evolution of the simulation.

4.1 Rational Unified Process

RUP divides a (spiral) cycle into four phases: Inception, Elaboration, Construction, and Transition. In Inception, the scope and vision of the project are defined. The organization may also specify a business case during this phase. Elaboration is characterized by the development of an architecture, the planning of activities, and the gathering of resources.

During Construction, work products from earlier phases are evolved and the product is actually built. In the Transition phase, the product is delivered to the user and maintenance activities are performed. At the end of each phase, a milestone is reached that serves as a point to evaluate the overall progress of the project and to affirm the practicality in continuing. The exact nature of the phases will also depend on context in which RUP is applied (Krutchen 1999). While each phase has overlapping activities, the dependencies and characteristics (i.e., order, priority, duration) exhibit differences. Therefore, as a shift occurs from one phase to another the activity model needs to be updated transparently.

4.2 Software Process Engineering Metamodel (SPEM)

To provide a self-representation software process simulation taking place at the base level, the MSF uses SPEM, which is a metamodel (see Figure 2) for defining software engineering process models and their components (OMG 2006). SPEM is a generic metamodel that needs to be instantiated to depict alternative software process models such as RUP, Extreme Programming (XP), etc. SPEM enables the interpretation of ontologies (domain spaces) that may vary during simulation.

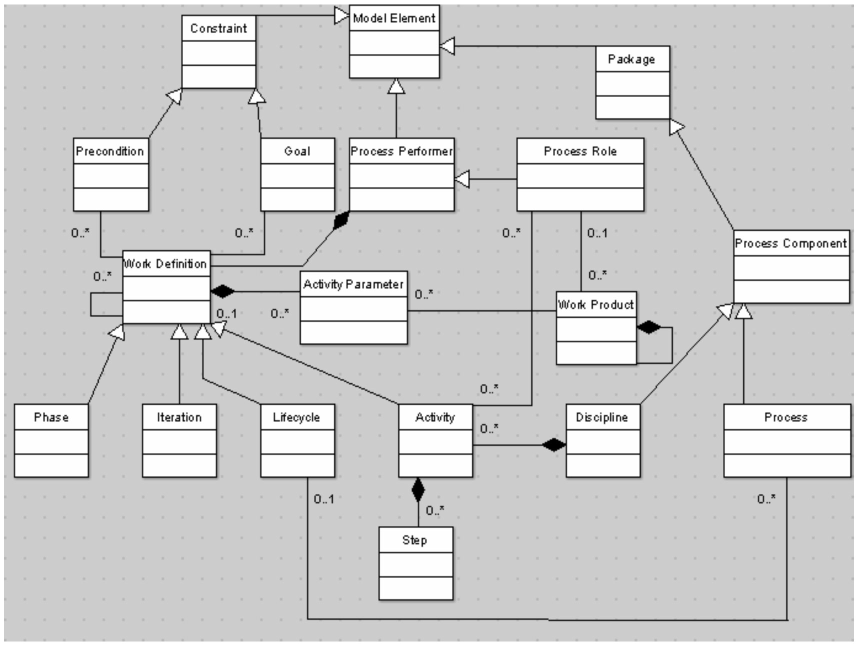

Figure 2: A Partial Software Process Engineering Ontology

SPEM can be used as a service for MSF agents to understand dynamically changing (updated) domain space that represents a metamodel for alternative processes. At the core of SPEM is that software process is a collaboration between abstract entities classified as *process roles* that perform operations, called *activities*, on concrete tangible artifacts, called *work products*. An overview of the elements of SPEM along with their role in process modeling is presented in (Lonchamp 2005).A process model such as RUP involves a specific interpretation of the metamodel. That is, the types of workproducts, the process roles, and specific activities along with their dependencies in RUP due to incremental, iterative, and evolutionary nature of the process is different than a waterfall lifecycle. Furthermore, within a given domain space for RUP distinct phases of the process (i.e., inception, elaboration) may involve specific activity models pertaining to unique task dependencies.

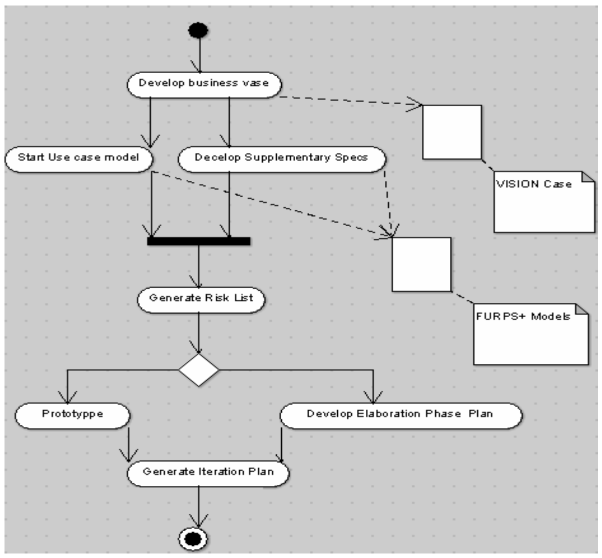

Figure 3: Activity Model for the Inception Phase

For instance, while the inception phase focuses on vision and business case, supplementary non-functional requirements, and risk management, the elaboration phase is consumed with use case development, architectural design, detailed design, and implementation of the selected use cases. Figure 3 depicts the conceptualization of hypothetical activity dynamics for the inception phase. An introspective simulation system may utilize the activity model shown in Figure 3 to generate its behavior using an activity scanning approach. Separation of the activity model from the baseline simulation system facilitates its seamless update. Section 5.1 elaborates on how this is possible. As soon as the elaboration phase starts the MSF needs to update the activity model to switch into a set of activities associated with the elaboration phase of the RUP.

4.3 Dynamic Model Updating

The challenges in dynamic model updating are the issues involved in substituting a new model without taking the simulator offline. The following five conditions present the basic requirements (Litmus test) for dynamic model replacement.

- **Activation:** Submodel replacement must be initiated, either internally by a submodel or externally by a scheduler.
- **Integrity:** The consistency of submodels undergoing replacement needs to be preserved. The event scheduling and simulation protocol need to be governed to facilitate interleaving module replacement activities with the simulation events.
- **Submodel Instantiation:** The new submodel must be dynamically loaded and linked into the run-time environment of the simulator (simulation engine). This requires new model and simulator decoupling strategies that avoid persistent connections. Also, the intricate details of complex submodel construction process should be as independent of the simulation as possible to enable flexible update.
- **State Reconstruction:** The state of a model must be reconstructed or at least resume from a specific state when re-instantiated after an update operation. This requires externalization through abstraction, state saving, transmission, and reconstruction after the update operation.
- **Simulator Rebinding:** Once a model is loaded and linked to the run-time environment, the simulator needs to be bound to the new model.

5 THE CONCEPTUAL MODEL OF MSF

The strategy underlying the design of the MSF is influenced by the requirements presented in sections 3.2, and 4.3. The protocol of MSF includes facilitator, mediator, broker, and matchmaker agents to facilitate seamless updating, interoperation, and composability of disparate simulations.

5.1 The Facilitator Agent

The facilitator agent acts as a gateway between the baseline simulation infrastructure and the MSF agent organization that orchestrates the simulation interoperation. Simulations join a society of simulations by registering their facilitator with the meta-level interoperation protocol. Facilitators encapsulate those aspects of the simulation that are subject to change. By altering the behavior of their own facilitator agents via the MSF protocol, the simulations in the baseline effect their own behavior. For instance, the facilitator agent in Figure 4 encapsulates the activity models (i.e., inception, elaboration) for RUP and makes them interchangeable. As such, the facilitator agent assures that the protocol can vary independently from the simulations in the baseline. The responsibility for updating the activity model may be allocated either to an observer component that monitors the state of the process or to the baseline simulation that instructs the MSF protocol to update the facilitator.

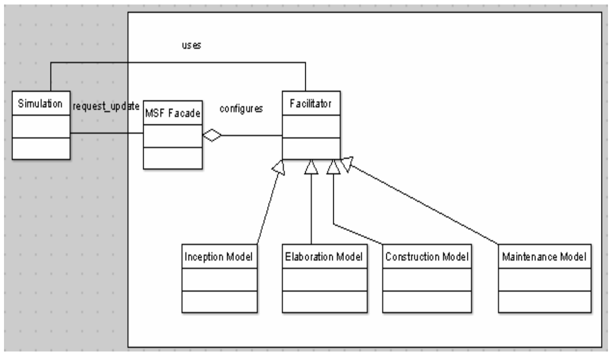

Figure 4: The Facilitator Component

The MSF façade and facilitator interact to implement the chosen protocol (i.e., activity model). The MSF façade may pass all the data required by the model to the facilitator when the new activity protocol is enacted. The MSF façade forwards the request for update from the simulation baseline to the facilitator (meta-simulation) of the simulation. Simulations can create concrete facilitators and pass them as parameters to the MSF façade. Thereafter, the simulation interact with the concrete facilitator exclusively.

5.2 The Mediator Agent

The mediator agent is responsible for converting simulation content to/from a common reference model (i.e., C2IEDM). As such, the mediator agent is critical for interoperation via meaningful exchange of services and data. To facilitate mediation, conflicts between the assumptions and obligations of simulations need to be resolved. As discussed in (Tolk and Diallo 2005) four types of conflicts are common between the desired and provided services. (Note that interface mismatches can also be categorized as *semantic* or *structural*.)

- *Semantic* conflicts occur when the local schemata objects must be aggregated or disaggregated, but fail to match.
- *Descriptive* conflicts occur when the same concept is described in term of synonyms, homonyms, or different attributes, values,
- *Heterogeneous* conflicts occur when concepts are described in terms of substantially different methodologies.
- *Structural* conflicts occur when concepts are defined in terms of different structures in the schemata (metadata).

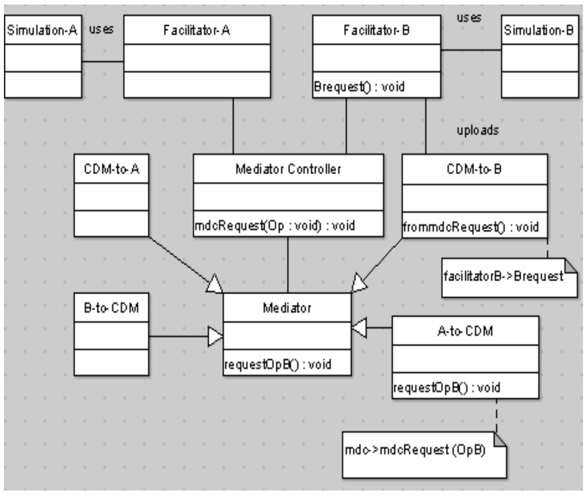

Figure 5: The Mediator Component

In our strategy, each simulation is responsible for uploading two adapter algorithms encapsulated in a class to translate the content and interface to/from the common conceptual model. Figure 5 depicts the structure of the mediator agent.

5.3 The Broker Agent

The interaction between content requesting simulation entities and potential service providers (producers) are achieved via flexible mechanisms that can vary depending on the characteristics of the application domain. Various brokering protocols are as follows:

- **Recruiting:** The consumer consults with the broker agent to select a producer that provides the most relevant content via matchmaking and delegates the request to the facilitator of that producer (i.e., content management system, simulation application, or C2 system). However, any subsequent responses from the producer are sent directly to the consumer through their facilitators.
- **Recommendation:** The consumer consults with the broker agent to perform matchmaking to retrieve a list of matched models. The broker agent presents the list of specifications along with their rankings to the consumer (i.e., facilitator agent), which then selects and contacts the desired service provider.

- **Notification/Subscription:** The consumer subscribes via a broker to a specific type of content and is notified when it is available or published by a producer. The facilitator of the consumer then interacts directly with the facilitator of the producer to include the new model.

Brokering among content producers and consumers in a simulation system brings flexibility via fine grain interoperation that takes specific constraints of individual simulations into account. For instance, two simulations that use a common ontology and vocabulary may not require mediation. In that case, the recommendation protocol would be sufficient, as the producer and the consumer can directly exchange data and services through their facilitators. On the other hand, when conflicts among simulations need to be resolved, the recruitment protocol can be used so that the broker can submit the requests to the producer along with instructions for mediation.

5.4 The Matchmaker Agent

For dissemination and recommendation of services, various protocols in different contexts have been studied (Sycara et al. 2002). Matchmaking is the cooperative partnership between information providers and consumers along with the help of an intelligent facilitator (Genesereth 1992). Using this approach information providers actively publish their capabilities and services to the matchmaker, and the consumers send requests for their desired information to the matchmaker. Matchmaking enables the providers and requesters to exchange dynamically changing information in a more effective and active manner than traditional methods. Several metrics are proposed in the simulation modeling community to measure the conceptual and structural distance as a similarity metric between two data elements to be exchanged between two simulations (Yilmaz and Paspuletti 2006). Matchmaking mechanisms have been widely used in various applications and fields, where information changes rapidly, like product development and crisis management (Kuokka and Harada 1995). Yet, dynamic interoperation of such services is critical for coherent operation of the overall system. Furthermore, a matchmaking strategy with quantitative distance metrics provides insight about the extent of alignment needed for two models to interoperate.

6 DESIGNING ONTOLOGIES FOR DYNAMIC COMPOSABILITY

Conceptual alignment between a model and its new context needs to be established before it can be assessed for interoperability. Unless the composition of two models are expected to generate the desired outcome and satisfy the requisite objectives, assuring their interoperability may not be of value for the simulation study. Ontologies 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 needs 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 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 is a safe extension of 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 doe 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 (Yilmaz 2004) 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.

7 CONCLUSIONS

The separation of interoperation and 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 federates 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 interoperation. First, the agent organization encapsulates the knowledge about the interoperation administration, alignment, transformation, and management function. 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.

REFERENCES

Brutzman, D., K. J. Morse, M. Pullen, and M. Zyda. 2002. Extensible Modeling and Simulation Framework (XMSF): Challenges for Web-Based Modeling and Simulation, Interim Technical Report. Naval Postgraduate School, Monterey, CA, U. S. A.

Davis, K. P., and A. R. Anderson. 2003. Improving the Composability of Department of Defense Models and Simulations, RAND.

Genesereth, R. M. 1992. An agent-based Framework for Software Interoperability, In *Proceedings DARPA Software Technology Conference*. 17-23.

Hoffmann, M. 2004. Challenges of Model Interoperation in Military Simulations. *SIMULATION, Vol. 80*, 659-667.

Kuokka, D., and L. Harada. 1995. Matchmaking for Information Agents, In P*roceedings of the International Joint Conference on Artificial Intelligence* (IJCAI). 672-678, Montreal, Canada..

Larman, C. (2005). *Applying UML and Patterns* (3rd Edition). Prentice-Hall.

Mikalsen, T., and I. Rouvellou 2002. Transactional attitudes: Reliable composition of autonomous Web services, IBM Watson Research Center Technical Report.

Morris E., L. L., C. Meyers, P. Place, and D. Plakosh. 2004. *System of Systems Interoperability (SOSI) Final Report*.

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

Sycara, K., S. Widoff, M. Klusch, and J. Lu. 2002. Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace, Autonomous Agents and Multi-Agent Systems, 5, 173–203.

Tolk, A. 2004. Composable Mission Spaces and M&S Repositories-Applicability of Open Standards. In *Proceedings of the Spring 2004 Simulation Interoperability Workshop* (SIW). Available at `<http://www.sisostds.org>`.

Tolk, A., and Muguira, J.A. 2003. The Levels of Conceptual Interoperability Model (LCIM), Proceedings Fall Simulation Interoperability Workshop, IEEE CS Press.

Tosic, V., B. Pagurek,, B. Esfandiari, and K. Patel. 2001. On the Management of Compositions of Web Services. Workshop on Object-Oriented Web Services (OOPSLA 2001).

Turnitsa, C. 2005. Extending the Levels of Conceptual Interoperability Model. In *Proceedings of the Summer Computer Simulation Conference*. IEEE CS Press.

Page, E. H., R. Briggs, and J.A. Tufarolo. 2004. Toward a Family of Maturity Models for the Simulation Interconnection Problem. In *Proceedings of the Spring Simulation Interoperability Workshop*. IEEE CS Press.

White, L. E., and J. M. Pullen. 2003. Adapting Legacy Computational Software for XMSF. Simulation Interoperability Workshop (SIW).

Yilmaz, L. 2004. On the Need for Contextualized Introspective Simulation Models to Improve Reuse and Composability of Defense Simulations. Journal of Defense Modeling and Simulation, 1(3). pp. 135-145.

Yilmaz, L., and S. Paspuletti. 2006. Toward a Meta-Level Framework for Agent-supported Interoperation of Defense Simulations, Journal of Defense Modeling and Simulation (to appear).

AUTHOR BIOGRAPHIES

LEVENT YILMAZ is Assistant Professor of Computer Science and Software Engineering in the College of Engineering at Auburn University. He earned his Ph.D. and M.S. degrees from Virginia Tech. His research interests are on advancing the theory and methodology of simulation modeling, agent-directed simulation to explore dynamics of socio-technical systems, organizations, and human/team behavior, and education in simulation modeling. He is a member of ACM, IEEE Computer Society, Society for Computer Simulation International, and Upsilon Pi Epsilon. His Web address is: `<http://www.eng.auburn.edu/~yilmaz>`.

ANDREAS TOLK is Senior Research Scientist at the Virginia Modeling Analysis and Simulation Center (VMASC) of Old Dominion University, Norfolk, VA. He has over 12 years of international experience in the field of Applied Military Operations Research and Modeling and Simulation of and for Command and Control Systems. In addition to his research work, he gives lectures in the Modeling and Simulation program of ODU. His domain of expertise is the integration of M&S functionality into related application domains, such as C4ISR or web-based services, in particular based on open standards. His Web address is: `<http://myprofile.cos.com/atolk>`.