Artificial Intelligence in Modeling and Simulation [a,b,c]

[a]Bernard Zeigler, Arizona Center for Integrative Modeling and Simulation, University of Arizona,Tucson, Arizona, USA

[b] Alexander Muzy, CNRS, Università di Corsica, France

[c] Levent Yilmaz, Aurburn University, Alabama

Glossary

Behavior
The observable manifestation of an interaction with a system

DEVS
Discrete Event System Specification formalism describes models developed for simulation; applications include simulation based testing of collaborative services

Endomorphic Agents
Agents that contain models of themselves and/or of other endomorphic Agents.

Levels of Interoperability
Levels at which systems can interoperate such as syntactic, semantic and pragmatic. The higher the level, the more effective is information exchange among participants.

Levels of System Specification
Levels at which dynamic input/output systems can be described, known, or specified ranging from behavioral to structural

Metadata

Data that describes other data; a hierarchical concept in which metadata are a descriptive abstraction above the data it describes.

Model-based automation
Automation of system development and deployment that employs models or system specifications, such as DEVS, to derive artifacts.

Modeling and Simulation Ontology
The SES is interpreted as an ontology for the domain of hierarchical, modular simulation models specified with the DEVS formalism.

Net-Centric Environment
Network Centered, typically Internet-centered or web-centered information exchange medium

Ontology
Language that describes a state of the world from a particular conceptual view and usually pertains to a particular application domain

Pragmatic Frame
A means of characterizing the consumer's use of the information sent by a producer; formalized using the concept of processing network model

Pragmatics
Pragmatics is based on Speech Act Theory and focuses on elucidating the intent of the semantics constrained by a given context. Metadata tags to support pragmatics include Authority, Urgency/Consequences, Relationship, Tense and Completeness

Predicate logic
An expressive form of declarative language that can describe ontologies using symbols for individuals, operations, variables, functions with governing axioms and constraints

Schema
An advanced form of XML document definition, extends the DTD concept

Semantics
Semantics determines the content of messages in which information is packaged. The meaning of a message is the eventual outcome of the processing that it supports

Sensor
Device that can sense or detect some aspect of the world or some change in such an aspect

System Specification
Formalism for describing or specifying a system. There are levels of system specification ranging from behavior to structure.

Service Oriented Architecture
Web service architecture in which services are designed to be 1) accessed without knowledge of their internals through well-defined interfaces and 2) readily discoverable and composable.

Structure
The internal mechanism that produces the behavior of a system.

System Entity Structure
Ontological basis for modeling and simulation. Its pruned entity structures can describe both static data sets and dynamic simulation models.

Syntax
Prescribes the form of messages in which information is packaged.

UML
Unified Modeling Language is a software development language and environment that can be used for ontology development and has tools that map UML specifications into XML

XML
eXtensible Markup Language provides a syntax for document structures containing tagged information where tag definitions set up the basis for semantic interpretation.

1.Definition and the Subject and its Importance

This article discusses the role of Artificial Intelligence (AI) in Modeling and Simulation (M&S). AI is the field of computer science that attempts to construct computer systems that emulate human problem solving behavior with the goal of understanding human intelligence. M&S is a multidisciplinary field of systems engineering, software engineering, and computer science that seeks to develop robust methodologies for constructing computerized models with the goal of providing tools that can assist humans in all activities of the M&S enterprise. Although each of these disciplines has its core community there have been numerous intersections and cross-fertilizations between the two fields. From the perspective of this article, we view M&S as presenting some fundamental and very difficult problems whose solution may benefit from the concepts and techniques of AI.

2. Introduction

To state the M&S problems that may benefit from AI we first briefly review a system-theory based framework for M&S that provides a language and concepts to facilitate definitive problem statement. We then introduce some key problem areas: verification and validation, reuse and composability, and distributed simulation and systems of systems interoperability. After some further review of software and AI-related background, we go on to outline some areas of AI that have direct applicability to the just given problems in M&S. In order to provide a unifying theme for the problem and solutions, we then raise the question of whether all of M&S can be automated into an integrated autonomous artificial modeler/simulationist. We then proceed to explore an approach to developing such an intelligent agent and present a concrete means by which such an agent could engage in M&S. We close with consideration of an advanced feature that such an agent must have if it is fully emulate human capability – the ability, to a limited, but significant extent, to construct and employ models of its own "mind" as well of the 'minds" of other agents.

3. Review of System Theory and Framework for Modeling and Simulation

Hierarchy of System Specifications

Systems theory [1] deals with a hierarchy of system specifications which defines levels at which a system may be known or specified. Table 1 shows this Hierarchy of System Specifications (in simplified form, see [2] for full exposition).

Level	**Name**	**What we specify at this level**
4	Coupled Systems	System built up by several component systems which are coupled together
3	I/O System	System with state and state transitions to generate the behavior
2	I/O Function	Collection of input/output pairs constituting the allowed behavior partitioned according to the initial state the system is in when the input is applied
1	I/O Behavior	Collection of input/output pairs constituting the allowed behavior of the system from an external Black Box view
0	I/O Frame	Input and output variables and ports together with allowed values

Table 1: Hierarchy of System Specifications

- At level 0 we deal with the input and output interface of a system.
- At level 1 we deal with purely observational recordings of the behavior of a system. This is an I/O relation which consists of a set of pairs of input behaviors and associated output behaviors.

- At level 2 we have knowledge of the initial state when the input is applied. This allows partitioning the input/output pairs of level 1 into non-overlapping subsets, each subset associated with a different starting state.

- At level 3 the system is described by state space and state transition functions. The transition function describes the state-to-state transitions caused by the inputs and the outputs generated thereupon.

- At level 4 a system is specified by a set of components and a coupling structure. The components are systems on their own with their own state set and state transition functions. A coupling structure defines how those interact. A property of coupled system which is called "closure under coupling" guarantees that a coupled system at level 3 itself specifies a system. This property allows hierarchical construction of systems, i.e., that coupled systems can be used as components in larger coupled systems.

As we shall see in a moment, the system specification hierarchy provides a mathematical underpinning to define a framework for modeling and simulation. Each of the entities (e.g., real world, model, simulation, and experimental frame) will be described as a system known or specified at some level of specification. The essence of modeling and simulation lies in establishing relations between pairs of system descriptions. These relations pertain to the the validity of a system description at one level of specification relative to another system description at a different (higher, lower, or equal) level of specification.

Based on the arrangement of system levels as shown in Table 1, we distinguish between vertical and horizontal relations. A vertical relation is called an association mapping and takes a system at one level of specification and generates its counterpart at another level of specification. The downward motion in the structure-to-behavior direction, formally represents the process by which the behavior of a model is generated. This is relevant in simulation and testing when the model generates the behavior which then can be compared with the desired behavior.

The opposite upward mapping relates a system description at a lower level with one at a higher level of specification. While the downward association of specifications is straightforward, the upward association is much less so. This is because in the upward direction information is introduced while in the downward direction information is reduced. Many structures exhibit the same behavior and recovering a unique structure from a given behavior is not possible. The upward direction, however, is fundamental in the design process where a structure (system at level 3) has to be found which is capable to generate the desired behavior (system at Level 1).

3.1 Framework for Modeling and Simulation

The *Framework for M&S* as described in [Zeigler] establishes *entities* and their *relationships* that are central to the M&S enterprise (see Figure 1). The entities of the

Framework are*: source system, model, simulator,* and *experimental frame*; they are related by the *modeling* and the *simulation* relationships. Each entity is formally characterized as a system at an appropriate level of specification of a generic dynamic system.

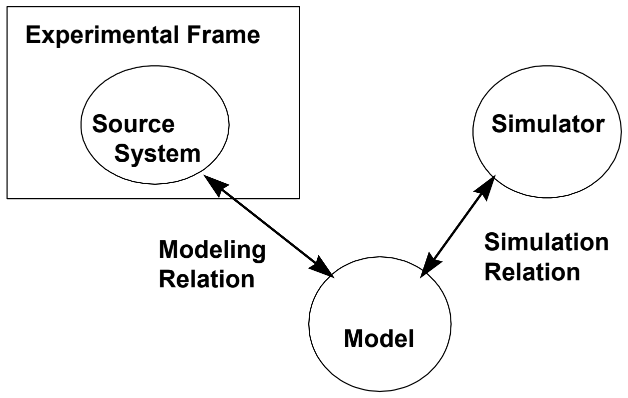

Figure 1. Framework Entities and Relationships

Source System

The source system is the real or virtual environment that we are interested in modeling. It is viewed as a source of observable data, in the form of time-indexed trajectories of variables. The data that has been gathered from observing or otherwise experimenting with a system is called the system behavior database. This data is viewed or acquired through experimental frames of interest to the model development and user. As we shall see, in the case of model validation, these data is the basis for comparison with data generated by a model. Thus these data must be sufficient in scope to enable reliable comparison as well accepted by both the model developer and the test agency as the basis for comparison. Data sources for this purpose might be measurement taken in prior experiments, mathematical representation of the measured data, or expert knowledge of the system behavior by accepted subject matter experts.

Experimental Frame

An experimental frame is a specification of the conditions under which the system is observed or experimented with [3]. An experimental frame is the operational formulation of the objectives that motivate a M&S project. A frame is realized as a system that interacts with the system of interest to obtain the data of interest under specified conditions.

An experimental frame specification consists of 4 major subsections:

- *input stimuli*: specification of the class of admissible input time-dependent stimuli. This is the class from which individual samples will be drawn and injected into the model or system under test for particular experiments.
- *control*: specification of the conditions under which an the model or system will be initialized, continued under examination, and terminated.
- *metrics*: specification of the data summarization functions and the measures to be employed to provide quantitative or qualitative measures of the input/output behavior of the model. Examples of such metrics are performance indices, goodness of fit criteria, and error accuracy bound.
- *analysis*: specification of means by which the results of data collection in the frame will be analyzed to arrived at final conclusions. The data collected in a frame consists of pairs of input/output time functions.

When an experimental frame is realized as a system to interact with the model or system under test the specifications become components of the driving system. For example, a generator of output time functions implements the class of input stimuli.

An experimental frame is the operational formulation of the objectives that motivate a modeling and simulation project. Many experimental frames can be formulated for the same system (both source system and model) and the same experimental frame may apply to many systems. Why would we want to define many frames for the same system? Or apply the same frame to many systems? For the same reason that we might have different objectives in modeling the same system, or have the same objective in modeling different systems. There are two equally valid views of an experimental frame. One, views a frame as a definition of the type of data elements that will go into the database. The second views a frame as a system that interacts with the system of interest to obtain the data of interest under specified conditions. In this view, the frame is characterized by its implementation as a measurement system or observer. In this implementation, a frame typically has three types of components (as shown in Figure 3): *generator*, that generates input segments to the system; *acceptor* that monitors an experiment to see the desired experimental conditions are met; and *transducer* that observes and analyzes the system output segments.

An experimental frame can be viewed as a system that interacts with the system under test (SUT) to obtain the data of interest under specified conditions. In this view, a frame typically has three types of components (as shown in Figure 2a): *generator* that generates input segments to the system; *acceptor* that monitors an experiment to see the desired experimental conditions are met; and *transducer* that observes and analyzes the system output segments.

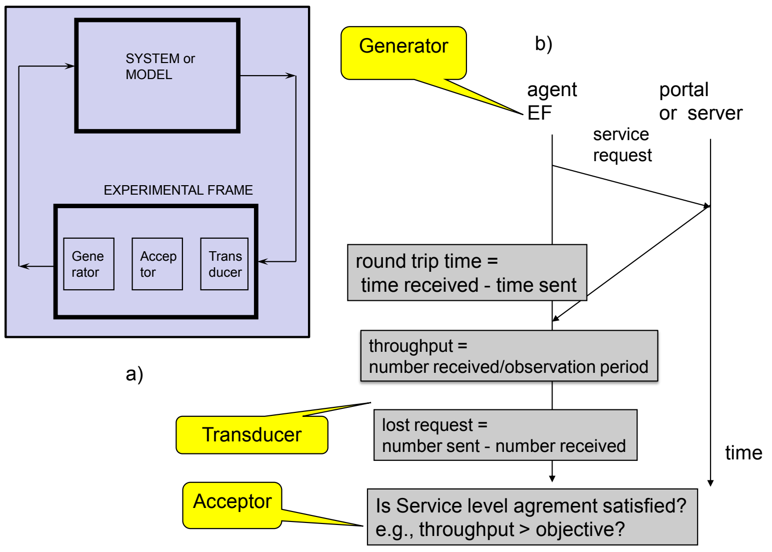
SYSTEM or MODEL
EXPERIMENTAL FRAME
Gene rator
Accep tor
Trans ducer
a)
Generator
b)
agent EF
portal or server
service request
round trip time = time received - time sent
throughput = number received/observation period
Transducer
lost request = number sent - number received
time
Acceptor
Is Service level agrement satisfied? e.g., throughput > objective?

Figure 2: Experimental Frame and Components

Figure 2b) illustrates a simple, but ubiquitous, pattern for experimental frames that measure typical job processing performance metrics, such as relate to round trip time and throughput. Illustrated in the web context, a generator produces service request messages at a given rate. The time that has elapsed between sending of a request and its return from a server is the round trip time. A transducer notes the departures and arrivals of requests allowing it to compute the average round trip time and other related statistics, as well as the throughput and unsatisfied (or lost) requests. An acceptor notes whether performance achieves the developer's objectives, for example, whether the throughput exceeds the desired level and/or whether say 99% of the round trip times are below a given threshold.

Objectives for modeling relate to the role of the model in systems design, management or control. Experimental frames translate the objectives into more precise experimentation conditions for the source system or its models. We can distinguish between objectives concerning those for verification and validation of a) models and b) systems. In the case of models, experimental frames translate the objectives into more precise experimentation conditions for the source system and/or its models. A model under test is expected to be valid for the source system in each such frame. Having stated the objectives, there is presumably a best level of resolution to answer these questions. The more demanding the questions, the greater the resolution likely to be needed to answer them. Thus, the choice of appropriate levels of abstraction also hinges on the objectives and their experimental frame counterparts.

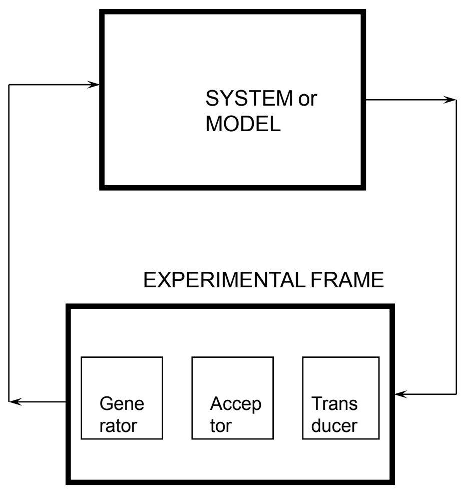

Figure 3 Experimental Frame and its Components.

In the case of objectives for verification and validation of systems, we need to be given, or be able to formulate, the requirements for the behavior of the system at the IO behavior level. The experimental frame then is formulated to translate these requirements into a set of possible experiments to test whether the system actually performs its required behavior. In addition we can formulate measures of the effectiveness (MOE) of a system in accomplishing its goals. We call such measures, *outcome* measures. In order to compute such measures, the system must expose relevant variables, we'll call *output* variables, whose values can be observed during execution runs of the system.

Model

A model is a system specification, such as a set of instructions, rules, equations, or constraints for generating input/output behavior. Models may be expressed in a variety of formalisms that may be understood as means for specifying subclasses of dynamic systems. The Discrete Event System Specification (DEVS) formalism delineates the subclass of discrete event systems and it can also represent the systems specified within traditional formalisms such as differential (continuous) and difference (discrete time) equations [4]. In DEVS , as in systems theory, a model can be *atomic*, i.e., not further decomposed, or *coupled*, in which cases it consists of a components that are coupled or interconnected together.

Simulator

A simulator is any computation system (such as a single processor, or a processor network, or more abstractly an algorithm), capable of executing a model to generate its behavior.
The more general purpose a simulator is the greater the extent to which it can be configured to execute a variety of model types. In order of increasing capability, simulators can be:

- Dedicated to a particular model or small class of similar models
- Capable of accepting all (practical) models from a wide class, such as an application domain (e.g., communication systems)
- Restricted to models expressed in a particular modeling formalism, such as continuous differential equation models
- Capable of accepting multi-formalism models (having components from several formalism classes, such as continuous and discrete event).

A simulator can take many forms such as on a single computer or multiple computers executing on a network.

4. Fundamental Problems in M&S

We have now reviewed a system-theory based framework for M&S that provides a language and concepts in which to formulate key problems in M&S . Next on our agenda is to discuss such problem areas: verification and validation, reuse and composability, and distributed simulation and systems of systems interoperability. These are challenging, and heretofore, unsolved problems at the core of the M&S enterprise.

Validation and Verification

The basic concepts of verification and validation (V&V) have been described in different settings, levels of details, and points of views and are still evolving. These concepts have been studied by a variety of scientific and engineering disciplines and various flavors of validation and verification concepts and techniques have emerged from a modeling and simulation perspective. Within the modeling and simulation community, a variety of methodologies for V&V have been suggested in the literature [5, 6, 7]. A categorization of 77 verification, validation and testing techniques along with 15 principles has been offered to guide the application of these techniques [8]. However, these methods vary extensively – e.g., alpha testing, induction, cause and effect graphing, inference, predicate calculus, proof of correctness, and user interface testing and are only loosely related to one another. Therefore such a categorization can only serve as an informal guideline for the development of a process for V&V of models and systems.
Validation and verification concepts are themselves founded on more primitive concepts such as system specifications and homomorphism as discussed in the framework of M&S [2]. In this framework, the entities *system, experimental frame, model, simulator* take on real importance only when properly related to each other. For example, we build a model of a particular system for some objective only some models, and not others, are suitable.

Thus, it is critical to the success of a simulation modeling effort that certain relationships hold. Two of the most important are *validity* and *simulator correctness*.

The basic modeling relation, *validity,* refers to the relation between a model, a source system and an *experimental frame*. The most basic concept, *replicative validity*, is affirmed if, for all the experiments possible within the experimental frame, the behavior of the model and system agree within acceptable tolerance. The term *accuracy* is often used in place of validity. Another term, *fidelity*, is often used for a combination of both validity and detail. Thus, a high fidelity model may refer to a model that is both highly detailed and valid (in some understood experimental frame). However when used this way, the assumption seems to be that high detail alone is needed for high fidelity, as if validity is a necessary consequence of high detail. In fact, it is possible to have a very detailed model that is nevertheless very much in error, simply because some of the highly resolved components function in a different manner than their real system counterparts.

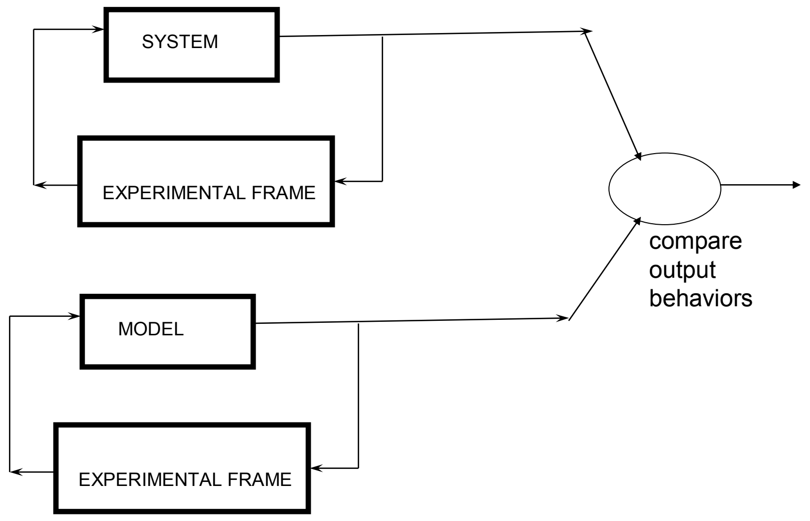

Figure 4. Basic Approach to Model Validation

The basic approach to model validation is comparison of the behavior generated by a model and the source system it represents within a given experimental frame. The basis for comparison serves as the reference against which the accuracy of the model is measured.

The basic *simulation* relation, simulator correctness, is a relation between a *simulator* and a *model.* A *simulator correctly simulates a model* if it is guaranteed to faithfully generate the model's output behavior given its initial state and its input trajectory. In practice, as suggested above, simulators are constructed to execute not just one model but also a family of possible models. For example, a network simulator provides both a simulator and a class of network models it can simulate. In such cases, we must establish that a simulator will correctly execute the particular class of models it claims to support. Conceptually, the approach to testing for such execution, illustrated in Figure 5, is to perform a number of test cases in which the same model is provided to the simulator under test and to a "gold standard simulator" which is known to correctly simulate the model. Of course such test case models must lie within the class supported by the simulated under test as well as presented in the form that it expects to receive them. Comparison of the output behaviors in the same manner as with model validation is then employed to check the agreement between the two simulators.

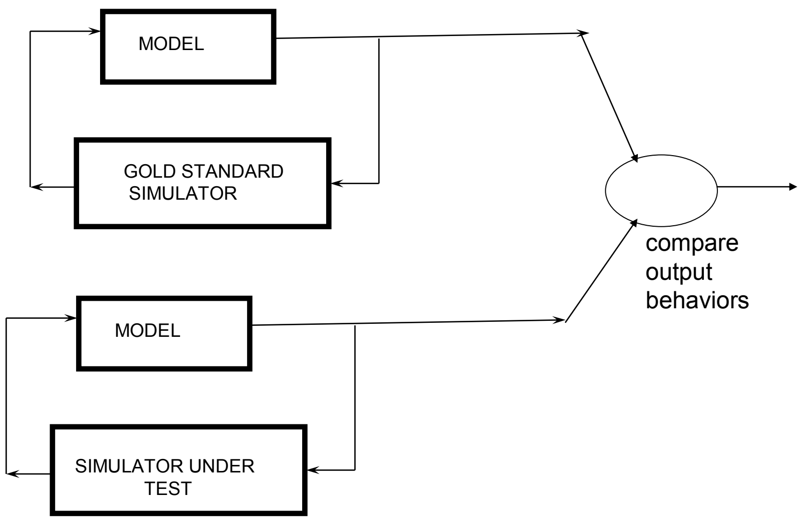

Figure 5. Basic Approach to Simulator Verification

If the specifications of both the simulator and the model are available in separated form where each can be accessed independently, it may be possible to prove correctness mathematically.

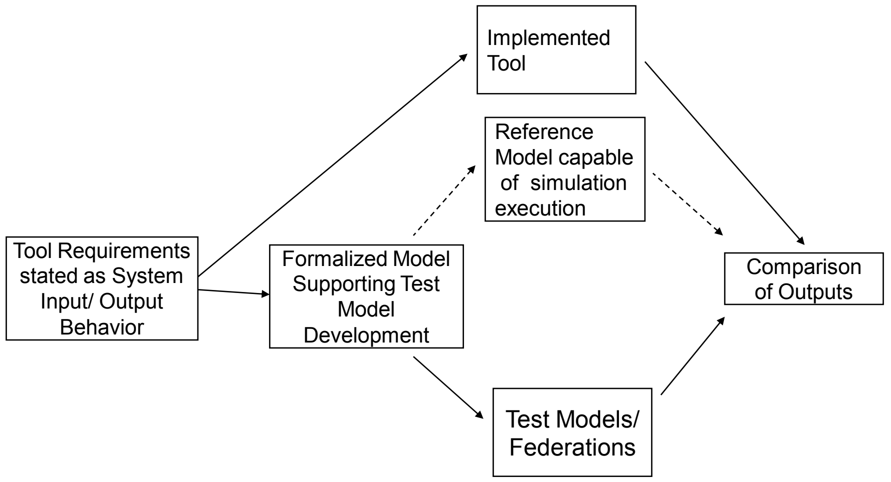
Implemented Tool
Reference Model capable of simulation execution
Tool Requirements stated as System Input/ Output Behavior
Formalized Model Supporting Test Model Development
Comparison of Outputs
Test Models/ Federations

Figure 6 Basic Approach to System Validation

The case of system validation is illustrated in Figure 6. Here the system is considered as a hardware and/or software implementation to be validated against requirements for its input/output behavior. The goal is to develop test models that can stimulate the implemented system with inputs and can observe its outputs to compare them with those required by the behavior requirements. Also shown is a dotted path in which a reference model is constructed that is capable of simulation execution. Construction of such a reference model is more difficult to develop than the test models since it requires not only knowing in advance what output to test for, but to actually to generate such an output. Although such a reference model is not required, it may be desirable in situations in which the extra cost of development is justified by the additional range of tests that might be possible and the consequential increased coverage this may provide.

Model Reuse and Composability

Model reuse and composability are two sides of the same coin – it is patently desirable to reuse models, the fruits of earlier or others work. However, typically such models will become components in a larger composite model and must be able to interact meaningfully with them. While software development disciplines are successfully applying component-based approach to build software systems, the additional systems dynamics involved in simulation models has resisted straight forward reuse and composition approaches. A model is only reusable to the extent that its original dynamic systems assumptions are consistent with the constraints of the new simulation application. Consequently, without contextual information to guide selection and refactoring, a model may not be reused to advantage within a new experimental frame. Davis and Anderson [9] argue that to foster such reuse, model representation methods should distinguish, and separately specify, the model, simulator, and the experimental frame. However, Yilmaz and Oren [10] pointed out that more contextual information is needed beyond the information provided by the set of experimental frames to which a model is applicable [11], namely, the characterization of the context in which the model was constructed. These authors extended the basic model-simulator-experimental frame perspective to emphasize the role of context in reuse They make a sharp distinction between the objective context within which a simulation model is originally defined and the intentional context in which the model is being qualified for reuse. They extend the system theoretic levels of specification discussed earlier to define certain behavioral model dependency relations needed to formalize conceptual, realization, and experimental aspects of context.

As the scope of simulation applications grows, it is increasingly the case that more than one modeling paradigm is needed to adequately express the dynamics of the different components. For systems composed of models with dynamics that are intrinsically heterogeneous, it is crucial to use multiple modeling formalisms to describe them. However, combining different model types poses a variety of challenges [12, 9, 13]. Sarjoughian [14], introduced an approach to multi-formalism modeling that employs an interfacing mechanism called a Knowledge Interchange Broker to compose model components expressed in diverse formalisms. The KIB supports translation from the semantics of one formalism into that of a second to ensure coordinated and correct execution simulation algorithms of distinct modeling formalisms.

Distributed Simulation and System of Systems interoperability

The problems of model reuse and composability manifest themselves strongly in the context of distributed simulation where the objective is to enable existing geographically dispersed simulators to meaningfully interact, or federate, together. We briefly review experience with interoperability in the distributed simulation context and a linguistically based approach to the System of Systems (SoS) interoperability problem [15]. Sage and Cuppan [16] drew the parallel between viewing the construction of SoS as federation of systems and the federation that is supported by the High Level Architecture (HLA), an IEEE standard fostered by the DoD to enable composition of simulations [17,18].

HLA is a network middleware layer that supports message exchanges among simulations, called federates, in a neutral format. However, experience with HLA has been disappointing and forced acknowledging the difference between technical interoperability and substantive interoperability [19]. The first only enables heterogeneous simulations to exchange data but does not guarantee the second, which is the desired outcome of exchanging meaningful data, namely, that coherent interaction among federates takes place. Tolk and Muguirra [20] introduced the Levels of Conceptual Interoperability Model (LCIM) which identified seven levels of interoperability among participating systems. These levels can be viewed as a refinement of the *operational* interoperability type which is one of three defined by [15]. The operational type concerns linkages between systems in their interactions with one another, the environment, and with users. The other types apply to the context in which systems are constructed and acquired. They are *constructive* –

relating to linkages between organizations responsible for system construction and *programmatic* – linkages between program offices to manage system acquisition.

5. AI-Related Software Background

To proceed to the discussion of the role of AI in address key problems in M&S, we need to provide some further software and AI-related background. We offer a brief historical account of object-orientation and agent-based systems as a springboard to discuss the upcoming concepts of object frameworks, ontologies and endomorphic agents.

Object-Orientation and Agent-Based Systems

Many of the software technology advances of the last 30 years have been initiated from the field of M&S. Objects, as code modules with both structure and behavior, were first introduced in the SIMULA simulation language [21]. Objects blossomed in various directions and became institutionalized in the widely adopted programming language C++ and later in the infrastructure for the web in the form of Java [22] and its variants. The freedom from straight-line procedural programming that object-orientation championed was taken up in AI in two directions: various forms of knowledge representation and of autonomy. Rule-based systems aggregate modular if-then logic elements – the rules – that can be activated in some form of causal sequence (inference chains) by an execution engine [23]. In their passive state, rules represent static discrete pieces of inferential logic, called declarative knowledge. However, when activated, a rule influences the state of the computation and the activation of subsequent rules, providing the system a dynamic or procedural, knowledge characteristic as well. Frame-based systems further expanded knowledge representation flexibility and inferencing capability by supporting slots and constraints on their values – the frames – as well as their taxonomies based on generalization/specialization relationships [24]. Convergence with object-orientation became apparent in that frames could be identified as objects and their taxonomic organization could be identified with classes within object-style organizations that are based on sub-class hierarchies.

On the other hand, the modular nature of objects together with their behavior and interaction with other objects, led to the concept of agents which embodied increased autonomy and self-determination [25]. Agents represent individual threads of computation and are typically deployable in distributed form over computer networks where they interact with local environments and communicate/coordinate with each other. A wide variety of agent types exists in large part determined by the variety and sophistication of their processing capacities – ranging from agents that simply gather information on packet traffic in a network to logic-based entities with elaborate reasoning capacity and authority to make decisions (employing knowledge representations just mentioned.) The step from agents to their aggregates is natural, thus leading to the concept of multi-agent systems or societies of agents, especially in the realm of modeling and simulation [26].

To explore the role of AI in M&S at the present, we project the just-given historical background to the concurrent concepts of object frameworks, ontologies and endomorphic agents. The Unified Modeling Language (UML) is gaining a strong foothold as the defacto standard for object-based software development. Starting as a diagrammatical means of software system representation, it has evolved to a formally specified language in which the fundamental properties of objects are abstracted and organized [27 , 28]. Ontologies are models of the world relating to specific aspects or applications that are typically represented in frame-based languages and form the knowledge components for logical agents on the Semantic Web [29]. A convergence is underway that re-enforces the commonality of the object-based origins of AI and software engineering. UML is being extended to incorporate ontology representations so that software systems in general will have more explicit models of their domains of operation. As we shall soon see, endomorphic agents refer to agents that include abstractions of their own structure and behavior within their ontologies of the world [30].

The M&S Framework within Unified Modeling Language (UML)

With object-orientation as unified by UML and some background in agent-based systems, we are in position to discuss the computational realization of the M&S framework discussed earlier. The computational framework is based on the Discrete Event System Specification (DEVS) formalism and implemented in various object oriented environments. Using UML we can represent the framework as a set of classes and relations as illustrated in Figures 8 and 9. Various software implementations of DEVS support different subsets of the classes and relations In particular, we mention a recent implementation of DEVS within a Service Oriented Architecture (SOA) environment called DEVS/SOA [31,32]. This implementation exploits some of the benefits afforded by the web environment mentioned earlier and provides a context for consideration of the primary target of our discussion, comprehensive automation of the M&S enterprise.

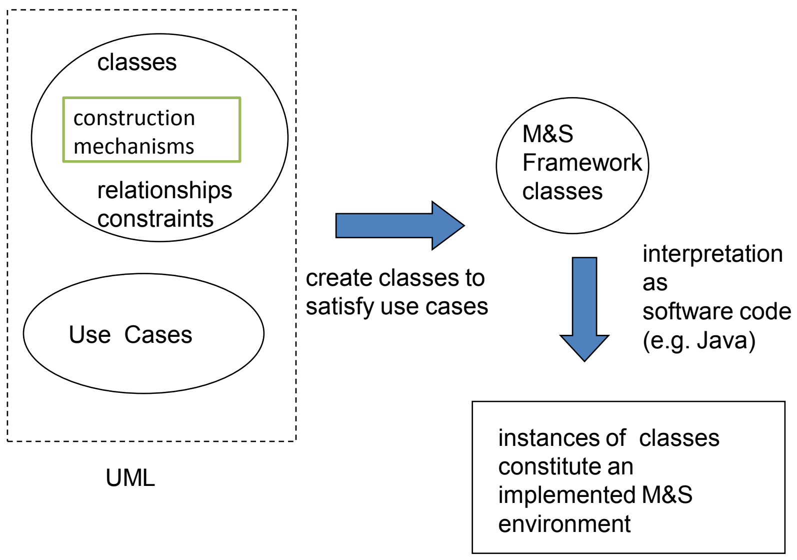
classes
construction mechanisms
relationships constraints
Use Cases
UML
create classes to satisfy use cases
M&S Framework classes
interpretation as software code (e.g. Java)
instances of classes constitute an implemented M&S environment

Figure 7: M&S Framework formulated within UML

We use one of the UML constructs, the use case diagram, to depict the various capabilities that would be involved in automating all or parts of the M&S enterprise. Use

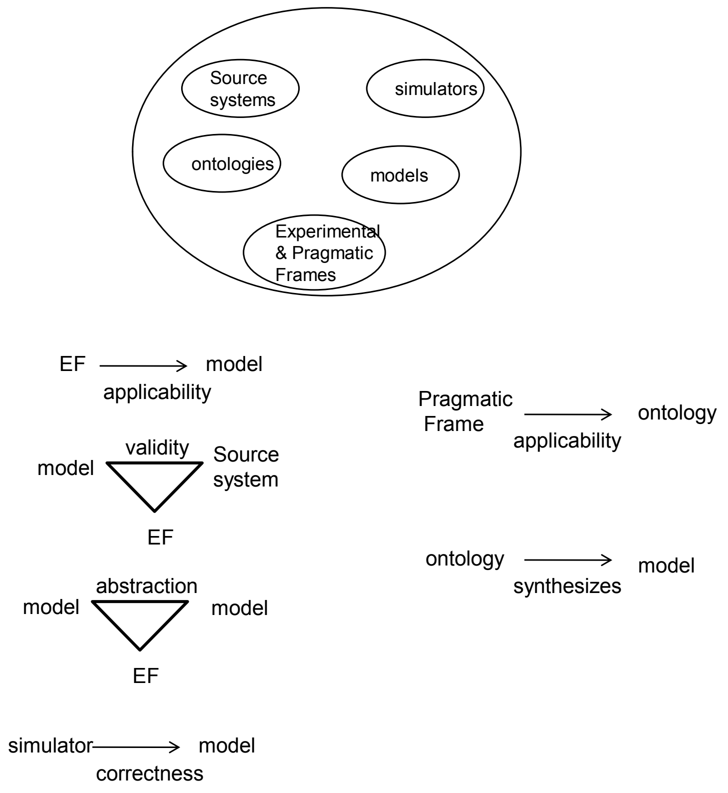

Figure 8: M&S Framework Classes and Relations in a UML representation

cases are represented by ovals that connect to at least one actor (stick figure) and to other use cases through “includes” relations, shown as dotted arrows. For example, a sensor (actor) collects data (use case) which includes storage of data (use case). A memory actor stores and retrieves models which include storage and retrieval (respectively) of data. Constructing models includes retrieving stored data within an experimental frame. Validating models includes retrieving models from memory as components and simulating the composite model to generate data within an experimental frame. The emulator-simulator actor does the simulating to execute the model so that its generated behavior can be matched against the stored data in the experimental frame. The objectives of the human modeler drive the model evaluator and hence the choice of experimental frames to consider as well as models to validate. Models can be used in (at least) two time frames[33]. In the long term, they support planning of actions to be taken in the future In the short term, models support execution and control in real-time of previously planned actions .

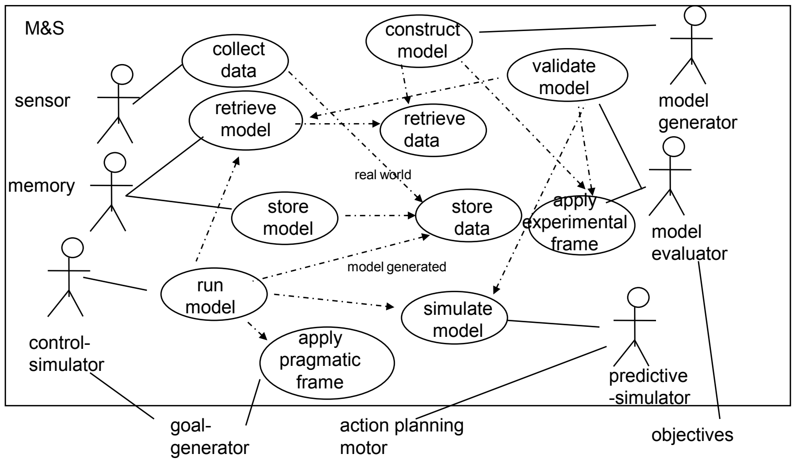

Figure 9: UML Use Case Formulation of the Overall M&S Enterprise

6. AI Methods in Fundamental Problems of M&S

The enterprise of modeling and simulation is characterized by activities such as model, simulator and experimental frame creation, construction, reuse, composition, verification and validation. We have seen that valid model construction requires significant expertise in all the components of the M&S enterprise, e.g., modeling formalisms, simulation methods, and domain understanding and knowledge. Needless to say, few people can bring all such elements to the table, and this situation creates a significant bottleneck to progress in such projects. Among the contributing factors are lack of trained personnel that must be brought in, expense of such high capability experts, and the time needed to construct models to the resolution required for most objectives. This section introduces some AI-related technologies that can ameliorate this situation: Service Oriented Architecture (SOA) and Semantic Web, ontologies, constrained natural language capabilities, and genetic algorithms. Subsequently we will consider these as components in unified, comprehensive, and autonomous automation of M&S.

Service Oriented Architecture (SOA) and Semantic Web

On the World Wide Web, a Service Oriented Architecture (SOA) is a market place of open and discoverable web-services incorporating, as they mature, Semantic Web technologies [34]. The eXtensible Markup Language (XML) is the standard format for encoding data sets and there are standards for sending and receiving XML [35]. Unfortunately, the problem just starts at this level. There are myriad ways, or Schemata, to encode data into XML and a good number of such Schemata have already been developed. More often than not, they are different in detail when applied to the same domains. What explains this incompatibility?

In a Service Oriented Architecture, the producer sends messages containing XML documents generated in accordance with a schema. The consumer receives and interprets these messages using the same schema in which they were sent. Such a message encodes a world state description (or changes in it) that is a member of a set delineated by an ontology. The ontology takes into account the pragmatic frame, i.e., a description of how the information will be used in downstream processing. In a SOA environment, data dissemination may be dominated by "user pull of data", incremental transmission, discovery using metadata, and automated retrieval of data to meet user pragmatic frame specifications. This is the SOA concept of data-centered, interface-driven, loose coupling between producers and consumers. The SOA concept requires the development of platform-independent, community-accepted, standards that allow raw data to be syntactically packaged into XML and accompanied by metadata that describes the semantic and pragmatic information needed to effectively process the data into increasingly higher-value products downstream.

Ontologies

Semantic Web researchers typically seek to develop intelligent agents that can draw logical inferences from diverse, possibly contradictory, ontologies such as a web search might discover. Semantic Web research has lead to a focus on ontologies [34]. These are logical languages that provide a common vocabulary of terms and axiomatic relations among them for a subject area. In contrast, the newly emerging area of ontology integration assumes that human understanding and collaboration will not be replaced by intelligent agents. Therefore the goal is to create concepts and tools to help people develop practical solutions to incompatibility problems that impede "effective" exchange of data and ways of testing that such solutions have been correctly implemented.

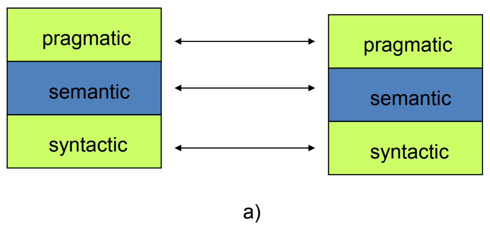

Figure 10: Interoperability levels in distributed simulation

As illustrated in Figure 10, interoperability of systems can be considered at three linguistically-inspired levels: *syntactic, semantic,* and *pragmatic.* The levels are summarized in Table 2. More detail is provided in [36].

Linguistic Level	**A collaboration of systems or services interoperates at this level if:**	**Examples**
Pragmatic – how information in messages is used	The receiver reacts to the message in a manner that the sender intends	An order from a commander is obeyed by the troops in the field as the commander intended. A necessary condition is that the information arrives in a timely manner and that its meaning has been preserved (semantic interoperability)
Semantic – shared understanding of meaning of messages	The receiver assigns the same meaning as the sender did to the message.	An order from a commander to multi-national participants in a coalition operation is understood in a common manner despite translation into different languages. A necessary condition is that the information can be unequivocally extracted from the data (syntactic interoperability)

Syntactic – common rules governing composition and transmitting of messages	The consumer is able to receive and parse the sender's message	A common network protocol (e.g. IPv4) is employed ensuring that all nodes on the network can send and receive data bit arrays adhering to a prescribed format.

Table 2 Linguistic levels

Constrained Natural Language

Model development can be substantially aided by enabling users to specify modeling constructs using some form of constrained natural language [37]. The goal is to overcome modeling complexity by letting users with limited or nonexistent formal modeling or programming background convey essential information using natural language, a form of expression that is natural and intuitive. Practicality demands constraining the actual expressions that can be used so that the linguistic processing is tractable and the input can be interpreted unambiguously. Some techniques allow the user to narrow down essential components for model construction. Their goal is to reduce ambiguity between the user's requirements and essential model construction components. A natural language interface allows model specification in terms of a verb phrase consisting of a verb, noun, and modifier, for example "build car quickly." Conceptual realization of a model from a verb phrase ties in closely with Checkland's [38] insight that an appropriate verb should be used to express the root definition, or core purpose, of a system. The main barrier between many people and existing modeling software is their lack of computer literacy and this provides a incentive to develop natural language interfaces as a means of bridging this gap. Natural language expression could create modelers out of people who think semantically, but do not have the requisite computer skills to express these ideas. A semantic representation frees the user to explore the system on the familiar grounds of natural language and opens the way for brain storming, innovation and testing of models before they leave the drawing board.

Genetic Algorithms

The genetic algorithm is a subset of evolutionary algorithms that model biological processes to search in highly complex spaces. A genetic algorithm (GA) allows a population composed of many individuals to evolve under specified selection rules to a state that maximizes the "fitness". The theory was developed by John Holland [39] and popularized by Goldberg who was able to solve a difficult problem involving the control of gas pipeline transmission for his dissertation [40]. Numerous applications of GAs have since been chronicled [41, 42]. Recently, GAs have been applied to cutting edge problems in automated construction of simulation models, as discussed below [43].

7. Automation of M&S

We are now ready to suggest a unifying theme for the problems in M&S and possible AI-based solutions, by raising the question of whether all of M&S can be automated into an integrated autonomous artificial modeler/simulationist. First, we provide some background needed to explore an approach to developing such an intelligent agent based on the System Entity Structure/Model Base framework, a hybrid methodology that combines elements of AI and M&S.

System Entity Structure

The System Entity Structure (SES) concepts were first presented in [44]. They were subsequently extended and implemented in a knowledge based design environment [45]. Application to model base management originated with [46] Subsequent formalizations and implementations were developed in [47-51]. Applications to various domains are given in [52].

A System Entity Structure is a knowledge representation formalism in which focuses on certain elements and relationships that relate to M&S. Entities represent things that exist in the real world or sometimes in an imagined world. Aspects represent ways of decomposing things into more fine grained ones. Multi-aspects are aspects for which the components are all of one kind. Specializations represent categories or families of specific forms that a thing can assume. provides the means to represent a family of models as a labeled tree. Two of its key features are support for decomposition and specialization. The former allows decomposing a large system into smaller systems. The latter supports representation of alternative choices. Specialization enables representing a generic model (e.g., a computer display model) as one of its specialized variations (e.g., a flat panel display or a CRT display.) Based on SES axiomatic specifications, a family of models (design-space) can be represented and further automatically pruned to generate a simulation model. Such models can be systematically studied and experimented based alternative design choices. An important, salient feature of SES is its ability to represent models not only in terms of their decomposition and specialization, but also their aspects. The SES represents alternative decompositions via aspects. The system entity structure (SES) formalism provides an operational language for specifying such hierarchical structures. An SES is a structural knowledge representation scheme that systematically organizes a family of possible structures of a system. Such a family characterizes decomposition, coupling, and taxonomic relationships among entities. An *entity* represents a real world object. The decomposition of an entity concerns how it may be broken down into sub-entities. In addition, coupling specifications tell how sub-entities may be coupled together to reconstitute the entity and associated with an aspect. The *taxonomic* relationship concerns admissible variants of an entity. The SES/Model-Base framework [52] is a powerful means to support the plan-generate-evaluate paradigm in systems design. Within the framework, entity structures organize models in model base. Thus, modeling activity within the framework consists of three sub-activities: specification of model composition structure, specification of model behavior, and synthesis of a simulation model.

The SES is governed by an axiomatic framework in which entities alternate with the other items. For example, a thing is made up of parts; therefore, its entity representation has a corresponding aspect which, in turn, has entities representing the parts. A System Entity Structure specifies a family of hierarchical, modular simulation models, each of which corresponds to a complete pruning of the SES. Thus, the SES formalism can be viewed as an ontology with the set of all simulation models as its domain of discourse. The mapping from SES to the Systems formalism, particularly to the DEVS formalism, is discussed in [36]. We note that simulation models include both static and dynamic elements in any application domain, hence represent an advanced form of ontology framework.

8. SES/Model Base Architecture for an Automated Modeler/Simulationist

In this section, we raise the challenge of creating a fully automated modeler/simulationist that can autonomously carry out all the separate functions identified in the M&S framework as well as the high level management of these functions that is currently under exclusively human control. Recall the use case diagrams in Figure 9 that depicts the various capabilities that would need to be involved in realizing a completely automated modeler/simulationist. To link up with the primary modules of mind, we assign model construction to the belief generator – interpreting beliefs as models [54]. Motivations outside the M&S component drive the belief evaluator and hence the choice of experimental frames to consider as well as models to validate. External desire generators stimulate the imaginer/envisioner to run models to make predictions within pragmatic frames and assist in action planning.

The use case diagram of Figure 9, is itself a model of how modelling and simulation activities may be carried out within human minds. We need not be committed to particular details at this early stage, but will assume that such a model can be refined to provide a useful representation of human mental activity from the perspective of M&S. This provides the basis for examining how such an artificially intelligent modeler/simulationist might work and considering the requirements for comprehensive automation of M&S.

The SES/MB methodology, introduced earlier, provides a basis for formulating a conceptual architecture for automating all of the M&S activities depicted earlier in Figure 9 into an integrated system. As illustrated in Figure 11, the dichotomy into real-time use of models for acting in the real world and the longer term development of models that can be employed in such real-time activities is manifested in the distinction between passive and active models. Passive models are stored in a repository that can be likened to long term memory. Such models under go the life-cycle mentioned earlier in which they are validated and employed within experimental frames of interest for long term forecasting or decision-making. However, in addition to this quite standard concept of operation, there is an input pathway from the short term, or working, memory in which models are executed in real-time. Such execution, in real-world environments, often results in deficiencies, which provide impetus and requirements for instigating new model construction. Whereas long term model development and application objectives are characterized by experimental frames, the short term execution objectives are characterized by pragmatic frames. As discussed in [36], a pragmatic frame provides a means of characterizing the use for which an executable model is being sought. Such models must be simple enough to execute within, usually, within stringent real-time deadlines.

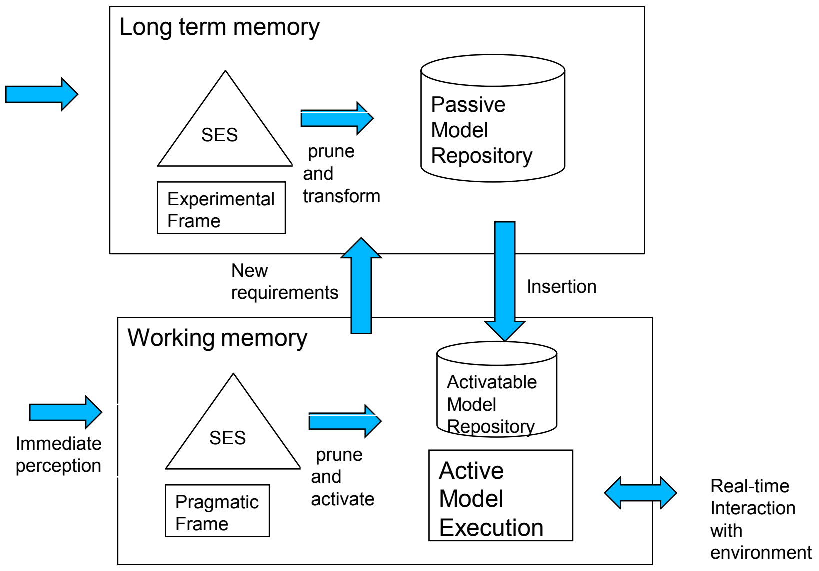

Figure 11. SES/Model Base Architecture for Automated M&S

The M&S Framework within Mind Architecture

An influential formulation of recent work relating to mind and brain [53] views mind as the behavior of the brain, where mind is characterized by a massively modular architecture. This means that mind is, composed a large number of modules, each responsible for different functions, and each largely independent and sparsely connected with others. Evolution is assumed to favor such differentiation and specialization since under suitably weakly interactive environments they are less redundant and more efficient in consuming space and time resources. Indeed, this formulation is reminiscent of the class of problems characterized by Systems of Systems (SoS) in which the attempt is made to integrated existing systems, originally built to perform specific functions, in to a more comprehensive and multifunctional system. As discussed in [15], the components of each system can be viewed as communicating with each other within a common ontology, or model of the world that is tuned to the smooth functioning of the organization. However, such ontologies may well be mismatched to support integration at the systems level. Despite working on the results of a long history of pre-human evolution, the fact that consciousness seems to provide a unified and undifferentiated picture of mind, suggests that human evolution has to a large extent solved the SoS integration problem.

The Activity Paradigm for Automated M&S

At least for the initial part of its life, a modeling agent needs to work on a "first order" assumption about its environment, namely, that it can exploit only semantics-free properties [39]. Regularities, such as periodic behaviors, and stimulus-response associations, are one source of such semantics-free properties. In this section, we will focus on a fundamental property of systems, such as the brain's neural network, that have a large number of components. This distribution of activity of such a system over space and time provides a rich and semantics-free substrate from which models can be generated. Proposing structures and algorithms to track and replicate this activity should support automated modeling and simulation of patterns of reality. The goal of the activity paradigm is to extract mechanisms from natural phenomena and behaviors to automate and guide the M&S process.

The brain offers a quintessential illustration of activity and its potential use to construct models. Figure 12 describes brain electrical activities [54]. Positron emission tomography (PET) is used to record electrical activity inside the brain. The PET method scans show what happens inside the brain when resting and when stimulated by words and music. The red areas indicate high brain activities. Language and music produce responses in opposite sides of the brain (showing the sub-system specializations). There are many levels of activity (ranging from low to high.)

There is a strong link between modularity and the applicability of activity measurement as a useful concept. Indeed, modules represent loci for activity – a distribution of activity would not be discernable over a network were there no modules that could be observed to be in different states of activity. As just illustrated, neuroscientists are exploiting this activity paradigm to associate brain areas with functions and to gain insight into areas that are active or inactive over different, but related, functions, such as language and music processing. We can generalize this approach as a paradigm for an automated modeling agent.

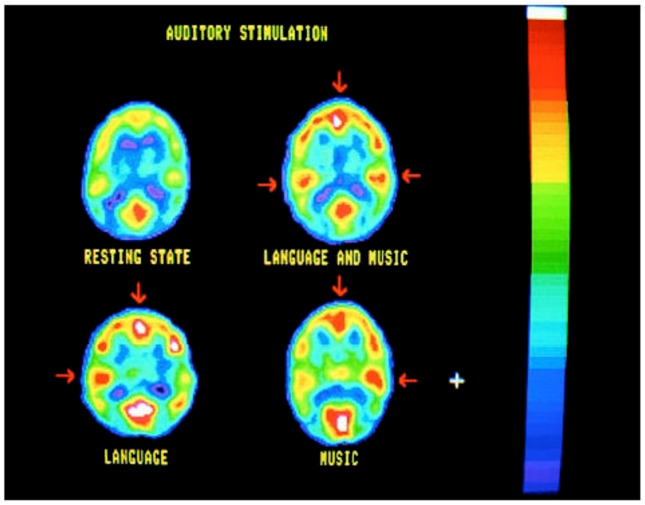

Figure 12. Brain module activities

Component, activity and discrete-event abstractions

Figure 13 depicts a brain description through components and activities. First, the modeler considers one brain activity (e.g., listening music.) This first level activity corresponds to simulation components at a lower level. At this level, activity of the components can be considered to be only on (grey boxes) or off (white boxes.) At lower levels, structure and behaviors can be decomposed. The structure of the higher component level is detailed at lower levels (*e.g.*, to the neuronal network.) The behavior is also detailed through activity and discrete-event abstraction [55]. At the finest levels, activity of components can be detailed.

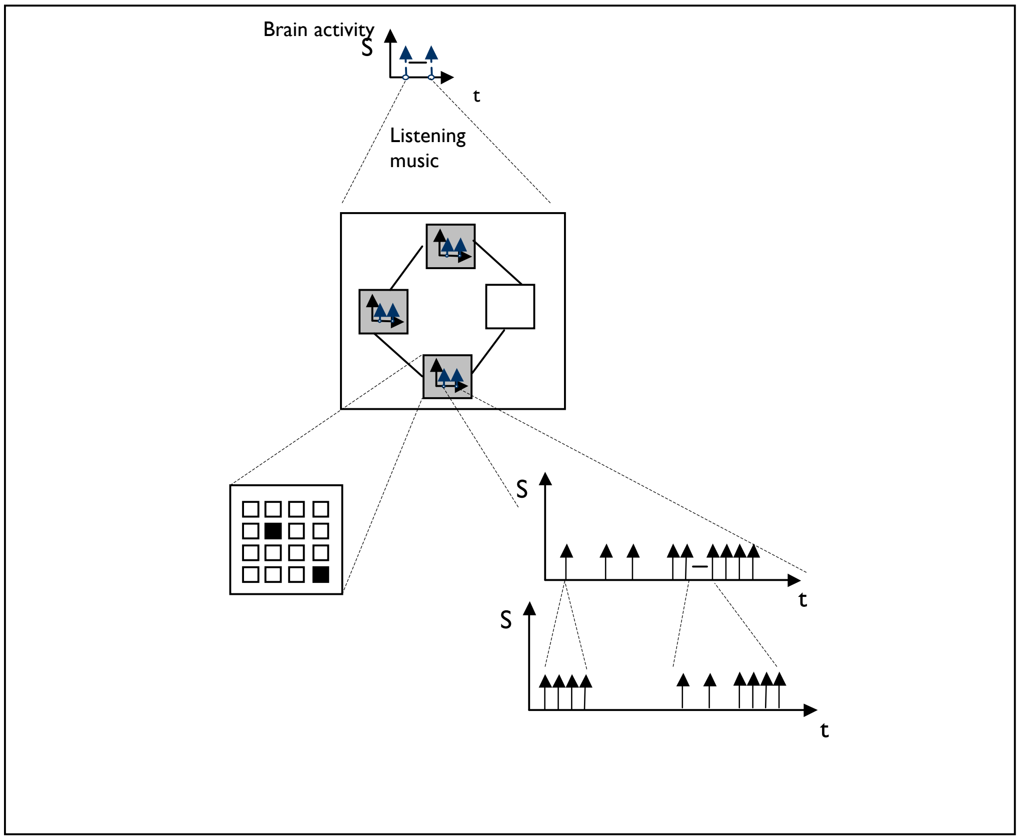

Figure 13. Hierarchy of components, activity and discrete-event abstractions

Using the pattern detection and quantization methods continuously changing variables can also be treated within the activity paradigm. As illustrated in Figure 14, small slopes and small peaks can signal low activity whereas high slopes and peaks can signal high activity levels. To provide for scale, discrete event abstraction can be achieved using quantization [56]. To determine the activity level, a *quantum* or measure of significant change has to be chosen. The quantum size acts as a filter on the continuous flow. For example, one can notice that in the figure, using the displayed quantum, smallest peaks will not be significant. Thus, different levels of resolution can be achieved by employing different quantum sizes. A genetic algorithm can be used find the optimum such level of resolution given for a given modeling objective [43].

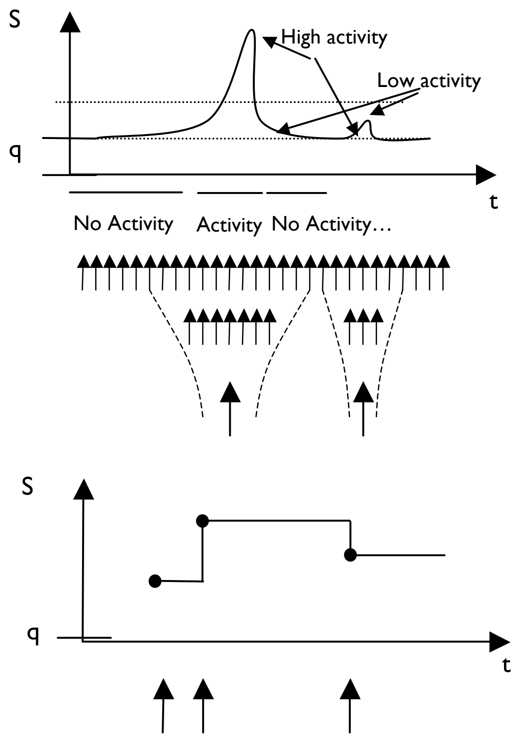

Figure 14. Activity sensitivity and discrete-events

Activity tracking

Within the activity paradigm, M&S consists of capturing activity paths through component processing and transformations. To determine the basic structure of the whole system, an automated modeler has to answer questions of the form: *where and how is activity produced, received, and transmitted*? Figure 15 represents a component-based view of activity flow in a neuronal network. Activity paths through components are represented by full arrows. Activity is represented by full circles. Components are represented by squares. The modeler must generate such a graph based from observed data – but how does it obtain such data? One approach is characteristic of current use of PET scans by neuroscientists. This approach exploits a relationship between activity and energy – activity requires consumption of energy, therefore, observing areas of high energy consumption signals areas of high activity. Notice that this correlation requires that energy consumption be localizable to modules in the same way that activity is so localized. So for example, current computer architectures that provide a single power source to all component do not lend themselves to such observation. How activity is passed on from component to component can be related to the modularity styles (none, weak, strong) of the components. Concepts relating such modularity styles to activity transfer need to be developed to support an activity tracking methodology that goes beyond reliance on energy correlation.

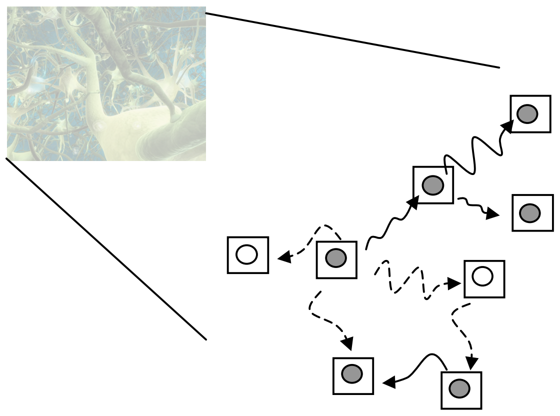

Figure 15. Activity paths in neurons

Activity Model Validation

Recall that having generated a model of an observed system (whether through activity tracking or by other means), the next step is validation. In order to perform such validation, the modeler needs an approach to generating activity profiles from simulation experiments on the model and to comparing these profiles with those observed in the real system. Muzy and Nutaro [57] have developed algorithms that exploit activity tracking to achieve efficient simulation of DEVS models. These algorithms can be adopted to provide an activity tracking pattern applicable to a given simulation model to extract its activity profiles for comparison with those of the modeled system.

A forthcoming monograph will develop the activity paradigm for M&S in greater detail [58].

9. *Intelligent Agents in Simulation*

Recent trends in technology as well as the use of simulation in exploring complex artificial and natural information processes [62,63] have made it clear that simulation model fidelity and complexity will continue to increase dramatically in the coming decades. The dynamic and distributed nature of simulation applications, the significance of exploratory analysis of complex phenomena [64], and the need for modeling the micro-level interactions, collaboration, and cooperation among real-world entities is bringing a shift in the way systems are being conceptualized. Using intelligent agents in simulation models is based on the idea that it is possible to represent the behavior of active entities in the world in terms of the interactions of an assembly of agents with their own operational autonomy.

The early pervading view on the use of agents in simulation stems from the developments in Distributed Artificial Intelligence (DAI), as well as advances in agent architectures and agent-oriented programming. The DAI perspective to modeling systems in terms of entities that are capable of solving problems by means of reasoning through symbol manipulation resulted in various technologies that constitute the basic elements of agent systems. The early work on design of agent simulators within the DAI community focused on answering the question of how goals and intentions of agents emerge and how they lead to execution of actions that change the state of their environment. The agent-directed approach to simulating agent systems lies at the intersection of several disciplines: DAI, Control Theory, Complex Adaptive Systems (CAS), and Discrete-event Systems/Simulation. As shown in Figure 16, these core disciplines gave direction to technology, languages, and possible applications, which then influenced the evolution of the synergy between simulation and agent systems.

Distributed Artificial Intelligence and Simulation

While progress in agent simulators and interpreters resulted in various agent architectures and their computational engines, the ability to coordinate agent ensembles was recognized early as a key challenge [65]. The MACE system [66] is considered as one of the major milestones in DAI. Specifically, the proposed DAI system integrated concepts from concurrent programming (e.g., actor formalism [67]) and knowledge representation to symbolically reason about skills and beliefs pertaining to modeling the environment. Task allocation and coordination were considered as fundamental challenges in early DAI systems. The contract net protocol developed by [68] provided basis for modeling collaboration in simulation of distributed problem solving.

Agent Simulation Architectures

One of the first agent-oriented simulation languages, AGENT-0 [69], provided a framework that enabled the representation of beliefs and intentions of agents. Unlike object-oriented simulation languages such as SIMULA 67 [70], the first object-oriented language for specifying discrete-event systems, AGENT-O and McCarthy's Elephant2000 language incorporated speech act theory to provide flexible communication mechanisms for agents. DAI and cognitive psychology influenced the development of cognitive agents such as those found in AGENT-0, e.g., the Belief- Desires-Intentions (BDI) framework [71]. However, procedural reasoning and control theory provided a basis for the design and implementation of reactive agents. Classical control theory enables the specification of a mathematical model that describes the interaction of a control system and its environment. The analogy between an agent and control system facilitated the formalization of agent interactions in terms of a formal specification of dynamic systems. The shortcomings of reactive agents (i.e., lack of mechanisms of goal-directed behavior) and cognitive agents (i.e., issues pertaining to computational tractability in deliberative reasoning) led to the development of hybrid architectures such as the RAP system [72].

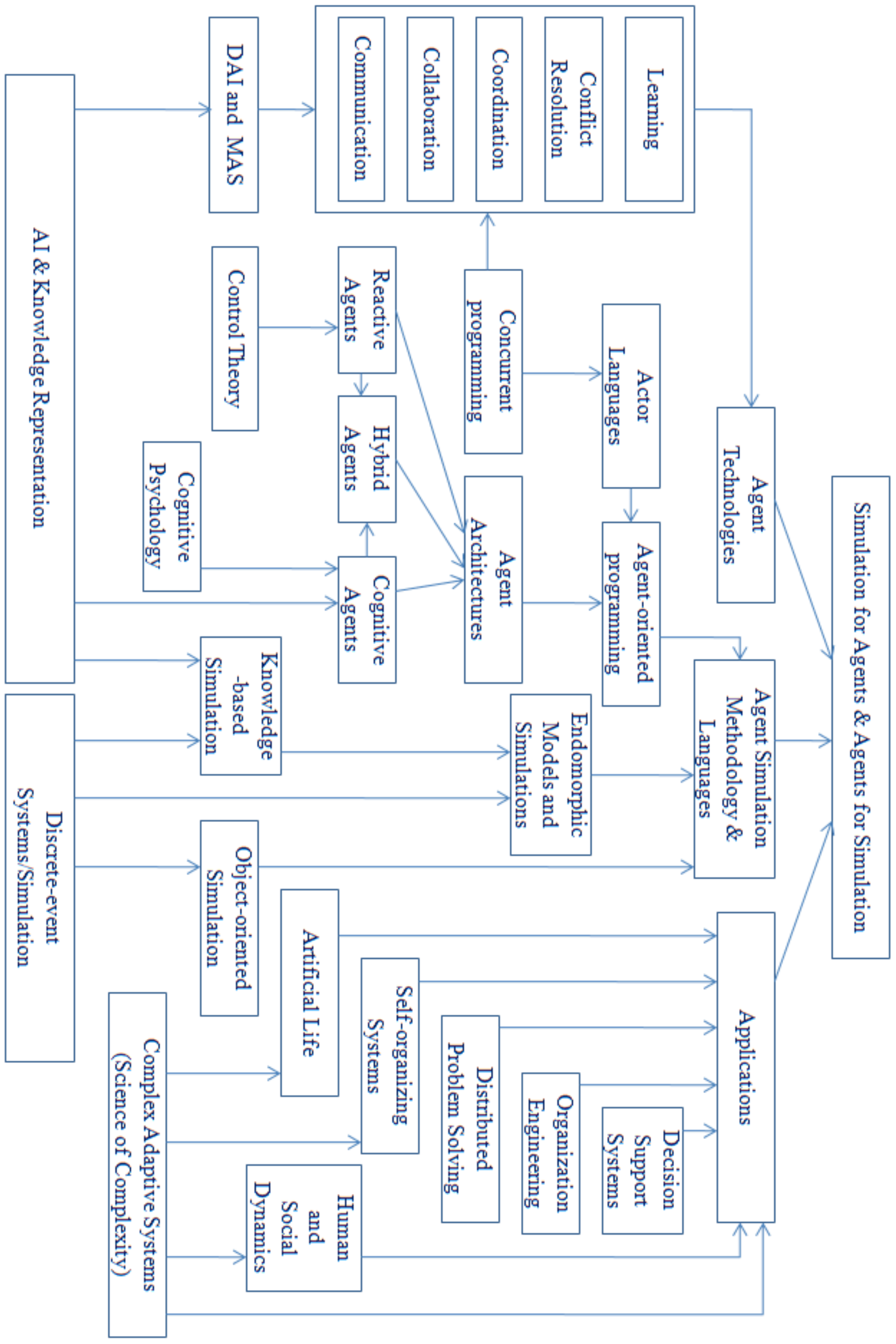
AI & Knowledge Representation
Discrete-event Systems/Simulation
Complex Adaptive Systems (Science of Complexity)
DAI and MAS
Communication
Collaboration
Coordination
Conflict Resolution
Learning
Control Theory
Reactive Agents
Hybrid Agents
Cognitive Psychology
Cognitive Agents
Concurrent programming
Agent Architectures
Actor Languages
Agent-oriented programming
Agent Technologies
Knowledge -based Simulation
Endomorphic Models and Simulations
Agent Simulation Methodology & Languages
Object-oriented Simulation
Artificial Life
Self-organizing Systems
Distributed Problem Solving
Organization Engineering
Decision Support Systems
Human and Social Dynamics
Applications
Simulation for Agents & Agents for Simulation

Figure 16. Evolution of the use of Intelligent Agents in Simulation

Agents are often viewed as design metaphors in the development of models for simulation and gaming. Yet, this narrow view limits the potential of agents in improving various other dimensions of simulation. To this end, Figure 17 presents a unified paradigm of Agent- Directed Simulation that consists of two categories as follows: (1) Simulation for Agents (agent simulation), i.e., simulation of systems that can be modeled by agents (in engineering, human and social dynamics, military applications etc.) and (2) Agents for Simulation that can be grouped under two groups: agent-supported simulation and agent-based simulation.

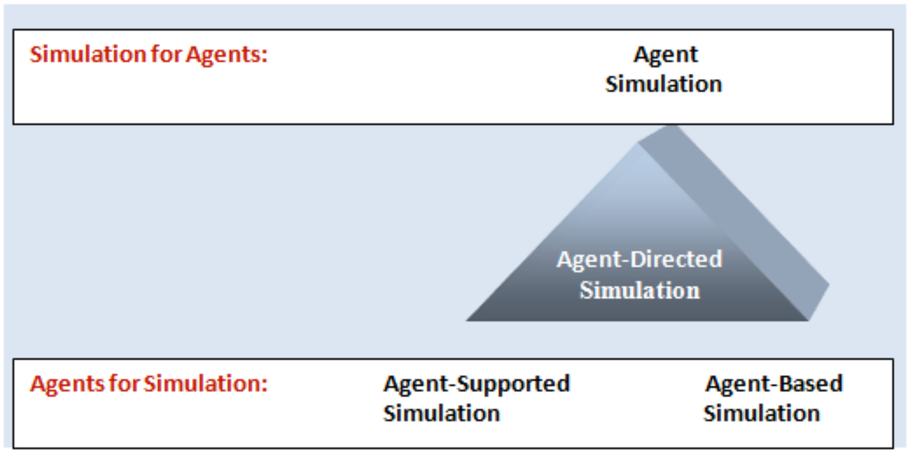

Figure 17. Agent-directed Simulation Framework

Agent Simulation

Agent simulation involves the use of agents as design metaphors in developing simulation models. Agent simulation involves the use of simulation conceptual frameworks (e.g., discrete-event, activity scanning) to simulate the behavioral dynamics of agent systems and incorporate autonomous agents that function in parallel to achieve their goals and objectives. Agents possess high-level interaction mechanisms independent of the problem being solved. Communication protocols and mechanisms for interaction via task allocation, coordination of actions, and conflict resolution at varying levels of sophistication are primary elements of agent simulations. Simulating agent systems requires understanding the basic principles, organizational mechanisms, and technologies underlying such systems.

Agent-based Simulation

Agent-based simulation is the use of agent technology to monitor and generate model behavior. This is similar to the use of AI techniques for the generation of model behavior (e.g., qualitative simulation and knowledge-based simulation). Development of novel and advanced simulation methodologies such as multisimulation suggests the use of intelligent agents as simulator coordinators, where run-time decisions for model staging and updating takes place to facilitate dynamic composability. The perception feature of agents makes them pertinent for monitoring tasks. Also, agent-based simulation is useful for having complex experiments and deliberative knowledge processing such as planning, deciding, and reasoning. Agents are also critical enablers to improve composability and interoperability of simulation models [73].

Agent-supported Simulation

Agent-supported simulation deals with the use of agents as a support facility to enable computer assistance by enhancing cognitive capabilities in problem specification and solving. Hence, agent-supported simulation involves the use of intelligent agents to improve simulation and gaming infrastructures or environments. Agent-supported simulation is used for the following purposes:

- to provide computer assistance for front-end and/or back-end interface functions;
- to process elements of a simulation study symbolically (for example, for consistency checks and built-in reliability); and
- to provide cognitive abilities to the elements of a simulation study, such as learning or understanding abilities.

For instance, in simulations with defense applications, agents are often used as support facilities to

- see the battlefield,
- fuse, integrate and de-conflict the information presented by the decision-maker,
- generate alarms based on the recognition of specific patterns,
- Filter, sort, track, and prioritize the disseminated information, and
- generate contingency plans and courses of actions.

A significant requirement for the design and simulation of agent systems is the distributed knowledge that represents the mental model that characterizes each agent’s beliefs about the environment, itself, and other agents. Endomorphic agent concepts provide a framework for addressing the difficult conceptual issues that arise in this domain.

Endomorphic Agents

We now consider an advanced feature that an autonomous, integrated and comprehensive modeler/simulationist agent must have if it is fully emulate human capability. – This is the ability, to a limited, but significant extent, to construct and employ models of its own mind as well of the minds of other agents. We use the term “mind” in the sense just discussed.

The concept of endomorphic agent is illustrated in Figure 18 in a sequence of related diagrams. The diagram labelled with an oval with embedded number 1 is that of Figure 9 with the modifications mentioned earlier to match up with human motivation and

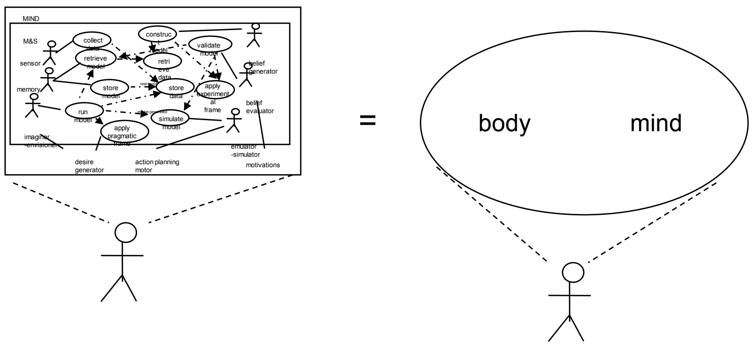

Figure 18 . M&S within Mind

desire generation modules. In diagram 2, the label "mind" refers to the set of M&S capabilities depicted in Figure 9. As in [30], an agent, human or technological, is considered to be composed of a mind and a body. Here," body" represents the external manifestation of the agent, which is observable by other agents. Whereas, in contrast, mind is hidden from view and must be a construct, or model, of other agents. In other words, to use the term of evolutionary psychology, agents must develop a "theory of mind" about other agents from observation of their external behavior. An endomorphic agent is represented in diagram 3 with a mental model of the body and mind of the agent in diagram 2. This second agent is shown more explicitly in diagram 4, with a mental representation of the first agent's body and mind. Diagram 5 depicts the recursive aspect of endomorphism, where the (original) agent of diagram 2 has developed a model of the second agent's body and mind. But the latter model contains the just-mentioned model of the first agent's body and mind. This leads to a potentially infinite regress in which – apparently –each agent can have a representation of the other agent, and by reflection, of himself, that increases in depth of nesting without end. Hofstadter [59] represents a similar concept in the diagram on page 144, in which the comic character Sluggo is "dreaming of himself dreaming of himself dreaming of himself, without end." He then uses the label on the Morton Salt box on page 145 to show how that not all self reference involves infinite recursion. On the label, the girl carrying the salt box obscures its label with her arm, thereby shutting down the regress. Thus the salt box has a representation of itself on its label but this representation is only partial.

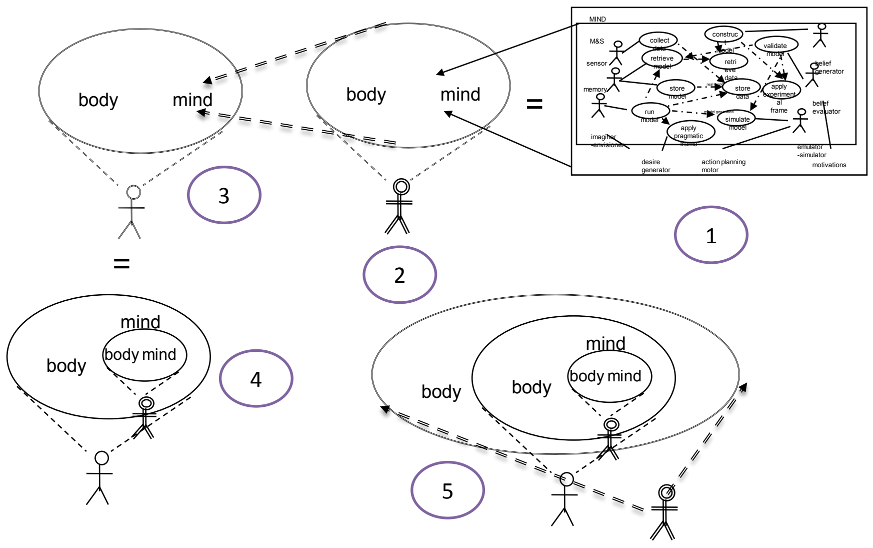

Figure 19. Emergence of endomorphic agents

[30] related the termination in self-reference to the agent's objectives and requirements in constructing models of himself, other agents, and the environment. Briefly, the agent need only to go as deep as needed to get a reliable model of the other agents. The agent can stop at level 1 with a representation of the other's bodies. However, this might not allow predicting another's movements, particularly if the latter has a mind in control of these movements. This would force the first agent to include at least a crude model of the other agent's mind. In a competitive situation, having such a model might give the first agent an advantage and this might lead the second agent to likewise develop a predictive model of the first agent. With the second agent now seeming to become less predictable, the first agent might develop a model of the second agent's mind that restores lost predictive power. This would likely have to include a reflected representation of himself, although the impending regression could be halted if this representation did not, itself, contain a model of the other agent. Thus, the depth to which competitive endomorphic agents have models of themselves and others might be the product of a co-evolutionary "mental arms race" in which an improvement in one side triggers a contingent improvement in the other – the improvement being an incremental refinement of the internal models by successively adding more levels of nesting.

Minsky [60] conjectured that termination of the potentially infinite regress in agent's models of each other within a society of mind might be constrained by shear limitations on the ability to martial the resources required to support the necessary computation. We can go further by assuming that agents have differing mental capacities to support such computational nesting. Therefore an agent with greater capacity might be able to "out think" one of lesser capability. This is illustrated by the following real-life story drawn from a recent newspaper account a critical play in a baseball game.

Interacting Models of Others in Competitive Sport

The following account is illustrated in Figure xx.

> *A Ninth Inning to Forget*
> *Cordero Can't Close, Then Base-Running Gaffe Ends Nats' Rally*
> *Steve Yanda - Washington Post Staff Writer*
> *Jun 24, 2007*
>
> *Copyright The Washington Post Company Jun 24, 2007Indians 4, Nationals 3*
>
> *Nook Logan played out the ending of last night's game in his head as he stood on second base in the bottom of the ninth inning. The bases were loaded with one out and the Washington Nationals trailed by one run. Even if Felipe Lopez, the batter at the plate, grounded the ball, say, right back to Cleveland Indians closer Joe Borowski, the pitcher merely would throw home. Awaiting the toss would be catcher Kelly Shoppach, who would tag the plate and attempt to nail Lopez at first. By the time Shoppach's throw reached first baseman Victor Martinez, Logan figured he would be gliding across the plate with the tying run. Lopez did ground to Borowski, and the closer did fire the ball home. However, Shoppach elected to throw to third instead of first, catching Logan drifting too far off the bag for the final out in the Nationals' 4-3 loss at RFK Stadium. "I thought [Shoppach] was going to throw to first," Logan said. And if the catcher had, would Logan have scored all the way from second? "Easy."*

We'll analyze this account to show how it throws light on the advantage rendered by having an endomorphic capability to process to a nesting depth exceeding that of an opponent.

The situation starts the bottom of the ninth inning with the Washington Nationals at bats having the bases loaded with one out and the trailing by one run. The runner on second base, Nook Logan plays out the ending of the game in his head. This can be interpreted in terms of endomorphic models as follows. Logan makes a prediction using his models of the opposing pitcher and catcher, namely that the pitcher would throw home and the catcher would tag the plate and attempt to nail the batter at first. Logan then makes a prediction using a model of himself, namely, that he would be able to reach home plate while the pitcher's thrown ball was travelling to first base.

In actual play, the catcher threw the ball to third and caught Logan out. This is evidence that the catcher was able to play out the simulation to a greater depth then was Logan. The catcher's model of the situation agreed with that of Logan as it related to the other actors. The difference was that the catcher used a model of Logan that predicted that the latter (Logan) would predict the he (the catcher) would throw to first. Having this prediction, the catcher decided instead to throw the ball to the second baseman which resulted in putting Logan out. We note that the catcher's model was based on his model of Logan's model so it was at one level greater in depth of nesting than the latter. To have succeeded, Logan would have had to be able to support one more level, namely, to have a model of the catcher that would predict that the catcher would use the model of himself (Logan) to out-think him and then make the counter move, not to start towards third base.

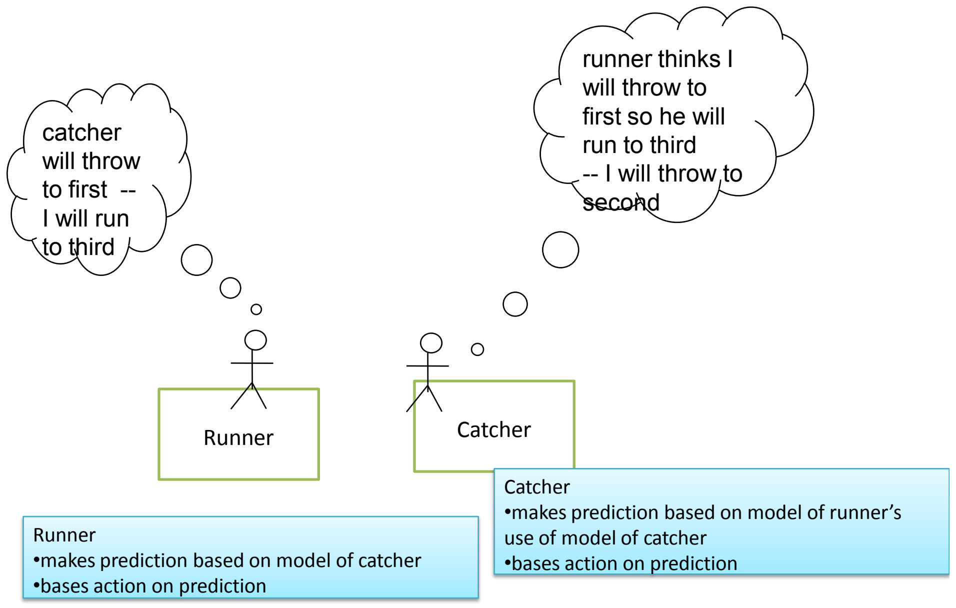
catcher will throw to first -- I will run to third
runner thinks I will throw to first so he will run to third -- I will throw to second
Runner
Catcher
Runner
•makes prediction based on model of catcher
•bases action on prediction
Catcher
•makes prediction based on model of runner's use of model of catcher
•bases action on prediction

Figure 20. Interacting Models of Others in Baseball

The enigma of such endomorphic agents provides extreme challenges to further research in AI and M&S. The formal and computational framework that the M&S framework discussed here provides may be of particular advantage to cognitive psychologists and philosophers interested an active area of investigation in which the terms "theory of mind", "simulation", and "mind reading" are employed without much in the way of definition [61].

10. Future Directions

M&S presents some fundamental and very difficult problems whose solution may benefit from the concepts and techniques of AI. We have discussed some key problem areas including verification and validation, reuse and composability, and distributed simulation and systems of systems interoperability. We have also considered some areas of AI that have direct applicability to problems in M&S, such as Service Oriented Architecture and Semantic Web, ontologies, constrained natural language, and genetic algorithms. In order to provide a unifying theme for the problem and solutions, raised the question of whether all of M&S can be automated into an integrated autonomous artificial modeler/simulationist. We explored an approach to developing such an intelligent agent based on the System Entity Structure/Model Base framework, a hybrid methodology that combines elements of AI and M&S. We proposed a concrete methodology by which such an agent could engage in M&S based on activity tracking. There are numerous challenges to AI that implementing such a methodology in automated form presents. We closed with consideration of endomorphic modeling capability, an advanced feature that such an agent must have if it is fully emulate human M&S capability. Since this capacity implies an infinite regress in which models contain models of themselves without end, it can only be had to a limited degree. However, it may offer critical insights into competitive co-evolutionary human or higher order primate behavior to launch more intensive research into model nesting depth. This is the degree to which an endomorphic agent can marshal mental resources needed to construct and employ models of its own "mind" as well of the 'minds" of other agents. The enigma of such endomorphic agents provides extreme challenges to further research in AI and M&S, as well as related disciplines such as cognitive science and philosophy.

11. Bibliography

1. References

1. Wymore, A.W., Model-based Systems Engineering: An Introduction to the Mathematical Wymore, A.W., Model-based Systems Engineering: An Introduction to the Mathematical Theory of Discrete Systems and to the Tricotyledon Theory of System Design. 1993, Boca Raton: CRC.

2. Zeigler, B. P., T. G. Kim, and H. Praehofer. (2000). Theory of Modeling and Simulation. New York, NY, Academic Press.

3. Ören T.I. and Zeigler, B.P. (1979). Concepts for Advanced Simulation Methodologies. Simulation, 32:3, 69-82.

4. http://en.wikipedia.org/wiki/DEVS

5. Knepell, P.L. and D.C. Aragno, Simulation Validation: A Confidence Assessment Methodology. 1993, IEEE Computer Society Press: Los Alamitos.

6. Law, A.M. and W.D. Kelton, Simulation Modeling and Analysis, 3rd Edition. 1999: McGraw-Hill.

7. Sargent, R.G. Verification and Validation of Simulation Models. in Winter Simulation Conference. 1994.

8. Balci, O. Verification, Validation, and Testing. in Winter Simulation Conference. 1998.

9. Davis K. P. and Anderson A. R. (2003). Improving the Composability of Department of Defense Models and Simulations, RAND Technical report. http://www.rand.org/pubs/monographs/MG101/. Last accessed, Nov. 2007(see also Journal of Defense Modeling and Simulation: Applications,Methodology, Technology 1 (1): 5{17.

10. L.Ylmaz and T.I. Oren, "A Conceptual Model for Reusable Simulations within a Model-Simulator-Context Framework, Conference on Conceptual Modeling and Simulation" , Conceptual Models Conference, Genoa, Italy , October 28-31 2004

11. Traore M, Muxy A, Capturing the Dual Relationship between Simulation Models and Their Context, Simulation Practice and Theory, Elsevier, 2004

12. Page, E., and J. Opper. 1999. Observations on the complexity of composable simulation. In Proceedings of Winter Simulation Conference, pp 553-560. Orlando, FL, USA.

13. Kasputis, S., and H. Ng. 2000. Composable simulations.In Proceedings of Winter Simulation Conference, pp 1577-1584. Orlando, FL, USA.

14. Hessam S. Sarjoughain, Model Composability, in 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.

15. M.J. DiMario System of Systems Interoperability Types and Characteristics in Joint Command and Control, Proceedings of the 2006 IEEE/SMC International Conference on System of Systems Engineering, Los Angeles, CA, USA - April 2006

16. A.P. Sage and C. D. Cuppan, "On the Systems Engineering and Management of Systems of Systems and Federation of Systems," Information Knowledge Systems Management, vol. 2, pp. 325 - 345, 2001.

17. Dahmann, J.S., F. Kuhl, and R. Weatherly, Standards for Simulation: As Simple As Possible But Not Simpler The High Level Architecture For Simulation. Simulation, 1998. 71(6): p. 378

18. Sarjoughian, H.S., B.P. Zeigler, "DEVS and HLA: Complimentary Paradigms for M&S?" Transactions of the SCS, (17), 4, pp. 187-197, 2000

19. 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, vol.1, no. 3, pp. 135-145.

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

21. http://en.wikipedia.org/wiki/Simula
22. http://en.wikipedia.org/wiki/Java
23. http://en.wikipedia.org/wiki/Expert_system
24. http://en.wikipedia.org/wiki/Frame_language
25. http://en.wikipedia.org/wiki/Agent_based_model
26. http://www.swarm.org/wiki/Main_Page
27. Unified Modeling Language (UML), http://www.omg.org/technology/documents/formal/uml.htm
28. Object Modeling Group (OMG) www.omg.org
29. http://en.wikipedia.org/wiki/Semantic_web

30. Zeigler, B.P., Object Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems, Academic Press, Orlando, FL, 1990.

31. http://en.wikipedia.org/wiki/Service_oriented_architecture

32. Mittal.S., Mak, E. Nutaro, J.J., "DEVS-Based Dynamic Modeling & Simulation Reconfiguration using Enhanced DoDAF Design Process", special issue on DoDAF, Journal of Defense Modeling and Simulation, Dec 2006

33. Simulation Methodology/Model Manipulation, in Encyclopedia of Systems and Controls, Editor of subject area, Pergamon Press, England, 1988.

34. Alexiev V., M. Breu, J. de Bruijn, D. Fensel, R. Lara, and H. Lausen, Information Integration with Ontologies, John Wiley, 2005

35. Kim, Larry, Official XMLSPY handbook, Indianapolis, IN : Wiley Pub., c2003

36. Zeigler, B.P., and P. Hammonds, "Modeling&Simulation-Based Data Engineering: Introducing Pragmatics into Ontologies for Net-Centric Information Exchange", 2007, New York, NY: Academic Press.

37. R.J. Simard, B.P. Zeigler, and J.N. Couretas, "Verb Phrase Model Specification via System Entity Structures", AI, Simulation, and Planning in High Autonomy Systems, 1994. 'Distributed Interactive Simulation Environments'., Proceedings of the Fifth Annual Conference, 7-9 Dec. 1994 Page(s):192 - 1989
38. Peter Checkland,Soft Systems Methodology in Action, 1999 Wiley
39. John H. Holland Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence The MIT Press (April 29, 1992

40. D.E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley Professional, 1989

41. L Davis, "Genetic Algorithms and Simulated Annealing", Morgan Kaufmann Publishers Inc. San Francisco, CA, 1987

42. Zbigniew Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs", Springer, 1996

43. S. Cheon, "Experimental Frame Structuring for Automated Model Construction:Application to Simulated Weather Generation", Doct. Diss. Dept of ECE, U. Ariz., Tucson, AZ, 2007

44. Zeigler, B.P., Multifaceted Modelling and Discrete Event Simulation, Academic Press, 1984.

45. J.W. Rozenblit, J. Hu, B.P. Zeigler and T.G. Kim, "Knowledge-Based Design and Simulation Environment (KBDSE): Foundational Concepts and Implementation," J. Operations Research Society, Vol. 41, No. 6, pp. 475-489, 1990.
46. T.G. Kim., C. Lee, B.P. Zeigler and E.R. Christensen, "System Entity Structuring and Model Base Management," IEEE Trans. Sys. Man & Cyber., Vol. 20, No. 5, pp. 1013-1024, 1990.

47. B.P. Zeigler and G. Zhang, The System Entity Structure: Knowledge Representation for Simulation Modeling and Design," in: Artificial Intelligence, Simulation and Modeling, (Eds. L.A. Widman, K.A. Loparo, and N. Nielsen), J. Wiley, pp. 47 73, 1989.

48. Luh, C. and B.P. Zeigler, Model Base Management for Multifaceted Systems. ACM Trans. on Modeling and Comp. Sim, 1991. 1(3): p. 195-218,

49. J. Couretas, "System Entity Structure Alternatives Enumeration Environment (SEAS)," Ph.D. Thesis, Dept of ECE, University of Arizona, 1998.

50. Hyu C. Park and Tag G. Kim, "A Relational Algebraic Framework for VHDL Models Management," Trans. of SCS, vol. 15, no.2, pp. 43-55, June, 1998.

51. Chi, S.D., J. Lee and Y. Kim, Using the SES/MB Framework to Analyze Traffic Flow. Trans. of SCS, 1997. 14(4): p. 211-221.

52. T.H. Cho, B.P. Zeigler and J.W. Rozenblit, "A Knowledge Based Simulation Environment for Hierarchical Flexible Manufacturing", IEEE Trans. Sys, Man & Cyber- Part A: Systems and Humans, Vol. 26, No. 1, pp. 81-91, January 1996.

53. Carruthers P. Massively Modular Mind Architecture The Architecture of the Mind, Oxford University Press, USA 2006, 480 pps
54. Roger Ressmeyer/Corbis
55. Zeigler, B.P., Discrete Event Abstraction: An Emerging Paradigm For Modeling Complex Adaptive Systems Perspectives on Adaptation in Natural and Artificial Systems, Essays in Honor of John Holland, Ed; Lashon Booker, Oxford University Press

56. Nutaro, J. and B.P. Zeigler, On the Stability and Performance of Discrete Event Methods for Simulating Continuous Systems. Journal of Computational Physics, 2007. 227(1): p. 797-819.
57. Muzy, A. and J.J. Nutaro. Algorithms for efficient implementation of the DEVS & DSDEVS abstract simulators. in 1st Open International Conference on Modeling and Simulation (OICMS). 2005. Clermont-Ferrand, France. p. 273-279.
58. Muzy, A. The Activity Paradigm for Modeling and Simulation of Complex Systems (in process).
59. Douglas Hofstadter, I Am a Strange Loop,. Basic Books (March 26, 2007)

60. Marvin Minsky, Society of Mind Simon & Schuster (March 15, 1988)

61. Alvin I. Goldman Simulating Minds: The Philosophy, Psychology, and Neuroscience of Mindreading Oxford University Press, USA (June 8, 2006)

62. P. J. Denning. Computing is a natural science. Communications of the ACM, 50(7):13-18, July 2007.

63. M. Luck, P. McBurney, and C. Preist. Agent Technology: Enabling Next Generation Computing - A Roadmap for Agent Based Computing. Agentlink, 2003.

64. J. H. Miller and S. E. Page. Complex Adaptive Systems: An Introduction to Computational Models of Social Life. Princeton University Press, Princeton, New Jersey, 2007.
65. J. Ferber. Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. Addison-Wesley, 1999.
66. L. Gasser, C. Braganza, and N. Herman. Mace: A extensible testbed for distributed

AI Research. Distributed Artificial Intelligence - Research Notes in Artificial Intelligence, pages 119-152, 1987.

67. G. Agha and C. Hewitt. Concurrent programming using actors: Exploiting large-Scale Parallelism. Proceedings of the Foundations of Software Technology and Theoretical Computer Science, Fifth Conference, pages 19-41, 1985.
68. R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Transactions on Computers, 29(12):1104-1113, 1980.
69. Y. Shoham. Agent-oriented programming. Artificial Intelligence, 60(1):51{92, 1993.
70. O. J. Dahl and K. Nygaard. SIMULA67 Common Base Definiton. Norweigan Computing Center, Norway, June 1967.
71. A. S. Rao and M. P. George_. BDI-agents: from theory to practice. In Proceedings of the First Intl. Conference on Multiagent Systems, San Francisco, 1995.
72. R. J. Firby. Building symbolic primitives with continuous control routines. In Procedings of the First Int. Conf. on AI Planning Systems, pages 62{29, College Park, MD, 1992.
73. L. Yilmaz and Swetha Paspuletti. Toward a meta-level framework for agent-supported interoperation of defense simulations. Journal of Defense Modeling and Simulation, 2(3):161-175, 2005.

2. Additional Reading

1. Alexiev V., M. Breu, J. de Bruijn, D. Fensel, R. Lara, and H. Lausen, Information Integration with Ontologies, John Wiley, 2005

2. Carruthers P. Massively Modular Mind Architecture The Architecture of the Mind, Oxford University Press, USA 2006, 480 pps

3. Alvin I. Goldman Simulating Minds: The Philosophy, Psychology, and Neuroscience of Mindreading Oxford University Press, USA (June 8, 2006)
4. John H. Holland Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence The MIT Press (April 29, 1992

5. Zeigler, B.P., Object Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems, Academic Press, Orlando, FL, 1990.

6. Zeigler, B.P., and P. Hammonds, “Modeling&Simulation-Based Data Engineering: Introducing Pragmatics into Ontologies for Net-Centric Information Exchange”, 2007, New York, NY: Academic Press.

7. Zeigler, B. P., T. G. Kim, and H. Praehofer. (2000). Theory of Modeling and Simulation. New York, NY, Academic Press.