

# **SANDIA REPORT**

SAND2007-1274  
Unlimited Release  
Printed May 2007

## **Design Theoretic Analysis of Three System Modeling Frameworks**

Michael J McDonald

Prepared by  
Sandia National Laboratories  
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,  
a Lockheed Martin Company, for the United States Department of Energy's  
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.



Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

**NOTICE:** This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from

U.S. Department of Energy  
Office of Scientific and Technical Information  
P.O. Box 62  
Oak Ridge, TN 37831

Telephone: (865) 576-8401  
Facsimile: (865) 576-5728  
E-Mail: [reports@adonis.osti.gov](mailto:reports@adonis.osti.gov)  
Online ordering: <http://www.osti.gov/bridge>

Available to the public from

U.S. Department of Commerce  
National Technical Information Service  
5285 Port Royal Rd.  
Springfield, VA 22161

Telephone: (800) 553-6847  
Facsimile: (703) 605-6900  
E-Mail: [orders@ntis.fedworld.gov](mailto:orders@ntis.fedworld.gov)  
Online order: <http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online>



SAND2007-1274  
Unlimited Release  
Printed May 2007

## **Design Theoretic Analysis of Three System Modeling Frameworks**

Michael J McDonald  
Intelligent Systems Principles  
Sandia National Laboratories  
P.O. Box 5800  
Albuquerque, New Mexico 87185-MS1004

### **Abstract**

This paper analyzes three simulation architectures from the context of modeling scalability to address System of System (SoS) and Complex System problems. The paper first provides an overview of the SoS problem domain and reviews past work in analyzing model and general system complexity issues. It then identifies and explores the issues of vertical and horizontal integration as well as coupling and hierarchical decomposition as the system characteristics and metrics against which the tools are evaluated. In addition, it applies Nam Suh's Axiomatic Design theory as a construct for understanding coupling and its relationship to system feasibility. Next it describes the application of MATLAB, Swarm, and Umbra (three modeling and simulation approaches) to modeling swarms of Unmanned Flying Vehicle (UAV) agents in relation to the chosen characteristics and metrics. Finally, it draws general conclusions for analyzing model architectures that go beyond those analyzed. In particular, it identifies decomposition along phenomena of interaction and modular system composition as enabling features for modeling large heterogeneous complex systems.



## Contents

Introduction .....	7
Prior System Modeling Analyses.....	9
Analysis of Simulation Model Complexity .....	11
Architectural Analysis .....	19
Conclusions .....	22

## Figures

Figure 1: Ecological system modeling issues.....	9
Figure 2: Rasmussen and Chandler’s design concept for multi-UAV system simulation.....	12
Figure 3: Buss-based coupling across multiple levels of a hierarchy .....	14
Figure 4: The Swarm hierarchical modeling approach as applied to UAVs.....	15
Figure 5: Interaction across mechanically modeled.....	16
Figure 6: Relationship between the agent-specific and World Modules that make up a typical Umbra agent model.....	18
Figure 7: System of Systems (SoS) view on interaction layers.....	19
Figure 8: Graphical representation of the design equation for UAV Swarm problem.....	19

## Acronyms

ABM	Agent Based Modeling
API	Application Programming Interface
FCS	Future Combat System
HLA	High Level Architecture
OODA	Observe, Orient, Decide and Act
RRW	Reliable Replacement Warhead
SDAC	Sense, Decide, Act, Communicate
SDI	Strategic Defense Initiative
SoS	System of Systems
TTPs	Tactics, Techniques & Procedures
UAV	Unmanned Flying Vehicle



## Introduction

Complex Systems, including complex System of Systems (SoS), have been and are being developed at a variety of scales. These systems are difficult to analyze: Their exact performance can be impossible to predict and they are likely to exhibit many unexpected behaviors. New and old analytic techniques are being developed and deployed to aid in this prediction but users are finding that it is practically infeasible to apply many of these analytic approaches to address the problems at hand. This paper analyzes the characteristics of one class of analytic tools to identify general design features that lead to more tractable analytic models using a wide variety of analytic tools.

There are many examples of SoS. Historically, the Strategic Defense Initiative (SDI) represented the first significant attempt to build a large-scale SoS.<sup>1</sup> Today, the United States Army is developing very complex SoS including the Future Combat System (FCS) to achieve its Objective Force goals.<sup>2,3</sup> The United States (US) Department of Energy's (DOE) approach to developing the Reliable Replacement Warhead (RRW) as stimulus for transforming its whole nuclear weapons enterprise<sup>4</sup> exemplifies a different kind of SoS engineering activity. Finally, architectural concepts like Surety<sup>5</sup> (integrating safety, reliability, and security) and SDAC<sup>6</sup> (Sense, Decide, Act, Communicate) drive many of characteristics of the large scale SoS that they are envisioned to be part of.

A unique characteristic of modern complex systems is in how the elements interact. Modern SoS use communications to distribute sensing, knowledge (or information), behavior (including control), and effects or action. For example, various nodes in SDAC systems sense phenomena, process the sensor data, make decisions, and act on the environment. Because each stage of the activity is linked by a resource and speed limited communications network, knowledge is distributed unevenly and percolates slowly through the network. The distributed control and uneven information distribution aspect makes it difficult to predict the behaviors of these SDAC systems.

SoS subsystems are also often closely coupled to one another through the environments within which they operate. The phrase "fog of war" is indicative of the problem all SoS face. Physical effects dominate what sensors see, how warfighters move, what people can observe, and the TTPs (Tactics, Techniques & Procedures) that work best. In many cases, these SoS literally change the environments within which they operate and make it impossible for

---

<sup>1</sup> Bordaman, J.; Sausser, B.; Verma, D.; Distinguishing Characteristics for a System of Systems, Stevens Institute of Technology

<sup>2</sup> Concepts for the Objective Force; United States White Paper. Forward by General Eric K. Shinseki, 1999. (Available at <http://www.army.mil/features/WhitePaper>)

<sup>3</sup> Ibid. Objective Force White Paper.

<sup>4</sup> Jonathan Medalia; Nuclear Weapons: The Reliable Replacement Warhead Program, Congressional Research Service Report for Congress, March 9, 2006 (CRS Library of Congress Order Code RL32929)

<sup>5</sup> Surety White Paper; White Paper for the Surety Science and Engineering Workshop, September 23, 1998, Sponsored by the National Academy of Engineering, the National Academy of Sciences, and the Department of Energy; Produced by Sandia National Laboratories (see <http://www.sandia.gov/Surety/SurWP.htm>).

<sup>6</sup> Berry, Davis, Ko, Kyker, Pate, Stark, Stinnett, Baker, Cushner, Dyke, and Kyckelhahn; Sense, decide, act, communicate (SDAC): next generation of smart sensor systems; Proceedings of SPIE -- Volume 5417

Unattended/Unmanned Ground, Ocean, and Air Sensor Technologies and Applications VI, Edward M. Carapezza, Editor, September 2004, pp. 370-381

systems engineers to isolate the systems from their environments. To exemplify this issue, see the point-counterpoint nature of how adversaries react to each other's warfighting system upgrades or how the insurgency in Iraq responds to US efforts to neutralize IEDs.

It is perhaps impossible to predict the behavior of any complex system. Size, heterogeneity, multifacetedness and complexity all play to this problem. This difficulty is not new. Intractability was historically considered the Achilles heel of the SDI by its top technical leaders<sup>7</sup> and proposals to simulate it were massive.<sup>8</sup> While exact prediction is likely to remain impossible, it is essential that the systems engineer understand the relationships between the elements as a way of knowing how best to configure the system and to grasp and understand situations as they unfold. Modern SoS engineers resonate with D. D. Eisenhower's caution that "Plans are worthless. Planning is essential." They need powerful tools to serve their planning and assessment needs as well as to help them answer their many system-level questions to prepare them for the unknowns ahead.

Throughout development, analysts and system engineers ask and re-ask a variety of questions about the system design. A typical first-order question they ask is whether the pieces will work together at a system level and whether all the right pieces have been considered. Answering this question requires bringing together the system elements and assessing their net functional performance and compatibility. Here, attributes from many fields and disciplines must be considered. A second question they ask is whether the key pieces make sense. This requires understanding the coupling of the elements so that they can be decomposed correctly and analyzed part by part. These parts must be analyzed in great detail by various teams of specialists working as independently as possible. A third question is whether the parts will still perform appropriately as a whole once the details have been specified. They need to know whether there are devils in the detail that will foul the system. This necessitates bringing together the pieces in detail and analyzing the system as a whole.

System designers and analysts have a few fundamental options for analyzing systems. A first is a theoretical analysis, where the system is modeled mathematically and analyzed through the models. This includes linear mathematical equations, diagrams, construction (i.e., geometric) models and non-linear decision-based computer simulation models. The second is through prototype experimentation where representative physical systems are built and tested. A third is in-vivo analysis where the entire system is built and experimented upon. The fourth is a hybrid approach. Functional system simulation represents an emerging form of hybrid analysis that can be thought of as model-based experiential analysis. Here, models are built, operated, studied and refined as if they were prototypes of the final systems. In many cases, these models are even hooked to operating components to provide a hybrid theoretical/in-vivo analytic environment.

Functional system modeling technologies have been advancing over the past decade to support the study of complex systems and other system theories. Of particular interest are agent-based modeling technologies that treat the elements of a system as interacting but otherwise independent entities. Agent modeling techniques let analysts represent systems of systems in terms of agent systems that integrate into large-scale systems. Sophisticated agent-based

---

<sup>7</sup> Parnas, David Lorge Software Aspects Of Strategic Defense Systems; Communications of the ACM December 1985 Volume 28 Number 12

<sup>8</sup> Zionics, G.A.; SDI national test bed status and projections; 21st Annual Electronics and Aerospace Conference, 1988. 'How will Space and Terrestrial Systems Share the Future?' Conference Proceedings, IEEE EASCON '88., 9-11 Nov 1988

models can address the multiplicity of features and issues that real systems address. For example, issues that current state of the art Unmanned Air Vehicle (UAV) agent models can simultaneously address include

- flight physics and control;
- target sensing and following;
- logistics and sustainment;
- battle damage, part reliability, energy use and other failure models;
- air space integration;
- communications, command and control; and
- mission effectiveness

For complete evaluation, each UAV agent is integrated into large-scale SoS simulations that include all the other platforms, people, and elements important to their use. For battle simulation, these systems are integrated both through the command and control networks as well as through the inherent nature of their interactions. For example, UAV models may sense opposing force elements which are themselves SoS. Target reports are processed through simulated battle and control networks (of simulated people or decision-makers) who then decide whether and how to pursue the opposing forces using other system elements. With these integrated models, systems engineers ask questions such as whether the entire observe, orient, decide and act (OODA) loop of the system under consideration, with all its delays from positioning the UAV to communicating results to commanding the response elements, is sufficient to support the battle concept under consideration.

### **Prior System Modeling Analyses**

Little has been published concerning the theoretical aspects of system modeling to address simulation complexity and feasibility. Page and Opper<sup>9</sup> address the issue of composability as a design principal for simulation. In their analysis, they consider composability along horizontal (e.g., peer to peer coupling for integration) and vertical (e.g., coupling for aggregation) dimensions and view composition as a special case of the reusability problem. Yilmaz and Paspuleti<sup>10</sup> describe issues and approaches for addressing dynamic composability and also identify coupling as a key constraining issue.

Wu and David<sup>11</sup> examine the overlapping ideas of granularity and scale as horizontal and vertical axes of the analysis of ecological systems. They conclude that “the relative importance or relationship between top-down constraints and bottom-up forces in determining system dynamics is a key to understanding most if not all complex systems.” Figure 1 illustrates their concept of dimensions of scale in studying the spatial nested hierarchy in the Alaskan tundra. Here, the lowest hierarchical level and the smallest landscape spatial unit corresponds to the individual plant and the largest level corresponds to the ecological region. Of interest is finding the right techniques for simultaneously adjusting analytic grain size and extent without losing problem traction.

---

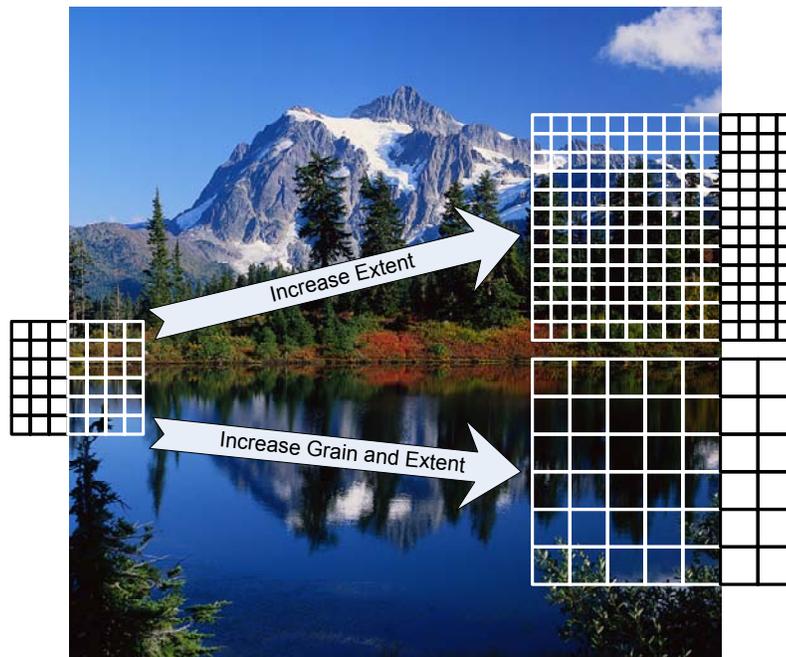
<sup>9</sup> Page, E.H.; Opper, J.M.; Observations on the complexity of composable simulation; 1999. Winter Simulation Conference Proceedings, Volume 1, Issue , 1999 Page(s):553 - 560

<sup>10</sup>Yilmaz, Levent; Paspuleti, Swetha; Toward a Meta-Level Framework for Agent-Supported Interoperation of Defense Simulations; JDMS, Volume 2, Issue 3, July 2005 Pages 161–175 © 2005 The Society for Modeling and Simulation International

<sup>11</sup> Wu, Jianguo; David, John L.; A spatially explicit hierarchical approach to modeling complex ecological systems: theory and applications; International Journal on Ecological Modelling and Systems Ecology, 153 (2002) 7–26; 2002 Elsevier Science B.V.

Outside of simulation and computer science, Nam Suh's Axiomatic Design theory<sup>12, 13, 14</sup> is becoming an important way for understanding design complexity in general. Through his application of Information Theory, Suh relates design complexity to the probability that a given design can be implemented. His theory addresses complexity in terms of the number of functional requirements and the coupling between the requirements and design parameters (or features). His analysis corresponds to observations that the difficulty for implementing a completely uncoupled design (where each feature can be developed as a stand-alone entity) grows almost linearly with the number of design requirements. Conversely, the difficulty for implementing fully coupled solutions (where each requirement depends upon all the system's features) grows geometrically or worse.

Hierarchies provide another way for people to uncouple elements. In studying cognitive processes, Miller proposed that people actively engage in the "chunking" of information as it is encoded in their memory. Miller suggested that the short term memory span is fixed at  $7 \pm 2$  chunks of information and that the amount of data that it actually contains can be increased by creating larger and larger chunks of information, each containing more bits than before.<sup>15</sup>



**Figure 1: In modeling ecological systems, Wu and David consider the problem of breaking a region into grids and studying the systems at various temporal and spatial scales. The dynamics between regions must be studied over large areas. Some issues get lost as grain size increases.**

<sup>12</sup> Suh, N. P. The Principles of Design, Oxford University Press, Oxford, 1990.

<sup>13</sup> Suh, Nam P.; Axiomatic Design: Advances and Applications; Oxford University Press, Oxford, 2001

<sup>14</sup> Suh, Nam P.; Axiomatic Design Theory for Systems, Research in Engineering Design (1998), 10:189-209, Springer Verlag London LTD.

<sup>15</sup> Miller GA; The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychol Rev, 1956, 63:81-97

While it is difficult to measure quantities like the difficulty of implementing a design, Hirschi & Frey<sup>16</sup> have shown through human experiments that geometric growth in complexity can be seen in small problem-solving experiments. Their experiment involved measuring how long it took subjects to solve an increasingly complex set of problems. Their study found that while the formal CS solution to their problem is order  $n^3$ , the subject's normalized problem completion time was order  $3.4^n$ . Relating their findings to Miller, they assert that the poor scaling of human solution on coupled parameter design problems seems to arise from the stochastic features of the iterative process humans employ. They suggest that this issue lies at the root of why similar scaling issues are seen in solving coupled design problems.

As practitioners, we at Sandia have informally observed how model complexity, in the form of implementation difficulty, varies greatly with basic architectural issues. Of particular interest, we observed that the difficulty of building robot system models seems to grow geometrically with the coupling between the modules or elements that make up the computer model. For example, researchers might spend a week building each of several uncoupled single module models and then take months to build a small set of tightly coupled but similarly scaled multi-module systems.

Upon identifying that the probable root of the model complexity problem was in coupling, we decided to address the issue by developing a software framework called Umbra.<sup>17,18</sup> Umbra, described later in this paper, simultaneously addresses both horizontal and vertical coupling in agent models. From the late 1990s on, we began applying this framework to a large number of increasingly difficult modeling problems with great success. In the summer of 2004, we completed a successful series of experiments to test the scalability of our solution. In September, 2006, we began putting the system to practical use for analyzing complex small-scale battle problems. Here, a team of experts worked through a battle plan verbally in a classical red-blue tabletop exercise. Within a week, our simulation team of two people modeled the battle, simulated it in Monte Carlo fashion and reported the results back to the experts. In essence, we closed a loop around an analysis problem that had previously taken large teams months to address.

### **Analysis of Simulation Model Complexity**

The remainder of this paper analyzes the Umbra framework and two other agent modeling frameworks. The analysis focuses on model scalability as it relates to complexity. It is constructed to highlight the salient features of the new framework so that other simulation framework developers might utilize the experience in producing subsequent tools. It is important to note that a large variety of techniques can be utilized within any modeling framework. There is no magic that forces a programmer to do the right thing in one framework and forces them to do the wrong thing within the other. The best a framework can do is make it easier for programmers to do the right thing each time. In this spirit, the analysis here focuses upon the types of models that each framework is meant to support and not the possible approaches that programmers would use to overcome their deficiencies.

---

<sup>16</sup> Hirschi, N.W.; Frey, D.D.; "Cognition and complexity: An experiment on the effect of coupling in parameter design" Research in Engineering Design 13 (2002) 123–131, Springer Verlag (DOI 10.1007/s00163-002-0011-3)

<sup>17</sup> Xavier; Patrick G., Gottlieb; Eric J., McDonald; Michael J., Oppel, III; Fred J., Patent 7,085,694, Apparatus and method for interaction phenomena with world modules in data-flow-based simulation.

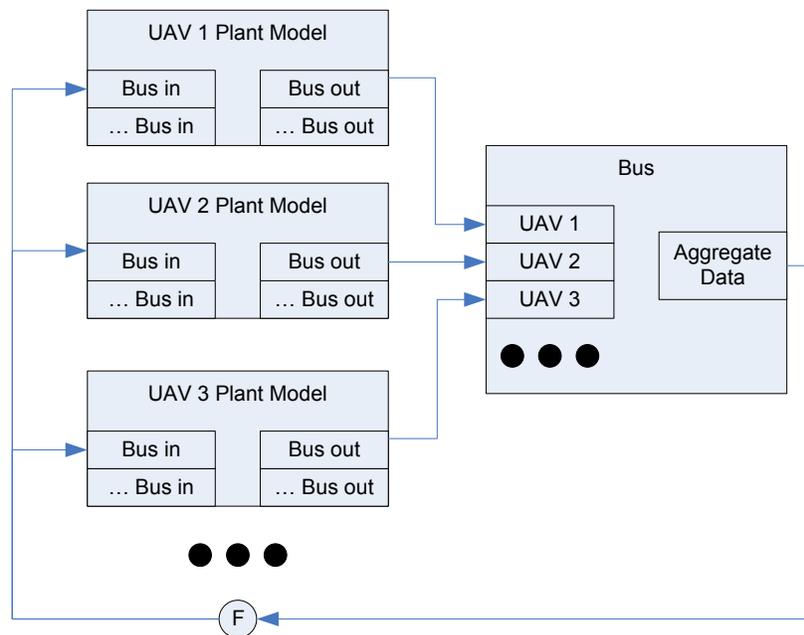
<sup>18</sup> Gottlieb, Eric J.; McDonald, Michael J.; Oppel, Fred J. III; Rigdon, J. Brian; Xavier, Patrick G., **The Umbra simulation framework as applied to building HLA federates**, 2002 Winter Simulation Conference, San Diego, CA December, 2002, SAND2002-4269P

## Data Flow Modeling of Modeling Complex Systems

Data flow modeling has been used to model a variety of complex systems. Much of this work has been done using MATLAB. MATLAB<sup>19</sup> is a high-level technical computing language and interactive environment used for algorithm development, data visualization, data analysis, and numeric computation. It provides a classical data flow model to support control systems analysis and related work. The tool is widely used by universities and industries for science, engineering and related fields. While unique for other reasons, its models are architecturally similar to those of a wide class of tools including Goldsim<sup>20</sup> and IThink.<sup>21</sup>

A key attribute of the data flow modeling approach is in how it supports horizontal decomposition. Data flow techniques let the modeler to build very detailed block-diagram control system models of systems like UAVs. Each block represents a transfer function and the blocks integrate horizontally to model the overall device. Using this technique, researchers can build very high fidelity models of the individual devices or agents.

A key issue in multi-agent simulation is in how the agents interact. For example, UAV swarm agents interact with one another through their observation of the environment (e.g., how they sense and interact with the terrain, each other, and targets they may be following) and through their communications with one another. This interaction is an issue of vertical integration.



**Figure 2: Data flow diagram showing Rasmussen and Chandler's design concept for multi-UAV system simulation. Separate buses are used for each interaction phenomena modeled.**

<sup>19</sup> MATLAB is a product of the Mathworks, which is located in Natick, MA.

<http://www.mathworks.com/>

<sup>20</sup> Goldsim ([www.goldsim.com](http://www.goldsim.com))

<sup>21</sup> IThink (<http://www.sciencesoftware.com>)

Rasmussen and Chandler's method for simulating multi-agent systems using MATLAB<sup>22</sup> serves as an example of how interaction can be modeled in data flow systems. Figure 2 reproduces the high-level data flow diagram used for their model. As can be seen in Figure 2, all agents within the data flow diagram are integrated via data busses which Rasmussen and Chandler implement as MATLAB elements. Data flows through these busses much as it would in other typical MATLAB objects. These data busses serve as central collection and distribution points for sharing data between simulated elements. The busses drawn in Figure 2 represent communications and other interactions between the UAVs and the simulation buss for conveniently accessing simulation data (for presentation). Busses can also enable data flows between, for example, targets that the UAVs followed and the UAV target following systems. They can also be used as an integration point for hybrid simulation and federation with external simulation models. For example, Stolarik et al extends the buss concept to integrate MATLAB UAV models with other simulation federates through High Level Architecture<sup>23</sup> (HLA) federation.<sup>24</sup> (Simulink<sup>25</sup> provides similar integration capabilities.) Additionally, computation within the buss element can be quite complex. For example, Stateflow<sup>26</sup> provides event simulation within MATLAB modules.

While using data busses to address vertical integration is perhaps the best that can be achieved using data flow techniques, it does present problems associated with coupling. Significant effort must be expended to design busses that minimize the amount of recoding needed to add each UAV to the simulation. Unique busses must be developed for each interaction phenomena brought under consideration. Finally, the buss concept is one of sharing data that flows through the network. The concept breaks down when the computation needed at the point of integration is so complex that it, too, is best represented in a modular fashion with elements placed throughout the system under study. (e.g., giving each UAV a radio rather than having the communications buss flow communicated information to each device.)

---

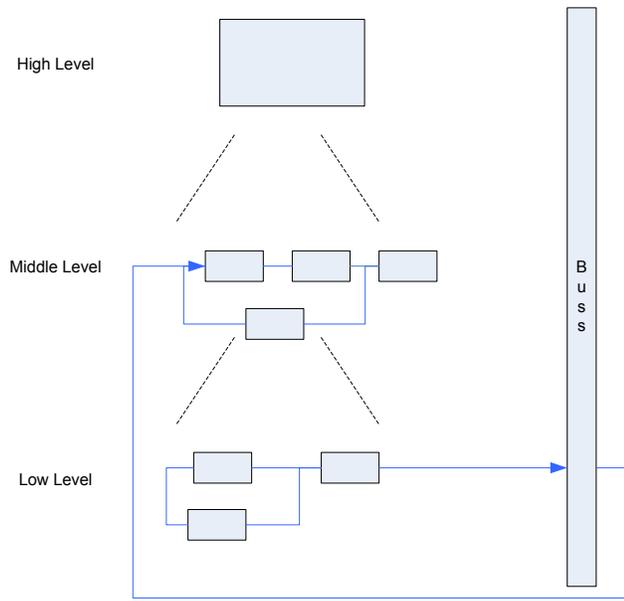
<sup>22</sup> S. J. Rasmussen & P. R. Chandler; MultiUAV: A Multiple Uav Simulation for Investigation of Cooperative Control; Proceedings of the 2002 Winter Simulation Conference; E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

<sup>23</sup> Kuhl, F, Weatherly, R Dahmann, I; **Creating Computer Simulation Systems; An Introduction to the High Level Architecture**, Prentice Hall, 1999. ISBN 0-13-02251 1-8.

<sup>24</sup> Stolarik, B., Niland, B., and Givens, B., "Multiple UAV Communication Modeling Using the HLA", *Proceedings of the 2004 SISO Spring Simulation Interoperability Conference*, Alexandria, VA, April 2004.

<sup>25</sup> Simulink® by The Mathworks <http://www.mathworks.com/products/simulink/>

<sup>26</sup> Stateflow® by The Mathworks, <http://www.mathworks.com/products/stateflow/>



**Figure 3: Buss-based coupling across multiple levels of a hierarchy.**

The buss technique presents additional concerns for hierarchical systems that are coupled at low levels of the model hierarchy. It is noteworthy that Figure 2 shows only the top-level model for each UAV model. In Rasmussen and Chandler’s work, each UAV model is built from a variety of modules. These include an aircraft dynamics module, an embedded flight software module, as well as route, target, sensor, and weapons manager modules. Generally speaking, coupling may appear at any level of the hierarchy. For example, sensors effectively couple low level physical attributes (e.g., physical presence) with higher-level technical components (e.g., sensors) that exist on different parts of the hierarchy. Figure 3 illustrates this point by diagramming the buss-based coupling between a low-level attribute and a high-level component. This threading of the coupling points complicate the development of vertically integrated systems that couple at various levels of the hierarchy. In highly coupled complex systems, the numbers of busses and connection points can be large and complex.

### ***Hierarchical Agent Modeling of Complex Systems***

A variety of tools have been developed to model agent systems hierarchically. Swarm, a classical Agent Based Modeling (ABM) software system developed by the Santa Fe Institute in the middle ‘90s by Minar, Burkhart, Langton and Askenazi,<sup>27</sup> is an excellent exemplar of this approach. Swarm remains in use with primary support from the University of Michigan Center for the Study of Complex Systems.<sup>28 29</sup>

Swarm is “a multi-agent software platform for the simulation of complex adaptive systems. In the Swarm system the basic unit of simulation is the swarm, a collection of agents executing a schedule of actions. Swarm supports hierarchical modeling approaches whereby agents can be composed of swarms of other agents in nested structures... The core of a Swarm simulation is the modeled world itself. In the simplest case, a model consists of one swarm inhabited by a

<sup>27</sup> Minar, N., R. Burkhart, C. Langton, and M. Askenazi. 1996. The Swarm simulation system: A toolkit for building multi-agent simulations. Working Paper 96-06-042, Santa Fe Institute, Santa Fe.

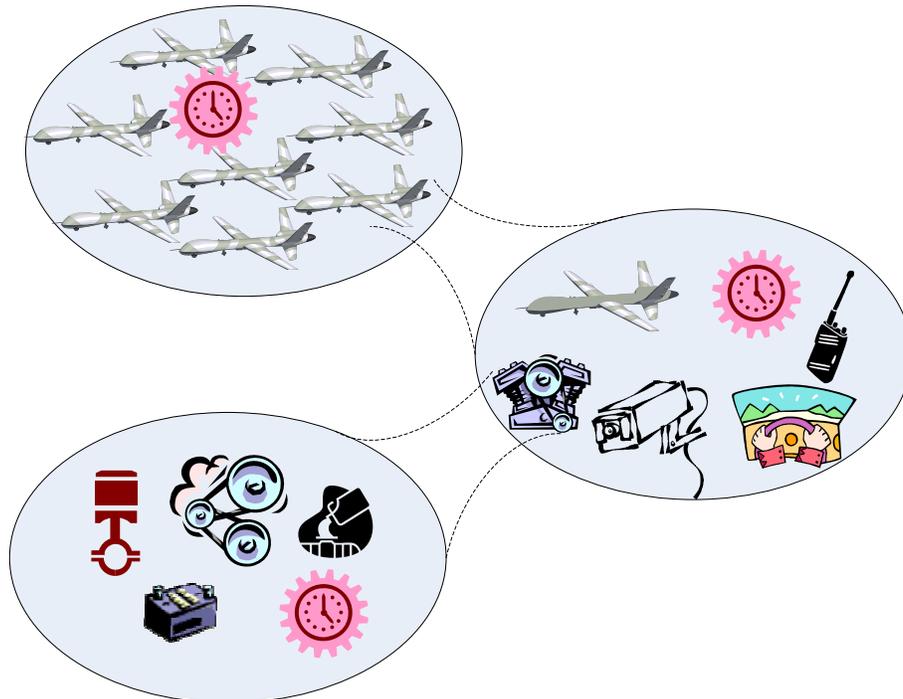
<sup>28</sup> <http://www.cscs.umich.edu>

<sup>29</sup> <http://www.swarm.org/swarmdocs/set/swarm.activity.sgml.reference.html>

group of agents and a schedule of activity for those agents. The agents themselves are implemented as objects. Agents are created by taking a class from the Swarm libraries, specializing it for the particular modeling domain, and then instantiating it, one object per agent.”<sup>30</sup>

Figure 4 illustrates the hierarchical modeling approach used in Swarm. Each ellipse represents a Swarm as defined at different levels of the system hierarchy. As Figure 4 shows and the authors explain, agents can be swarms and swarms are simply groups of agents and the schedule of activity that they command the agents to execute. Swarms of UAVs can be modeled as agents that define the swarm and its interactions. These swarms contain individual agents that define the parts of each UAV and how the parts work together (including the flight physics, sensing, and logic systems). The UAVs themselves can be defined swarms of UAV part-components (e.g., engine components). Unlike the MATLAB model, the world model or shared data as well as the mechanisms for sharing the data between agents resides within each Swarm agent. The world gathers and distributes communications between the UAVs. It must also gather position and other attributes from targets and deliver the information to the UAV sensors.

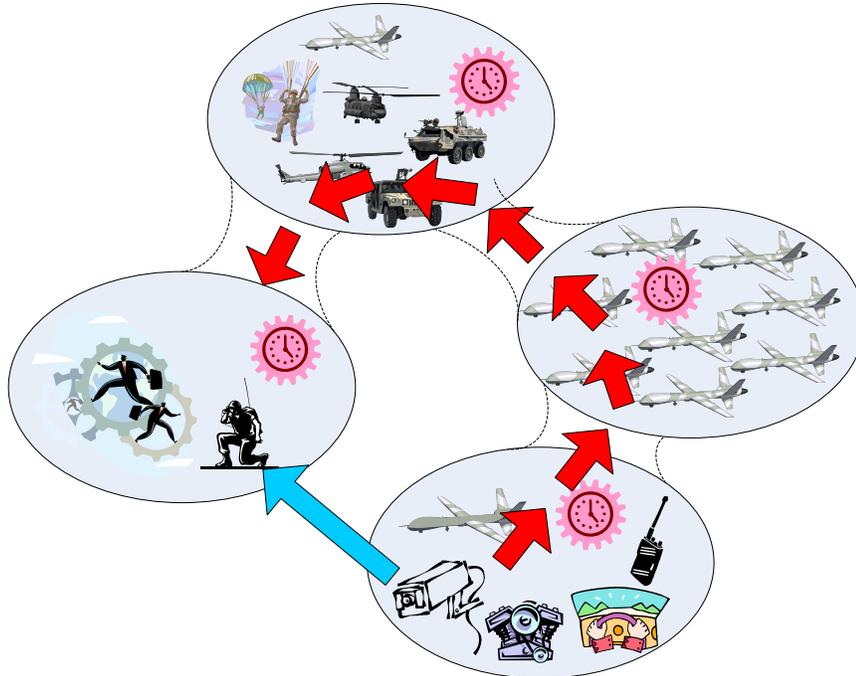
The Swarm framework’s focus on vertical integration can be problematic in modeling systems that couple at a low-level. In Swarm, systems are built as collections of subsystems and decomposed around the features that describe the systems. Likewise, system boundaries are defined at each layer of the system in a contained fashion. UAVs interact with other UAVs at the UAVs system level.



**Figure 4: The Swarm hierarchical modeling approach as applied to UAVs. Each agent in a UAV Swarm is made from UAV agents which are, in turn, Swarms of parts. The parts can also be defined as Swarms of subcomponents. Separate schedulers (time) exist at each level.**

<sup>30</sup> Ibid Minar et al.

A drawback of this vertical integration approach is in modeling systems with tight or numerous points of horizontal integration. Consider, for a moment, modeling UAV agents that contained separate physical representations, flight controllers, target sensing systems, and communications systems. In integrating the swarm of UAVs, the interfaces for each of these subsystems would need to be available (exposed) at the UAV agent level so that the UAV Swarm agent could pass the data from one UAV to the other. Additionally, the UAV Swarm would need to expose much of this data at the Swarm level in order to integrate it with other systems. As illustrated in Figure 5, the software complexity needed to have, for example, a camera see a person could be quite high.



**Figure 5: This figure illustrates the complexity of having one entity (a camera) interact with another entity (a person) that exists on a different part of the model hierarchy. In order to let the camera see the person (blue arrow), the model must connect the camera up through its swarms to the highest level swarm then down to the specific features that it is trying to observe (red arrows).**

### ***System of System Modeling with Umbra***

Umbra is a software framework developed at Sandia National Laboratories and available commercially through Orion International.<sup>31,32</sup> It is used for simulation, system integration, and system control. The most basic computational element in Umbra is called a Module. Modules combine attributes of data flow simulation with hierarchically modeled agents. As with classical data flow modelers, Umbra modules provide standard data port interfaces that support specific data exchanges between modules. It also includes an automatically generated yet extensible application programming interface (API) that users can use interactively to monitor and adjust data values and invoke module-specific events.

<sup>31</sup> <http://www.orionint.com/Capabilities/Umbra.cfm>

<sup>32</sup> Orion International Technologies, Inc. ; 2201 Buena Vista, SE, Suite 211; Albuquerque, NM 87106

As with classical ABM tools, Umbra modules can behave as individual agents and organize collections of modules by creating or parenting modules. In Umbra, we use the term World Module to describe these constructs as they are generally used to implement elements of underlying world models. World Modules create child modules by using the Factory design pattern described in the literature.<sup>33, 34</sup> Worlds can act as data repositories (e.g., black boards), react to events posted by their children and schedule events for their children. Like Swarm agents, World Modules retain all the properties of regular modules and parent-child relationships can be nested.

Umbra includes several general mechanisms for scheduling computation among families of modules. Users can and often do add additional scheduling engines to address specific computational needs. The first engine schedules computation on a regular (typically time-based) heartbeat. Here, the Umbra run-time system orders and schedules a computational event called “update” according to the module’s connectivity. This scheduler guarantees that upstream connected modules will always receive their updates before downstream modules.

The second engine builds from the first to provide an irregular updating mechanism. Every Umbra module can generate and receive named events (which become module method calls). Umbra’s SimClock consumes, schedules and posts timed events generated by other modules and application programs. At each update, the SimClock advances its model of time and then identifies and instantiates all events for the given time period. In this way, events are accurately threaded into the heartbeat of fine-grained simulation elements.

Finally, the World Module concept is often used in applications to provide additional schedulers. For example, in distributed (e.g., HLA) applications, we drive the SimClock via a federation interface World Module. Our digital communications World Module contains an event engine to schedule computation on fine-grained communication events. Using this architecture, we have been able to efficiently co-simulate the thousands of events that occur every second in large-scale wireless communications networks along side of the continuous time-based physics models needed to represent the dynamic physical elements of the system.

Figure 6 shows the relationship between the agent-specific and World Modules that make up a typical agent in Umbra. Each box in the diagram represents an individual Umbra Module. The entity or agent model is contained inside a System meta-module that wraps the agent Modules and contains system-level behaviors.<sup>35</sup> The boxes within the System represent the various component Modules that make up the individual agent. The lines between these boxes represent the Ports and Connections between those modules. The top row of grey boxes represent various worlds. The solid lines connecting the worlds represent the API relationships between the worlds. The dotted lines represent parentage relationships.

Referring back to the Rasmussen and Chandler model, it is easy to see that Umbra and MATLAB use similar composition concepts to address horizontal integration. In fact, Umbra can use compiled MATLAB modules. As with MATLAB, this approach leads to high module reuse and rapid extensibility. For example, the new sensors can be developed in stand-alone environments then connected to the behavior engine when ready. Modules representing radiant

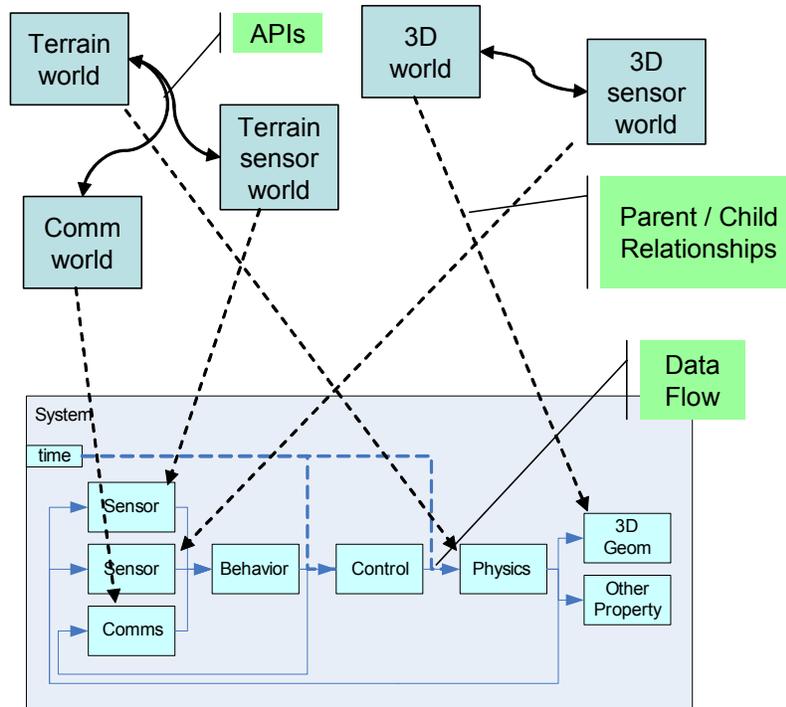
---

<sup>33</sup> Gamma, Helm, Johnson, and Vlissides, **Design Patterns, Elements of Reusable Object-Oriented Software**, Addison-Wesley, 1995

<sup>34</sup> Small, DE Gottlieb, W, Edlund, K Slutter, C; **A Design Patterns Analysis of the Umbra Simulation Framework**, October 2000, Sandia Internal Report, SAND2000-2380.

<sup>35</sup> McDonald, Michael James, **Umbra’s System Representation**, July 2005, SAND2005-3819

properties like sound and radiation can be attached in parallel with the 3D model without worrying about any other code changes. Modules representing the same phenomena but at different levels of fidelity can even be hot-swapped during comparison studies.

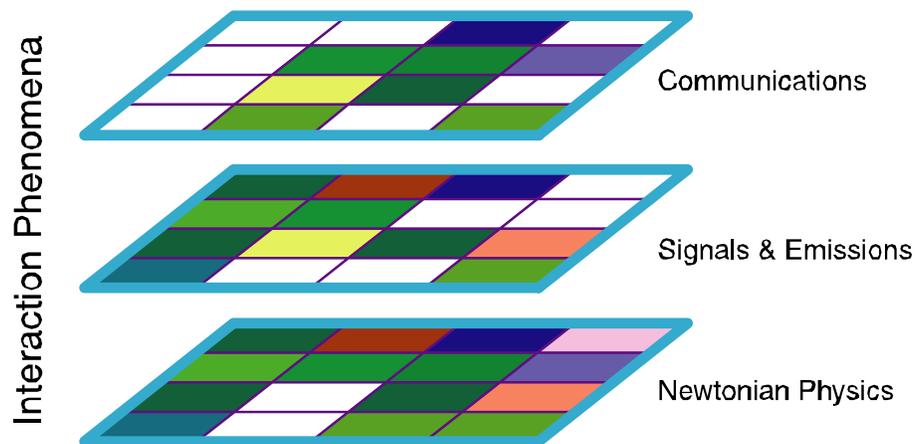


**Figure 6: Diagram illustrating the relationship between the agent-specific and World Modules that make up a typical Umbra agent model.**

The distinction between these approaches comes in the coupling between agents. Umbra's World Module construct provides a significant advancement over bus-based coupling. In addition to being more expressive and easier to manipulate, the computation it supports goes well beyond data flow by providing modularity to many forms of computation. Modules that represent complex concepts are made as intimate parts of the agent models. Using it, agents sense and follow the same terrain via the Terrain World with their terrain sensing modules. They sense their targets and one another through their shared presence in the 3D World and 3D Sensor World-based modules. They communicate with one another to coordinate their motions through radio models provided by the Communications World module.

It is noteworthy that the various worlds shown in Figure 6 are composed as pieces that serve specific functions. This is analogous to the Swarm concept of using Swarms of abstract Swarms to manage detail. Shown are Terrain (which uses an X, Y, elevation representation) and 3D worlds (where objects can take on any shape or rotation) which are separated to take advantage of specialized computational techniques that are applicable to each representation. Their associated sensor worlds were separated for similar reasons. The Comms World references the terrain world so that it can accurately calculate the terrain impacts on radio signal strength. A special adaptor class is used so that the Comms World can also connect to worlds built with other representations (e.g., the 3D world).

The Umbra framework thus supports both horizontal and vertical integration. The elements that form an agent are integrated horizontally via the data flow and system representation paradigms. Coupling along the horizontal is minimized by using well defined and easily manipulated port interfaces. Umbra’s vertical integration contrasts with the vertical integration in Swarm. In Swarm, the decomposition is made around the agent or system’s features. In Umbra, the decomposition is around interaction phenomena. The two approaches converge when all the interaction phenomena converge into one model.



**Figure 7: In any SoS, the individual Systems interact along any of several interaction layers. In the diagram, each box color represents a system that is represented as existing on several layers. Some systems are present on all layers while others only exist on just a few layers.**

The idea of decomposing a system model around interaction phenomena, as opposed to features, is unique and worthy of further discussion. Refer back to the problem of modeling sensor-based interaction using the strict hierarchies of Figure 5. In order for a camera to see a person, software was needed to bring appropriate interfaces from low to high levels of interaction. In contrast, the Umbra approach lets the programmer isolate and organize the interaction phenomena into special worlds. The results of the interaction are then brought to the hierarchically organized systems in the form of contained, well-defined modules.

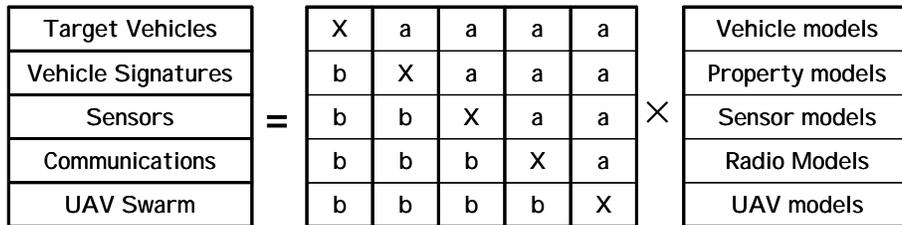
As is illustrated in Figure 7, systems within any SoS interact directly along many layers of interaction. By organizing the decomposition along interaction phenomena, modelers can slice directly through the hierarchy along the path of the physical interaction. Sensing goes directly from phenomena to sensor. Messages go from transmitting radios, through the world, then out to receiving radios. Enriching models over time by adding interaction phenomena imposes only small increases in the system complexity. What complexity is added is directly related to the issue being addressed and not the complexity associated with extending the system and SoS level models.

### Architectural Analysis

The framework-driven modeling techniques available from data flow modeling (e.g., MATLAB), classical ABM modeling (e.g., Swarm), and Umbra have strong implications concerning model scalability in terms of complexity and feasibility. To characterize the approach, we examine how each framework is typically used to model of the swarm of target-

following UAVs described previously. We then apply Nam Suh’s Axiomatic Design Theory (see Appendix) as a first construct and hierarchical organization is as a second construct for analyzing and comparing the general scalability of the three modeling architectures. As a follow-on, we consider how each modeling approach would scale as additional systems and phenomena were added to the system.

Figure 8 shows graphically how Nam Suh’s Axiomatic Design equation  $\{FR\} = [A] \{DP\}$  would be expressed for this problem.<sup>36</sup> The functional requirements (FRs) for representing terrain, wind, communications, target ID and entity behaviors are shown on the left as the primary functional requirements for the problem. A listing of proposed Design Parameters (DPs) in the form of model libraries for implementing these requirements are shown on the right. The basic form of the design matrix ( $[A]$ ) is shown values representing the basic strength order of the relationship. The  $X$  terms represent the primary relationship, the  $a$  terms represent posterior dependencies and the  $b$  terms represent consequent dependencies.



**Figure 8: Graphical representation of the design equation for UAV Swarm problem.**

According to Nam Suh, model complexity is the information content in the  $[A]$  matrix which can be characterized by the shape of the  $[A]$  matrix. Design complexity is highest when the  $a$  and  $b$  terms are non-negligible. For intricate models, these *coupled designs* are the least feasible as they require that all the problems must be solved simultaneously. Design complexity is modest when the  $a$  terms are negligible. These *decoupled designs* have modest complexity as they can be solved in order. Design complexity is lowest when both  $a$  and  $b$  terms are negligible. These *uncoupled designs* have the lowest complexity as each element can be solved independently.

One way to characterize the  $a$  and  $b$  terms is to imagine that each element (design parameter) is developed by a separate team and to then consider the sorts of agreements and discussions that would be needed between teams to develop and then integrate the elements. In a coupled design, the interaction would be continuous and intense. In an uncoupled design, little interaction would be needed.

Using Umbra, a design team would typically build the sensors and signatures within one interacting world. This world would produce property modules that were attached to target vehicles and sensor modules attached to the UAVs. The signature and sensor modules would be lightly coupled in that the sensors would need to know how to extract the appropriate signature property to compute the sensing function. Little coupling would exist between the target vehicles and the signature modules. Continuing, the UAV behavior would need to read values from the sensors and the communications network to make decisions and generate

<sup>36</sup> In his formulation (see references 12, 13, and 14), Nam Suh defines FR as a vector of functional requirements, DP as a vector of design parameters, and A as the design matrix.

messages for other UAVs. This coupling is modest in that the behavior would be tuned to the presence and data exchanges (but not implementation) with the sensor and radio models. Overall, the  $a$  and  $b$  terms would be modest or nearly negligible. Our experience is that only a few agreements need to be made to allow teams to develop these elements independently and integrate the results quickly.

Rasmussen and Chandler's method could be used to develop a similar solution. The use of the buss would act similarly to a world model. The solution may introduce additional coupling in terms of the complications in making sure that each interacting element was properly connected to its buss. Additionally, the use of connectors at the buss may increase their coupling with the swarms in terms of ensuring that the buss size matched the number of elements in the world. In practice, the popularity of MATLAB stems significantly from the ability for separate researchers to independently develop and share solutions elements developed this way.

The Swarm method introduces significant coupling between the elements. It is important to note that the Swarm hierarchy introduces new elements, the high-level swarms, that are not formally addressed by the other two models. In developing the component models, the lowest-level vehicle and vehicle signature libraries would need to be coordinated with these mid-level vehicle swarm models so that the target vehicle swarm models expose the appropriate data at all other levels of the swarm hierarchy. Likewise, the UAV, sensor, and communications teams would need to work closely together to ensure that data could flow appropriately through the UAV swarms. Finally, the swarm of swarms model would need to be carefully constructed to transfer property data to the sensors. These issues create a high degree of overall coupling between the system elements. It would be difficult to combine independently developed swarms that interact via low-level phenomena.

The  $a$  and  $b$  terms can be further understood in the context of hierarchical decomposition. As Nam Suh points out, ill-structured decompositions can result in unintended system couplings. While buss-based data flow models can be organized hierarchically, organization alone may not provide sufficient isolation and abstraction for large-scale SoS analysis. In particular, the buss-based couplings across multiple levels of a hierarchy, as shown in Figure 3, make it difficult to isolate various levels of the hierarchy. In the limiting case, whole system must be understood in order to connect appropriate elements to the busses. In practice, this limit makes it difficult to use the buss-based approach to model large, multi-faceted SoS.

The strict hierarchies that ABM systems like Swarm provides leaves the initial appearance that systems can be appropriately contained. However, as was noted, the architecture does not provide a means for managing aspects of interactions between the hierarchies. As a result, a significant amount of low-level information must be brought up to high levels of the hierarchy in order to facilitate phenomena-based interactions as shown in Figure 5. This limits the practical utility of the hierarchical decomposition and the size or complexity of problems that can be practically modeled.

Umbra's use of worlds and systems provides two orthogonal dimensions for hierarchical decomposition. The ability to decompose the world in terms of interaction phenomena allows developers and analysts to understand and analyze the interactions independently from the systems. For example, individual signature sensor modules can be instanced, placed in the environment, and studied independently of the systems that they are intended to be part of. Likewise, Umbra's system representation allows developers to group and isolate systems such as the UAVs and target vehicles in a hierarchical manner. Subsystems can be created and

studied independently from the systems. The practical result is that models developed via independent activities can and have been quickly brought together to address very complex SoS analysis problems.

## **Conclusions**

SoS and complex systems are difficult to design as their performance can be impossible to predict and they are likely to exhibit many unexpected behaviors. New analytic techniques are being developed and deployed to aid in this prediction. The feasibility of these techniques rests in basic issues that underlie their architectures. In particular, analytic tools that do not allow the analysts to both decouple and hierarchically organize the analytic elements will limit the problem scale and complexity that can be addressed.

This paper described and analyzed three analysis architectures that have been successfully used to model and analyze swarms of systems. In particular, it looked at the how each tool supports hierarchical decomposition and coupling between the decomposed elements. Both vertical (system/subsystem) and horizontal (function to function) decomposition were considered. Nam Suh's Axiomatic Design principals were used to guide the analysis. In addition, we described the need to address hierarchical decomposition, which is only treated lightly by Nam Suh.

In the analysis, we described how approaches that emphasize vertical integration alone (e.g., Swarm) make it difficult to represent the system-to-system interactions that are typical in SoS and complex systems. In particular, SoS elements interact via a large variety of phenomena. The vertical integration approach forces the interactions to be represented at the highest levels of the system. A result is that the model elements become highly coupled. The Axiomatic Design theory states that these coupled elements would difficult to develop and maintain.

In addition, we described the potential complications from integration elements, in the form of data busses, that require the system analyst to explicitly connect elements at many layers of the hierarchy. In particular, we found that it is difficult to implement and manage busses that connect elements across different layers of the hierarchy. In addition, we noted that some buss implementations may be difficult in themselves to implement. Finally, the as the buss architecture only implements one type of interaction (data flow), it is limited in its expressiveness.

In contrast, our analysis of Umbra's architectural features for horizontal and vertical integration overcame the limitations of the other two approaches. In particular, Umbra's ability to integrate vertically over multiple phenomena, via it's Worlds Modules concept, is well suited to SoS representations. Using this approach, interaction phenomena can be modeled independently from the systems and quickly integrated with system models to study overlapping issues. Like the Swarm architecture and unlike the buss architecture, Umbra's approach lets the modeler introduce a variety of data sharing and model scheduling mechanisms into the system. This provides for greater expressiveness. When combined with module-to-module connection concepts for horizontal integration, the Worlds Module approach dramatically reduces coupling over the competing architectures.

As was mentioned previously, it is likely possible that beneficial architectural features discussed here could be implemented in all the modeling environments discussed. For example, MATLAB's global variable mechanisms could be used as a work-around for buss-based modeling to implement modules that work like world modules. Likewise, various mechanisms could be used in Swarm models to provide lower-level integration interfaces. If implemented carefully, these approaches could provide significant benefit to modelers using those libraries.

In conclusion, modeling system architects and analysts should address both horizontal and vertical integration in determining the suitability of modeling tools for addressing SoS and complex systems analysis problems. Particular attention should be given to how well the approaches let the system analysts study interaction across system hierarchies. Approaches that couple the interaction throughout the system hierarchy should be avoided. Clean and expressive mechanisms for implementing the interaction models should be used whenever possible.

**DISTRIBUTION:**

1	MS 0431	Wendell Jones, 00511
1	MS 0486	Mark De Spain, 2126
1	MS 0672	Brian Van Leeuwen, 5615
1	MS 0721	Chuck Duus, 6301
1	MS 0721	Steve Roehrig, 6300
1	MS 1002	Philip Heerman, 6470
3	MS 1004	Fred Opper, 6433
3	MS 1004	Michael McDonald, 6433
1	MS 1004	Eric Parker, 6341
1	MS 1004	Brian Hart, 6344
1	MS 1004	Michael Skroch, 6344
2	MS 1004	Robert Brittain, 6344 (Copy for Engstrom)
1	MS 1005	Russell Skocypec, 6340
1	MS 1131	Regan Stinnet, 6429
1	MS 1137	Arlo Ames, 6321
1	MS 1188	John Wagner, 6341
1	MS 1374	Elaine, Hinman-Sweeney, 6927
2	MS 9018	Central Technical Files, 8944
2	MS 0899	Technical Files, 4536
1	MS 0161	Legal Intellectual Property, 11500
1	MS 0123	D. Chavez, LDRD Office, 1011