

Commonality Faults In Superscaled Systems:

An example using the simulation domain and network-centric operations

Andrew S. Loebel

Oak Ridge National Laboratory

Dean S. Hartley III

Hartley Consulting

Abstract: Development of superscaled systems embedded in hardware is expensive and difficult to accomplish. Such systems require commonality among all domains of use. This requirement of commonality sounds reasonable, if not especially important. However, investigation reveals that this commonality is of critical importance to produce desired functionality of the expected quality when the functionality is envisioned as a result of technology improvements. In general, modeling and simulation pervades both operational system development and the processes in the system. Simulations must feed development of the operational software. The method of developing models to substitute for elements of a system of systems, for example, is necessary for development, but insufficient, unless once the element is developed, the model is withdrawn and the development is plugged in and performs seamlessly and functionally as expected. This makes modeling and simulation a necessary ingredient of testing and evaluation.

Key Words: Commonality, Superscaled Systems, System of Systems, Software Systems

INTRODUCTION

Initial results of our research on systems development indicate that **without explicit treatment of commonality, undesirable and wholly unexpected performance will emerge to compromise the**

predictability, reliability, and consistency of complex systems and systems of systems (SoS).

Systems may be complex because of the nature of their content, such as large knowledge management and decision systems. Or complexity may arise from the size of the system, as mandated by a very large array of operational requirements. When the body of requirements, together, uniquely assure capabilities never before envisioned, it may comfortably be concluded that the system or SoS will be complex. In the past and currently, integration failures in complex systems resulted in failed development, often for reasons not understood.

Throughout this paper we will discuss examples to illustrate our conclusions. To our sponsors we have offered recommendations, but here we can only cite non-scaled examples from common lessons learned data. Consistency among components is critical to effective SoS performance. Superscaled or not, if inconsistency persists, compensation in person-years is often the price. Experiments and demonstrations spanning virtual and real world phenomena are expenses in this regard.

BACKGROUND--SUPERSCALED SYSTEMS

We characterize superscaled systems as having a set of requirements that encompass system(s) functionality of a type and holistic nature that could not be envisioned prior to the most recent advances in several components of modern computational hardware performance, software development methods and computational science. These advances include the power of serial processors, the capacity of integrated circuits, the performance of serial processing with respect to dimensionality and speed, the improvements in design methods which narrow the gap between design and development, and the notion that solution environments may not be limited to the one-dimensional nature of the serial-processing paradigm. Superscaled systems address complicated objectives that push the envelope of uses of modern tools and theory but focus on the strength of computers and computer science, avoiding the anthropomorphisms

(Boyer, 1996) of computational artifacts. Some example complicated objectives include:

- A reconfigurable, distributed environment both computationally and in communications among nodes, designed so that reconfiguration does not negatively affect performance, fidelity, accuracy.
- Engaging a non-trivial number of users (hundreds to thousands) whose primary responsibilities are operationally distinct and interdependent .
- A mix of software and hardware systems, with human-in-the-loop controls, whether the software is a single software system or a SoS.

Envisioned superscaled system functionality may reflect some or all of the following:

- Problem parsing at a sophisticated and detailed level.
- Requirements for horizontal and vertical integration.
- Interoperable, connected, distributed operation nodes, data collection, or/and computation power.
- Support of system and operational functions bound by laws of physics.
- Support of rapid and flexible speed of service and accuracy of results (often for decision making or human abstraction).
- System and operational functions decomposable to elemental levels.
- Complete understanding of desired properties, confirmed by decomposition; composable, complete solution matrix.

Our goal is to close the gap between systems design and systems development. Our work currently seeks to understand these types of efficient and effective software systems' construction and functionality

determination at a level of automated data management that ensures that computers do what they do best (sort) and humans do what they do best (think). Human intervention in sorting is counterproductive and computer intervention in thinking is null.

EXAMPLES OF COMMONALITY AND ACTUAL SYSTEMS

Military simulations are generally very complicated with very serious purposes; however, the illustration of faults in this paper are not complex and not of super scale. Without full investigation one could postulate that faults in subscale systems are not superscale but ordinary errors. However, the sensitivity to faults is as relevant for systems of large scale and complexity as for those simple and deterministic systems. When systems are designed to operate in integration, failure from lack of commonality resulting in system-to-system inconsistency can be missed and effects can be confusing.

Simple System 1: Commonality Failure

An example of a fault of an early version of the SIMNET model of the Bradley Fighting Vehicle of the Army is illustrative. The old Simulation Network (SIMNET) created a common virtual reality among the people training in Abrams Tank or Bradley Fighting Vehicle simulators. SIMNET was an early, relatively small superscaled system. A validation study found some problems with the trajectory functions (Hartley, 1990). There was a clear difference between the trajectory function of the model and the real world trajectory function. Because, at that time, there was no direct connection between the real world and the model, the question then had to do with the degree of accuracy that was needed for the intended uses of the model. *Here we will be more interested in what could go wrong if this lack of commonality between functions were perpetuated in the network centric battlespace, where the model and the real world will be in intimate contact.*

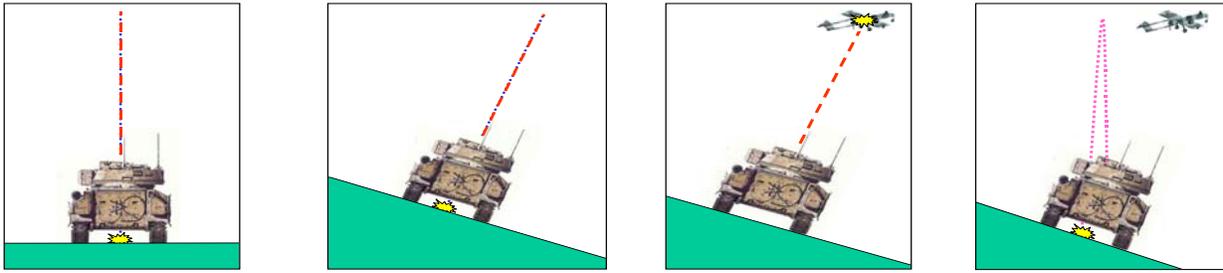


Figure 1. a

b

c

d

Figure 1.a depicts a Bradley firing its main gun directly away from the viewer toward a distant point (seen between the treads, under the hull). The trajectory is represented by the vertical dashed and dotted lines as it rises and then falls to the distant point. The model trajectory function agrees with the real world function in this case.

Figure 1.b illustrates the model's trajectory when the Bradley is tilted to one side by the terrain, but still aiming at a distant point directly away from the viewer. The model simply rotates the trajectory so that it is perpendicular to the ground, irrespective of gravity. In the model, the round hits the aim point.

Figure 1.c illustrates the current problem. Suppose the virtual simulation is connected to a "live" simulation and there is a real UAV flying through the apex of the model's trajectory as the round passes through that point. The required connection between virtual simulations and live training must generate a simulated hit on the UAV, despite the fact that the real world trajectory for the desired aim point would not intersect the UAV, as shown in **Figure 1.d**. In the real world, the gunner would have to rotate the turret slightly in the uphill direction to account for the force of gravity.

In this case, the strictures of validation coincide with the desirability of commonality. The model's

trajectory may or may not have been accurate enough for the use in the original SIMNET; however, it certainly would not be accurate enough in the situation illustrated here. A rigorous search for potentially common functions would have highlighted these two functions (model trajectory and real trajectory) as potentially common functions.

Simple System 2: Success

Another example of commonality use across real and simulated systems is relevant. Sandia National Laboratories (Feddema, Schoenwald) was conducting a robot swarm experiment for coverage and automated control of spaces inside a multi-story building (Figure 2). Although physically small, this experiment created a small, but complex, superscaled system, with emergent properties.

Four examples of emergence evolved out of this experiment.

- Inadequate number of robots could not cover the entire building with the radio communications
- Too many robots overwhelmed the communications network.
- Initial placement of the robots strongly affected the final network topology, which was also sensitive to the building geometry via the line-of-sight radio performance.
- Robots jittered while negotiating corners



Figure 2. Robot swarm inside building in autonomous operation

At first, it was thought that the robot jitter was an artifact of the simulation due to an error in the simulation. However, close observation of the actual robotic motion showed that the simulation was successfully predicting a property of the robot swarm. The control algorithms (Figure 3) developed for real and simulated robots were identical - successful commonality. This is an example of how commonality between real and simulated systems results in faithful operational functionality.

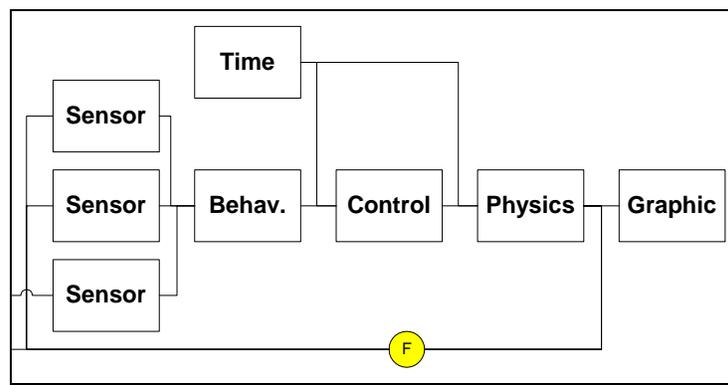


Figure 3. Simulation and experiment flow of sensor-physics of wall determination at corners

Simple System 3: Commonality Failure

One useful fault example is the lack of interoperability of Secure/Multipurpose Internet Mail Extensions (S/MIME) from lack of sufficient commonality. S/MIME defines protocols for secure electronic mail encompassing such attributes as; digital signature, non-repudiation, encryption, message integrity, authentication, and message privacy. The standards specification allows for multiple implementations; however, key interoperability features lack sufficient definition. As a result, several commercial “S/MIME enabled” products exist and do not interoperate. In this example: Commonality means that the same instantiation MUST be used wherever a specific functionality is needed. This example can be tested with as few as two systems interconnected and would seem not to be superscaled. However, in practice, one must consider the system to be the entire commercial Internet, with individual computer systems

purchasing variants of the S/MIME protocol from different commercial vendors, expecting universal secure communications. With no one in charge of the Internet, no one is responsible for determining why the system fails in this functionality. An individual user would not know why the system was failing.

System-to-System Inconsistency

The following situation occurred (during work on the Korean Battle Simulation Center (Bell, 1992). Three military models were involved, CBS, RESA, and AWSIM. The point to be made in this example is independent of the actual validity of any of the three. The point is that it is possible for three completely valid models to be linked in electronically flawless fashion and have a significant part of the resulting SoS be clearly invalid.

We will discuss the example using the names of the actual models; however, we will actually be using idealized versions of the models to make the point. The idealized versions are assumed to be completely valid for their intended uses.

CBS is a land warfare simulation, which was developed by the Army to model its units and their activities in combat. Hence it models the many types of helicopters in the Army inventory. CBS models attrition to its combat vehicles as a two-step process: $\text{probability of kill} = \text{probability of hit} * \text{probability of kill given a hit}$. Perhaps the rationale was that damaged vehicles had a fair probability of being recovered and refurbished for reuse in land warfare.

RESA is a naval warfare simulation, which was developed by the Navy to model its units and their activities in combat. Hence it models the many types of helicopters in the Navy inventory. RESA models attrition to its combat vehicles as a one-step process: probability of kill is an input variable.

Perhaps the rationale was that damaged vehicles had a fair probability of being lost at sea.

AWSIM is an air warfare simulation, which was developed by the Air Force to model its units and their activities in combat. Hence it models the many types of helicopters in the Air Force inventory. AWSIM models attrition to its combat vehicles as a one-step process: probability of kill is an input variable. Perhaps the rationale was that damaged vehicles had a fair probability of being lost over land not controlled by the Air Force.

The Aggregate Level Simulation Protocol (ALSP) was the thread to stitch these models together. It was the predecessor to the High Level Architecture (HLA) which all current models must (by DOD standard) either support or justify non-support. It was built to use some of the conventions developed in the Simulation Network (SIMNET) world of joint viewpoint simulators. In particular, in SIMNET if one simulator fired a weapon at another simulator, the firing simulator calculated the trajectory of the round and sent packets of information containing that trajectory. The other simulators received the information and determined if their position coincided with the trajectory at the proper time, and thus they were hit, and, if so, what damage was incurred. ALSP applied this philosophy to the connection of its simulations. Thus, if an entity in one simulation fired at an entity in another simulation, the simulation of the firing entity calculated the parameters of firing and the simulation of the target entity calculated the result. If both the firer and the target were in the same simulation, that simulation did all the calculations. Because the simulations in the domain of ALSP were aggregated simulations and did not calculate trajectories, the information passed was more in the nature of, "I just shot weapon W at your entity E and hit it." We will assume, for this discussion, that ALSP performs perfectly, as designed.

As it turned out, there was at least one helicopter type that was in the inventory of the Army, the Navy,

and the Air Force. The differences in the versions were small enough that the flight characteristics and vulnerabilities to weapons were identical in so far as these simulations should be concerned. The result is that CBS helicopters are less vulnerable to all weapons than are the other equivalent helicopters because a CBS helicopter that is hit might still survive, where as a RESA or AWSIM helicopter will always die (Table 1).

This "unfair situation" results because of the lack of commonality of representation, not because of any flaws in any of the simulations or of the ALSP. Table 1 reduces the data in to the bare facts: if a model hits a helicopter that is originated by the Navy or the Air Force model, the helicopter dies. This is shown graphically in Figure 4. If the helicopter is originated by the Army model, it may live. It would appear that this inconsistent result could be repaired by artificially increasing the vulnerability of the ArmyCopter to all weapons in CBS, RESA and AWSIM; however, then ArmyCopter would not have the proper vulnerability with respect to other Army helicopters in CBS.

Table 1. Helicopter Model-Victim Scoreboard

Model	Hits	Result
CBS	ArmyCopter	May live
CBS	NavyCopter	Kill
CBS	AF Copter	Kill
RESA	ArmyCopter	May live
RESA	NavyCopter	Kill
RESA	AF Copter	Kill
AWSIM	ArmyCopter	May live
AWSIM	NavyCopter	Kill
AWSIM	AF Copter	Kill

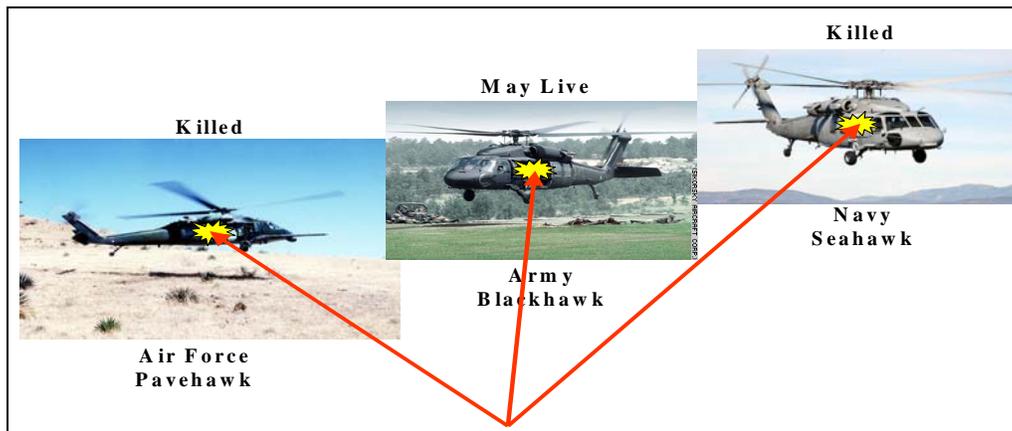


Figure 4. One type, 3 models, 2 results

It would appear to be a simple matter to make selective modifications in the RESA and AWSIM probability of kill values. For example, if RESA said that its P_k value for NavyCopter and AFCopter was 0.25, but the value for ArmyCopter was 0.50. Then the Confederation P_k value in all three cases would be 0.25. Unfortunately, when RESA targets the helicopter, all it sees is a type, not model owner. RESA can tell which helicopters it owns because it has to compute the kill; however, RESA cannot distinguish between a CBS-owned helicopter and an AWSIM-owned helicopter. Thus, RESA only has one choice for P_k value for foreign helicopters. If it uses 0.25, the result will be correct for AWSIM helicopters and incorrect for CBS helicopters. If it uses 0.50, the results will be reversed. The inconsistency cannot be resolved in this manner because there are three simulations, not just two.

This example is a simplification. In the actual case, the only solution that would yield consistent results was to nullify the entire CBS two stage attrition process for all weapons and target pairs by setting all $P_{h|k}$ values to 1.0 and using the P_k values of AWSIM and RESA. Unfortunately, this would have done violence to other parts of the CBS algorithms. As a result, helicopters were excluded from the entity sharing process to prevent any interactions among the simulations with helicopters -- a different problem

of validity. Obviously, the lack of commonality of representations causes problems in the confederation or system of simulations.

RESULTS TO DATE

Our pathfinder project results reveal **design issues**, such as functional inconsistency, which will cause SoS instability. A specific example of functional inconsistency is the error interjected into a data base through approximation. Two different (no matter to what degree) approximations or measurements of the value or identification of a target require human deliberation for resolution. (There exist many examples of computational algorithms which help with this problem, but it may be that in interjecting them, intractable complications are introduced.) Functional inconsistency, in turn, will precipitate system failure at the point knowledge is determined. This finding is based on functional decomposition and analysis of certain systems development examples of lessons learned from systems operational failures. Such failures appear to be a direct result of developing systems in a stove-piped manner without a designed-in means for integration with ‘partner’ systems, other than electronics interfaces. Systems integration has typically taken the form of connectivity and interface. Standards exist to prescribe how connectivity and interface must be maintained to produce this deterministic type of integration (however, see the S/MIME example, above).

Simpler approaches have succeeded for many previous systems (e.g., the current and older form of the Federal Aviation Administration commercial and civilian airspace management system, the World Wide Military Command and Control System, expected performance of space exploration devices and systems, specialized missile defense (e.g. Patriot), and ‘smart’ weaponry, etc.) But those systems require and cannot function without human intervention. Modern computational technology being developed will require, for example; functionality decomposition and specification that protects orthogonality and

consistency, effective data integration, and system performance so that decision making support by computational systems is not complicated by those tools. Also, any “testing-in” or use case (integration) approaches will be overwhelmed by the complexity of interactions among complex systems components, SoS, families of systems, scenarios, and the immense number of possible permutations and combinations of those interactions for any one evaluation or authentication. Existing systems analysis tools are incapable of resolving, even at electronic speed (not at human speed), inconsistencies among data records. Traditional treatment of data and technology has been serial and reductionist. Aside from "normal accidents" (Perrow, 1984), complicated systems often fail due to causes that are indecipherable, given the practiced methods and materials of design and development.

Potential Problem: Network-Centricity and Complexity

The network centric-based concept serves as a useful example: In a network centric system, the functionality of the entire system is dependent upon the functionalities of the individual subsystems and components AND their interactions. The interactions lead to non-linear effects sometimes referred to as 'emergence'. Experiments with networked centric approaches have shown that while explicit mathematical models of emergence cannot currently be implemented, simulations that capture interactions of systems elements do show the same emergent properties that physical systems display (e.g., the robot swarm example, above). Thus, any simulation used to represent a network centric system must capture the impact of element interactions so that the non-linear impact of those interactions can be understood. It is necessary to address emergence relative to expected performance, but that is not sufficient.

Since the elemental composition of a network centric system will be dynamic (system elements move into and out of a dynamic environment), the interactions of system elements will change, resulting in

nonlinear impacts (desirable and undesirable) on the overall system behavior. An example of this is the degree of communications connectivity and the availability of nodes, under different circumstances. Loss of nodes in one situation may have no effect on communication connectivity; whereas, loss of only a limited number of nodes could be catastrophic in other situations. This means that, in a communications simulation, there must be commonality of representations of functionality. Otherwise, the system-level performance will not be properly reflected when subsystem and component functionality enters and leaves the comprehensive networked system. If there is no commonality of functional representations, then there will be no guarantee that the interactions of the functions will result in representative system level behaviors. Thus a simulation of network centric systems concepts must represent similar functionality with identically coded functions, for stable results.

One may thus deduce that for systems of systems elements of operational and other domain use must be similar in the sense that:

- discrete elements in physical systems must be represented as individual elements in software systems;
- behaviors of elements among domains of use must be identical to behaviors of elements in physical systems; and
- interactions among elements in virtual systems must be identical to interactions among elements in physical systems.

Further, to keep important and/or counterintuitive effects clear, the architecture of the virtual system must reflect the architecture of the physical system.

Potential Superscale Problem: Modeling and Simulation in Training and Analysis

The bulk of our past work leading to this paper has been concerned with Modeling and Simulation (M&S) in support of acquisition, development, testing and evaluation. However, a short note on other uses for M&S in the superscaled system context is important to further elucidate faults and their implications. The virtual superscaled system may call for on-board simulations to support course-of-action-analysis (COA) and to support training. Training simulations are normally expected to match physical real time; however, analysis simulations must run at orders-of-magnitude faster than real time to support multiple executions with a minimum time cost. Even with a lower time budget and much lower resolution than that which may be assumed for the network system simulation above, training simulations have historically had difficulty in maintaining pace with real time. Analysis simulations must have restricted scope or even lower resolution to satisfy their time requirements. Current thinking indicates that either the time requirements or the requirement for operational commonality will have to be sacrificed or the virtual system computers will have to be much faster than current computers. Sacrificing the time requirements will make the simulations worthless and while future computers can be expected to be faster, the expectation is not sufficient to satisfy this requirement. A different approach to this (and other challenges) is necessary. This takes the form of a different type of commonality being understood/required.

Potential Superscale Problem: M&S in T&E

Test and evaluation of a fully operational system is a complex undertaking; however, it has the advantage of having all of the components and all of the connections among the components in existence. Figure 5 illustrates some significant elements of the operational system. In this generic example, each platform or object of the physical system will have an imbedded computer, with its software systems, various attached physical sensors and instrumentation, and its designed and implemented capabilities /

functionalities. The computers will be connected via an operational network, providing connections among and perhaps within platforms. This SoS will exhibit various properties that are to be tested and evaluated against the desired properties.

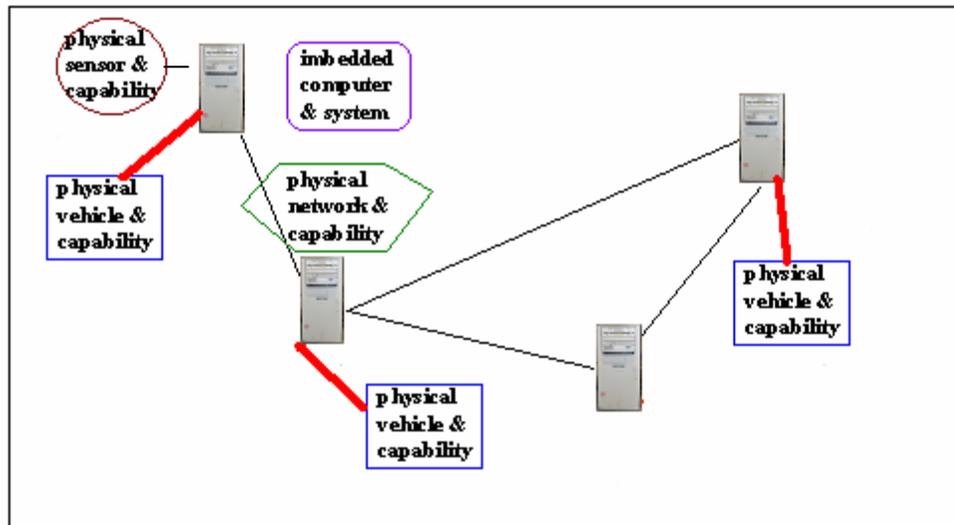


Figure 5. Operational 'schematic'

Prior to Full Operational Capability (FOC), test and evaluation is problematic. Some of the vehicles/devices may not exist, which means that their sensors will not be integrated in final fashion, nor will their computers and systems be integrated. Some of the sensors may not exist. The actual network linking the computers may not yet exist. As Figure 6 shows, the missing parts will have to be simulated or T&E cannot be performed. Note, not only will the missing hardware have to be simulated, but also the missing integrations and network connections will have to be simulated.

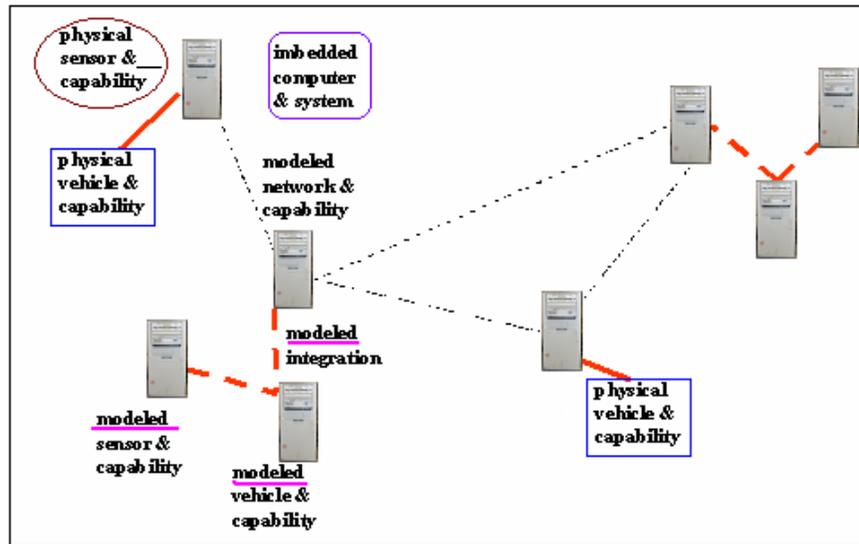


Figure 6. Developmental 'schematic'

The previous figures illustrated benign test scenarios. Such scenarios are adequate for T&E of basic capabilities, such as moving and communicating; however, they cannot test the reasons for the existence of the system, the functions that are meant to make the superscaled system superior in operation to some alternative. Figure 7 illustrates the situation that the T&E must ultimately address.

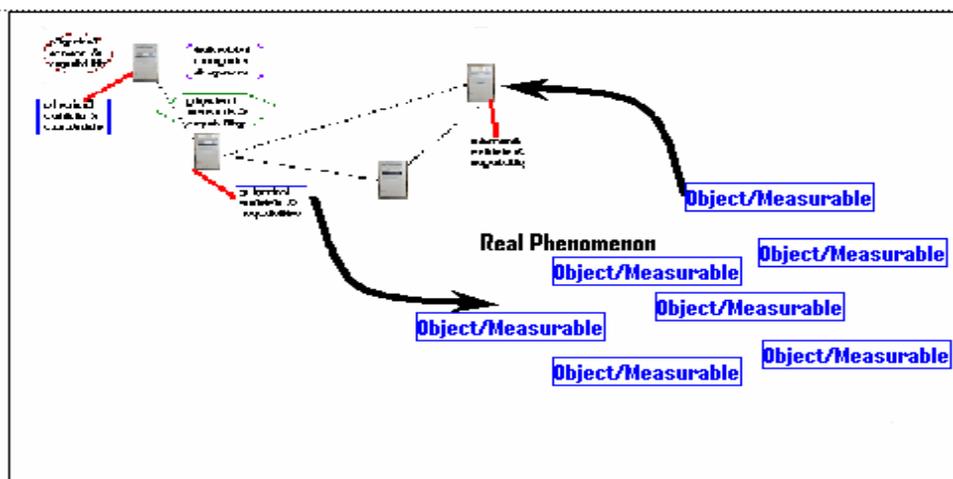


Figure 7. Testing against real phenomena

Even with an operational system, T&E does not introduce the system being tested into real world environments in order to evaluate its worth. Yet one more simulation (or set of simulations) is required. The real world objects and related processes are simulated. Figure 8 illustrates this simulation, with a developmental system. The figure also indicates that an additional simulation is required. This final simulation involves tactical and operational processes in execution by all objects and phenomenon and the adjudication of results. It should be noted that some or all of these latter two simulations may be performed by humans in conjunction with computer simulations.

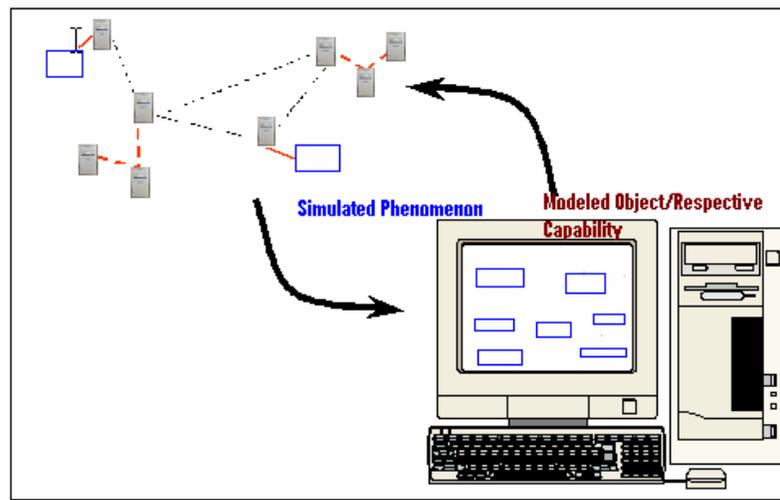


Figure 8. Testing developmental version with simulated phenomenon

Table 2 lists the simulations that are implicit in the discussion above. For each simulation, a brief phrase is used to describe what is being simulated. The final column shows that some of the simulations will not be needed in the superscaled system. However, the logic in the simulations for other parts must be identical to the operational logic.

Table 2. Simulations required for T&E

Simulation	What is Simulated	Retained in SoS?
Vehicle	Physical properties of vehicle	No?
Sensor	Physical properties of sensor	No?
Vehicle computer	Physical and processing Properties of computer	No?
Vehicle computer system	Logic of computer system	Must be common
Vehicle to computer integration	Physical & logical connections	Logic must be common
Vehicle to sensor integration	Physical & logical connections	Logic must be common
Sensor to computer integration	Logical connections	Must be common
Network	Connectivity: physical & logical Connections	Logic must be common
Network	Content: logical connections	Logic must be common
Real objects/phenomenon	Capabilities and connections	Required for COA analysis
Object/phenomenon processes	Decisions & adjudication	Required for COA analysis

Further, the complete simulations for all objects and all processes are important to address if the superscaled system is to be properly accredited. Also, a comparable simulation of all the processes associated with how the superscaled system will be used is implied for the developed superscaled system, but not specified. Such a simulation might not be required for T&E; however, it would be required for COA analysis in the operational system (see M&S in Training and Analysis, above). More complete research might discover additional required simulations and would amplify the descriptions of the simulations. Additional information is needed to decide on the parts of the simulations that must be part of the final operational system.

However, some points are obvious:

- the number and scope of required simulations for such systems is large;
- some hardware simulations will be required that have little or no use in the operational system;
- some simulations must replicate the operational functionality of parts of the system with identical software functionality;
- simulation of both the processing logic and the content handling logic will be required for integration and networked simulations; and
- simulations supporting COA analysis (and probably those supporting training) will require lower resolution models to meet processing time requirements; however, this brings into question their validity unless (a) some newly powerful computational architecture and methods are developed for faster than real time analysis or (b) a method is created to automatically generate valid lower resolution models from commonality-compliant high resolution models.

CONCLUSION

An important logical statement emerging from this work is that in order to achieve the expected performance that technology now promises, it is essential that Information Technology applications be designed and developed to work together. That is, more than (simple) electronic connectivity and interface (e.g. ATMs, consumer purchasing, inventory and logistics management, common wiring harnesses for vehicle electronics) must be the goal. This is hardly a new concept (Boutelle, 2004; Warner, 2004; Nielsen, 2004), but now we envision systems and systems performance that cannot mathematically tolerate any less.

Commonality in M&S for superscaled systems is a hard requirement in two senses:

- It is not a "nice to have" feature; it is a "must" have or "hard" feature.
- Commonality in M&S (our domain examples, here) is a difficult requirement to meet.

The simulations must be built before the operational hardware and software is built; it is difficult to have commonality with something that doesn't yet exist. This means that the real hardware and software will have to be built using the simulations as templates and sources of code, which is the reverse of the normal pattern. This means that the simulations will have to be built by simulation experts in close collaboration with the individual systems experts and subject matter experts. Such collaboration is expensive and difficult to maintain. On the plus side, this gives the experts the opportunity to get "it" right before "it" is deployed in operational settings.

ACKNOWLEDGEMENTS

Thanks are due to all of the Commonality Pathfinder Project team members at the Oak Ridge National Laboratory. Special mention is due to Raymond Harrington, PhD, of Sandia National Laboratories, for his contributions of thought, articulation, and ability to step outside the box.

REFERENCES

Bell, Richard E., et al., 1992. Battle Simulation Center Technical Support Plan, K/DSRD-1300, Martin Marietta Energy Systems, Inc., Oak Ridge, TN.

Boutelle, LT GEN, 2004. *Government Computer News*, 9/27/04.

Boyer, Pascal, 1996. Anthropomorphism and the evolution of cognition. *The Journal of the Royal Anthropological Institute*, 2, 83.

J. T. Feddema, D. A. Schoenwald, E. P. Parker, and J. S. Wagner, "Analysis and Control of Distributed Cooperative Systems," Sandia Report SAND2004-4763, Sandia National Laboratories (2004)

Hartley III, D.S., Quillinan, J.D., Kruse, K.L., 1990. Phase I Verification and Validation of SIMNET-T. K/DSRD-116, Martin Marietta Energy Systems, Inc., Oak Ridge, TN.

Nielsen, P. Maj Gen, 2004. *Government Computer News*, 9/20/04.

Perrow, C., 1984. *Normal Accidents*, Basic Books, Inc., New York.

D.A. Schoenwald, J. T. Feddema, and F. J. Oppel, "Decentralized Control of a Collective of Autonomous Robotic Vehicles," *Proc. 2001 Amer. Control Conf.* (Arlington, VA) June 25-27, 2001.

Warner, Senator J., 2004. *Government Computer News*, 9/27/04.