Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, eds.

A PROPOSED OPEN COGNITIVE ARCHITECTURE FRAMEWORK

Jeffrey S. Steinman
Craig N. Lammers
Maria E. Valinski

Joint Program Executive Office, Chemical and Biological Defense
Software Support Activity

Space and Naval Warfare Systems Center, Pacific
53560 Hull Street
San Diego, CA 92152-5001

ABSTRACT

This paper describes a proposed general-purpose cognitive architecture known as the *Open Cognitive Architecture Framework* (OpenCAF) capable of representing complex intelligent behavior for a variety of applications. This new cognitive architecture framework is combined with the *Open Modeling and Simulation Architecture* (OpenMSA) and the *Open System Architecture for Modeling and Simulation* (OSAMS) to form the foundation of the *Open Unified Technical Framework* (OpenUTF) that provides an open-source, high-performance, scalable, parallel and distributed, infrastructure for supporting both real-time operational service-oriented systems and modeling & simulation applications. The *WarpIV Kernel* provides a freely available open source C++ reference implementation of the OpenUTF that will host the proposed OpenCAF. One of the unique revolutionary features of the OpenCAF is its ability to explore multiple decision-branches within a five-dimensional simulation framework. The proposed cognitive architecture constructs described in this paper are designed to automatically provide scalable high performance execution on emerging multicore computers.

1 INTRODUCTION

Advanced human-behavior-modeling software is becoming increasingly needed to improve the automation, efficiency, and overall capability of US forces. Such software can mimic human decision-making, reasoning, learning and, in many cases, cultural and social biases, as well as perceptual, motor, and cognitive limitations. Cognitive architectures provide powerful, proven computational platforms for this type of software, including core computational abstractions and processes such as goal management, working memory, pattern matching, inference, long-term memory and learning. However, models developed within these architectures must currently be programmed at a fine-grained level roughly equivalent to assembly languages in software systems. This makes building intelligent models for these architectures time consuming and costly, requiring cognitive architecture experts. The process is also error prone, and the resulting models are difficult to maintain, combine, and expand. Traditional software development has benefited from the emergence of increasingly abstract high-level languages that make software programs for traditional computer systems orders of magnitude faster to build. A key aspect of that methodology is to exploit these abstractions in order to trade minor performance optimizations in favor of the ability to build increasingly complex systems by combining previously developed pieces of functionality. Similar advancements are required in human-behavior model development to increase the cost efficiency of model development and to increase the range of developers that can build such models. Even more fundamentally, an important goal is to raise by orders of magnitude the complexity ceiling of human-behavior models and the tasks that they can perform.

Research efforts to define high-level languages for cognitive architectures have shown that high-level languages can be defined and applied successfully to reduce cognitive model development time. However, research to this point has focused on small-scale problems and narrow domains. What is needed are more complete representations and compilation solutions that include the following: (1) support for multiple target cognitive architectures, (2) robust compilation algorithms capable of processing arbitrary cognitive models at a high-level of specification and generating robust execution code, (3) sufficient tool support that includes editors and debuggers that operate on the high-level representation rather than the architecture-specific representation.

There are a number of challenging issues associated with such an effort and a number of dimensions along which a representation and compiler must consider. First, to increase developer efficiency, a representation must abstract the details of model development while simultaneously retaining the power that cognitive architectures provide. Second, the representation must be complete, allowing it to be useful as a general-purpose language without requiring development across multiple layers of abstraction. Third, the language and its tools must be transparent, allowing behavior to be understandable and debuggable. Fourth, the compiler/interpreter must produce models that execute efficiently. Finally, the representation must be scalable with automatic support for multicore computing, and allow for incrementally building large models through plug-and-play components or modules.

2 COGNITIVE ARCHITECTURE BACKGROUND

A cognitive architecture (Langley 2006) is a blueprint for a system of intelligent agents. An intelligent agent functions to complete some task while simultaneously perceiving and acting upon its surrounding environment. Intelligent agents use prior knowledge and experience in order to appropriately respond and act upon external stimuli. When necessary, intelligent agents are capable of communicating with and leveraging the assistance of other intelligent agents.

One of the more popular cognitive architectures is Soar (Lehman, 2006), which has been actively developed by researchers over the past 25 years. The goal of Soar is to approximate rational behavior by supporting all capabilities required of a general intelligent agent. Soar decisions are not precompiled or predetermined, but rather are made based on relevant long-term and working knowledge. The Soar architecture consists of a (1) framework for representing tasks and subtasks, (2) representation of long-term memory, (3) representation of short-term memory, (4) mechanism for generating goals, and (5) mechanism for learning. Collectively, this architecture provides the means to facilitate reactive decision-making, situational awareness, reasoning, comprehension, planning, and learning.

Soar maintains three types of long-term memory: (1) procedural, (2) semantic, and (3) episodic. Procedural memory relates to how-to knowledge, semantic memory relates to facts, and episodic memory relates to experiences. Soar represents long-term memory in the form of program statements and if/then/else type rules. These rules are pre-defined in an input file, but can be modified and created during execution. Long-term memory is not directly available, but is searchable and internally organized according to the problem space. Short-term memory, or working memory, is directly available knowledge that Soar believes is relevant to the current situation. Working memory is created in response to changes in perception of the surrounding environment. Working memory contains both perceived state data and knowledge retrieved from long-term memory. Decisions are made based on available working memory knowledge, or rules, that provide a transition path from the current state to some future state. The goal is to ultimately transition to a state that completes the task at hand. If a direct transition is not available due to a knowledge gap, Soar attempts to resolve the decision based on available knowledge. After acting upon the decision, the outcome or effectiveness of the transition is stored in long-term memory. If the transition had never been conducted before, a new rule is automatically generated and saved to fill the knowledge void in long-term memory.

Cognitive architectures require an extensive knowledge base to provide realistic behavior. For example, TacAir-Soar (TacAir-Soar 2009) contains over 7,500 rules to semi-realistically control simulated aircraft in real-time training exercises, making it among the largest fielded expert systems.

3 OPEN COGNITIVE ARCHITECTURE FRAMEWORK CONSTRUCTS

This section briefly describes the fundamental set of proposed constructs that collectively provide a general-purpose cognitive architecture, known as the Open Cognitive Architecture Framework (OpenCAF) that can be applied to a wide variety of problems. Subsequent sections will tie these constructs to the WarpIV Kernel open source reference implementation (Steinman 2005) of two parallel and distributed M&S interoperability architectures, known as the Open Modeling and Simulation Architecture (OpenMSA) and the Open System Architecture for Modeling and Simulation (OSAMS) that are currently being investigated by the broad M&S community. Collectively, these three architectures form the Open Unified Technical Framework (OpenUTF) (Steinman 2008-3, Steinman 2009) that is capable of supporting M&S and other operational applications in a unified manner. A unique feature of the OpenCAF is its ability to leverage new five-dimensional HyperWarpSpeed (Steinman 2008-1) simulation technology to improve the decision-making process.

3.1 General Purpose Modeling Constructs

Entities are special *simulation objects (SimObjs)* that are distributed across processors for parallel operation. In military simulations, entities normally represent real-world battlefield entities such as soldiers, tanks, ships, aircraft, and missiles. Entities are hierarchically composed of *model components* that implement the actual behavior of the entity. Model composition is never hard-coded, but is configurable based on run-time configuration files. Models are derived from component classes and can be hierarchically composed of other model components. Models are typically implemented as a collection of one or more associated user-defined objects that reside and are encapsulated within a component.

Events are a one-shot computation that is scheduled to occur at a discrete point in time for a particular entity. Events can be scheduled by an entity (a) for another entity, or (b) for itself. Programmatically, events are generally implemented as a method on an object. Applications can define their event methods with up to 20 strongly type-checked user-defined arguments. Four basic types of events are supported. *Autonomous* events are dynamically created event-objects that act on an entity from the outside when the event is processed. Autonomous events are then destroyed after the event has been processed and committed. Autonomous events are active "agent" objects that act on passive entities by calling methods provided by the entity. *SimObj* events are directly implemented as methods on entities without the need for autonomous agents. This provides a more direct way to schedule activity between simulation objects. *Local* events are implemented as methods on any object. Local events are different from Autonomous and SimObj events in that they are scheduled from within an entity and never scheduled for other entities. *Interactions* provide a fourth type of event that allows entities, models, and objects to arbitrarily register handlers for each interaction type. Multiple handlers can be triggered when an interaction is received. Undirected interactions are delivered through publish/subscribe distribution mechanisms, while directed interactions can be scheduled for one or more specifically identified entities. In addition to these four basic event types, entities can be dynamically instantiated and deleted in simulated time.

Behaviors provide a natural modeling extension to one-shot events in that they can be active over an extended period of time. Behaviors can be scheduled just like events. However, once executed, a behavior is capable of (a) waiting for fixed periods of time before continuing, (b) waiting for specific conditions to occur before waking up, (c) waiting for requested resources to become available, and/or (d) being interrupted by user-defined interrupt conditions set by other events or behaviors. Entities, components, and other user-defined objects can have multiple behaviors operating simultaneously. An interrupt can simultaneously wake up multiple behaviors.

Federation Objects (FOs) allow exportable state variables that are embedded within entities, components, and user-defined objects to be mapped into special proxy objects that are published and then automatically distributed to subscribing entities and their components. Updated attributes are automatically provided to subscribers, where the subscriber is notified of the attribute changes. Publish/subscribe mechanisms provide the means for distributing entity state information within (a) parallel and distributed computing environments and (b) composable federations interacting through standard High Level Architecture (HLA) services. A scalable interest management service that efficiently filters FO distribution and attribute updates is essential in supporting massively parallel large-scale scenarios. The most important type of interest management is range-based filtering, which provides FOs to subscribing entities within a specified range. Scalable range-based filtering for arbitrarily moving entities in parallel and distributed simulation environments offers significant challenges (Steinman 1994).

3.2 Special Cognitive Architecture Constructs

Sensitive Methods allow any method on any object to register itself as being sensitive to any number of state variables that act as stimulus. No class inheritance is required and methods can be named anything. Sensitive methods are automatically invoked whenever a registered stimulus is modified. While processing a triggered sensitive method, additional state variables registered as stimulus for other sensitive methods may be modified, thereby triggering additional sensitive methods. In this manner, cognitive thought processes occur. Priorities can be assigned to each stimulus as they are registered with sensitive methods. Ordering the triggering of sensitive methods based on stimulus priority, used to indicate relative importance, helps to achieve faster and more meaningful cognitions.

Stimulus can be represented as encapsulated state variables residing within entities, models, and objects, or as data within a locally managed database. To provide stimulus information to sensitive methods, a registration system with the local database is provided to help lookup arbitrary stimulus by name. Lookups must also work for attributes residing within discovered FOs and parameters residing within Run Time Classes (RTCs). The RTC utility can also be used to parse XML documents or messages, which is critical for supporting web-service interfaces. Stimulus can be thought of as volatile *short-term memory*. Stimulus should always be based on perception and not truth. *Long-term memory*, on the other hand, might consist of input parameters, doctrine, rules of engagement, and operational concepts. Long-term memory rarely changes during model execution and would not require use of the stimulus infrastructure, which can add significant overhead.

Thoughts are represented as sensitive methods on classes derived from a thought base-class that work within a plug-and-play thinking framework. User-derived classes can implement multiple thoughts that share data encapsulated within object instances. Processing a thought may stimulate new thoughts as stimulus data is modified. Thinking stops when running out of ideas (i.e., there are no more new thoughts) or when the thinking process is intentionally terminated. The same thought (i.e., method on a thought object) can be triggered multiple times as the same stimulus becomes modified by other thoughts. In this manner, cognitive models can change their minds or converge to a solution during the thinking process as refinement occurs. Thoughts are conceptual rules that are automatically triggered by stimulus. In this manner, thoughts are much more general and individually encapsulated than hardcoded monolithic if-then-else logical expressions commonly coded in many systems. Models register/unregister their thoughts dynamically as models and behaviors change over simulated time.

Actions are potentially generated by thoughts to change model behaviors. Because thoughts can change their mind, actions must be committed after the thinking process completes. Actions can interrupt behaviors, terminate behaviors, and/or launch new behaviors.

Goals are represented as a prioritized list of desired objectives. Instead of scripting behaviors, each entity's goals are listed and prioritized. Goals specify desired entity accomplishments such as an aircraft dropping a bomb on a specific target, surviving a chemical or biological weapon attack, delivering logistic supplies to troops, or securing a strategic position on the battlefield. Goals and their priorities can dynamically change as new circumstances arise. User-defined goals inherit from a goal base class that plugs into a goal-oriented framework within the OpenCAF.

State must be characterized in a formal manner to determine tasking. State transitions can be represented using Finite State Machine (FSM) formalisms. Stimulus, thoughts, and actions all work together to provide non-deterministic (i.e., scripted) state transitions.

Tasks are represented as one or more behaviors on user-defined classes that inherit from a task base class. Tasks provide the set of behaviors that are required to accomplish goals. Tasks provide the means to transition from one state to another. One can think of the arcs in a state diagram as defining the tasks needed to accomplish a state transition. In the simplest case, a single task can be performed to transition from the current state to accomplish the highest priority goal. However, sometimes, multiple tasks (i.e., multiple state transitions) in a sequence are required to accomplish a desired goal. A simple way to manage multiple tasks is to establish a *Task Table* that defines the next task based on the current state and goal. This technique is conceptually similar to a network routing table that is used to identify which network hop to take when routing a message to a final destination. Multiple tasks (similar to multiple hops in a routing system) may be required to complete a goal. Branching can be used in more complicated situations to determine the best sequence of tasks when multiple tasks could be used to complete a goal. Again, this is analogous to a message routing system with dynamically changing congestion, where it is not always clear what is the best route. Branching constructs and features are described in the next section on HyperWarpSpeed five dimensional simulation.

3.3 HyperWarpSpeed Constructs/Features for Five Dimensional Simulations

Branching automatically occurs when a behavior explores two or more decisions. Probabilities are assigned to each decision branch to facilitate statistically correct Bayesian analysis. Multiple timelines are launched for a model behavior when this occurs. Similar to the familiar Switch-Case statement in C++, each branch continues at a different entry point within the behavior. Multivariable data structures and local variables within the behavior keep track of state variables that store multiple values for different timelines. All of this is automated within the HyperWarpSpeed algorithm.

Splitting occurs when a behavior or one-shot event accesses state variables that contain different values set during the processing of multiple parallel timelines. Event splitting provides the dependency framework that causes branches to propagate their effects to other models and entities. The modeler uses branching constructs to create multiple timelines, while splitting happens automatically when necessary while processing events.

Merging automatically occurs whenever possible for identical behaviors or events operating on different timelines. Merging is essential in supporting the execution of scalable five dimensional simulations. Without merging, HyperWarpSpeed simulations would quickly diverge into excessively large and fragmented parallel universes.

Multiple Futures are generated during a simulation that branches behaviors. These futures are captured in a *Futures Graph* that can be analyzed and used to go back in time to optimize one or more decisions. A rollback infrastructure embedded within HyperWarpSpeed rewinds only those events that are affected by the optimized decision. In this manner, the cognitive architecture is able to learn from decision mistakes and make corrections. The construction of the futures graph is straightforward. However, the analysis of the futures graph is an active area of research and will require considerable investigation. Furthermore, rollback-based backward-time event scheduling to lock-in critical decisions could introduce unstable time vortices. This new technique has never been fully explored, but offers significant benefit for a wide variety of live estimation and prediction problems.

4 PROPOSED ARCHITECTURE STANDARDS

Many legacy M&S tools developed for the Department of Defense (DoD) are not in use today, primarily because they do not easily interoperate with other tools, run too slow, and/or are too expensive to maintain. The DoD M&S community must address these serious problems. A new Parallel and Distributed Modeling & Simulation Standing Study Group (PDMS-SSG) was recently formed within the Simulation Interoperability Standards Organization (SISO 2009) to investigate next-generation architectures and technologies that specifically address emerging multicore computing revolution and interoperability goals.

Three synergistic architectures are proposed as a comprehensive solution to the problem. First, the *Open Modeling and Simulation Architecture* (OpenMSA) provides a high-performance parallel-and-distributed event-processing framework for modeling and simulation. Second, the *Open System Architecture for Modeling & Simulation* (OSAMS) compliments the OpenMSA and focuses on modeling constructs and methodologies necessary to support highly interoperable plug-and-play model components. Third, the *Open Cognitive Architecture Framework* (OpenCAF) is an extension to OSAMS that provides a structured general-purpose framework for modeling complex human behaviors or other cognitive processes. Collectively, these three architectures form the foundation of a larger *Open Unified Technical Framework* (OpenUTF) that can support high-performance, net-centric, interoperable, and intelligent applications beyond just M&S. The scheduling framework within the OpenUTF can support real-world service-oriented applications as well.

OpenMSA provides a cross-platform layered architecture that focuses on the critical *technologies* necessary to support scalable parallel and distributed M&S. The OpenMSA provides technologies such as (a) High Performance Computing communications, (b) network-based ORB services, (c) advanced real-time and logical-time scheduling services, (d) an elegant modeling framework and suite of utilities that significantly simplifies software development of complex systems of systems, (e) scalable publish/subscribe data distribution services with automated interest management, (f) LVC interoperability through standards such as HLA, DIS, and TENA, and (g) integration with web-based standards such as SOAP and WSDL to host network-based SOA on multicore computers across the world wide web and the GIG.

OSAMS defines a standard hierarchical component-based *modeling architecture* for developing highly reusable plug-and play model components. OSAMS is represented as a layer within the OpenMSA. OSAMS isolates and encapsulates components that are dynamically configured at run time into hierarchical service composites that can be distributed across processors in a flexible manner to transparently achieve scalable parallel performance on networks of multicore computers. One of the goals of OSAMS is to minimize the effort in constructing complex models. Past OSAMS users have reported as much as a factor of five in terms of code reduction in comparison with other modeling frameworks.

OpenCAF is as an extension to OSAMS that provides a general-purpose *intelligent* framework for modeling complex human behaviors or other cognitive processes. The OpenCAF incorporates rule-based behaviors triggering, goal-oriented behavior modeling, and five dimensional simulation for rapidly exploring multiple futures. Without OpenCAF, developers are required to manually code human behaviors, or worse, script behaviors.

OpenUTF combines the OpenMSA, OSAMS, and OpenCAF architectures into a common execution framework that can universally support high-performance scalable M&S and net-centric service-oriented operational systems within a common open architecture. The OpenUTF unifies the representation of models and services within a common parallel and distributed interoperability framework.

The **PDMS Standing Study Group** is currently investigating these and other M&S architectures, and will collectively develop a comprehensive report on the technology and implementation of parallel and distributed modeling and simulation. This report will likely be read by policy makers within the DoD M&S community and will help shape the vision for M&S. An example of this is the Chemical Biological Radiological and Nuclear Defense Modeling and Simulation Strategic Plan (JPEO-CBD 2009), which provides a clear vision for developing and integrating future M&S. The WarpIV Kernel provides an open source reference implementation of the OpenUTF.

5 OPEN SOURCE REFERENCE IMPLEMENTATION

The *WarpIV Kernel* (WarpIV Technologies, Inc. 2009) provides a freely available reference implementation of the OpenUTF and is a completely self-contained open source software framework that consists of approximately 317,000 lines of C++ and Java code. No 3rd party software is required to install and use the provided software. All mainstream operating systems are supported, including Windows (XP/Vista), Mac OS X, Linux, Solaris, HPUX, etc. (Lammers 2009). All language extensions are provided through innovative use of macros, which means that software developers are able to use their favorite editors and debuggers during development and debugging. In addition, strong interface type checking for user-defined events and methods is provided at compile time to quickly catch programming errors. OpenCAF users will therefore develop models using standard C++ programming syntax, with extensions to support the new cognitive architecture constructs. The WarpIV

Kernel reference implementation synergizes three basic capability areas into a common framework to enable the development of high-performance scalable software systems.

First, the WarpIV Kernel provides several *communications* frameworks. Examples include (a) the High Speed Communications (HSC) library that is used to support massively parallel supercomputing on networks of multicore machines, (b) an Object Request Broker (ORB) used to support distributed client/server applications, (c) grid computing used to farm tasks to workers across a available machines distributed across a network, (d) Parallel Application Launcher (PAL) that can launch distributed applications from anywhere across a network, and (e) a High Performance Computing (HPC) High Level Architecture (HLA) Run Time Infrastructure (RTI), known as the HPC-RTI that supports federations operating on multicore or massively parallel supercomputing platforms.

Second, the WarpIV Kernel provides a large assortment of *software utilities* that significantly simplify the development of complex software systems. Examples of these utilities includes (a) math utilities for vector and matrix operations, (b) motion algorithms and coordinate system transformations, (c) random number generation for a wide variety of distributions, (d) container classes for many data structures, (e) numerous curve fitting algorithms, (f) data parsing of Extensible Markup Language (XML) documents and Run Time Class (RTC) input files, (g) signal processing algorithms such as the Fast Fourier Transform (FFT), (h) Kalman Filters, (i) error handling, (j) and embedded object-oriented persistence.

Third, the WarpIV Kernel provides high-performance *compute engines* for several domains, with the most mature being its simulation engine. The simulation engine universally supports sequential, conservative, and optimistic logical-time simulation on all platforms. It also supports real-time execution that allows applications interoperate with humans, hardware systems, and other systems that are required to execute in real time. The WarpIV simulation engine was designed to integrate with other M&S standards such as HLA, DIS, TENA, and standard XML-based web services. The simulation engine executes with scalable high performance on virtually every imaginable computing configuration. The simulation engine was designed to implement the three fundamental architectures (OpenMSA, OSAMS, and OpenCAF) of the OpenUTF.

6 FIVE DIMENSIONAL SIMULATION

An important computational breakthrough, known as HyperWarpSpeed (Steinman 2008-1), was made in late 2007 that supports the exploration of multiple behavior timelines within a single execution of the simulation. This concept has been previously explored by others (Hybinette 2001), but never at the granularity of branching/splitting/merging individual behavior timelines while automatically tracking timeline dependencies of state variables. This capability provides a new paradigm for rapidly exploring many future behavior timelines with the goal of predicting future outcomes and making better decisions. This section briefly describes HyperWarpSpeed.

Normally, computer simulations are able to represent modeled systems in three spatial dimensions plus time. This is known as four-dimensional simulation. HyperWarpSpeed is able to internally spawn multiple behavior timelines at key decision or branch points during a single execution of the simulation. Coordinating how alternative timelines mutually interact within the five-dimensional world is fully automated through new multidimensional state variable data management and timeline splitting/merging algorithms. Models transparently operate and interact in a five-dimensional overlapping parallel universe as they explore alternative timelines. Computations are maximally shared between independent timelines, which gives never-before-seen performance as large numbers of decision or stochastic permutations are explored.

Perhaps a good way to understand this technology is to consider a chess game, where instead of two players taking turns exploring possible moves, you had instead, an unlimited number of players potentially exploring possible moves whenever desired. HyperWarpSpeed is able to support this kind of capability for many Force Modeling and Simulation (FMS) applications with processing speeds that are orders of magnitude faster than any other known approach.

Instead of running hundreds or thousands of independent simulation excursions to explore command decisions or uncontrollable stochastic outcomes, HyperWarpSpeed can obtain faster and more accurate results within a single execution. Carefully designing experiments to hopefully capture critical effects with a management set of runs is no longer required. In many cases, all permutations can be explored within a single run. Because independent computations that are common between parallel timelines are shared whenever possible, the execution time is typically orders of magnitude faster than traditional methods.

HyperWarpSpeed was designed to run on emerging multicore computing architectures. All computations are reversible, which means that this technology can also support real-time prediction with live data feeds that continually rewind and calibrate the modeled system. With this approach, predictions are always based on the best estimates of the dynamically changing current world state. HyperWarpSpeed may provide an ideal framework for supporting Dynamic Situation Assessment and Prediction (DSAP) for command centers of the future (Steinman 2006).

United States national defense and intelligence communities require revolutionary technology to rapidly and continually predict uncertain future outcomes based on live intelligence net-centric data feeds. It is critical for the U.S. to provide this capability faster than its adversaries. HyperWarpSpeed enables warfighting decision makers to more rapidly outthink their enemies. The ramifications of this new predictive technology for decision making are staggering. HyperWarpSpeed is an essential and distinguishing feature of the OpenCAF.

7 MULTICORE COMPUTING REVOLUTION

The OpenCAF must take advantage of emerging computing technologies. The multicore computing revolution has begun and will soon demand a paradigm shift in how software is developed, tested, and deployed. Chip manufacturers have hit the performance wall. Three primary technical factors limit the ability to make software running on single CPU microprocessor architectures execute faster.

1. Power and overheating considerations make it difficult to increase CPU clock speeds beyond current capabilities. Power consumption goes up by a factor of four when doubling the clock speed.

2. Memory access is a bottleneck and cannot be easily optimized any further. Multilevel caching techniques are used to reduce the overhead of memory access bottlenecks, but there is little room for further improvement.

3. Instruction-level parallelism (e.g., pipelining, branch prediction, and hyper-threading) cannot be easily optimized beyond the current state of the art. Performance gains from single CPU chip designs have apparently reached their limit.

Even if one or two of these performance walls could be improved, all three must advance together to minimize bottlenecks and achieve actual performance gains for typical software applications. There is only one foreseeable way to improve computational performance. The universally accepted path forward is to pack multiple CPUs, known as cores, onto a single chip. Because chips can be manufactured with multiple layers, there is virtually no limit to the number of cores that can be provided on a single chip. Thus, the multicore computing revolution has begun. Hardware, operating system, and commercial software vendors alike are all competing to rapidly bring these technologies to market.

Experts uniformly agree (Manycore Computing Workshop 2007) that parallel programming is much too difficult for the average software developer and that a new suite of powerful tools must be provided to abstract away the associated programming difficulties. A suite of parallel software development tools might include (a) programming frameworks, (b) new languages or extensions to existing languages that automate parallelism through embedded services and/or programming constructs, (c) new optimized compilers that automatically parallelize code whenever possible, (d) transactional memory mechanisms that automatically coordinate reading and writing between threads or processes, (e) optimally parallelized math libraries for supporting massive number crunching, and (f) improved communication libraries that simplify communication between processors through shared memory. Hardware and software vendors alike are now feverishly working to bring these capabilities to market. Fortunately, an open source reference implementation of the OpenMSA and OSAMS exists today that is readily available for use by the United States government and other agencies. The OpenCAF needs to be developed as an extension to OSAMS to complete capabilities required by the OpenUTF.

It is critical for the United States to maintain its information processing edge over its adversaries, especially in its fight against global terrorism. The United States Defense Department must begin utilizing parallel processing techniques to (a) speed up calculations, (b) provide higher fidelity results, (c) perform more computations, (d) analyze more data, and (e) simulate larger scenarios. The world of software is changing rapidly and will soon demand new programming techniques to embrace the coming revolution. For example, the following excerpt taken from a recent news article from the Associated Press, signifies the importance of developing new scalable software solutions to address the emerging multicore hardware revolution. Note that the *gigantic advantage* described below directly applies to United States Warfighters.

"New Generation of Processors Presents Big Problems, Potential Payoffs for Software Industry."

… Experts predict dire consequences if the software for more complicated applications isn't brought up to speed soon. They warn that programs could suddenly stop getting faster as chips with eight or more cores make their way into PCs. The software as it's currently designed can't take advantage of that level of complexity.

"We'd be in uncharted territory," Patterson said. "We need to get some Manhattan Projects going here – somebody could solve this problem, and whoever solves this problem could have this gigantic advantage on everybody else."

Jordan Robertson, Associated Press, July 22, 2007.

8 OPEN UNIFIED TECHNICAL FRAMEWORK

The United States military is undergoing a major technology transformation to empower warfighters of the 21st century with global and secure access to critical information anytime and anyplace. Providing this net-centric warfighting vision goes well beyond simply interconnecting networks of distributed systems into more complex systems. Besides requiring a robust global communication capability, tapping the full potential of the emerging Global Information Grid (GIG) requires the careful integration of several key software technologies, methodologies, ontologies, and interoperability standards.

On the other end of the spectrum, multiple requirements and emerging high performance computing technologies are simultaneously converging that argue for migrating to a common Open Unified Technical Framework (OpenUTF) for hosting both operational net-centric systems and high-performance M&S applications. Such a framework would be required to provide the following general capabilities:

1. Support scalable high performance simulation on emerging networks of multicore computers operating within a net-centric operational battlefield environment. Scalable run time performance should embrace emerging multicore, net-centric, and software technologies, along with a flexible model composition strategy.

2. Provide interoperability with existing simulations, models, test articles, and humans in Live, Virtual, and Constructive (LVC) execution environments. Both real-time and logical time execution modes must be supported. Mainstream LVC standards include the High Level Architecture (HLA), Distributed Interactive Simulation (DIS), Test and Training Enabling Architecture (TENA), and Synthetic Environment Data Representation and Interchange Specification (SEDRIS).

3. Automate the ability for simulations developed within a standards-based framework to provide net-centric Service Oriented Architecture (SOA) web-based interfaces. These include XML-based standards such as Web Service Description Language (WSDL), Simple Object Access Protocol (SOAP), and Universal Description Discovery and Integration (UDDI) web standards.

4. Provide cost-effective methodologies and modeling constructs to support the development and integration of highly interoperable and reusable model-components. Such standards will foster the creation of maintainable and validated plug-and-play model-component libraries. Configurable simulations can be configured from these model-components to efficiently satisfy specific M&S needs and requirements.

5. Provide a flexible user-definable way to map individual models, collections of composite models, and legacy simulation tools to processors, machines, and supercomputers connected across local and wide area networks. The goal is to support scalable parallel and distributed simulations executing on machines ranging from laptops to multicore supercomputers connected across the Global Information Grid (GIG).

6. Provide a cognitive architecture framework able to represent complex human behaviors without resorting to rigid entity scripting or complex decision logic. The cognitive architecture must support human behavior representation, cognitive thought triggering, goal optimization, and decision support in live operational settings.

7. Provide the ability to rapidly develop scenarios, model composition tools, data visualization tools, and statistical analysis utilities that operate in a common manner independent of any specific application or implementation. Common tools and utilities will greatly simplify the operation of CBRND simulations.

The OpenCAF will join the OpenMSA and OSAMS to form the OpenUTF (see Figure 1), which is still evolving. This means that the cognitive architecture capabilities outlined in this paper can be used for a wide variety of applications and not just M&S. The DoD net-centric research community is largely focused on interoperability between systems of systems, enterprises, and M&S federations. Yet, software developers across a wide spectrum of application domains fundamentally build services and/or model-components. The difference in interaction overheads between enterprise technologies and direct interactions through low-level method or function calls is six or seven orders-of-magnitude.

The OpenUTF must have the flexibility to compose reusable software components (e.g., Model Components) into composites for tight model-to-model interactions. These composites (e.g., Distributed Entities) are distributed across processors and machines to achieve parallel speedup. The OpenUTF must be able to compose distributed simulations, federations, enterprises, and systems of systems in the most flexible manner possible. The location of models, services, composites, simulations, federations, enterprises, and systems of systems must be factored into the overall composition of any distributed capability where performance plays an important factor.

Within the OpenUTF, the fundamental plug-and-play building block is the software component. Components interact through (a) abstract interfaces that are specified independent of any particular implementation, and (b) publish/subscribe services that distribute state information across the fully integrated system. These concepts are similar to the fundamental HLA interoperability mechanisms, but implemented at the component level with streamlined interfaces where software development really occurs.

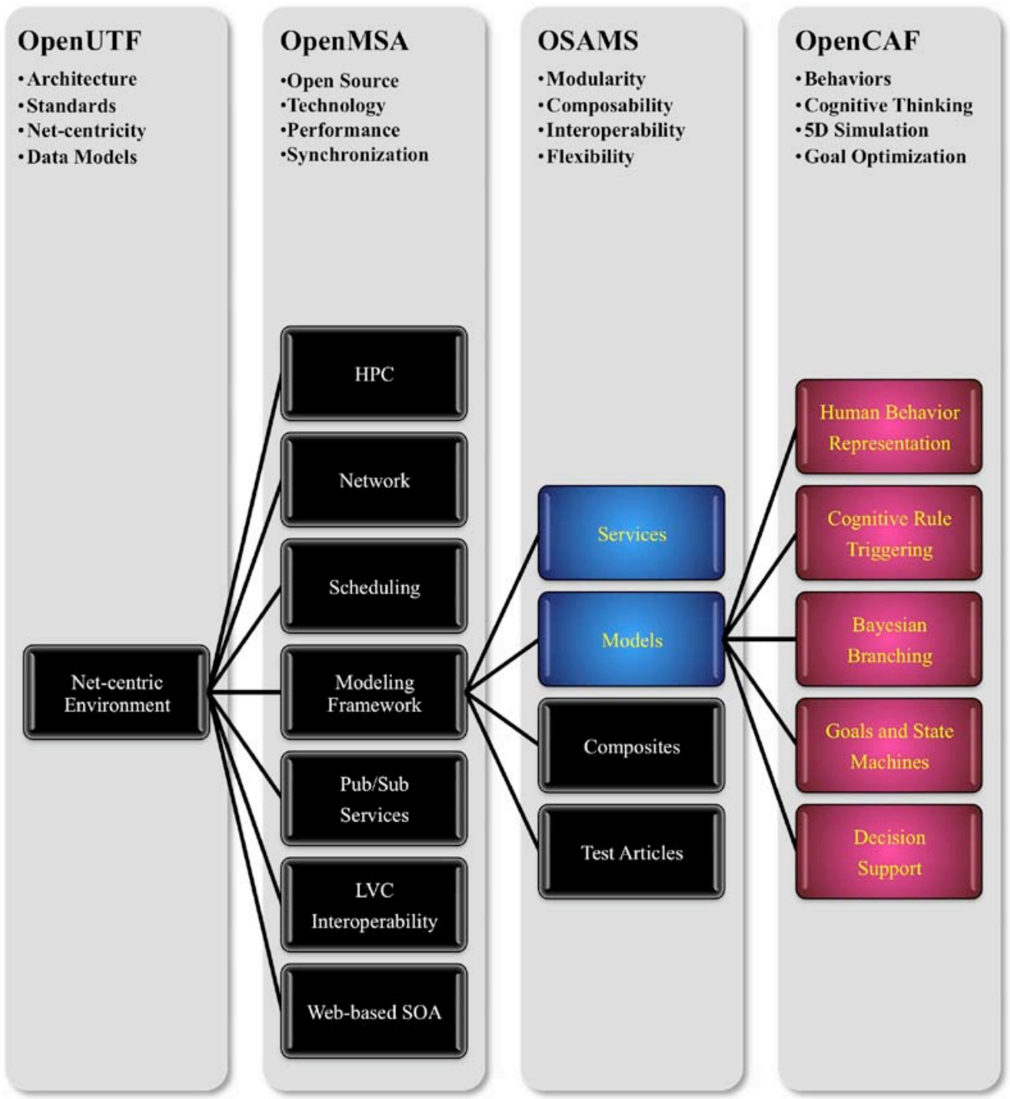

Figure 1: Conceptual view of the Open Unified Technical Framework (OpenUTF). OpenMSA defines the technology layers for supporting scalable and interoperable application execution on a wide variety of computing architectures. OSAMS provides the core plug-and-play software constructs to support composable model/service interoperability. OpenCAF provides the framework for supporting intelligent cognitive behaviors. OpenUTF combines these three architectures to provide a unified high-performance, net-centric, scalable, standards-based, interoperable execution architecture that transparently supports both models and operational services within a common architecture.

ACKNOWLEDGEMENTS

The authors of this paper would like to thank Dave Ferris, Chief Systems Engineer, Joint Program Executive Office for Chemical and Biological Defense (JPEO-CBD), who directs the Software Support Activity (SSA) Modeling, Simulation, and Integration (MSI) Team. The authors of this paper would also like to thank both Doug Hardy, Software Support Activity program manager, and John Wrigley, Technical Lead of the MSI Team for their encouraging support of the OpenMSA, OSAMS, and OpenCAF concepts. Members of the MSI Team involved in establishing the vision for the OpenUTF and creating the M&S Strategic Plan for CBRND are Alex Vagus, Ken Kaiser, James Campbell, and Yvonne Rocco. Their help in reviewing this paper is greatly appreciated.

The authors of this paper would like to thank Karen Roth, Dr. Timothy Busch, and Dawn Trevisani for their continuing support and technical insight in real-time Air Force command and control operational systems requiring Dynamic Situation Assessment and Prediction (DSAP). Their collective insight has helped shape much of the HyperWarpSpeed concept.

The authors of this paper would finally like to thank Bob Pritchard, Program Manager of the Joint Virtual Network Centric Warfare (JVNCW) program at SPAWAR Systems Center, for encouraging the development of many of the LVC and web-based interoperability capabilities for the OpenMSA. Bob Pritchard has also been a strong supporter of the PDMS Standing Study Group, along with open source software methodologies.

REFERENCES

Hybinette, M. and Fujimoto R. M. 2001. Cloning parallel simulations. *ACM Transactions on Modeling and Computer Simulation*, 11: 378-407.

Joint Program Executive Office for Chemical and Biological Defense (JPEO-CBD) 2009. Chemical Biological Radiological and Nuclear Defense Modeling and Simulation Strategic Plan, Draft Version 0.6., 2009

Lammers C., Valinski M., and Steinman J., 2009. Multiplatform Support for the OpenMSA/OSAMS Reference Implementation. In proceedings of the *Fall 2009 Simulation Interoperability Workshop, 09F-SIW-033.*

Langley, P., Laird J., and Rogers, S., 2009. Cognitive Architectures: Research Issues and Challenges. *Cognitive Systems Research*, 10: 141-160.

Lehman J., Laird J., and Rosenbloom P., 2006. A Gentle Introduction to Soar, An Architecture for Human Cognition: 2006 Update. University of Michigan, 2006.

Manycore Computing Workshop, 2007. Sponsored by Microsoft. Participation was by invitation only. Available via <`http://science.officeisp.net/ManycoreComputingWorkshop07/Program.aspx`>.

SISO Parallel and Distributed Modeling and Simulation Standing Study Group (PDMS-SSG) 2009. Available via <`http://www.sisostds.org/index.php?tg=articles&idx=More&article=543&topics=152`>.

Steinman J., Wieland F., 1994. Parallel Proximity Detection and the Distribution List Algorithm. In proceedings of the *1994 Parallel And Distributed Simulation Conference*. Pages 3-11.

Steinman J. 2005, The WarpIV Simulation Kernel. In proceedings of the *2005 Principles of Advanced and Distributed Simulation (PADS) conference.*

Steinman J., and Busch T., 2006, Real Time Estimation and Prediction Using Optimistic Simulation and Control Theory Techniques. In proceedings of the *Spring 2006 Simulation Interoperability Workshop, 06S-SIW-017*.

Steinman J. 2008-1. Simulating Parallel Overlapping Universes in the Fifth Dimension with HyperWarpSpeed Implemented in the WarpIV Kernel. In proceedings of the *Spring 2008 Simulation Interoperability Workshop, 08S-SIW-025*.

Steinman J. 2008-2. Open Source Licensing and the WarpIV Kernel. In proceedings of the *Fall 2009 Simulation Interoperability Workshop, 08F-SIW-065*.

Steinman J., Lammers C., and Valinski M., 2008-3. A Unified Technical Framework for Net-centric Systems of Systems, Test and Evaluation, Training, Modeling and Simulation, and Beyond... In proceedings of the *Fall 2008 Simulation Interoperability Workshop, 08F-SIW-041*.

Steinman J. 2009, Introduction to Parallel and Distributed Force Modeling and Simulation. In proceedings of the *Fall 2009 Simulation Interoperability Workshop, 09F-SIW-021.*

TacAir Soar 2009. Available via <`http://www.soartech.com/projects/TacAir-Soar.pdf`>.

WarpIV Technologies, Inc. 2009. Available via <`http://www.warpiv.com`>.

AUTHOR BIOGRAPHIES

JEFFREY S. STEINMAN is the President and CEO of WarpIV Technologies, Inc. He received his Ph.D. in High Energy Physics from UCLA in 1988. In 1990, Dr. Steinman developed the Synchronous Parallel Environment for Emulation and

Discrete Event Simulation (SPEEDES) framework that eventually transitioned to industry and several mainstream programs including BMDS SIM, JMASS, JSIMS, and EADTB. Dr. Steinman currently provides technology and open systems architecture support for the JPEO-CBD, SSA, where he is helping to establish and implement its long-range M&S strategic plan. Dr. Steinman has published more than 60 papers and articles in the field of high performance simulation, has five U.S. patents in high performance M&S technology, is a frequent invited speaker at conferences, seminars, and forums, and is the principle designer/developer of the open source WarpIV Kernel Reference Implementation of the OpenMSA, OSAMS, and OpenCAF architectures. Dr. Steinman is currently heading the newly formed Parallel and Distributed Modeling and Simulation Standing Study Group (PDMS-SSG) that operates within the Simulation Interoperability Standards Organization (SISO). Dr. Steinman can be contacted at `<steinman@warpiv.com>`.

CRAIG N. LAMMERS is a Vice President and Senior Analyst at WarpIV Technologies, Inc. He is currently providing technical support for the JPEO-CBD, SSA, and is the program manager of an effort with AFRL, Rome Labs conducting new research and development in parallel and distributed simulation to support formal estimation and prediction of complex systems using the WarpIV Kernel. His professional interests are in the areas of modeling and simulation, artificial intelligence, user interface design, and music. Mr. Lammers received his Bachelor of Science degree in Industrial and Systems Engineering from the University of Michigan, Dearborn in 2001, where he graduated with high honors. Mr. Lammers can be contacted at `<craig.lammers@warpiv.com>`.

MARIA E. VALINSKI is a Vice President and Senior Software Engineer at WarpIV Technologies, Inc. She is currently providing modeling, simulation, and integration support for the JPEO-CBD, SSA, and is the lead software engineer for the Joint Virtual Network Centric Warfare (JVNCW) program that involves modeling wireless communications on supercomputers using the WarpIV Kernel for the Navy, Army, and Air Force. Ms. Valinski received her Bachelor of Science degree in Computer Science from East Stroudsburg University in 1997. Ms. Valinski can be contacted at `<maria.valinski@warpiv.com>`.