A Real-time, Cloud-based Architecture for Integrated Intelligence Driven Operations

Jeffrey Wallace, Ph.D.
I2 Sentinel Associates, Inc.
Fairfax, VA 22030
wallacej@i2sentinel.com

Professor Thomas Preston
George Mason University
Fairfax, VA 22030
prestoth@nro.mil

Abstract— Many “big data” software systems are not interactive, automated, or run in a real-time mode. The true utility of cloud computing and “big data systems” can be increased by providing an execution framework and control software that is native to cloud architectures and supports interactivity and time synchronization. In addition, a framework to integrate different artificial intelligence and machine learning algorithms is combined with the execution framework to create a powerful cloud computing system development platform.

Keywords—cloud computing; analytics; complex system representation; integration; interoperability; conceptual graphs; prediction; intelligent systems; anticipatory intelligence; timing; synchronization

1. INTRODUCTION

Current cloud computing application development and systems integration suffer from the following problems:

- **Batch oriented** – the systems are not made to run interactively, or continuously.
- **Brittle** – easily fail with slight perturbations to the information transacted
- **Difficult to maintain** as systems are upgraded
- **Complicated to scale** when adding additional information and constituent systems
- **Cannot provide clear, concise implementations** among diverse sub-system sets

Developing operationally relevant computational capabilities requires access to a more cost-effective, compute-focused path. Some jobs or tasks require high performance computing platforms and some do not – the goal is to optimize execution based on available assets. This will be accomplished using an orchestrating system leveraging the runtime performance, throughput and features of enterprise supercomputing platforms. Additionally, this architecture and its realization employ highly optimized software coding, algorithms and applications, specific hardware, benchmarks and logic workflows to achieve the premier supercomputing conditions.

Consolidation and execution of high performance and supercomputing (HPC-SC) systems – the goal is to provide a heterogeneous environment where resource and task are optimally matched. Understanding the interplay between algorithm and hardware is key to optimization at the architectural level.

A cloud application development and integration framework for intelligent systems, called the Joint Execution Environment (JEE) [1], address these challenges. JEE’s biologically inspired characteristics have the ability to:

- Create complex, realistic, and scalable networks of component inter-relationships
- Distribute autonomous controls and monitors
- Implement complex webs of cause and effect
- Dynamically alter the execution structure
- Adapt and evolve the system.

The goal is to build systems where decision algorithms are integrated and interoperable with a wide variety of system components. Such “intelligent algorithms” can now be capable of implementing robust monitoring and active control systems that have human oversight and intervention. The speed of the battlespace required that systems anticipate and handle that gap in time when data and measurements exist to draw a conclusion with a high probability that something is going to happen, and any human decision-maker drawing the same conclusion.

Systems, responses, and countermeasures can now be activated in advance of events that negatively affect operations. Consider the task of controlling a ground vehicle in an urban environment. Standard “rules of the road” provide the example for one type of behavior – that of following either crisp or fuzzy rules via an inference or rules engine. Now consider the case where children are present. The system must recognize the presence of children, which is most efficiently calculated employing a fundamentally different algorithm type than a rule or inference system.

To implement a robust system response, or anticipatory intelligence, advancement beyond traditional Bayesian networks, sometimes referred to as...
belief networks, is required. A Bayesian network\textsuperscript{1,2}, roughly speaking, would say that since the probability is higher of something suddenly coming into the vehicle path, given the presence of children, speed should be lowered and a higher scan rate/decision cycle until the hazard is passed.

As stated, entity extraction from unstructured data, or recognizing something from an image of video stream, is a set of algorithms and processes that must be coordinated in the context of other decision algorithms and processes, such as feeding the Bayesian network monitor for road hazards. The goal is to integrate multiple algorithms as appropriate to provide robust automated systems with anticipatory intelligence. What is required is an approach that naturally permits algorithm integration, in addition to system component integration and interoperability.

In addition, JEE addresses the challenges by fusing:
- Advanced systems theory and practice
- Advanced software development
- Low-latency, high throughput, reliable, and robust computer communications
- Sophisticated software integration, interoperability, and synchronization

Common approaches to complex system infrastructure, such as systems based on Microsoft’s .NET framework \textsuperscript{[5]}, process-based programming (e.g., systems utilizing threads, semaphores, and locks) \textsuperscript{[6][7][8]}, object request brokers \textsuperscript{[9][10]}, ERP infrastructure \textsuperscript{[11][12][13]}, and cluttered web-based technologies, \textsuperscript{[16]} fail in one or more of the problem areas listed above. The tremendous number of constructs causes significant setbacks with most application development and integration methods. JEE (with a macro-based sub-language) easily represents and constructs these complex system capabilities.

As outlined by the Chairman of the Joint Chiefs of Staff \textsuperscript{[2]}, it becomes imperative to support mission command by effectively leveraging, and/or developing and distributing the requisite technological solutions to align with situational and regional needs. Reference Section 6, Sub-Factor 4: Innovation and Vision, Paragraph 1.

2. MULTI-INT ARCHITECTURE

This architecture enables intelligence community (IC) and Department of Defense (DoD) standards including the use of the IC Trusted Data Format (TFD)

![Diagram](image)

**Figure 1 Real-Time, Multi-INT Architecture.** The architecture solution improves information content and flow to the warfighter using superior regional domain awareness through Joint Execution Engine (JEE).

\textsuperscript{1} http://en.wikipedia.org/wiki/Bayesian_network

\textsuperscript{2} http://www.dmoz.org/Computers/Artificial_Intelligence/Belief_Networks/Software/
and smart data tagging schemas including the
Enterprise Data Header (EDH), Access Rights Header (IC-ARH) and Information Resource Metadata (IC-IRM) as components of the Standard Data
Representation 2.0.

Appropriate interfaces to the IC’s Unified
Authorization and Attribute Service (UAAS) federated
Identity and Access Management (IdAM) service
(ICITE/AA provider) and the community cloud,
widget, widget interface and data interface standards
are available.

Collaborative integrated intelligence efforts within
the IC (notably object based production(OBP), Red
Force Tracker (RFT) pilot) to facilitate unifying
information, conforming to the IC in shaping and
organizing the “known,” as Object Based Targeting
(OBT) expands and scales to new intelligence domains
called INTs), and object types will increase the
effectiveness of monitoring the areas of responsibilities
(AORs). Our approach implements Object Management
as a Service (OMaaS) to enable users, through war
gaming and discovery for governance, policy, security
and accountability processes associated with OBP.

In particular, intelligence can be linked to known
Objects facilitating semantic methods and
understanding. This approach will expand OBP
capability to the any AOR, improving situational
awareness through large object identification to include
named areas of interest, missile and launch facilities.
Additional INTs may include near real-time video,
infrared and MASINT to improve situational
awareness. The objective is to establish end-to-end
performance of calculating new knowledge and
determine how to send new information to the Global
Command and Control System (GCCS) for improved
understanding.

2.1 Real-time Access and Understanding of Multi-
agency Data

Real-time access and understanding of intelligence
data is the hallmark of this approach. The infrastructure
has been architected to enable true real-time system
integration, interoperability, and implementation. Every
algorithm has been extensively analyzed and its
implementation optimized.

A key differentiator to existing enterprise offerings
is the ability of this infrastructure to be ported to any
operating system and compiler set. This enables direct
integration between IT, C4ISR, weapons systems, and
emerging unmanned systems in a way previously not
possible.

Several technologies are combined to implement
real-time access and multi-agency understanding. A key
implementation feature is the incorporation of
Accumulo product within an enterprise hardened
commercial version of the Hadoop technology
ecosystem. Our approach unifies the current efforts by
various agencies and leverages commercial versions of
technologies that include advanced enterprise features
focused around performance, reliability and
availability. The improvement in this version of
Hadoop is the elimination of a single name node. We
eliminate a key bottleneck in the existing architecture
and enable a system that delivers linear performance as
the size of the system grows.

Our implementation also features an expanded set
of ingest APIs and methods that permit real-time
streaming and other standardized file and database
system interfaces. These factors improve performance,
integration, reuse, and interoperability.

Expanded interfaces also permit the usage of a
diverse commercial analytics, data, analysis, and data
management tools. Our technical approach also permits
the usage of high performance solid state storage
technology that can revolutionize the manipulation and
analysis of large objects and real-time, or near real-
time, performance of data analytics. A HPC-SC event
service bus is employed to enable data analytics scheme
to be implemented. The event bus infrastructure enables
programming complex orchestration while delivering
real-time performance. The nature of the bus permits
inter-architecture integration between multiple systems
simultaneously.

2.2 Advanced Analytics Capability to leverage High
Performance Environment

Tailored algorithm implementation is the key to
optimizing advanced analytics for HPC-SCs to
understand how to phrase the analytic algorithms to
avoid processing bottlenecks. Algorithm optimization
occurs in several different parts of the architecture
beginning with the HPC-SC event service bus. The
implementation enables event queue distribution over
the processing nodes and coordinates an advanced
synchronization algorithm. We support data marshaling
between the data sources and stores in the analytic
algorithms and integration and interoperability between
analytic algorithms and entire analytic architectures.

This technical implementation permits the
federation of heterogeneous analytic architectures in
real-time providing not only improved performance but
the ability to utilize all interagency and operational
analytics.

A wide variety of libraries of algorithms, data
structures, and message distribution and
synchronization constructs which are optimized for
HPC-SC infrastructure will be provided. This enables
analytics to be parallelized leading to drastically
reduced run times. Analysis performance can also be
enhanced by the inclusion of graphics processing
engine providing the system with the ability to support
emerging GPU APIs that can support the type of vector
and matrix operations required by advanced analytics. Such a polymorphic computing architecture optimizing algorithm implementation and execution, in addition to algorithm scheduling and synchronization, provides a comprehensive approach to the full utilization of HPC-SC resources.

2.3 Command Support

Central to the proposed architecture and implementation is the ability to integrate and interoperate with existing military command information, C2, and C4ISR systems. This is accomplished thru the external module interface framework in the overall HPC-SC event service bus framework, Object Based Production, Red Force Tracker, Army Advanced Analytics, GCCS I3 COP, NGA GEOINT, and other OZONE Widget capabilities.

This implementation includes a gateway for GCCS, Army Battle Command System (ABCS) publish and subscribe system, Command and Control PC (C2PC), Command Post of the Future (CPOF) and the Theater Battle Management System (TBMCS). This will provide a framework and re-useable code base intended to incorporate other sensor interfaces and systems as necessary for instance, integrating the sensor suite from the Global Hawk/Trident surveillance programs.

The ability to integrate new analytics and permit composition to meet emerging requirements is fully optimized and supported by the framework. For example, systems multiple domain analytic systems will be integrated and incorporated easily. The framework has a natural spot for the inclusion of security and the ability to support cross-domain solutions. This approach will also revolutionize the use of personnel assets in terms of their location in the Battlespace / AOR.

2.4 Advanced Analytical Capabilities, Framework, Data Layer, and High Performance Computing System

This approach permits the integration or implementation of any new analytic tool or algorithm. This effort will provide a framework for the ingestion of all source data, both structured and unstructured; expand the heuristics for problem solving, learning and discovery, and provide mediation and normalization unifying access to and analysis of all data. Specifically this effort provides several new commercial analytics tools and utilities.

A novel video and sensor analysis classification and metadata tagging tool and framework will be provided so that for the first time, full motion video (FMV) and other sensors can be classified and metadata tagged upon ingest, and both the object and the metadata can be streamed into a unified Data Layer. This increases the value of the data for a wide variety of analytics and provides critical improvements to intelligence and operational decision systems and processes, leading to faster production of decision quality data and acceleration of the mission command and the decision process.

A variety of analytics engines are featured that will be employed capable of creating indexing schemes which can be used separately or combined in the decision process. This capability will lead to the advancement of analysis, intelligence and decision processes in new ways. For example an index can be created based upon the ingestion of a variety of classified data sources and integrated with indexes from other departments or classification levels without compromising source data or methods and improving analytics queries.

Combining capabilities together is the HPC-SC event service bus and associated HPC-SC optimized support libraries, packages, and utilities. This team will work with both commercial vendors and relevant Government and academic analytics packages and technologies to improve their native utilization of HPC-SC technologies and development methodologies to enable individual analytic tool performance.

2.5 Trusted Systems

One of the true challenges is the computationally efficient trusted systems. The need to constantly enforce security policy at every stage of operation is a significant overhead. The JEE provides a set of execution control primitives that are “lightweight” from the perspective of memory and processor cycle consumption.

The development of trusted systems is clearly associated with three driving development areas:

1. Enterprise testing
   a. Create the necessary baseline
   b. Freeze the baseline and testing
   c. Maintain the state

2. Representing composite relationships within the cloud
   a. Functions and functionality
   b. Identify where the work is being performed
   c. Understand how we establish trust in the cloud

3. Implement trust in the cloud
   a. Emphasize workload and metrics
   b. Constant software modeling, simulation, and coding from test to operational readiness
   c. Code management
   d. Repeatable, auditable, and demonstrable
This architecture and implementation demonstrates how cloud and system tuning can be simplified by allowing all aspects of the hardware and software infrastructure to be optimally accessed. State-of-the-practice enterprise systems always want to define their own “pure” architecture based on some seemingly relevant philosophical concept. This approach allowing the most efficient system functions, and when trust is implemented the systems perform poorly.

Regarding how cloud computing techniques and technologies could be used to enhance interoperability and improve the sharing of geospatial information across a broad range of users and devices, the architecture allows both high performance and resilience to system failure. Geospatial information sharing in operational environments must be resilient to infrastructure attacks and failure. This approach allows not only highly available “traditional” resources in terms of compute and data storage, but also in terms of communication. In all other systems, a communication link problem results in data transmission failure, whereas this architecture and implementation feature the ability to have completely reliable, yet efficient, data communication.

This approach provides a consistent and secure single map solution for appropriate users through open APIs. A wide variety of messages, formats, and protocols can be used based on what different communities independently support. This avoids costly enterprise standardization, and focuses on letting communities immediately interoperate, while providing a smooth evolution path to increased shared development operations (DEVOPS), resulting in a natural technical convergence. Legislated, top-down convergence efforts rarely succeed, and when they do it is only at great cost.

This architecture also enable success in the development of broad international collaboration efforts by virtue of the flexible nature of data sharing APIs and methods. Many partner nations utilize a large quantity of legacy systems, and modern enterprise approaches do not easily support older systems and alternative architectures.

3. ARCHITECTURE IMPLEMENTATION

The architecture shown in Figure 1 will improve information content and flow to the warfighter, enabling superior regional domain awareness through a robust synchronization capability inherent in the Joint Execution Engine (JEE) software component. This is provided as an optimizing and enhancement mechanism to the HPC-SC and Enterprise Service Bus (ESB) functionality currently used in the IC. The JEE introduces a mediation layer that effectively enables introduction of analytics to Map Reduce in the cloud. This also introduces a multi-source input, a co-creative environment and other commercially developed efforts in guided information search/discovery/filtering to optimize information flow to the warfighter.

Data flow modeling is employed to document flow through the interface to ensure data credibility from source to product. Through this interface we will ingest C4ISR resources to support strategic and tactical data into the data layer to include emerging advanced analytics widgets. Our proposal highlights innovation in several dimensions that address requirements as set in the above. The design and implementation of the Data Layer delivered via the JEE and its integration of key products, unifies the inter-agency Data Layer strategies.

The Data Layer ingest is significantly expanded to permit streaming data, POSIX file system files, and existing database standard interfaces. The implementation provides an enterprise level by the automation of mirroring and other persistence services in the Data Layer. The architecture supports the existing analytics and Widget framework.

The scope of real-time support is extended by employing a high performance, real-time event engine optimized to provide real-time, continuous analytics support to the Warfighter. Existing C2 and C4ISR systems and envisioned future Warfighting capabilities including unmanned systems, future sensors, automation frameworks and intelligence sources are supported. The HPC-enabled event bus permits the orchestration of widgets, Map Reduce jobs, any commercial analytics, other analytics models (i.e., SQL, POSIX, or JDBC/ODBC), and simulations as desired by an AOR or IC user.

3.1 JEE Architecture Description

JEE is an object-oriented, event-based, high performance execution system. JEE [1] provides high speed communications, which is central to its framework [2], and utilizes numerous messaging fabrics for inter-processor communication: shared memory, wireless, fiber optic, ATM, TCP, IP, and multicast (implemented in a variety of media). The Communications Services provide a variety of mechanisms linking clients to intelligent application services, or the hosting processors. Communications Services API (internal and external) is standardized for simplified integration [1]. An abstraction is then supported for unicast and multicast, and permits various implementations, and thus protocols, to work simultaneously.

State-Saving Framework and State-Saving Services support reliability, synchronization, fault-tolerance, and implementation of persistence services. The Core Programming Services provides Standard Template Library (STL) programming API, and utilizes the state-saving and persistence features.
Event Management Services (EMS) provide high-performance data structures to develop exceedingly complex, reliable, interactive intelligent applications more rapidly.

Synchronization Management Services, closely coupled with EMS, control synchronization and timing, which are essential for real-time intelligent applications interfacing with hardware [2].

Standard Application and Integration API (SAIA) expedites development of complex, robust intelligent applications by code generation macros and APIs. SAIS synchronizes components as the overall system executes, and scales to simultaneously execute large numbers of components (an interactive synchronization mechanism hides programming complexity).

Distributed Object Management Services and Data Translation Services provide location transparency and a powerful, yet easy to use, distributed object computing framework (e.g., the complexity in using different inter-processor communications).

- Logical constraints, consequences, and role-based behaviors entailed by the relationships
- A system of primitive relational abstractions
- Method to change abstractions into conceptual hierarchies
- Means to compose and structure relationships into frameworks and architectures

JEE utilizes additional state of the art technologies in its architecture: Knowledge Execution Engine (KEE) and Knowledge Representation Integration Infrastructure (KRII) employ the lower layers for location transparency, and are based on conceptual graphs (CGs) [3].

Collectively, the extensions embodied in JEE signify precision in the art. All services adhere to the External Integration Framework, and easily integrate intelligent applications and WAN (Figure 2) [1].

4. INTELLIGENT SYSTEMS

The JEE utilizes CGs [1] to integrate hardware and software systems. Visually, a CG mimics knowledge representation in common diagrams for discussions (using whiteboards, slides, or even table napkins). The diagrams, or drawings, are often text snippets (typically enclosed in squares or ovals) and lines (such as labels) connecting one snippet to another. Experts often use visual aids to quickly communicate complex details during brainstorming sessions (see Figure 4).

In CGs, text snippets (in a square) are called Concepts. The line connections are enhanced with ovals, called Relationships, containing additional text. Hence, representation of semantic relations, between various concepts, occurs in a manner consistent with common "brainstorming pictures". Actors (diamond shaped symbols) provide a method to encapsulate interfaces to hardware or software components, and indicate data or signal transforming activity is occurring.
Structurally, a CG provides the following advantages for representing and integrating complex systems:

- Inherently hierarchical; permits operation at increased aggregated levels, when beneficial
- Can decompose components to appropriate levels of detail, to meet requirements
- Ability to conceptualize the entire system, or one specific concept
- Capable of capturing any aspect of the system

Using CG’s, JEE simplifies hardware and software component integration; and simultaneously, concisely represents the entire system control logic.

Developers typically manage functional blocks with various data sending and receiving protocols. However, most developers lack standard approaches across the enterprises (usually one method works as well as the next). Consequently, accurate prediction of time and cost is difficult. JEE provides an organized template for encapsulating functional blocks (hardware or software), which is easily mastered by integrator engineering staffs, thereby solving the prediction challenge.

CG Relationships have simple rules - one concept node must be connected by an incoming arc to a relationship; and one concept node must be connected by an outgoing arc. Relationship nodes provide critical semantic structure to system descriptions, and frequently represent modifiers, qualifiers, and constraints.

Concept nodes may be connected to relationship nodes or actor nodes, but direct connections between concept nodes is not permitted. A concept node may have any number of incoming arcs or outgoing arcs, and represent a variety of system features. Concepts may be components (or objects), or they may represent actions (or verbs). Relationships act as modifiers (adjectives and adverbs).

Actor nodes can have any number of incoming arcs from other nodes. However, they have only one outgoing arc. The outgoing arc may be connected to another actor node, or a concept node. Actor nodes provide the critical ability to encapsulate hardware or software components. Hence, actor nodes enable system integration.

One of the purposes of the Knowledge Execution Engine (KEE) involves controlling the execution of a collection of hardware and software components, as a cohesive and robust system. Relationships have the simplest rules - one concept node must be connected by an incoming arc to a relationship; and one concept node must be connected by an outgoing arc. Relationship nodes provide critical semantic structure to system descriptions. They frequently represent modifiers, qualifiers, and constraints.

5. IMPLEMENTATION EXAMPLES

One of the central tenets of the JEE is to make the programmers job simpler by virtue of API design. The act of sending data from one component to the other can often be a challenging task in modern enterprise implementations. Monolithic, complex data models must be maintained, and complied with, leading to what practically results in an expensive and slow moving social engineering process.

In addition, the state-of-the-practice for the implementation of data interoperability has resulted in a philosophy that requires internal data structures to be rectified and flattened into text files in XML syntax. Ideally, the natural data structures used by developers would be sent, and there would be no monolithic data model. Any one component typically only required the interaction with a small set of other components, so practically speaking this should be decoupled technically and in process implementations.

5.1 System Synchronization

One of the most difficult tasks is the precision implementation of causality between system components. The current state-of-the-practice in real-time systems focuses on the accurate control and estimation of the time to complete a function. The JEE provides such timing constructs, and extends the paradigm to integrate logical controls with timing. This permits the notion of an interrupt to a timed component, a strictly logical synchronization of components in a real-time environment, and component synchronization timeouts.

Consider the following example where the firing of a gun is controlled over the network or locally. In some fashion the gun engages a target by moving the barrel into position and then firing the weapon. The process of moving a gun into position can be expressed and
implemented with `slew` and `elevate` functions, as indicated in the figure below.

![Fire When Slew and Elevate are Complete](image)

Figure 5. Gun dynamics

The code fragment to implement this is:

```c
void Turret::fire()
{
    // Wait until the turret movement is completed
    WAIT_FOR(1, slewComplete, -1);
    WAIT_FOR(2, elevateComplete, -1);
    // Fire the weapon, this would activate the real gun
    Fire_M256();
    RB_cout << "Flash, Boom, Bang, Echo" << endl;
    fireComplete = 1;
}
```

The key to ease of programming is illustrated by the use of programming macros to generate code patterns and implement transparent semantics. The WAIT FOR macro implements a thread-like suspend and resume, but utilizes the local stack defined in the P_VAR-P_BEGIN block, which drastically reduces memory from megabytes to store the stack frames normally used in thread packages to do the same function to bytes.

The `slewComplete` and `elevateComplete` are types of semaphores, and can be released by network or local API calls. This implements net-centric systems optimally, and with precision. These same primitives can be used to implement trust policies that control execution paths.

**5.2 Data Distribution**

The following example is from the Joint Strike Fighter (JSF) Shared Synthetic Environment (SSE) Risk Reduction Activity (RRA). In this case, JSF (F-35) data must be sent to a variety of clients. One can envision a JEE collective as depicted below. Some numbers of different types of functional elements (FE) perform computation, and some variable speed communications fabrics provide communication between them. Systems and data sources are plugged in to the collective via an easy to program publish and subscribe API.

![Figure 6. JEE polymorphic computing architecture](image)

The native data structure `StateUpdate` is sent as a segment of binary data. The `CdParameterSet` has the ability to contain both structured and unstructured data in a single package using a series of `Set` functions.

On the receiving side in a system to be integrated with the JSF would insert the following code in order to operate with the JSF state data.

```c
void *Env = NULL;
...
Env =Sub1GetInteractionCharacters("UpdateOwnership", "UpdateOwnershipHandle", 1);
...
void *Env = 0;
...
```

The ability to simply send any collection of data in a lossless manner is now possible. The developer needs to know nothing of the communications method, security method or controls, encryption, or compression being used. The code is then more resilient and flexible at the developer level. This approach permits code in Java, Python, Ruby, or PHP.

**5.3 System Control**

Consider a more complex scenario than driving, such as large-scale battlespace threat monitoring and response, specifically that aspect regarding anti-access/area-denial (A2AD). There is always an order of battle, or sequence of initializing actions that have to take place before any complex offensive or defense operation, the goal is to have key sensors that can collect, ideally, sufficient statistics for relevant phenomena, infer meaning from those measurements, such as from a Baysian, or belief, network, and act on that information.
The term proposed for this new pattern of activity is collect-correlate-alert-act (C2A2). Various algorithms can now be inserted as part of the mechanism that actually triggers activity. In the case of battlespace monitoring consider the case where a connection is lost with an unmanned aerial vehicle (UAV) and a satellite. An alert is generated warning of a possible electromagnetic pulse attack, and appropriate actions are taken such as powering certain systems down and activating others, such as a mobile AOC.

In other words, based on multiple and certain sequences of events, both systemically and socially there exists a probability of event that would necessitate activating AOC (M) system, thereby placing a pilot (UAV or in-cockpit) in an advantageous rather than disadvantageous situation in a potentially denied area, as opposed to having to improvise independently due to the slow speed of human and organizational coordination. The goal is to not create another Situational Awareness System, but a cognitive/active anticipatory system that reduces and reverses the disadvantage being created by our adversaries.

![Figure 7. A2AD conceptual graph](image)

The code to implement this is straightforward. A conceptual graph entity is initialized with the CG in figure 7, and then periodically executed. The complexity is in the actor subgraphs, and these can be modified by reloading the top level CG, or the CG replaced completely. This allows the system to adapt in terms of behavior, and with dynamic binding we can even add new functional components.

6. SUMMARY

There are many advantages of this approach to complex, automated, intelligent system development. Key advantages include ease of use, with an intuitive implementation framework for developers; and reduction of rework – getting the job done right the first time. The JEE helps meet practical needs in system development. Frequently testing, security, and interoperability are considered too late; and capitalizing on software engineering research rarely occurs.

Furthermore, existing assets can be re-cast and given “a new life”; and existing hardware assets have a longer lifespan. As such, the next generation of solutions is enabled, which mitigates many of the major problems associated with application development and systems integration. Scalable, robust solutions are provided for complex, real-world problems.

7. REFERENCES


