Joint Synthetic Battlespace Integration Framework (JSB IF)

S. David Kwak, PhD
ESC-CXE, The MITRE Corporation
781-271-6431
dkwak@mitre.org

Lt. Col. Emily Andrew
ESC/CXE
781-377-6421
Emily.Andrew@hanscom.af.mil

Keywords:

Joint Synthetic Battlespace (JSB), Architecture, Integration Framework, Ontology, XML, Semantic Integration, Syntactic Integration, Plug-and-play, Legacy System Integration

ABSTRACT: *The Joint Synthetic Battlespace or JSB will support training, acquisition, test and evaluation, and research and development communities. To accomplish this, the JSB must operate at many levels of detail, including: engineering level; entity level; mission level; operational level; and strategic level. The JSB's broad scope makes it impractical to build all new simulation components. Instead, the JSB will rely on existing components as much as possible. Integrating legacy simulations is, therefore, one of the most critical issues for the JSB.*

The JSB Integration Framework (IF) is being designed to address this critical issue. It is hoped that JSB IF concepts will eventually reduce the effort required for large-scale simulation integration by an order of magnitude. If successful, the JSB IF would permit construction of complex test beds or experiments in one or two months rather than one or two years. This paper describes an approach to developing JSB's integration framework, which clearly separates integration syntax and semantics, with an emphasis on innovative ontological semantic integration.

1 Introduction

In the United States Air Force (USAF), there is an emerging need to support development, acquisition, and deployment of capabilities to support the functions of task forces. For example, the Global Strike Task Force (GSTF) has been defined to require existing and new systems to achieve a higher level of integrated capability to achieve rapid air strike capabilities under a variety of possible circumstances. Because existing systems have been typically developed in a standalone fashion, their designs have become obstacles to achieving the full vision of the GSTF. To prevent this from occurring in the future, systems will have to be explicitly conceptualized using computer simulation. The Joint Synthetic Battlespace (JSB) will become one of the enabling technologies that supports development of new systems and migration of existing systems into the integrated GSTF vision, as well as that of other task forces.

The JSB's mission includes integration of both legacy and newly developed simulations. Current simulations have been developed to meet specific needs and, therefore, do not usually interoperate well with other simulations. Because of this phenomenon, the simulation world, in general, lacks the ability to support the highly integrated systems envisioned by task force concepts.

It is for these reasons that the JSB requires a common simulation architecture and core services, designed to ease integration of current and future simulations for a variety of defense modeling and simulation users. This paper briefly describes the JSB vision and the development approach being used to realize this vision, most notably, the JSB Integration Framework (IF).

2 Joint Synthetic Battlespace Vision

The Joint Synthetic Battlespace will be an interactive, simulated environment and battlespace, which will allow its users to simulate a variety of defense systems,

at varying levels of detail, using common simulation components. This JSB concept of operations is described, in detail, in the JSB CONOPS [1], which was approved by the AFROC in 2001, and will be used as a foundation for the JSB Program, which is being formally instantiated in FY04. This paper presents the four key areas of the JSB vision and strategy: 1) JSB is addressing a new problem; 2) many of the required component simulation capabilities already exist; 3) existing simulations are not interoperable; and 4) simulations need a common environment.

JSB is addressing a new problem

Traditional military systems have been designed for a narrow application domain, however, future weapon systems are different. These new systems are being design as more complex systems of systems. This approach results in a new level of complexity because of the many possible behavioral interactions within and among the systems [2]. As system size and complexity increase, the cost to engineer, test and evaluate these systems also increases. Therefore, the cost to build future systems will be much higher than is presently the case. A sophisticated simulation-based process then becomes critical, to construct and test these systems more inexpensively using simulation software, before building the systems. This is the new problem addressed in the JSB vision.

Many of the required component simulation capabilitiess already exist

JSB simulations will not usually be built from scratch, but will be largely composed of existing simulations. The United States Air Force Electronic Systems Center's preliminary analysis shows that many of the simulation models needed to meet JSB requirements already exist. While further analysis and use will determine whether these models are really adequate to support all JSB users, this is an encouraging preliminary result. Simulations for JSB users will be obtained from a variety of Government organizations and industry.

Existing simulations are not interoperable

Although many relevant simulations already exist, they do not generally interoperate without significant effort because they were created to serve their original users' specific needs. Incompatible requirements, architectures, technologies and standards are some of the main obstacles. While integration approaches do exist, their utility is usually limited to one-time use. A new and systematic approach to integrating existing simulations to form more complex composite simulation environments is needed.

Simulations need a common environment

True interoperation is more than just having a common interface specification. A common interface provides details of the data exchange protocol, which typically addresses syntactic differences among simulations. Each simulation still has meets different requirements and, therefore, has its own algorithms, limitations, and resolutions. These semantic differences are another challenge when it comes to true interoperability. An environment that helps resolve these semantic differences is needed. Just as human beings need common semantics to communicate in the same language, JSB must provide a common environment to foster meaningful simulation interoperability. JSB's prototype of a multi-spectral sensor environment is an example of the need for providing common semantics. This prototype effort provides a common synthetic environment to provide correlated views for sensors across a variety of spectra and sensor types: electro-optical; infrared; ground moving target indicator and synthetic aperture radar. A new effort proposes the creation of an integration framework to provide a common contextual understanding, which is described in this paper.

3 JSB Overarching Framework

The fact that JSB must support both training and acquisition communities makes flexibility of structure and composition a key requirement. This flexibility, when combined with ongoing changes in training and acquisition requirements, argues against using a single monolithic system to meet all the JSB requirements. Therefore, the JSB must include an architectural framework, a set of simulation components and standards for combining these components to produce valid simulations. Each resulting JSB "instance" will meet its specific user's needs, for military training, acquisition support, or concept development. The full extent of JSB requirements will be satisfied by the JSB instances, collectively.

Rapid generation of JSB instances is envisioned to be achieved by the JSB Overarching Framework, which is notionally shown in Figure 1. As shown in the figure, a JSB instance will be a product of the JSB system engineering process, which is tightly coupled with the JSB architecture. The JSB architecture will allow for early exploration, investigation, and design of internal structures, and control of flow of the target JSB instance without actual internal simulation components. Inputs to these JSB processes include legacy and new simulation systems, users' requirements, and existing and new technologies. The proposed JSB Integration Framework (IF) is envisioned to provide a means to put together the JSB internal components following the overall structure and the control flow captured in the JSB executable architecture. Finally, a JSB IDE (Integrated Digital Environment) will be designed to facilitate retrieving, storing and moving data and information throughout the JSB. That is, the JSB IDE will become a library and associated tool set for JSB system configuration management, documentation control, program management, collaboration, and distributed manufacturing of JSB instances.

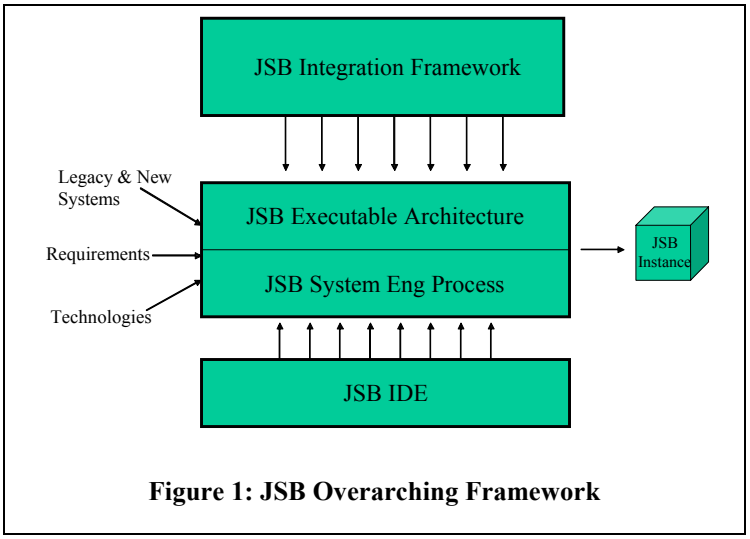

Figure 1: JSB Overarching Framework

Figure 2 shows how a JSB IDE could support creation of a JSB instance. The set of documents needed to create a JSB instance would be managed by the IDE. For example, a user requirements document, technical specification document, and program management document could all be managed in the central knowledge repository. The IDE's multiple view generation capability will support semi-automated creation of multiple documents from a single central knowledge store in the IDE. Because these documents are different views of a JSB instance, they may each be generated from the formal specification of that JSB instance.

Presently, human authors manually create these documents. Collecting necessary knowledge by parsing the existing document, adding additional knowledge, and assembling it in a specific format, a target document is created. For example, JSB CONOPS (Concept of Operations) is one of the JSB documents, and is manually created by a group of people. The JSB IRD (Interim Requirements Document) is subsequently created by another group of people. JSB CONOPS, the baseline document is manually parsed to form a knowledge base, and new knowledge specific to JSB IRD is added to the knowledge. If the first group of the people, who created the JSB CONOPS, are not

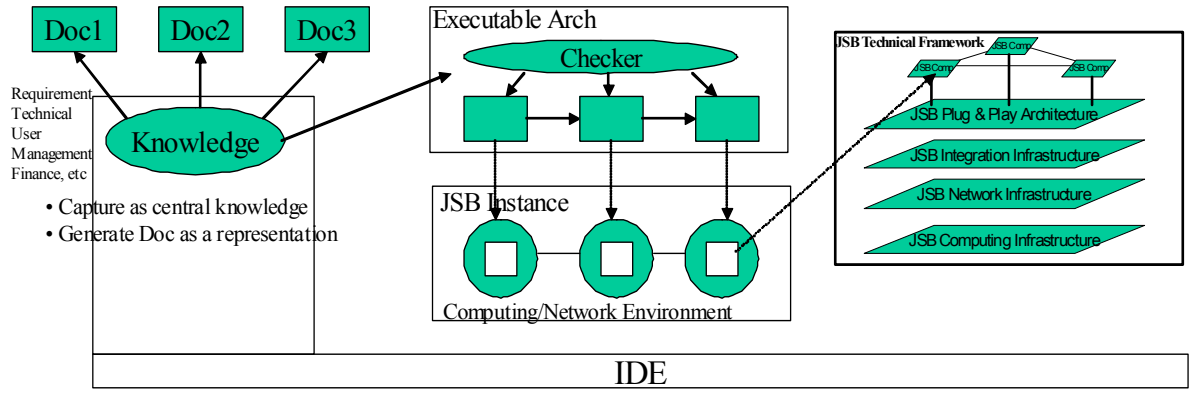

Figure 2: JSB IDE allows for shortening the JSB overall process from Requirement Documentation to creation of a JSB instance.

involved in the JSB IRD generation process, a large portion of the valuable knowledge collected and created during JSB CONOPS creation process is lost. The knowledge captured in a form of a JSB CONOPS document is really limited compared to the knowledge actually captured by the first documentation group. In reality, most of the knowledge is simply stored in the brains of the humans participating in the documentation process. After the CONOPS generation process, the knowledge captured in the human brains has little chance to be carried over to the subsequent JSB IRD generation process unless the same people are involved. Although the original group has been involved, the temporal gap between the two document generations undermines the effectiveness of communication. The current form of the CONOPS is, after all, one of the best-known means to capture the knowledge; however, it is certainly a narrow channel for transferring knowledge. Again, there exists inefficiency of restoring the knowledge captured in the CONOPS by reading the CONOPS document, reassembling the knowledge base in the human brains, and creating the new JSB IRD documentation.

During the above parsing and reassembly process, new knowledge is added. As new documentation is created following the process of creation of a JSB instance, the size of accumulated knowledge is increased as well as the size of the documents. Thus, the inefficiency of unpacking and packing knowledge from one document to the next is also increased. This process would continue until a complete set of documentation becomes available to build a JSB instance.

This conventional knowledge accumulation process is manually intensive. Even so, the current process focuses on manual creation of written documentations partly because it is a familiar form of knowledge capturing since the invention of writing and partly because there is no other alternative. It is also true that this manual process has been focused on standardization on manifestation of knowledge on papers, rather than the standardizing of storing the source knowledge. Thus, reusing the captured knowledge at the source level could not have been achieved at all. In reality, many documents have been historically created to simply satisfy a documentation requirement of a given process rather than to facilitate knowledge transfer. There is little chance to expect a significant change in the near future to having a truly re-usable knowledge representation among the document generation groups[1].

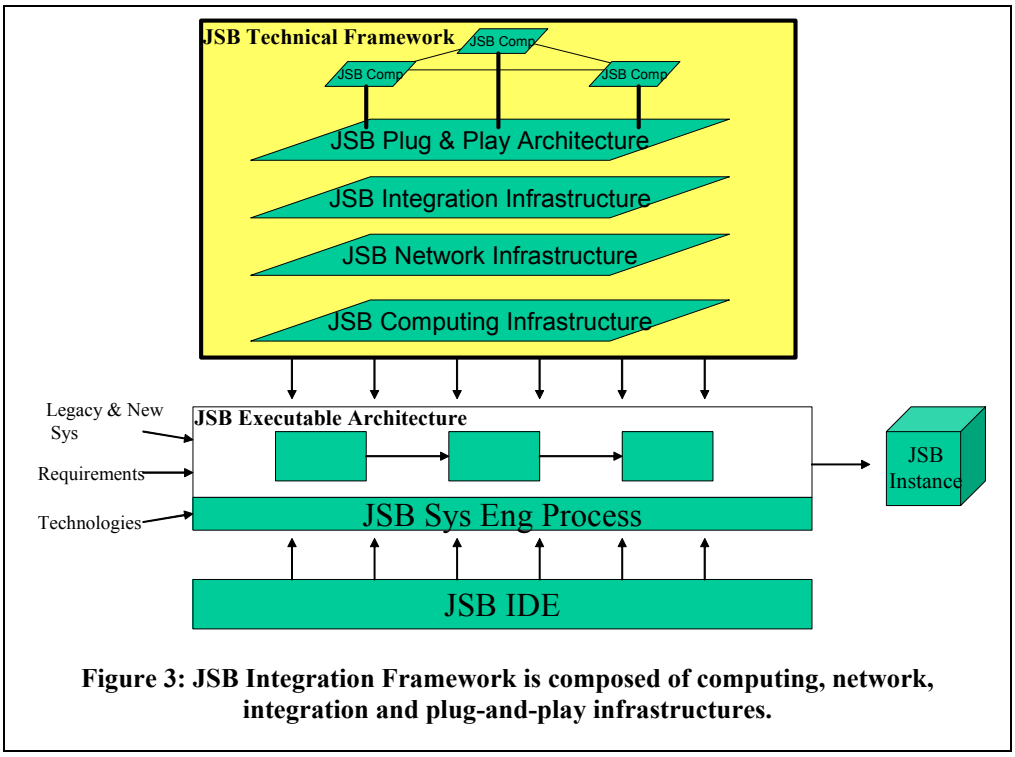

Figure 3: JSB Integration Framework is composed of computing, network, integration and plug-and-play infrastructures.

There is emerging technology in this area, and the JSB IDE is an example. It supports standardization of knowledge representation, and provides tools and utilities for capturing, manipulating and extracting

[1] Although a single group of people may generate a series of documents, there is little chance of systematic generation and maintenance of a single knowledge base.

various views and formats. Once knowledge is electronically captured in the tool rich environment, it becomes much more powerful than any textual form of knowledge, which is static and not-easily transformable. Unpacking and packing knowledge stored in conventional documents are undeniably time consuming, but the JSB IDE will allow for a direct management of knowledge, and automatically maintain the evolution of stored knowledge.

When the above IDE knowledge approach is fully adopted, JSB instance creation will be greatly facilitated. JSB system engineers will directly interact with the knowledge in a form that they need rather than passive documents written on paper. Moreover, the JSB IDE will support a backward compatibility so that the knowledge in the central JSB IDE repository will support semi-automatic creation of paper documentations mandated by DOD.

The captured knowledge in the JSB IDE does not just facilitate creation of mandated acquisition documentations, but also supports other functions that required supporting instantiations of JSB instances. The JSB IDE's multiple views of the central knowledge repository permit generation of other documents and information such as engineering, managerial, and financial aspects of JSB. Again, all of the knowledge is captured in the central repository, and evolves together. Thus, although vastly different documents and artifacts are generated from the central repository, the JSB IDE will automatically maintain consistency across all of the documents that are generated from the IDE. Traditionally, maintaining consistencies across totally disjoint groups of people such as financing, management, engineering, etc has been a difficult and time-consuming task. The JSB IDE will vastly improve this problem.

4 JSB Integration Framework

The JSB Integration Framework will be designed to permit JSB to have true plug-and-play capability. The internal structure of the JSB Integration Framework is shown in Figure 3.

The ultimate goal of JSB is to rapidly instantiate JSB instances by integrating existing, i.e., legacy, simulation systems and models to produce a JSB instance. When the required simulation does not already exist, a new simulation component may be developed. Therefore, creating a new JSB instance becomes a composition task rather than a development task. This concept has is similar to constructing models from LEGO [2] pieces rather than from raw materials [4].

Historically, many approaches, architectures, and protocols have been introduced to achieve LEGO-like plug-and-play capability for simulation systems, but they typically fall short. Often a proposed plug-and-play scheme is narrow in scope and does not cover the required simulation domain, or it operates at too high a level and requires custom development to fill in the details. The JSB Integration Framework will attempt to address these issues; it will divide the plug-and-play problem into four levels: 1) computational infrastructure; 2) networking infrastructure; 3) integration infrastructure; and 4) plug-and-play architecture. These levels are illustrated in Figure 3. The bottom two layers, network infrastructure and computing infrastructure, use today's commercial technologies. In contrast, the top two layers, plug-and-play architecture and integration infrastructure are more unique to JSB.

JSB's Computing Infrastructure layer is envisioned to use commercial technology to take advantage of the exponential increase in capability that occurs each year in this market. This increase is quantified by Moore's Law, which states that storage density, in terms of transistors per square inch, doubles every 18 months [5]. In fact, Moore's law has held true for several decades. By using commercial computing technology in a layered architecture, the JSB will benefit from annual improvement in computing technology.

The JSB Network Infrastructure is also based on commercial technology to take advantage of the remarkable progress in networking technology. In some areas, the growth rate of network infrastructure technology has been surpassing the rate predicted by Moore's law. Today's World-Wide Web concept and industry were established over little more than the last 10 years and have flourished over an even shorter, more recent period. Therefore, synchronizing the JSB computing and network infrastructures with the commercially available technologies is a wise tactic for the JSB Integration Framework.

The top two layers of the JSB Integration Framework, on the other hand, are more JSB-specific. Commercial technology has less to offer in these areas, although significant advances are being made in the areas of IT Centric Enterprise Application Integration (EAI) and Enterprise Information Integration (EII). The JSB will use these emerging technologies, as appropriate, within the JSB Infrastructure layer. The JSB Integration Infrastructure (JSB I2) will be uniquely designed to allow for integration of legacy simulation systems

[2] LEGO is a registered trademark of The LEGO Group

regardless of their protocols and standards, e.g., ALSP, DIS and HLA/RTI. JSB I2 will also facilitate integration of new JSB-compliant simulation models, components and systems. The JSB I2 will help transform simulation components, either legacy or new, into a JSB-compliant plug-and-play components.

In contrast, the JSB Plug-and-Play layer will rely on development and integration of JSB-specific capabilities. The JSB Plug-and-play architecture will provide a true plug-and-play capability of JSB compliant components. Therefore, the top two layers are unique to the JSB Integration framework, and together with the bottom two layers, a truly plug-and-play architecture is constructed. The details of the top two layers will be discussed in later sections.

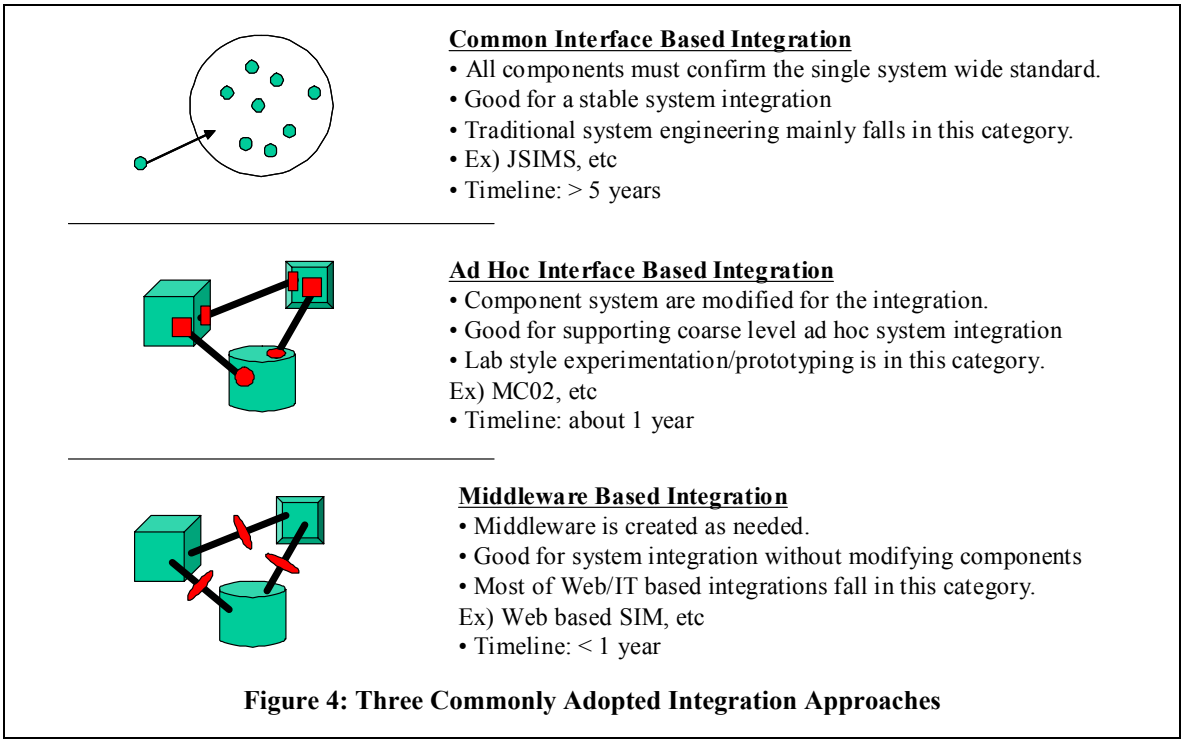

Figure 4: Three Commonly Adopted Integration Approaches

5 Commonly Used Integration Approaches

Before discussing the details of the JSB Integration Framework (IF), three commonly used integration approaches are presented to provide a common basis of understanding for discussion of the JSB IF. These three approaches to integration are illustrated in Figure 4.

The first approach is Common Interface Based Integration. Traditional system engineering falls under this approach in which the interfacing components are precisely defined and controlled by a strict engineering process and specification. In this approach the entire interface definition of all participating components is pre-defined. It is usually captured in an interface control document. Creating a complete set of static interface definitions is a rather natural process because the functions of the internal components are statically definable except for small changes in later parts of the system's life. Thus, the internal components can be manufactured under precise engineering control using the pre-defined system and interface specifications. In the simulation world, JSIMS (Joint Simulation System) falls in this category. That is, all of the JSIMS components have to conform to the JSIMS interface specifications before integration into JSIMS. Figure 4 illustrates this by making all components circles, while the component being integrated has the same circular shape. Due to the strict traditional engineering process, i.e., all components are made to confirm the given standards, the time to construct such a system tends to be long. A typical time line is several years or more.

Another commonly used approach is Ad Hoc Interface Based Integration. This approach is often adopted by a lab style experimentation or prototyping. Unlike the first approach, interfaces of the components have not been manufactured under a single predefined interface specification. Therefore, most of the participating components do not interoperate without being changed. Case-by-case interface modifications are needed. This integration approach is usually adopted to create a temporary experimental system. The MC02 (Millennium Challenge 02) modeling and simulation

system is a good example. The time span for such integration is about one year or less.

Recently, Middleware Based Integration has been gaining in popularity. The advantage of this approach is not requiring modification to the participating components. This avoids the pitfalls associated with modifying working systems, which often introduces errors into previously working systems. Additionally, a proper modification requires an in-depth knowledge of the component, which often requires that the original developer be involved.

The pros of the middleware approach have discussed in the previous section. Thus, the cons are presented here. First, the middleware interfaces are usually constructed manually, and they are created case-by-case due to the point-to-point nature of connecting two adjacent components. Therefore, the total number of middleware creations quickly rises, in a geometric fashion, as the number of components increases. The theoretical upper bound of the required middleware constructions is on the order of N^2, where N is the number of components to be integrated.

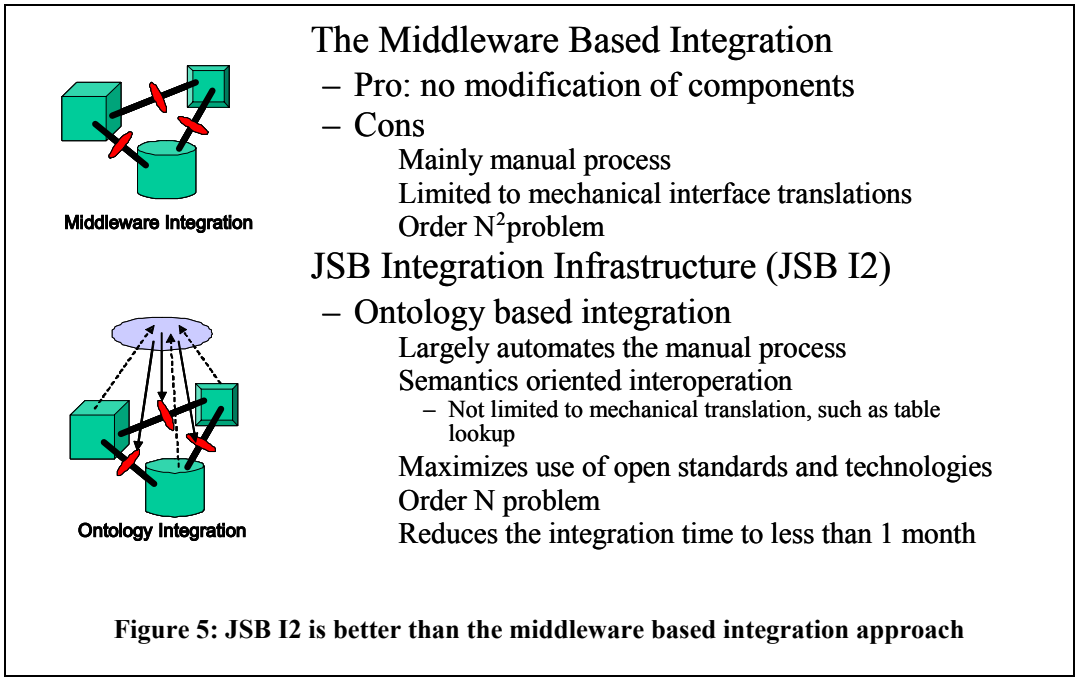

Figure 5: JSB I2 is better than the middleware based integration approach

Middleware-based integration eliminates the need to modify existing applications, as long as the required information is being exchanged through the existing application interfaces in one form or another. Instead, middleware is added, as needed, to make the required translation. In the Web/IT (Information Technology) world, this approach has been widely accepted and has achieved great successes. There has been a recent movement to duplicate this success in the domain of simulation integration. A typical timeline for this approach is shorter than for the Ad Hoc Interface modification approach.

6 JSB Integration Infrastructure (JSB I2)

The JSB Integration Infrastructure is not one of the traditional integration approaches, although it shares many similarities to the middleware approach. To facilitate our further discussion, a comparison is made drawing in Figure 5.

Another common attribute of the middleware approach is that it is constructed as a set of value-to-value translators, connected from one system to another. This point-to-point translation is inherent to the middleware, which presumes a one-to-one mapping of values in one system to those of another. Contextual aspects of the data, which are usable and sometimes critical for subsequent processing, are rarely considered or implemented[3]. Therefore, the middleware essentially becomes a table lookup operation. If data in one system may represent more than one possible values in the other system, then this approach will not work. A middleware approach, which is a context-free one-to-one translator, cannot handle this complex situation.

[3] Anyway, the middleware approach does not facilitate such contextual transitivity relationship. Often, middleware implementation approaches make practically impossible implementation of a contextual transitivity relationship.

One-to-many translations are commonly performed in human language translations. Due to semantic structural differences of two languages, one representation (i.e., one meaning such as a word, a phrase or a sentence) in one language often has multiple representations in another language. Therefore, there is a low applicability of mechanical translations of human languages. Non-determinism should be resolved by a common context between two languages. It is common that machine translated text becomes a laughable object due to out-of-context usage of translated languages.

The JSB I2 approach will explicitly address the above disadvantages of the middleware approach. First, the JSB I2 will turn the N^2 implementation issue into an order N problem. Instead of manual implementation of each middleware case-by-case, a commonly ontology is implemented. The N systems are directly connected to this ontology. Thus, only the N number of connections is implemented. Figure 5 captures this concept. Then, JSB I2 automatically creates interfaces between two systems as needed. Second, the ontology maintains a common context. Thus, it is capable of resolving non-determinism of one-to-many translation cases. Moreover, it updates and maintains the common context during run-time so that it reflects the latest common context among the N participating systems. The advantage of JSB I2 is, consequently, its implementation economy in reducing the order N^2 problem to an order N implementation and its power of resolving non-determinism with the common context. It is expected that this ontology approach will significantly reduce integration of many (i.e., around 40 to 50) systems. Integration of 40 to 50 systems is a typical complexity targeted by JSB. JSB I2 is also believed to be cable of significant reduction of the current order of 1 year integration time. We are currently targeting for reducing down to one month.

The preceding discussion was about the semantic sub-layer of JSB I2, which is one of the two aspects of the JSB I2. The JSB I2 has a syntactic sub-layer, and it supports the semantic sub-layer by sending and receiving a data between systems without concern for the semantic details. This context free syntactic interoperation is implemented by this sub-layer, and this approach greatly simplifies implementation of JSB I2. A simple analogy of this syntactic integration layer is a LEGO piece's dimples. They allow for integration of LEGO pieces without worrying about the semantic baggage associated with leg pieces, such as whether they are being used to construct a castle, truck, human soldier, etc.

On top of the above syntactic interoperation, a semantic interoperation is implemented, which is the ontological portion. The current choice for the semantic representation is XML (eXtensible Markup Language). XML is also a logical choice. This choice matches the current trend of DOD and of commercial industry, which encourages using of XML as system interface data representations.

XML is not a just one standard of representation of data, but it comes with a family of utilities such as XSLT (eXtensible Stylesheet Language Translation), which allows for point to point XML translations. XML and its family of utilities really facilitate building machine understandable knowledge representations, and a direct knowledge transfer between machines. Thus, XML provides a foundation for M2M (Machine to machine). Finally, XML is also human readable.

Adopting XML as a data representation standard effectively creates multiple islands of "XML-ized" data language groups. XSLT easily translates one XML language to another. However, its capability is limited to a point-to-point translation. Therefore, if we rely on the standard XSLT, we essentially recreate the middleware approach discussed before. Instead, JSB I2 uses ontology to represent the common context and to translate one set of XML data to another. Although there has been an issue related to non-standard representation of the ontology itself, luckily the XML industry has started to develop standard ontology representations such as OWL (Web Ontology Language) [6], RDF (Resource Description Framework), etc. Therefore, the reuse of ontology will be greatly facilitated, and an incremental accumulation of ontology becomes possible.

Adopting the commercial standard is crucial. The commercial sector continuously improves technologies with their own investments, and JSB IF, which heavily leverages the commercial technologies, will be a beneficiary. JSB IF will benefit by this recent advance in technology as well as other technologies such as XML. The current approach is to prototype the JSB IF while leveraging the above mentioned technologies.

7 Summary and Conclusion

The JSB is a concept that facilitates design, analysis, development, acquisition, training, and simulated operation of future mission-oriented C2/weapon systems, which are essentially a complex system of systems such as C2 constellation. Realizing the JSB's concept requires leveraging all existing assets (i.e., simulation systems, communication infrastructures, organizations, etc) rather than building from scratch. Thus, the JSB needs to have proper and adequate abilities and processes that allow for quick integration of existing assets. Many of these efforts are directly

associated with human organization and programmatic aspects, but new technical break-through will also be used as they become available. This new JSB concept will benefit from existing technologies as well as from the continual improvement of technology over the years. As the JSB core technologies, architectures and processes mature, the JSB will start to reach the full scope of the vision while significantly impacting everyday operations of USAF and other DOD services. Some of the technologies described in this paper are essential as a necessary step toward the "real" JSB. Our current efforts will surely bear fruit while providing much needed valuable data moving forward to the "real" JSB in the future.

8 References

1. USAF, *Joint Synthetic Battlespace For Simulation Based Acquisition*, JSB Concept of Operation (CONOPS), 2001.
2. Kwak, S.D., "A Multiple Paradigm Behavior Architecture: COREBA (Cognition Oriented Emergent Behavior Architecture)", *Proceedings of 1998 Fall Simulation Interoperability Workshop*, SISO, Orlando, FL, Sept. 14-18, 1998.
3. Kwak, S.D., *JSB Overall Framework*, Power Point Presentation, USAF ESC/CXE JSB, May 2002.
4. Kwak, S.D., Andrew, E.B., "Technical Challenges for Joint Synthetic Battlespace (JSB)", *Proceedings of 2002 Fall Simulation Interoperability Workshop*, SISO, Orlando, FL, Sept. 9-13, 2002
5. Moore, G.E., "Cramming more components onto integrated circuits", *Electronics*, Volume 38, Number 8, April 19, 1965.
6. http://www.w3.org/TR/2003/WD-owl-features-20030331/, W3C, March, 2003