ACHIEVING INTEROPERABILITY OF COMMAND AND CONTROL SYSTEMS USING TRANSLATION GATEWAYS[1]

David PERME, Mark WHELAN and William P. LOFTUS

Issue of Interoperability

Over the last several decades, the military has greatly benefited from the increased knowledge and capabilities provided by using computerized command and control systems. As this use has expanded exponentially, so has the need to integrate these systems. The breadth of computing technology at the component, functional, and mission level has further complicated the issue of interoperability. By their nature, these disparate systems have varying levels of fidelity, granularity, quality and availability. The cost of establishing collaboration between these systems is typically high, and is complicated by differing organizational readiness levels, willingness, and technical ability to affect collaboration. The opportunity to enable interoperability, therefore, has great value, provided it can address these factors and more.

The need for translation of information and data to forms that are readable and interpretable has continuously challenged users of computer systems. Over time, the technologies employed to accomplish interoperability have evolved. Initially, and still prevalent today, one-to-one interfaces explicitly define how two systems interact. This type of approach works but does not scale. Other approaches, such as shared databases, common data repositories, and defined common standard messaging and interface formats, present solutions to some interoperability issues but are not panaceas. Each approach is appropriate in given circumstances. Attempts to provide a single solution for all scenarios typically fall short due to technical challenges, adoption resistance, and funding availability.

The following sections present an approach that we have used successfully to simplify system communication and interoperability by a system-neutral layered gateway. The approach addresses the realities and complexities of the systems' environment and leverages domain expertise and emerging technologies including web technologies and expert systems.

Layered Translation Gateways

The best response to the issue of system interoperability is one that recognizes that successful systems are those that add value, minimize impact to existing systems, and can evolve over time. Our experience in integrating diverse systems (such as C4I systems) is that layered gateway architectures enable successful interoperability between systems, while isolating the impact of changes to any system. Value is derived by delivering to a system only that information that is used by that system. Moreover, layered translation gateways deliver and receive information to or from a system, in the format and medium native to that system. Additionally, a well designed layered architecture provides the opportunity to insulate layers from the impact of change (new systems, modification to existing systems, system retirement), thereby reducing the overall impact facilitating evolution of the gateway when change occurs.

At the core, system communication is translation. Translation is the conversion of one data format or protocol to another while retaining the meaning and context of the original. The key factors in translation include the data itself, the format of the data, the medium of transmission, and the context of the data that turns it into useful information. A gateway must be able to deal with all of these factors. The data, format, and medium translation challenges are relatively straightforward, discrete, and solvable transformations. The context translation challenge is more complex and involves the application of subject matter knowledge and expertise.

There are for basic components to flexible and adaptable gateway architectures:

1. System-neutral data interchange format
2. External systems interface layers
3. Translation layer
4. Intelligence layer

The first step is to define a *system-neutral, data interchange format*. By developing a common data model and schema, a gateway repository or data warehouse is created to facilitate the integration of disparate systems. Today, this data model is usually described by class diagrams and an XML schema. With a defined data model, appropriate programming interfaces to the data translation layer are easily described and developed.

The focus of the *external systems interface layer* is the integration with various lower level architectures and protocols using reader-writers and adaptors. This approach provides the greatest flexibility for many-to-many system integrations as depicted in

Figure 1. Within a system of systems, there exist three types of relationships: one-to-one, one-to-many, and many-to-many. A one-to-one relationship is often referred to as a point-to-point interface. Two systems talk to each other directly through a defined method or protocol. There is only one interface and changes to a system will result in changes to, at most, one interface. A one-to-many relationship describes multiple interfaces from a single system to a number (N) of other external systems. A change to the single system has the potential to affect N interfaces. A many-to-many relationship describes interfaces among and between a number of systems (N). Every system has an interface with every other system. While this provides the maximum potential for information exchange, it also creates [N*(N-1)]/2 interfaces. These relationships, however, can be normalized using the system-neutral data interchange format by reducing a many-to-many relationship to a one-to-many relationship. Each system needs to only read and write a single canonical format.

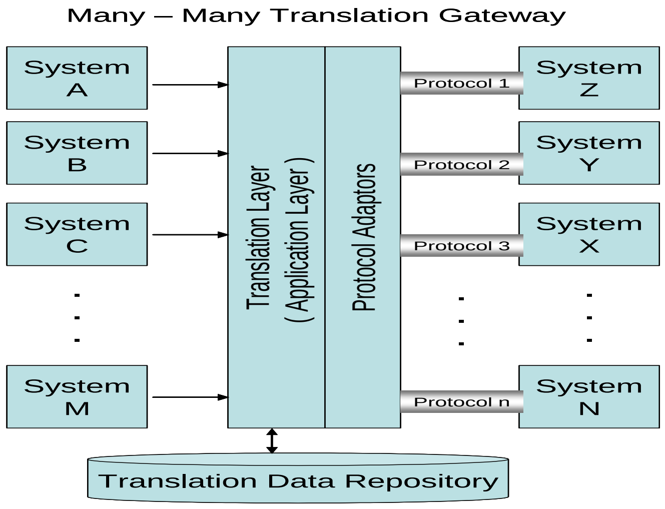

Figure 1: Translation Gateway

A *translation layer* transforms the data from the exporting systems into the common representation. The protocol adaptor extracts information from the repository, formats it, and communicates the information in the correct protocol to the receiving systems. Gestalt's experience in integrating real world C4I systems with simulation models has shown that a common representation becomes a forcing function for the information and is necessary to be able to translate the intent of the communication. The focus of a translation layer is to manage both data and business rules. Data is the information that is exchanged between systems. Business rules apply the logic of translation and transformation.

The data becomes the focus point for the incorporation of intelligent agent technology, via an *intelligence layer*. This layer reduces man-in-the-loop dependencies, enables smart system-wide decision making, and maintains meaningful communication. Often the business rules for the transformation and communication can be encapsulated into the intelligence layer. This approach allows the entire gateway translation to be applied to an enterprise, where the intelligent agents manage the variance between organizations.

High-level Development Approach

To develop a system-independent translation protocol, the data models and business rules of the target systems must be examined. In addition, current and future requirement sets for these systems should be understood. From this examination, a common system and data-neutral data schema can be designed and developed. The first step is to select an appropriate set of candidate systems that will form the basis for the development of a common system-neutral data schema. Typically, these systems are large in scale and produce complex message sets. There is a need to examine these message sets to identify key data elements and business rules associated with the data contained in the messages themselves. From this examination, a data population set from each of the systems can be derived. In this step, data discovery and cataloging is performed. The cataloging can be as simple as capturing the data population set in a spreadsheet. The key is to represent a system's data in a template form that retains information regarding its native schema while allowing the commonality of a template to begin the process of relating data elements from separate systems to each other.

Once all the data has been categorized from the candidate systems, the initial step of an object-oriented analysis can begin, that is, to build an initial Unified Modeling Language (UML) class diagram of the intended data-neutral data schema. The purpose of establishing a class diagram is to represent the class structures and hierarchies as well as the relationships between the class structures. The task of normalization is to then analyze the UML class diagram and promote like attributes into superclasses, compress the depth of the class structure wherever possible and test it against scenarios developed in the data discovery and cataloging phase.

Once the data to be communicated is well-understood the functions and duties of each layer in the architecture must be defined. The interfaces between the layers must also be defined and documented. Armed with the class diagram, the interface between the native system data format and the common data model can be defined and designed. In our experience, the best approach for reuse and interoperability is to use a loosely-coupled Application Programming Interface (API) approach as the interface into each layer. An API for the architectural layer performs the data transformation into the common data model. This API should allow for interfaces with all of the exporting systems that are selected and should be designed to allow for extensibility to future systems integration. XML has quickly gained traction in the commercial field as the document interchange description format for interoperability between systems. In the digital world today, there exist multiple XML vocabularies as well as software that perform the translation from one XML vocabulary to another. XML also brings along many tools, which enable XML documents to be processed with a minimum of programming effort. For these reasons, an XML schema should be derived from the class structure defined in the above steps.

The last step of a successful translation gateway is incorporation of business rules. This refers to the data fusion or disaggregating of incoming or outgoing data explicit for each integrated system. These rules must exist in a dedicated architectural component of a translator. This component is normally defined as an interface API between the normalized and the system-specific data representations. The success of a translator is dependent on the ease of configuring the business rules. Rule-based expert systems can be used to accomplish a flexible implementation of business rules. More specifically, commercially available open system standard software tools can be used. The use of these tools eliminates specific translation idiosyncrasies from the overall translator, and instead allows the inference engine and corresponding rules to ensure that the correct business rules are applied. These translations can be tested and easily adjusted or tuned to produce the desired outcomes. By isolating the business rules in a separate architectural layer, the impact of tuning on the translation software baseline is minimized.

Example

In the late 1990s, Gestalt personnel began the integration of the Air Operations Center (AOC) command and control systems to a suite of simulations. The AOC to Simulation Interface (ASI) initially integrated the Air Force’s Air Warfare Simulation, AWSIM, to the main AOC command and control system. Since that initial integration several command and control systems have been integrated including the Theater Battle Management Core System (TBMCS). ASI employs a layered architecture. Three primary layers constitute the ASI architecture. Two service layers, the Simulation Services Layer and the C4I Services Layer, handle the data capture and dissemination processes. These processes employ publish and subscribe mechanisms, and are compatible with both the Aggregate Level Simulation Protocol (ALSP) and the High Level Architecture (HLA) Runtime Infrastructure (RTI), and produce USMTF, TADIL, and XML messages. The third ASI architectural layer is the Translation Service Layer that provides the data translation services between the integrated systems using Gestalt’s proprietary Command and

Control Data Interchange Format (C2DIF), a common, system neutral data representation that acts as the data and knowledge repository. The ASI system has been used at every major exercise (well over 70) since 1997 and has consistently demonstrated its versatility and viability. ASI's layered architectural approach is extensible, providing the capability for easily establishing interoperability between and among other existing legacy, joint, and coalition systems, as well as future systems. The ASI system has been extended beyond its original objectives via the integration with numerous simulation models including the National Air and Space Warfare Model (NASM), and the Navy's Research, Evaluation and System Analysis Simulation (RESA) and Joint Semi-Automated Forces (JSAF) models.

In 2002, the Gestalt team, sponsored by the Air Force Research Lab (AFRL), initiated the deployment of the Intelligent Mission Controller Node (IMCN) system. The IMCN system employs expert system technology using an intelligent agent framework. Concurrent with this development effort, the C2DIF data representation was expanded and matured to incorporate a more robust representation of the information elements associated with air missions and air warfare. The IMCN system is implemented to reason over any of the C2DIF data elements, providing the ability to develop intelligent agents that act across multiple air mission tasks. Using IMCN allows data to be represented devoid of any consideration for the business rules of any particular translation, knowing that the expert system can handle the business rules. A system prototype was first successfully used at Blue Flag '00-4 and at all major exercises since, including UFL '01.

The key success factor in the use of intelligent agents is to define them such that the complexity of the rule base does not outweigh the value gained by their use. Seeking a complex rule base to address all issues would have resulted in failure. However, Gestalt segmented intelligent agent scope and functionality, establishing reasonable rule sets that could be easily implemented and extremely impactful.

One example of an intelligent agent we have employed in this fashion is in air mission route planning which automates the ingress and egress of air missions, taking into account ground-based threats. Another example is in weaponeering, i.e., the automation of the process of pairing squadrons and airframes with weapons and targets.

Overall, ASI and IMCN have allowed the Air Force to integrate several command and control systems with a suite of simulators. The business results have been significant. ASI is a system that has reduced manning budgets by a factor of four, produced higher-quality execution, and lower future integration costs.

Summary

Translation gateways are a viable method for increasing interoperability between systems and decreasing the complexity of the integration. The development and use of system-neutral data schemas, coupled with translation services, enables the exponential power of many-to-many collaborative relationships for the linear cost and complexity of a one-to-many integration. This is achieved through an approach that incorporates sound design principles (rigorous analysis and object-oriented techniques), commercial best practices (Application Programming Interfaces and XML), and advanced technologies (intelligent agents).

Notes:

1 This article is based Technical Report 2002-CC-01 -TG of Gestalt LLC. The company provides products and services to governments and Fortune 500 companies that address their collaboration and interoperation needs related to network-centric decision support systems including command and control, modeling and simulation, and enterprise business systems. Gestalt is an information technology firm that helps decision-makers increase their return on investment in existing systems through the application of state-of-the-art interoperation technologies.

DAVID PERME has twelve years experience in the executive management and operations of advanced software solution providers, beyond his ten years of experience in the Air Force. He holds a BS degree in Aerospace Engineering, Kent State University, and MS in Computer Science, Boston University. Mr. Perme has directed, supported, and evaluated hundreds of US Air Force, NATO, and Joint exercises and experiments world-wide. He was a principal lead and developer on one of the most successful interoperability programs in use today, the Aggregate Level Simulation Protocol (ALSP). His work and influence enabled the US Army's standard aggregate level simulator, the Corps Battle Simulation (CBS), to interoperate with the US Air Force's standard aggregate level simulation, the Air Warfare Simulation (AWSIM). The program, originally designed to be a prototype only, was so successful in execution that it has yet to be supplanted today. Mr. Perme was the designer and principal developer of the most successful C4I-to-simulation development effort to date. The program, initially begun as a Defense Modeling and Simulation Office effort termed *Project Real Warrior* (PRW), has evolved into the AOC Simulation Interface (ASI). Mr. Perme restructured and selectively re-engineered components of the C4I interface prototype, at the same time that the system was being used for major joint exercises. Currently, Mr. Perme is a Managing Director and Co-Founder of Gestalt. LLC. Contact address: 11 Federal Street, The Ops Building. Camden. NJ 08103, USA. Fax: 276-200-0541. *E-mail:* dperme@gestalt-llc.com.

MARK WHELAN has twenty years experience in information technology, systems integration, and web services. He holds BS in Computer Science, Penn State University, and MBA, St. Joseph's University, Philadelphia. His background includes various management positions in engineering and technology arenas serving the government and commercial sectors. Mr. Whelan combines strong experience in modeling and simulation, operational excellence through technology innovation, and strategic consulting. Prior to Gestalt, Mr. Whelan was Vice President, Operations, for Breakaway Solutions. Mr. Whelan led all professional services activities in the Mid-Atlantic region covering systems development and integration of web services for e-business, e-commerce, content management, and customer relationship management solutions. Currently, Mr. Whelan is the Managing Director of Gestalt. *E-mail:* mwhelan@gestalt-llc.com.

WILLIAM P. LOFTUS has seventeen years experience in the executive management and operations of advanced software solutions providers. He holds BS and MS in Computer Science, Villanova University, Villanova, PA. Currently he is the President. CEO, and Co-Founder of Gestalt, LLC. Previously, Mr. Loftus has served as the CEO of Breakaway Solutions, Chief Development Officer of the same company, founder and CEO of WPL Laboratories, manager of R&D at Unisys. Mr. Loftus has consulted numerous Fortune 500 and emerging companies as well as a number of investment bankers and venture capitalists. Mr. Loftus has co-authored numerous papers, IEEE standard P1430, and a best selling textbook. *Java Software Solutions,* currently used in over 400 universities world-wide and translated into Korean and Italian. He has also contributed to research in compiler theory, real-time software, software architectures, and interoperability. Mr. Loftus has received many awards including a Special Achievement Award from DARPA. recognition by the City of Philadelphia as one of the 40 most accomplished individuals under 40 years old in 1999, and was named as a finalist for the E&Y Entrepreneur of the Year award in 1999.
El-mail: wloftus@gestalt-llc.com.