Enterprise Composition©

John Wunder
Lockheed Martin Systems Integration

Enterprise information system (EIS) architecture is a system of EISs composed to meet strategic enterprise goals. This composition requires the application of a different set of processes, design patterns, and metrics than those used for stand-alone system architectures. For most enterprise architects, creating EIS architectures can be complicated and fraught with pitfalls, detours, and dead ends. These problems generally are not related to technology but rather caused by misperceptions and culture clash. This article defines a new, agile, incremental approach to EIS architectures and enterprise composition, and shows how it supports the creation and evolution of large EIS architectures such as the Air Force's Global Combat Support System.

Enterprise architects pride themselves on their ability to make stakeholder requirements trade-offs, yet experience shows that there comes a point when the size and complexity of enterprise requirements, especially in nontechnical areas, necessitate extending traditional enterprise system framework approaches (e.g., Department of Defense architecture framework [1], Federal Enterprise Architecture Framework [2], The Open Group Architecture Documentation [3], and Zachman Framework [4]). This article identifies the areas where current enterprise architecture approaches are too rigid or brittle to deal with certain nontechnical and nonfunctional architecture issues associated with architecting (or re-architecting) any large-scale enterprise.

In particular, this article focuses on enterprises with funding, staffing, or political constraints that require new technology/services that replace or must be added to those found in an existing set of applications. This article introduces the term *enterprise composition* to describe a collection of agile processes, metrics, and design patterns that have demonstrated applicability in dealing with these issues.

Gap Analysis: Enterprise IT Lessons Learned

There is an ever-expanding body of knowledge dealing with enterprise architecture frameworks [1, 2, 3, 4] as well as architecture description [5, 6]. Experience has shown that current approaches to enterprise architecture dealing with large-scale enterprises can do the following:

- Lead to unnecessarily rigid designs.
- Require wholesale technology upgrades (i.e., a *big bang*).
- Focus on information technology (IT) cost savings versus process cost savings.
- Result in local optimizations of systems, leading to suboptimal overall enterprise system performance.
- Become bogged down in stakeholder political and cultural *considerations*.
- Rely on traditional metrics such as source lines of code to determine progress.

Table 1 summarizes how enterprise composition addresses some of the shortcomings associated with current approaches to enterprise architecture with respect to large-scale enterprise IT (EIT) systems. The sections that follow will elaborate on lessons learned.

Enterprise Composition Processes

The following sections describe enterprise composition extensions to (1) EIT decision making, (2) EIT framework boundary definition, (3) EIT product selection, and (4) strategic enterprise metric definition.

EIT Decision-Making Process

Martin Fowler [5] recognized that most architecture definitions consist of two elements: (1) breaking the system into parts, and (2) decisions that are hard to change. While it is often the case that enterprise architects consider their architectural decisions to be carved in stone for posterity, when dealing with large enterprise systems, the advice of Gen. George Patton may be more applicable: "A good plan violently executed today is better than a perfect plan executed tomorrow." That is, the composer, while acknowledging that key decisions in structure and policy need to be made, recognizes that making every decision critical, absolute, and perfect, results in bigger risk and higher expense than having a (marginally) less-than-perfect architecture.

From an enterprise composition perspective, composers should apply a customer-centric view following a seven-step process:

1. Define customer goals.
2. Determine how to measure achievement of those goals.
3. Compose a strategic target state that accomplishes those goals.
4. Define the next tactical state on the path to the strategic (i.e., final) state.
5. Assess which of customer's goals will be met in that next incremental implemen-

Table 1: *Comparison of Enterprise Architecture and Enterprise Composition*

Enterprise Architecture Problems	Enterprise Composition Solutions
• Imposes a rigid abstract specification on all aspects of design.	• Establishes a minimum set of flexible interfaces between existing enterprise components.
• Requires mandated modernization efforts just to comply with architecture.	• Focuses on integrating existing capabilities. Modernizations are driven by improved operational processes.
• Primarily justified by cost savings through information technology efficiencies such as enterprise licenses and reduced life-cycle costs.	• Primarily justified by improved higher-level mission processes with IT efficiencies also applicable.
• Is technology-centric with either an Enterprise Resource Planning or a particular commerical off-the-shelf vendor product set as the *Silver Bullet*.	• Is mission-centric and focused above the technology infrastructure.
• Results in agonizingly slow decisions focused on making the *right* choice followed by possible holy wars demanding endless justification of every decision.	• Results in customer-centric decisions based on what works.
• Measures compliance and technology efficiencies through reduction of resources (e.g., systems turned off, reduced operations staff, consolidated hardware and software).	• Measures delivered capabilities and mission efficiencies tied to enterprise metrics (e.g., cost/flying hour, mission capability, kill chain cycle time).

tation of the architecture.
6. Determine how customer goal metrics (to be discussed further in a section that follows) will improve.
7. Commit to those improvements.

While the first three steps are often the easiest, step four is the most important and typically one that many enterprise architects overlook. That is, determining how the enterprise and its existing resources get from their current state to the strategic state (i.e., determining what the most efficient and timely path is to incrementally achieve this [a road map to the] final state, and establishing a process to determine what the next step should be in that direction given the current state and other requirements that have evolved since the last incremental change in the whole EIT). This determination involves steps four through seven.

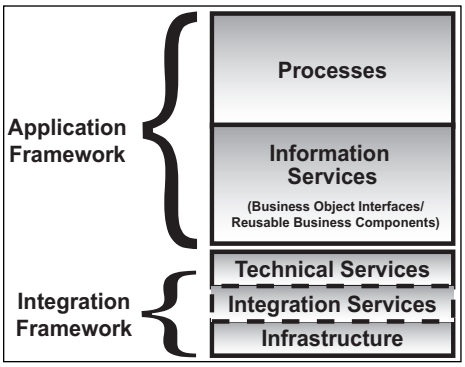
In this way, when the next increment is fielded, its success will be judged not on meeting a date but by measuring how well customer's goals are met. This establishes consistency in the direction of enterprise improvement from increment to increment and through leadership changes.

EIT Framework Boundary Definition Process

As stated previously, composers break the enterprise system architecture into parts. These parts often are organized into a framework within which components providing certain services reside. The Global Combat Support System-Air Force (GCSS-AF) in Figure 1 shows an example of the boundaries in an EIT framework. Enterprise composition guides the composer to minimize the enterprise boundary points to *natural* boundaries and enforce those minimum boundaries rigorously. This insight is the result of the composer following these process steps:

1. Study the problem and solution domain.
2. Correlate the solution domain's technical architecture with existing standards, products, and practices.
3. Define natural boundaries that cleanly separate the EIT into services (see examples in Figure 1).
4. Define objective criteria for boundary implementation.
5. Communicate all boundary information to all enterprise architecture stakeholders.

Figure 1: *GCSS-AF EIS Framework Boundaries*



The GCSS-AF enterprise information system (EIS) [7] shown in Figure 1 has two layered boundaries: the Application Framework and the Integration Framework. The natural dividing line between these layers is the natural separation between Air Force mission information and commercial IT. All Air Force mission-specific information is in the Application Framework, and all generic IT enablers are in the Integration Framework.

Within the GCSS-AF frameworks [8] there are further sub-boundaries or layers. In the Application Framework, the Open Application Group (OAG) Interface Specification [9] provides a natural boundary (or interface) for services upon which components supporting the GCSS-AF Air Force Doctrine 2-4 [10] can be structured. The doctrine creates organizational and information stewardship responsibilities mapped to the OAG standard components such as Inventory, Warehouse, General Ledger, or Budget. In addition, the OAG Interface Specification provides a set of extensible, coarse-grained component boundaries supported by National Institute of Standards and Technology content and syntax tests.

The Application Framework components rely on services provided by the Integration Framework, which relies on standards such as Kerberos, Lightweight Directory Access Protocol V3, Java Authentication and Authorization Service, Public Key Infrastructure, eXtensible Markup Language, HyperText Transfer Protocol, HyperText Markup Language, Web services, Structured Query Language, Portable Operating Systems Interface, Transmission Control Protocol/Internet Protocol, Simple Object Access Protocol, Java 2 Enterprise Edition, or network to create natural security, view, persistence, and messaging boundaries. Objective tests are based on reference implementations of the pertinent standards.

By communicating these boundaries effectively throughout the enterprise, the composer enables the rapid delivery (i.e., composition) of capabilities. This allows implementers to focus within their bounded areas of concern and eliminates the need to address areas outside their particular area of concern. This approach results in a reduction of overall life-cycle cost through reuse of existing services in the GCSS-AF EIT.

EIT Product Selection Process

When enterprise architects address the selection of commercial off-the-shelf products to implement the technical architecture of an enterprise system, they usually start by focusing on each product's capabilities and cost (initial and life-cycle). They conduct extensive trade studies documenting the requirements, weighing the requirements, and assessing the products against those weights.

Often it is the case that, at the end of the evaluation process, the difference between the top products is not statistically significant. Furthermore, a month later the results could change because a new version is released, the chosen product has problems during implementation, or the architect comes to the conclusion that most of the top products could have *done the job* in the first place. Enterprise composition guidelines help the composer improve the product selection process by focusing on a more customer-centric approach rather than a technology-centric approach. This agile and incremental process consists of the following steps:

1. Define the minimum set of mandatory features the customer requires in the product.
2. Determine what existing customer enterprise assets satisfy any of the mandatory features, and allocate them to those assets.
3. Perform high-level, paper, and trade studies on the remaining unsatisfied features using assessments by industry analysts like Gartner, Giga, or Forester to enable a down select to a few products.
4. Instead of taking a technology-centric approach, ask each vendor to provide product compliance levels against the remaining mandatory features. The next step reflects the customer-centric enterprise composition view, as the composer would now ask each vendor to provide at least two reference accounts where existing vendor customers are already using the product in a similar context.
5. Create a survey of the pertinent questions to ask these customer-reference accounts.
6. Set up calls to those customers.
7. Collate the survey results to be used as the prime input to the final selection.
8. Look at the leading candidate product and compare it to the existing personnel skills in the enterprise.
9. If there is a major disconnect between the skill set required to implement the product and the existing skills in the enterprise, then consider the next candidate. The result may be that a less desirable product is preferable because it could be implemented by the enterprise at less cost and risk.

Following this customer-centric, enter-

prise composition-based selection process is usually less expensive than a rigorous, technical, architecture trade-study approach and leads to a product proven to work with built-in expertise from the reference account.

Enterprise Metrics

Architecture metrics have always been a difficult topic to quantify because of their multidimensional nature and lack of good modeling tools. Often these metrics are technology-focused and deal with the performance attributes of the system such as throughput, up time, or even implementation cost. From an enterprise-composition perspective, enterprise architecture metrics measure strategic enterprise goals. In the case of the U.S. Air Force., a set of enterprise productivity measures could include mission capability (aggregate status of the force) and sortie generation capacity. From an enterprise-composition perspective, the metrics chosen are used to show how each increment of the EIT (the addition of new technology/services or mission capabilities) has moved the enterprise closer to the strategic enterprise goals.

Incremental Enterprise Architecture Development Process

Most enterprise architects use the Unified Modeling Language as the design notation to document their architectures [6]. The Unified Software Development Process (USDP) [11] provides a sound, repeatable process model for software development and can be used by enterprise architects to establish the minimum, mandatory artifacts for each increment of the enterprise architecture (e.g., an analysis, tactical, and strategic collaboration diagram would be used to document the goal state and each incremental step).

From an enterprise composition perspective, USDP needs to be extended at both ends of the life cycle. For example on GCSS-AF, the requirements definition phase is proceeded by a business model specification using activity diagrams and use-case diagrams, and the deployment/production phase is extended by using a component repository of XML metadata to facilitate message routing and integration of services.

Enterprise Composition Patterns

The role architecture and design patterns [12] play in enterprise architecture is well recognized [5]. The underlying premise of a design pattern is that,

> ... each pattern describes a problem that occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice. [13]

From an enterprise composition perspective, the key patterns that are most useful to the enterprise architect can be labeled as boundary patterns in that they help organize the components and their interfaces so that they form natural boundaries and hide some of the dependencies that otherwise would complicate these interfaces. Following are the boundary patterns discussed in the next sections (sections 2, 3, and 4 are applicable within application framework):

1. Layers Pattern.
2. Canonical/Domain Model Pattern.
3. Model/View/Controller Pattern.
4. Façade Pattern.

Layers Pattern

Usually the Layers pattern is used to define the highest-level boundaries of an EIS. One of the earliest and most widely known examples of the Layers pattern is the seven-layer International Organization for Standardization Reference Model (i.e., Application, Presentation, Session, Transport, Network, Data-Link, and Physical layers). Fowler states that the purpose of layering is "to break apart a complicated software system," [5] giving an architect the following:

1. Intellectual control and understanding within layers.
2. Flexibility to substitute appropriate capabilities at layers.

The number of layers varies according to the area of focus. Fowler advocates three layers [5] (Presentation, Domain, and Data Source). Within GCSS-AF, the EIS is divided into two main layers, or frameworks, which are subdivided into five sub-layers (see Figure 1).

Canonical/Domain Model Pattern

From an enterprise composition perspective, the Canonical/Domain Model pattern can be used to reduce the number of point-to-point interfaces. This allows the architect to select the best tools for his or her job, know the primary interfaces, and only support interfacing to the canonical model decoupling the point-to-point interfaces.

Model/View/Controller Pattern

The Model/View/Controller (MVC) pattern is another long-standing technique used by system designers and architects to separate (via boundary layers) the functionality (the model) from the presentation (the view) through an intermediary interface boundary (the controller) that communicates between component's model and the view.

A derivative of the MVC pattern is the Document View pattern. In this case, the view is dictated by the graphical user interface development tools that link graphical forms with a relational database. The separation of concerns is still maintained between the document/model and the view but the controller function is subsumed within the view function. This is a good pattern for reports and is well supported by Microsoft's Toolset keeping the view synchronized with the record set that typically provides the document or model.

Façade Pattern

The Façade pattern is used to wrap a component in order to simplify its interfaces. A façade can be as simple as an extended script Language Translation script for an XML message or as complicated as an Enterprise Application Integration Extract/Translate/Load tool for a complex, proprietary system interface.

Enterprise Maturity Levels

From an enterprise composition perspective, enterprises that employ EIT mature in a pattern similar to the levels described by the Software Engineering Institute's Capability Maturity Model®. Large, enduring enterprises follow a pattern as they mature. In that pattern, an enterprise determines what governance will provide the most effective support in evolving the EIS. Furthermore, enterprises evolve over long periods of time, and the type and amount of legacy system technology can determine their maturity as well. Using Moore's law as the driving force in the IT industry, the timeline in Figure 2 summarizes the technology shift compared to the number of processors per person.

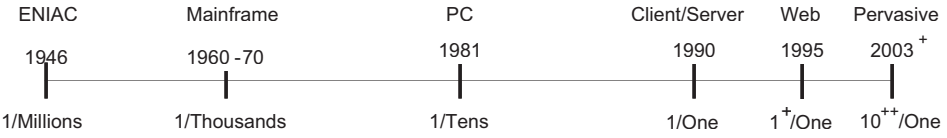


Figure 2: *Timeline of Technology Shift Compared to Processors Per Person*

You should note that most enterprise processes were automated in the 1960s and 1970s when there was little engineering guidance and some severe technology constraints. The client/server era started the shift from mainframe mindset in that most functional areas felt the central enterprise staff was slow and unresponsive, and the central staff felt that the functional depart-

Maturity Level/ Attributes	Chaotic	Dictatorial	Capability	Optimized
Decision	• Sub-optimal, focused on specific need. • Vendor/Technical criteria. • Looking for silver bullet.	• Sub-optimal, focused on specific need. • Mandated standards. • ERP focus.	• Optimum product selections considering all costs. • Customer-centric approaches. • Core competencies identified and emphasized. • Business case analysis of mandates.	• Driven by optimum enterprise growth. • Members of key industry leadership groups. • Core competencies target predator capabilities.
Artifacts	• Closely coupled throughout. • Holistic deliverables all required capabilities every deliverable. • No separation of layers.	• Layering framework. • Infrastructure administration efficiencies. • Enterprise licenses.	• Everything from Dictatorial. • Canonical Model. • Enterprise Value Chains.	• Information warriors creating own weapons against Canonical Model. • In process measurements mission performance models for continuous improvement.
Metrics	• Meet delivery date.	• IT efficiencies. • Percent Earned Value measurements.	• IT efficiencies. • Earned Value based on complete deliveries work products. • Enterprise Mission Measures.	• Metric capture built-in to Enterprise Value Chains. • Direct measurement of each enterprise contribution.
Rewards	• Subjective assessment.	• Subjective assessment.	• Rewards tied to measured capability delivery.	• Rewards tied to measured capability delivery.

Table 2: *Enterprise Maturity Levels*

ments did not understand the complexities of what they were asking. It was at this point in time that the functional departments took control of their own destiny and within their own control and budgets built the tools that allowed them to respond to mission demands.

These two sets of systems continued to devolve apart along their own paths. The central systems held onto the enterprise applications – such as payroll – while the departments grew department-centric processes starting with simple analysis tools and reports but growing into sophisticated mission critical systems. Soon, with the Web and office tools collecting information from the abundance of individually designed applications, it became clear that the industry had lost control of the information. Today, enterprises are consolidating and trying to get control of their information resources.

Assessment EIT Maturity and Appropriate Governance

Enterprises are trying to regain control of their information flow without restricting the benefits gained from distributed composition. Table 2 details the maturity levels that an enterprise evolves through, and the respective decision characteristics, artifacts delivered, measurements taken, and rewards criteria for success.

Summary

Enterprise composition extends the range of an enterprise architect to allow him or her to address complex, evolving enterprise system architectures. This article has described the processes, metrics, and architectural design patterns that have demonstrated applicability in dealing with these unique challenges.◆

References

1. Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance Architecture Framework, v.2.0. 18 Dec. 1997 <www. afcea. org/education/courses/archfwk2.pdf>.
2. Federal Enterprise Architecture Framework, v.1.1. Sept. 1999 <www.cio.gov/ documents/fedarch1%2Epdf>.
3. The Open Group Architecture Framework <www.opengroup.org/products/ publications/catalog/ar.htm>.
4. Zachman Framework <www.zifa.com>.
5. Fowler, Martin. Patterns of Enterprise Application Architecture. Boston, MA: Pearson Education Inc., Mar. 2003.
6. Clements, Paul, et al. Documenting Software Architectures: Views and Beyond. Addison-Wesley, 2003.
7. U.S. Air Force. Global Combat Support System-Air Force UML Model, 2003 <www.gcss-af.com/cfs/uml>.
8. Global Combat Support System-Air Force <www.gcss-af.com>.
9. Open Applications Group Inc. Open Applications Group Interface Specification v.8. OAGI, 2002 <www.open applications.org>.
10. Cresta, Lt. Col. James. U.S. Air Force, Combat Support Air Force Doctrine Doc. 2-4. U.S. Air Force, 22 Nov. 1999 <www.dtic.mil/doctrine/jel/service _pubs/afd2_4. pdf>.
11. Jacobson, Ivar, Grady Booch, and James Rumbaugh. The Unified Software Development Process. Addison Wesley Longman, Inc. 1999.
12. Gamma, Eric, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.
13. Alexander, Christopher, et al. A Pattern Language. New York: Oxford University Press, 1977.

About the Author

John Wunder is a certified Lockheed Martin architect and has been the lead system architect/composer on the Global Combat Support System-Air Force since 1999, and was lead architect for the Dow Chemical Process Control and software architect for the U.S. Army battlefield digitization project. Wunder has been involved in information technology for more than 20 years.

Lockheed Martin
Systems Integration
1801 State RTE 17C
MD 0605
Owego, NY 13827
Phone: (607) 751-6096
Fax: (607) 751-2538
E-mail: john.wunder@lmco.com