THE Open GROUP

IT Architecture and Design Patterns/Assets

An Initial Assessment

A White Paper by:

Jonathan Adams (Retired IBM DE)
George Galambos (IBM Fellow)

December, 2014

IT Architecture and Design Patterns/Assets: An Initial Assessment

Document No.: W14E

Published by The Open Group, December, 2014.

Any comments relating to the material contained in this document may be submitted to:

The Open Group, 44 Montgomery St. #960, San Francisco, CA 94104, USA

or by email to:

ogpubs@opengroup.org

Table of Contents

Boundaryless Information Flow ™
achieved through global interoperability
in a secure, reliable, and timely manner

Executive Summary

This White Paper builds further on the material on Architecture patterns introduced in Chapter 25 of the TOGAF® 9.1 standard. It describes a family of IT architecture and design patterns that should be of value to both Enterprise Architects and Solution Architects working with the TOGAF standard. For those wishing to extend these patterns or harvest patterns in additional areas, guidance is also offered in the form of a pattern development process based on many years of experience in developing the Solution Architecture patterns. This document also shows how the spectrum of Solution Architecture patterns and assets can be mapped usefully as a Pattern/Asset Continuum against the TOGAF Solutions Continuum.

Enterprise and Solutions Architects can benefit from this White Paper, which supports The Open Group vision of Boundaryless Information Flow™, by showing how Architecture patterns can help enterprises and the industry to harvest a number of proven end-to-end IT solution architectures and designs.

Introduction

In the TOGAF 9.1 standard, Chapter 25, there is a placeholder for Architecture patterns with "more rigorous descriptions and references to more plentiful resources". This White Paper has been written to further contribute to the material on Architecture patterns by describing the latest work on Solution Architecture patterns [4].

There can be many possible patterns in any field of endeavor so the following sections include a rationale for a family of IT architecture and design patterns that could be of value to both Enterprise Architects and Solution Architects using the TOGAF standard. For those wishing to extend these patterns or harvest patterns in additional areas, guidance is offered in the form of a pattern development process based on many years of experience in developing the Solution Architecture patterns.

The Requirement for IT Architecture and Design Patterns/Assets

Patterns can be identified and harvested in many fields of IT, so a key requirement is to determine the key goals of potential pattern users (e.g., Enterprise Architects) and then target patterns and assets to support those goals.

A Solution Architect left to their own resources can certainly be expected to lead the architecture and implementation of a successful solution. However, several such Solution Architects are likely to be needed with a variety of specializations such that, over time, there is little commonality of architecture, design, products, product versions, and system management. This can result in an operational nightmare of complexity and excessive costs.

Enterprise Architects are ultimately responsible for the enterprise delivery of well-managed and effective business solutions which conform to enterprise standards and strategies. It is very easy to focus only on single products or vertical application and infrastructure stacks which represent these enterprise standards and strategies, but then forget that the Solution Architects need help to deliver end-to-end hardware, middleware, and application-based business solutions using these standard products or stacks. It is important that enterprise re-usable assets and patterns should include individual components, vertical stacks of components, and horizontally integrated components capable of supporting and/or delivering and/or managing end-to-end business solutions.

It is therefore very valuable to have an Enterprise Architecture function which reviews solutions across the enterprise and works with Solution Architects and the industry to harvest a number of proven end-to-end IT solution architectures and designs which can be implemented using a manageable number of strategic product choices.

Valuable Attributes for IT Architecture and Design Patterns/Assets

IT architects in the roles of Enterprise Architects and Solution Architects regularly have to apply their skills in conceptualization, abstraction, decomposition, composition/integration, asset re-use, visualization, communication, etc. It is important that the IT architecture and design patterns support the IT architect when performing some or all of these tasks.

Positioning Existing and Proposed Patterns

In the context of developing business solutions, the architecture of the solution should address two major domains: the *functional domain* and the *operational domain*. The functional domain addresses the business functionality of the solution, the structure and modularity of the software components that will be used to implement the business functions, the interactions between the components, and the interfaces between them. The operational domain describes the system organization (such as hardware platforms, connections, locations, and topology), the placement of software and data components, the service requirements (such as performance, availability, and security), and the system management (capacity planning, software distribution, backup, and recovery).

There is clear evidence of patterns in both of these domains. *System-level patterns* identify and describe the overall structure and interactions that can occur between components of a system. These patterns can typically be used during the high-level design phase to develop the structure of the overall application solution. The authors of *Pattern-Oriented Software Architecture: A System of Patterns* [1] identified patterns for system architecture at a higher level than the original design patterns. Their patterns relate to the macro-design of system components such as operating systems or network stacks.

Design patterns represent the original description of more granular patterns that can aid the design of software components. In *Design Patterns: Elements of Re-Usable Object-Oriented Software* [2], the authors document 23 patterns recognized as providing accepted solutions for recurring problems in object-oriented design. These patterns help software designers establish the structure and operation of the code that supports the functional components of the application.

In subsequent years there have been many additions to the patterns literature, but these patterns have focused primarily on the analysis, design, and programming phases of application development, dealing with the structure and behavior of the code that supports the application in the *functional domain*.

The Solution Architecture patterns extend the domain of software patterns to earlier phases of the application development lifecycle, as shown in Figure 1.

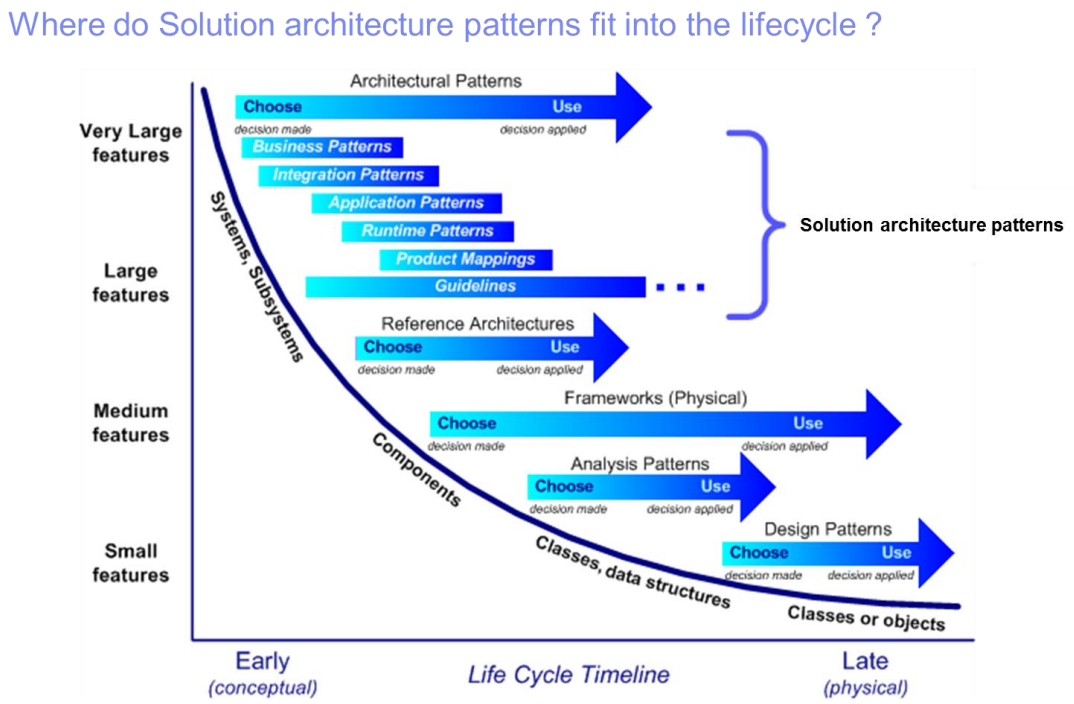

Figure 1: Positioning of Solution Architecture Patterns

The Solution Architecture patterns are based on a very coarse-grained view of the *functional domain* (based on the fundamental business interactions) together with many proven high-level design and architecture patterns in the *operational domain*.

These Solution Architecture patterns therefore help us understand and analyze complex business problems and break them down into smaller, more manageable functions that can then be implemented using system and design patterns appropriate to the *functional domain*.

Characterizing Required Solution Architecture Patterns/Assets

Because Solution Architecture patterns need to be used early in the solution development lifecycle, they must be abstracted from working solutions (like all good patterns) but generalized sufficiently to make them relevant to a wide range of business solutions across many industries. It should be possible to match the given business task or process to be automated with an end-to-end Solution Architecture pattern such that it embraces the consumer of the business task or process, the automation of that function, and the confirmation of success to the invoking consumer where required. It is to be expected that a number of these business tasks or processes will need to be combined into a compound business process – therefore it will also be important for these Solution Architecture patterns to be composable. If this were the full scope of the Solution Architecture patterns they would have limited value – but if they were somehow linked to alternative high-level design options together with outline logical and physical operational model options this could provide significant extra value.

Process to Derive Solution Architecture Patterns/Assets

There are many ways to abstract individual patterns from proven assets – the challenge is to abstract a family of Solution Architecture patterns with wide applicability, to harvest additional assets which can be used to elaborate a design, and to enable the discovery of these patterns and assets by an architect during his/her solution design process.

The following steps have been found to be useful in developing such a pattern language:

1. Identify and understand the target audience for your patterns and practices.
2. Brainstorm the key activities/artifacts involved in the best practice.
3. Organize the related activities into groups.
    a. Focus on the core; hide the non-essential.
    b. Group the activities by their characteristics which relate to the nature of the problem – maybe by business interactions, by roles, by practices, by technology elements, etc.
4. Start to shape a searchable hierarchy from the groups/activities.
    a. In the case of solution architecture activities, start with the business problem, then logical decomposition, then physical solution structure.
    b. Target 80% of requirements.
    c. Non-overlapping groups become new branches/roots of the hierarchy – allowing for future composability.
    d. Apply separation of concerns within a group to add depth to the hierarchy.
    e. Design the hierarchy to reflect the phases of the associated method.
    f. Keep the cardinality down by use of variations.
5. Determine which of the activities/artifacts can be accelerated by use of assets.
6. Exploit graphic abstractions to provide consistent visual cues for pattern assets.
7. Select one or two use-cases to test/improve one or two paths of the hierarchy.
8. Review the use-cases with activity Subject Matter Expert (SME), audience SME, and patterns SME.
9. Document consistent pattern attributes and decision tables for each group level within the hierarchy.
10. Repeat with use-cases for more paths.
11. Review the hierarchy, visual cues, practices, patterns, and other assets with activity SMEs, audience SME and patterns SME.
12. Iterate and refine.

Applying the Process to Derive Solution Architecture Patterns/Assets

Applying steps 1 to 2 above in 1998, a number of business solutions and technical activities were identified, as shown in Figure 2.

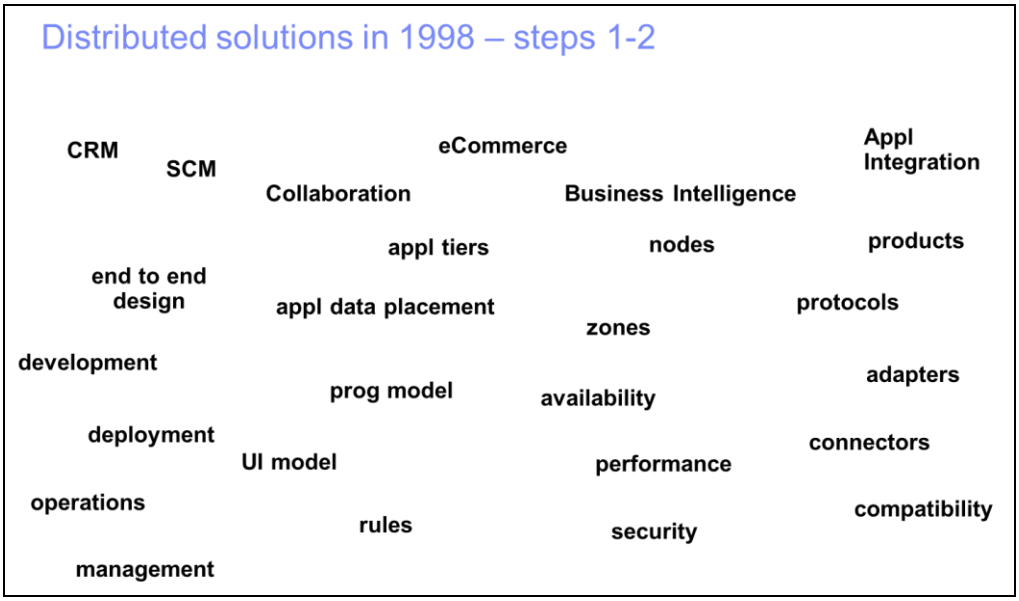

Figure 2: 1998 Business Solutions and Technical Activities

By applying step 3 based on separation of roles we obtained Figure 3.

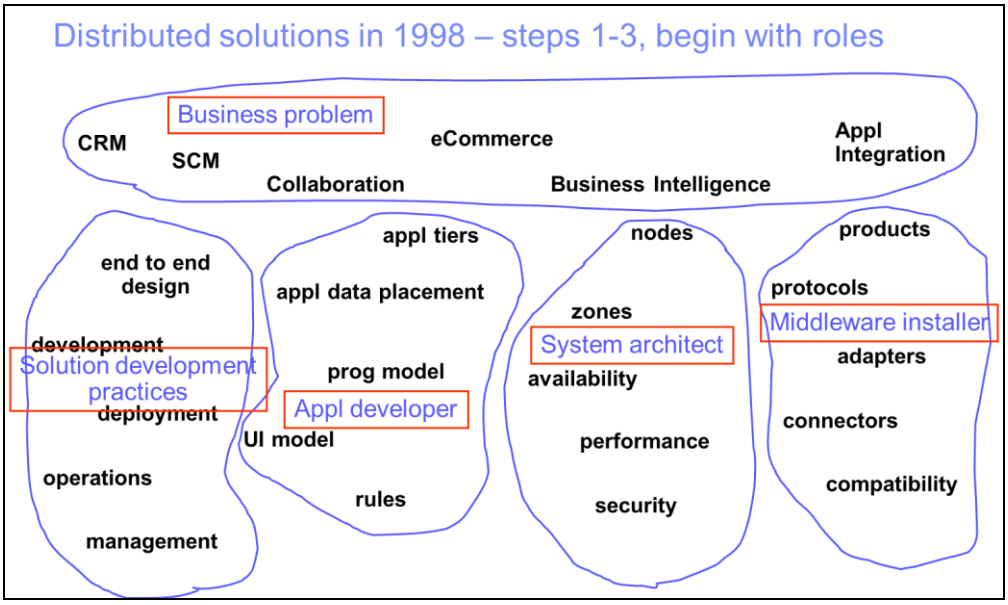

Figure 3: Separation of Activities by Role

The next steps 4 to 9 resulted in an inverted tree structure based on Figure 4.

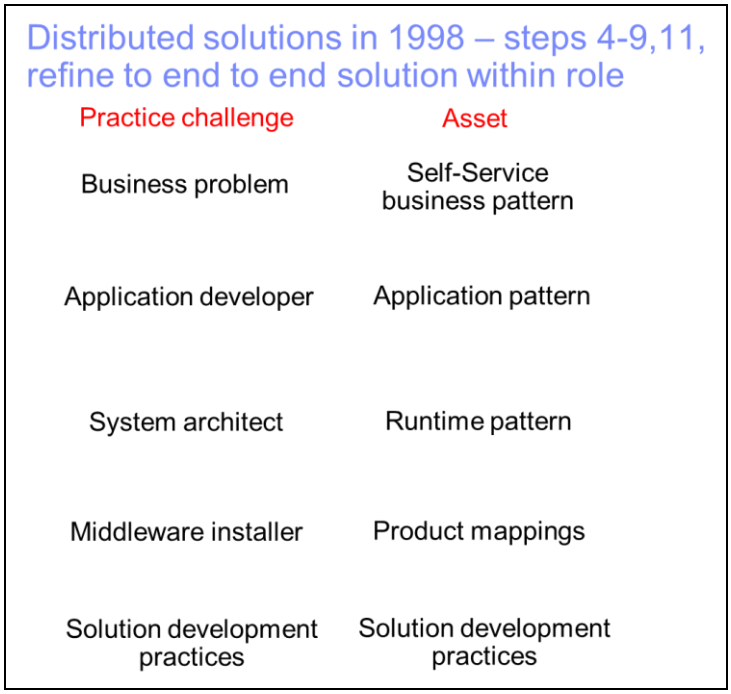

Figure 4: Outline Design and Refinement of an End-to-End Solution

This structure can now be used to populate a hierarchy which flows from a business context to a logical and then physical implementation – thus enabling a very efficient discovery of relevant assets and patterns.

Solution Architecture Patterns/Assets

The early work on the Solution Architecture patterns coincided with the early days of the commercial Internet. As a result, most business solutions exploiting the intranet/Internet required new end-to-end IT solutions. Many years later this is no longer true – and there is commonly a requirement to enhance an existing end-to-end IT solution with a range of fine-grained patterns. Indeed these enhancements will often require a range of heterogeneous services – for example, when many years ago organizations added user feedback (collaborative services) and "other users bought …" (insight services). It is therefore important that any modern pattern-based engineering process should accommodate either new or enhanced business solutions.

For many years IT architects have understood how to decompose a target business scenario into a set of business subsystems and integration subsystems, where each business subsystem corresponds to an end-to-end business task or process. The challenge is then to abstract each business subsystem and match it to some generic business solution characteristics, which we refer to as *Foundation Business patterns*. The "Foundation" qualifier is used because, as we have seen, there may be a need to take a Foundation Business pattern and add to it with some finer-grained patterns.

Based on an extensive analysis of modern IT solutions, we have identified seven Foundation Business patterns, as shown in Figure 5.

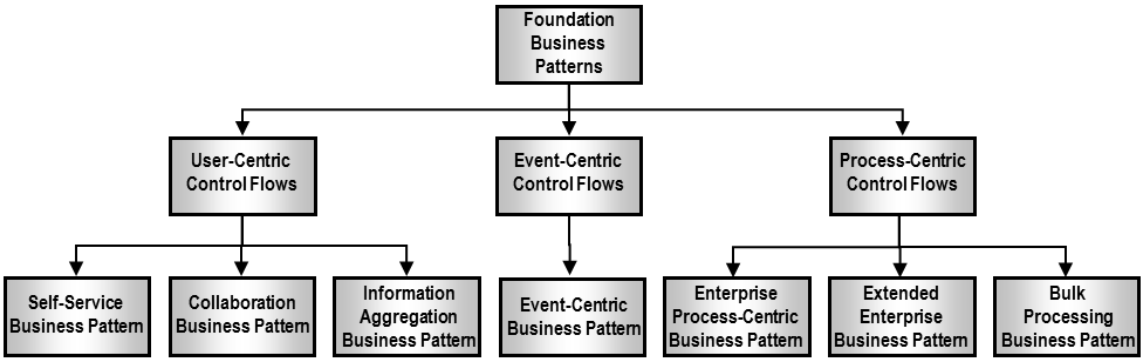

Figure 5: The Foundation of Most Modern Business Systems

In order to capture some fine-grained Solution Architecture patterns, it is valuable to decompose the business subsystem into its constituent logic layers. With the advent of modern distributed applications built on Service-Oriented Architecture (SOA) principles, it is now necessary to allow business subsystems to be invoked programmatically through a channel logic layer and to support loose-coupling to both business logic and data logic using a business logic integration layer and a data logic integration layer. These internal integration layers enable the flexible connection of the other logic layers within a single business subsystem. In addition, end-to-end integration subsystems can be used to provide either front-end integration or back-end integration of two or more business subsystems such that the composite subsystem also has end-to-end integrity. This is still a fairly high-level decomposition so it is helpful to further specialize these logic layers into Logic patterns, as shown in Figure 6.

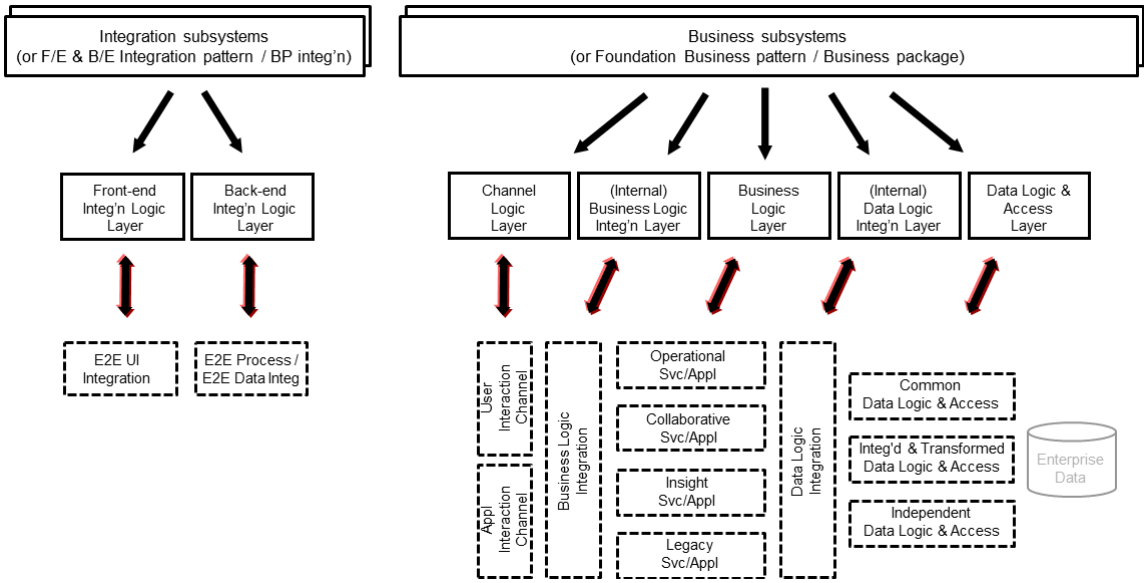

Figure 6: The Specialized Logic Layers (*aka* Logic Patterns)

Finally, these Logic patterns can be used as anchor points for a family of Atomic integration and Atomic application patterns which lead to alternative design approaches and ultimately products for enhancing business subsystems, as shown in Figure 7.

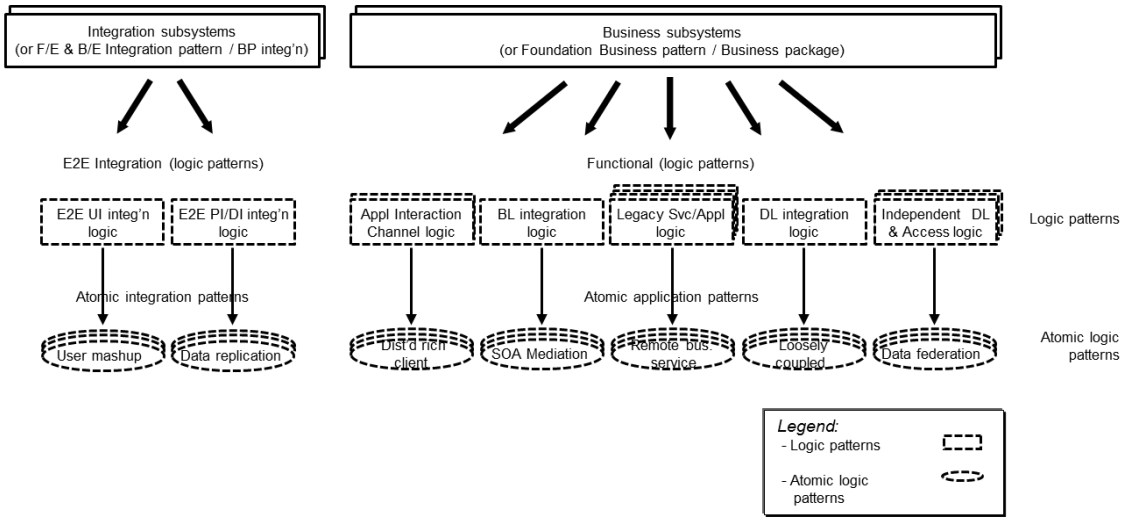

Figure 7: Atomic Logic Patterns – Atomic Application Patterns and Atomic Integration Patterns

Another essential task for IT architects is to visualize and communicate the existing and proposed enhancements to executives and fellow practitioners. This can be done at a high level for executives by overlaying a subsystem relationship diagram with the appropriate Foundation Business patterns and Integration patterns, as shown in Figure 8.

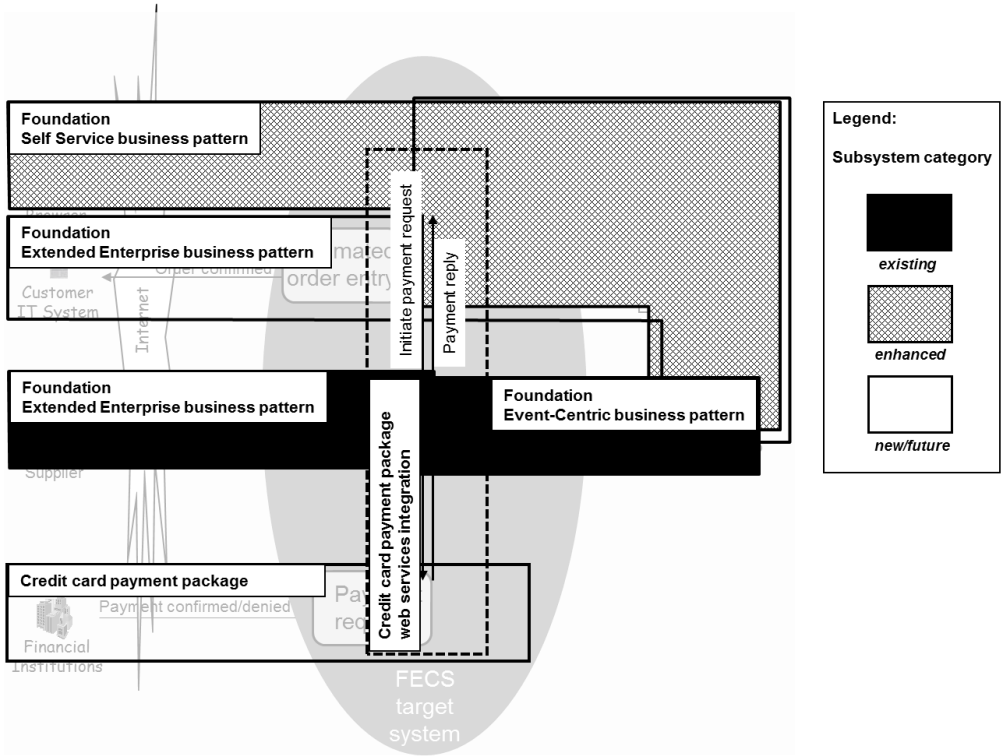

Figure 8: Outline Solution Design

This is very valuable for a high-level view, but a more detailed view is needed for fellow practitioners, as shown in Figure 9.

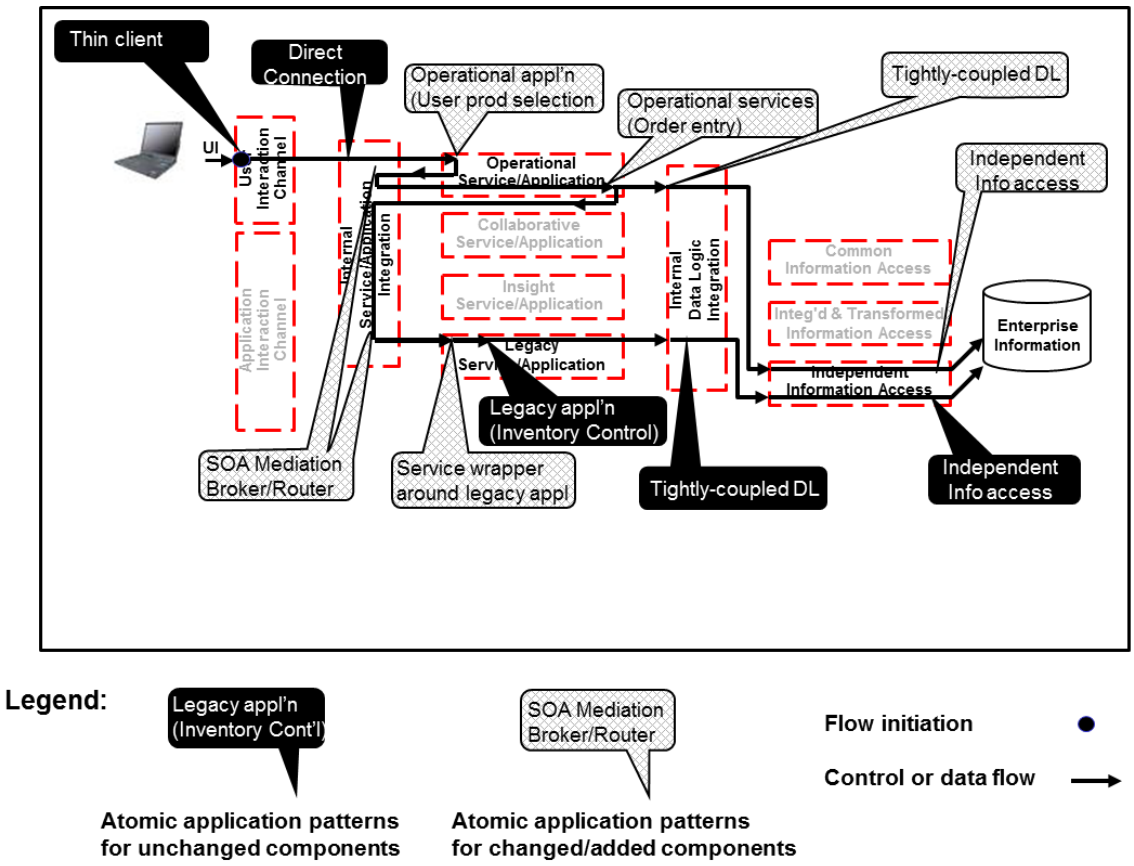

Figure 9: Atomic Application Patterns for the Unchanged/Changed/Added Components

This has been a brief introduction to the latest Solution Architecture patterns – when combined with design pattern refinement hierarchies and best practices they can contribute significantly to a number of key architecture tasks:

- Solution architecture analysis (for new and enhanced solutions)
- Strategic solution planning (including merger analysis)
- Operational asset re-use
- Development asset re-use
- Education

Positioning Solution Architecture Patterns/Assets

A brief analysis suggests that the spectrum of Solution Architecture patterns and assets can be mapped usefully as a Pattern/Asset Continuum against the TOGAF Solutions Continuum, as shown in Figure 10.

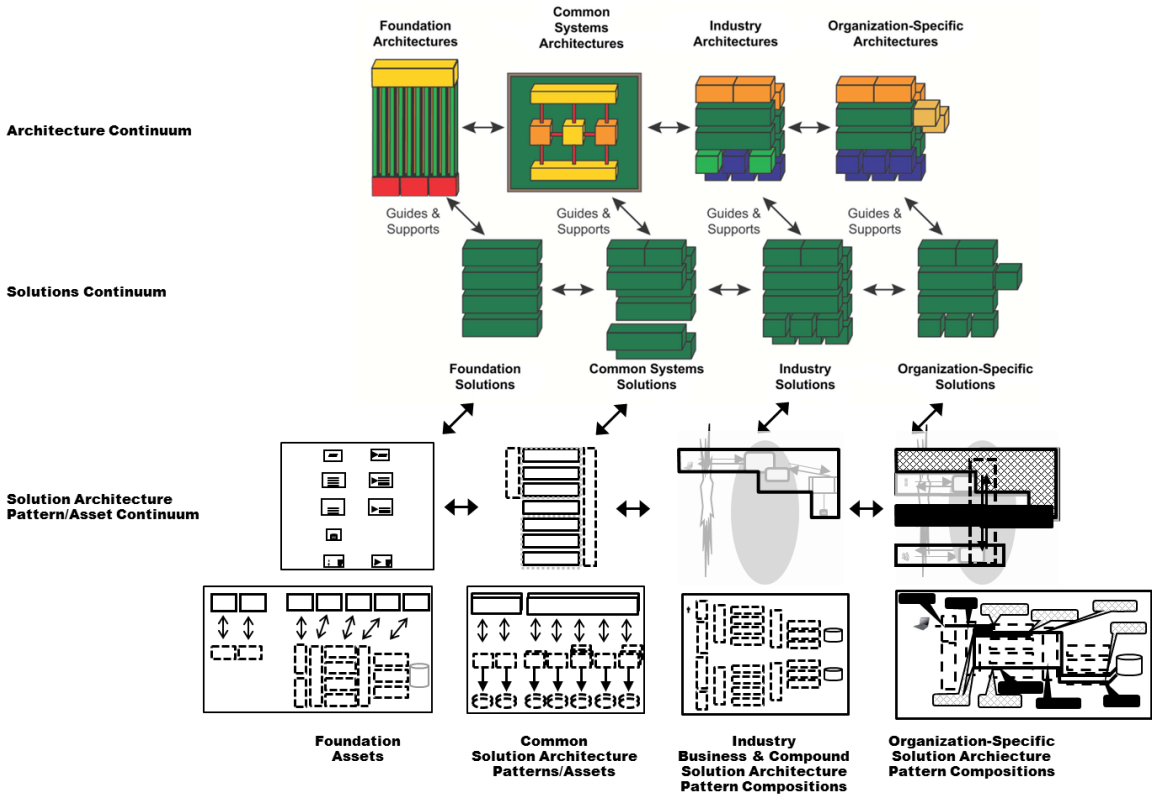

Figure 10: The Pattern/Asset Continuum Mapped against the TOGAF Solutions Continuum

References

[1] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal: Pattern-Oriented Software Architecture: A System of Patterns, New York: John Wiley & Sons (1996).

[2] E. Gamma, R. Helm, R. Johnson, J. Vlissides: Design Patterns: Elements of Re-Usable Object-Oriented Software, Reading, MA: Addison-Wesley (1995).

[3] J. Adams, G. Galambos, S. Koushik, G. Vasudeva: Patterns for e-business: A Strategy for Reuse, IBM Press (2001).

[4] J. Adams, G. Galambos, S. Koushik, G. Vasudeva: Solution Architecture Patterns (*aka* Patterns for e-business), IBM Press (Planned 2015).

[5] TOGAF® Version 9.1, Enterprise Edition, Open Group Standard (G116), December 2011, published by The Open Group; available at www.opengroup.org/bookstore/catalog/g116.htm.

About the Authors

Jonathan Adams

Jonathan Adams is a retired IBM Distinguished Engineer. He helped IBM customers over 42 years in their development of business solutions based on end-to-end middleware. For the last 15 years of his career he managed the project and technical development of the Patterns for e-business which were later re-engineered and then extended into the Solution Architecture patterns. He led a worldwide team of IBM architects and solution designers who were responsible for the abstraction and documentation of Solution Architecture patterns based on industry best practice.

George Galambos

Dr. Galambos is an IBM Fellow who has spent 40 years in Information Processing as a systems engineer, focusing on high performance, high availability systems and networks, with emphasis on quantitative methods. He has performed assignments in management consulting on IS issues, project reviews and risk assessments, evaluation of new opportunities and defining strategic IT plans, in addition to designing and remediating large business systems. He has consistently focused on the extraction of re-usable experiences and artifacts leading to the definition of design methods and to the recent work on Solution Architecture patterns. His current work is on monitoring, event management, and automation as a means of improving the efficiency and quality of IT operations. His other interests include high value cloud services and the opportunities presented by the convergence of emerging technologies.

About The Open Group

The Open Group is a global consortium that enables the achievement of business objectives through IT standards. With more than 400 member organizations, The Open Group has a diverse membership that spans all sectors of the IT community – customers, systems and solutions suppliers, tool vendors, integrators, and consultants, as well as academics and researchers – to:

- Capture, understand, and address current and emerging requirements, and establish policies and share best practices
- Facilitate interoperability, develop consensus, and evolve and integrate specifications and open source technologies
- Offer a comprehensive set of services to enhance the operational efficiency of consortia
- Operate the industry's premier certification service

Further information on The Open Group is available at www.opengroup.org.