SYNERGIZING METHODOLOGIES TO ACHIEVE ARCHITECTURE TARGETS
(towards SOA)

Author

Rakesh Radhakrishnan
Lead IT Architect
Sun Microsystems
SYNERGIZING METHODOLOGIES TO ACHIEVE ARCHITECTURE TARGETS

Table of Contents

I Abstract..............................................................................................................................................Page 3

II Introduction.....................................................................................................................................Page 4

III Alignment of Methodologies to Architectural Layers.........................................................Page 18

IV Conclusion......................................................................................................................................Page 25
SYNERGIZING METHODOLOGIES TO ACHIEVE ARCHITECTURE TARGETS

Abstract

Contemporary Enterprise IT Architect’s are perplexed with the different Tactics, Techniques and Tools – such as Architectural Framework/Methodologies from MDA, RUP, TOGAF, ITIL, Prince 2, etc., and Techniques – such as Meta Model Repository based EII, Patterns (Architecture and Design), Abstraction Layers, ABB/SBB, Description, Specifications, Markup, etc. This paper identifies a taxonomy and an organizing principle of these Tactics, Tools and Techniques that are so immensely useful for (Enterprise/Systems) Architects to do there job of building Architectures that are strategically maneuverable, i.e., Architectures that can handle changes in requirements and are inherently flexible, constantly adaptable and ever extensible and completely agile. Agile Architectures also referred to, as “Organic Architectures” by many is fundamentally an “Architecture” that embraces principles such as “Modifiability”, “Portability”, “Integrate-ability”, “Reusability”, “Replace-ability”, “Implement-ability” and “Extensibility”. More complex Architectural Qualities than the traditional Qualities such as “Security”, “Availability”, “Scalability”, etc. This effort is geared towards harnessing methodologies, tools and techniques that can leveraged effectively for building “Agile” Architectures.
Introduction

One amongst the core principles and concepts taught in Executive IT Management programs from Harvard in Massachusetts to Wharton in Pennsylvania and Stanford in California to Darden/McIntire in Virginia is the need for an Enterprise Architecture that has a well-defined Information/Data Architecture and one that is well aligned with the Application/Process Architecture and the underlying Technology/Infrastructure. If these three layers of Architectures are truly well aligned and align-able on an ongoing basis then the Enterprise Architecture is said to be one that is Agile in nature. Therefore, Architecture and the role of Architects in modern times primarily deal with alignment:

- aligning the Business with the Technology using SOA,
- aligning Architectural Approaches around SOA,
- aligning Standards initiatives for Open SOA,
- aligning methodologies towards a Target SOA and
- aligning Attribute Requirements to Patterns within SOA.

This alignment of the Business and Technology, Architectural Approaches, Standards Initiatives (within the respective Industry domain), Methodologies to a Target Architecture and Requirements to Patterns is achieved primarily when there is good alignment of Information/Data Architecture, the Application/Process Architecture and the Technology/Infrastructure.

Information and Data is the lifeblood of an Enterprise – it is needed by all users of systems; employees/management, customers and suppliers/partners to conduct their day-to-day activities. When information and data is provided to Customers via a set of customer-facing applications and services – it is generally referred to as the Enterprise’s CRM (Customer Relationship Management) system. Similarly, when information and data is provided to Suppliers and Partners via a set of supplier/partner-facing applications and services – it is generally referred to as an Enterprise’s SCM (Supply Chain Management) system, and for internal employee and management-facing services and applications it is more commonly known as ERP (Enterprise Resource Planning) systems. When these 3 core CRM, SCM, and ERP systems along with other related business systems (BI, Data-warehousing, etc.) are built on Common and Open Technology/Infrastructure Architecture (a.k.a., by the term – Integrated Information Infrastructure Reference Model- IIIRM) they build what The Open Group refers to as a “Boundary-less Information Flow”.

The approach recommended in this paper to Align Information/Data Architecture with the Application/Process Architecture and the Technology/Infrastructure Architecture is to leverage the advances in five key methodologies. Open standards based methodologies that are defined by open bodies and forums, namely, Object Management Group’s MDA (Model Driven Architecture – Modeling Approach to defining and specifying a System), Software Engineering Institutes ADDM/ATAM/PLA (attribute driven design methodology, architecture trade-off analysis method and product line architecture), The Open Groups TOGAF/ADM (Architecture Framework and Architecture development Methodology), OGC’s ITIL Methodology (IT Infrastructure Library) and OGC’s Prince 2 Methodology (Project Management in a Controlled Environment).

There are certain core/fundamental drivers behind each of these methodologies and each one of them can be used to reach a specific end/target Architecture for a specific layer (such as Data Layer, Application Layer, Technology Layer, etc.).
SYNERGIZING METHODOLOGIES TO ACHIEVE ARCHITECTURE TARGETS

- **OMG’s MDA** can be viewed as a *meta-data/model driven* approach to build a highly integrated Enterprise *Information/Data* Architecture, and more.

- **ADDM/ATAM/PLA** can be viewed as an *attribute driven* iterative approach to build a highly integrated Enterprise *Application/Process* Architecture, and more.

- **TOGAF/ADM** can be viewed as a *service driven* approach to build a highly integrated Enterprise *Technology/Infrastructure* Architecture, and more.

- **OGC/ITIL** can be viewed as a *SLA/OLA driven* approach to build a highly integrated Enterprise *Service Management* Architecture, and more.

- **OGC/PRINCE2** can be viewed as a *control and discipline driven* approach that helps implement a highly Integrated Enterprise *Physical/Deployment* Architecture, and more.

Note: This alignment of Methodology to a specific layer is primarily to identify the significant area of value and contribution the methodology offers, however it is not just limited to that layer alone. For example, MDA from OMG lends itself to Systems Specification that ADM from TOG can leverage. ADDM defines a system design that can be applicable to a family of software systems along with its product line architecture that again can be reused in the ADM process.

If we treat 3 distinct phases – Architect, Implement and Manage, Both the OGC’s methodologies are associated with the Implementation and Management phases of Architecture, wherein Prince2 is geared more towards the Implementation and ITIL/ITSM more around Management. The 1st 3 methodologies are obviously geared to the significant phase associated with Architecting end-to-end Solutions that includes design/specifications for the 3 primary layers of an Enterprise IT Architecture.

However in the Architecture phase, reaping the benefits associated with the blending of MDA, ADDM/ATAM and TOGAF offers tremendous value and an enormous amount of synergy, especially to build “Agile” Data, Process and Technology Architectures that can change in accordance to the need of the Business and other environmental factors (such as legal, logistics, etc.).

Hence the next section first describes each of these methodologies briefly, followed by a section that describes key Tools, Techniques and Tactics (TTT) that are relevant in each space. This is then followed with Taxonomy of TTT and how they can be mapped while building real world Enterprise IT Architectures. This is then followed up with a section describing the benefits and value proposition of the blended Architecture Methodology where TOGAF acts the glue between MDA/ADDM/ATAM on one end and PRINCE2/ITIL on the other (the central GLUE). This is followed by a description of the Implementation and Management phases.
Introduction to OMG’s MDA

This is a brief introduction to MDA. For an in-depth understanding a reference book titled “Model Driven Architecture” authored by David S. Frankel and http://www.omg.org/mda will be a good start. The emphasis in Model Driven Architecture is the element associated with Modeling (your data and your systems) in conjunction with meta-data which keeps a record of an enterprise’s architecture in-terms of data, information, application, services, technology and implementation (platform specifics). Since the Architecture is model driven there is an expectance to have a maintain-able Architecture, one where the data, meta-data and models change, over a period of time. Key Characteristics of MDA are:

- MDA suite of standards include Unified Modelling Language (UML); Meta-Object Facility (MOF); XML Meta-Data Interchange (XMI); and Common Warehouse Meta-model (CWM)
- Applying MDA to Enterprise Computing by David Frankel (excellent resource) – move from Machine Centric Computing to Application Centric Computing to Enterprise Centric Computing
- http://www.omg.org/mda Is also a very useful resource
- Sample implementation of MDA's tools - http://www.metamatrix.com (that includes a MOF, XMI and CWM)
- Core value proposition – addresses Enterprise Information Integration with a meta repository of all the data/business models in an enterprise
- Model Driven Architecture - focus on an Integrated Information Architecture (Business Models)
- MDA aids in the development of Conceptual Business Models and Data Models
- Model Driven implies that its typically not platform specific and maintenance over time is possible
- Can scale from one focus area (customer focused, employee focussed, supplier focussed) to the entire enterprise
- Industry specific/domain specific modelling tools/templates can be developed/reused
- Forms the basis for building applications (based on OO technologies) and services

Figure 1: MDA
OMG also has conceptual data models defined for different industries that individual enterprises can adopt. There is also this notion of PIM (platform independent model) and PSM (platform specific model). The PIM maps PSM to application interfaces, codes, etc.

---

**Figure 2: PIM and PSM**

- PIM
  - CORBA Model
  - Java/EJB Model
  - XML/SOAP Model
  - Other Model
  - Map PSM to application interfaces, code, GUI descriptors, SQL queries, etc.

- PIM
  - Legacy App
  - COTS App
  - Reverse-engineer existing application into a model and re-deploy.

---

**Figure 3: MDA Tools and PIM**

- MDA Tools combine application and platform knowledge to generate bridges

**Source:** OMG
Introduction to ADDM/ATAM and PLA

This is a brief introduction to ADDM, ATAM and PLA from the Software Engineering Institute. For an in-depth understanding of these SEI initiatives a series of reference book titled "Software Architecture in Practice" by Lenn Bass, Paul Clements and Rick Kazman, followed by "Evaluating Software Architectures" by Paul Clements, Rick Kazman and Mark Klein and "Software Product Lines" by Paul Clements and Linda Northrop at http://www.sei.cmu.edu/publications/books/index.html will be a good start.

The Attribute Driven Design (ADD) method is a systematic approach to designing the architecture for a family of systems. It is based on knowing both the functional and quality requirements for your systems and knowing the architectural approaches that have proven successful in achieving those qualities in other systems. The ADD method is a recursive decomposition method. At each stage of the decomposition there are the following steps:

1. Choose architectural drivers. The architectural drivers are the combination of quality, business, and functional goals that "shape" the architecture.
2. Choose patterns and children component types to satisfy drivers. There are known patterns to achieve various qualities. Choose the solutions that are most appropriate for the high priority qualities.
3. Instantiate children design elements and allocate functionality from use cases using multiple views. The functionality to be achieved by the product family is allocated to the component types.
4. Identify commonalities across component instances. These commonalities are what define the product line, as opposed to individual products.
5. Validate that quality and functional requirements and any constraints have not been precluded from being satisfied by the decomposition.
6. Refine use cases and quality scenarios as constraints to children design elements. Because ADD is a decomposition method, the inputs for the next stage of decomposition must be prepared.

Note that Modeling (conceptual business model and data model) feed into the ADD method for (attribute) Requirements, Analysis & Design, Implementation, Test and Planning/evaluation, after which deployment (more around Service Execution, Service Management) kicks in.

Quality attributes of a large software system reside principally in the system's software architecture. In large systems, the achievement of qualities such as performance, availability, survivability, and modifiability depends more on the overall software architecture than on code-level practices such as language choice, detailed design, algorithms, data structures, and testing. It is cost effective to try to determine, before a system is built, whether it will satisfy its desired qualities. Although methods for analyzing architectures with respect to quality attributes exist, these analyses have typically been performed on specific qualities in isolation. However, the attributes (and hence their analyses) interact. Performance affects modifiability. Availability affects safety. Security affects performance. Everything affects cost. While experienced designers know that these trade-offs exist, there is no codified method for characterizing them and, in particular, for characterizing their interactions. More importantly, these tradeoffs present the areas of highest risk in an architecture.
SYNERGIZING METHODOLOGIES TO ACHIEVE ARCHITECTURE TARGETS

The SEI has developed architecture evaluation methods to analyze an architecture for these multiple quality attributes and their tradeoffs. The Architecture Tradeoff Analysis Method℠ (ATAM℠) is a risk mitigation method that can be done early in the software development life cycle when it is relatively inexpensive to change architectural decisions.

It is also important to be able to analyze the architecture of the countless legacy systems in existence. Unfortunately, the architectures of legacy systems are frequently undocumented or existing documentation is inaccurate due to the unavoidable architectural drift and erosion making analysis impossible. To address this phenomena, the SEI has developed Dali, a workbench for software architecture extraction and reconstruction, to reconstruct architectures from source code and to check the conformance of as-built systems to their documented architectures.

SEI work in the Architecture Tradeoff Analysis (ATA) Initiative includes development and validation of the technology and techniques necessary for analyzing software architectures, specifically: attribute-specific models, representation approaches, analysis methods, reconstruction and conformance tools and techniques, as well as arranged evaluations and reconstructions of architectures for customer systems. The primary goals of ATAM, are:

- Establish and transition validated techniques for analyzing the effect of software architectural decisions on selected product quality attributes.
- Establish and transition validated techniques for reconstructing the architecture of legacy systems and for determining the conformance of as-built systems to defined architectures.
- Establish and transition validated techniques for representing software architectures.
- Promote understanding of software architecture and architecture analysis.

Product line analysis (PLA) is an initial and relatively brief pass at requirements engineering for a product line of software-intensive systems. Its goal is to identify opportunities for large-grained reuse across the product line. PLA is the link between the recognition of a business opportunity and the design of a product line architecture. It incorporates the views of multiple product line stakeholders in a preliminary requirements model that includes the functional features of products and the software quality attributes (e.g., performance, modifiability) of both the products and their...
SYNERGIZING METHODOLOGIES TO ACHIEVE ARCHITECTURE TARGETS

development. The stakeholders providing the necessary input include marketers, managers, customer representatives, and architects. The requirements model created by PLA identifies common requirements across the product line and their allowed variants.

The primary benefit of PLA is its early identification of the major issues affecting a product line and its development (e.g., how much of the functionality and quality attributes could be implemented in core assets and how much will need to be done by product developers, and the technical and business consequences of the proposed reuse strategy). PLA is based on a systematic modeling process that incorporates the organization’s business goals and constraints, clarifies and refines assumptions about the product line scope, and provides early information about the technical feasibility of the product line. PLA also provides input to the decisions and tradeoffs concerning allocation of resources to core asset and product development. PLA helps managers identify risks in their approach to adopting software product lines. It also helps asset developers (including architects) who need usable AND useful representations of requirements, not a set of "shelfware" binders.

What the SEI now calls “product line analysis" grew out of its early domain-analysis and requirements-modeling work with industrial customers building software product lines. Because product lines span multiple domains and have multiple stakeholders, domain-analysis techniques alone did not provide a complete solution. As a consequence, the SEI approach uses techniques from domain analysis, but also borrows from work in object technology, use-case modeling, and architecture definition and evaluation.

A joint SEI-customer team used an early version of PLA to define the requirements for a product line of automotive software. The requirements model captured common requirements and variation points and allowed early exploration of the product line architecture to proceed in parallel with detailed requirements engineering.

![Figure 5: Product Line Architecture and Enterprise Line Architecture](image)

Because PLA is focused on large-grained reuse across a product line, it does not address all the requirements. It is an iterative, incremental process of eliciting, analyzing, specifying, and verifying the early requirements for a product line based on an initial business case and market analysis. The output of the process is a requirements model comprising four interrelated work products. The work products are based on object modeling, use-case modeling, and feature-modeling techniques:
• The use-case model specifies the product line stakeholders and their key interactions with the product line. These stakeholders will verify the acceptability of the product line (and of the requirements).

• The feature model specifies the stakeholders’ views of the product line. It captures the functional features of products and the software quality attributes of the product line and its products.

• The object model specifies the product line responsibilities that support those features and the commonality and variability of the responsibilities.

• The dictionary defines the terminology used in the work products and supports a consistent view of the product line requirements.

Together these work products form the basis of a systematic method for capturing and refining requirements for current products, future envisioned products, stakeholder needs and expectations, and associated rationales and tradeoffs. The work products are first populated with an initial set of use cases, features, responsibilities, and terminology, and then refined iteratively and incrementally. Use cases and features determine the elicitation of the product line requirements. Features and objects are analyzed for commonalities and variabilities, consistency, quality, interactions, and priority. The product line stakeholders verify the accuracy and completeness of the resultant requirements model. The modeling stops when opportunities for large-grained reuse can no longer be identified.

A software product line succeeds because the commonalities shared by the products can be exploited to achieve economies of production. PLA mitigates the risk of product line adoption by combining information from multiple product line stakeholders in a form that permits reasoning about the allocation of functional features and quality attributes to assets and products. The premise of PLA is that a sound initial understanding of the problem to be solved is essential before an organization embarks on a software product line as a solution. The model of a product line that emerges from PLA also provides early identification of the architecturally-significant requirements. The modeling process incorporates the organization’s business goals and constraints, clarifies and refines assumptions about the product line scope, and provides an early indication of the technical feasibility of the product line.

(Source: SEI)
Introduction to The Open Group’s AF and ADM

This is a brief introduction to TOGAF/ADM. For an in-depth understanding a good reference book titled “Enterprise IT Architecture” authored by Col Perks and Tony Beveridge and http://www.tog.org/togaf8 will be a good start (Note: This book talks to ADM as of version 7 when it was technology centric, since then The Open Group has enhanced ADM to version 8, 8.1 and 9 with an Enterprise focus).

The Open Group’s Architecture Framework and Architecture Development Methodology (ADM) emphasize a Common (Enterprise) System Approach to build Service Oriented Architecture. The notion if Foundation Services, Basic Services, Business Services, Portal Services, Architecture Building Blocks and Solution Building Blocks, and more exist in TOGAF. Key characteristics of TOGAF/ADM are:

- The Iron Age (60’s to early 90’s) - Mainframe – Separation of a purely software architectural approach and the architecture of the reminder of the infrastructure and environment.
- The Renaissance (70’s to 2000) - Client Server/UNIX – Architecture by Specifications – Bodies such as IEEE and ISO were the first to begin to describe an instance of a technology landscape
- The Industrial Revolution (80’s to now early 2000) - PC and the Internet explosion – Architecture by Products
- Galactic Enlightenment (03/04 and beyond maybe) - realization of “the network is the computer” - The future – “:basic infrastructure consolidation: networks, operating systems and software architectures will merge into a small number of key technologies all necessary to support “INTER-OPERABLE WEB APPLICATIONS”

Overview of The Open Group’s Architecture Framework/Architecture Development Methodology & its focus on Services Architecture

TOGAF + ADM + TRM + more helps with the build up of a Technical Architecture

It defines, describes and specifies the basic foundation services that are needed to build business services in an Enterprises Technical Architecture

It has a comprehensive set of Services Taxonomy (including System Services, Storage/management services, Security Services, etc.).

It acts as glue between the logical and the physical Architectures

Overview of The Open Group’s Architecture Framework/Architecture Development Methodology & its focus on Services Architecture

TOGAF + ADM + TRM + more helps with the build up of a Technical Architecture

It defines, describes and specifies the basic foundation services that are needed to build business services in an Enterprises Technical Architecture

It has a comprehensive set of Services Taxonomy (including System Services, Storage/management services, Security Services, etc.).

It acts as glue between the logical and the physical Architectures

ADM treats the EA development as a Continuum, Architecture Continuum followed by Solution Continuum.

Iterative process that starts with an Architecture Vision to Architecture Change Management.
SYNERGISING METHODOLOGIES TO ACHIEVE ARCHITECTURE TARGETS

Figure 6: ADM

Figure 7: Architecture and Solution Continuum
(Source: TOG)

13
Introduction to OGC’s ITIL

This is a brief introduction to OGC's ITIL/ITSM. For an in-depth understanding a good reference for books are at [http://www.itilsurvival.com/ITILbooksfree.html](http://www.itilsurvival.com/ITILbooksfree.html). The IT Infrastructure Library, ITIL (®), is a series of documents that are used to aid the implementation of a framework for IT Service Management (ITSM). This framework defines how Service Management is applied within specific organizations. Being a framework, it is completely customized for application within any type of business or organization that has a reliance on IT infrastructure. This directory is intended to serve as a start point for your ITIL/ITSM needs. Whether you are entirely new to the topic, or whether you are a seasoned practitioner, you should hopefully find something of value.

There are 2 areas within ITIL/ITSM under Service Delivery and Service Support;

Service Delivery is the management of the IT services themselves, and involves a number of management practices to ensure that IT services are provided as agreed between the Service Provider and the Customer.

Service Delivery consists of 5 disciplines. These are:

- Service Level Management
- Capacity Management
- Contingency Planning
- Availability Management
- IT Financial Management

Service Support is the practice of those disciplines that enable IT Services to be provided. Without these disciplines, it would be almost impossible to provide these IT Services, and at best in a very unmanaged and haphazard way.

The 6 Service Support disciplines are:

- Configuration Management
- Problem Management
- Incident Management
- Change Management
- Service / Help Desk
- Release Management
Key characteristics of ITIL/ITSM are:

- IT Infrastructure Library & ITSM focus on the Management of a deployed Architecture
- Post deployment management of operations (sustenance of the architecture)
- Clearly defined SLA's & OLA's
- Clearly defined disciplines – change management, problem management, incidence management, etc.
- Addresses the optimization of an architecture and infrastructure on a continuous basis.
- Management of the full life cycle of Services from creation to End-of Life.
Introduction to OGC's PRINCE2

This is a brief introduction to OGC's Prince2. For an in-depth understanding go to the official site at http://www.ogc.gov.uk/prince2/.

PRINCE, which stands for Projects in Controlled Environments, is a project management method covering the organization, management and control of projects. PRINCE was first developed by the Central Computer and Telecommunications Agency (CCTA) now part of the Office of Government Commerce (OGC) in 1989 as a UK Government standard for IT project management. Since its introduction, PRINCE has become widely used in both the public and private sectors and is now the UK's de facto standard for project management.

PRINCE2 is a process-based approach for project management providing an easily tailored, and scalable method for the management of all types of projects. Each process is defined with its key inputs and outputs together with the specific objectives to be achieved and activities to be carried out.

Figure 9: Prince 2 Process Model
SYNERGIZING METHODOLOGIES TO ACHIEVE ARCHITECTURE TARGETS

The method describes how a project is divided into manageable stages enabling efficient control of resources and regular progress monitoring throughout the project. The various roles and responsibilities for managing a project are fully described and are adaptable to suit the size and complexity of the project, and the skills of the organization. Project planning using PRINCE2 is product-based which means the project plans are focused on delivering results and are not simply about planning when the various activities on the project will be done. A PRINCE2 project is driven by the project's business case which describes the organization's justification, commitment and rationale for the deliverable or outcome. The business case is regularly reviewed during the project to ensure the business objectives, which often change during the life-cycle of the project, are still being met.

There are often different groups of people involved in projects: the customer, one or more suppliers, and of course the user. PRINCE2 is designed to provide a common language across all the interested parties involved in a project. Bringing customers and suppliers together typically involves contracts and contract management, although these aspects are outside the scope of PRINCE2, the method provides the necessary controls and breakpoints to work successfully within a contractual framework.

PRINCE2 is a structured method providing organisations with a standard approach to the management of projects. The method embodies proven and established best-practice in project management. It is widely recognised and understood, and so provides a common language for all participants in the project.

PRINCE2 provides benefits to the organisation, as well as the managers and directors of the project, through the controllable use of resources and the ability to manage business and project risk more effectively. PRINCE2 enables projects to have:

• A controlled and organized start, middle and end;
• Regular reviews of progress against plan and against the Business Case;
• Flexible decision points;
• Automatic management control of any deviations from the plan;
• The involvement of management and stakeholders at the right time and place during the project;
• Good communication channels between the project, project management, and the rest of the organisation.
• OGC’s Prince 2 as a methodology focuses on the implementation of a Physical/Deployment Architecture (and development efforts).
  • Prince 2 leverages the best practices around project/program management processes.
  • Addresses risk mitigation techniques
  • Logistics Issues
  • People and Resource concerns
  • Expectations Management
  • Ensures successful deployments for Large Scale multi-vendor deployment Architectures (including network, server and storage elements)
  • Leverages prior efforts and blends logical architecture with a real deployment architecture

(Source: OGC)
Alignment of Methodologies to Architectural Layers

Synergies = where $3 + 2 = 6$

The integration of these 3 Architecture development methodologies (ADM, ADDM, MDA) after certain refinements and shakeouts can lead to building more “Agile Architectures” that can address the one constant “Change”. Change in all dimensions, business conditions, environmental conditions, technology advances, etc., that needs to addressed by SOA.

Change - the ONE CONSTANT

Figure 10: Alignment that addresses Change
SYNERGIZING METHODOLOGIES TO ACHIEVE ARCHITECTURE TARGETS

As noted earlier:

- **OMG’s MDA** can be viewed as a *meta-data/model driven* approach to build a highly integrated Enterprise *Information/Data* Architecture, and more.

- **ADDM/ATAM/PLA** can be viewed as a *attribute driven iterative* approach to build a highly integrated Enterprise *Application/Process* Architecture, and more.

- **TOGAF/ADM** can be viewed as a *service driven approach* to build a highly integrated Enterprise *Technology/Infrastructure* Architecture, and more.

Although the main emphasis of each methodology is highlighted here, it must be understood that there is more to each one of them. For example, OMG’s MDA is not only about Enterprise Information and Data integration, its an approach that allows for the ongoing maintenance of an Enterprise’s System Architecture – by modeling all the systems in an enterprise, both PIM (Platform Independent Models) and PSM (Platform Specific Models). Similarly, ADM takes into account the implementation and management of Services as they get deployed.

The capability to keep and maintain the alignment between these architectures is achieved when the core principles behind “Agile Architectures” such as “Modifiability”, “Portability”, “Integrate-ability”, “Reusability”, “Replace-ability”, “Implement-ability”, “Extensibility” and more, forms the basis for building Enterprise Architectures.

“**Modifiability**” – making changes without disruption -like not having to restart a JVM when new code is applied.

“**Portability**” – a fundamental capability is portability not just between hardware platforms and operating systems; but also, between multiple Application Server (Containers), and more.

“**Integrate-ability**” - open and published API based integration.

“**Reusability**” - leveraging a set of logic across multiple applications and systems.

“**Replace-ability**” - minimal effort required to replace one implementation from another.

“**Implement-ability**” - dynamic code generation and model driven design

“**Extensibility**” - refractor-ing capabilities and extensions made possible

“**Flexibility**” - accommodating changes in requirements through disintermediation, i.e, separation of concerns as a service.
Architectural Organization that addresses Architectural Drifts vs Architectural Conformance (Governance)

From a more practical basis this would require an Organizational structure, where; there are clear and well aligned roles within an Enterprise Architecture and Standards Technology (EAST) group where the members constantly communicate and collaborate with one another.

The idea behind this type of an Organization or a Group is to have a team of highly experienced and skilled Architects working in synch to achieve common Business Goals. Each individual adds value to the team with their respective expertise (such as Data Architecture and Data Integration Methodologies) and the team combined with the stewardship/leadership provided by the Enterprise Architect and the Chief Architect produces synergies that result in a cohesive, streamlined Architecture. The focus of the EA is more on the integration of Data, Process and Technology Architectures and the focus of the CA is to leverage best practices associated with not only Architectures but also the Implementation and Management (AIM); program management/governance aspects of the Architecture.

The “Architectural” when developed with this Integrated Methodology makes it a SMART “Strategically Maneuverable Architectural Target”.

Figure 11: Organizing Architects
TOGAF at the Center that lends itself to MDA/ADDM and ITIL/PRINCE2

TOGAF/ADM has its root in Technology Architectures however has evolved into a methodology and frameworks for Enterprise Architectures. There are linkages between TOGAF and ADDM/ATAM and PLA that are covered extensively by the Open Group. ADDM and ATAM kick in as a design methodology in the Phase C where ADM defines Information Systems Architecture. Phases A through D are in direct alignment with MDA and ADDM. Phase E through H are in direct alignment with ITIL and Prince2 and has there appropriate references made by TOG. ITIL and PRINCE2 coming from the same parent Organization OGC where ITIL addresses the Service Delivery and Service Support Models of Managed Services and PRINCE2 as a project management methodology for implementing these service oriented architectures from definition stage, to the design and deployment stages. This combined set of methodologies address the end to end life-cycle associated with Services.

- Service Modeling
- Service Creation
- Service Interaction
- Service Integration
- Service Calibration
- Service Execution
- Service Level Management
- Service Support
- Service Delivery
- Service Capacity Management
- Service Contingency Planning

- Service Availability Management
- Service Configuration Management
- Service Problem Management
- Service Incident Management
- Service Change Management
- Service / Help Desk
- Service Release Management
- Service Broker-ing
- Service Packaging
- Service Rendering
- Service Management
- Service Billing
Common (Enterprise) System Architecture across industry domains:

Across industries Enterprises are embracing common enterprise level system architecture for all customer, employee and partner facing services. From Delta to United Health Care to Sprint, this macro architecture pattern is prevalent. It is generally made up vendors implementation of these solutions such as, with a common/central messaging bus (near real time that can support events as well), a common Business Logic tier, a Common Security Tier Unified Identity Services, and so on.

Figure 14: Common Systems Architecture supporting SOA

In order to ensure that all the synergistic values of this integrated Meta methodology can be achieved, there has to be clearly defined handoff (interfaces) between these Architects through the use of a workflow of tools, techniques and templates.
## Mapping the Tools, Templates & Techniques

<table>
<thead>
<tr>
<th><strong>Data, Data, Information Architecture</strong></th>
<th><strong>Business Process Architecture</strong></th>
<th><strong>Application, System Architecture</strong></th>
<th><strong>Integration Messaging Architecture</strong></th>
<th><strong>Network Physical Architecture</strong></th>
<th><strong>Operations Management Architecture</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>ERD</td>
<td>MDA</td>
<td>TOGAF</td>
<td>TOGAF</td>
<td>Converged Network Architectures</td>
<td>ITIL</td>
</tr>
<tr>
<td>Generic Industry Conceptual Data Models</td>
<td>Process Flow Diagrams</td>
<td>TOGAF</td>
<td>Reference Architectures</td>
<td>SDNA Patterns</td>
<td>Prince 2</td>
</tr>
<tr>
<td>Object-Data Mapping</td>
<td>Domain specific Process Models</td>
<td>Reference Architectures</td>
<td>Integrated Message Bus (ETL, EAI, etc.)</td>
<td></td>
<td>SM Best Practices</td>
</tr>
<tr>
<td>Domain specific Data Models</td>
<td>UML</td>
<td>IIIRM</td>
<td>IIIRM</td>
<td></td>
<td>CMDB</td>
</tr>
<tr>
<td>MOF/XMI</td>
<td></td>
<td>SIB</td>
<td>SBB</td>
<td></td>
<td>SLA</td>
</tr>
<tr>
<td>CWM</td>
<td></td>
<td>ADML</td>
<td>ADML</td>
<td></td>
<td>OLA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Data Integration Patterns</strong></th>
<th><strong>Common Data Model (service specific)</strong></th>
<th><strong>MOF</strong></th>
<th><strong>Service Building Blocks</strong></th>
<th><strong>TRM</strong></th>
<th><strong>IIIRM</strong></th>
<th><strong>Container based Techniques (system)</strong></th>
<th><strong>Connector Building Blocks</strong></th>
<th><strong>Network Modeling</strong></th>
<th><strong>Network Abstractions</strong></th>
<th><strong>Network demarcation/zoning</strong></th>
<th><strong>Configuration Data Repository</strong></th>
<th><strong>Patch Database</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instrumenting Abstraction</td>
<td>BPEL</td>
<td>Container based Techniques (component)</td>
<td>Container based Techniques (system)</td>
<td>Connector Building Blocks</td>
<td>Network Modeling</td>
<td>Network Abstractions</td>
<td>Network demarcation/zoning</td>
<td>Configuration Data Repository</td>
<td>Patch Database</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model driven Patterns</td>
<td>Integrated Development SOA Patterns</td>
<td>Messaging Patterns</td>
<td>Event Patterns</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model driven Patterns</td>
<td>Integrated Development SOA Patterns</td>
<td>Messaging Patterns</td>
<td>Event Patterns</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model driven Patterns</td>
<td>Integrated Development SOA Patterns</td>
<td>Messaging Patterns</td>
<td>Event Patterns</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model driven Patterns</td>
<td>Integrated Development SOA Patterns</td>
<td>Messaging Patterns</td>
<td>Event Patterns</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model driven Patterns</td>
<td>Integrated Development SOA Patterns</td>
<td>Messaging Patterns</td>
<td>Event Patterns</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model driven Patterns</td>
<td>Integrated Development SOA Patterns</td>
<td>Messaging Patterns</td>
<td>Event Patterns</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conclusion

IT Architect’s role is to address “trade-offs” between systemic qualities or what is refereed to as “Architectural Principles” to ensure that the system or enterprise system that one is building is scalable, secure, available, etc., through there focus on Alignment. Perceptions in tradeoffs such as high security results in low scalability, etc., are only perceived trade-offs and there are many techniques, tactic and tools that one can use to ensure such trade-offs are addressed. However from project/program Management perspective the trade-offs associated with cost, quality and time – is more than perception and need constant tuning. Focusing more on the business viability of a project. If we can complete a project within a given time frame under a given budget with the required set of “systemic qualities” – then afford-ability as a trade-off is addressed. In this case the economic principle associated with economies of scale kicks in for Business projects. If the project is built on stable technologies, experienced staff, and the optimal set of resources- that lend themselves to Alignment -one can attempt to address the trade-offs associated with “viability”. 
About the Author

Rakesh Radhakrishnan is a Lead IT Architect in the Communications Market Area of Sun Client Solutions. He has covered Telecom Companies, Network Equipment Providers (NEP) and Service Provider accounts in Europe, Canada, USA and Latin America.

He has over 15 years of experience and has an MBA (MIS) and MS (MIT). He is an active member of Customer Engineering Technology Council (CETC) and was the Chairman of a Working Group on Container Alignment Engine (CAE patent received from Europe and US) at Sun. He has published more than 15 papers on IT Architectures (Frameworks, Process and Techniques) and is a frequent speaker in conferences including OMG, TOG, CMG, IRM, SuperG, SunNetwork, Java ONE, etc. He has led multiple Architecture Workshops and Architecture Assessments for IT Consolidation and Network Identity projects. He was recently featured on Officer Outlook for his work on Aligning Architectural Approaches (Sun’s WS-Incite Award for 2005).

Rakesh is also Certified by TOG (on TOGAF 8), SEI (as a SW Architect) and OGC (Prince 2 and ITIL). He has Green Belt Six Sigma training and is a ECCSE (Enterprise Computing Certified Systems Engineer-Competency 2000) and as a Systems Architect Pro (from Peoplesoft).