Towards a Holistic Architecture Platform

Tony C Shan[1], and Winnie W Hua[2]

[1] Bank of America, 200 N College St,
Charlotte, North Carolina 28255, USA
[2] CTS Inc, 10712 Hellebore Rd,
Charlotte, North Carolina 28213, USA
{tonycshan, winniehua}@yahoo.com

Abstract. This paper defines a three-dimensional architectural framework, named Technology and Information Platform (TIP), to effectively handle the architecture complexity and manage the architectural assets of enterprise information systems in a service-oriented paradigm. This comprehensive model is composed of a Generic Architecture Stack (GAS) comprising a stack of architecture layers, and the contextual spectrums consisting of the Process, Abstraction, Latitude, and Maturity (PALM) dimensions. The GAS stack contains seven interrelated layers: Enterprise Business, Enterprise Technical, Cross Business-line, Channel Specific, Application Solution, Aspect-Oriented, and Component Technology Architectures. A concept of Meso-Architecture is proposed in this work to facilitate the service- and channel-level architecture modeling in a service-oriented computing style. The key practitioners responsible for these architectural models in the platform are also specified in the context. Part of this pyramid blueprint has been extensively utilized in one form or another to design various IT solutions in different industries such as finance, telecommunications, and government.

Keywords: Architecture, framework, pattern, model, infrastructure, application, aspect, component, technique, business process, solution, domain, stack, reference model, platform, layer, view, practitioner, and perspective.

1 Introduction

As business operations continue growing to face the global competition, the information technology (IT) division in an organization must adapt and perform to keep pace with the business expansion. The success of the eCommerce business relies on higher levels of IT services at a lower cost. It becomes compulsory for the information systems, though becoming more complex, to be even more scalable, reliable, flexible, extensible, and maintainable. IT must innovate to produce forward-thinking technical solutions, to meet the constantly-changing business needs.

Through either organic growth or mergers/acquisitions in the past years, large organizations typically possess thousands of information systems and applications using diversified architectures and technologies, which provide external clients and internal employees with services and products to satisfy a wide variety of functional

requirements from different lines of business. In the financial institutions, for example, the business process generally contains different business sectors in consumer, commercial, small business, wealth management, investment banking, and capital market. The service delivery channels range from traditional brick-and-mortar branches, call centers, and Automated Teller Machines (ATMs), to online web browsers, interactive voice response, emails, mobile devices, and so on. A highly structured solution is of vital importance to abstract concerns, divide responsibilities, encapsulate the complexity, and manage the IT assets in such a diversified environment.

2 Challenges of Architecture Complexity

There have been a plethora of previous studies in the last few decades to address the issue of architecture complexity, which has grown exponentially as the computing paradigm has evolved from the monolithic to a service-oriented architecture. Zachman [1] created a pioneering framework in the form of a two-dimensional matrix to classify and organize the descriptive representations of an enterprise IT environment. These representations are significant to the organization management and the development of the enterprise's information systems. As a planning or problem-solving tool, the framework structure has achieved a level of penetration in the domain of business and IT architecture/modeling. However, it tends to implicitly align with the data-driven approach and process-decomposition methods, and it operates above and across individual project level. In a similar approach and format but more technology-oriented, Extended Enterprise Architecture Framework (E2AF) [2] contains business, information, system, and infrastructure in a 2-D matrix. Both these two approaches are heavyweight methodologies, which require a fairly steep learning curve to adopt.

In an attempt to overcome the shortcomings in the above two methods, Rational Unified Process (RUP) [3] take a different route by applying the Unified Modeling Language (UML) in a use-case driven, object-oriented and component-based approach. The overall system structure is viewed from multiple perspectives – the concept of 4+1 views. RUP is process-oriented to a large extent, and is generally a waterfall approach in its original root. The software maintenance and operations are inadequately addressed in RUP, which also lacks a broad coverage on physical topology and development/testing tools. It mainly operates at the individual project level. RUP has been recently extended to Enterprise Unified Process (EUP) and Open Unified Process (OpenUP) in open source form.

The Open Group Architectural Framework (TOGAF) [4], as another heavyweight approach, is a detailed framework with a set of supporting tools for developing enterprise architecture to meet the business and information technology needs of an organization. The three core parts of TOGAF are Architecture Development Method (ADM), Enterprise Architecture Continuum, and TOGAF Resource Base. The scope of TOGAF includes Business Process Architecture, Applications Architecture, Data Architecture, and Technology Architecture. The focal point of TOGAF is not at the level of individual application architecture, but enterprise architecture. On the other

hand, Model-Driven Architecture (MDA) [5] takes a different approach, with an aim to separate business logic or application logic from the underlying platform technology. The core of MDA is the Platform-Independent Model (PIM) and Platform-Specific Model (PSM), which provide greater portability and interoperability as well as enhanced productivity and maintenance. MDA is primarily intended for the architecture modeling part in the development lifecycle process.

Other related work on IT architecture frameworks is largely tailored to particular domains. They can be used as valuable references when an organization plans to create its own model. There are three prominent frameworks developed in the public services sector. The comprehensive architectural guidance is documented in C4ISR Architecture Framework [6], for the various Commands, Services, and Agencies within the U.S. Department of Defense, in order to ensure interoperable and cost effective military systems. A counterpart in the Treasury Department is the Treasury Enterprise Architecture Framework (TEAF) [7], which is intended to guide the planning and development of enterprise architectures in all bureaus and offices within that division. The Federal Enterprise Architecture (FEA) framework [8] provides direction and guidance to U.S. federal agencies for structuring enterprise architecture.

The Purdue Enterprise Reference Architecture (PERA) [9] is aligned to computer integrated manufacturing. ISO/IEC 14252 (a.k.a. IEEE Standard 1003.0) is an architectural framework built on POSIX open systems standards. The ISO Reference Model for Open Distributed Processing (RM-ODP) [10] is a coordinating framework for the standardization of Open Distributed Processing in heterogeneous environments. It uses "viewpoints" and eight "transparencies" to describe an architecture that integrates the support of distribution, interworking and portability. The Solution Architecture for N-Tier Applications (SANTA) [11] defines a service-oriented solution model comprising a stack of six interrelated layers, coupled with six vertical pillars. A comprehensive mechanism is presented in the Solution Architecting Mechanism (SAM) [12], composed of eight interconnected modules for architecture design. The Service-Oriented Solution Framework (SOSF) [13] describes a pragmatic approach designed for Internet banking in financial services, utilizing service patterns, architecture process, hybrid methodology, service model, and solution platform.

A new model is proposed in the next section, with more detailed descriptions of the key artifacts and features of the generic architecture stack in Section 4. Section 5 specifies the contextual spectrums and a particular aspect in one of the four dimensions – practitioners who are responsible for each architecture layer, followed by the conclusions section.

3 Comprehensive Approach

As discussed in the foregoing section, virtually all previous investigations revealed the architectural aspects of an information system to some extent from single or limited perspectives. The necessity of a comprehensive solution to describe the end-to-end IT solution and portfolio architecture becomes more and more evident, demanding a systematic and disciplined approach. A highly structured framework is thus designed in this paper to meet this ever-growing need, and present a

comprehensive and holistic model covering the prominent architectural elements, components, knowledge, and their interrelationships. Operation processes can be established accordingly based on this model to facilitate the creation, organization, and management of the architecture assets at different levels in a large firm.

3.1 Design Philosophy

The design principles that are applied to develop the overarching model are as follows:

- A model should have flexibility to be not only adaptive but also proactive.
- A model should provide multi-perspective views of all architecture artifacts.
- A model should be independent of specific technology choices and therefore can operate on a variety of technology platforms.
- A model should be based on an open structure, following the industry best practices.
- A model should be dynamic and allow users to visualize details on demand while retaining the overview.
- A model should enable users to define the correlations between the artifacts, and provide an easy navigation to identify dependencies.
- A model should leverage the maximum support from the existing architecture standards and tools.
- The domain layering technique should be considered.
- A layer or spectrum should be created where a different level of abstraction is needed.
- Each layer should perform a well-defined function, and focus on a particular scope.
- The function of each layer should be chosen with an eye toward incorporating industry standards.
- The layer boundaries should be chosen to minimize the information exchange across the interfaces.
- The number of layers should be large enough that distinct functions need not be thrown together in the same layer out of necessity, and small enough that the architecture does not become unwieldy.
- The layers are loosely coupled.
- The layers are service-oriented, leveraging software patterns and frameworks.
- A layer should only know and interact with the neighboring layers.
- The contextual spectrum should cover a broad range of artifacts in each layer, and group them in appropriate categories.

3.2 Conceptual Model

The Technology and Information Platform (TIP) model is designed in this work as a systematic solution. It employs a divide-and-conquer strategy to abstract concerns, separate responsibilities and encapsulate complexity from one level to another. The

TIP model is a comprehensive framework to organize and visualize the architectural artifacts, and further help analyze and optimize the strategy, resources, process, systems, and applications. *TIP* comprises a Generic Architecture Stack (GAS) and contextual spectrums. Figure 1 shows a graphical representation of the platform in a pyramid shape. *GAS* is organized as a series of layers, each one built upon its predecessor, as illustrated in the vertical direction in the diagram. Every layer has a contextual spectrum, which consists of Process, Abstraction, Latitude, and Maturity (PALM) dimensions, as shown on the four sides of the pyramid bottom in Figure 1.

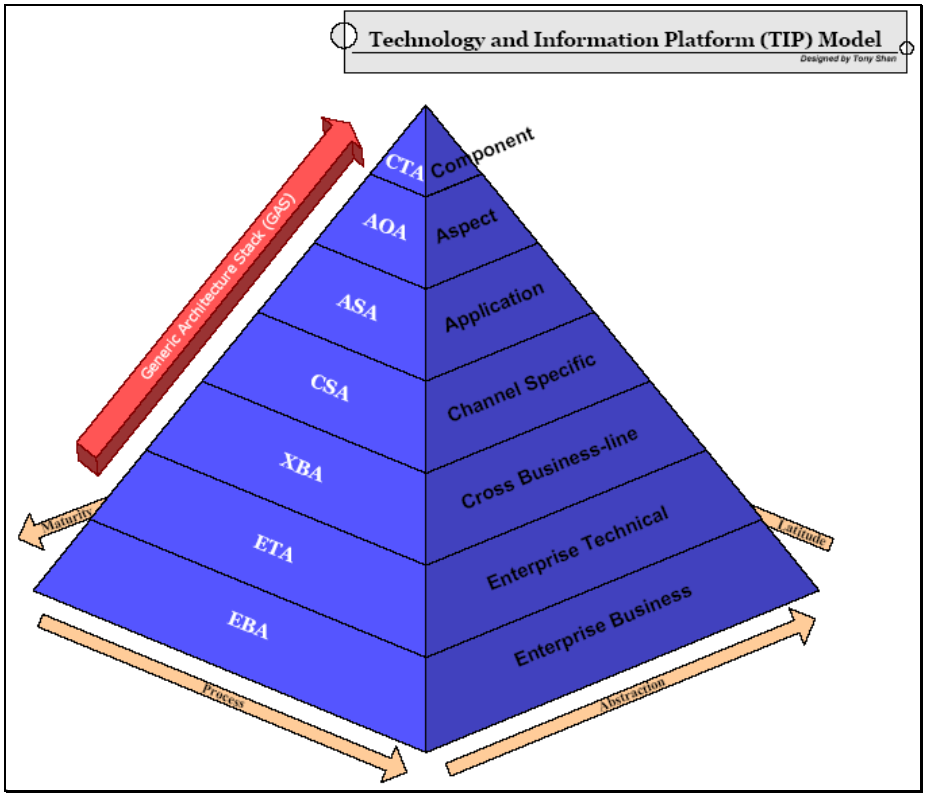

Fig. 1. TIP Pyramid Model

The *TIP* model provides multi-perspective views of the architecture assets in a large organization from both business and technical standpoints. The contextual spectrum is depicted in Figure 2, which contains four core parts: Process, Abstraction, Latitude, and Maturity (PALM). The *Process* dimension covers operations, risk, financial, resources, estimation, planning, execution, policies, governance, compliance, organizational politics, and so forth. The *Abstraction* dimension deals with what, why, who, where, when, which and how (6W+1H). The *Latitude* dimension includes principles, functional, logical, physical, interface, integration & interoperability, access & delivery, security, quality of services, patterns, standards, tools, skills, and so forth. Finally the *Maturity* dimension is about performance,

metrics, competitive assessment, scorecards, capacity maturity, benchmarks, service management, productivity, gap analysis, transition, etc.

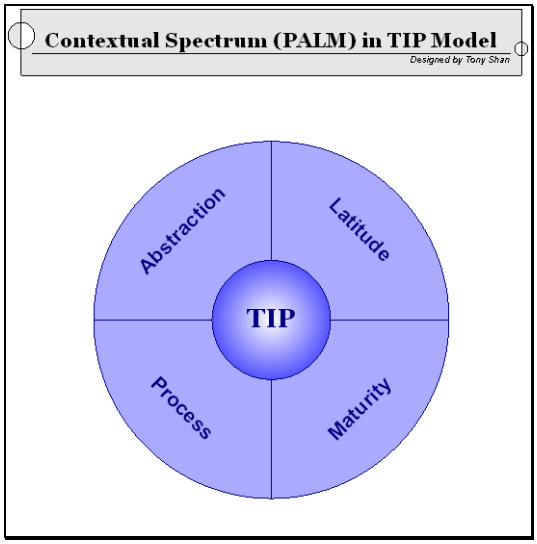

Fig. 2. Contextual Spectrum in TIP Model

Even though it is primarily targeted towards traditional online transaction processing (OLTP) systems by design, this model is extensible to be utilized in other areas such as enterprise resource planning (ERP) and analytics (business intelligence), with minor modifications or expansions.

4 Generic Architecture Stack

Various architectures have been used to describe the application structure in the design practices, such as data architecture, network architecture and security architecture. The need for a stack of multiple architectures within the enterprise is evidently indispensable, as the stack represents progressions from logical to physical, horizontal to vertical, generalized to specific, and an overall taxonomy. The architecture stack in the *TIP* model provides a consistent way to define and understand the generic rules, representations, and relationships in an information system portfolio. It represents categorization for classifying architecture assets – an aid to organizing reusable solution assets. It assists communications and understanding, within enterprises, between enterprise partners, and with vendor organizations. It is not uncommon that IT professionals talk at cross-purposes when discussing architecture issues because they are referencing different points in the architecture stack at the same time without realizing it. The stack helps avoid unnecessary misunderstandings and miscommunications.

The Generic Architecture Stack (GAS) in the *TIP* model comprises seven interrelated layers:

- Layer 1 – Enterprise Business Architecture.
- Layer 2 – Enterprise Technical Architecture.
- Layer 3 – Cross Business-line Architecture.
- Layer 4 – Channel Specific Architecture.
- Layer 5 – Application Solution Architecture.
- Layer 6 – Aspect-Oriented Architecture.
- Layer 7 – Component Technology Architecture.

The definitions and features of each layer will be articulated in the following sections.

4.1 Enterprise Business Architecture

The bottom layer in *GAS* is Enterprise Business Architecture (EBA), which deals with the goodness-of-fit between information systems and the business operations they are meant to facilitate. EBA is the business driver to all other technical models in the stack, forming the foundation of the strategic alignment of technical models with the business process mission. Driven by the business vision and strategy, EBA includes business operation model, process analysis and, where appropriate and feasible, business process re-engineering. The goals are common solutions for business process needs shared by multiple entities within the organization, development of business service models and components that can be reused across multiple applications, and increase of the efficiency of enterprise business processes. Business patterns are generally identified to group processes into different categories based on common ontology and taxonomy in the business domain.

4.2 Enterprise Technical Architecture

The layer next to the bottom is Enterprise Technical Architecture (ETA), which serves as the technical foundation to all enterprise applications. It deals with the overall architecture and infrastructure at a high level across the enterprise. ETA provides firms with methods, processes, governance, disciplines, and structure to create, organize, and use architecture-based assets, policies, strategies, and techniques. A ratification process is usually imposed in the governance. It generally includes four perspectives: business, application, information, and technology. The interrelated core architectures making up the ETA are the infrastructure architecture, system architecture, integration architecture and information architecture. The primary elements in ETA are guiding principles, architecture models, architecture frameworks, architecture patterns, technology policies, technology standards, and product/tool standards. The core architectures comprise a number of key components: business process, system development, shared services, middleware, integration, interoperability, technology patterns/frameworks, data access, data management, data design/modeling, system management/deployment, network, information security, and platform.

4.3 Cross Business-line Architecture

The next layer in the stack is Cross Business-line Architecture (XBA), which accounts for the core and composite business functionalities sharable across lines of business. XBA describes a business-line-agnostic architecture that can be leveraged by multiple business-delivery applications to improve the complete customer experience and reduce overall expenses. The architecture also addresses the cross-channel concerns if a business unit delivers services through multiple channels like Internet, voice, Personal Digital Assistant (PDA) and mobile devices. It defines service patterns, state data, service layers, and deployment models. Core business services and common functional services are constructed as basic services. Advanced feature-enriched services are built as composite shared services, consumed by different business units.

XBA becomes increasingly important in the service-oriented computing paradigm. The business services and corresponding IT implementations must be carefully identified and specified in a top-down approach. A service repository should be established to document the available services defined in this architecture, in order to maximize the reuse of the services across the lines of business, domains and channels. Service attributes and applicable policies are also captured and stored in a semantic fashion. Guidelines and patterns are created as well.

4.4 Channel Specific Architecture

Channel Specific Architecture (CSA) lies on top of XBA, which addresses the cross-application concerns and operational quality of services in a particular channel or line of business. A typical implementation is a common portfolio baseline to deal with the universal architectural concerns in an application set. The key architecture points addressed are the application dependency, interaction patterns, integration methods, cross-portfolio data management, service reusability, cross-application monitoring and management, single sign-on (SSO), unified authorization, cross-channel session management, and other infrastructural services. In addition, an architecture template is defined to specify the solution patterns for various system attributes such as load balancing, scalability, high availability, disaster recovery, capacity, storage, security, reliability, performance, collaborations, traceability, and deployment.

4.5 Application Solution Architecture

The fifth layer is Application Solution Architecture (ASA), which copes with the system architecture for individual applications. It covers the overall solution architecture, realization of business functionalities, process orchestration, workflow, rule management, business logic implementations, user interface, logical layering, service access interfaces, interaction mechanisms, multi-tier physical topology, networking for distributed solutions, storage management, product and technology

selections, etc. ASA is generally project-based at the system level and is aimed at a specific solution domain.

To make the software portion of a solution more flexible and adaptive, the inversion of control is often applied in ASA. The dependency injection can be realized declaratively via annotations or deployment descriptors, to minimize the coupling between the application components and the underlying implementation technologies. In addition, application architecture patterns and models are leveraged to design and build SOA applications. For example, Service Component Architecture (SCA) [14] describes a model for building applications and systems using a SOA style. SCA extends and complements prior approaches to implementing and assembling services, and SCA builds on open standards such as web services.

4.6 Aspect-Oriented Architecture

Aspect-Oriented Architecture (AOA) is the sixth layer, which deals with various application-wise aspects, largely software-related. It includes module-level frameworks such as Model-View-Controller (MVC) pattern-based structures, programming models such as Object-Oriented design (OOD), development tools such as Integrated Development Environment (IDE) workbenches, and automated unit testing such as JUnit and NUnit. Additionally, it deals with the classic crosscutting concerns via Aspect-Oriented Programming (AOP), like exception handling, logging, transactions, caching, data validation, session and state management, threading, synchronization, and remote access.

4.7 Component Technology Architecture

At the top of the pyramid is Component Technology Architecture (CTA), which handles the component-level internal structures and specialized technologies for specific technical concerns. These solutions can be in the format of packages, utilities, libraries, techniques, patterns, and implementation styles. Examples include Object-Relational (OR) mapping for data persistence, data access services, presentation-rendering mechanisms like XSL and template engines, page flow navigation, UI Look & Feel, XML parsing, service aggregation, Ajax, REST, and Gang-of-Four design patterns.

4.8 Interrelationships of the Layers

The layers in the *GAS* stack reveal the architecture artifacts gradually at the macro, meso, and micro level.

- **Macro-Architecture**: “global” vision – the overall structure in an enterprise (Layer 1 and 2)
- **Meso-Architecture**: “division” vision – the service and channel level properties and interactions across the application portfolios and domains (Layer 3 and 4)

- **Micro-Architecture**: "local" vision – the system attributes, relationships between components and component composition at the individual project and application level (Layer 5, 6, and 7)

The concept of Meso-Architecture defined in this paper has rarely received sufficient attention in the IT solution design in past practices. With the primary focus being only on the macro and micro designs, variants in one format or another of the Meso-Architecture might be scarcely crafted randomly, but then left in the dust. A lot of IT shops have not even recognized the significance of this artifact in their blueprints, let alone any formal design or patterns about it. However, the Meso-Architecture is a critical continuum between the macro-level and the micro-level concerns. The gap is bridged by the Meso-Architecture in terms of disciplined specification and validation of static structure and dynamic behavior of IT solutions at the service and channel levels. This part is becoming increasingly important in the lifecycle process of IT asset management and optimization. It is also critical to employ a hybrid methodology that combines both the top-down and bottom-up approaches in defining the service- and channel-level models to transform the existing IT portfolio into a service-oriented computing paradigm.

Each layer in the *GAS* is focused on particular technical and business domains and the granularity grows progressively from the bottom up to become more application-specific and technology-oriented. The upper layers leverage the services and solutions built in the lower layers. The lower layers are not tied with any upper layers, but they contain common architectural disciplines and shareable artifacts for the upper layers. The architectural rules are enforced so that the lower levels do not "call" the upper layers. The relationships between the layers are very loosely coupled, which makes this model adaptive and expandable. Each layer is self-encapsulated, and strictly adheres to the interfaces designed. The technologies and platforms that are used in one layer can be easily swapped, without affecting other adjacent layers. The architectures in the upper layers may augment or aggregate the customized implementations of the functionalities in the lower layers and incorporate other modular extensions for particular business domains.

5 Contextual Spectrum

The *TIP* model presents a holistic framework to describe the key artifacts in an IT environment from a variety of viewpoints. Figure 3 illustrates a top-down view from the tip of the pyramid model, which shows the multiple layers in the architecture stack as well as the major attributes in the four dimensions of the contextual spectrum. To exemplify the key characteristics of the attributes in these dimensions, we will concentrate on the *Who* attribute in the *Abstraction* dimension, and discuss the primary practitioners across the architectural layers in the *GAS* stack.

As each layer is focused on different architectural concerns and artifacts, it is natural that distinctive domain knowledge and practices as well as skillsets/tools are needed to design the architecture models at various levels. The key technical stakeholders who are responsible for each layer in *GAS* are listed as follows:

- **EBA** – Strategy Architect, Business Architect, Governance Architect, Information Architect, and Enterprise Architect.
- **ETA** – Enterprise Architect, Infrastructure Architect, Information Architect, Security Architect, Network Architect, Storage Architect, Governance Architect, and Data Architect.
- **XBA** – Enterprise Architect, Infrastructure Architect, Security Architect, Network Architect, Storage Architect, Domain Architect, and Data Architect
- **CSA** – Enterprise Architect, Solutions architect, Infrastructure Architect, Security Architect, Network Architect, Storage Architect, Channel Architect, Information Architect, Systems Architect, and Data Architect.
- **ASA** – Solutions architect, Systems Architect, Application Architect, Infrastructure Architect, Network Architect, Information Architect, Portfolio Architect, and Data Architect.
- **AOA** – Software Architect, Solutions architect, Application Architect, Information Architect, and Data Architect.
- **CTA** – Technology Architect, Component Architect, and Software Architect.

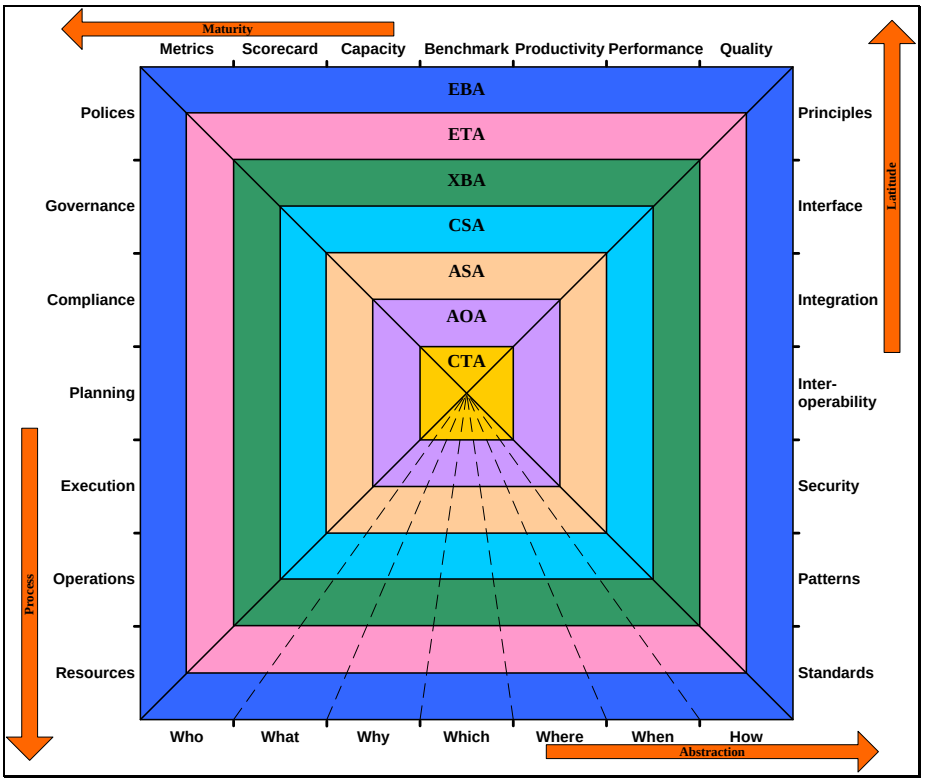

Fig. 3. Key Aspects in Contextual Spectrum

Different architects play distinct roles in the architectures at each layer. In practice, appropriate practitioners should be engaged in the architecting process to plan, analyze, specify, evaluate, validate, optimize and manage the models in the stack. Incorrect or insufficient staffing of qualified architects possessing the right skillsets would impose great risks in the architecture design, which most likely would lead to project setbacks later in the development lifecycle. Collaborations between the architects are critically important in large-scale system and infrastructure developments, particularly on the relationship, integration, and interoperations of different models in the architecture stack.

Table 1 summarizes the major features and functions of the *GAS* in the *TIP* framework, along with the practitioners and practices/patterns.

In contrast with existing frameworks as reviewed in Section 2, our model is more coherent and rational, covering a wide range of complex aspects represented in a three dimensional fashion. The logical grouping via a stack helps separate concerns and more accurately define roles and responsibilities in the architecting practices. Moreover, the aspect-oriented architecture and component technology architecture in this model reformulate the scope and emphasis of the traditional application solution architecture, expanding the breadth and depth of what architecture covers in the service-oriented design paradigm. This promotes the design-by-contract principle to another level, and facilitates the decision making and objective tradeoff justifications in solution design. Another key contribution in this framework is the Meso-architecture, composed of cross business-line architecture and channel-specific architecture, which lays out the crucial foundation for service-oriented engineering and portfolio rationalization.

Due to space constraints, other artifacts in the contextual spectrums of *Process*, *Abstraction*, *Latitude*, and *Maturity* (*PALM*) in each layer are articulated in a separate publication [15]. Additionally, a reference model has been developed to demonstrate the application of the key aspects and capabilities of the *TIP* framework in a financial institution scenario, which is to be presented in another paper.

6 Conclusions

To effectively manage the architecture complexity and organize diverse architectural assets in large organizations, a comprehensive solution is a necessity to abstract concerns, define responsibilities, and present a holistic view of the architectural aspects in a highly structured way. The Technology and Information Platform (TIP) model is designed as a multi-layered framework to facilitate architecting information systems. It provides comprehensive perspectives of the architecture designs from both business and technical standpoints. It builds concrete architecture solutions focused on different domains and portfolios, and in the meantime keeps the agility, flexibility and adaptiveness of the overall model.

Table 1. Feature summary of GAS in TIP model

Layer	Name	Features	Practitioners	Practices/Patterns
1. EBA	Enterprise Business Architecture	• High-level enterprise-wide • Business-oriented • Business process analysis and design • Business logic models and components • Business analysis patterns	- Business Architect - Strategy Architect - Governance Architect - Enterprise Architect - Information Architect	▪ Business operations model ▪ Business process architecture framework ▪ Zachman Framework ▪ Industry models (e.g. ACORD, IFX, eTOM, IFW)
2. ETA	Enterprise Technical Architecture	• High-level technology-oriented • Policies & governance • Corporate standards & strategies • Infrastructure, system, integration and data • Business, application, information, and technology	- Enterprise Architect - Infrastructure Architect - Information Architect - Security Architect - Network Architect - Storage Architect - Governance Architect	▪ Zachman Framework ▪ MSA blueprints and reference guides ▪ TOGAF ▪ E2AF ▪ FEA
3. XBA	Cross Business-line Architecture	• Business-line-independent functionality • Service patterns • State data • Service layers • Deployment • Channel patterns	- Enterprise Architect - Infrastructure Architect - Security Architect - Network Architect - Storage Architect - Information Architect	▪ TOGAF ▪ MSA blueprints and reference guides ▪ FEA ▪ Service-Oriented Architecture (SOA) ▪ BPM ▪ Industry models (e.g. ACORD, IFW, eTOM)
4. CSA	Channel Specific Architecture	• Channel-dependent architecture • Common baseline to address major cross-application concerns • Quality of services • Best practices • Application patterns and frameworks • Inter-application collaborations and integration	- Enterprise Architect - Solutions architect - Infrastructure Architect - Security Architect - Network Architect - Storage Architect - Information Architect - Systems Architect	▪ TOGAF ▪ MSA blueprints and reference guides ▪ FEA ▪ MDA ▪ Service-oriented business service model ▪ BPM ▪ Industry models (e.g. ACORD, IFW, eTOM)

5. ASA	Application Solution Architecture	• Application-specific architecture • Business functionality realization • Business logic implementation • Technology & system architecture • Service access and n-tier model • Networking, storage, & resource integration	- Solutions architect - Systems Architect - Application Architect - Infrastructure Architect - Network Architect - Information Architect - Portfolio Architect	▪ MDA ▪ SCA ▪ Java EE platform ▪ Application Architecture for .NET ▪ Architectural styles ▪ LAMP ▪ Ruby on Rails
6. AOA	Aspect-Oriented Architecture	• Application-wise aspects • Crosscutting concerns • Module framework, e.g. MVC • Module technology (data validation) • Programming model (OOD) • Development/Testing tools • Exception handling • Data caching • Session and state management • Transactions • Threading • Workflow • Business rules • Authentication & authorization	- Software Architect - Solutions architect - Application Architect - Information Architect - Data Architect	▪ Struts, JSF, Tapestry, Rife ▪ Ajax ▪ EJB ▪ MQ, JMS, ActiveMQ ▪ AspectJ, AspectWerkz, Spring AOP, JBoss AOP ▪ Log4J ▪ ESB ▪ WS-BPEL ▪ OFBiz ▪ Patterns ▪ MS UIP application block ▪ Genetics ▪ Annotations
7. CTA	Component Technology Architecture	• Component-level internal structure and technologies • Object-Relation (OR) mapping • Presentation-rendering mechanisms like XSL and template engines • Page flow navigation • Look & Feel • XML parsing and construction • Persistent data model • Web Services invocation • Collaboration • Integration	- Technology Architect - Software Architect - Component Architect	▪ Design patterns ▪ SDO, JDO, Hibernate ▪ Beehive, Spring WebFlow ▪ JAX-WS, Axis ▪ WS-Security, WSRP, WS-* ▪ JAXP, DOM/SAX, StAX ▪ XDoclet ▪ JUnit, HttpUnit, NUnit, Cactus ▪ MySQL, mSQL, Derby ▪ Application blocks in MS Enterprise Library

The design principles of the pyramid platform are discussed in this context. A concept of Meso-Architecture is introduced, which emphasizes the important architectural artifacts at the service and channel levels in the architecture modeling practices. Seven interrelated layers are defined in the Generic Architecture Stack.

The strength of this comprehensive platform is its loose-coupling nature and interoperability. In our practices, different formats and variants of this model have been successfully used in developing and integrating various IT solutions in a SOA fashion. Furthermore, this framework is scalable and flexible for dynamic expansions and customization.

References

1. John Zachman: Zachman Framework, http://www.zifa.com
2. Institute for Enterprise Architecture Developments: Extended Enterprise Architecture Framework
3. Philippe Kruchten: The Rational Unified Process: An Introduction, 3rd Edition, Addison Wesley, Massachusetts (2003)
4. The Open Group: The Open Group Architecture Framework, http://www.opengroup.org/architecture/togaf8/index8.htm
5. Object Management Group: Model Driven Architecture, http://www.omg.org/mda
6. DoD C4ISR Architecture Working Group: C4ISR Architecture Framework, Version 2
7. Treasury Department CIO Council: Treasury Enterprise Architecture Framework. Version 1
8. Federal Office of Management and Budget: Federal Architecture Framework, http://www.feapmo.gov/fea.asp
9. Purdue University: The Purdue Enterprise Reference Architecture, http://pera.net
10. Janis R Putman: Architecting with RM-ODP, Prentice Hall PTR, New Jersey (2001)
11. Tony Shan, and Winnie Hua: Solution Architecture of N-Tier Applications, 3rd IEEE Conference on Services Computing (SCC 2006), September 2006, 349-256
12. Tony Shan, and Winnie Hua: Solution Architecting Mechanism, 10th IEEE Enterprise Distributed Object Computing Conference (EDOC 2006), October 2006, 23-34
13. Tony Shan and Winnie Hua: Service-Oriented Solution Framework for Internet Banking, International Journal of Web Services Research, Vol. 3, No.1 (2006), 29-48
14. The Open Service Oriented Architecture Collaboration: Service Component Architecture, http://www.osoa.org
15. Tony Shan, and Winnie Hua: Contextual Spectrums in Technology and Information Platform, 3rd IEEE Conference on Services Computing (SCC 2006), September 2006, 508