Chapter I
Towards a Systematic Method for Solutions Architecting

Tony C. Shan
Bank of America, USA

Winnie W. Hua
CTS Inc., USA

ABSTRACT

Solutions architecting method (SAM) is defined as a methodical approach to dealing with the architecture complexity of enterprise information systems in IT solution designs. This comprehensive method consists of eight interconnected modules: framework for e-business architecture and technology, prescriptive artineering procedure, technology architecture planning, architecture stack and perspectives, rapid architecting process, architecture readiness maturity, generic application platform, and Tao of IT development & engineering. Collectively, these modules form a holistic discipline guiding the process of developing architected solutions in an enterprise computing environment. Several unconventional concepts and thinking styles are introduced in this overarching structure. This systematic method has been customized and adapted to be extensively applied in one form or another to develop various IT solutions across a broad range of industrial sectors. Reference solutions are presented and articulated to illustrate the exemplary implementations of some key elements in SAM. Best practice and lessons learned as well as future trends are discussed in the context.

INTRODUCTION

The e-business models in today's fast-paced on-demand business world mandate increasing flexibility of information systems applications. It is compulsory for the information technology (IT) group to provide a higher level of services at a lower cost for the business to compete and succeed in a globalized economy. The reality is that IT must build more complicated, flexible, scalable, extensible, and forward-thinking technical solutions, to satisfy the ever-growing business needs.

In large organizations like worldwide financial institutions, virtually hundreds, if not thousands of IT applications and systems have been built, acquired, or purchased through the years, to provide both external customers and internal employees with

reliable electronic services, utilizing heterogeneous technologies and architectures to satisfy diverse functional requirements from different lines of business. In the financial services industry, the banking business processes generally involves different business divisions that address retail, commercial, investment, wealth management, treasury, and capital markets. In particular, services are delivered via different channels. To effectively manage the architecture assets and design top-quality IT solutions in such a diverse environment, a highly structured methodology is of critical importance to achieve an array of goals—separate concerns, divide responsibilities, encapsulate the complexity, utilize patterns, leverage best practices, control quality, ensure compliance, and establish execution processes.

BACKGROUND

The computing paradigm has gone through several generations of evolution in the last five decades: monolithic, client/server, multi-tier, structured methods, object-oriented, component-based, service-oriented, and event-driven model. The overall solution architecture has become increasingly complicated and thus hardly manageable through a traditional waterfall process. Previous work over the past few years has strived to address the complexity issue in the architecture design and process. A pioneer effort in this space was the Zachman framework (Zachman, 1987), which is a logical structure to classify and organize the descriptive representations of an enterprise computing environment, which are important to the development of the enterprise systems and the enterprise management. In a form of a two-dimensional matrix to symbolize the enterprise architecture environments, it has achieved a substantial level of penetration in the domain of business and information systems architecture as well as modeling. Though it is primarily used as a planning or problem-solving tool, it tends to implicitly align with data-driven and process-decomposition methods and processes, and it operates above and across the individual project level. A similar approach is taken in the extended enterprise architecture framework (E2AF) (IEAD, 2004) with a scope of aspect areas containing business, information, system, and infrastructure in a 2-D matrix. Rational unified process (RUP) (Kruchten, 2003) overcomes these shortcomings by taking a use-case driven, object-oriented and component-based approach, using a standard notation—unified modeling language (UML). The concept of 4+1 views offers multi-perspective interpretations of the overall system structure. RUP is more process-oriented, and to some extent is a waterfall approach. RUP has little to address software maintenance and operations, and lacks a broad coverage of physical topology and development/testing tools. It generally operates at the individual project level. Enterprise unified process (EUP) (Nalbone, 2005) attempts to extend the RUP to cover the entire IT lifecycle. An open source unified process (OpenUP/Basic) is also under development in Eclipse (OpenUP, 2007).

Another heavyweight approach, the open group architecture framework (TOGAF) (Open Group, 2007), is a comprehensive framework with a set of supporting tools for developing an 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 resource base. The scope of TOGAF covers business process architecture, applications architecture, data architecture, and technology architecture.

All these approaches are heavyweight methodologies, which require a fairly steep learning curve to get started. On the other hand, model-driven architecture (MDA) (OMG, 2007) takes a lightweight approach. MDA aims to separate business logic or application logic from 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. The primary focus of MDA is on software modeling in the development life-cycle process.

Quite a few agile methods are available such as extreme programming (XP), dynamic systems development method (DSDM), agile modeling (AM), feature driven development (FDD), crystal, adaptive software development (ASD), scrum, and