Enterprise Architectures: Purposes, definitions, scope, and methods

Bo Sundgren
2012-05-14

Paper presented at the EA conference in Härnösand, Sweden, 2012-05-14

Historical background

According to Sessions (2007), it was shortly before 1990 that a new field was born that soon came to be known as enterprise architecture. The field initially began to address two problems:

1. System complexity – Organisations were spending more and more money building and maintaining IT systems

2. Poor business alignment – Organisations were finding it more and more difficult to keep those increasingly expensive IT systems aligned with business needs

Expressed in a bit more elaborated way, organisations experienced that:

- the IT systems were unmanageably complex and increasingly costly to maintain
- the IT systems were hindering the organisation's ability to respond to current, and future, market conditions in a timely and cost-effective manner
- mission-critical information was consistently out-of-date and/or just wrong
- a culture of distrust existed between the business and technology sides of the organisation

Some definitions

Here are three selected definitions from the literature:

1. Weill (2007): … the organising logic for business processes and IT infrastructure reflecting the integration and standardisation requirements of the company's operating model … the desired state of business process integration and business process standardisation for delivering goods and services to customers

2. United States Government (2007): … defines the mission of an agency and describes the technology and information needed to perform that mission, along with descriptions of how the architecture of the organisation should be changed in order to respond to changes in the mission (United States Government)

3. Gartner (2012): … the process of translating business vision and strategy into effective enterprise change by creating, communicating and improving the key requirements, principles and models that describe the enterprise's future state and enable its evolution
Scope

The term “enterprise” in “enterprise architecture” is a generic and potentially complex term:

- The term *enterprise* is generally applicable in many circumstances, including
  - Public or private sector organizations
  - An entire business or corporation
  - A part of a larger enterprise (such as a business unit)
  - A conglomerate of several organisations, such as a joint venture or partnership
  - A multiply outsourced business operation
  Wikipedia (2012)

- The term *enterprise* includes the whole complex, socio-technical system, including:
  - people
  - information
  - technology
  - business (e.g. operations)
  Giachetti (2010)

- Defining the boundary or scope of the enterprise to be described is an important first step in creating the enterprise architecture. “Enterprise” means more than the information systems employed by an organisation.
  Giachetti (2010)

- Composing holistic solutions that address the business challenges of the enterprise and support the governance needed to implement them. Gartner (2012).

Methods and tools

According to Wikipedia (2012):

- Enterprise architects use various methods and tools to capture the structure and dynamics of an enterprise. In doing so, they produce taxonomies, diagrams, documents and models. These artifacts describe the logical organization of business functions, business capabilities, business processes, people organization, information resources, business systems, software applications, computing capabilities, information exchange and communications infrastructure within the enterprise.

- Normally an EA takes the form of a comprehensive set of cohesive models that describe the structure and functions of an enterprise.

- The individual models in an EA are arranged in a logical manner that provides an ever-increasing level of detail about the enterprise: its objectives and goals; its processes and organization; its systems and data; the technology used and any other relevant spheres of interest.
• An enterprise architecture framework bundles tools, techniques, artifact descriptions, process models, reference models and guidance used by architects in the production of enterprise-specific architectural description.

Top four EA methodologies

According to Sessions (2007) four of the most popular EA methodologies are:

• The Zachman Framework for Enterprise Architectures; Zachman (1987), Zachman&Sowa (1992) — Although self-described as a framework, is actually more accurately defined as a taxonomy

• The Open Group Architectural Framework (TOGAF); The Open Group (2012) — Although called a framework, is actually more accurately defined as a process

• The Federal Enterprise Architecture; The White House (2007) — Can be viewed as either an implemented enterprise architecture or a prescriptive methodology for creating an enterprise architecture

• The Gartner Methodology; Gartner (2005) — Can be best described as an enterprise architectural practice

For many enterprises, none of these methodologies will be a complete solution. Another approach, a blended methodology, means choosing bits and pieces from each of these methodologies, and modify and merge them according to the specific needs of your organisation.

The challenge hasn't changed: reducing IT cost and complexity, while increasing business value and effectiveness.

The Zachman Framework

DataGovernance.com (2008) summarises the Zachman Framework as follows:

“The Zachman Framework is a de facto world standard for expressing the basic elements of an Enterprise Architecture. It provides a formal and highly structured way of defining an enterprise that allows people to clearly describe what group of stakeholders within an enterprise is under discussion, and what concern is being considered.

The framework was originally conceived by John Zachman at IBM in the 1980s. Originally the full technical name was the Zachman Framework for Information Systems Architecture; it was changed in the early 90's to The Zachman Framework for Enterprise Architecture.

The Zachman Framework can be represented as a grid. Stakeholder groups are represented in six rows: (Visionary, Owner, Designer, Builder, Implementer and Worker)

Columns depict "What" (Data), "How" (Function), "Where" (Network), "Who" (People), "When" (Time), and "Why" (Motivation). Within each cell of the grid are typical artifacts to be used in modeling the enterprise.”
For more information:


The official Zachman Framework site: http://www.zifa.com/

JPEG and PDF version of the Zachman Framework: http://www.zifa.com/framework.html


**TOGAF’s Enterprise Architecture**

The following description of TOGAF follows Sessions (2007):

The Open Group Architecture Framework is best known by its acronym, TOGAF. TOGAF is owned by The Open Group. TOGAF's view of an enterprise architecture is shown in the figure below.
As shown in the figure, TOGAF divides an enterprise architecture into four categories, as follows:

1. **Business architecture**—Describes the processes the business uses to meet its goals
2. **Application architecture**—Describes how specific applications are designed and how they interact with each other
3. **Data architecture**—Describes how the enterprise datastores are organized and accessed
4. **Technical architecture**—Describes the hardware and software infrastructure that supports applications and their interactions

TOGAF describes itself as a "framework," but the most important part of TOGAF is the Architecture Development Method, better known as ADM. ADM is a recipe for creating architecture. A recipe can be categorized as a *process*. Given that ADM is the most visible part of TOGAF, I categorise TOGAF overall as an *architectural process*, instead of either an *architectural framework* (as The Open Group describes TOGAF) or a *methodology* (as it describes ADM).

Viewed as an architectural *process*, TOGAF complements Zachman. Zachman tells you how to categorise your artifacts. TOGAF gives you a process for creating them.

TOGAF views the world of enterprise architecture as a continuum of architectures, ranging from highly generic to highly specific. It calls this continuum the Enterprise Continuum. It views the process of creating a specific enterprise architecture as moving from the generic to the specific. TOGAF's ADM provides a process for driving this movement from the generic to the specific:

**Level 1.** TOGAF calls most generic architectures *Foundation Architectures*. These are architectural principles that can, theoretically, be used by any IT organisation in the universe.

**Level 2.** TOGAF calls the next level of specificity *Common Systems Architectures*. These are principles that one would expect to see in many—but, perhaps, not all—types of enterprises.

**Level 3.** TOGAF calls the next level of specificity *Industry Architectures*. These are principles that are specific across many enterprises that are part of the same domain.
**Level 4.** TOGAF calls the most specific level the *Organisational Architectures*. These are the architectures that are specific to a given enterprise.

**Gartner EA Process Model**

According to *Sessions (2007)*, the best summation of the Gartner practice is the following one:

- Architecture is a verb, not a noun.

Sessions continues the explanation as follows. It is the ongoing process of creating, maintaining, and, especially, leveraging an enterprise architecture that gives an enterprise architecture its vitality. An architecture that is just a bunch of stiff artifacts that sit in a corner gathering dust is useless, regardless of how sophisticated your taxonomy is for categorising those artifacts or how brilliant your process is that guided their development.

Gartner believes that enterprise architecture is about bringing together three constituents: business owners, information specialists, the technology implementers. If you can bring these three groups together and unify them behind a common vision that drives business value, you have succeeded; if not, you have failed. Success is measured in pragmatic terms, such as driving profitability, not by checking off items on a process matrix.

Gartner believes that the enterprise architectures must start with where an organisation is going, not with where it is. If we are going to clean house, we don't need to exhaustively document everything we are throwing out. Let's focus our energy on what we want to end up with. As soon as we know our goal, we can see how what we have relates to that goal.

Gartner recommends that an organisation begin by telling the story of where its strategic direction is heading and what the business drivers are to which it is responding. Gartner will want this story in plain language, without worrying about prescribed documentation standards, acronyms, or techno-babble. The only goal is making sure that everybody understands and shares a single vision.

Most organisations are facing major changes in their business processes. The process of creating an enterprise-architecture vision is the organization's opportunity to sit down, take a collective breath, and ensure that everybody understands the nature, the scope, and the impact of those changes.

As soon as an organisation has this single shared vision of the future, it can consider the implications of this vision on the business, technical, information, and solutions architectures of the enterprise. The shared vision of the future will dictate changes in all of these architectures, assign priorities to those changes, and keep those changes grounded in business value.

Enterprise architecture, in the Gartner view, is about strategy, not about engineering. It is focused on the destination. The two things that are most important to Gartner are *where an organisation is going and how it will get there*. Any architectural activity that is extraneous to these questions is irrelevant. "Just enough enterprise architecture, just in time," is another saying you will hear from the Gartner analyst. So far *Sessions (2007)*.
The Federal Enterprise Architecture (FEA)

FEA is described in several documents issued by the United States Government, e.g. The United States Government (2007), and The White House (2007). The following description is based on those documents, and on Sessions (2007).

The figure below is from The United States Government (2007), page 8.

FEA is an attempt by the United States Government to unite its agencies and functions under a common enterprise architecture. FEA is a rather complete methodology. It has both a comprehensive taxonomy, like Zachman, and an architectural process, like TOGAF. FEA can be viewed as

- either a methodology for creating an enterprise architecture
- or the result of applying that process to a particular enterprise—namely, the U.S. Government.

FEA consists of:

- A perspective on how enterprise architectures should be viewed (the segment model)
- A set of reference models for describing different perspectives of the enterprise architecture (business, service, components, technical, and data)
- A process for creating an enterprise architecture
- A transitional process for migrating from a pre-EA to a post-EA paradigm
- A taxonomy for cataloging assets that fall within the enterprise architecture
- An approach to measuring the success of using the enterprise architecture to drive business value

There is a hope that FEA could help to transform the Federal government into a citizen-centered, results-oriented, and market-based organization.

The FEA Perspective on EA is that an enterprise is built of segments. A segment is a major line-of-business functionality, such as human resources. There are two types of segments: core mission-area segments and business-services segments.
A "core mission-area segment" is one that is central to the mission or purpose of a particular political boundary within the enterprise. For example, in the Health and Human Services (HHS) agency of the federal government, health is a core mission-area segment.

A "business-services segment" is one that is foundational to most, if not all, political organisations. For example, financial management is a business-services segment that is required by all federal agencies.

**Service Oriented Architecture (SOA)**

As a smooth continuation from the description of FEA above, we will start our discussion of Service Oriented Architectures by describing an initiative by the Chief Information Officers of the United States federal administration.

**SOA as an extension to EA**

The Chief Information Officers of the United States federal administration have taken an initiative to extend the Federal Enterprise Architecture (FEA) to include the concept of a Service Oriented Architecture. See [United States Government (2008)](https://example.com). This is how they motivate their initiative:

"The world is changing at an accelerating rate and the federal government needs to keep pace. Broad-based change is always difficult, but the federal government is plagued by a variety of inhibitors to change, including vertical vs. mission organizational orientation; bureaucratic culture; budgetary cycles and processes that do not facilitate agility or reuse; and a large and diverse current technology base. Service Oriented Architecture (SOA) promises to help agencies rapidly reconfigure their business and more easily position IT resources to serve it. Improved business agility – through sharing and reuse of infrastructure, services, information, and solutions - is a growing requirement in the federal government today and will be increasingly critical in the future.

The purpose of this document is to describe a target federal service oriented architecture vision and to provide guidance in the management and governance of enterprise-wide services. Many federal organizations are considering or planning for a broad based adoption of SOA. In order to effectively move to an SOA environment, an organization must conduct careful planning and assessments for a variety of organizational, architectural, and technological challenges.

With recent advances in federal enterprise architecture, federal chief architects and chief information officers have a deeper insight into their current IT architectures at all levels of government. In most organizations, this visibility has exposed many inefficiencies and undesirable redundancies, as well as disconnect between the promise and the reality of technology for improving business outcomes. In turn, this has led to a variety of consolidation initiatives and reengineering efforts at all levels of the federal government. The most widely publicized and recognizable are those government-wide initiatives compiled into the annually published Federal Transformation Framework (FTF) from the Office of Management and Budget (OMB).

While the FTF is concerned with cross-agency initiatives which leverage reuse efficiencies and improved organizational performance, agencies themselves are faced with similar internal
challenges. Recognizing this concern, as well as others, OMB published the Federal Enterprise Architecture (FEA) Practice Guidance that introduces Segment and Solution Architectures and their relationships with Enterprise Architecture (EA) through a notional framework (see Figure 1-3 of the FEA Practice Guidance document). The Solution Architecture is equivalent to an IT system that is reconciled to the Segment Architecture. The FEA Practice Guidance strongly indicates that Segment and Solution Architectures inherit their structure, policies and standards and re-usable and sharable solutions from the Enterprise Architecture. This is directly aligned with the direction of Service Oriented Architecture.

Just as industry has adopted SOA best practices, it stands to reason that federal organizations will turn to SOA best practices to optimize their IT and business architectures. SOA is not just a technology to be leveraged; it is a true paradigm shift and requires substantial organizational, cultural and management changes to be effective.”

The following figure from United States Government (2008) illustrates how SOA best practices could extend an enterprise architecture:

What is a Service Oriented Architecture (SOA)?

When designing information systems, one may use standardised structuring methods and architectures, such as database orientation, process orientation, client/server architecture, and service orientation.
Today’s applications are often database-oriented, that is, different functions of the system interact with each other via a common database, including both data and metadata.

Until recently, database-orientation has often been combined with a structuring of the information system according to the client/server principle. In its original form, the client/server architecture consists of two types of subsystems: user-oriented client systems, which are served by server systems, handling common resources like printers and databases. There are developments of the client/server architecture, using three or more types of subsystems, called tiers. In a three-tier client/server architecture there is a distinction between

- subsystems for user interactions
- subsystems for business logic
- subsystems for data management

With the rapidly growing importance of the Internet and web-based information systems, the client/server architecture is becoming replaced by service-oriented architectures (SOA), based on well-defined, standardised services, which can be used in a standardised way, via standardised messages and communication protocols, by other services.

Service-oriented architectures are based on the following design principles; Erl (2005):

- **Loose coupling** – Services maintain a relationship that minimises dependencies and only requires that they retain an awareness of each other.
- **Service contract** – Services adhere to a communications agreement, as defined collectively by one or more service descriptions and related documents.
- **Autonomy** – Services have control over the logic they encapsulate.
- **Abstraction** – Beyond what is described in the service contract, services hide logic from the outside world.
- **Reusability** – Logic is divided into services with the intention of promoting reuse.
- **Composability** – Collections of services can be coordinated and assembled to form composite services.
- **Statelessness** – Services minimise retaining information specific to an activity.
- **Discoverability** – Services are designed to be outwardly descriptive so that they can be found and assessed via available mechanisms.

More briefly and concretely expressed, a service is a piece of reusable software, smaller or bigger, which performs a well-defined function, described in a standardised way. The service can be requested by other pieces of software, which may themselves be services, through standardised messages. The service requestor should not have to know anything about the internal functioning of the activated service, and the latter should not have to know anything about its external environment, but only perform its function and (possibly) provide a standardised response message in return. During its execution a service may itself request the execution of other services in the same way.

The figure below illustrates the SOA concept.
Service-Oriented Architecture (SOA)

Service-orientation, as defined above, has the great advantage that it can be introduced step by step in an organisation, e.g. a statistical agency. Any large organisation today has an enormous burden of legacy systems that cannot quickly and easily be redesigned and redeveloped. A legacy system that has not been developed in accordance with modern design principles can be encapsulated into a large black box component, which is not internally consistent with service-oriented principles, but which interacts with its environment according to such principles. Of course it requires some work to develop the “sarcophagus” surrounding the black box, making it look and behave like a true service to the other services in the system, with which it interacts, but this is a small effort in comparison with a total make-over or redevelopment of the whole legacy system. The figure below illustrates the sarcophagus approach for coping with complex legacy systems.

**SOA and legacy systems: using “sarcophaguses”**

Service-orientation, as defined above, has the great advantage that it can be introduced step by step in an organisation, e.g. a statistical agency. Any large organisation today has an enormous burden of legacy systems that cannot quickly and easily be redesigned and redeveloped. A legacy system that has not been developed in accordance with modern design principles can be encapsulated into a large black box component, which is not internally consistent with service-oriented principles, but which interacts with its environment according to such principles. Of course it requires some work to develop the “sarcophagus” surrounding the black box, making it look and behave like a true service to the other services in the system, with which it interacts, but this is a small effort in comparison with a total make-over or redevelopment of the whole legacy system. The figure below illustrates the sarcophagus approach for coping with complex legacy systems.

Complex monolithic legacy system ...

... encapsulated by a "sarcophagus" (or a cloud) ...

... which makes the complex, monolithic system behave towards its environment like a set of loosely coupled, well-defined services (although they are faked) ...

... and step by step the faked services are replaced by real services in the new, emerging and truly service-oriented system
SOA, process-orientation, object-orientation, and the systems approach

Service-orientation often goes hand in hand with process-orientation. On the business level – for example the business of statistics production – the employees interact with customers, suppliers (respondents and data providers in the case of statistics production), colleagues, and external and internal service systems (typically computerised), in order to provide services, demanded by the customers, to the customers. This work may be organised into processes, preferably standardised processes, so as to ensure that the work is done according to best methods and best practices and will give the same good quality results to the customer, regardless of which individual persons are executing the processes.

Service-orientation can be seen as a further development of earlier software design methodologies like modular programming and object-orientation. It is obviously well in line with the general systems approach and systems thinking; cf the description of services above with treatments of the systems concept and about how to manage complexity and unperceivable systems; see for example Langefors (1966).

A short summary of the systems approach for managing complex, unperceivable systems:

- The systems approach is a paradigm for managing the design, operation, and evaluation of unperceivable systems, systems which are too big and complex for the human brain to grasp in one go
- Combination of top-down and bottom-up, overview and detail, comprehensiveness and precision, soft and hard

Software and exit strategies

Another recent trend is to replace in-house software developments, and even in-house licensing and installation of commercial software packages, with software components that are provided as services, for free or for a fee, via the Internet. This is called “cloud computing” or “Software as a Service”, SaaS, and is also consistent with service-oriented architectures and process orientation.

Here are some things to consider in this context:

- Having a sound architecture (e.g. SOA) is more important than exactly which software products are chosen
- But having an exit strategy for each software product is essential – and more essential the more monopolistic the vendors are, and the more closed the software products are
  - Worst: SAS, ORACLE, Apple, ...
  - Not quite so bad: Microsoft, ...
  - Rather good: Google, ...
  - Open source software? Good, but competitive service providers are necessary, otherwise you must have your own technical experts, which is costly and makes you vulnerable
Typical concepts and terms in EA practice

From Wikipedia (2012). Note that most of the concepts and terms belong to the solutions level of an Enterprise architecture:

<table>
<thead>
<tr>
<th>Fields</th>
<th>Requirements analysis, Systems analysis, Software design, Computer programming, Formal methods, Software testing, Software deployment, Software maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concepts</td>
<td>Data modeling, Enterprise architecture, Functional specification, Modeling language, Programming paradigm, Software, Software architecture, Software development methodology, Software development process, Software quality, Software quality assurance, Software archaeology, Structured analysis</td>
</tr>
<tr>
<td>Orientations</td>
<td>Agile, Aspect-oriented, Object orientation, Ontology, Service orientation, SDLC</td>
</tr>
<tr>
<td>Models</td>
<td><strong>Development models</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Other models</strong></td>
</tr>
<tr>
<td>Modeling languages</td>
<td>IDEF, UML, SysML</td>
</tr>
<tr>
<td>Related fields</td>
<td>Computer science, Computer engineering, Enterprise engineering, History, Management, Project management, Quality management, Software ergonomics, Systems engineering</td>
</tr>
</tbody>
</table>

References

DataGovernance.com (2008) *Zachman Framework*  
http://www.datagovernance.com/fwo3_zachman_framework.html

Prentice-Hall


CRC Press, Boca Raton, FL.

Langefors, B. (1966) *Theoretical analysis of information systems*  
Studentlitteratur, Sweden.


Sundgren, B. (2010a) *The SOA paradigm*  
Sundgren, B. (2010b) *The systems approach to Official Statistics*
http://www.opengroup.org/togaf/

http://www.whitehouse.gov/omb/e-gov/fea/


