



Enterprise Architectures and the International e-Framework

Title:	Enterprise Architectures and the International e-Framework
Editor(s):	Phil Nicholls (e-Framework Technical Editor)
Version:	1.3
Release date:	July 2009
Status:	Final
Distribution:	e-Framework website
Summary:	This document details the e-Framework in the context of Enterprise Architectures.

Revision History

Version Number	Release Date	Comments
1.0	16 th April 2008	Created
1.1	4 th June 2008	Updated after comments from WK
1.2	24 th June 2008	Updated after comments from GB
1.3	6 th July 2009	Published by e-Framework

This work is licensed under the Creative Commons Attribution-Share Alike 2.5 Australia license. To view a copy, visit <http://creativecommons.org/licenses/by-sa/2.5/au/> or send a letter to 171 2nd Street, Suite 300, San Francisco, California, 94105, USA



Table of Contents

1	INTRODUCTION.....	4
1.1	SCOPE AND CONTEXT	4
1.2	STRUCTURE OF THIS DOCUMENT	4
2	THE E-FRAMEWORK	5
2.1	INTRODUCTION	5
2.2	SERVICE GENRES	6
2.3	SERVICE EXPRESSIONS.....	7
2.3.1	SERVICE IMPLEMENTATIONS AND SERVICE INSTANCES	8
2.3.2	SPECIFICATIONS AND STANDARDS	9
2.4	SERVICE USAGE MODELS (SUMS).....	9
3	ENTERPRISE ARCHITECTURES	11
3.1	INTRODUCTION	11
3.2	ENTERPRISE ARCHITECTURE FRAMEWORKS.....	11
3.2.1	COMPONENTS OF AN ENTERPRISE ARCHITECTURE	12
4	COMPARISONS	14
4.1	INITIAL IMPRESSIONS.....	14
4.2	LIMITATIONS	15
4.2.1	HORIZONTAL BEHAVIOURS VS. VERTICAL STACKS	15
4.2.2	OTHER ISSUES.....	17
4.3	A BRIEF NOTE ON FEA.....	17
5	CONCLUSIONS.....	19
	REFERENCES.....	20

1 INTRODUCTION

1.1 Scope and Context

This document is an introductory report covering the relationships between Enterprise Architecture and the International e-Framework. Both Enterprise Architecture and the e-Framework are introduced separately, before comparisons between them are made. Conclusions are offered following the comparison.

1.2 Structure of this Document

The structure of the rest of this document consists of:

SECTION	Summary Description
2. THE E-FRAMEWORK	This section presents a brief overview of the e-Framework, the technical components of the e-Framework, what the e-Framework seeks to achieve and how it does it.
3. ENTERPRISE ARCHITECTURE	This section presents a brief overview of Enterprise Architecture, what it describes, what it seeks to achieve and how it does it.
4. COMPARISONS	This section undertakes a comparison of the e-Framework and Enterprise Architectures; how they complement each other, how they work together and so forth.
5. CONCLUSIONS	This section presents conclusions based on the findings within the rest of the document.
REFERENCES	

2 THE E-FRAMEWORK

2.1 Introduction

This section describes the e-Framework, the problem that it solves, and how it does it. The components of the e-Framework are described, and how these are typically used.

The International e-Framework is an initiative that seeks to promote the use of the *service-oriented approach* in the analysis and design of software for use within education and research. Components of the e-Framework are shared across the four funding partners of the e-Framework (UK – JISC, Australia – DEEWR, Netherlands – SURF, New Zealand – MoE).

The service-oriented approach promotes the creation and re-use of distinct software *services* – loosely coupled, autonomous, interoperable, discoverable and reusable software components that can be used by different applications in different contexts.

As a (trivial) example, consider a desktop software application such as a word processor. The word processor offers a 'spell checker' function. This function only exists within the word processor; other applications do not come with a spell checker and thus do not provide this function. In order for other applications to make use of the spell checker, a number of options exist. First, a human could copy the information to be spell checked from one application to the word processor, verify the spelling, and copy any alterations back to the first application. Secondly, the software company providing the other application could be approached to add a spell check function. If that company also implemented the word processor, then it should not be difficult to re-use the spell checker library in the other application. However, if the word processor and the other application are owned by different companies, or implemented in different programming languages, then the task of adding the spell checker becomes much more complicated and time consuming.

In the service oriented view of software design, only one spell check service is needed. This exists somewhere on the internet, and provides a standard interface by which other applications can make use of it. Thus any company could potentially use the spell check service; and the means of the standard interface should eliminate any problems caused by implementing in different languages. The spell check service will not have any knowledge itself of why it is being called and would be ignorant of the applications which are calling it.

The e-Framework provides a mechanism by which services may be modelled in an implementation-neutral manner. Further, the e-Framework as an ontology enables different service components to be compared. The e-Framework provides a consistent vocabulary by which services may be described. The individual components of the e-Framework describe the behaviours of the different services, as well as showing the precise interfaces to them. The e-Framework also demonstrates how different services may be combined to support a business process within education and research.

Figure 2-1 shows the first view of the e-Framework, that of its atomic components. Figure 2-1 shows not only the internal components of the e-Framework, but also those entities that are identified within the e-Framework but are not themselves described within it. The technical components of the e-Framework are Service Genres and Service Expressions (and SUMs, see figure 2-2). Service Implementations and Service Instances are explained in more detail in the section on Service Expressions. Specifications and Standards (e.g. IMS Metadata, LOM) are also used by Service Expressions – but are not defined by the e-Framework, merely used by it.

All of the e-Framework components are described in template documents, which can be found on the e-Framework website¹. It is recognised that e-Framework components will change over time, due to modelling and other funded project activities.

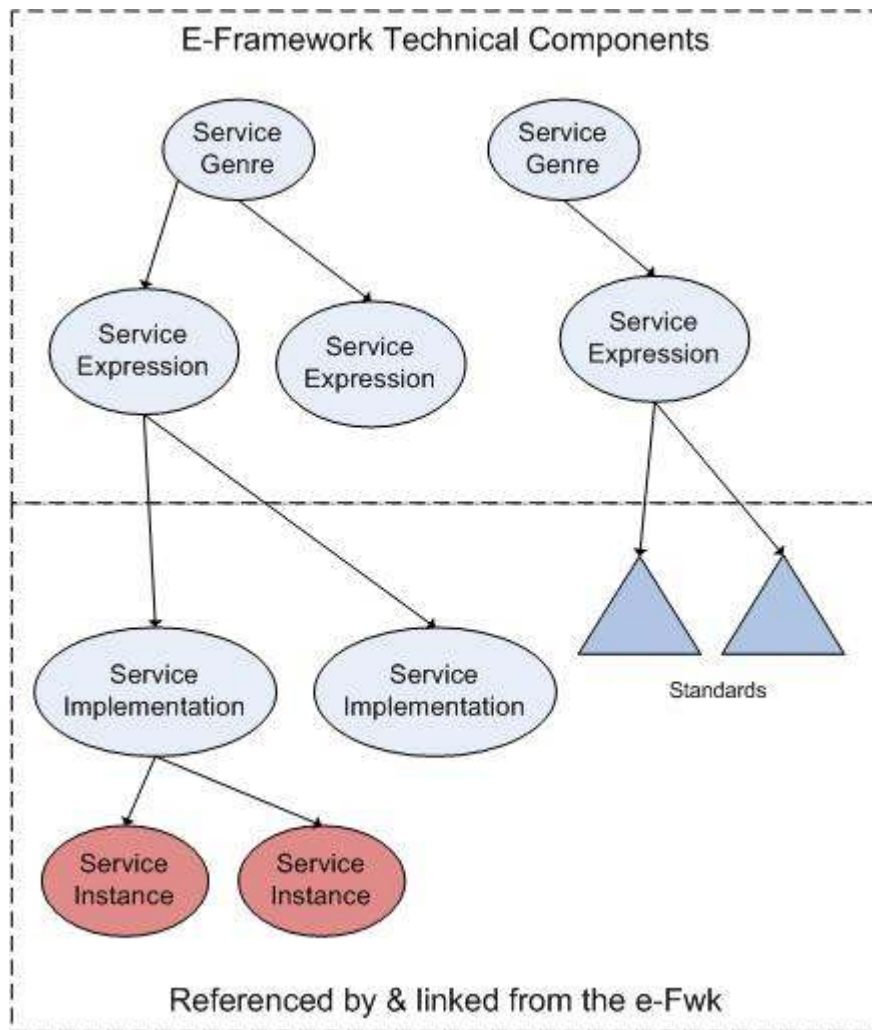


Figure 2-1: e-Framework Components

The following sections will introduce these components and detail how they work.

2.2 Service Genres

Service Genres lie at the heart of the e-Framework. The Service Genre lists a collection of behaviours that might be invoked by a user. By listing the behaviours offered by a service, a Service Genre describes what services do without going into any detail on how it's done. An implemented service (i.e. an implementation or instance) can be classified as belonging to a particular genre if that service exhibits those behaviours. Service Genres do not overlap; they define discrete atomic behaviours in terms of requests and the sorts of parameters that apply to them.

It should be noted that a Service Genre does not specify *how* a service works; rather it only specifies *what* a service should do. Services could implement Genres however they choose, as long as they provide the behaviours specified in the Genre itself.

Service Genres are the root building blocks of the e-Framework, defining core capabilities. Service Genres are typically named to reflect the behaviours that they describe, for example "alert", "search" etc.

The Alert² Service Genre defines the set of behaviours that services wishing to provide alert style functionality need to offer. The template document contains descriptions of that functionality.

The following extracts show some of the information that is present within the Alert Service Genre.

“[Alert] details the notion of notifying a user or system as to the occurrence of an event. The event itself could be extra-ordinary (such as a fire alarm), or it could be something more mundane (such as informing students of a cancelled lecture). The important notion in alerting is that the alert data itself is actively propagated to receivers rather than passively propagated.

The model for Alert assumes that requesters (which may be users or machines) issue an alert message to an Alert Service. The Alert Service then propagates the Alert Message to a range of different receivers. Those receivers could be other Services or users. At the Genre level, there are no limits on the transport mechanisms that Alert Services can use to send the Alert Message. Receivers are assumed to be pre-registered with the Alert Service (for humans, this might be via an email address or telephone number that the Alert Service has access to).

At least one request SHOULD specify the Create Alert functionality.

At least one request MAY specify the Cancel Alert functionality.

Create Alert SHALL meet the following conditions:

- * The Request SHOULD specify the Alert Message:
 - o The Alert Message SHOULD contain a timestamp.
 - o The Alert Message SHOULD contain a description of the alert itself.
 - o The Alert Message SHOULD contain a source from where the Alert Message originated.
 - o The Alert Message MAY contain a severity.
 - o The Alert Message MAY contain a duration.
- * The Request MAY specify the intended Receivers of the Alert Message.
- * Responses SHALL include error messages or other needed control information.

Cancel Alert SHALL meet the following conditions:

- * The Request SHOULD specify the Alert Message which is being cancelled.
- * Responses SHALL include error messages or other needed control information”.

2.3 Service Expressions

A Service Expression is a specialisation of a Service Genre. A Service Expression takes the behaviours specified in a Service Genre and then binds them to particular technologies. For example, a Service Expression for ‘Search’ might define the SRW protocol for the transport, and IMS Metadata as the payload. A Service Expression thus defines *how* a service should provide the functionality (that was specified in the Service Genre).

The Service Expression is primarily a tool for promoting interoperability between services. This is because a Service Expression specifies a precise interface, in a human readable language, for a service. A large part of the e-Framework’s mission is to promote interoperability via the use of open standards. Often, the open standards will specify data models or service models that include optional elements¹ or commands² in order to facilitate interoperability between systems. There is nothing wrong with this *per se*, although optional components to a specification can lead to problems with interoperability.

In the world of data standards (where the standardised component is data which conforms to a specification), communities have taken to creating *Application Profiles*³ that more precisely specify the data payload (for example, by the removal of optional elements, or the specification of multiplicities, or of controlled vocabularies

¹ For example, IMS Metadata, which contains many optional elements.

² For example, OAI-PMH Harvesting, which specifies many different data payloads may be used.

etc). Service Expressions are analogous to Application Profiles – they are almost ‘Service Profiles’ – detailing all of the information that a requester needs to know about a specific service interface. The e-Framework asserts that interoperability is only possible with this level of detail.

Again, the Service Expression does not specify what a service does, only how the service does it.

The following extracts are taken from the Harvest Resource OAI-PMH LOM WS Service Expression⁴.

This service expression is a specialization of the harvest service genre. It uses the OAI-PMH specification for harvesting metadata. All of the OAI-PMH operators (verbs) are supported. The resource (provider repository) being harvested exposes an interface for harvesting. Communications to the provider service implementation interface are represented in requests defined via web services, using SOAP over HTTP transport through the SOAP HTTP binding. Requests and (dissemination) results are communicated between the client/requestor and the provider service implementation using HTTP, with results encoded in XML. The service expression supports disseminating LOM v1.0 metadata from the resource.

As specified by OAI-PMH, the harvest resource service expression supports six distinct functions.

* ListMetadataFormats: Get the set of metadata formats that the repository is capable of providing as results for a harvest request. The repository may be capable of returning (disseminating) the metadata in different formats. This information enables a client to request metadata in a particular format

ListMetadataFormats (See OAI-PMH specification clause 4.4). The request and response SHALL be as specified. Additional requirements:

* The service implementation SHALL support dissemination of LOM v1.0. The description of the metadataFormat SHALL be

```
<metadataFormat>
  <metadataPrefix>lom</metadataPrefix>
  <schema>http://ltsc.ieee.org/xsd/lomv1.0/lom.xsd</schema>
  <metadataNamespace>http://ltsc.ieee.org/xsd/LOM</metadataNamespace>
</metadataFormat>
```

The service expression interface SHALL conform to the FRED Profile for Core Service Standards

There is no WSDL currently defined as a standard for OAI-PMH: OAI-PMH is defined only as an HTTP-transport protocol. Absent an official WSDL, a FRED OAI-PMH WSDL SHALL be used. The FRED WSDL is based on the IVOA WSDL (<http://www.ivoa.net/internal/IVOA/RegistryInterface/RegistryHarvest-v1.0.wsdl>), and is hosted at [\[link\]](#). It has the namespace [\[link\]](#). Requests to OAI-PMH SHALL follow the format given in the WSDL.

2.3.1 Service Implementations and Service Instances

A Service Implementation is working code that implements a Service Expression. The implementation is not a part of the e-Framework, but would typically be referenced by it. The Service Implementation may be a code library or an executable.

A Service Instance is a deployed working service, built from the code in a Service Implementation, which exists at a particular end point for requesters to make use of. Again, Service Instances are not part of the e-Framework but are referenced by it.

2.3.2 Specifications and Standards

As mentioned earlier, part of the e-Framework message is about the promotion of the use of open standards and open specifications. The e-Framework does not detail individual specifications or standards, but it does reference them. Service Expressions may add further refinements to a specification or standards; they may also define precise message formats for some of the commands with the specification.

2.4 Service Usage Models (SUMs)

Figure 2-2 provides schematic views of Service Usage Models (SUMs). There are two varieties of SUM: Genre based and Expression based.

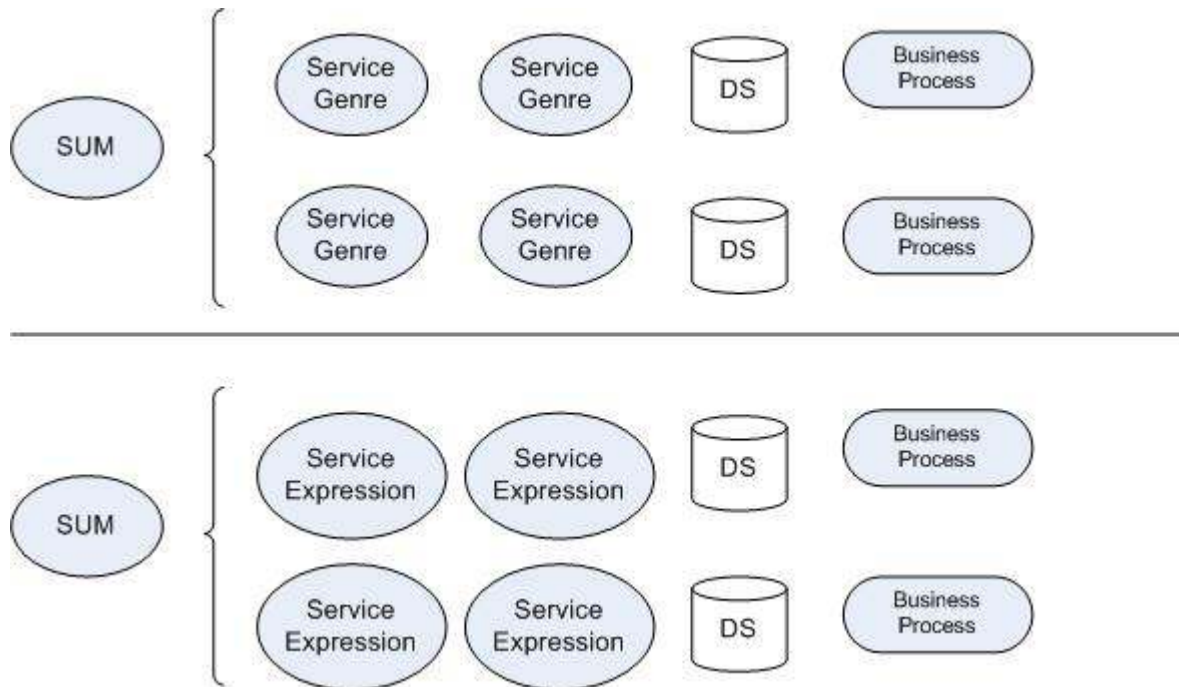


Figure 2-2: Service Usage Models

Service Usage Models are designed to give context to the Service Expressions and Service Genres. SUMs take specific Genres or Expressions, and then use them as technical components to support a specific business process. Thus, a SUM consists of a set of summaries of business processes, the Service Genres (or Expressions) that are involved in each stage of those business processes, and the data sources upon which the services operate.

SUMs are described in template documents in the same manner as the other e-Framework components. The approach used in constructing SUMs is to determine first the overall functionality required by the business process. Thus a business process may be broken into one or more functions. Each function is defined in terms of input, process and output. Once the core set of functions has been defined, each function is then implemented in terms of the Service Genres or Service Expressions. Thus for a given function, the SUM will define the e-Framework components used, the interactions between them, and the data that is processed.

A Genre-level SUM might be used by a business analyst or similar individual looking to break a business process into service-oriented concepts. A Genre-level SUM could be used to bridge communications between software developers and business analysts. An Expression-level SUM, which explains precise interfaces, would be used by software developers as a blueprint towards building a system.

The SUM should be viewed as a bridging artefact – a bridge between the business world and the software services that will support it.

The following example “SUM diagram” (figure 2-3) is taken from the UK HE Admissions structured personal statement SUM³. The Diagram shows a very simple view of how Business Processes are decomposed and implemented by the use of Service Genres.

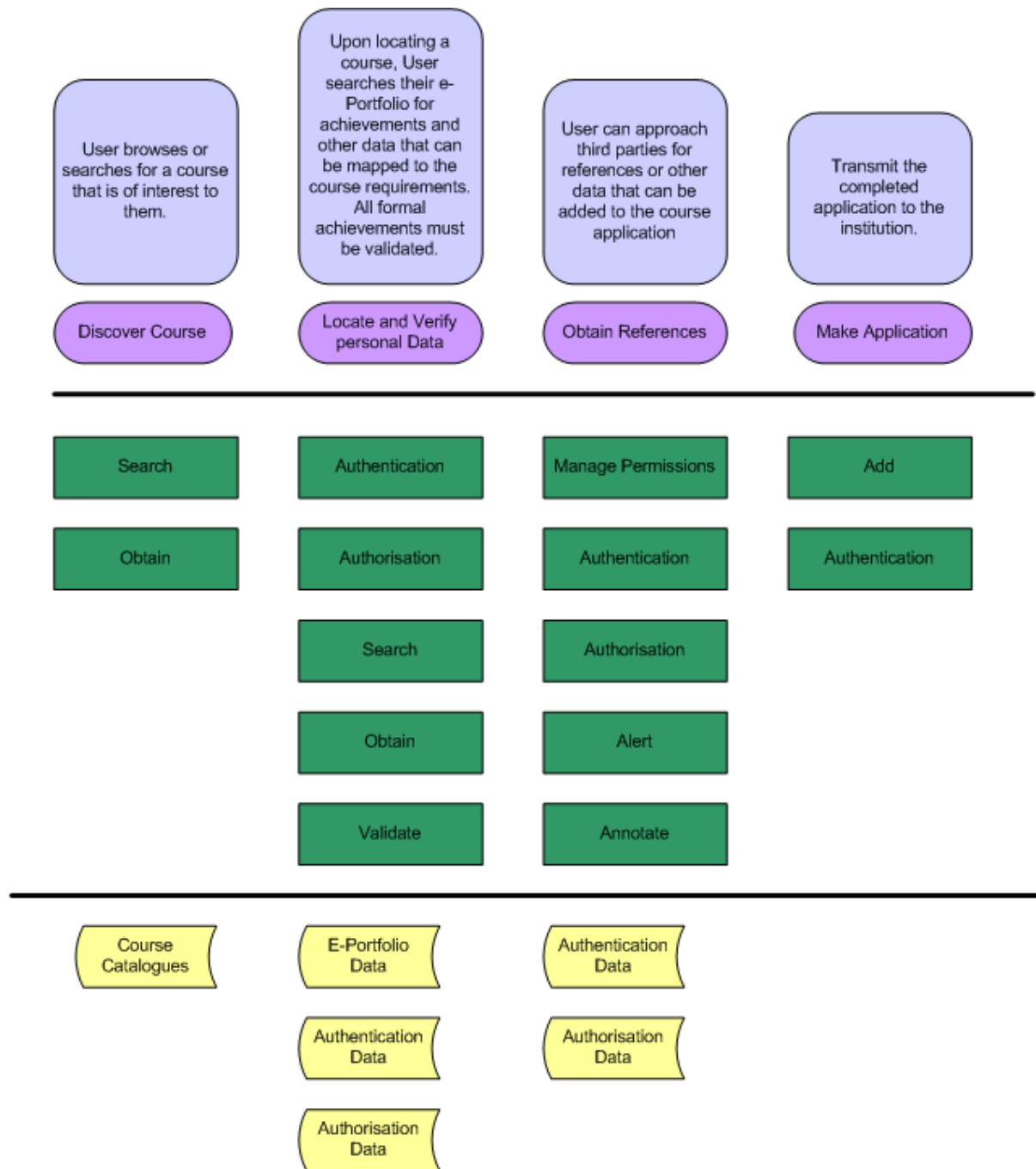


Figure 2-3: SUM Diagram for UK HE Admissions

Figure 2-3 shows the four key business processes (in purple); and then breaks down each process into a set of Service Genres (in green) that are used to support those processes. The rest of the documentation contains more detail on how the services are arranged. Data sources used by the Service Genres are also shown (in yellow).

Newcomers to the e-Framework often assume that SUMs are akin to software applications. It should be noted that a SUM is not a software application; rather it is the set of services that might be used by an application in order to support a business process. A SUM does not detail user interfaces or other matters that are important in a complete application.

3 ENTERPRISE ARCHITECTURES

3.1 Introduction

“A Comparison of the Top Four Enterprise Architecture Methodologies” by Sessions⁶ contains a very concise description of what Enterprise Architecture (EA) is, and what it is for. Sessions starts his paper by outlining two assertions:

- “Organisations [were] spending more and more money building IT systems”;
- “Organisations [were] finding it more and more difficult to keep those systems aligned to business need”

Sessions also notes that “Large organisations can no longer afford to ignore these problems.”

These three points sum up the space in which EA activity takes place. EA is primarily an exercise in modelling on a grand scale. Modelling an organisation’s structure, processes and operations; detailing the governance and change processes of an organisation; defining the current view of the organisation and the ideal view of the organisation; and defining how the organisation moves from where it is now to where it wants to be.

The core emphasis in EA work is this notion of ‘Business need’. Everything that is reflected within an EA is there precisely because it is vital to the Business. Where an EA shows differences between the current and ideal views of an organisation, projects are commenced to move the organisation from the current to the ideal. The literature notes that the language of EA borrows from engineering, specifically the building/engineering industry as exemplified in the notion of town planning. Sessions notes that a log cabin probably does not need an architect, whereas a skyscraper in New York City does. It seems likely (one of Sessions’ early points relates to ‘Large Organisations’) that EA will not be the province of smaller organisations – firstly the smaller business is probably not of sufficient complexity to require an EA; secondly, the huge modelling effort required would probably be beyond the resources of smaller organisations.

Proponents of Enterprise Architecture point to benefits which include improved decision making, optimisation of business processes, reduction of costs and so forth. At face value, it can be postulated that having thorough, accurate and in-depth detail on how an organisation functions is an asset in itself strategically; however, the cost of producing an Enterprise Architecture should not be underestimated.

Sessions provides a good history of the development of EA, primarily in terms of the release years of the various different methodologies (“Frameworks”) that can be used to develop the architecture. These Frameworks tend to be where the focus of the discussion on EA lies.

3.2 Enterprise Architecture Frameworks

EA Frameworks are methodologies, processes and procedures that enable an organisation to create an Enterprise Architecture. In 1987, Enterprise Architecture was born when Zachman⁷ wrote a paper in the IBM Systems Journal called “A Framework for Information Systems Architecture”. This paper detailed how the challenge for the future was (broadly) ensuring that IT Systems supported and aligned to the demands of the Business. Zachman’s Framework for Enterprise Architecture⁸ presents a grid-like view of the components of an Enterprise Architecture.

Ibrahim and Long⁹ identify a number of other Frameworks: NIST EA¹⁰, FEA¹¹, DoDAF¹², NASCIO¹³, FEAF¹⁴ and TEAF¹⁵. TOGAF¹⁶ is another.

The Frameworks are very good at showing the scale of the arena in which EA operates. That is, that EA ranges from Business Processes all the way down to the communications infrastructure (i.e. wires). Brown¹⁷ notes that ‘the elements of the enterprise architecture ... are incredibly interdependent’. Brown further notes that traditionally, organisations divided up IT across departments, developing silos which were very independent of each other. The Enterprise wide view, as depicted in EA, challenges this situation; even more strongly so when service orientation (see later) is brought in as well.

Ibrahim and Long also note that an Enterprise Architecture consists of more than just architecture, “in particular, EA includes architecture, governance and a roadmap”. The roadmap enables an Enterprise to move from the ‘current’ architecture to the ‘ideal’ architecture by way of a series of change enabling projects. Governance specifies the policies and authorities for any particular part of the EA, as well as the processes for introducing later change. This viewpoint is not contentious, although each framework expresses the viewpoint in its own manner.

Sessions contrasts four EA frameworks, and finds (perhaps unsurprisingly) that each of the evaluated frameworks score highly in different areas to each other. Sessions argues that a ‘blended’ approach is perhaps the best, taking parts of the various frameworks and using them together.

3.2.1 Components of an Enterprise Architecture

The components of an Enterprise Architecture (referred to as artefacts) are typically reports, models, analysis and design documents and so forth. The different EA ‘frameworks’ offer a range of processes, methods and insight to create and classify the correct set of artefacts to fully define the Enterprise. These artefacts are typically perceived as building blocks or pieces of a jigsaw. The Architecture defines how those pieces should fit together; and viewed as a whole can determine if any of the pieces need modification.

As stated previously, an organisation typically has a current architecture and a future (ideal/desired) architecture. Artefacts need to be created for both.

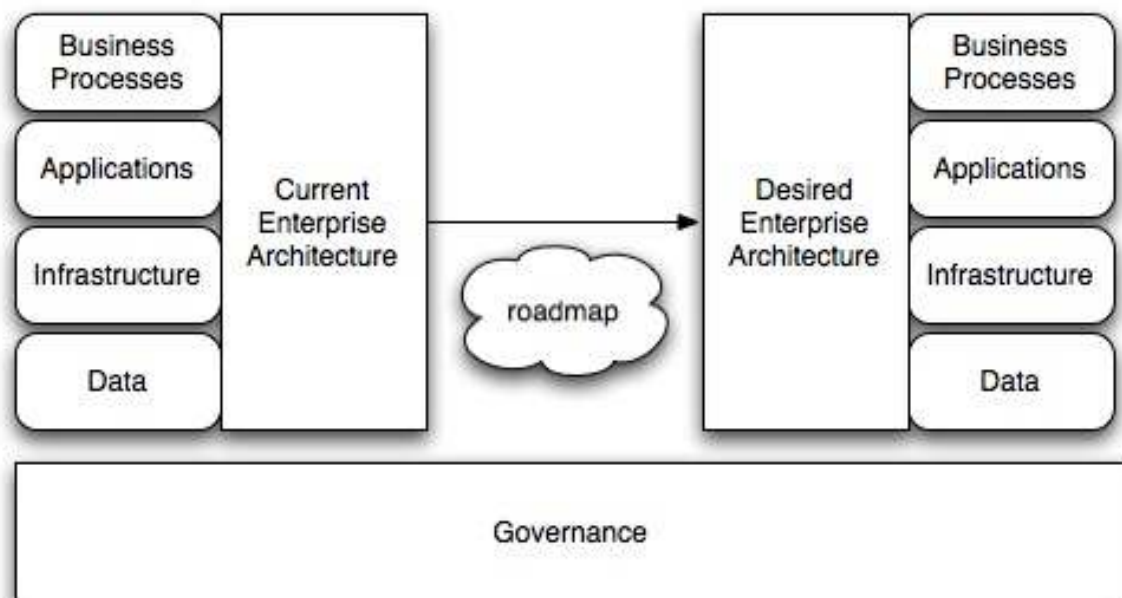


Figure 3-1- Enterprise Architecture Overview

Figure 3-1 shows the relationship between the various components of Enterprise Architecture. On the left hand is the current IT view of the enterprise, namely the business processes, applications, data and infrastructure architectures. There is a plan or roadmap in place to move to the Enterprise Architecture on the right hand side of the diagram (the desired architecture). All of this is underpinned by governance, which provides management processes, standards and oversight. The Governance models within EA tend to vary, but in essence different teams are given ownership of various parts of the Architecture. Within an organisation, there is likely to be overlap between the architecture and individual projects; and possibly between the EA and other enterprise wide initiatives (e.g. an organisation embracing SOA on a large scale). These issues of overlap of governance need to be clearly defined to avoid conflict within the organisation.

The components within each of the categories are typically created by having an Enterprise Architect (or team of analysts/architects lead by an Enterprise Architect) undertaking the analysis and modelling work for the organisation. This work is time consuming and needs to cover the whole of the organisation, gathering detail on

process and procedures, departments, structures, capabilities, customers, suppliers ... as guided by the EA framework being used. EA is therefore by definition enormous in scope.

Ibrahim and Long present the 'IBM EA-Framework' which breaks down the EA space into a stack of four 'domains'. These domains are also commonly seen in other EA frameworks, as shown in figure 3-2.

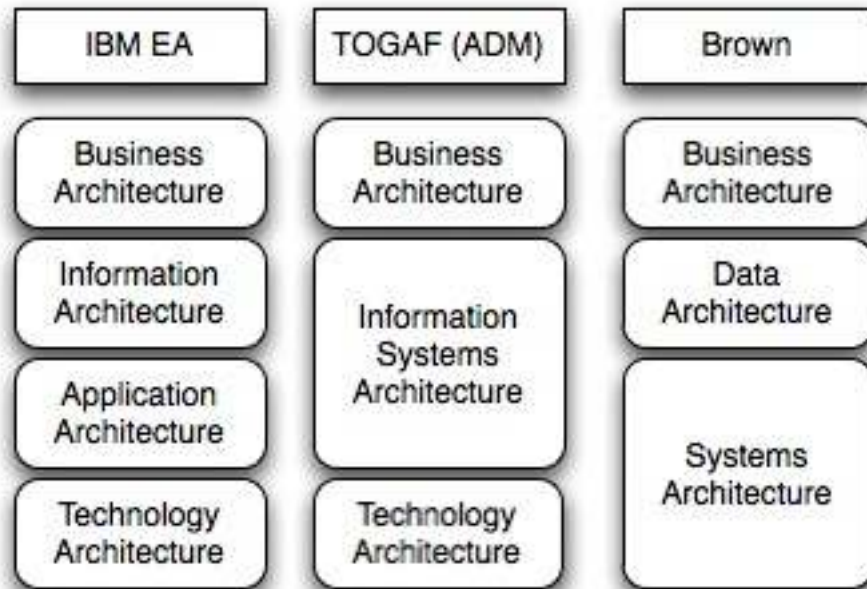


Figure 3-2: Different Views on EA

Figure 3-2 shows three different views on the constituents of EA. The first column shows the view as described by Ibrahim and Brown, that of:

- Business Architecture
- Information Architecture
- Application Architecture
- Technology Architecture

The second column presents the TOGAF view, which in this case makes use of phases B, C and D of the TOGAF Architecture Development Method (ADM). Here, there is an alignment of TOGAF Business Architecture phase B with the IBM Business Architecture. Similarly, there is alignment of TOGAF Technology Architecture (Phase D) with the IBM Technology Architecture. TOGAF Information Systems Architecture (Phase C), encompasses both IBM Information Architecture and Application Architecture. An interesting contrast is made with Brown's view which is focused on where software services (perhaps of the type described by the e-Framework) fit. Brown takes the view that the top level entities are Business Architecture (mapping to the IBM Business Architecture and TOGAF Phase B), Data Architecture (mapping to IBM information architecture and part of the TOGAF Phase C), and Systems Architecture, which consists of: Infrastructure (Technology) Architecture, Applications architecture, and Systems architecture.

The salient point from the Figure 3-2 is that although the frameworks have some slight differences between them, the broad message is much the same – that EA is about technology supporting the business needs. This is the same message as that of the international e-Framework; the following section looks at the e-Framework and Enterprise Architecture together.

4 COMPARISONS

This section now attempts to define the relationships between the e-Framework and the wider discipline of Enterprise Architecture.

4.1 Initial Impressions

The first point is that Enterprise Architecture is bigger in descriptive scope than the e-Framework. EA seeks to model *everything* for an organisation; the e-Framework is primarily describing service oriented software models. It should be noted that the e-Framework is seeking to model and document interoperable components. Such components may not belong to a single enterprise, but would be used by many enterprises.

There is also a question of detail. The e-Framework artefacts are very detailed on the precise interoperability and specification areas that they seek to address. Whilst it is certainly true that EA has a wide scope – without access to any example technical artefacts, it is difficult to assess if the levels of detail are similar.

Both EA and the e-Framework have similar goals – to define business and technical artefacts using a consistent vocabulary. EA frameworks such as TOGAF are not prescriptive about the choice of vocabulary, so there is good scope for the e-Framework to be included within an Enterprise Architecture.

Fortunately, the EA frameworks are decomposed into specific layers – and it is in those layers that the e-Framework components can begin to find themselves.

Figure 4-1 shows the relationship between the Enterprise Architecture layers and the e-Framework components. TOGAF has been used as an example here.

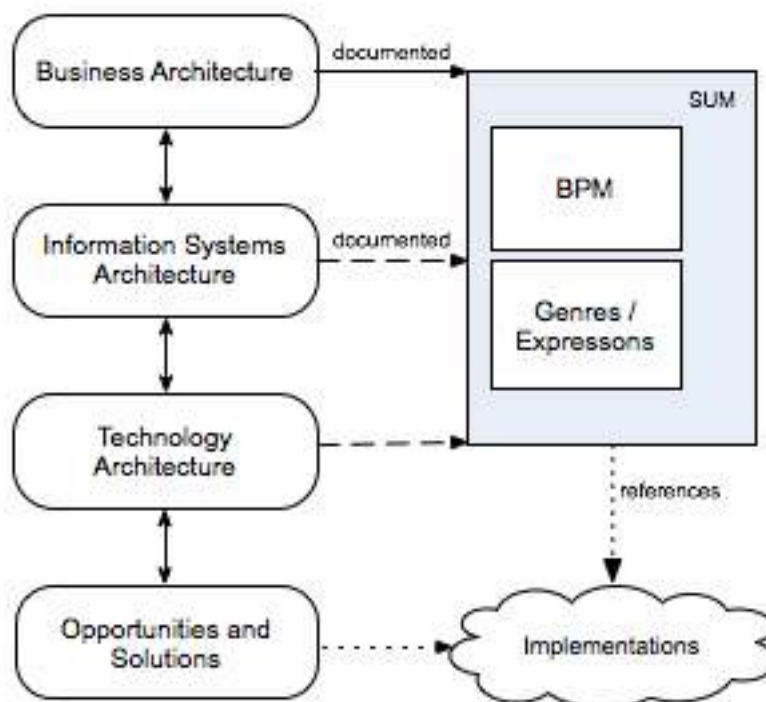


Figure 4-1: Relationship between e-Framework and Enterprise Architecture

Figure 4-1 builds on the previous assertion that a SUM is a bridging piece between the business world and the software world. Thus, the business process modelling portion of the SUM, those summaries of actual business processes, make up *part* of the business architecture. By no means is the set of the SUMs for an organisation the complete business architecture – as business architecture consists of more than just business processes. Certainly

though, a SUM may reference a complete Business Architecture – but a SUM itself would typically only document a part of a Business Architecture.

Similarly, SUMs provide a *part* of the Information Systems Architecture. Recall that IS Architecture consists of two main parts:

- Applications Architecture: There is a good analogue here to the SUM as a whole – a SUM can be regarded as the ‘back end’ of an application. It would be expected that a number of SUMs would be found within the Application Architecture.
- Data Architecture: Here the primary data types are defined. These are nouns (e.g. ‘invoice’ or ‘customer’ etc). The e-Framework primarily describes services, rather than data. Service Expressions do include an element of data definition. SUMs describe the passing of data between Service Genres or Expressions. In both cases, data is but typically defined in a standard or specification. There is no specific data construct in the e-Framework (although SUMs can be used – see later).

Beyond the IS Architecture is the Technology architecture, and here is where the e-Framework starts to reach its technical limit. Certainly part of the Technology Architecture is the list of software that will support the upper business processes; Genres and Expressions are definitely components in this layer as they describe specific services. However, the Technology Architecture also goes into details about hardware, middleware and IT governance, which is beyond what the SUM seeks to document.

Included for completeness is the implementation layer. In the EA world this is the ‘opportunities and solutions’ layer, which defines specific projects from the roadmap. In the e-Framework view, these projects would consist of Service Implementations and Service Instances – and would be referenced by the e-Framework.

4.2 Limitations

A recurring point regards the vastness of EA. EA is so huge that it practically envelops the entire e-Framework. However the e-Framework in its entirety cannot meet all of the demands of a particular architecture of an EA. Rather, the e-Framework spans parts of different architectures. This is not a weakness on the part of the e-Framework, rather it is a consequence of doing a comparison between two entities that are orders of magnitude different in terms of scope.

One of the key points of EA is that it identifies different groups of stakeholders and then provides views of the architecture to those different groups. The e-Framework does this also, but in a much more limited manner. The e-Framework has two key user groups – *business analysts* and *software developers*. The content of the e-Framework is geared towards these two groups. Less ‘technical’ audiences with interest in the domain, such as practitioners or other domain experts, are typically directed to other work outside of the e-Framework which exists at the business process/domain level.

The salient point is that whilst artefacts from the e-Framework are definitely of value to Enterprise Architects, the e-Framework itself is not a complete architecture that can simply be picked up and used. E-Framework artefacts will need to be supplemented with additional material if a complete Enterprise Architecture is required.

4.2.1 Horizontal behaviours vs. vertical stacks

Another potential tension between the e-Framework view and an EA view lies in the fundamental approaches that the two take. EA is primarily ‘noun’ driven where the e-Framework is more ‘verb’ driven. This introduces a subtle challenge for the architect when using the two together. A noun based approach has a good analogue with object oriented programming – a class is designed for a business object (e.g. ‘invoice’) and this class provides a range of operations that are possible on that object (e.g. ‘create invoice’, ‘calculate tax’ etc). Erl¹⁸ notes that in most enterprises, there are commonly two types of service:

1. Task Centric Business Services – which encapsulated the business logic specific to a task or business process.
2. Entity Centric Business Services – which encapsulate logic for an entity that exists within the business.

Erl further notes that of the two, it is the Entity Centric Services that are the most reusable, as they are business process agnostic. Reuse is one of the selling points of the service-oriented approach (although there is some anecdotal evidence coming to light now that re-use has been limited). As a result, enterprises are encouraged to create Entity Centric Services.

Figure 4-2 shows the entity centric view of services – remember, the core of the description here is on the ‘noun’ rather than the operations.

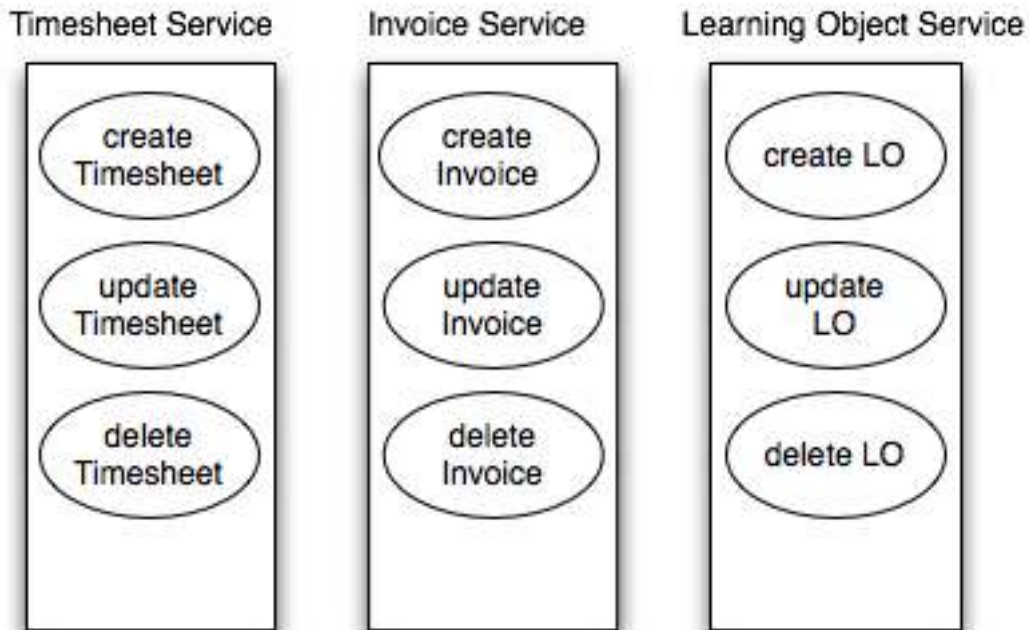


Figure 4-2: Entity Centric Services

The e-Framework takes a verb based approach, rather than a noun based approach. So the e-Framework defines behaviours – ‘Alert’, ‘Search’ etc, but these behaviours could apply to any service at all – there is perhaps an equal need to be able to search for ‘invoices’ as there is for ‘learning objects’. As a result, the e-Framework has an almost horizontal view of services. This is shown in figure 4-3.

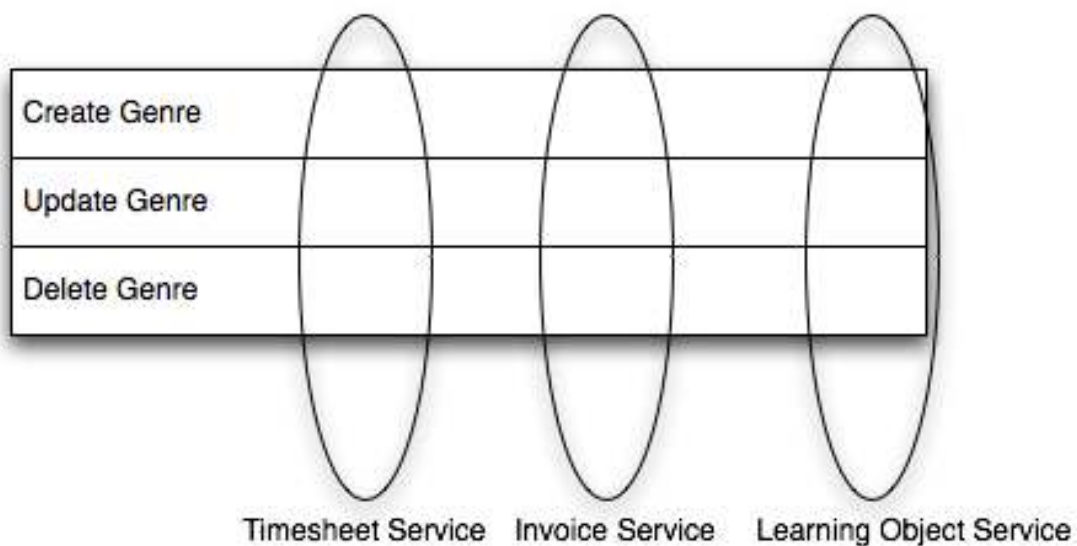


Figure 4-3: e-Framework Service View

This means that within the e-Framework, there is no such thing as an ‘invoice’ service; rather, there is a set of Service Genres that an invoice service may wish to implement. The analogue to the ‘invoice’ service is a SUM, which defines the set of Service Genres that will make up the ‘invoice’ service, along with the functions that the SUM will provide to requesters.

Note that these views are not incompatible; architects should be aware that the e-Framework uses a verb based modelling approach rather than a noun based approach. Thus architects using the e-Framework will primarily be working with SUMs.

4.2.2 Other issues

Ibrahim and Long note a number of other differences between SOA and EA, many of which are relevant to the e-Framework and EA:

- EA should not become totally focussed on services. Services are not always the best solution to a problem. Architects need to keep an eye on the other parts of EA in view whilst considering a service based approach to a particular problem.
- Developing e-Framework and EA artefacts in isolation from each other may result in a duplication of effort, if multiple teams are used. Architects may well focus on just the architecture, ignoring e-Framework contributions. Modellers may well be modelling business processes that already exist as EA artefacts.
- Architects will need to keep an eye Service Expressions, to ensure that any standards and specifications used in such Expressions are incorporated into the Architecture.

Ibrahim and Long also discuss governance issues. It is unlikely that an EA effort would directly affect the governance of the e-Framework. This is because the e-Framework is external to any organisation that would be undertaking EA work. However, the general points raised about governance should be borne in mind for a more successful project.

4.3 A Brief Note on FEA

Sessions’ comparison paper includes a section on the Federal Enterprise Architecture, which is the EA used by the US government. The enterprise in FEA is the US Government. Sessions describes how the FEA views things:

- FEA takes a ‘segment’ based view of the enterprise. There are two types of segment:
 - Core Mission Area Segments: one that is central to the mission or purpose of a political agency. (e.g. Health is a segment for a Health Authority).
 - Business Services Segment: one that is used by most political agencies (e.g. Payroll).
- There is also a notion of an Enterprise Service – and this is a Service that applies to the whole enterprise. (So, each agency has its own payroll (for example) – but all agencies are required to use the central Security Service).
- Services (and Segments) are not “Service Oriented” Services.
- FEA consists of five reference models which facilitate communication using shared vocabularies:
 - Business Reference Model: provides a business view of the various functions of the federal government.
 - Components Reference Model: provides a more IT based view of systems that support the business functionality.
 - Technical Reference Model: defines the specifications and standards used.

- Data Reference Model: defines ways of describing data.
- Performance Reference Model: defines ways of describing the value delivered by EA.

FEA is interesting because it has grown from the public sector. Within a UK context, it might be argued that if the Education Sector is viewed as the Enterprise, then different agencies (for example, universities, JISC, Becta, UCAS, etc) belong to that enterprise. Each agency will fulfil a number of core mission segments as well as making use of their own business services segments. Enterprise-wide services may apply to all of the agencies within the Enterprise.

Certainly the creation of a full model of UK education is an exhaustive and expensive task, but it may help to align the sector. Certainly, JISC may wish to consider looking at FEA in a more educational context on a smaller scale. Some of the themes that FEA addresses have occurred within e-Framework work.

FEA (as of 2007) was the newest EA.

5 CONCLUSIONS

Although Enterprise Architectures and the e-Framework address the same problem areas, they are fundamentally different creations. Enterprise Architecture is larger, and addresses the whole of the Enterprise. The e-Framework operates primarily at a technical level, describing services and the use of services. Also, an Enterprise Architecture is bound to one institution, where the e-Framework is intended to enable sharing between whole educational sectors.

Both EA and the e-Framework recognise the need to align technology to business, and both recognise that the use of a consistent vocabulary is essential to successfully communicating models and artefacts. Both the e-Framework and EA approach modelling via a top down methodology.

There are some differences in how EA and the e-Framework describe their technical artefacts. EA focuses very heavily on the use of Business Objects – and creates data based representations of them, supported by an application architecture which processes those business objects. The e-Framework is a more behaviour-based model – describing operations which can be applied across a wide range of different data types. The e-Framework's SUM construct bridges business processes to service genres or expressions.

An important notion in Enterprise Architecture is the notion of the roadmap, which provides a route from the current Architecture to a desired (future) Architecture. Such a roadmap defines the projects and other investments that are required to move the organisation forward. There is no direct analogue to this within the e-Framework because of its nature as sector-spanning, solution-sharing tool rather than an enterprise planning construct. It is recognised within the e-Framework that components will change over time.

The e-Framework artefacts and models can be used within Enterprise Architecture. The Architect needs to be mindful of where the boundaries of the e-Framework lie in relation to the relevant Enterprise Architecture being developed. The e-Framework touches Enterprise Architecture most strongly at the technological architecture levels, but there are also contact points in the Business Architectures and 'lower' Implementation Architectures.

An interesting point to consider is that the e-Framework has also been built as a dissemination tool. The e-Framework is perhaps more extroverted than an Enterprise Architecture – which by design covers the internals of an organisation (most likely a business). An Enterprise Architecture is probably something which an organisation would not share with others.

The US Government has invested in FEA – a public sector EA which is designed to treat the US government as an Enterprise. This EA has developed in a slightly different manner to the more traditional Enterprise Architectures, and this is worth further investigation. FEA may be particularly relevant to Domain Modellers.

In conclusion, both the e-Framework and EA provide a toolset for organisations. EA operates on a macro level, whereas the e-Framework operates on a micro level. It is important that architects and others select the correct tool for the job, and are mindful of the audience that they are communicating with. The e-Framework provides a consistent vocabulary to EA for describing a service based approach for technical artefacts. There are differences between the two on a business and technical level, but these differences can be overcome.

REFERENCES

- 1 <http://www.e-framework.org>
- 2 <http://www.e-framework.org/Default.aspx?tabid=844>
- 3 Heery, R. Patel, M. "Application Profiles: mixing and matching metadata schemas." <http://www.ariadne.ac.uk/issue25/app-profiles/>
- 4 <http://www.e-framework.org/Default.aspx?tabid=900>
- 5 <http://www.e-framework.org/Default.aspx?tabid=911>
- 6 Sessions, R. "A Comparison of the Top Four Enterprise-Architecture Methodologies." May 2007, MSDN <http://msdn.microsoft.com/en-us/library/bb466232.aspx>
- 7 Zachman, J.A. "A Framework for Information Systems Architecture." IBM Systems Journal, Volume 26, Number 3, 1987.
- 8 Zachman, J. "A Framework for Enterprise Architecture", Zachman International <http://www.zachmaninternational.com> 1986-2008
- 9 Ibrahim, M. Long, G. "Service Oriented Architecture and Enterprise Architecture." IBM developerworks 2007
- 10 <http://www.cio.gov/Documents/fedarch1.pdf>
- 11 <http://www.whitehouse.gov/omb/egov/a-2-EAModelsNEW2.html>
- 12 http://www.defenselink.mil/cio-nii/docs/DoDAF_v1_Volume_I.pdf
- 13 http://www.nascio.org/advocacy/washWatch/documents/whitepaper_07-03EnterpriseArchitecture.pdf
- 14 <http://www.gao.gov/bestpractices/bpeaguide.pdf>
- 15 <http://www.eaframeworks.com/TEAF/teaf.doc>
- 16 <http://www.opengroup.org/togaf>
- 17 Brown, P. "Succeeding with SOA: Realizing Business Value Through Total Architecture." Addison Wesley April 2007, Chapter 8
- 18 Erl, T. "Service Oriented Architecture: Concepts; Technology and Design." Pearson Education, 2005