

## Structural and behavioural views

### *and the ambiguity of "Function"*

Graham Berrisford and Marc Lankhorst

### Introduction

This paper is the third in a series in which we intend to shed light on the possibilities of using the ArchiMate language with an architecture method (in particular with TOGAF) and inform you of some challenges you may face. In the first paper of the series, "Using ArchiMate with an Architecture Method" [1], we outlined some of the fundamental ideas behind ArchiMate and TOGAF and posed nine questions to ask of a method to be used in conjunction with it. The second, "Using ArchiMate with TOGAF" [2], answered these questions specifically for TOGAF.

This paper is something of an aside - it contains some of the background discussion. We hope it will help you understand where we are coming from. We hope it will help us clear some debris out of the way before we turn to a detailed entity-by-entity analysis of ArchiMate and TOGAF meta models in the next paper.

While we were discussing the content of the first two papers, we discovered different interpretations of the terms and concepts we were writing about. We explored in particular how people use that most ambiguous of terms - Function. And to do that, we had to explore the different ways we distinguish between structural and behavioural views of an enterprise or system.

The result was the discussion you will find edited into this paper. First, we will address this elusive notion of "function". From there, we will go deeper into the structure-behaviour distinction, the linguistic notions behind ArchiMate, various notions of abstraction and composition, and the subtly differing "business function" concepts of TOGAF and ArchiMate.

### "Function"

Graham: People use the term "Function" loosely. It can mean a subsystem or a procedure. It can mean a purpose, or a service or a component. And people use "Functionality" as an ugly plural in place of "Functions".

Marc: Function is indeed one of the most troublesome concepts. ArchiMate and TOGAF define them thus:

ArchiMate's Function	TOGAF's Function
unit of internal behaviour that groups behaviour according to e.g. required skills, knowledge, resources, etc., and can be performed by a single role.	Delivers business capabilities closely aligned to an organization, but not necessarily explicitly governed by the organization. Also referred to as "business function".

Graham: I'm uncomfortable about aligning a Function to one role in a definition, though I do see that is possible and desirable in some cases.

Marc: I should clarify the wording of the standard. It means a Function should be assignable

(as a whole) to one role. But the same Function can be performed by other roles as well. There is not necessarily a one-to-one correspondence between Functions and roles. What it means mostly is that a Function is an integrated set of work.

In ArchiMate, a Business Function is not an active structural element. It is a set of behaviour we assign to one or more active structural elements (say, organisation units).

Graham: I feel uncomfortable about that too! Defining a Function as a unit of behaviour makes it sound like a procedure. In TOGAF, a Business Function is building block of a Function decomposition structure – which is similar in appearance to an organisation structure.

Marc: It could well be that ArchiMate and TOGAF draw the structure-behaviour dividing line in different places.

## The structure-behaviour distinction

Graham: TOGAF doesn't speak of structure and behaviour. However, ArchiMate's distinction between structure and behaviour is different from the distinction made in various methods that early TOGAF authors surely took for granted, such as Information Engineering. (I don't recall if learnt what follows from that source, or from the enterprise architecture method taught by Bob Jarvis.)

The challenge was and is how to build coherent models of systems. It does not matter whether the system is a human activity system (business) or computer activity system (application). It does not matter whether the system is a physical working system, or only a logical specification of a system.

You'll know for sure what a step-by-step procedure is. And you'll know that a subsystem is both a subdivision of the overall system, and a grouping of elementary activities within the system. This concept of a subsystem covers every kind of component (module, class, package, application, organisation unit or Business Function) that we define to group behaviour, to perform activities, or respond to service requests.

The wider system requires components to connect and cooperate. These connections give a structure to the wider system. So every component is both an element of the system structure and a package of behaviour.

The classical position is that behavioural and structural models are *orthogonal views of the same system specification*. This table contrasts the two views.

BEHAVIOURAL views	STRUCTURAL views
Name and connect the SUBPROCEDURES in a PROCEDURE	Name and connect the SUBSYSTEMS in a SYSTEM
E.g. a state chart a process flow chart a use case definition, an interaction or sequence diagram.	E.g. a class diagram a data flow diagram a hardware configuration diagram a data model (passive structure)
Show time-ordered control flow. Have a start and an end. Often have a main path and alternative paths.	Name subsystems and their inter-connections. Do not show time-ordered control flow.

You can decompose both system structures and process flows to several levels of granularity (though people rarely have the time and energy to manage more than two or three levels). If you complete both structural and behavioural views, then at the bottom level of decomposition, *the same elementary activities must appear in both views*.

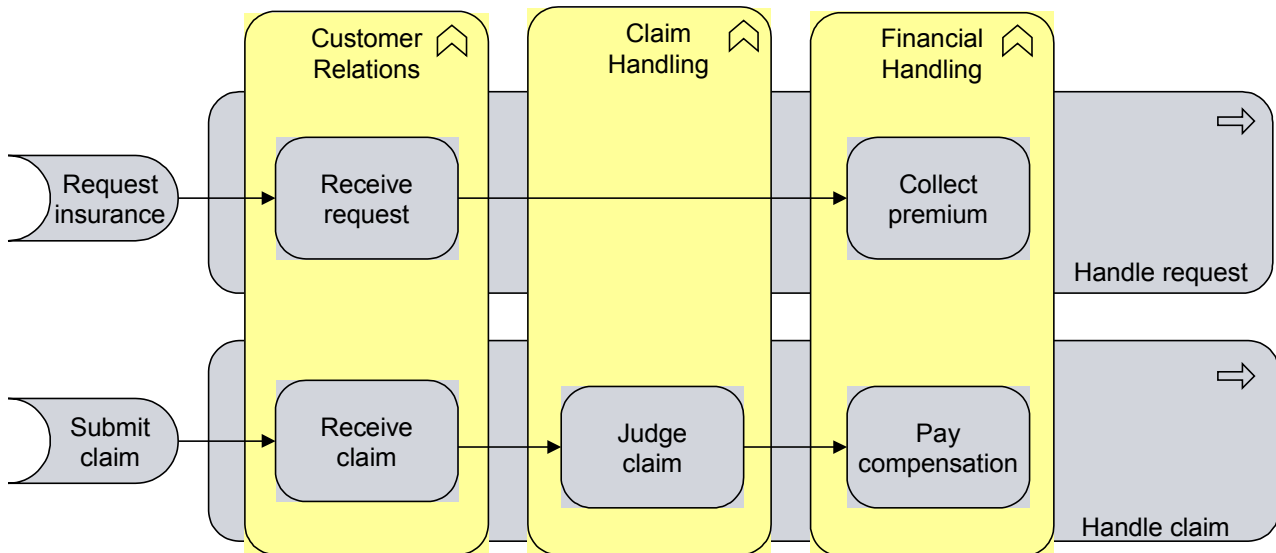
You might model the same application component (logical or physical) as an element in both the models in this table:

**Orthogonal views featuring application components**

BEHAVIOURAL model of a PROCESS	STRUCTURAL model of a SYSTEM
e.g. a UML sequence diagram	e.g. a data flow diagram
a behavioural model with a time flow.	a structural model with no time flow

Note that while the *whole* application appears as component in the system model, only *one activity* of the named application is involved in the process model (usually).

Marc: Your description of orthogonal views relates to the contrast ArchiMate makes between processes (horizontal) and Functions (vertical) in this picture.



Graham: Yes. Classical structured analysis would say a row shows a procedure composed of activities, and a column shows a subsystem into which activities are grouped according to expertise, location, or whatever. So I agree the rows in your picture are behavioural, but I'd say the columns are structural. The cells show elementary activities which feature in both views (elementary is not an absolute term, only relative to the model we have built).

A vertical column	A structural model. Shows the activities grouped in a subsystem. An assembly.	Clusters activities by role, location, or any cohesion rule you like. Not in a sequence.
A horizontal row		
A behavioural (time flow) model	Activity →	→ Activity
Shows the activities in a procedure	Activity →	→Activity
Connects activities in sequence	Activity →	→Activity
Under a logic control flow	Activity →	→Activity

At the bottom level of decomposition, an *elementary* activity is both an elementary procedure and an elementary Function. Every *elementary* activity appears as an element in both behavioural and structural views (assuming we fully define both views). But higher level groups of activities are columns, do not appear in behavioural rows.

Marc: ArchiMate places Functions in the behavioural view only. To us, it is obvious that a Function specifies behaviour, and is therefore in the behavioural view

Graham: A COBOL module, a Java class, an operating system, an application, an organisation unit and a Business Function are all specifications of behaviour. But they do not appear as units in behavioural models. Instead, they appear as units in structural models - in class diagrams, hardware configuration diagrams, organisation charts, Function decomposition structures. For that reason, they are (by me and others I believe) naturally called structural elements - in contrast to elements in behavioural models.

As your picture shows, what I call structural and behavioural models join at the bottom level of decomposition, where an *elementary* Business Function IS BOTH a structural element and a behavioural element. But all higher level groupings of these elementary Business Functions are groupings we invent when we design system structures.

Marc: Again, we do draw the structure-behaviour distinction in a different place. However, this is not of practical importance to modellers, since they can use the concepts regardless of where we choose to draw a line between structure and behaviour. Only we and language designers need to be concerned with this.

Graham: Don't forget those who have to read definitions of terms that contain the words structure and behaviour!

## On what is required and what is designed

Graham: It seems to me that:

- Processes and behaviour are required by our customers.
- Components and structure are designed by us (designers).

An end-to-end process delivers output a customer wants. We can of course refactor the way the process steps are distributed between components in our structure. But we can't remove the process or change its end result without changing our customer's requirement.

By way of contrast, we can redesign the structural model to our hearts content. We can completely remove even a top-level Business Function like "HR" and reassign all its component activities to Business Functions with other foci. That is because HR is a component of a structure that we created by design (presumably by using some kind of cohesion rule, like employee expertise). It was not required by our customer - its whole existence is matter over we - the system designers - have control. We organise the business structure to optimise our costs.

## Subject-verb-object distinctions

Marc: You are using sequential flow to distinguish behaviour from structure. In ArchiMate, the structure-behaviour distinction is derived from the subject-verb distinction.

Graham: I've not understood that to be more than an analogy.

Marc: Let me try an example: If X offers a service (at any layer), X has to do something to realise that service: that is X's internal behaviour, the "verb".

Graham: Agreed. The subsystem (structural element) has to execute one or more processes (internal behavioural elements) in order to deliver a service (external behavioural element).

Marc: X has to have a "place" to offer that service: that is the interface, the external structure.

Graham: Agreed. The subsystem should expose its services to its clients.

Marc: And X itself is the internal active structural entity, the "subject".

Graham: Agreed. Yes indeed. So now replace all X by "business function" in your sentences above. You are saying a business function is a structural entity – contrary to ArchiMate.

Marc: Only in the way you use the term. I continue to apply ArchiMate's meaning of structure and behaviour. Business functions contain behaviour specifications, and are therefore behavioural.

Graham: People used to teach data (entity-relationship) modelling based on the noun-verb distinction. It works in toy classroom examples. But every event can be named in either noun or verb form. And many words are ambiguous: is "order" a noun or a verb? Is order an event, an entity or a relationship? Ultimately, every event may have to be treated as an entity; every atom of behaviour may have to be recorded as an atom of passive structure.

Marc: I am thinking less about categories of words and more about sentence structure. All hu-

man languages have sentences consisting of subjects, verbs and objects; only their order differs. So we have languages with an SVO grammatical structure like English, SOV like Hindi, or VSO like Arabic. The ArchiMate active structure - behaviour - passive structure subdivision is directly derived from this subject - verb - object distinction.

Graham: Interesting. Of course, subjects and objects are interchangeable in natural language. An object in one sentence can be a subject in another. Also, a verb phrase can be used as a noun, as a subject or an object. Perhaps that explains some of the difficulties we have in forming a meta model in which architectural entities are uniquely defined and classified?

Marc: I agree. Most architecture frameworks and languages try to classify the entities they use to model the world, but the class of an entity depends on the context in which the entity occurs. We end up with a classification that works perhaps in 80% of the cases. That may be enough for practical use, but some things remain that you cannot easily classify and model that way.

## Transformation steps in architecture methods

Marc: There is a *behaviour-to-structure* step. We decompose a composite Function into its activities, then allocate each behavioural activity to one or more structural components.

Graham: If the ArchiMate language includes or implies transformation steps that you'd expect to find in a CBD or SOA method, then it does have at least a few implications for an architecture method.

Considering other transformations, we have discussed two related ways to turn a more idealised logical specification into a more physical specification:

- Realisation 1. Define an interface (a group of externally required services) then allocate each service to one or internal components.
- Realisation 2: Define a logical component (a group of activities) then allocate each activity of the logical component to one or more physical components.

The second process re-allocates behaviour from components in a structure of logical building blocks (say Business Functions) to components in a structure of physical building blocks (say organisation units).

Another common process thread starts with the business and ends with IT, which is course the way we try to ensure the IT supports the business rather than becoming an end in itself.

The table below shows these four transformation steps in ArchiMate and TOGAF.

Requirement-to-solution process threads	ArchiMate	TOGAF
Realisation (1)	External to Internal	
Realisation (2)		Logical to Physical
Construction	Behaviour to Structure	Behaviour to Structure
Business to IT	Business, Apps, Infrastructure	Business, Data, Apps, Technology

Each process thread tends to move us from requirement to solution, and to turn a more abstract specification into a more concrete, more physical, more real specification. By the way, do we agree what abstraction means?

## Aside on abstraction in general

Marc: I agree with the four dimensions in your paper [6]: idealisation, generalisation, composition and omission of detail. Although some might argue that omission is either idealisation or composition.

Graham: Omission means simply erasing details. Yes, this often yields a higher-level idealisa-

tion or generalisation. But it may instead be done in a way that leaves a slice (a view) at the same level of detail. The dimensions are not mutually exclusive.

Marc: In mathematics "dimensions" are orthogonal, mutually exclusive, axes. So I'd call these "types" or "kinds" of abstraction.

Graham: I use the term dimension because that's how methodologists (TOGAF, Zachman, Schekkerman and others) present the rows and columns of their two-dimensional frameworks! They present them as being orthogonal.

Marc: Where would you put a black-box/grey-box/white-box axis in your four dimensions?

Graham: Encapsulation combines the other basic kinds of abstraction. A capsule or component is an abstraction by all of:

- omission of detail (information is hidden within the component)
- composition (several services or activities are grouped to form the component)
- idealisation (the interface of required services is more idealised than the activities designed inside the component to provide those services)
- generalisation (the interface can be implemented by different components, or different internal designs of one component).

## On composition in particular

Graham: The picture earlier suggests a flat model, with one level of process and one level of Function, each composed of activities. Do you in practice model a deeper hierarchy than that?

Marc: Yes. We nest both Functions and processes. The number of levels and granularity depend on what enterprise/system we are describing, and why.

Graham: Enterprise, solution and software architects work at different levels of abstraction by composition/encapsulation. There could be anything from (say) 4 to 8 levels of granularity. The table below presents six levels at which people might draw models.

Six levels of granularity in models of processes and systems		
	BEHAVIOURAL / PROCESS view	STRUCTURAL / SYSTEM view
1	Value Stream (end to end)	Broad Business Function
2	Business Process	Narrow Business Function
3	Application Use Case (HCI)	Application
4	Coarse-grained service	Application Layer (front-end, back-end))
5	Fine-grained service	Application Component
6	Operation or Method	Module or Class

Marc: Your six levels are usable for our discussion.

Graham: What the ISEB [7] call a "solution-to-build" or a "project-ready architecture" has to be completed to around level 4.

Marc: Similarly, what the DYA approach calls a "project start architecture" has to be completed to around level 4. I see enterprise architects working work down to level 4, being concerned with high-level services and application layers. That is the level where enterprise architects interact with software architects.

Graham: I see enterprise architects working at levels 1,2,3 and software architects working at levels 4,5,6. I see also solution architects working at any level from 2 to 5 - working as a bridge that spans enterprise architects and software architects.

Marc: ArchiMate is used at levels 1 to 4. You can go on to address level 5 to a certain extent, although not at the level of detail a UML spec would provide.

## Terms used differently in business and application layers

Graham: It seems to me that using the term Function as a catch-all for everything that specifies behaviour distracts people from the separation they should make between the components of a structural view and processes in a behavioural view.

Marc: We use Function in the sense of activities that are performed at any level of granularity, be it a "Business Function" or "Application Function".

Graham: I hear people using those two terms in two contrasting senses. The use Business Function to mean a unit in a Functional decomposition hierarchy – a component of a structural view. They use Application Function to mean a use case - a process in a behavioural view.

Kind of function	In common parlance
A <b>business function</b> in the business layer is	A structural element: a <b>component</b> of a hierarchical business structure
An <b>application function</b> in the application layer is	A behavioural element: a <b>process</b> or use case.

People commonly confuse system decomposition with process decomposition. They use the term Function in a way that swaps from one side of the structure-behaviour dividing line to another as they decompose systems and processes through the architecture layers.

Decomposition	Structural / System View	Behavioural / Process View
	"BUSINESS FUNCTION"	"APPLICATION FUNCTION"
1	Broad Business Function	
2	Narrow Business Function	
3		Application Function (HCI use case)
4		Application Function (Service use case)

For me, ArchiMate entangles components with processes in its behavioural view.

Marc: ArchiMate takes a slightly different view of what is called "behaviour", but like in your view it also distinguishes between a functional decomposition of the behaviour (or what you call "process decomposition", i.e., a decomposition of what an entity is supposed to *do*, starting from the services it should deliver) and the constructional decomposition of the structural entities (your system decomposition, i.e., how the system is *built* from its constituent parts). The confusion is probably more in the various uses of the word 'function' than in the notions behind it.

## Mapping Business Functions to organisation units

Graham: Let us take HR as an example of a Business Function. It is a set of activities that must be performed or implemented by one or more organisation units. The relationship is captured, in TOGAF and other methods, in an organisation-to-Function matrix.

Functions	Function 1	Function 2 HR	Function 3
Organisation 1	*	*	
Organisation 2		*	*
Organisation 3		*	

Marc: Yes, your HR Business Function might consist of sub-Functions such as taking care of skills development, personnel administration, coaching, performance evaluation, etc. This overall HR Function is performed partly by the HR department and partly by line management.

Graham: So ArchiMate and TOGAF use the term Business Function in much the same way. We define a Business Function like HR as a set of activities to be performed. We allocate those activities to one or more organisation units.

Marc: You can build and use reference models for specific types of organisations (e.g. insurance companies) that show the essential Functions they need to perform. Mapping that onto a specific organisational structure by assigning responsibilities then makes it operational. In fact, I have seen several organisations using such generic Business Function models to derive their own 'operating model', and I have been involved in helping them develop such models myself.

## Business Functions as abstractions of organisation units

Graham: In classical structured analysis, Business Functions are logical abstractions of organisation units. They are more generic and more idealised than those organisation units which are given a manager, given goals, and given the Function's activities to perform.

Marc: A Business Function describes what an organisation does, not who does it.

Graham: Yes, that is one way in which the Function structure is more abstract than the organisation structure.

Marc: An organisation unit can perform or be responsible for one or more Business Functions (or parts thereof). A Business Function is one way of organising the activities of an organisation. E.g. according necessary expertise. This is not the same as a Business Function being an abstraction of an organisation unit.

Graham: I don't think the abstraction idea need trouble you here. Suppose you tried to replace the columns in your picture (shown above) by organisation units? Your difficulty would be that one activity could appear in several columns. In other words:

- The organisation structure is redundant in the sense that one activity can be assigned to two or more organisation units.
- The Business Function structure is non-redundant, since each activity is assigned to only one Function.

This is another way in which the Business Function structure is an abstraction (by idealisation) of the organisation structure.

The table below shows the essence of how this logical-physical distinction appears in the TOGAF meta model. Remember I am trying here to present TOGAF in a more systematic way than it presents itself!

	Logical		Physical
Layer	Behavioural	Structural	
Business	Business Service	Business Function	Organisation Unit
Application	Information System Service	Logical Application Component	Physical Application Component
Technology	Platform Service	Logical Technology Component	Physical Technology Component

## Aside on ISEB reference model definitions

The ISEB reference model [7] isn't perfect, but it does at least recognise some of the ambiguities we have discussed.

<b>Function</b>	<p>1: A component that offers one or more services.</p> <p>Or 2: A process that delivers a single service; which can involve many components.</p> <p>Or 3: A purpose or goal of a component or process.</p> <p>This reference model eschews the term Function except within the term Business Function.</p>
-----------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Other relevant terms are defined as below.

<b>Enterprise</b>	<p>A business or organisation with common goals and budget, in the public or private sector. A part of the real world that is directed and controlled by a management board to some human purposes, in which players cooperate to meet goals and objectives.</p> <p>Usually the highest level of an organization, spanning several relatively distinct organizations.</p>
<b>Business</b>	<p>An enterprise that offers services to its customers, usually in return for payment or funding. Business subdivisions offer services to each other.</p> <p>Usually a synonym of an enterprise or organisation. A business is usually distributed between organisation units.</p>
<b>Organisation</b>	<p>Usually a synonym for an enterprise or business. Often a synonym for the management hierarchy of an enterprise or business.</p>
<b>Organisation Unit</b>	<p>A physical subdivision of an enterprise's business. Usually managed by one manager. Sometimes located in one place. Organisation units are decomposed to bottom level (elementary) organisation units.</p>
<b>Management Hierarchy</b>	<p>A hierarchy showing the reporting line from each bottom-level manager to the management board. Usually in close correspondence to an organisation unit hierarchy, and to a top-down cascade of business objectives.</p>
<b>Business Function</b>	<p>A logical subdivision a business system. A business component or building block that performs processes and offers business services. Business Functions are decomposed to bottom level (elementary) Business Functions.</p>
<b>Core Business Functions</b>	<p>A Business Function that differentiates one business from others, if only in the manner that the processes are performed.</p>
<b>Support Business Function</b>	<p>An infrastructure Business Function (such as personnel, procurement and finance) that is similar to that in other businesses. An obvious candidate to be out-sourced and/or delegated to a "shared service".</p> <p>(Analogous at this level of definition to a reusable application component, but very different in practice.)</p>
<b>Business capability</b>	<p>A loosely-defined concept; usually a synonym of Business Function or business service; usually broad.</p> <p>E.g. legal compliance or customer service.</p> <p>Sometimes an aim of an organisation. Sometimes a Function that requires cross-organisational management.</p>

## Concluding remarks

Graham: That's surely enough on Function, and enough for this paper? I've posted a presentation called "What is a Function" [6] to explain what I believe the traditional view to be. I think we'll do well now to move on - in our next paper - to discuss the many architectural entities

other than Function!

Marc: I'd say the core of your statement is this: Processes and behaviour are required by our customers. Components and structure are designed by us. I fully agree. It's just a matter of us using the same term differently. You use "function" to designate units of structure, we use it for specifying functionality/behaviour needed to realise services. Our "bottom-level" is the assignment of elementary "verbs" to elementary "subjects", yours the join of two types of grouping the same bottom-level actions.

But let's not spend more time on this. Again, as in the previous papers in this series, we have uncovered several subtle but important discrepancies and ambiguities beneath the superficial similarities between ArchiMate and TOGAF. Nevertheless, our readers should not be deterred by these subtleties from using both in conjunction. Rather, they should be aware of them as they develop your own architectures and be clear on the notions of abstraction the definitions of the concepts they use. The developers of ArchiMate and TOGAF should ensure that in future versions of both methods, these wrinkles are ironed out as much as possible.

## References

- [1] Graham Berrisford & Marc Lankhorst, "Using ArchiMate with an Architecture Method", *Via Nova Architectura*, June 2009. <http://www.via-nova-architectura.org/magazine/magazine/using-archimate-with-an-architecture-method.html>
- [2] Graham Berrisford & Marc Lankhorst, "Using ArchiMate with TOGAF – Part 1: Answers to nine general questions about methods", *Via Nova Architectura*, June 2009. <http://www.via-nova-architectura.org/artikelen/tijdschrift/using-archimate-with-togaf.html>
- [3] The Open Group, *ArchiMate® 1.0 Specification*. Van Haren Publishing, 2009. <http://www.opengroup.org/archimate>, <http://www.opengroup.org/bookstore/catalog/c091.htm>,
- [4] The Open Group, *TOGAF™ Version 9*. Van Haren Publishing, 2009. <http://www.opengroup.org/togaf>
- [5] Marc Lankhorst, Hans van Drunen, "Enterprise Architecture Development and Modelling – Combining TOGAF and ArchiMate", *Via Nova Architectura*, 21 March 2007. <http://www.via-nova-architectura.org/magazine/magazine/enterprise-architecture-development-and-mode.html>
- [6] Graham Berrisford, Papers accessible at <http://avancier.co.uk>, in particular:  
"What is a Function?"  
"Architecture Meta Models"  
Papers on "Abstraction" in the "Library".
- [7] British Computer Society, *Reference model for ISEB certificates in enterprise and solution architecture (v11.3)*, BCS, 2009. <http://www.bcs.org/upload/pdf/reference-model-enterprise-solution-architecture.pdf>

## Copyright

This paper was edited from conversations between Marc Lankhorst (group leader Service Architectures at [Novay](#) and a founder and teacher of [ArchiMate®](#) [3]) and Graham Berrisford (a [TOGAF™](#) 9 [4] instructor for [Architecting-the-Enterprise](#) and author of the papers in [6]).

### **Creative Commons Attribution-No Derivative Works Licence 2.0**

No Derivative Works: You may copy, distribute, display only complete and verbatim copies of this document, not derivative works based upon it.

Attribution: You may copy, distribute and display this copyrighted work only if you clearly credit the authors Graham Berrisford of [Avancier Limited](#) and Marc Lankhorst of [Novay](#) before the start and include this footnote at the end.

For more information about the licence, see <http://creativecommons.org>

[ArchiMate®](#) and [TOGAF™](#) are registered trademarks of The Open Group.

Graham Berrisford

[graham.berrisford@architecting-the-enterprise.com](mailto:graham.berrisford@architecting-the-enterprise.com)

Marc Lankhorst

[marc.lankhorst@novay.nl](mailto:marc.lankhorst@novay.nl)