The functions & capabilities of activity systems

"Your paper hits a number of nails squarely on the head."

This paper addresses the description of what activity systems do, especially but not only business systems. The basic idea is that the essential architecture of a business activity system can be defined thus:

- **Business Services** (deliver value via output, obtain supplies via input)
- **Business Interfaces** (channels for external entities to access services)
- **Business Processes** (sequences of activities)
- **Business Components** (units, roles, people and machines that perform activities).

Some say a system has "functions" or its ugly sister "functionality". Some speak of "capabilities". What do they mean? If enterprise architects speak of **Business Functions** and **Business Capabilities**, then they should understand how those concepts relate to the four more basic concepts, and make sure that meaning is shared by all those who might read their architecture descriptions.

This paper also compares and contrasts three descriptions of architecture description. In doing this it addresses questions such as: What general principles and concepts underpin architecture frameworks? What is the difference between logical and physical? How can I square ArchiMate with TOGAF?

Contents

A glossary is not enough

The architects of activity systems use words loosely in everyday conversation. You might hear one say:

- The Function of a restaurant is to feed diners (its mission).
- A Function of a restaurant is to sell menu items (its services).
- A Function of a restaurant is to cook a meal (a process), or even…
- A Function of a restaurant is to meet health and safety regulations.

We can collect words used by the architects of activity systems into a glossary. But glossaries suffer from three problems.

1. Words have fans. The enthusiast's definition of a word looks OK on its own. But it turns out you cannot understand how it relates to other words. There are synonyms and homonyms, super types and subtypes, elementary and composite terms.

2. There is redundancy. If economical = efficient + effective, then do we need "economical"? Might the glossary be more efficient and effective without it?

3. Examples are nominal and/or vacuous. They look OK at first sight. But it turns out they have not been tested to ensure they truly distinguish between concepts. One example can be plausible in many concept definitions. Consider the Armed Forces as a system. It might be said of this system that:

- Its *Mission* is defence of the nation.
- Its *Aim* is defence of the nation.
- Its *Function* is defence of the nation.
- Its *Capability* is defence of the nation.
- The *Service* provided is defence of the nation.
- Its *Activity, Work or Job* is defence of the nation.

A glossary of words architects commonly use would be interesting, but huge and incoherent. We might refer to the glossary in real life, but it would be a very poor basis for an examination in architecture concepts. Architects need something better.

In a *really* useful glossary, the words would label distinct concepts in a coherent body of knowledge. We want a reference model that defines *concepts* that architects (not stakeholders) ought to understand. It should distinguish concepts from each other, relate them in a coherent whole, distinguish base entities from subtypes of them, and distinguish atomic entities from composite views of those entities.

The concepts have to be named. There will always be arguments about the right words to use. But the really important thing is to ensure the concepts are distinct.

Encapsulation and nesting of systems

An activity system transforms inputs into outputs. To external entities, the purpose and value of the system is in its outputs. To agree the scope of a system (at any level of granularity), you can define its boundary by inputs consumed and outputs delivered.

- These might be shown in a context diagram.

Systems can be nested. One person's system is another's subsystem. A cruise ship contains a restaurant, which contains a kitchen, which contains an oven. Each of these nested entities can be viewed as an encapsulated activity system. The same principles must be applicable to systems large and small; else architects would have no repeatable methodology. (Systems can also overlap.)

A principle of Component-Based Design (CBD) and Service-Oriented Architecture (SOA) is to define outputs as the results of **services** provided to external entities.

- A service can be documented in a service contract (name, input, output, rules and non-functional requirements).

The services an external entity is allowed to consume can be defined an **interface,** separable from the internal components of the system

- This might be documented in some kind of menu, or service catalogue or directory, or service level agreement.

Open the box of an encapsulated system and you see the inter-connected **components** that transform inputs into outputs.

- This internal structure may be shown as inter-related subsystems in some kind of goods/service/data flow diagram.

You can also watch the **processes** that transform inputs into outputs.

- This internal behaviour - these processes – may be listed in some kind of process map or use case diagram. The steps and flow of each process can be charted.

Decomposition

System decomposition (into smaller subsystems or components) is a tool of design. Process decomposition (into shorter steps) is a tool of requirements analysis. Both kinds of decomposition have the effect of dividing external services (requested by external entities) into sub-ordinate internal services (and perhaps also dividing longer external interfaces into shorter internal interfaces).

Composition

The reverse – composition of fine-grained processes and components into coarser-grained processes and systems - is a tool for managing complexity. This tool is essential to any enterprise architect who hopes to describe a whole enterprise as a system – which is indeed an implication in TOGAF.

For more on this topic: http://avancier.co.uk. > Library > Architecture Concepts > "Granularity"

Four base concepts in system description

This paper addresses the activity of a system (not architecture precursors: business drivers, strategies, goals, stakeholder concerns, principles, regulations etc.).

The basic idea is that, since a system delivers value to customers by transforming inputs into outputs, the essential architecture of a system can be defined using four base concepts. A restaurant provides a simplistic way to illustrate this idea.

System	**Restaurant** (perhaps a subsystem of a Hotel or a Cruise Ship)
Mission	Feed diners.
Services	Hot dinners. Cold snacks. Drinks.
Processes	Order. Cook. Serve.
Components	Waiter. Chef. Oven.
Interfaces	A la carte menu. Table d'hote menu. Today's specials.

Every activity system can be described as a set of inter-related components which execute processes to deliver services, ideally through pre-defined interfaces. For coherence, these four base concepts should be defined *in relation to each other.*

Concept	**Definition**
Service	Something a customer wants* from the system, or the system wants from a supplier. A result of processes, but defined in a service contract without reference to the internal logic of processes used.
Process	What happens. A logical sequence of activities that ends up delivering a service at some level of granularity. Executed by components.
Component	A subsystem that does work, executes activities. A cohesive group of related but distinctly invokable activities, encapsulated behind an interface.
Interface	A channel for a consumer to access a list of services. A facade that hides the workings of internal processes and components.

For more on this topic: http://avancier.co.uk. > Library > Architecture Concepts > "Modularity"

Note that the elementary particles of an activity system are indivisible actions and materials. And in an information system, the particles are executable instructions and data items. The focus here is on the atoms and molecules made from those elementary particles.

Aside: * What about waste? A system may produce products and by-products. Some by-products can be sold (are wanted and requested via services). Some by products are pure waste (are unwanted, and never requested via services). If a system produces waste, then it may be stored forever inside the system, or may be output from the system. Either way, it is not well described as a wanted output of any service requested by an external entity. Waste might better be documented as a *side effect* of services (that is, in the post-conditions of service contracts). Ideally, the cost of storing or exporting the waste produced by each service should be quantified in its service contract, and this should be accounted for in the price of that service. Not doing this (as in old nuclear power stations) leads to an over-optimistic system design and under pricing of services.

Structural and behavioural dimensions of description

In activity systems, things do work by performing actions on other things. In UML, the things and activities are categorised as structural and behavioural, along the lines shown below.

Structural (persistent things)	Behavioural (transient activities)
Entities	Events
Components	Processes
Interfaces	Services

ArchiMate divides materials between processors (active structure) and processed (passive structure). The next table shows some ways to articulate this distinction.

Active Structure	Behaviour	Passive Structure
Actors	Act on	Stages
Machines	Consume	Materials
Computers	Process	Data
People	Read	Books

Three asides: This paper does not address passive structure (materials or data). A component can be active in one activity and passive in another. And don't assume the active-passive distinction corresponds to the subject-object distinction in grammar: as the following counter examples show.

Passive Subjects	Verb	Active Objects
Materials	Feed	Machines
Service Contracts	Encapsulate	Processes
Books	Help	People

For more on this topic: http://avancier.co.uk. > Library > "ArchiMate and TOGAF"

External and internal levels of description

The scope or boundary of a system is a subjective decision made by one or more observers. One person's external entity is another's internal component.

But once a system boundary has been decided for design purposes, the external-internal distinction can be used (as in ArchiMate) to organise our four base concepts in a two-by-two overarching model, thus.

ArchiMate's overarching model of activity system concepts		
	Behavioural	Structural
External	Service	Interface
Internal	Process	Component

This overarching model is applied later in "Relating TOGAF to ArchiMate to ISEB ".

After the requirements of a system are crystallised by defining the Services it should offer, then mapping external to internal is sometimes called *realisation*, and mapping behaviour to structure is sometimes called *construction*.

For more on this topic: http://avancier.co.uk. > Methodology > "Underpinning concepts"

Logical and physical levels of description

TOGAF makes much of the distinction between logical components (architecture building blocks) and physical components (solution building blocks), both of which are defined by the services they offer.

TOGAF's (implicit) overarching model of activity system concepts		
Physical components	**Logical components**	**Services**

A **component** is an encapsulated subsystem, a processing unit, bounded by the services it offers. A **logical component** is an ideal, candidate or potential component; it is defined by external services and internal processes; it is organisation, vendor and technology neutral. A **physical component** is real; it can do work; it has the physical resources it needs; it is organisation, vendor or technology specific.

TOGAF proposes you define services and logical components in phases B, C and D of the method. You then map the logical components to physical components in phase E, where you choose either to buy or build the physical components.

E.g. suppose your sales Organisation unit require 20 services from a candidate logical component. It could be a human activity system, but you envisage it as candidate computer application. You call it a customer relationship management system.

Then, to realise that logical component, you choose a specific physical application (say Salesforce.com) because it offers 18 of the 20 of the required services. It also offers 5 other services you never thought to ask for, which are "opportunities".

Aside: don't forget to consider the data, lest it turn out the new system stores the same data already stored by other systems belonging to other Organisation units – leading to disintegrity issues and application integration challenges.

TOGAF applies the same overarching model to three activity system "domains".

Entities **Domains**	**Physical components**	**Logical components**	**Services**
Business	Organisation units	Business functions	Business services.
Applications	Physical application components	Logical application components	I.S services
Technology	Physical technology components	Logical technology components	Platform services

TOGAF even applies the same pattern to the passive structures of data (though data event or enquiry would be better here than data entity).

Data	Physical data components	Logical data components	Data entities

TOGAF's (implicit) overarching model is different from ArchiMate's overarching model. In ArchiMate and elsewhere, logical means external requirements expressed as services, and physical means the internal realisation of the external requirements as processes executed by components. Some TOGAF authors appear to mean that also.

For more on this topic: http://avancier.co.uk. > Library > Architecture Concepts > "Logicality"

Categories and domains of activity system

People do think and speak separately about:

- Human activity systems
- Data processing (information and application) systems
- Computer or IT platform systems
- Other kinds of technological, biological or chemical system.

Dividing an **enterprise activity system** into different categories of subsystem helps us to divide a massively complex system description into more manageable elements.

This table illustrates a possible separation of concerns, akin to that in TOGAF.

Entity class Subsystem	Component	Process	Service
Human	Business Function	Business Process	Business Service
Applications	Application	Use Case	I.S Service
Infrastructure	Platform technology	(known only to Vendor)	Platform Service

Note that the lines between these domains are blurred, and can be crossed. Before computers, data processing systems were business functions. Since the 1960s, the data processing clerks were gradually replaced, as business functions were turned into automated information systems, or applications.

- Today's application was yesterday's business function
- Today's platform technology was yesterday's generic application
- Today's data structure was yesterday's hard-coded procedure.

And one system may span several domains. E.g. SAP (love them or hate them) sell an *enterprise system*. It contains physical application components, data components and technology components. And in so far as it automates business processes, you might say contains some elements of business architecture also.

Underpinning ideas are not always explicit

The idea that an application is an automated business function used to be commonplace. It is still implicit in TOGAF, but, like much of the body of knowledge TOGAF was built on, it is no longer widely understood.

TOGAF does say explicitly that an IS (aka Application) Service *is a* Business Service, but leaves implicit that an Application Component *is a* Business Function.

Whereas ArchiMate has an explicit overarching model (of Service, Component, Process and Interface), TOGAF's overarching model (of Service, Logical Component and Physical Component) is revealed only by analysis.

The functions and capabilities of activity systems

Business Function as logical Component

In TOGAF, a Business Function is a bounded unit of business capability. It is an encapsulated component that delivers business services - a container of business processes (or partial processes) that are logically related, and for which you can define the skills needed.

Successive composition of elementary functions into composite functions produces a Business Function hierarchy - a logical composition structure - a structural model akin to a management hierarchy, but devoid of managers or resources. A bottom-level Business Function may be a role, which may or may not be supported by IT. If it is a purely clerical role (does nothing but data processing) then it might be automatable as a software application.

Organisation unit as physical Component

When you allocate managers and resources to execute the processes of a Business Function, then you have something new, an Organisation. To make an enterprise work, you can devise various management hierarchies. You may divide the Organisation by customer, by location, by product (or indeed by Business Function).

But you still have your logical Business Services, Business Functions and roles, which should be more stable than the physical Organisation structure.

For more on this topic: http://avancier.co.uk. > Methodology > "Business Architecture Rationalisation".

Capability conceived as broad Business Function

The idea of defining a Business Function as an organisation-independent unit of a business is very old. The concept of Capability is a relative newcomer, which has become fashionable through cross-organisational capability-based planning. But does the new word add a new concept?

In TOGAF, a Capability is a macro-level Business Function. In the TOGAF meta model, Capability is not related to other architectural entities - surely because to connect it would reveal it has all the same relationships as a Business Function.

Capability realised as Organisation

A colleague says: "I think of a Capability as a collection of skills, resources, processes and systems, brought together from one or more parts of the enterprise or extended enterprise, to meet a business outcome."

OK. So you set out to develop an enterprise architecture (EA) Capability. You define an EA Business Function, along with the processes, skills and roles required. Then, to realise it, you must add objectives, a budget, a manager, and employees with the skills needed to play roles and perform processes. So, the result (your managed EA team) has all the properties you would associate with an Organisation unit.

Capability-based planning is a process that starts with a high-level Business Function, then extends it for implementation to become an Organisation. The Organisation for a

Capability is likely to be cross-organisational. It may also be temporary. But it is still an Organisation, with all the properties you would expect an Organisation to have.

Do we need Capability?

Sometimes Capability is a *discrete entity*, as in the TOGAF meta model. But this discrete entity seems indistinguishable from a high-level Business Function – one that straddles organisation units in different lines of business. Smetimes Capability is a *composite view* spanning entities including Business Function and Organisation. Until we agree which more atomic entities a Capability is composed of, it remains a fluffy management consulting concept, not great material for examination questions.

Capability as in "capability maturity model" might be the same thing, or might be different

How the ISEB model maps TOGAF to ArchiMate

The ISEB reference model (for its certificates in Enterprise and Solution Architecture) was not built as a creative act, or new architecture framework. It extracts the essence of various frameworks (CBD, SOA, ArchiMate, TOGAF, Business Motivation Model, etc.) and generalises them. At the start are two tables that relate TOGAF to ArchiMate. The tables correspond, so Service and Component have the same meaning in both tables.

	Behaviour	Structure
External	Service	Interface
Internal	Process	Component

This reference model is based the notion that these four concepts can be applied in each of three architecture domains, shown here as a layered architecture domain hierarchy.

Business	Business Services	Services
	Business Functions	Components
Information Systems	Information System Services	Services
	Applications	Components
Technology Infrastructure	Platform Services	Services
	Technologies	Components

The first table shows the 4 base concepts in ArchiMate's overarching model. The term Function is eschewed because it gets confused with TOGAF's use of the term.

The second table shows the 6 core architectural entities in the TOGAF meta model. The separation of Service from Component is fundamental. TOGAF uses Service and Component in all four domains, and uses them in pretty much the same way as ArchiMate does.

The wording between the two tables would be clearer as: "This reference model is based on the notion (used in ArchiMate) that these same four concepts appear in each of three architecture domains. This can be illustrated by showing how TOGAF gives two of the concepts (Service and Component) more specific names in each domain."

Relating TOGAF to ArchiMate to ISEB

This paper compares and contrasts three descriptions of architecture description, by reference to underpinning ideas drawn from system theory, and by reference to three overarching (meta meta) models, presented for comparison below.

This table shows the essence of TOGAF's meta model of activity system description.

Essence of TOGAF meta model				
Entity class Domain	**Component**	**Process**	**Service**	**Interface**
Business	Business Function & Role Organisation Unit & Actor	Business Process	Business Service	-
Application	Logical App. Component Physical App. Component	-	I.S. Service	-
Technology	Logical Tech. Component Physical Tech. Component	-	Platform Service	-

Notice TOGAF doesn't use the concept of Interface in any domain. And it uses the concept of Process only in the business domain. Note also TOGAF divides components into logical components (architecture building blocks) and physical components (solution building blocks).

This table shows the essence of ArchiMate's meta model of system description.

Essence of ArchiMate meta model				
Entity class Layer	**Component**	**Process**	**Service**	**Interface**
Business	Role & Actor	Business Process / Function	Business Service	Business Interface
Application	Application Component	Application Function	Application Service	Application Interface
Infrastructure	Node & Device & System Software	-	Infrastructure Service	Infrastructure Interface

Notice ArchiMate does not address the internal processes of infrastructure components, which are the concern of the technology vendor.

The ISEB reference model implies the four base concepts in ArchiMate's meta model should appear in each architecture domain, and uses common-place and not-too-ambiguous terms for those concepts where they appear.

This table shows the essence of the ISEB reference model (at least, in this interpretation for comparison with the tables above).

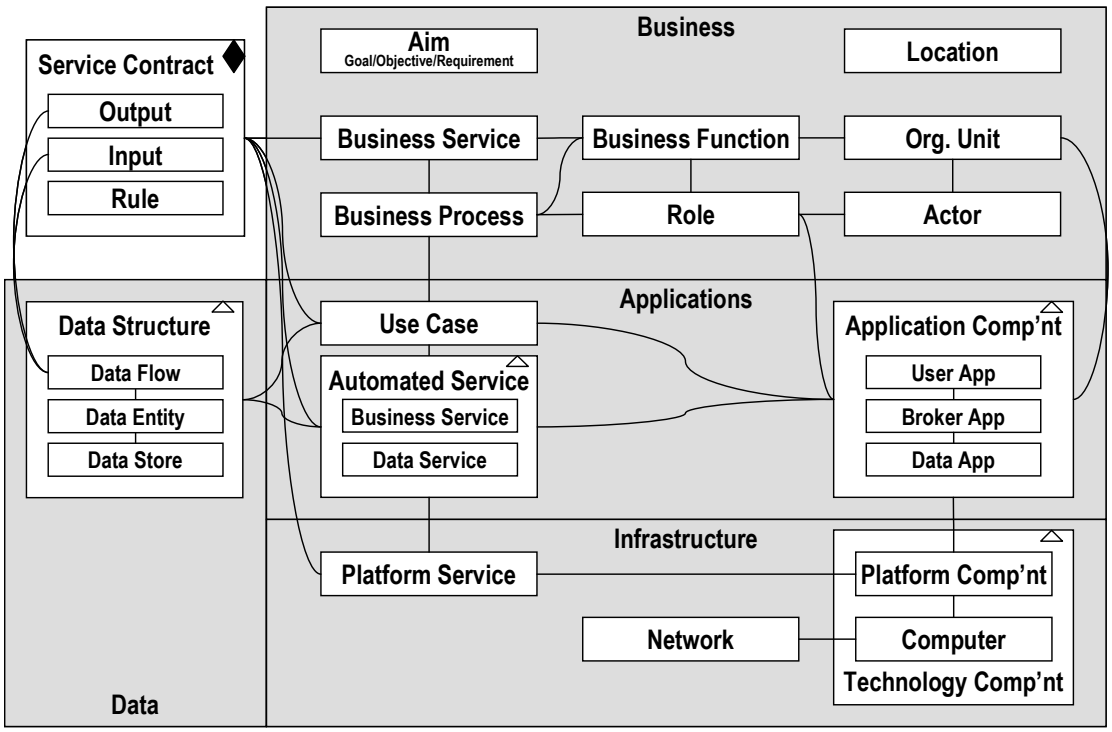
Essence of ISEB reference model (an interpretation)				
Entity class Layer	**Component**	**Process**	**Service**	**Interface**
Business	Organisation Unit & Actor Business Function & Role	Business Process	Business Service	Business Channel and SLA
Applications	Application	Use Case (main & other paths)	Use Case (service contract)	User Interface & Menu
Software	Application Component	Automated Service (server-side software)	Automated Service (service contract)	Service Directory & Catalogue
Infrastructure	Technology	-	Platform Service	API

Entities and domains in the ISEB reference model

The ISEB reference model (for its certificates in Enterprise and Solution Architecture) is not a general-purpose glossary.

For a copy of the ISEB reference model, find "PDF downloads" at http://www.bcs.org/category/10521

Some entries in the ISEB reference model can reasonably be regarded as core or atomic architectural entities. If there was a meta model, it might look like this.



For more on this topic: http://avancier.co.uk. > Methodology > "Architecture Meta Models"

Not all relationships can be shown. The bigger enclosing boxes are either composites or super types (with a white triangle). Use case, automated service and platform service embrace both a service and the highest-level process that delivers the service (though the nested services and processes are detailed separately).

Conclusions and remarks

To understand EA frameworks it helps to know that most have some roots in ways to model business processes and business data, techniques used in the analysis and design of business data processing systems. They also have some roots in ideas underpinning CBD and SOA, which became fashionable in the 1990s.

In fact, it is probably impossible to understand most EA frameworks unless you understand something of the roots above, notably:

- Systems and subsystems (components) can and should be encapsulated behind services and interfaces.
- Components and processes are different things - at right angles to each other.
- Components and processes are recursively composable and decomposable.

People who aren't interested in the application of these ideas surely won't feel comfortable with the engineering details of an EA framework like TOGAF, or with ArchiMate. And that will include most business stakeholders!

We can and do divide activity systems into different kinds and domains, and speak separately about them. There are very important differences to be understood. But there are also similarities that architects should understand, even if their stakeholders don't. All are "activity systems". People use the same basic ideas to describe all activity systems, just different words.

> The work to turn business data processing systems into software proved to be very complex and difficult. Software procedures must be precise and unambiguous to a much greater degree than clerical procedures. In formalising their art, software architects learnt to separate services from processes, and separate interfaces from components. Yet business architects also make similar separations in human activity systems. What is a call centre but an interface, a channel to services delivered by back office components?

Perhaps architects describing activity systems need four levels of vocabulary, say:

- For all kinds of system: a simple overarching model with only 4 concepts.
- For the traditional four enterprise architecture domains: a more complex four-part meta model with a vocabulary of 40 words.
- A structured reference model that helps architects to use the meta model in solving practical problems, and in examinations. Say 400 words.
- A general-purpose glossary containing words architects may have to use with stakeholders. Say 4,000 words.

The ISEB reference model is intended to help architects understand the basic ideas of activity systems, so they can manipulate them effectively in different contexts. It has to use words that a) examiners can use without ambiguity, and b) do not depart too far from popular approaches like ArchiMate, TOGAF and BMM. It is not an aim that architects use the same vocabulary in talking to stakeholders.

Communication with stakeholders is another matter altogether.

Aside on Function

Business Function has a strong place in the ISEB reference model, but "this reference model eschews the term function except within the term business function". In other words, Function it is not a core architectural entity or an examinable concept.

Remember the reference model is not meant to be a general-purpose glossary. It is to define distinct concepts that an architect ought to understand. The purpose of defining Function more forcefully or formally *ought to be* to add a new concept that could reasonably appear in an examination question.

In conversation, one might say a Function of a restaurant is to feed diners (a mission), to sell menu items (its services), to cook a meal (a process), or even to meet health and safety regulations.

Curiously, a Business Function is another thing altogether. It groups activities of the business into a logical component by some cohesion criteria. This seems the *least* natural language interpretation of the word.

So oddly, the technique of "functional decomposition" is not process decomposition (as you might think) it is *component decomposition*. (If you haven't twigged yet that there are two different approaches to system decomposition, then you probably haven't yet got the distinction between a process and a component.)

Consider what Function means in the following sources:

- In Information Engineering, a Business Function is a logical subdivision of an enterprise, which encapsulates some required activities or processes.
- In TOGAF's Business Architecture, a Business Function is the same, a logical component which offers business services and encapsulates processes.
- In TOGAF's TRM, the term function means a platform service that can be invoked in current technologies, but is not yet encapsulated behind a properly-defined service contract (or API).
- In SSADM (an old UK methodology), a function definition is what people call a "use case" today. In ArchiMate also: an application function is probably best defined as a use case. A use case is a process (in a system) that is used by a user; it is defined in terms of a process flow wrapped up in a service contract.
- In mathematics and in programming languages, a function often corresponds to a process – sometimes exposed for use as an external service, sometimes hidden as a private internal process.

In every day conversation, we can and do use Function to mean a purpose, service or process, and sometimes other things. And we sometimes use it to mean a composite of two or more distinguishable concepts.

We can speak freely of Functions (aka Functionality) *because* the term is so flexible. The listener does not need to read the term as meaning something precise, and because we never meant it to be precise, that is OK.

On the other hand, ambiguity can lead to misinterpretation, and it makes Function a hostage to fortune in any meta model. Because Function is commonly used in various ways, it sits uncomfortably attached to any one concept in a meta model. And adding

the term can tend to obscure what can and should be clear distinctions between Service, Interface, Process and Component.

Perhaps we could define it thus: "Function is a loose term that is used to mean one or more of the following:

- The mission of a system
- What is wanted of a system; crystallisable as the output and/or post conditions of a required Service.
- What is done by a system; a Process that is recognisable by Service consumers.
- A subsystem; a logically-related group of activities. E.g. a Business Function is a logically-related group of activities with no manager or resources.

That would be overly elaborate. Perhaps the definition should be reduced to "The term Function may be used as label for a specific mission, aim, service, or process; or for a logical component (as in Business Function); or for a composite view that embraces two or more of those entities."

Until we agree which atomic entities a Function corresponds to or is composed of, it remains a fluffy concept, not great material for examination questions. Currently, the term adds noise – more for students to read and worry about – rather than a distinct concept. It could in fact be dropped from the model without any harm done.

Aside on functional requirements and services

There are many ways to capture the requirements of a system. We might document a requirements catalogue, a data model, some business process models and some use case definitions. Their content will overlap and tend to duplicate information.

A requirements catalogue is commonly divided into functional and non-functional requirements. (Many technical infrastructure architects resent this, since the non-functional are all they care about - are to them functional.)

A system use case is a process in which system users are engaged to some end – some purpose - some goal – to deliver some kind of service. A use case definition template describes a process flow and wraps it up inside a service contract.

Many requirements can be, perhaps should be, captured in service contracts.

- Functional requirements appear in services' signatures and semantics (the input and output, and the rules by which input is transformed into output).
- Non-functional requirements appear as qualitative measures of those services.

Functional in this context is vaguely related the everyman use of Function above.

Asides on ArchiMate

Is Business Function a structural or behavioural concept?

Generally speaking, persistent entities and components are called structural. Transient events and processes are called behavioural. But the line between behavioural and structural is not universally agreed.

A class in UML is an encapsulated collection of processes that deliver services. It is a container of processes. We might argue about special cases, but primarily, a class is a *component*. And a class diagram showing the structure in which those components are related is called a *structural* model in UML. It is a structure imposed over the activities of the software system.

- Yet in UML, a use case diagram which shows a system as an encapsulated collection of process is called a *behavioural* model. Surely a mistake! The diagram is a list or structure of processes, it is not a process model.

A Business Function in TOGAF is an encapsulated collection of business processes that deliver business services. It is a container of business processes. We might argue about special cases, but primarily, a Business Function is *component* (a logical one). So, a Business Function decomposition hierarchy is a *structural* model. It is a structure imposed over the activities of the enterprise system.

- Yet Business Function appears in the *behavioural* column of the ArchiMate meta model. Surely a mistake!

Is the term Function necessary in ArchiMate?

In TOGAF and elsewhere, a Business Function is a logical component, akin to an Organisation unit. ArchiMate places Business Process/Function where you might expect only business process.

In common use, an Application Function is a use case, a process. ArchiMate has Application Function where you might expect Application Process.

I believe it would be clearer if Business Process/Function was called just Business Process, and Application Function was called use case.

What is a Collaboration in ArchiMate?

In ArchiMate, the concept of collaboration appears to be structural - more persistent than a transient interaction between components (a process). Though it is presumably more transient than those persistent components that engage in the collaboration.

Is it an interface agreed by two components? Is that really a pair of interfaces? One for each component, which defines the services one component offers to the other? Or is it a data flow or other data structure, which is to be written by one component and read by the other?

Is it a contract between two persistent entities? Does it define the rules by which the two entities will interact over a period of time to perform some business processes? What does that tell us that defining each cooperative process would not tell us? Is it simply the list of those processes in which the two parties are both involved?

(And by the way, a real business contract may say very little about the business processes to be performed. Rather, it says a great deal about how one or other party will get out of the contract if the cooperation is deemed to be failing in some way!).

Asides on what TOGAF says

Enterprise architecture is about describing an enterprise as a system. What is the scope of the enterprise, the system, that you describe? That is a judgement call made at an early stage in any architecture development method.

A key output of TOGAF is the Architecture Definition Document – which is intended to provide a formal description of an enterprise system. The table shows key elements you'd expect to find in a completed Architecture Definition Document.

Entity class Domain	Physical component	Logical component	Service
Business	Organisation unit	Business function	Business service
Applications	Physical application component	Logical application component	I.S. service
Data	Physical data component	Logical data component	Data entity
Technology	Physical technology component	Logical technology component	Platform service

The focus here is on the business domain.

TOGAF says: Enterprise architecture is a horizontal function that looks at enterprise-level optimization and service delivery. Capability-based planning and enterprise architecture are mutually supportive.

Phase A: A business capability assessment is used to define what capabilities an organization will need to fulfil its business goals and business drivers. A business capability can be thought of as a synonym for a macro-level business function.

Also: Function describes units of business capability.

[TOGAF inherits Business Function from Information Engineering and the like, but inherits Capability from Capability-Based Planning. It hasn't quite got the nerve or the will to modify their two vocabularies until they are integrated.]

Phase B: Identifies the key business functions within the scope of the architecture, and maps those functions onto the organizational units within the business.

Target Business Architecture, Version 1.0 (detailed), including:

- Business goals and objectives — for the enterprise and each organizational unit
- **Business services** — the services that the enterprise and each enterprise unit provides to its customers, both internally and externally
- **Business functions** — a detailed, recursive step involving **successive decomposition of major functional areas into sub-functions**
- **Organization structure** — identifying business locations and relating them to organizational units

- Correlation of organization and functions — **relate business functions to organizational units** in the form of a matrix report.
- Business roles, including development and modification of skills requirements
- Business processes, including measures and deliverables
- Business data model [no explanation of what it is or what to do with it]

Organization [unit]: a self-contained unit of resources with **line management responsibility**, goals, objectives, and measures. Organizations may include external parties and business partner organizations.

Business Function: delivers business capabilities closely aligned to an organization, but not necessarily explicitly governed by the organization.

["governed" surely means "managed". Managers monitor and control organisations and processes at run time. Governance is more to do with the addition and removal of business services and business functions at design time.]

The term ''function'' is used to describe a unit of business capability at all levels of granularity, encapsulating terms such as value chain, process area, capability, business function, etc. Any **bounded unit of business function** should be described as a function.

[Bounded unit = encapsulated component. The boundary is the interface, containing the business services offered.]

Business Service: supports business capabilities through an explicitly defined interface and is explicitly governed by an organization. Business services support organizational objectives and are defined at a level of granularity consistent with the level of governance needed. A business service operates as a boundary for one or more functions.

[functions here could be business functions or business processes.]

The granularity of business services is dependent on the focus and emphasis of the business (as reflected by its drivers, goals, and objectives). A service in Service Oriented Architecture (SOA) terminology (i.e., a deployable unit of application functionality) … may implement or support a business service.

[In ISEB terms, by successive process decomposition, the coarse-grained business services/processes are elaborated until they reveal application use cases, then automated application services, then platform services.]

Capability: an ability that an organization, person, or system possesses. Capabilities are typically expressed in general and high-level terms and typically require a combination of organization, people, processes, and technology to achieve. For example, marketing, customer contact, or outbound telemarketing.

[Let me analyse that]

Capabilities are typically expressed in general and high-level terms. For example, marketing, customer contact, or outbound telemarketing.

[Those sound like business functions. Then…]

"and typically require a combination of **organization, people, processes, and technology** to achieve."

[In other words, things are *added to the Capability* to realise it. They are not part of the initial definition. Capability is first defined as a high-level Business Function, then elaborated, gathering other architectural entities as it grows. Eventually, probably, it turns into a cross-organisational organisation unit, with all the resources need to do work.]

Capabilities are engineered/generated taking into consideration various dimensions that **straddle the corporate functional portfolios**.

[This surely means **straddle the lines-of-business** in the physical organisation structure, rather than the functions in the logical business function structure.]

Building Block

In TOGAF, the term Building Block appears to have two meanings. Sometimes it is an encapsulated *component*. Sometimes it is *any architectural entity*, including services, goals and locations. The term Building Block is eschewed in the ISEB reference model, and could be dropped without harm.

Aside on hierarchical taxonomies

A formal(ish) meta model of concepts is one thing; a glossary we might use in practice with stakeholders is another.

How do people manage the challenge of talking about the same or very similar things but at different levels of abstraction? People sometimes use three-level taxonomic hierarchies. For example: process, procedure and activity.

In the real world, it can be useful to use different words to distinguish between levels of abstraction. This stops us having to talk about (say) big goals and little goals, coarse-grained processes and fine-grained processes.

The hierarchical taxonomies in the Business Motivation Model from the OMG can be adapted and extended for use by the architects of activity systems thus:

- Aims: goal –< objective –< requirement.
- Directives: principle –< policy –< business rule.
- Solution description: solution vision –< solution outline –< detailed design.
- Plans: strategy –< programme –< project.

For more on this topic: http://avancier.co.uk. > Library > Architecture Concepts > "Granularity"

Bear in mind however that distinctions between the levels are not formal or universally repeatable in different contexts.

The distinctions cannot be formalised because the context, the scope, to which the words are applied varies from *very large* systems and projects to *very small* ones. So these word hierarchies are used in a subjective way to indicate how we move up and down through levels of abstraction in a particular context.

It seems OK to use these word hierarchies in the chaotic and informal management-consultant-speak of "architecture precursors". But it is more problematic to do the same thing in the more formal world of system modelling.

Systems are recursively nested. There is one indisputable level of abstraction - the bottom level. At the bottom level of a data processing system there are elementary data items and executable instructions. At the top level is the boundary of the system - a scoping decision made by stakeholders.

Between the top and bottom, you can define hierarchies of processes and components with many levels. (Though it is commonly said that a hierarchy with three of four levels is the limit of what is readily understood an maintained by one person.)

Given these variations and uncertainties, it would be risky to go down the road of defining different levels of system and process composition by using different words, such as:

- System –< subsystem –< component –< module.
- Process –< procedure –< activity –< executable instruction

This would give false impressions of rigour. There is no agreement that there are four distinguishable levels of abstraction for architecture, or that this could be generally applied across most systems.

Some do use process, procedure and activity as a taxonomic hierarchy, but the levels are not formally distinct or repeatable in all contexts. In a meta model of activity system concepts they are better generalised as “process”.

Architects understand that, since there are large systems and small systems, and infinite levels of system composition are possible, they have to make a judgement about the right level to pitch their architectural description.

Can any architecture language work for stakeholders?

Architects have to be taught architecture concepts. A 3-day course is surely a minimal expectation. An examination in architecture concepts has to be based on an agreed vocabulary of words. I can’t think how to get around these facts.

So how can we expect untrained-architects to readily understand this vocabulary? Or to understand any “language” of graphical symbols such as can be found in ArchiMate or UML? Surely, it must be the job of the architect to translate from the word/concept/symbol combinations they have been taught into the language of the stakeholder in front of them.