

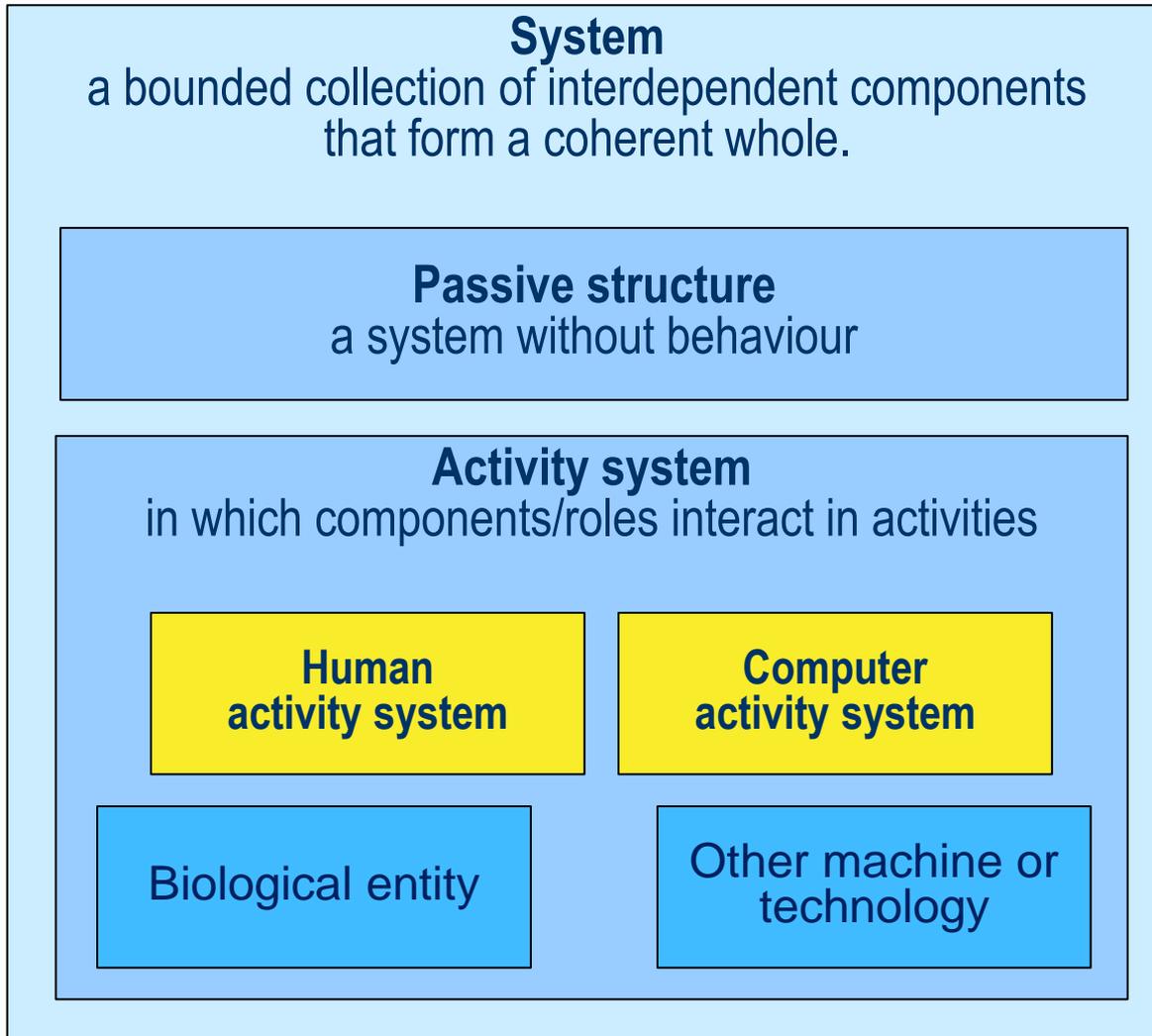
# Avancier Methods (AM) Conceptual Framework



A generic meta model of system elements

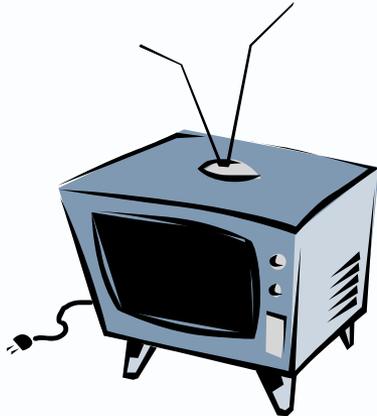
It is illegal to copy, share or show this document  
(or other document published at <http://avancier.co.uk>)  
without the written permission of the copyright holder

- ▶ TOGAF says
- ▶ “EA regards the enterprise as **a system of systems**”
- ▶ “architecture has two meanings:
  - 1. **A formal description of a system...**
  - 2. The structure of components, their inter-relationships...”
- ▶ And it features c1,400 appearances of the word “**system**”.
- ▶ What is a system?



- ▶ (EA) and solution architecture (SA) are focused on
  - enabling and improving business roles and processes that are
  - repetitive and deterministic enough to be
  - systematised and digitised.
- ▶ So, enterprise and solution architects are expected to build models of human and computer activity systems.

▶ An operational system



- ▶ is both a bounded collection of components, and
- ▶ a bounded collection of processes that transform inputs into outputs.

▶ An architecture description



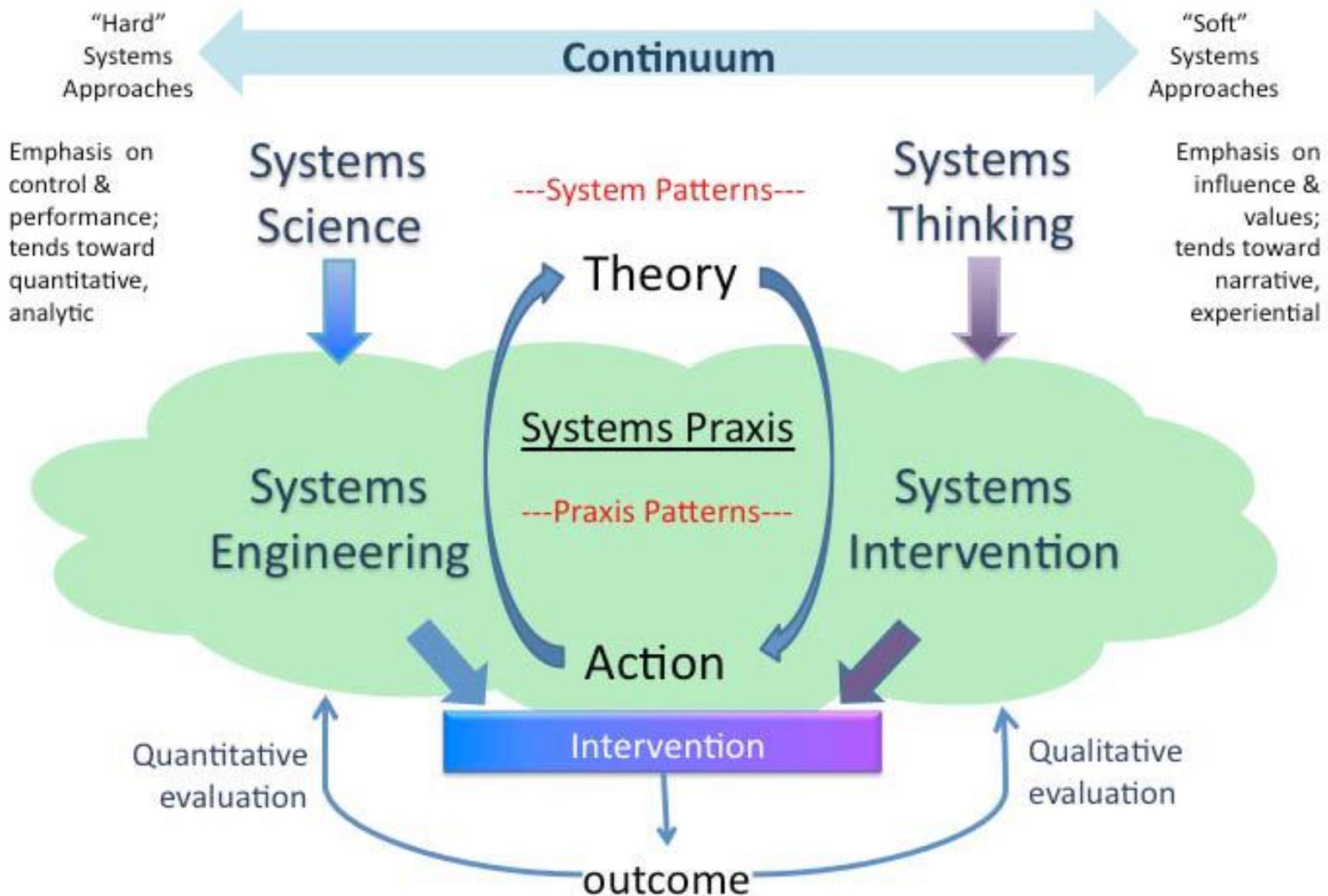
- ▶ has to specify system structure and behaviour.
- ▶ A description exists separately from an operational system, before it is built, and after it is changed or destroyed.

“architecture: fundamental concepts or properties of a system in its environment **embodied in its elements**, relationships, and in the principles of its design and evolution.”  
ISO 42010

- ▶ This paper discusses basic system elements.
- ▶ It serves as a preface to more substantial papers on
  - type theory,
  - system theory and
  - enterprise architecture.

- ▶ The academic spectrum runs from sciences to humanities:
  - Maths
  - Physics
  - Chemistry
  - Biology
  - Psychology
  - Sociology
  - Politics
  
- ▶ The schools of systems thinking spread from
  - the most scientific of engineering to
  - the most political of management consulting,

# General Systems Theory at the scientific end of a continuum



# General Systems Theory concepts

- ▶ “von Bertalanffy...emphasized that **real systems are open to, and interact with, their environments**, and that they can acquire qualitatively new properties through emergence, resulting in continual evolution.
- ▶ Rather than reducing an entity (e.g. the human body) to the properties of its parts or elements (e.g. organs or cells), systems theory focuses on **the arrangement of and relations between the parts which connect them into a whole (cf. holism)**.
- ▶ This particular organization determines **a system, which is independent of the concrete substance of the elements** (e.g. particles, cells, transistors, people, etc).
- ▶ Systems concepts include: **system-environment boundary, input, output, process, state, hierarchy, goal-directedness, and information.**”

Principia Cybernetica

A system interacts with its environment via **inputs** and **outputs**

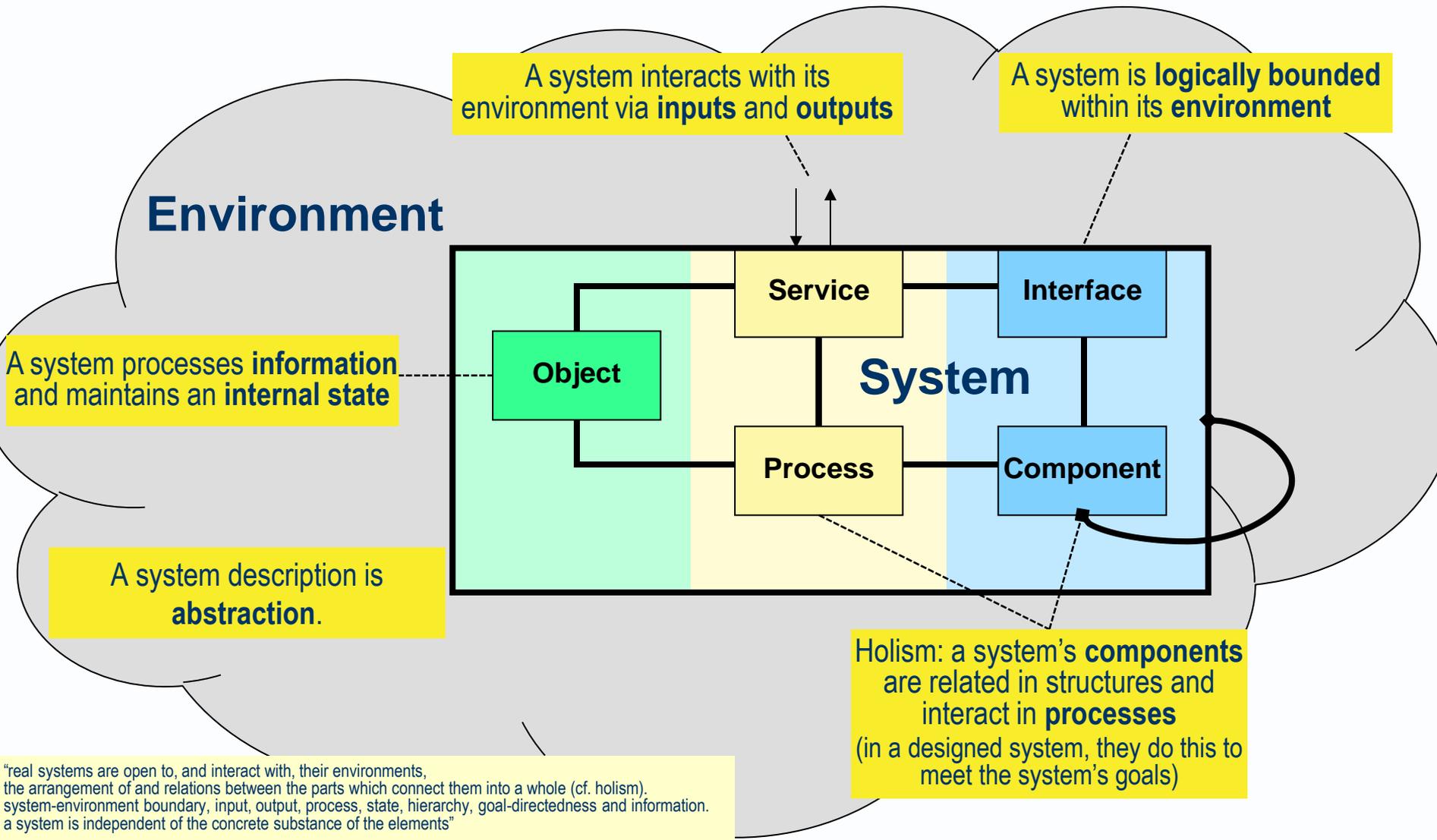
Holism: a system's **components** are related in structures and interact in **processes** (in a designed system, they do this to meet the system's goals)

A system description is **abstraction**.

A system is **logically bounded** within its **environment**

A system processes **information** and maintains an **internal state**

# Five tenets of general system theory



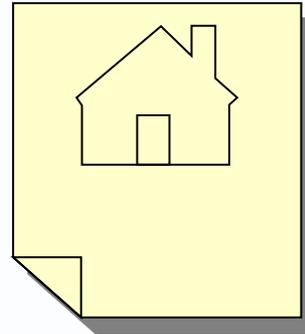
- ▶ Introduction
- ▶ **Context**
- ▶ System elements
- ▶ More about the internal view
- ▶ More about the external view
- ▶ Ambiguities
- ▶ Ten principles

## ► Contextual information

- stakeholders, concerns, requirements, principles, time, cost etc.
- all system description precursors that architects must respond to.

## ► System descriptions

- architect's drawings of buildings
- for builders to follow



A system description is **abstraction**.

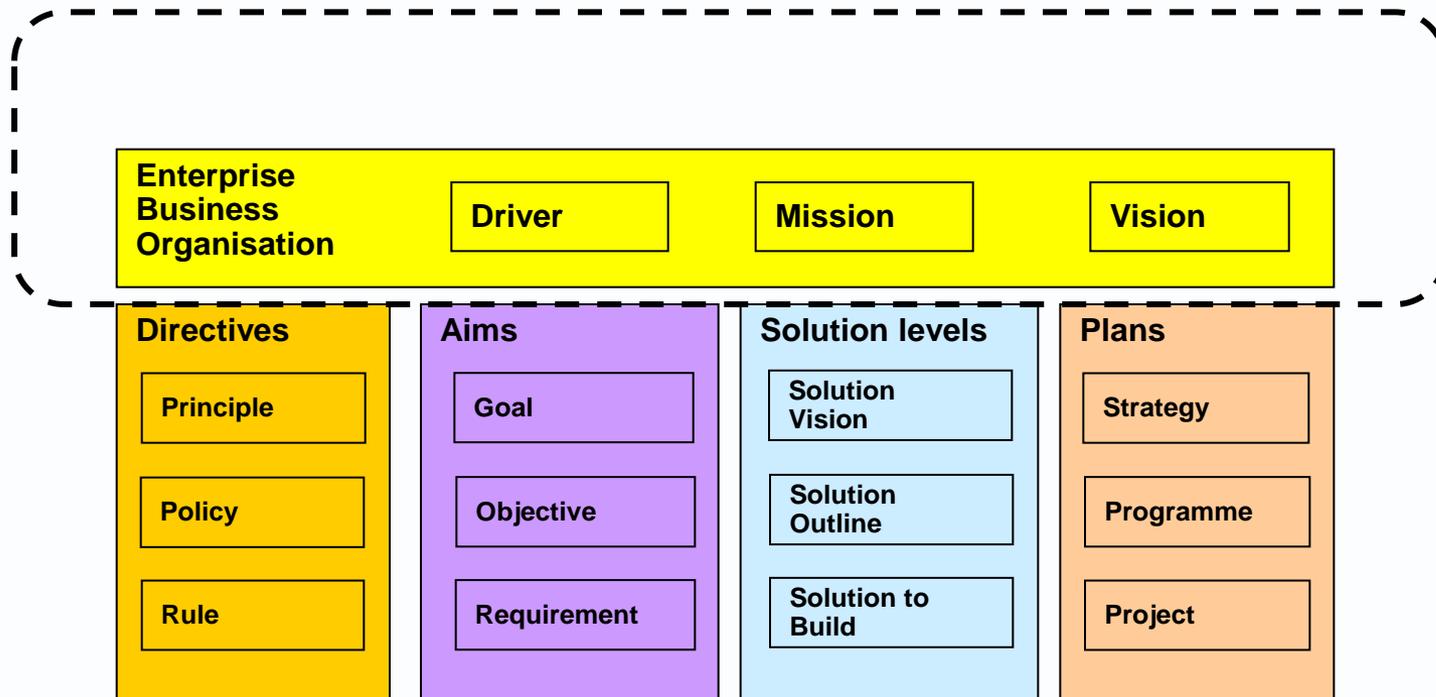
## ► Operational systems

- buildings
- both already built and to be built



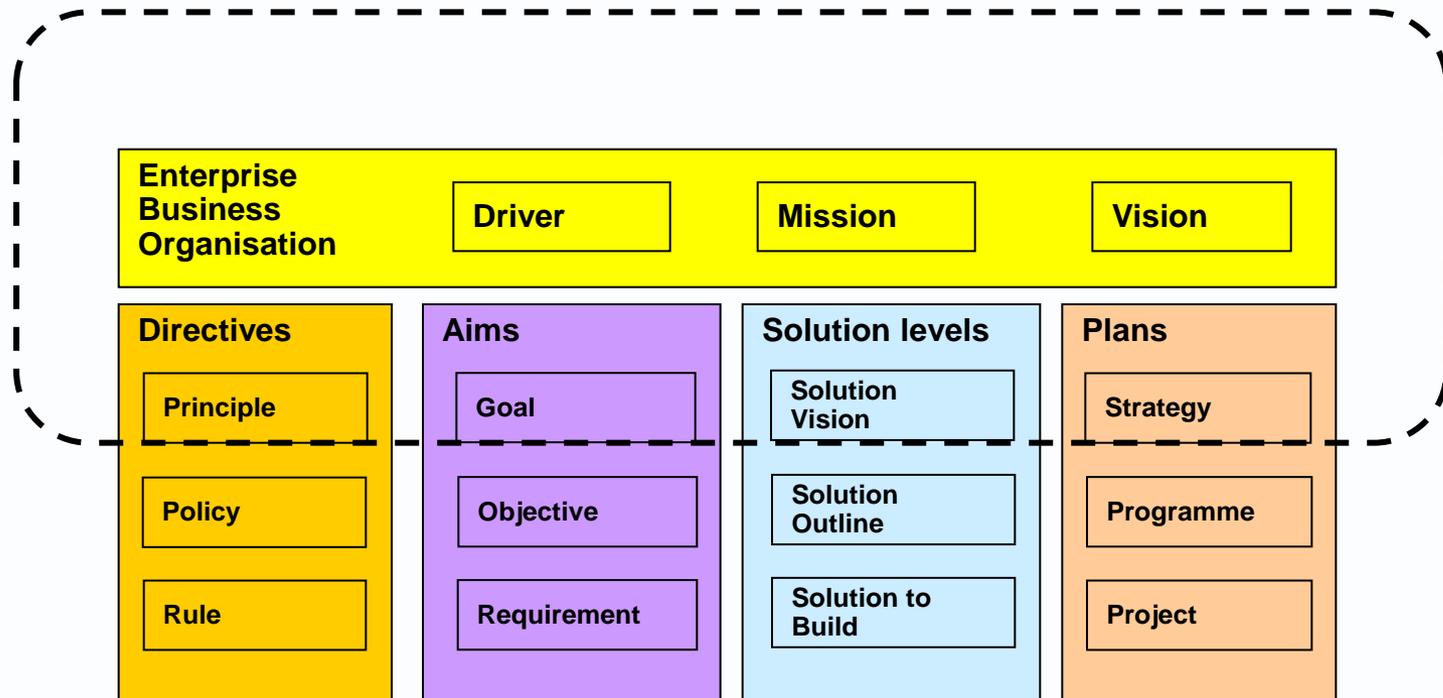
# The context for architecture – business executive level

- ▶ The architect must understand the requirements and context for the enterprise system(s) to be designed and described.
- ▶ The context includes business drivers, mission, vision



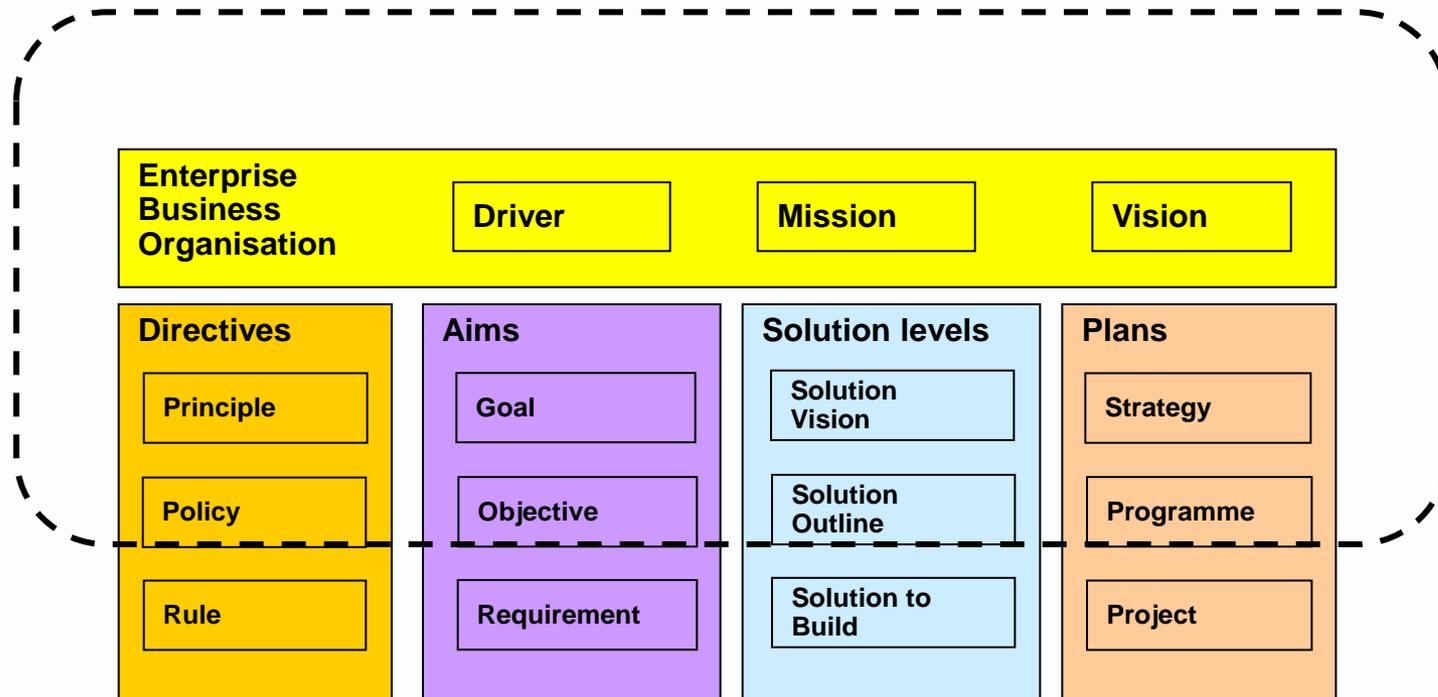
# The context for architecture – high level

- ▶ This context may include already-defined principles, goals, solutions visions and strategy.



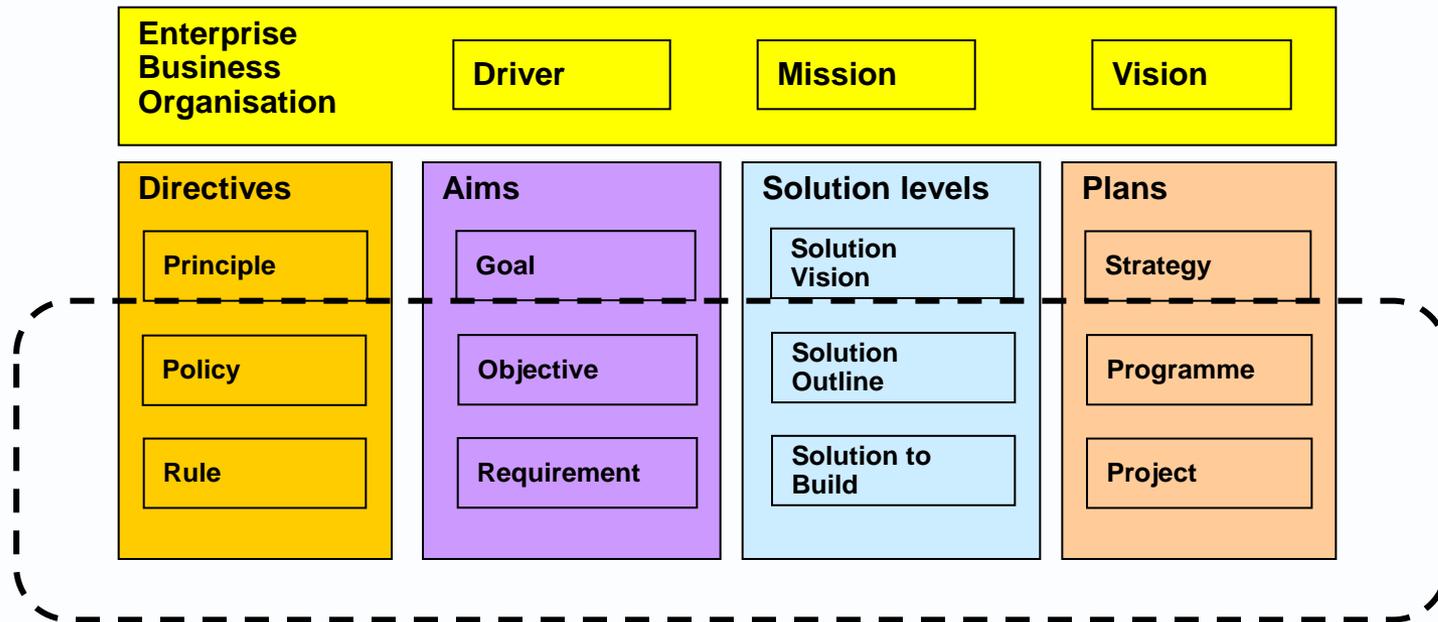
# The context for – lower level

- ▶ This context may include more detailed policies, objectives and programme-level plans.



# Successive refinement and elaboration

- ▶ The upper levels of this graphic are inputs
- ▶ Lower levels may emerge during architecture description, or as a result of it



- ▶ Introduction
- ▶ Context
- ▶ **System elements**
- ▶ More about the internal view
- ▶ More about the external view
- ▶ Ambiguities
- ▶ Ten principles

# The systems to be described

- ▶ Architects are employed to typify business system elements
- ▶ Human and computer activity systems have behaviour as well as structure

The duality of activity system elements	
Persistent structure	Transient behaviour
Actors	Activities
Entities	Events
Objects	Operations or use cases
People	Processes
Roles	Rules
Stocks	Flows
Static components	Dynamic behaviour

- ▶ “The principal heuristic innovation of the systems approach is what may be called ‘reduction to **dynamics**’ as contrasted with ‘reduction to **components**’” Laszlo and Krippner.

# Description has structural and behavioural elements

Structural view		Behavioural view
Noun		Verb
Form		Function
<b>Components</b>	<b>perform</b>	<b>Processes</b>

- ▶ “It is the pervading law of all things organic and inorganic, of all things physical and metaphysical, of all things human and all things super-human, of all true manifestations of the head, of the heart, of the soul, that the
- ▶ **life is recognizable in its expression, that**
- ▶ **form ever follows function.**
- ▶ This is the law.”
- ▶ American architect Louis Sullivan, 1896.

# What elements can be seen *inside* an activity system?

- ▶ Structural elements (components) cooperate in
- ▶ Behaviour elements (processes).

	Behavioural view	Structural view
External view		
Internal view	<b>Processes</b>	<b>Components</b>

Holism: a system's **components** are related in structures and interact in **processes** (in a designed system, they do this to meet the system's goals)

- ▶ This principle underpins most architecture frameworks.

# What elements can be seen from *outside*?

- ▶ Interfaces – which encapsulate components
- ▶ Services – which encapsulate processes.

A system interacts with its environment via **inputs** and **outputs**

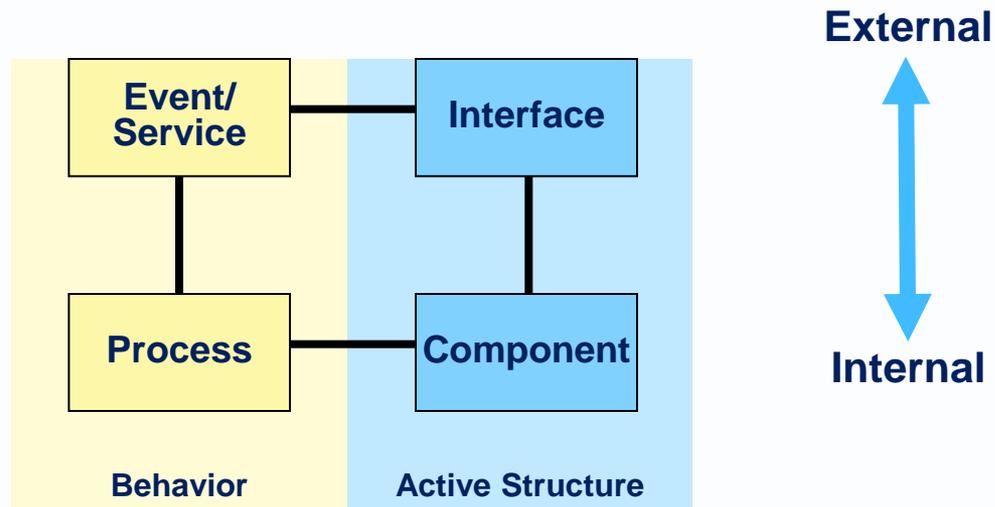
A system is **logically bounded** within its **environment**

	Behavioural view	Structural view
External view	Event/Services	Interfaces
Internal view	Processes	Components

- ▶ These principles underpin most architecture frameworks.

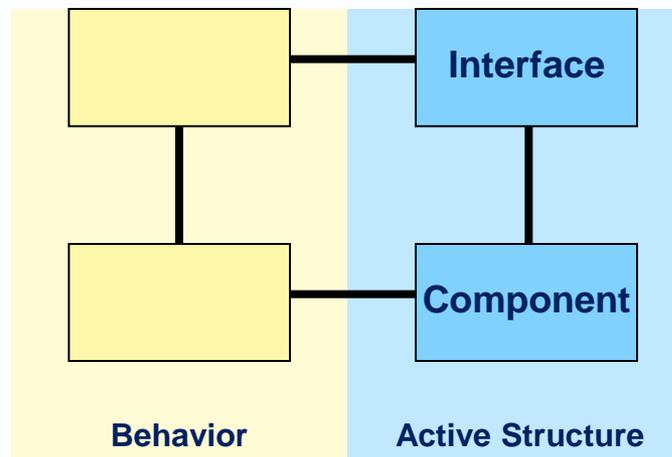
# Distinguishing external and internal elements

External views		Internal views
Facades	encapsulate	Contents
<b>Interfaces</b>	<b>encapsulate</b>	<b>Components</b>
<b>Services</b>	<b>encapsulate</b>	<b>Processes</b>
<b>Events</b>	<b>trigger</b>	<b>Processes</b>



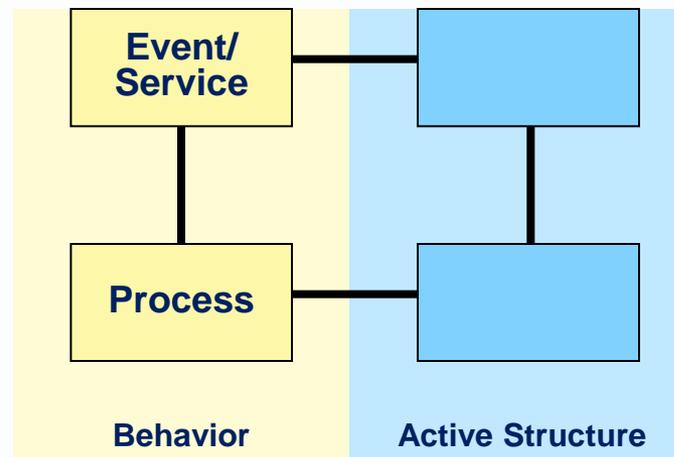
# A structural element is usually considered thus

- ▶ named using a noun
- ▶ persists, perhaps throughout the life of the system described.
- ▶ can be found at an address.

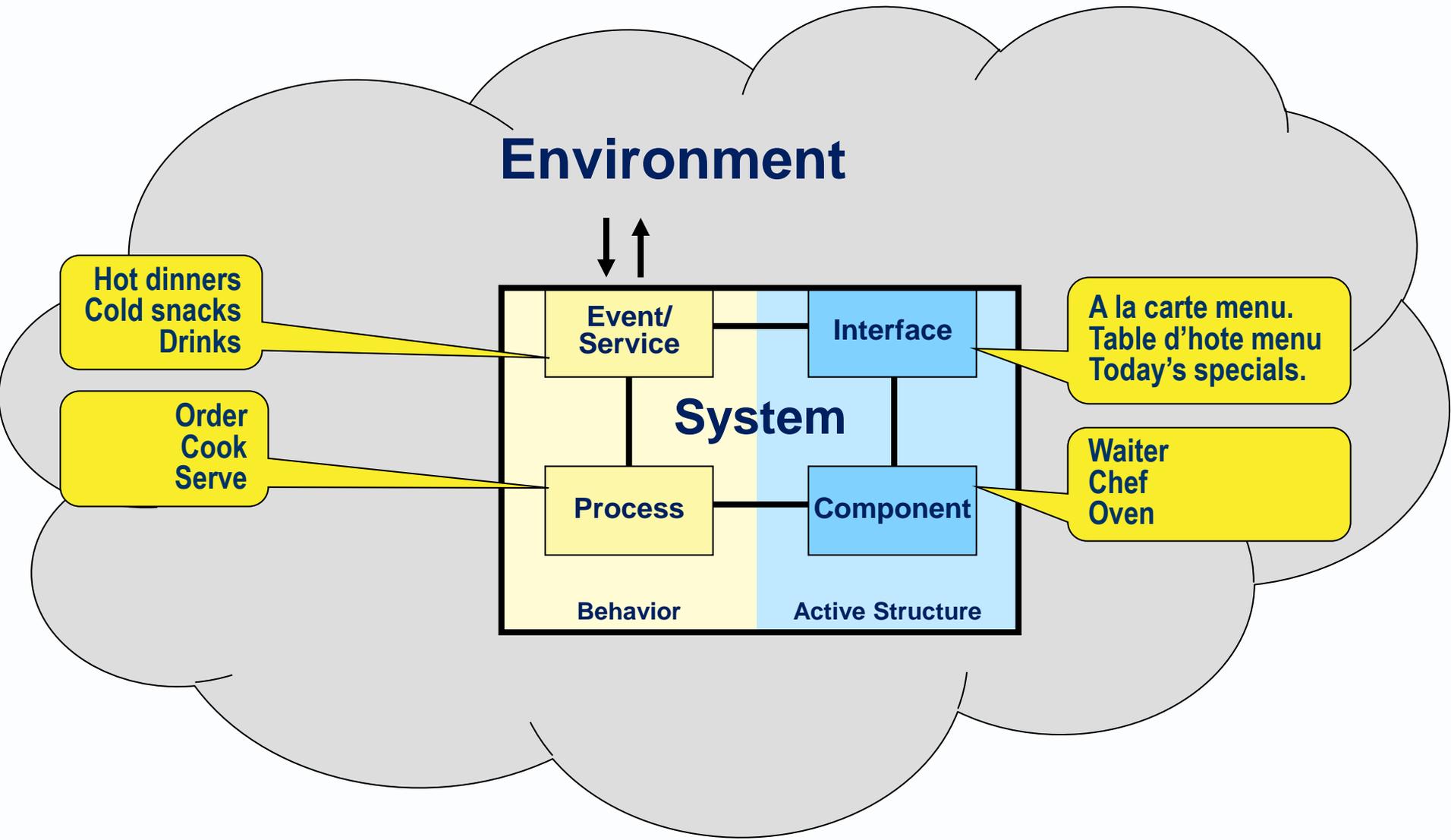


# A behavioural element is usually considered thus

- ▶ named using a verb
- ▶ transient, lives and dies within the life of the system described
- ▶ has a start and end in time (and usually repeats).
  - Usually, persistent component/roles outlive transient processes.
  - This is not always true, but it is a helpful way of thinking about the structure/behaviour distinction.



# A system is encapsulated in an environment



## What about objects that are acted on?

- ▶ There are
- ▶ active structural elements (actors or processors) and
- ▶ passive structure (acted on or processed objects),

Active Structure	Behaviour	Passive Structure
Subjects	Acts on	Objects
Actors	Act on	Stages
Machines	Consume	Fuel
Processes	Update	State / Data
People	Read	Books

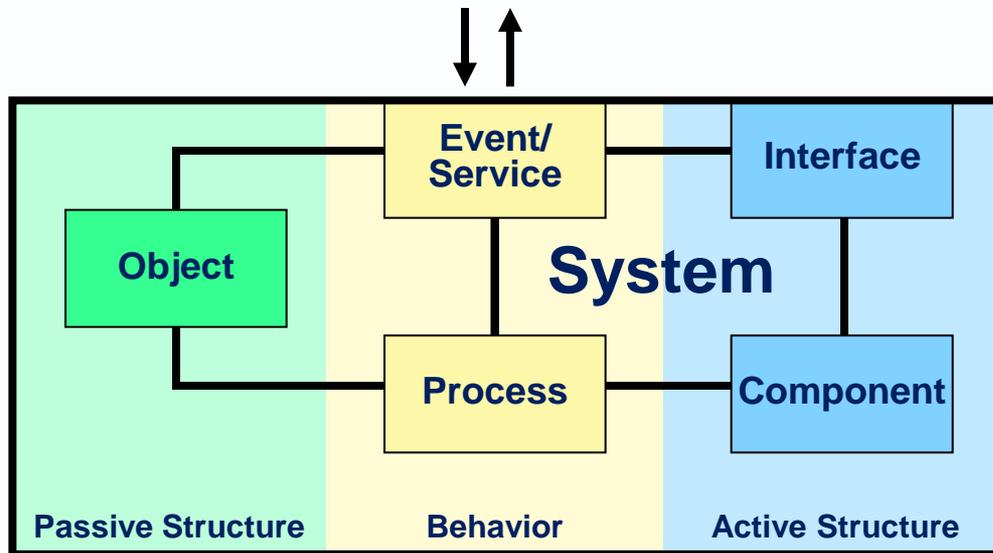
# Our concern is information-intensive organisations

- ▶ “EA is the determinant of survival in the **Information Age.**”
  - (Zachman)
  
- ▶ “the domain of **information-intensive organisations...**is the main focus of the language”
  - (The ArchiMate modelling language standard v2.1)
  
- ▶ "companies excel because they've [decided] which processes they must execute well, and have implemented the IT systems to **digitise those processes.**"
  - (Ross, Weill and Robertson)
  
- ▶ “Today’s CEOs know that the effective **management and exploitation of information** through IT is a key factor to business success.”
  - (TOGAF 9.1)

A system processes **information** and maintains an **internal state**

# Where is the system's state?

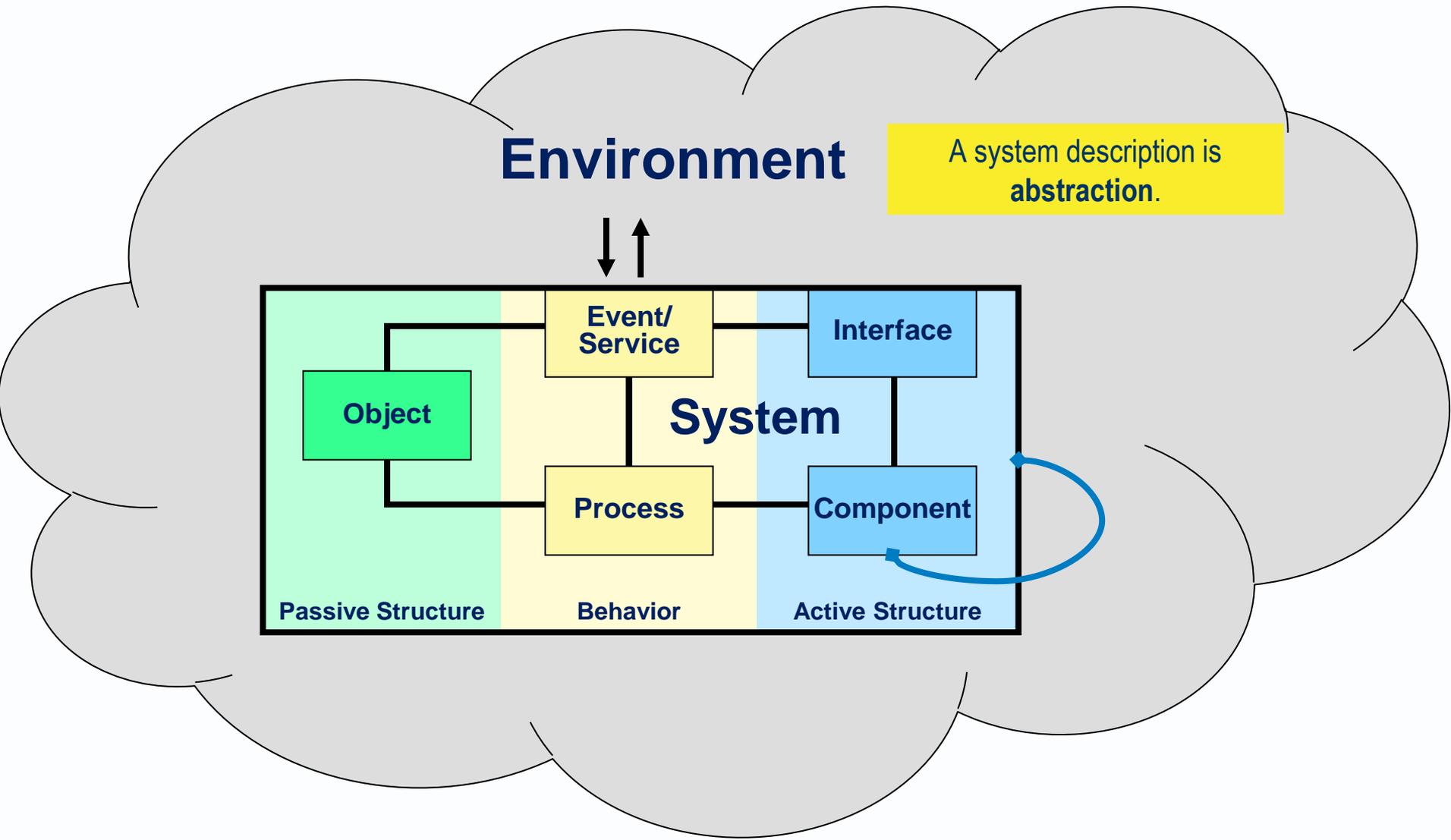
- ▶ The business system's information state
  - (its memory, updated and referred to by processes)
- ▶ contains structured business data objects



**Object:** an item or structure that is used, moved or made by processes

A system processes **information** and maintains an **internal state**

# System elements are recursive

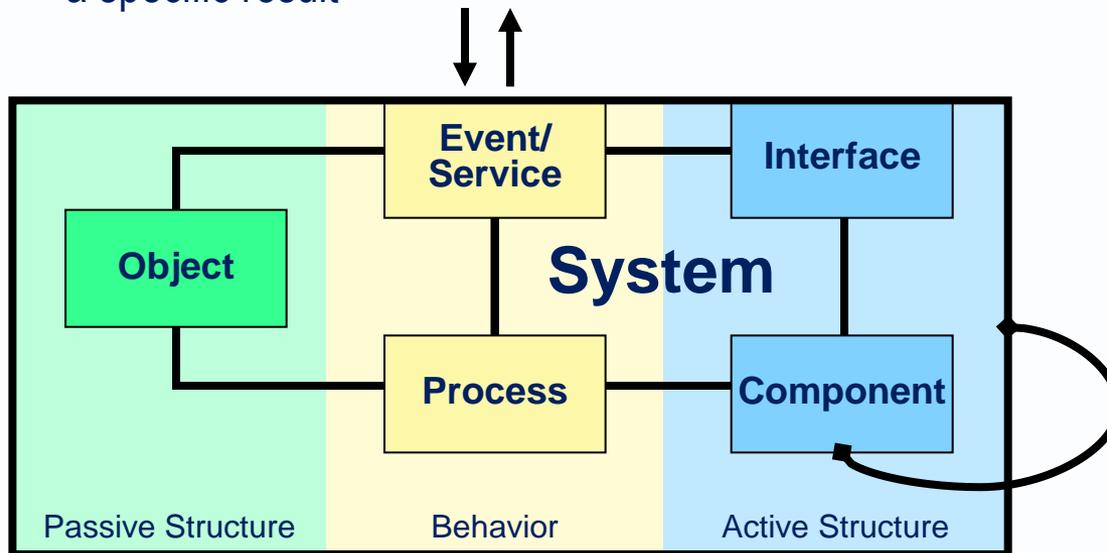


# 5 system description entities as Avancier defines them

**Service:** what a client can request to process a specific event or deliver a specific result

**Interface:** presents services for access by clients

**Object:** an item or structure that is used, moved or made by processes

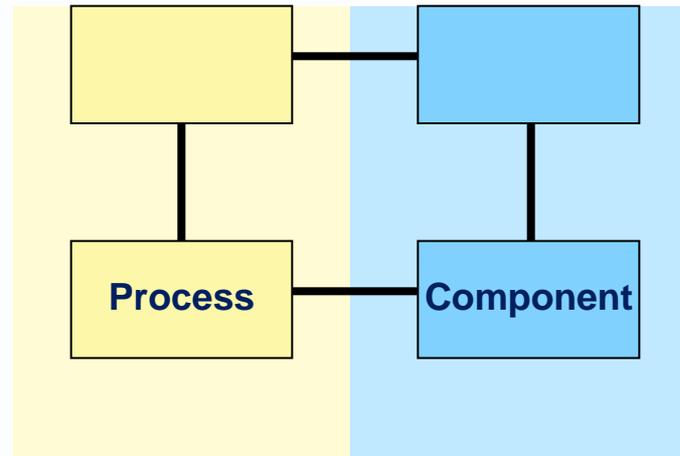


**Process:** a sequence of activities that respond to an event or meet a service request.

**Component:** a subsystem that performs process steps.

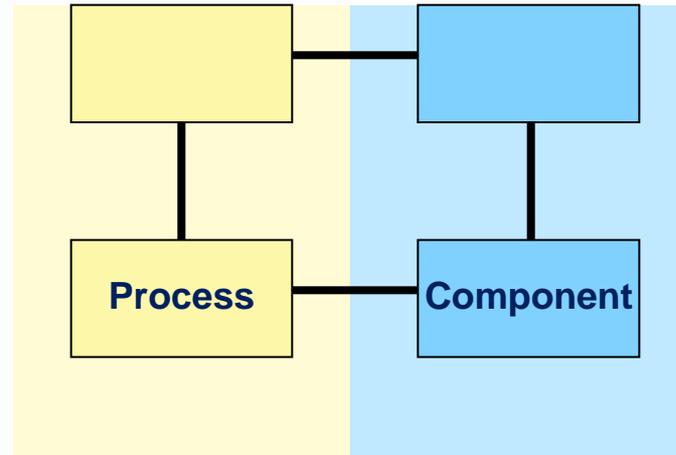
- ▶ Introduction
- ▶ Context
- ▶ System elements
- ▶ **More about the internal view**
- ▶ More about the external view
- ▶ Ambiguities
- ▶ Ten principles

- ▶ After general system theory, activity systems can be viewed as containing two kinds of system element.
- ▶ Inside the system, we can observe:



- ▶ Processes – activities and sequences of activities (describable in flow charts and interaction diagrams)
- ▶ Components/roles - requested or expected to perform activities.

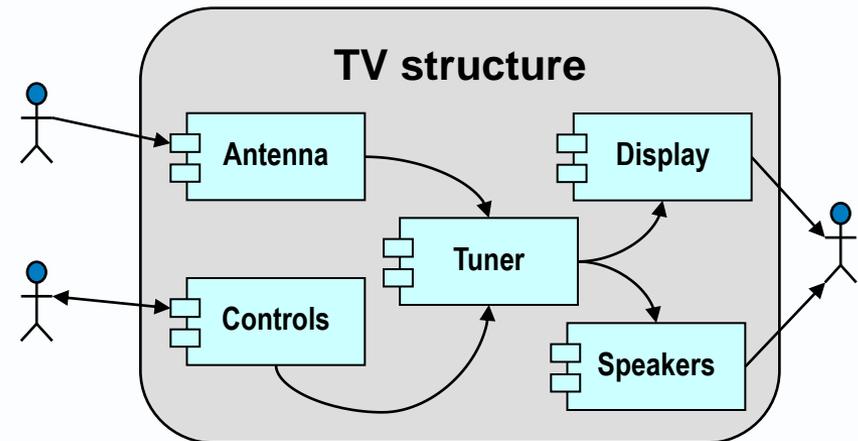
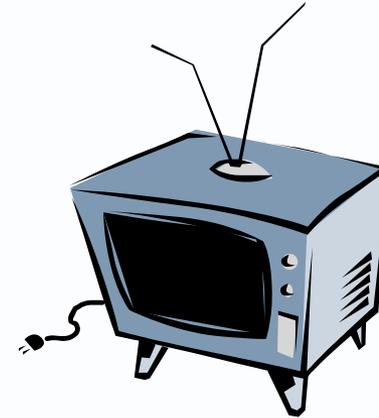
- ▶ Components/roles are persistent structural elements; they have attributes and perform activities.
- ▶ Processes are transient behavioural elements; they create objects and change the state of objects.



- ▶ A shorter process may be performed by one component;
- ▶ a longer process may require several components to cooperate.

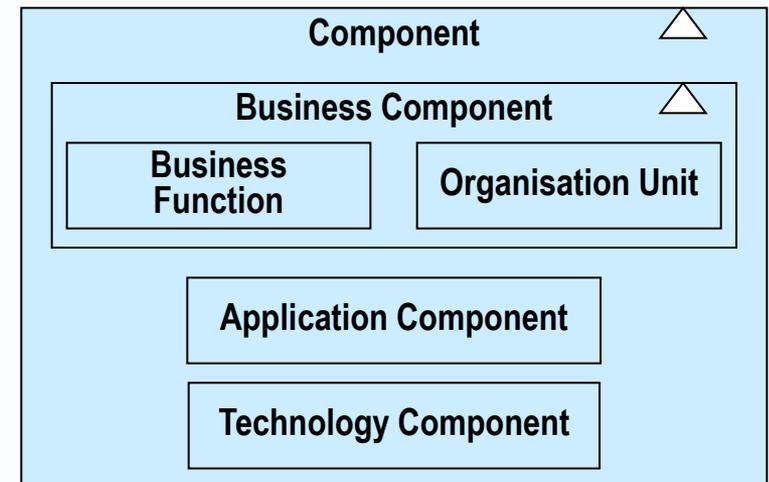
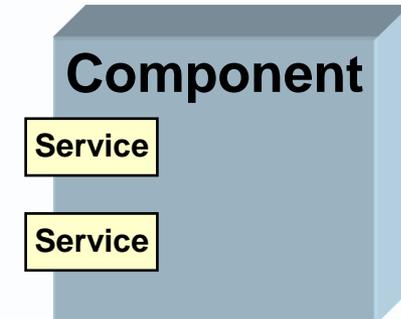
# The structure inside a system

- ▶ A system is a bounded collection of components, which are all connected to each other directly or indirectly (else it would be two or more systems).
- ▶ E.g. a bridge, car, television, IT network.
- ▶ Look inside and you can see the inter-connected components.
- ▶ The internal structure may be shown as inter-related subsystems in some kind of goods/service/data flow diagram.



# A component

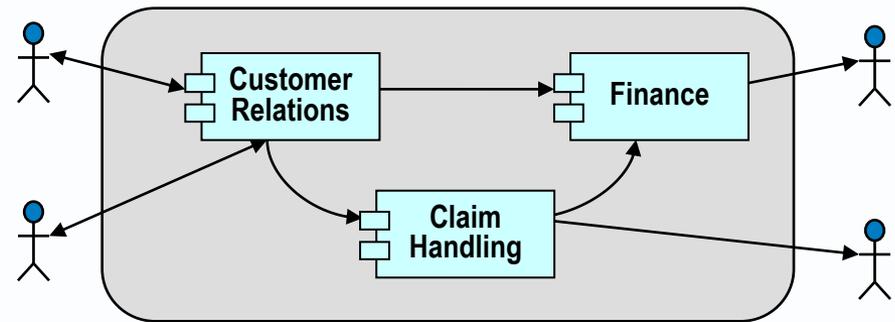
- ▶ is a subsystem
- ▶ is encapsulated behind an interface.
- ▶ can be replaced by any other with the same interface.
- ▶ is related to other subsystems by requesting or delivering services.
- ▶ can have several interfaces.



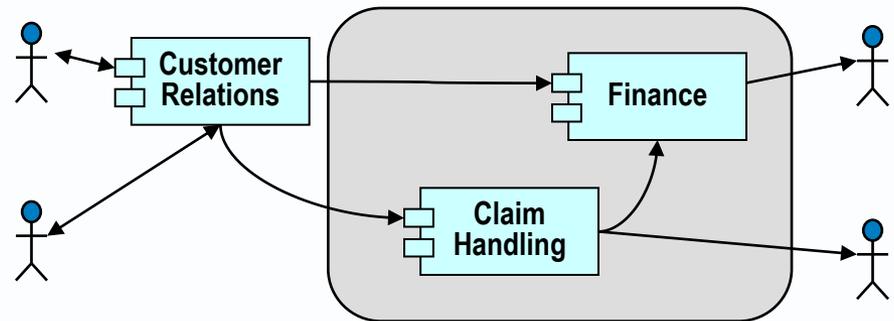
# The scope of the enterprise/system is subjective

▶ The scope or boundary is a decision made by one or more observers.

▶ One person's internal component is

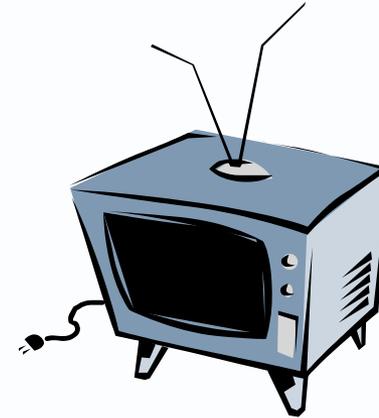


▶ another person's external entity.

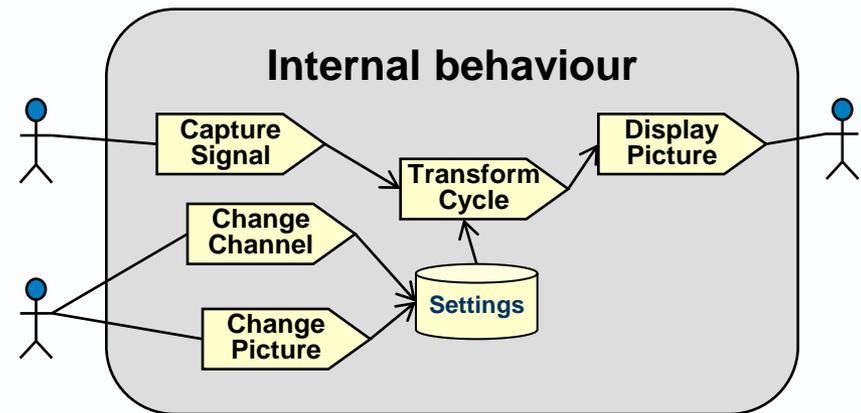


# The behaviour inside a system

- ▶ An activity system is also a bounded collection of processes that transform inputs into outputs.
- ▶ A television, IT network, software system, human activity system.



- ▶ Inside are processes that transform inputs into outputs.
- ▶ These processes may be listed on a process map or use case diagram.
- ▶ The steps and flow of each process can be charted.

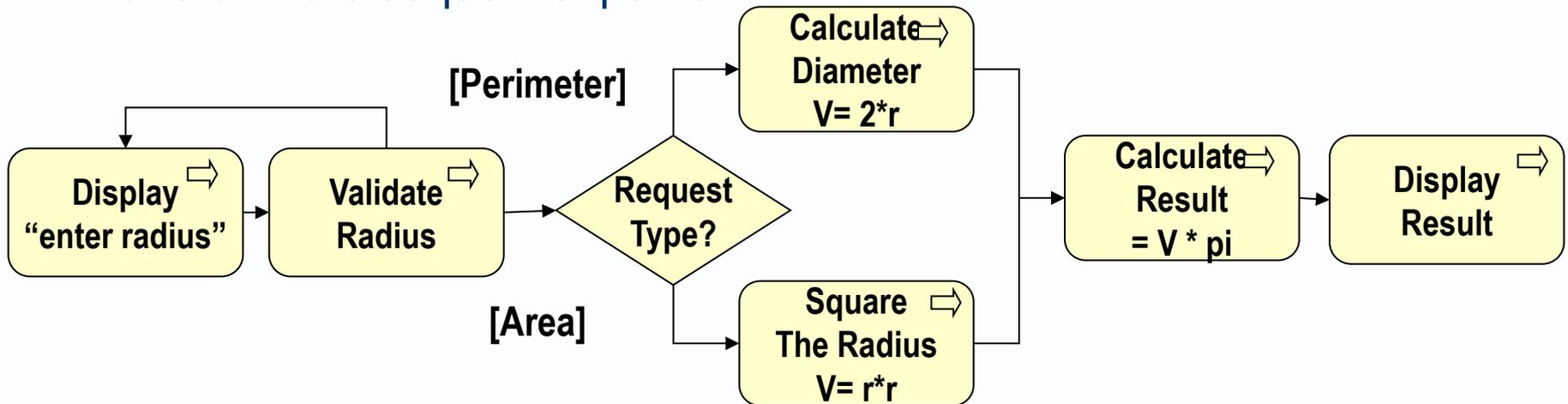


# A process

Process →

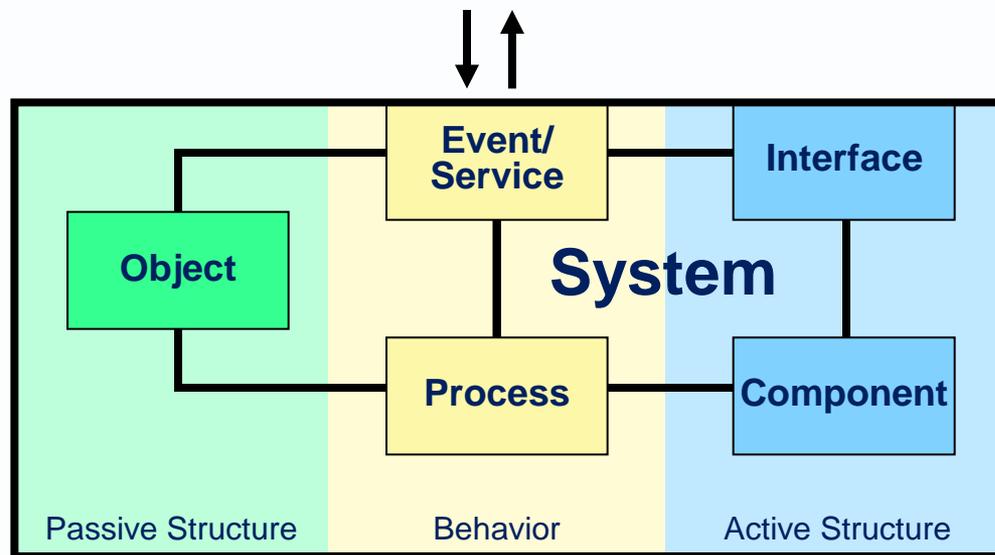
Avancier

- ▶ is a procedure, started by an event, which terminates with the delivery of an output or service.
- ▶ is a set of steps or activities arranged under a control flow in one or more sequential paths.



# Where is the human brain?

- ▶ Human psychology and culture are not addressed in this material
- ▶ The human brain appears as a component, playing a role

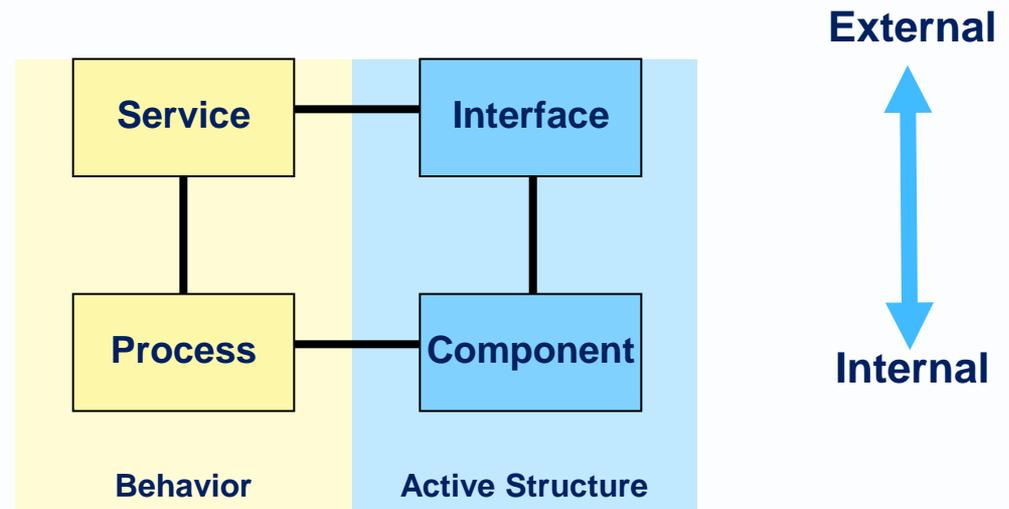


**Component:** a subsystem that performs process steps.

- ▶ Introduction
- ▶ Context
- ▶ System elements
- ▶ More about the internal view
- ▶ **More about the external view**
- ▶ Ambiguities
- ▶ Ten principles

# Taking an external view of a system

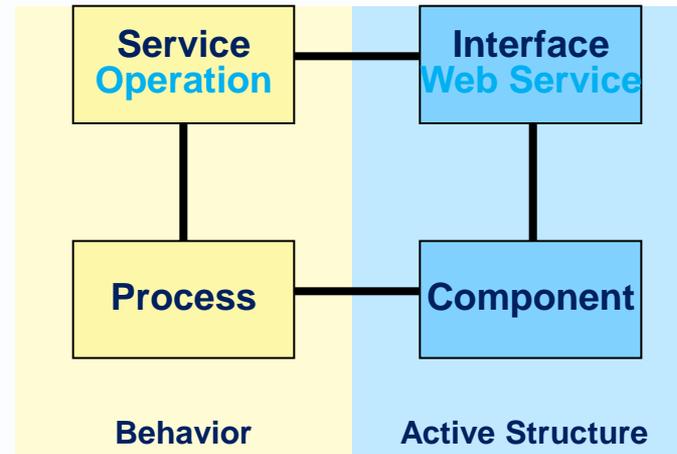
- ▶ We can define two additional basic system elements:
- ▶ **Services** encapsulate the processes – describable in service contracts.



- ▶ **Interfaces** encapsulate the components/roles – declare lists of services that are provided or required by a component/role.

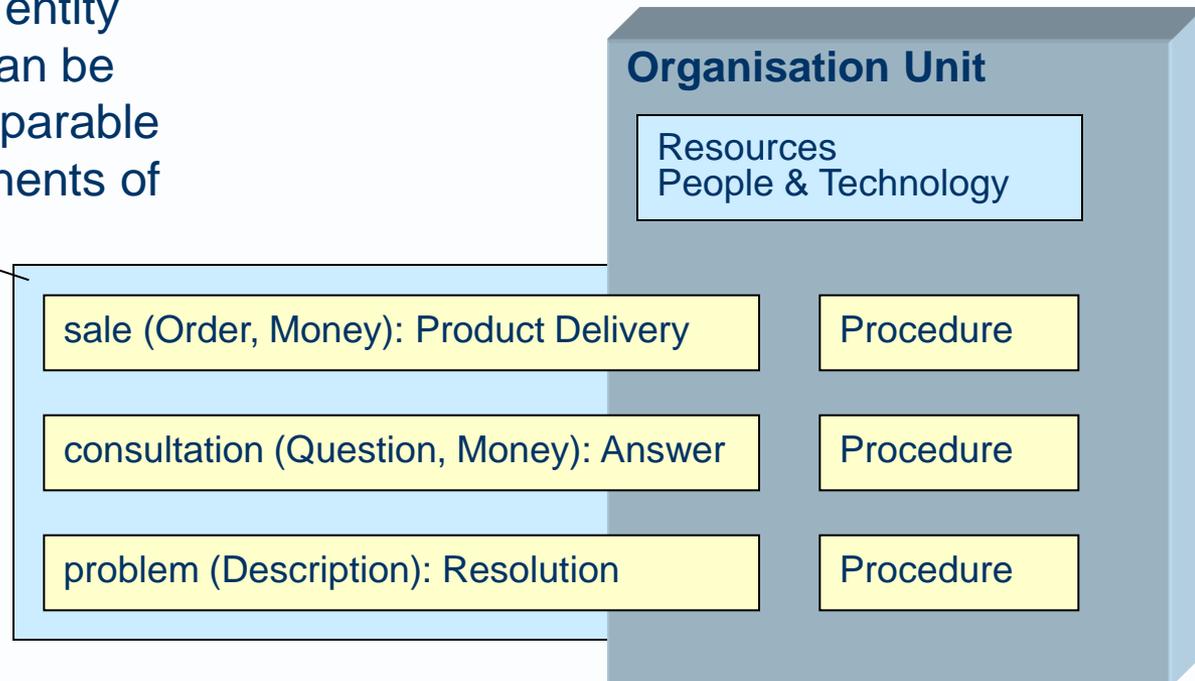
## But beware

- ▶ These four terms (and other system description terms) are used ambiguously and/or vaguely.
- ▶ E.g. the *Web Services Definition Language* is a kind of *Interface Definition Language*.



- ▶ The discrete services in a web service are called *operations*.

- ▶ are based on the idea that
  - ▶ a service can be documented in a **service contract** (name, input, output, rules and non-functional requirements), and
  - ▶ the services an external entity is allowed to consume can be defined an **interface**, separable from the internal components of the system.
- ▶ An interface might be documented in some kind of menu, or service catalogue or directory, or service level agreement.



# A service

Service

Avancier

- ▶ is something definable by a service contract.
- ▶ Here is a service provided by a barber shop.

Service Contract		Business Service 999
Signature	Name	Haircut
	Input	Hair length
	Output	Shorter hair
Semantics or rules	Preconditions	Customer can afford haircut
	Post conditions	Money transfer. Resource wear
Non-Functional Requirements	Response time	45 minutes
	Throughput	3 per hour per shop
	Availability	99% 09.30 to 18.00
	Security level	

## Services can be nested.

A yellow rounded rectangle with a black border, containing the word "Service" in black text. A small white oval is in the top right corner.

Service

The Avancier logo, consisting of three vertical bars of increasing height from left to right, with the word "Avancier" written below them.

Avancier

- ▶ A service is singular in the sense that it has a single service contract – at the boundary of a system.
- ▶ However, that singular service may be divided behind the scenes into countless other services.

# A service provided by a software application

Service

Avancier

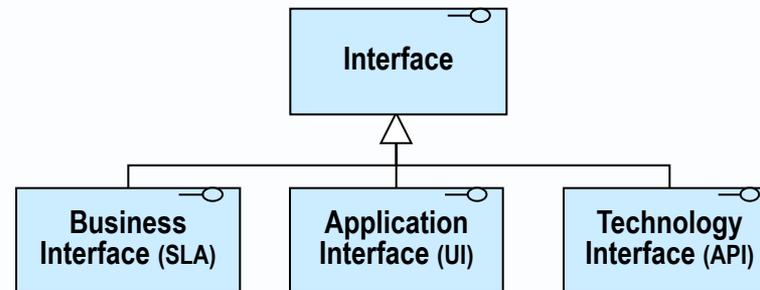
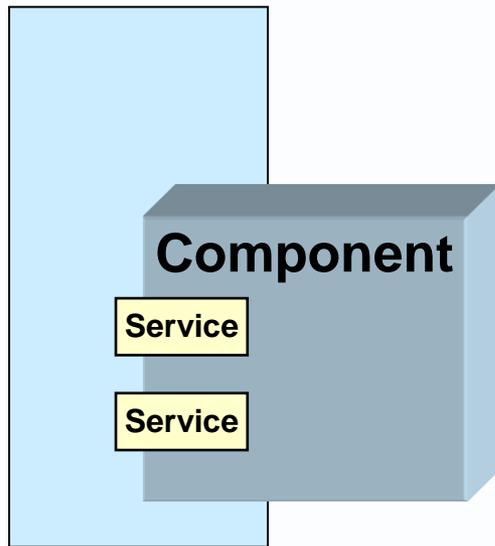
<b>Service Contract</b>		<b>Automated Service 999</b>
<b>Signature</b>	Name	Store New Claim
	Input data	Employee id, Claim Lines
	Output data	OK or Fail
<b>Semantics or rules</b>	Preconditions	Employee in a valid state to make claims
	Post conditions	Claim record stored in unapproved state
<b>Non-Functional Requirements</b>	Response time	0.5 second
	Throughput	5 per minute
	Availability	99% 07.00 to 19.00
	Security level	2

# An interface

Interface

Avancier

- ▶ provides the means of connecting to a system, process or component.
- ▶ presents a list of services, offered by one or more components.



# A Business Interface - in a Service Level Agreement (SLA).

- ▶ Real business contracts say surprisingly little about the services required and work to be done; since they are mostly about insurance and what happens if either party does or wants to break the contract.

## **SLA Body**

Definitions and Interpretation, Term of Agreement, Service Provider's Obligations, Client's Obligations, Fees, Payment and Records, Provision of the Services, Service and Agreement Monitoring, Performance Management and Monitoring, Confidentiality, Intellectual Property Rights, Termination, Post-Termination, Liability and Indemnity, Force Majeure, Nature of the Agreement, Severance, Relationship of the Parties, Notices, Law and Jurisdiction.

## **SLA Schedules**

The Services (and for each service or more generally)  
Service Levels, Performance Monitoring and Performance Reports, Fees and Payment & Penalty Fees

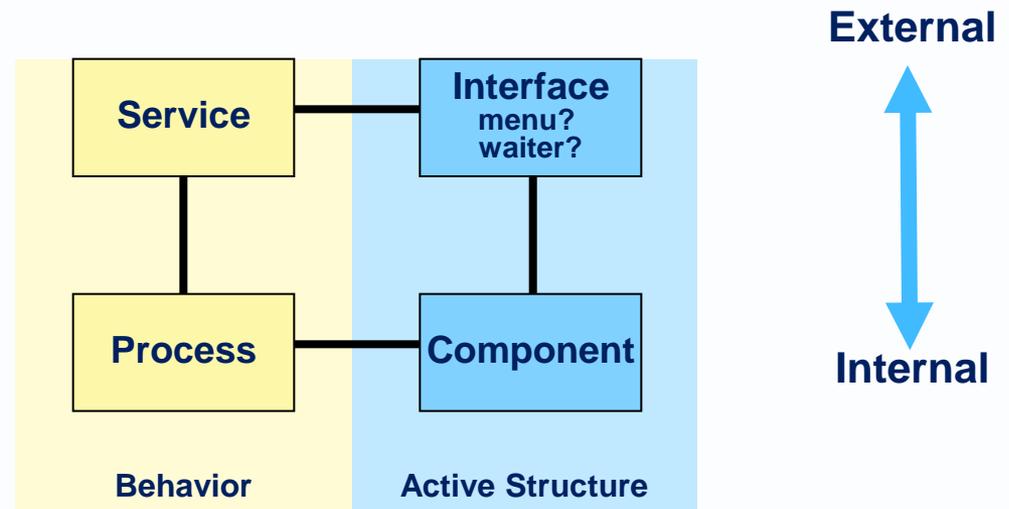
- ▶ Introduction
- ▶ Context
- ▶ System elements
- ▶ More about the internal view
- ▶ More about the external view
- ▶ **Ambiguities** *Is any ontology or model completely devoid of ambiguity? Probably not*
- ▶ Ten principles

# What about “function”?

- ▶ To some - a *purpose* of a system.
- ▶ In TOGAF and the BSC reference model - a *component*
  - (a logical organisation unit, business function rather than process).
- ▶ **In ArchiMate - a *process***
  - **though ArchiMate confusingly defines both services and components as “units of functionality”**
- ▶ In UML and maths - a *special kind of process*
  - that transforms a set of input values to a set of output values without reference to system state.

## Beware that “interface” can be interpreted two ways

- ▶ as a passive structural element – e.g. a menu.
- ▶ as an active structural element via which services are invoked – e.g. a waiter.

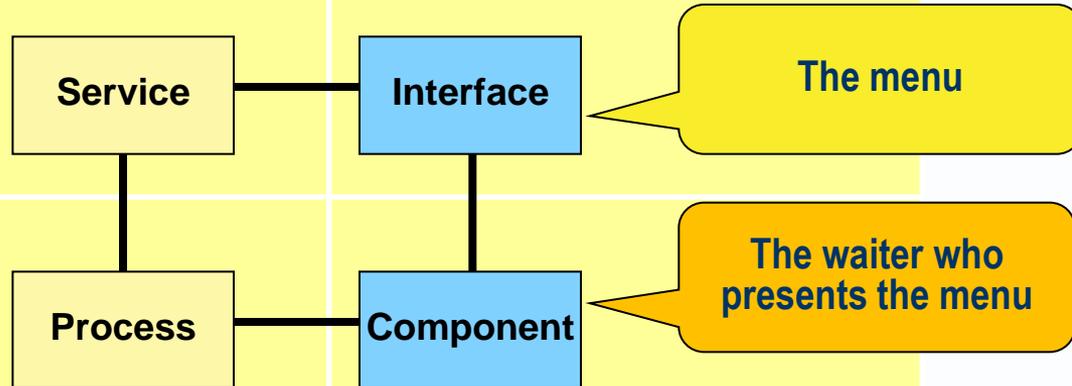


- ▶ The latter is a special kind of component, a facade component, which shows an interface definition to service consumers.

# Interface as a passive structural element – e.g. a menu

Purely descriptive of services offered

	<b>Behaviour</b> what the system does	<b>Structure</b> what the system is made of
<b>External</b> requirements of external entities	<b>Services:</b> operations clients can request to deliver a result	<b>Interfaces:</b> present services accessible by clients.
<b>Internal</b> the workings of the system	<b>Processes:</b> executed step by step to meet service requests	<b>Components:</b> subsystems that execute processes.

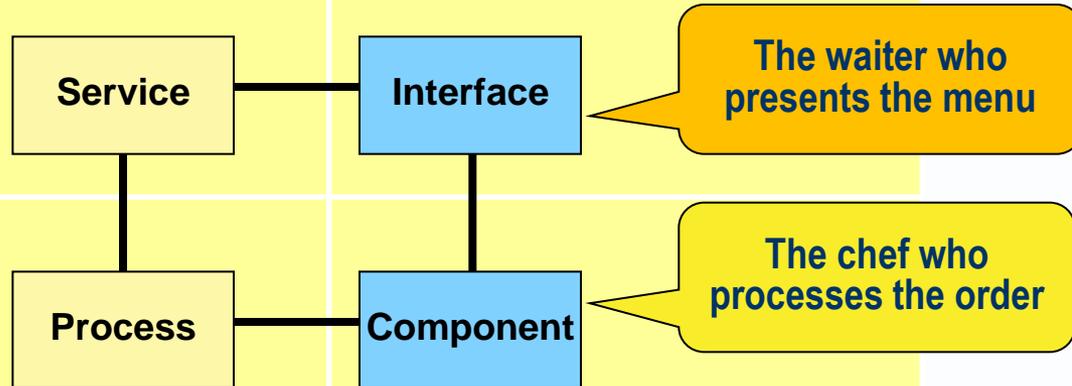
```

graph TD
    Service[Service] --- Interface[Interface]
    Process[Process] --- Component[Component]
    Service --- Process
    Interface --- Component
    
```

A façade component, giving access to services

	<b>Behaviour</b> what the system does	<b>Structure</b> what the system is made of
<b>External</b> requirements of external entities	<p><b>Services:</b> operations clients can request to deliver a result</p>	<p><b>Interfaces:</b> present services accessible by clients.</p>
<b>Internal</b> the workings of the system	<p><b>Processes:</b> executed step by step to meet service requests</p>	<p><b>Components:</b> subsystems that execute processes.</p>



```

graph TD
    Service[Service] --- Interface[Interface]
    Process[Process] --- Component[Component]
    Service --- Process
    Interface --- Component
    
```

# The term Interface can mask several distinct entities

The entities in an architecture meta model should reflect what architect practitioners do or should document as separate artifacts.

## 1. Data flow (not in ArchiMate)

A data container, a transient data store passed from a "sender" component to one or more "receiver" components. E.g. a document, a file, a message,

## 2. Data flow content (cf. an ArchiMate object)

A data structure definable separately from the data flow that contains it. A data entity or aggregate of data entities. Definable as a regular expression, in an XML schema.

## 3. Interface (cf. Interface in BCS sense)

An aggregate of services assignable to one or more "server" components for use by one more "client" components.

## 4. Channel (cf. an ArchiMate Communication Path)

Used to transmit a data flow (1 above) or make a client-server connection (3 above).  
E.g. telephone, HCI, internet, private network.

## 5. Protocol (perhaps an attribute of the above)

One or more layers of protocols needed to send a data flow (1 above) or invoke services (3 above) via a channel (4 above).

- ▶ A service catalogue:
  - Any list of services that is managed/governed.
- ▶ A service directory:
  - A list of services with their addresses and how to find them at run time
- ▶ An interface:
  - A list of services that is required or provided by a system or component
- ▶ A façade
  - An interface that is presented by some kind of broker or mediator component, and through which services can be invoked by a client

- ▶ Introduction
- ▶ Context
- ▶ System elements
- ▶ More about the internal view
- ▶ More about the external view
- ▶ Ambiguities
- ▶ **Ten principles**

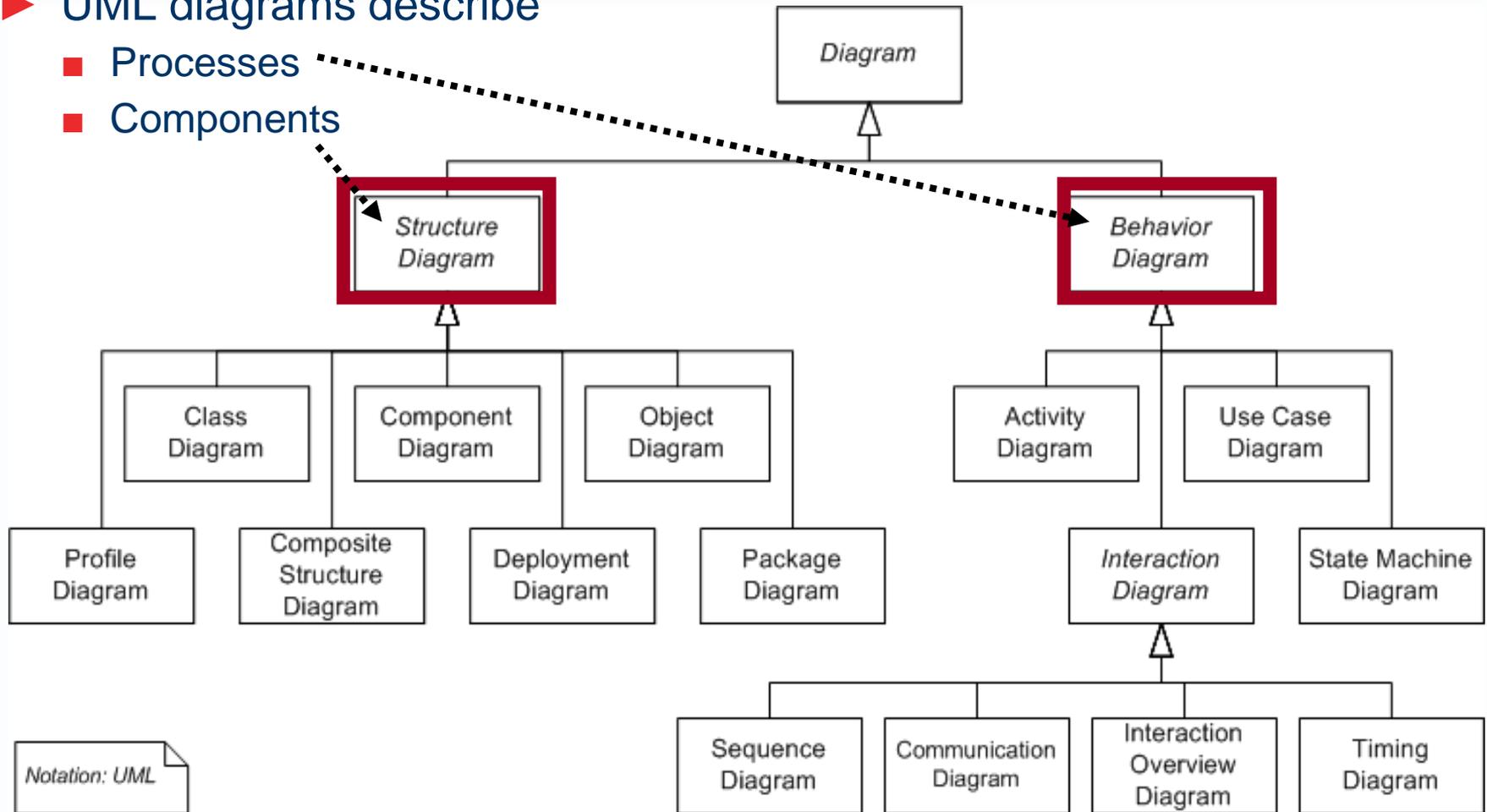
# 1. A system has structural and behavioural properties

- ▶ Materials and activities can be categorised as structural and behavioural. Material things can do work by performing actions on other material things.

# 1. A system has structural and behavioural properties

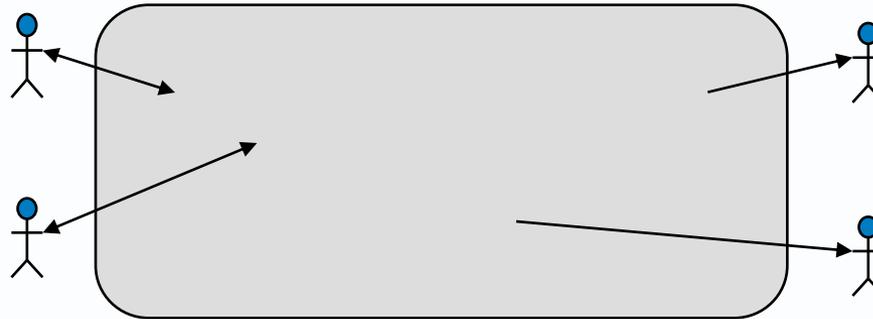
► UML diagrams describe

- Processes
- Components



## 2. A system is encapsulated:

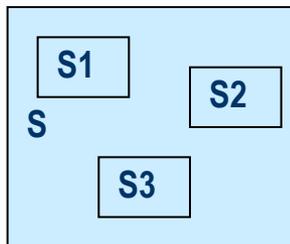
- ▶ An activity system contains structure and behaviour.
- ▶ It is a bounded collection of components.
- ▶ It is also a bounded collection of processes that transform inputs into outputs.



- ▶ To external entities, the purpose and value of the system is in its outputs.

### 3. Systems can be composed and decomposed:

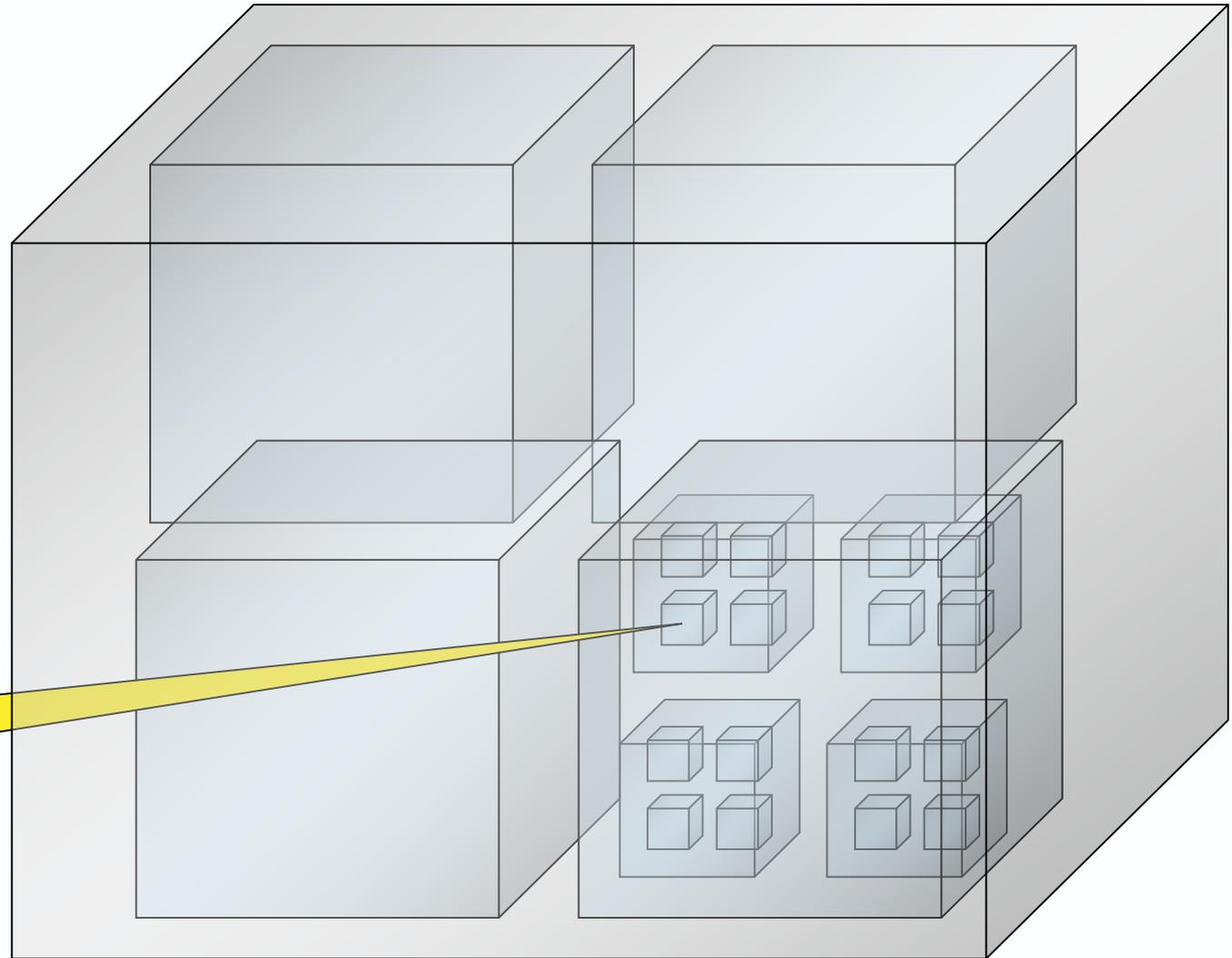
- ▶ One person's system is another's component.
- ▶ Components are composable into bigger (higher level) components, and decomposable into smaller components.
- ▶ Similar principles must be applicable to each level; else architects would have no repeatable methodology.



- ▶ In the systems we design, human components are indivisible, but they can be grouped into teams.

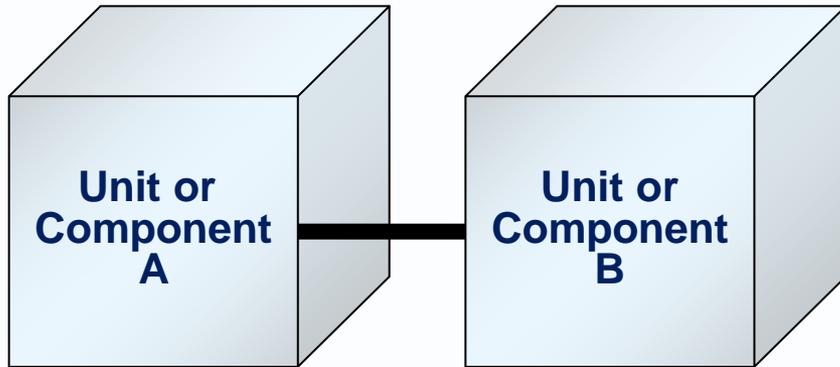
# Decomposition of a system's structure

- ▶ Box = unit or component
- ▶ Bigger boxes encapsulate smaller ones

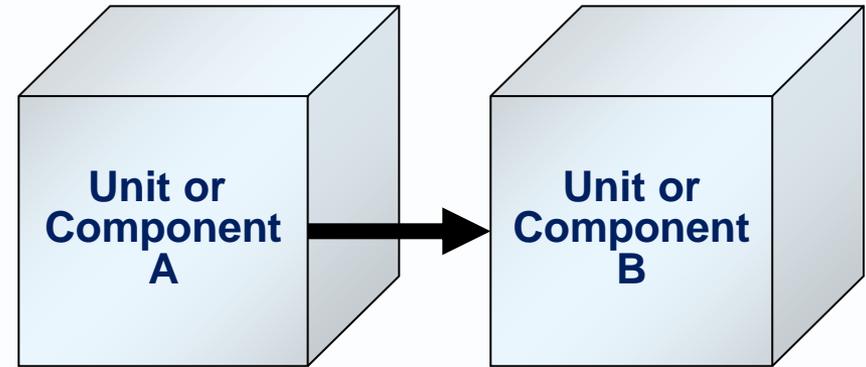


System Decomposition Results in smaller boxes

# What do lines and arrows between components mean?



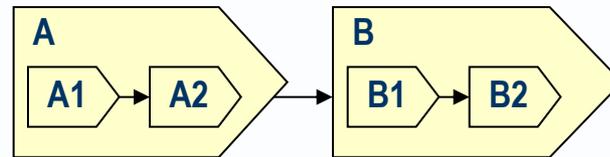
- ▶ **A channel - forming a network**
- ▶ Along which goods, service and/or data can flow
- ▶ (Not a goods, service and/or data flow)



- ▶ **A flow of goods, services or data**
- ▶ Component A does not stop after sending the flow
- ▶ (Not a process flow.
- ▶ Though it might indicate a sequence between an activity within component A and an activity within component B)

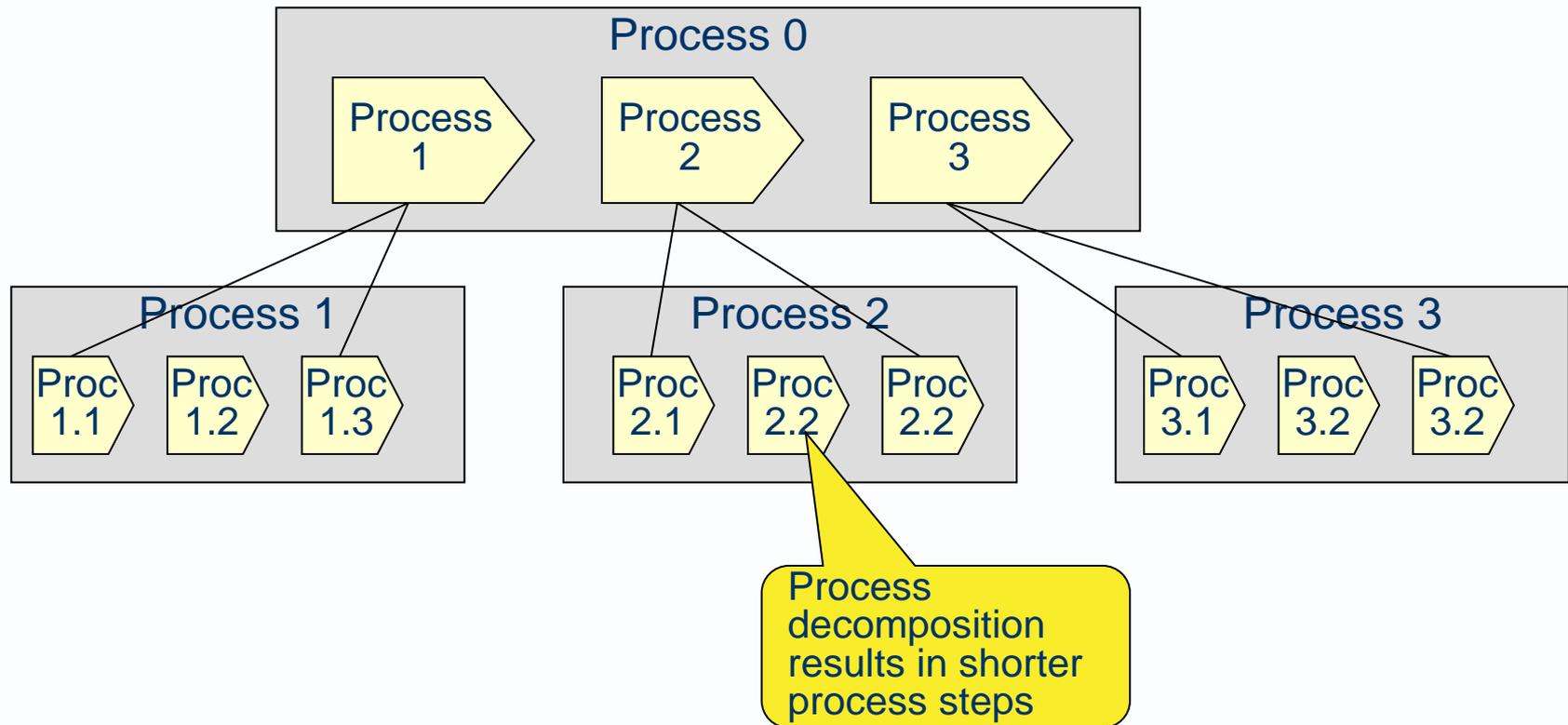
## 4. Processes can be composed and decomposed:

- ▶ Processes are composable into longer (higher level), and decomposable into shorter processes.



# Decomposition of a system's behaviour

- ▶ A process step at one level may be elaborated as a process at the next level down.



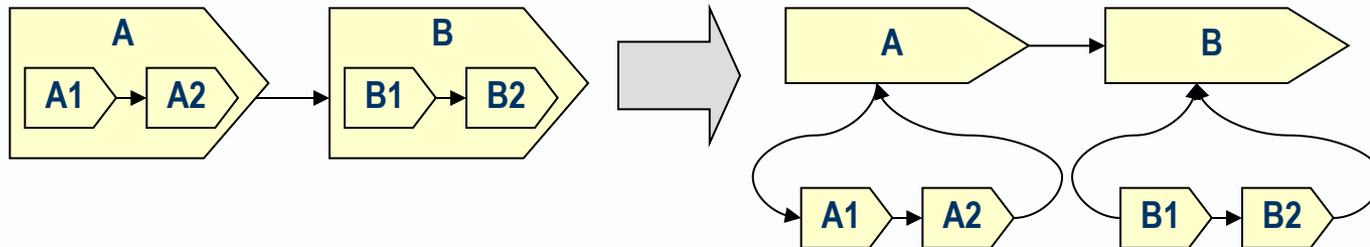
# What does the arrow in a process flow diagram usually mean?



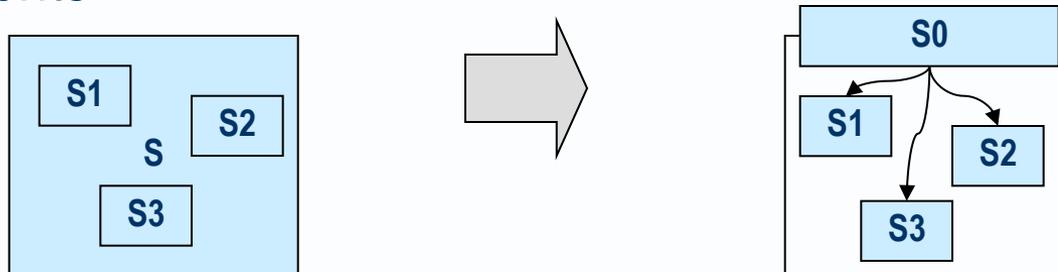
- ▶ **A flow of control, or transition**
- ▶ Step 2 stops when step 3 starts
- ▶ (Not a flow of goods, services or data
- ▶ Though this may happen at the same time)

## 5. Delegation is used to implement decomposition:

- ▶ Higher level processes delegate work to lower level processes.

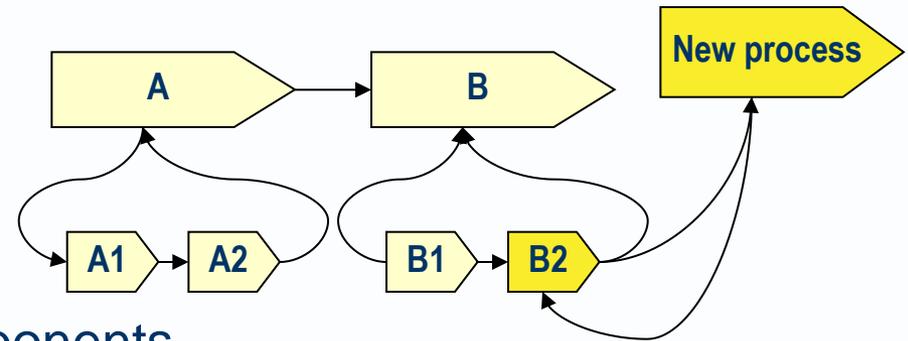


- ▶ Higher level components act as facades to lower level components.

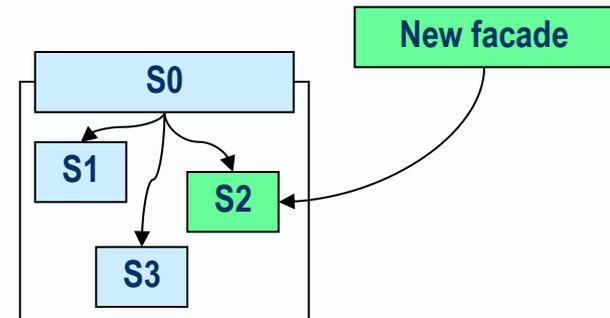


## 6. Delegation is the primary way to achieve reuse:

- ▶ Delegation is used to reuse processes.
- ▶ New long processes can delegate work to old short processes.

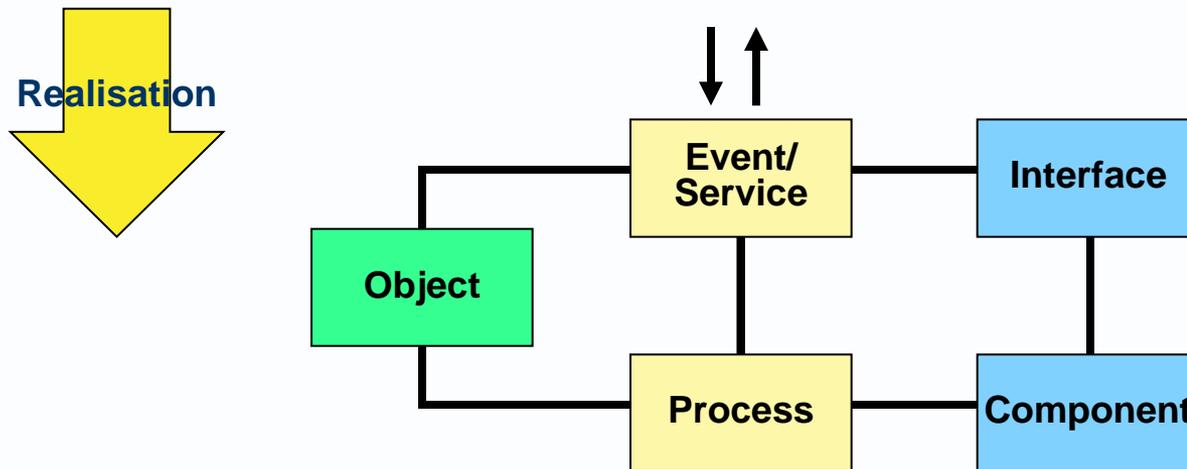


- ▶ Delegation is used to reuse components.
- ▶ New facades request services from old components.



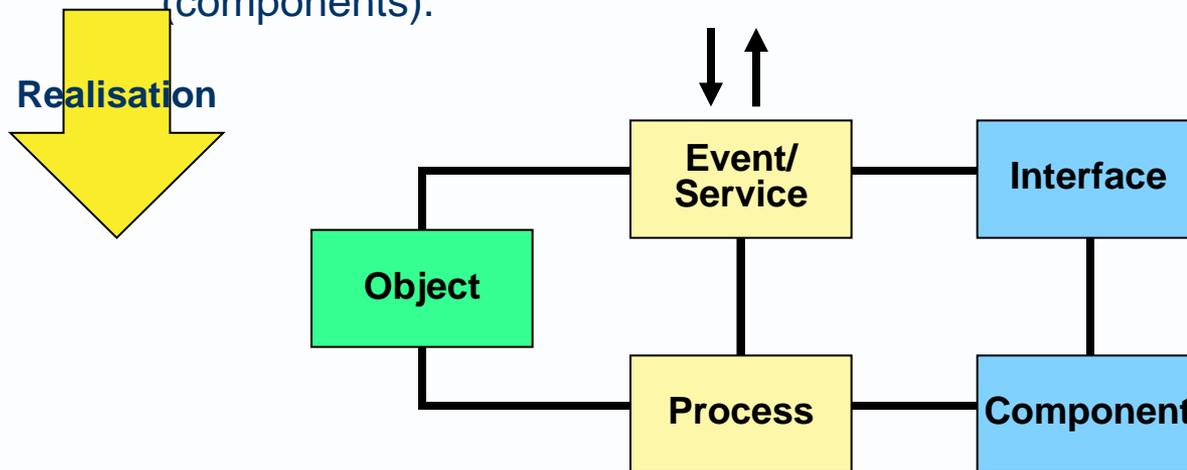
## 7. External precedes internal: “life is recognizable in its expression”.

- ▶ **Realisation:** the external view is implemented by the internal view.
  - services encapsulate and are realised by behaviour elements (processes)
  - interfaces encapsulate and are realised by active structure elements (components).



## 8. Behaviour precedes structure: “form ever follows function”.

- ▶ **Realisation:** the external view is implemented by the internal view.
  - services encapsulate and are realised by behaviour elements (processes)
  - interfaces encapsulate and are realised by active structure elements (components).

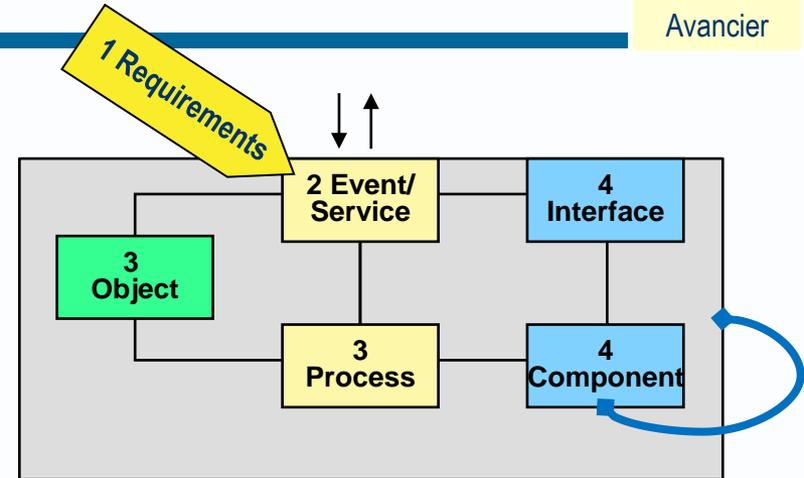


- ▶ **Construction:** behaviour is assigned to structure.
  - services are assigned to interfaces.
  - active behaviour (processes) are assigned to active structure (components).



# The process has a natural sequence

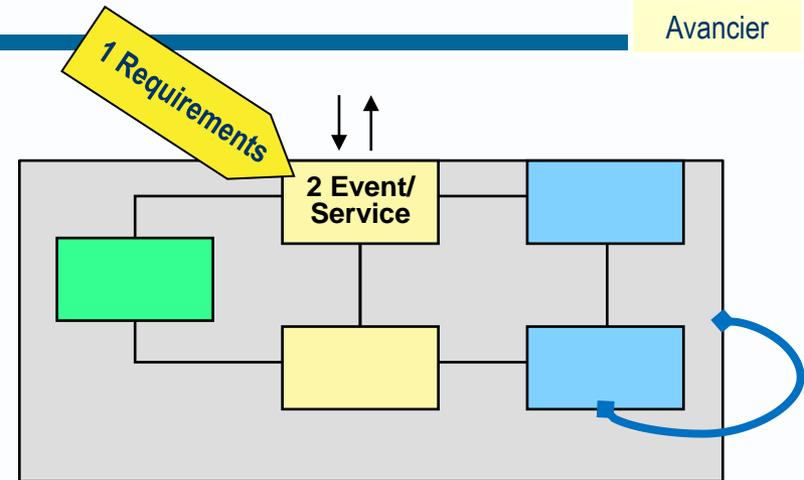
► For each use case and/or subsystem



- 1 The desired outcomes or effects
- 2 The external behaviour.
- 3 The internal state and behaviour
- 4 The internal and external structure

# A natural sequence for architecting

- ▶ 1 The desired outcomes or effects
  - the aims, along with other contextual information.
- ▶ 2 The external behaviour
  - the outputs, products or services the system can produce to meet the aims.
  
- ▶ “For external users, only this external functionality, together with non-functional aspects such as the quality of service, costs etc., are relevant.” ArchiMate

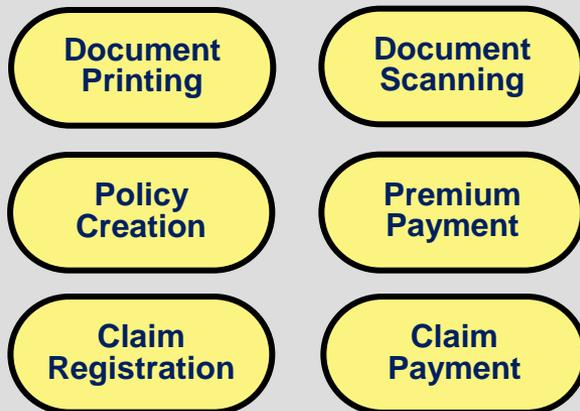


# Definitions of “Service”

## ▶ ArchiMate definition

- “a **unit of functionality** that a system exposes to its environment,
- hides internal operations,
- provides a value,
- accessible through interfaces.”

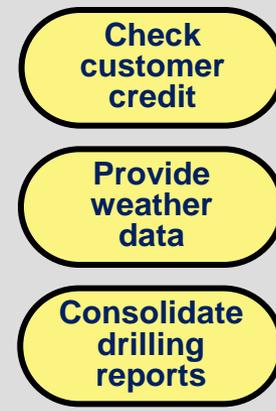
## ▶ ArchiMate examples



## ▶ TOGAF definition

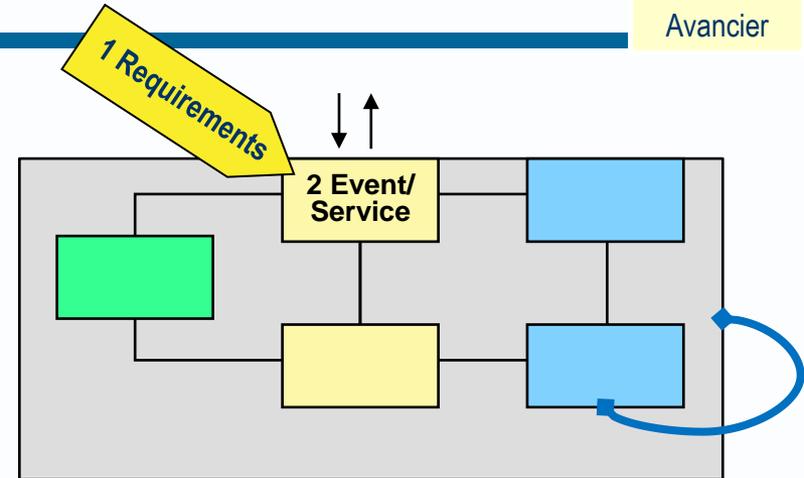
- “an **element of behaviour** that
- provides specific functionality in response to requests from actors or other services”
- “a logical representation of a repeatable business activity, has a specified outcome, is self-contained, is a “black box” to its consumers.”

## ▶ TOGAF examples



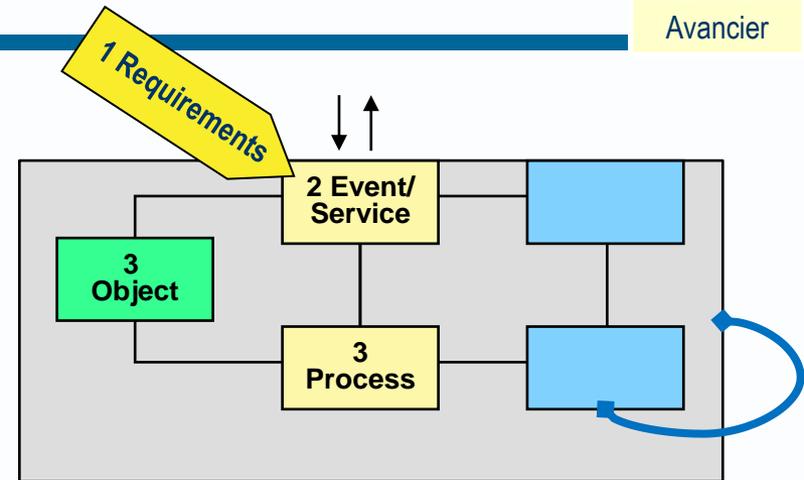
## At this point...

- ▶ Look for system or components that already produce the outputs you want, or near enough.
- ▶ The general principle being
  - Reuse before buy
  - Buy before build
- ▶ But beware before reuse or buy of
  - Non-functional requirements
  - Stored data requirements
  - TCO, including operation and maintenance



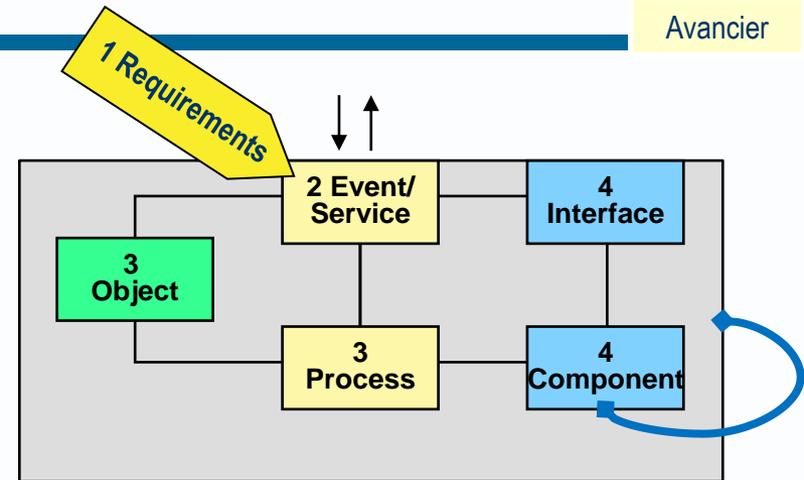
# A natural sequence for architecting

- ▶ 1 The desired outcomes or effects
  - the aims, along with other contextual information.
- ▶ 2 The external behaviour
  - the outputs, products or services the system produces to meet aims.
- ▶ 3 The internal state behaviour
  - the objects to be maintained
  - the processes (scenarios, value streams) needed to produce the outputs



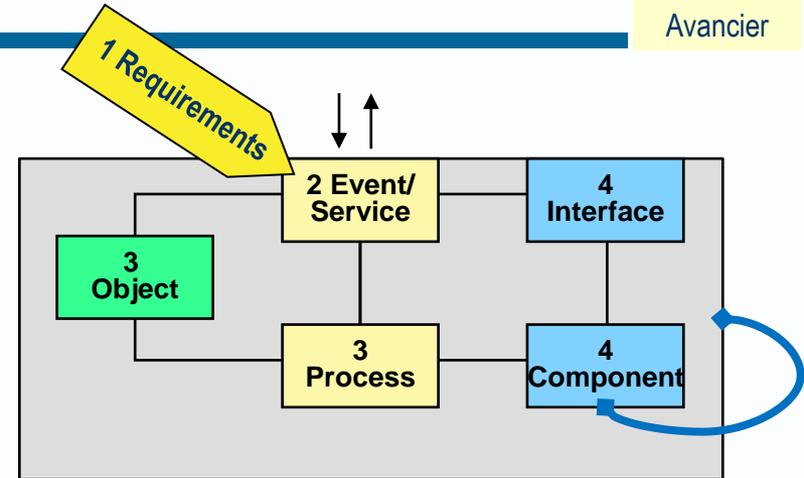
# A natural sequence for architecting

- ▶ 1 The desired outcomes or effects
  - the aims, along with other contextual information.
- ▶ 2 The external behaviour
  - the outputs, products or services the system produces to meet aims.
- ▶ 3 The internal state and behaviour
  - the processes (scenarios, value streams) needed to produce the outputs
- ▶ 4 The structure
  - the components, roles and actors needed to perform processes, with provided and required interfaces



# The process must involve

- ▶ Iteration
  - Interleave thinking about services, processes and components.
  - Decompose systems and processes, then repeat
- ▶ Prioritisation
  - Attend to critical and risky services first
  - If a service looks to be costly, risky or impractical, a change request may be made
- ▶ Requirements change management
  - Continual through the process
- ▶ Governance
  - Architects are governed from above
  - Architects govern system builders



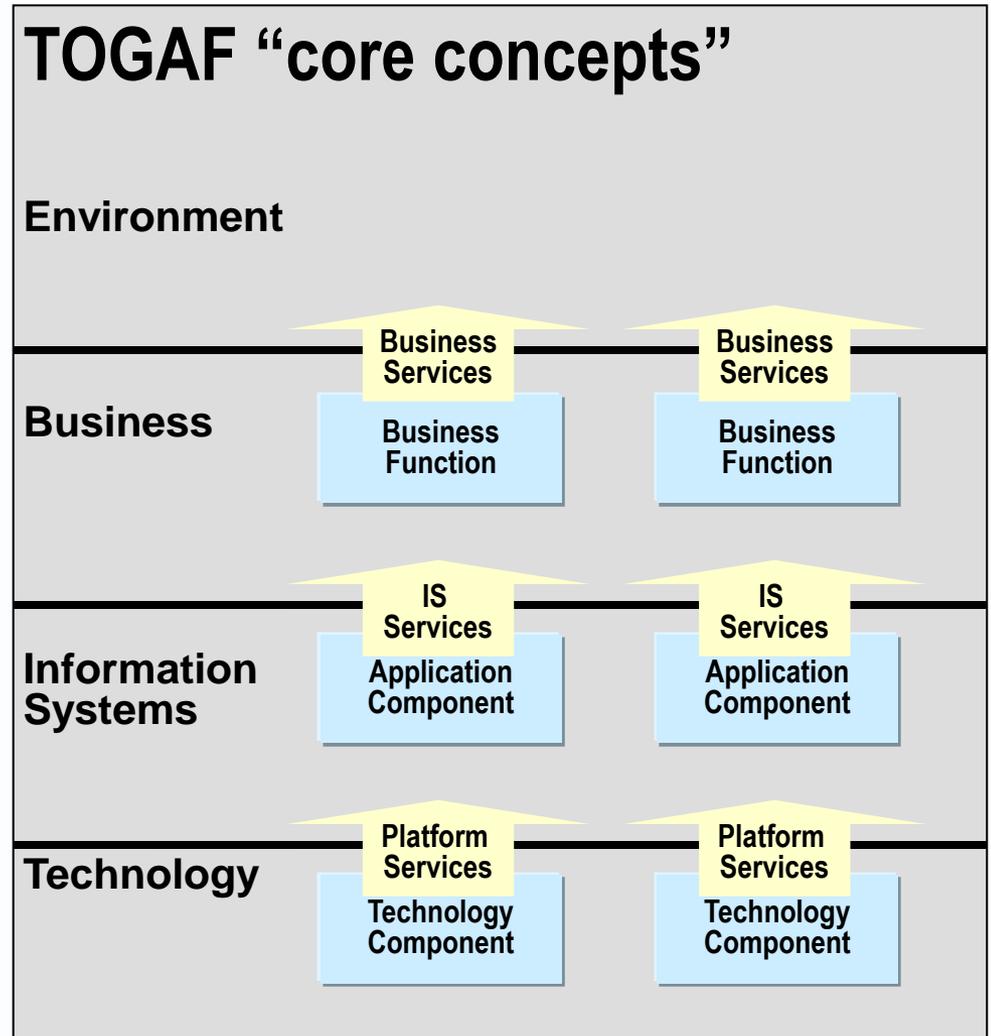
# The process should work from Business to IT

Architects divide human and computer activity systems into views or layers, in which components offer services to the layer above.

► **Business:** essentially business roles and processes

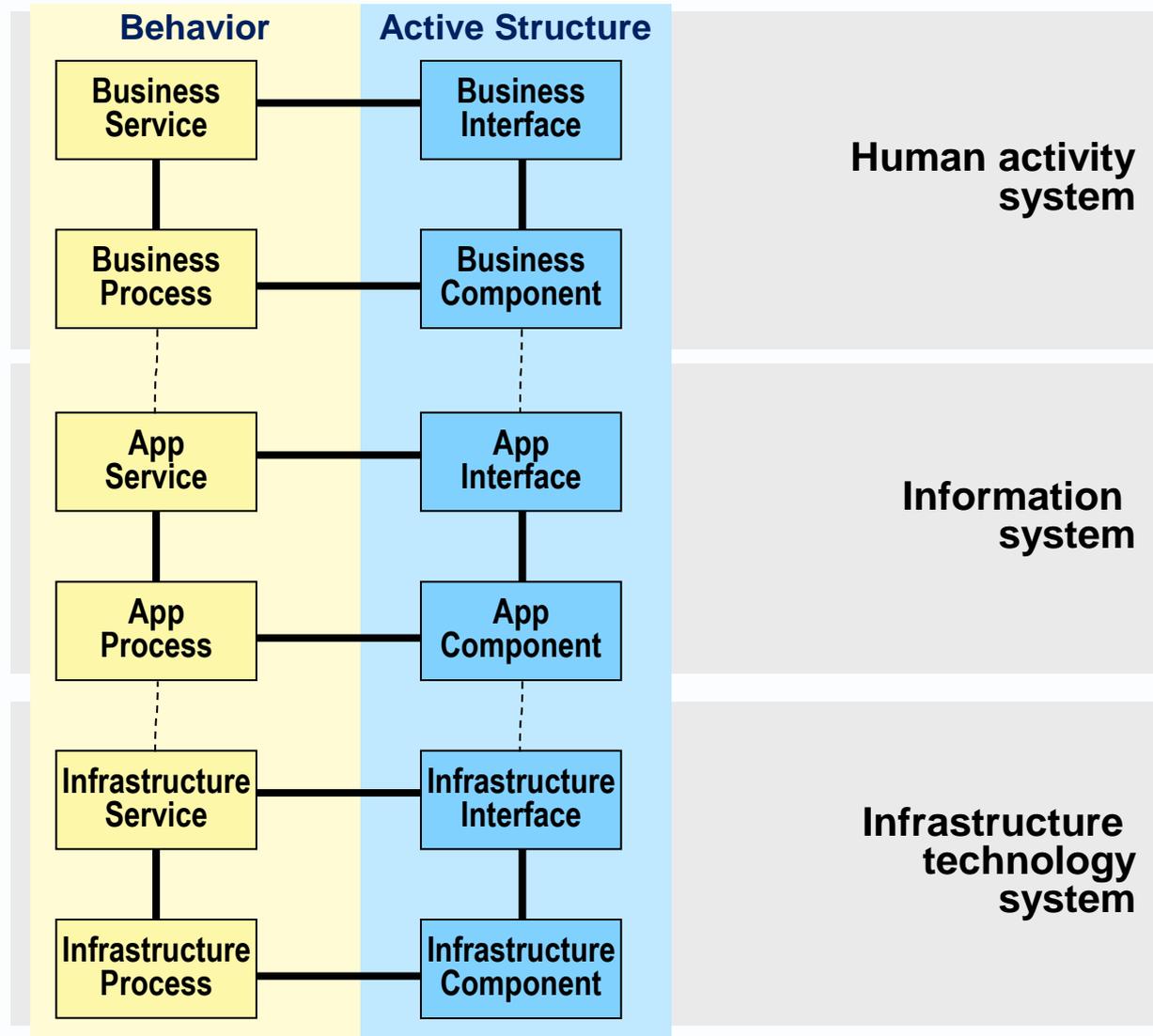
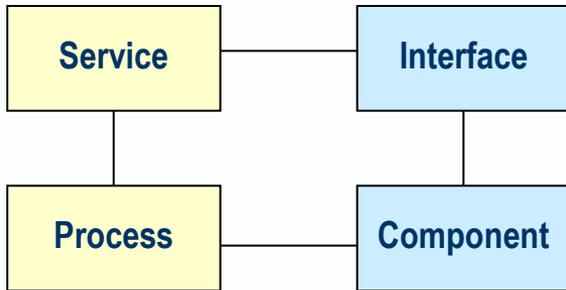
► **Applications:** information systems

► **Infrastructure:** the IT platform for the above.

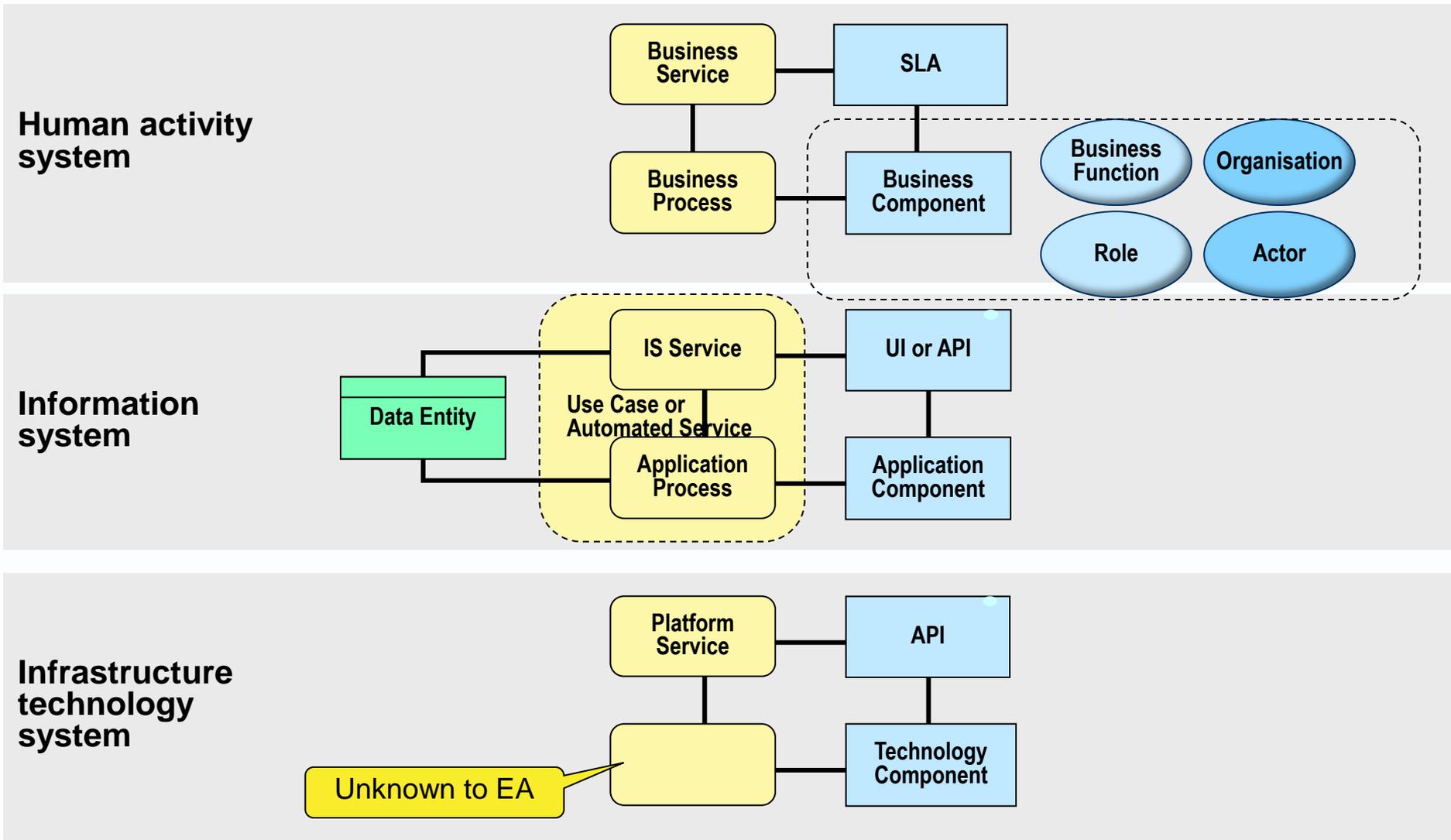


# 9. EA is based on our generic meta model

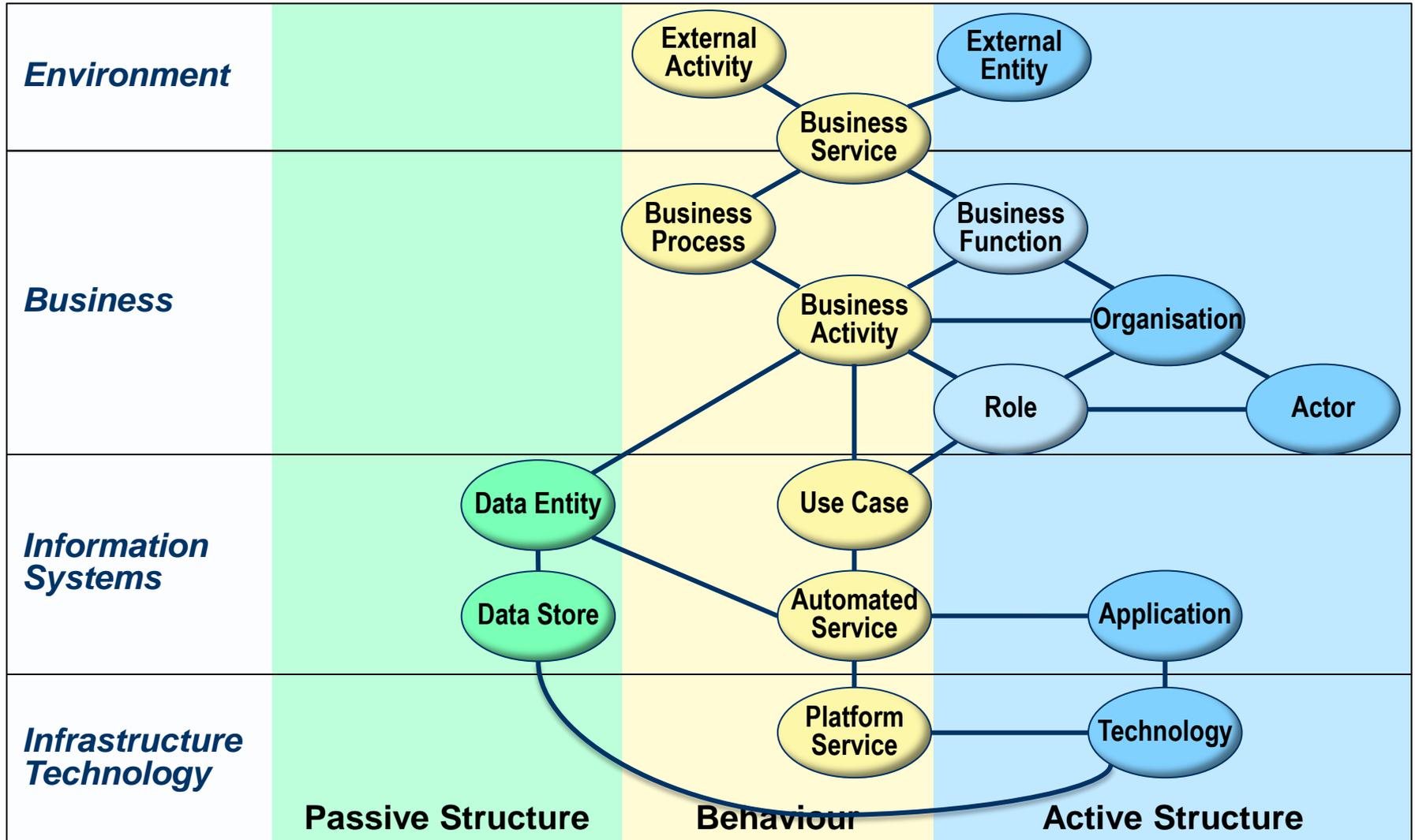
► Each domain/layer has a variant of each generic system element.



# Mapping the generic entities to typical EA entities



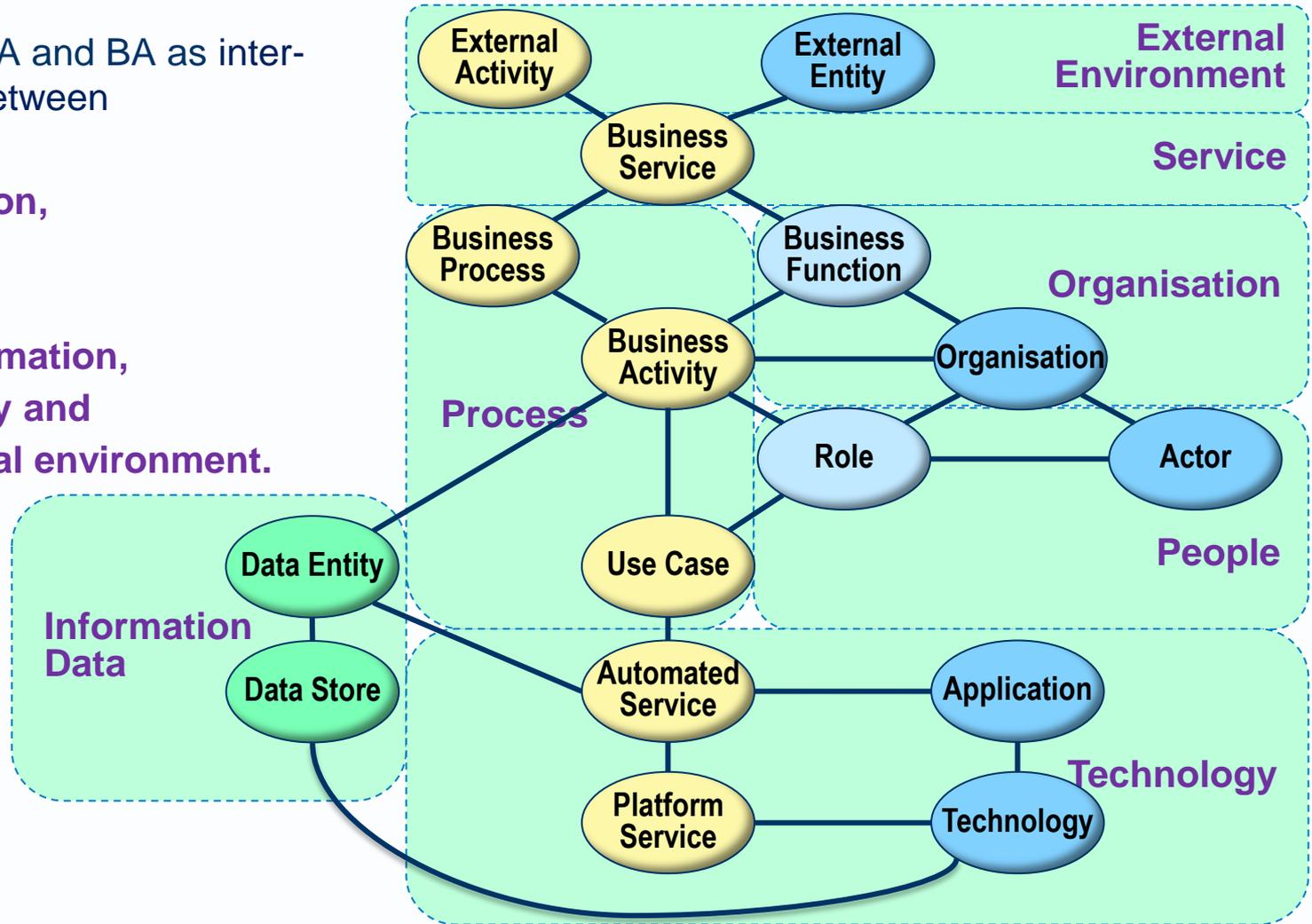
# BCS ESA meta model in the EA framework (after ArchiMate)



# BCS ESA meta model as a refinement of SFIA's definition

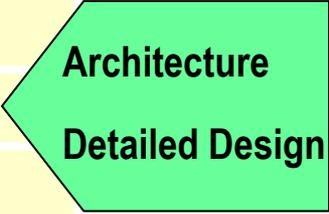
SFIA defines EA and BA as inter-relationships between

- ▶ people,
- ▶ organisation,
- ▶ service,
- ▶ process,
- ▶ data, information,
- ▶ technology and
- ▶ the external environment.



## 10. Architecture descriptions are abstract:

- ▶ Every architect needs to understand **abstraction** because
  - Architecture is more abstract than design and
  - Design is more abstract than implementation.

Omission	Composition	Generalisation	Idealisation	
Vacuous	Coarse-grained composite	Universal	Concept	
Sketchy	Mid-grained composite	Fairly generic	Logical Model	
Elaborate	Fine-grained composite	Fairly specific	Physical Model	
Complete	Elementary part	Uniquely configured	Physical Material	
Elaboration	Decomposition	Specialisation	Realisation	

- ▶ No clear line separates architecture from design, it depends on
  - how high level and abstract your starting point is – and
  - how low level and close to implementation you need to get.

- ▶ All EA frameworks
  - Zachman Framework,
  - DoDAF,
  - MoDAF,
  - FEA and
  - TOGAF
  
- ▶ are centred on the architectural descriptions of enterprise systems.

- ▶ “The Enterprise Architect has the responsibility for architectural design and documentation...
- ▶ an architecture is a formal description of a system, or a detailed plan of the system at component level...
- ▶ an architecture description is a collection of artifacts that document an architecture.”  
TOGAF 9.1

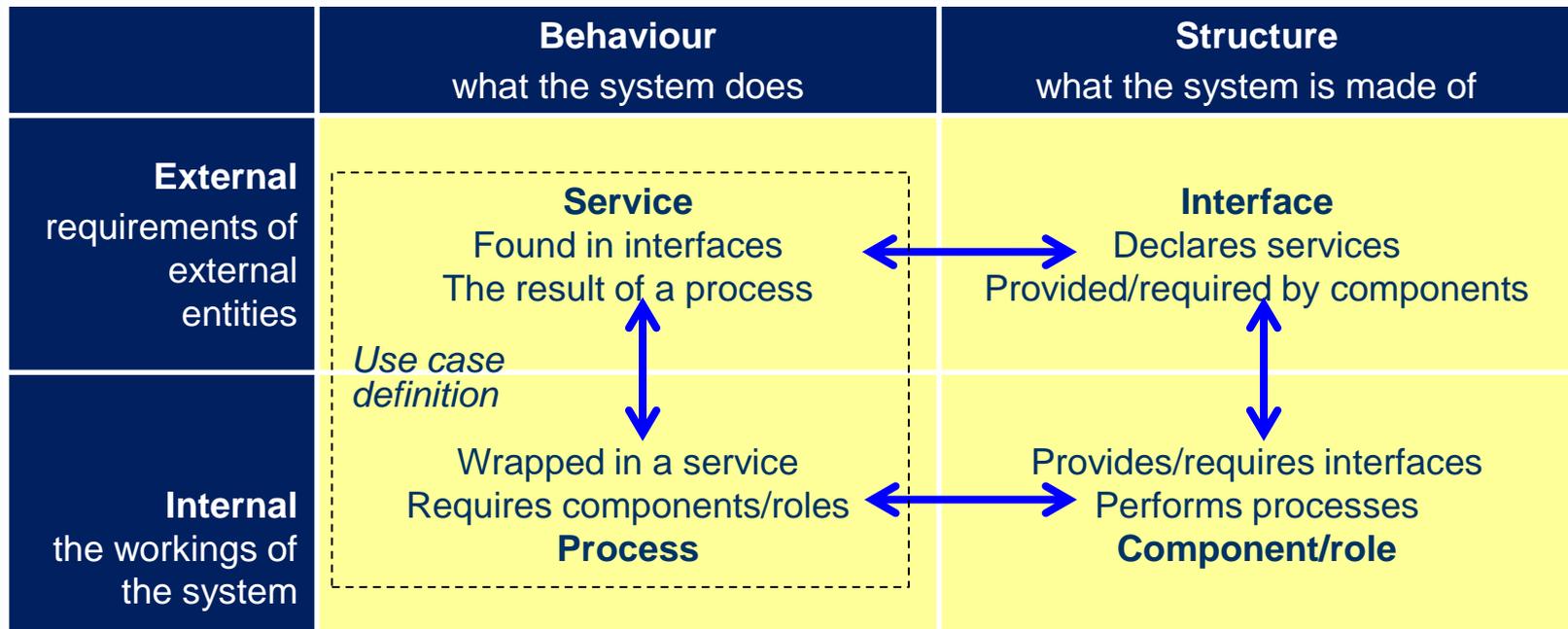
# How are the generic entities describable in EA?



Term	Meaning	Describable in EA using
Event or Service	the external view of the process that responds to a singular event or service request	A service contract
Process	a sequence of activities that responds to an event or meets a service request	A process flow chart
Component	a subsystem that executes process steps.	A component or role definition template
Interface	a list of services that a client can invoke	A service level agreement, a menu of services, a user interface, an API
Object	an item or structure that is used, moved or made by processes	An entity with attributes.

# How the four system elements are documented in practice

- ▶ We could model a system by documenting the system elements separately, with cross-references.



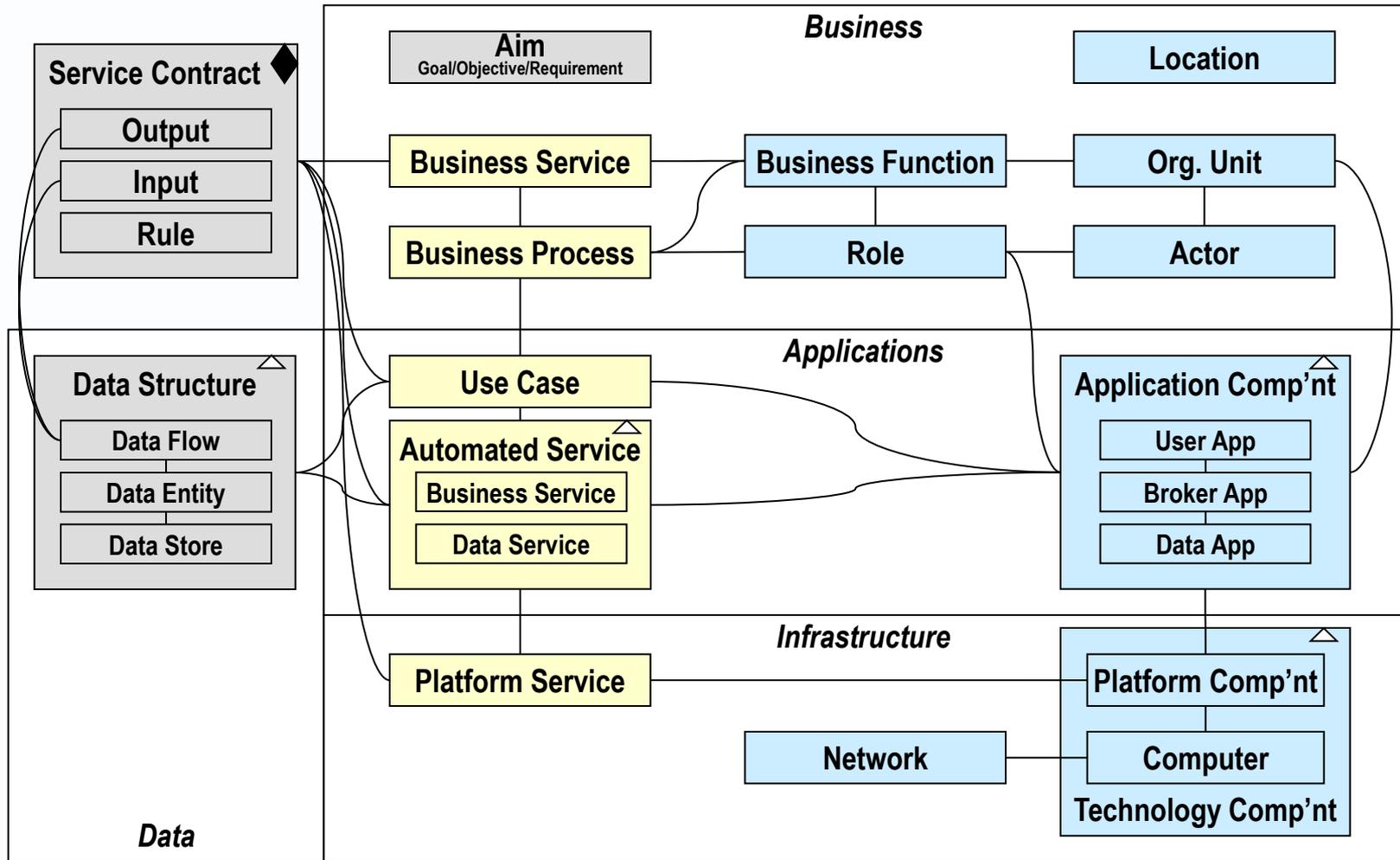
## But we often don't document the elements separately

- ▶ It is often convenient to document one element within another.
- ▶ In practice, elements are documented in aggregate docs
  - A *component* definition
    - includes or names one or more component *interfaces*.
  - An *interface* definition
    - names one or more *services*
  - A *component* definition
    - includes or names one or more performable *processes*.
  - A *process*
    - can be defined as a use case, which
    - wraps up the process flow inside a *service* contract.

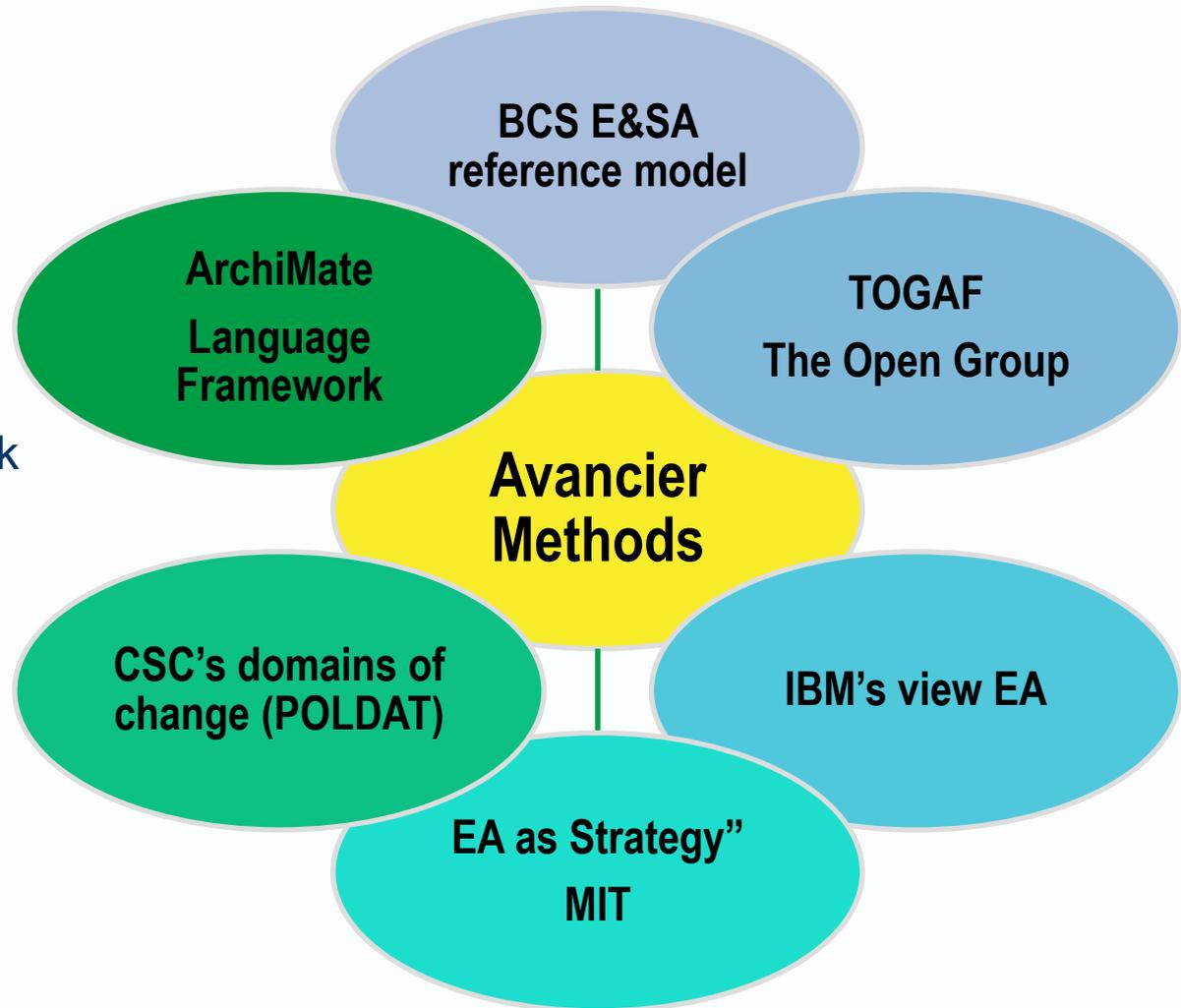
- ▶ Architecture frameworks provide structures and deliverables for architecture documentation.
  - Deliverables (documents)
  - Artefacts (tables and diagrams)
  - Entities (see below)

Context	Force (Driver, Mission, Vision)	Directive (Principle, Policy, Rule)	
	Requirements and behaviour	Logical structure	Physical structure
Business	Aim (Goal, Objective, Requirement)		Location
	Business Service	Business Function	Organization Unit
	Business Process	Role	Actor
Applications	IS Service (Use Case, Automated Service)		Application
Data		Data Entity	Data Store
Technology	Platform Service		Technology

# A meta model of key AM concepts



- ▶ are useful with all architecture frameworks that share similar domains and entities
- ▶ <http://avancier.co.uk>



- ▶ Introduction
- ▶ Context
- ▶ System elements
- ▶ More about the internal view
- ▶ More about the external view
- ▶ Ambiguities
- ▶ Ten principles
  
- ▶ Appendices
  - **Illustrations**
  - Thoughts
  - Notes on TOGAF and ArchiMate

# 4 Foundation Concepts

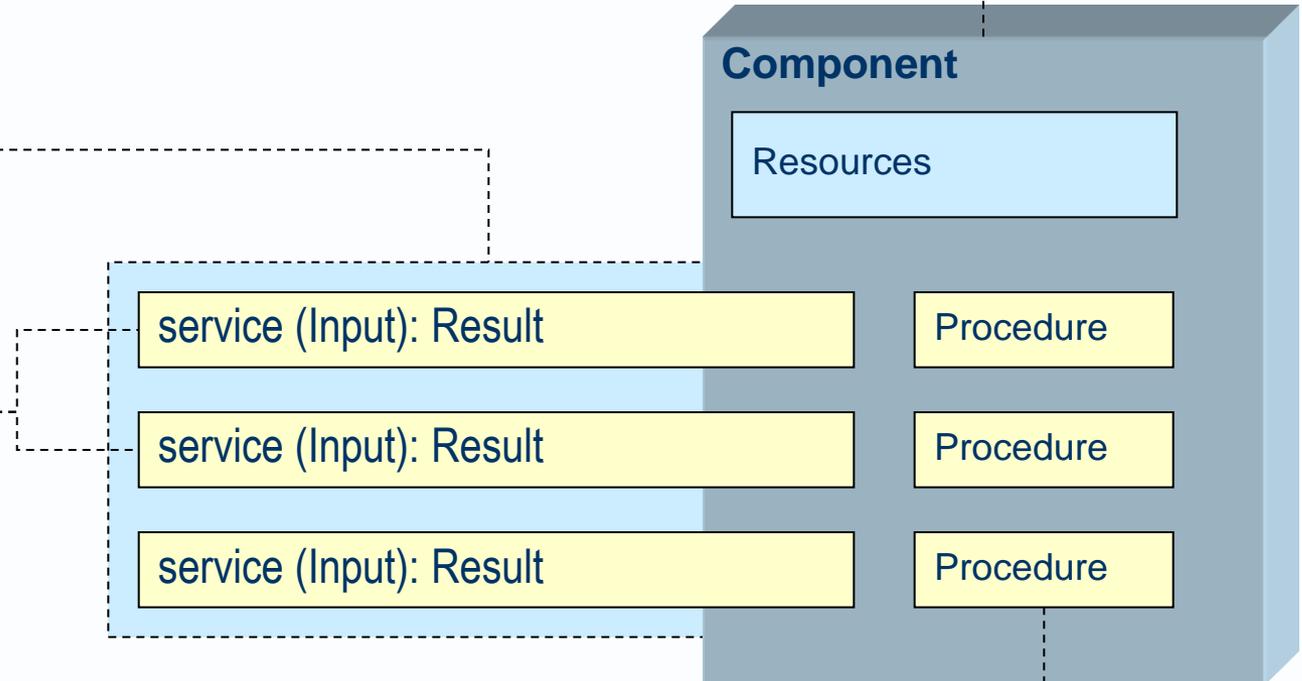
An architect understands and designs the componentisation of systems

▶ **Component**

▶ **Interface**

▶ **Service**  
(service signature)

▶ **Process**  
(service delivery)



# Architects address Business Components

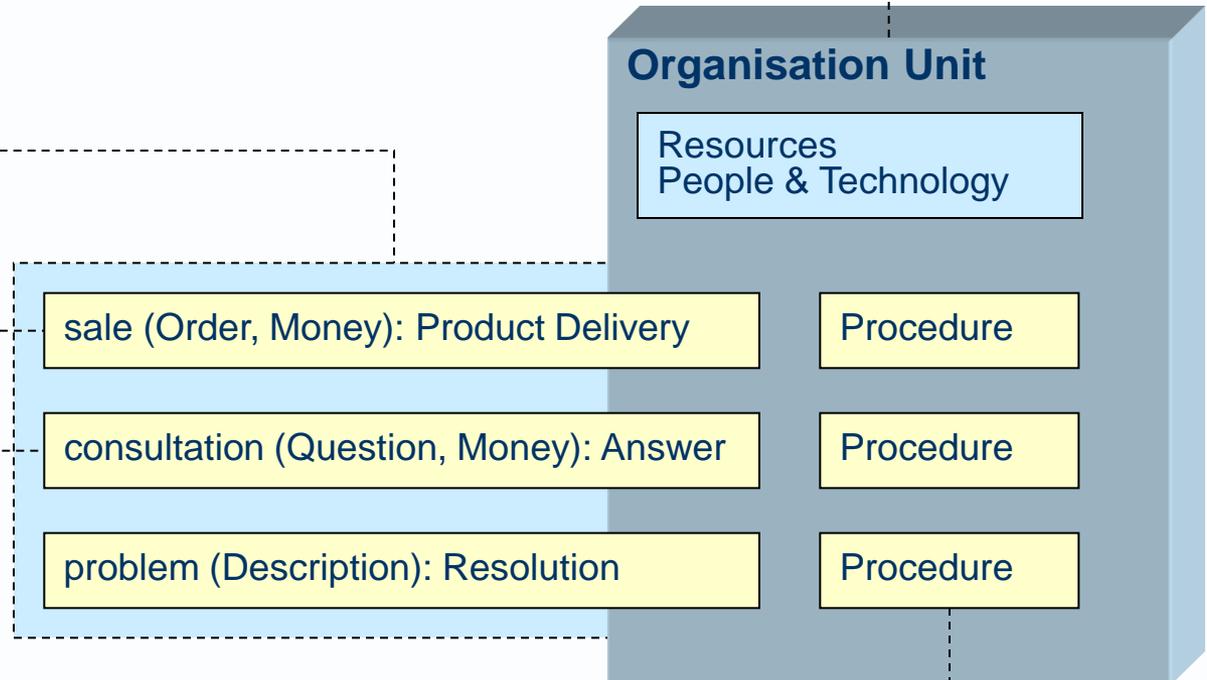
The business architect must understand the services customers want

▶ **Component**

▶ **Interface**

▶ **Service**  
(service signature)

▶ **Process**  
(service delivery)



# Architects address Application Software Components

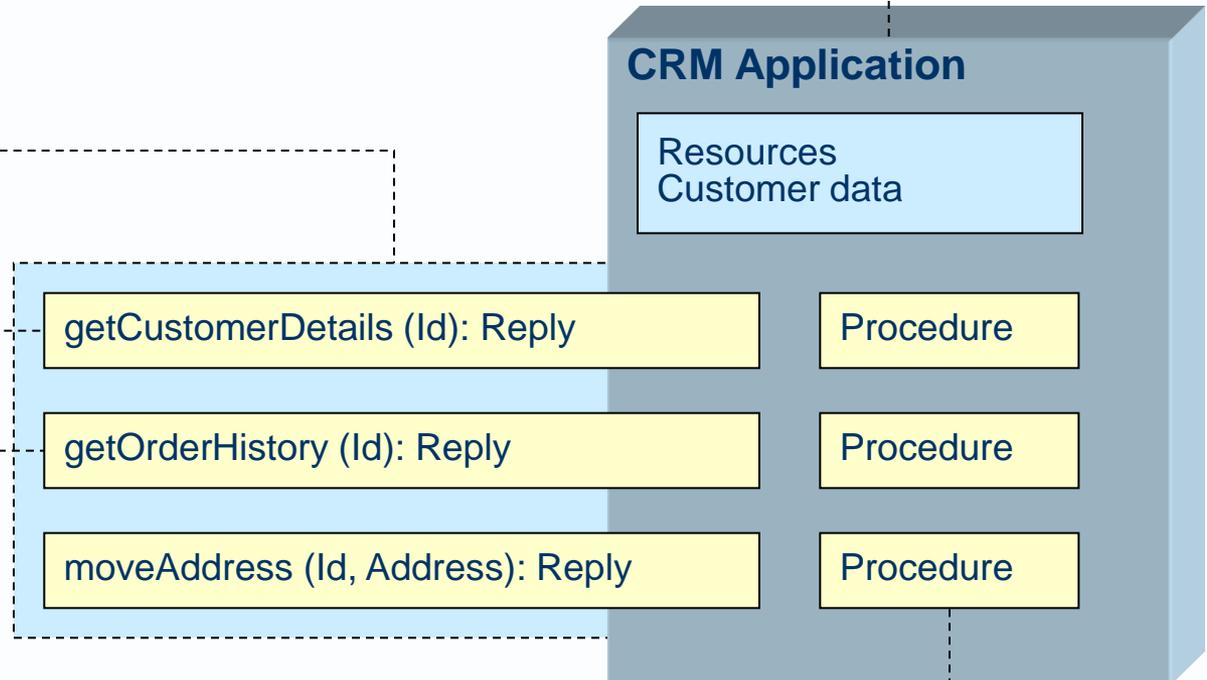
The business architect must understand the application services a business needs.

▶ **Component**

▶ **Interface**

▶ **Service**  
(service signature)

▶ **Process**  
(service delivery)



The 3<sup>rd</sup> service requires the component to find the state for a particular customer, but not to retain the state inside the component once the service is finished.

# Architects address Technology Infrastructure Components

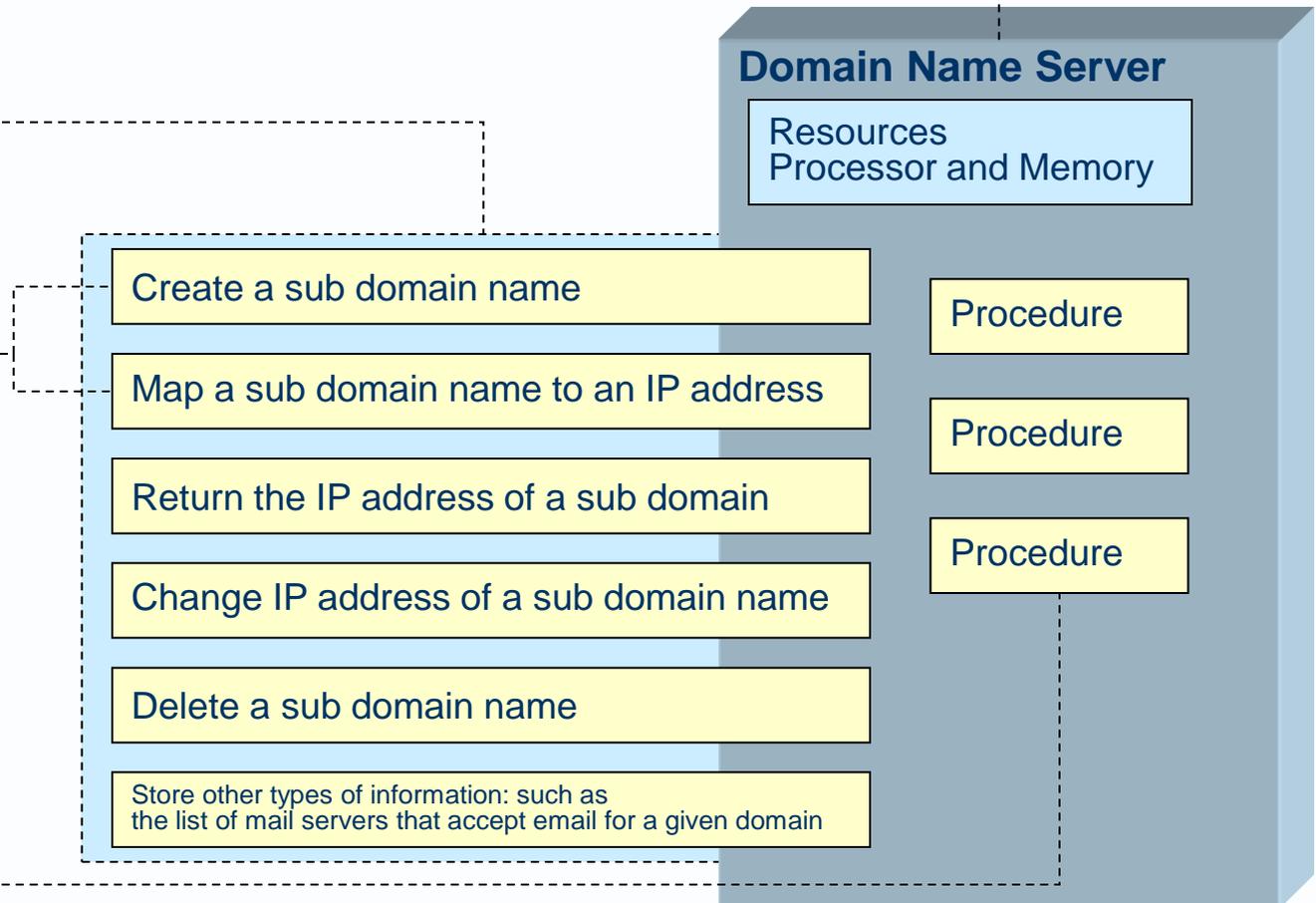
The applications architect must understand the platform services an application needs

▶ **Component**

▶ **Interface**

▶ **Service**

▶ **Process**



# Architects address Technology Infrastructure Components

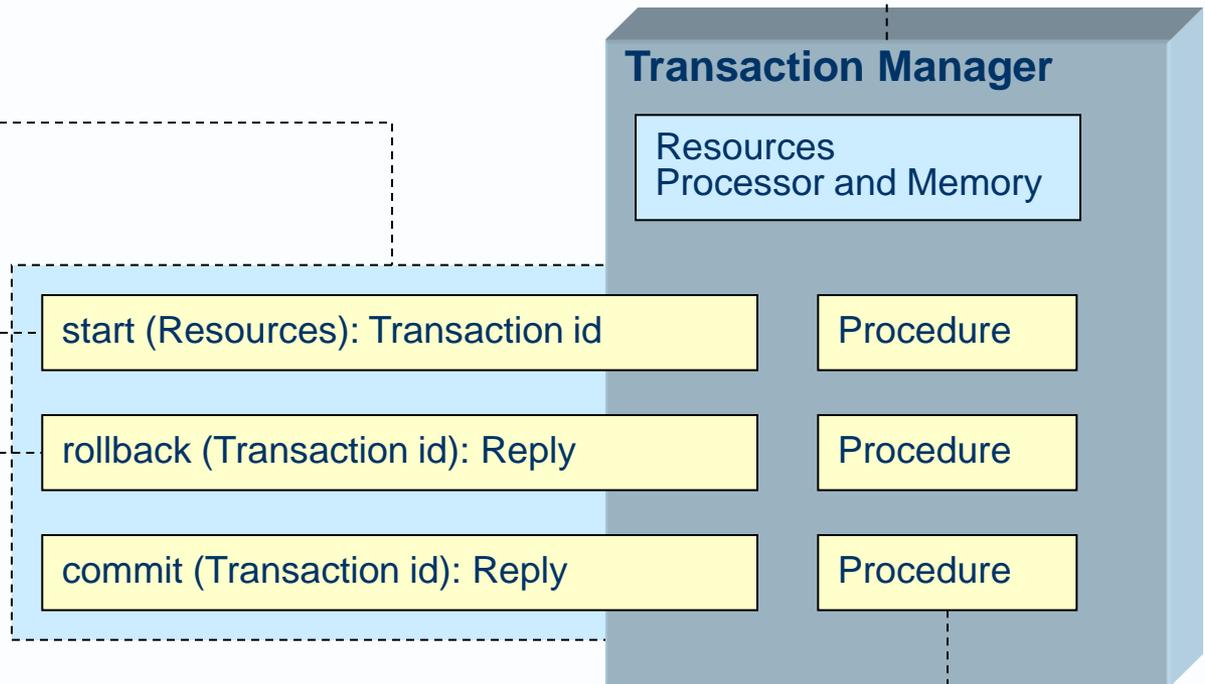
The applications architect must understand the platform services an application needs

▶ **Component**

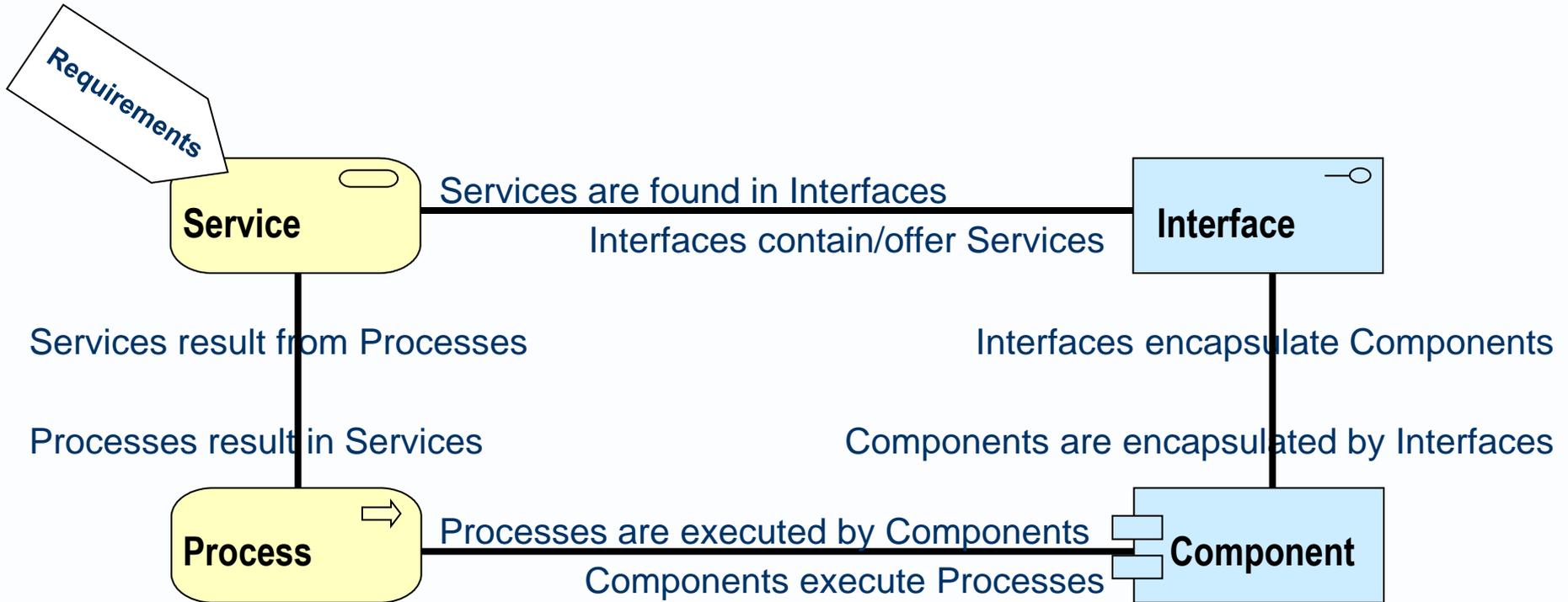
▶ **Interface**

▶ **Service**  
(service signature)

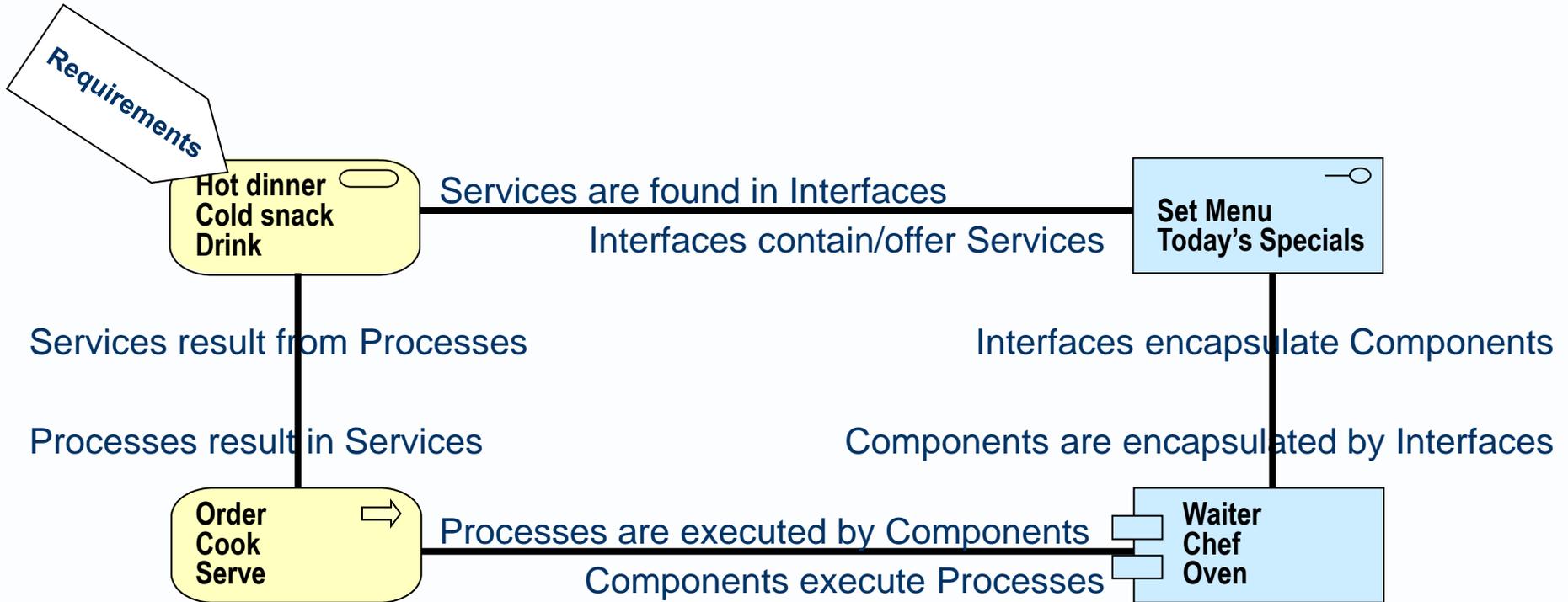
▶ **Process**  
(service delivery)



# Basic activity system concepts - related



# Basic activity system concepts – in a restaurant



- ▶ Introduction
- ▶ Context
- ▶ System elements
- ▶ More about the internal view
- ▶ More about the external view
- ▶ Ambiguities
- ▶ Ten principles
  
- ▶ Appendices
  - Illustrations
  - **Thoughts**
  - Notes on TOGAF and ArchiMate

- ▶ Although an architect is supposed to
  - define system behaviour and
  - facilitate system change.
  
- ▶ Many definitions of architecture fail to mention either
  - behaviour or
  - change.

# Architecture is structure?

- ▶ We set out to design an activity system, which has a structure and many behaviours.
- ▶ The behaviours are required
- ▶ The structure is designed to deliver the behaviours

# Every enterprise or system has an architecture?

- ▶ Every system has *architectural properties*
- ▶ But that does not mean it has an *architecture description*
- ▶ Or that it has been *architected*
  
- ▶ “If it ain't documented, it doesn't exist.” (Adele Goldberg)
- ▶ The mere *idea* that a system has an architecture – undocumented – has no value.
  
- ▶ If a business is successful, or a system operates efficiently, but its architectural properties ain't documented or constrained, then it has no architecture in a sense we can make use of.

# Architecting is evolution?

- ▶ Architecture descriptions are
  - the deliverables of an architect who
  - make decisions and
  - steer designers and builders.
  
- ▶ Evolution implies
  - there is no architect or designer, only
  - a system that changes in tiny incremental steps
  - in tune with a slowly changing environment.
  
- ▶ If a system evolves continually in response to change requests, then though we might document its structure and behaviour in an architecture description, this does not guide or constrain its structure and behaviour.

# Though the architect aims to enable evolution

- ▶ An important role of the architect is to design systems in such a way that they can be further improved and changed through evolutionary design
- ▶ That is, not only to design a change
- ▶ But also to design *for* change
- ▶ Enable business and technical agility

# A process must have more than one input? (Zachman says)

## In human and business activity systems

Execute a will (input = will)

Pay or reject an insurance claim (input = claim form)

Deliver a letter (input = addressed envelope)

Break up a car (input = car)

Translate text from english to french (input = english text)

Generate visa to USA (input = ETSA form) (before this process was fully automated)

## In business DP / information systems (convert data from one form into another)

Generate visa to USA (input = ETSA form) (after this process was fully automated)

Report management information: oranges sold this week in london (input = data warehouse)

Calculate area and circumference of circle (input = radius)

## In computer/platform technology systems

Transmit an email (e.g. by SMTP) (input = email with header)

Sort a file (e.g. Bubble sort) (input = unsorted file)

Start a transaction (input = list of data resources whose state must be remembered)

Store a new record in a database (input = attributes of the entity to be recorded)

## In other technology, biology and chemical systems

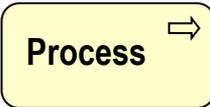
Radio reception (as in radio, TV or wireless router) (input = radio waves)

Play music on record (input = grooves in record)

Crush car (input = car)

Cook meat (input = uncooked meat)

Make cheese (input = milk)

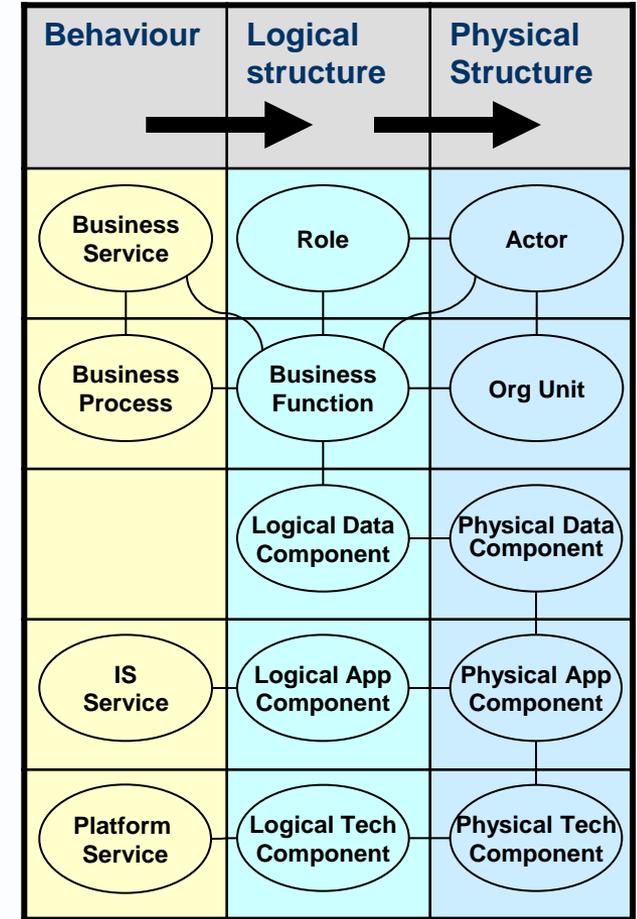
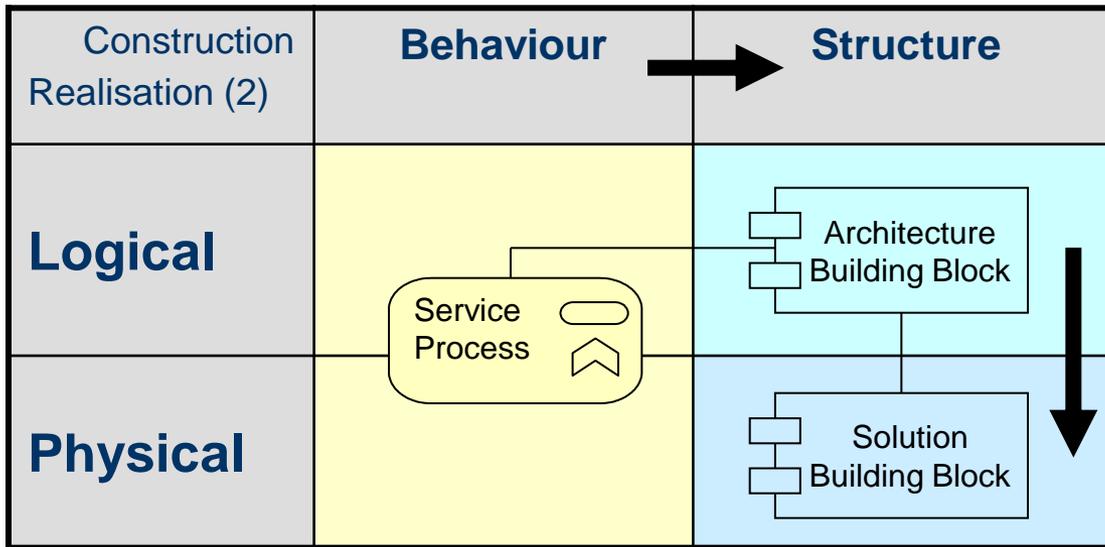


- ▶ Introduction
- ▶ Context
- ▶ System elements
- ▶ More about the internal view
- ▶ More about the external view
- ▶ Ambiguities
- ▶ Ten principles
  
- ▶ Appendices
  - Illustrations
  - Thoughts
  - **Notes on TOGAF and ArchiMate**

- ▶ A **Component** does work.
  - On request, it will execute activities (steps in Processes).
  - It may be hidden behind an Interface.
  
- ▶ A **Process** is a logical sequence of activities.
  - It is executed by Components.
  - It ends up delivering a Service at some level of granularity.
  
- ▶ A **Service** is what a consumer wants.
  - It is defined in a Service Contract, without reference to the logic of any internal Process used to deliver the Service.
  
- ▶ An **Interface** is list of Services, presented so they can be invoked by consumers.
  - It is a facade that hides the workings of internal Processes and Components.

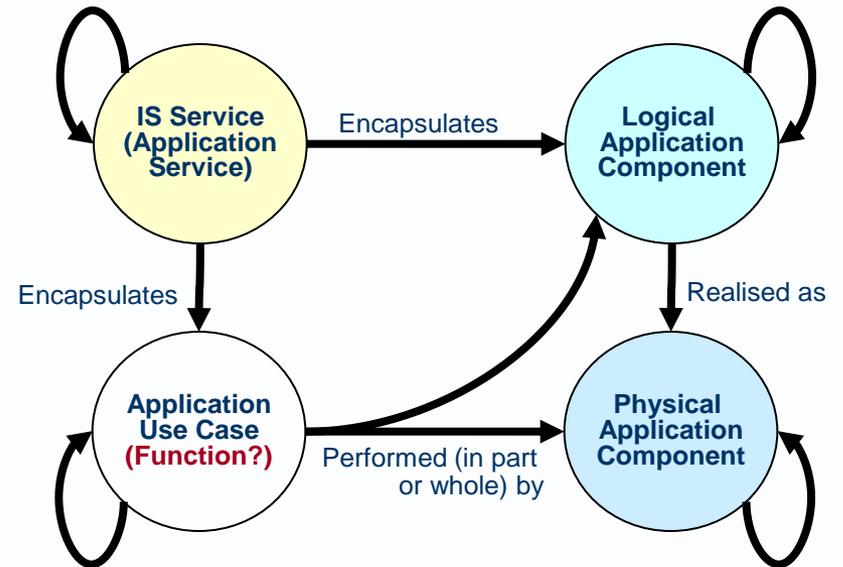
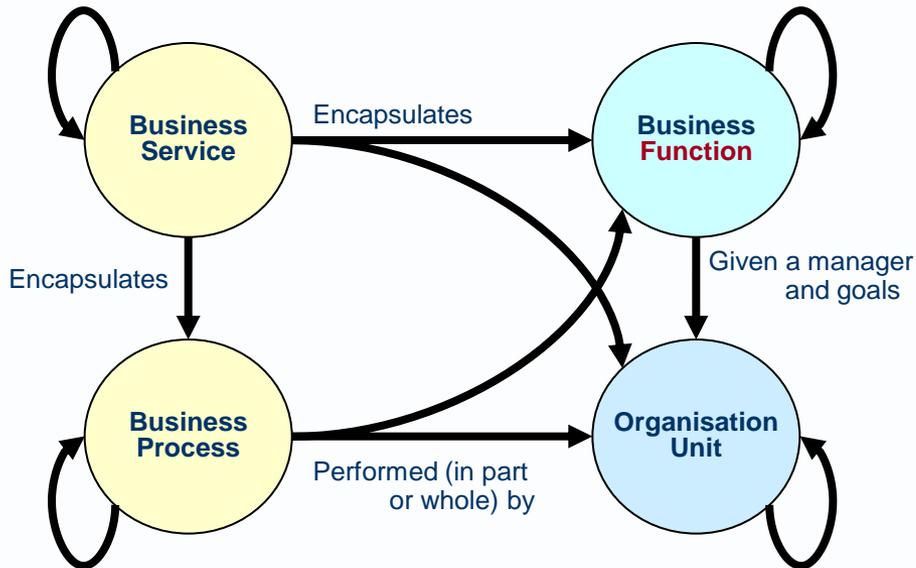
# TOGAF's variation of the generic meta model

- ▶ The TOGAF meta model does not separate Service and Interface
- ▶ It is based on a different logical-to-physical progression, shown in these diagrams



# My reading of core concepts in TOGAF meta model

- ▶ TOGAF treats Business Function as component rather than process
- ▶ TOGAF has no Application Function. (ArchiMate's Application Function is probably better mapped to process (use case) than component.)



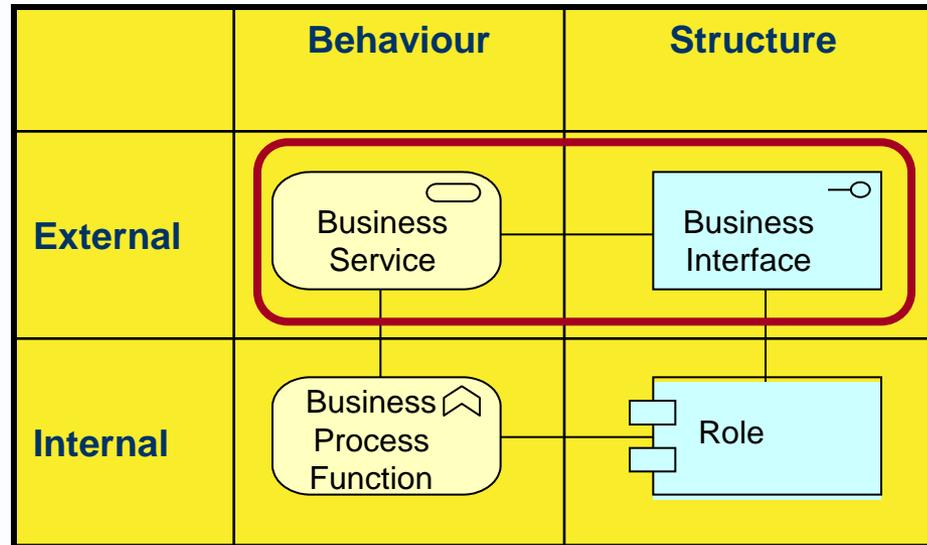
# Does Function = Process or Component?

- ▶ ArchiMate
  - uses Function as a catch-all for any specification of activities - covering both components and processes.
- ▶ The BCS reference model
  - uses Function *only* to mean a component in a Business Function hierarchy (eschews the term otherwise)
- ▶ The TOGAF meta model
  - uses the terms Function and Functionality loosely, however...

TOGAF	Logical		Physical
	Behavioural & External	Structural & Internal	
Business	Business Service	<b>Business Function</b>	Organisation Unit
Application	Information System Service	Logical Application Component	Physical Application Component
Technology	Platform Service	Logical Technology Component	Physical Technology Component

# Alignment of ArchiMate with TOGAF: challenges 1

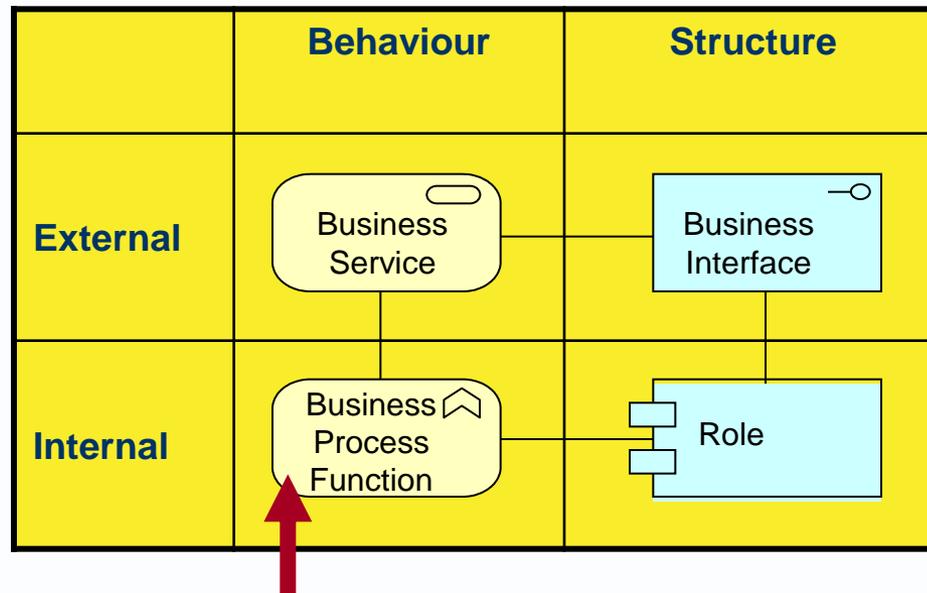
1. The TOGAF meta model does not separate Service and Interface



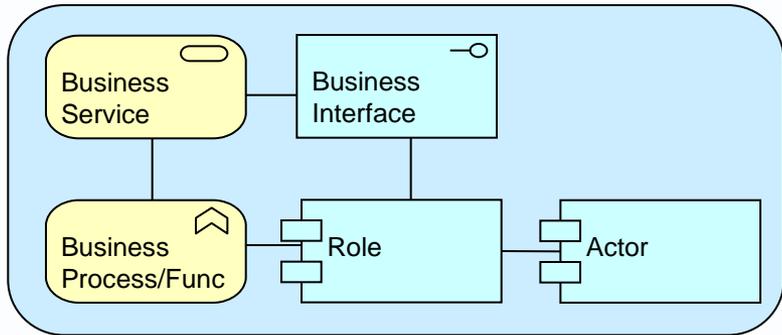
In truth, TOGAF uses Business Function, Business Service and Business Capability almost interchangeably

# Alignment of ArchiMate with TOGAF: challenge 2

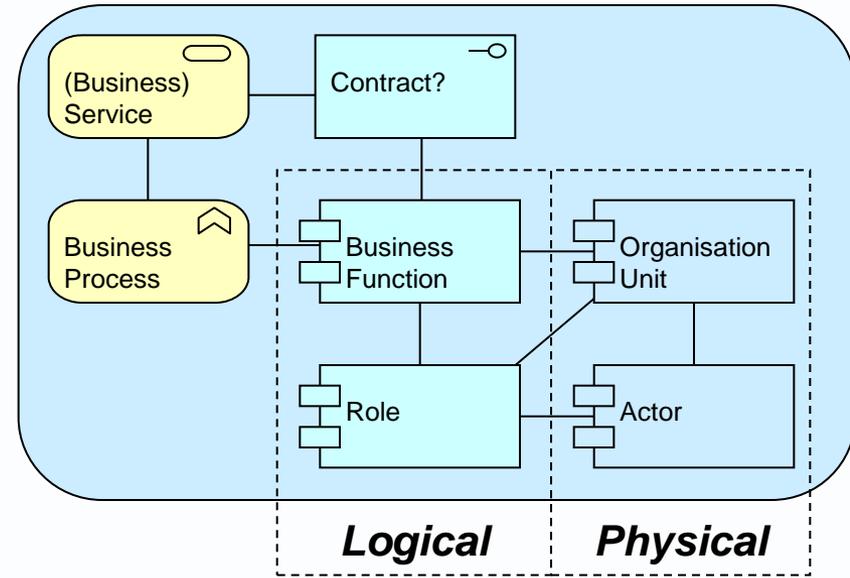
1. ArchiMate's Behavioural Element is a generalisation of TOGAF's Business Process and Business Function (surely therefore both Behaviour and Structure?)



# Alignment of ArchiMate with TOGAF: challenge 3

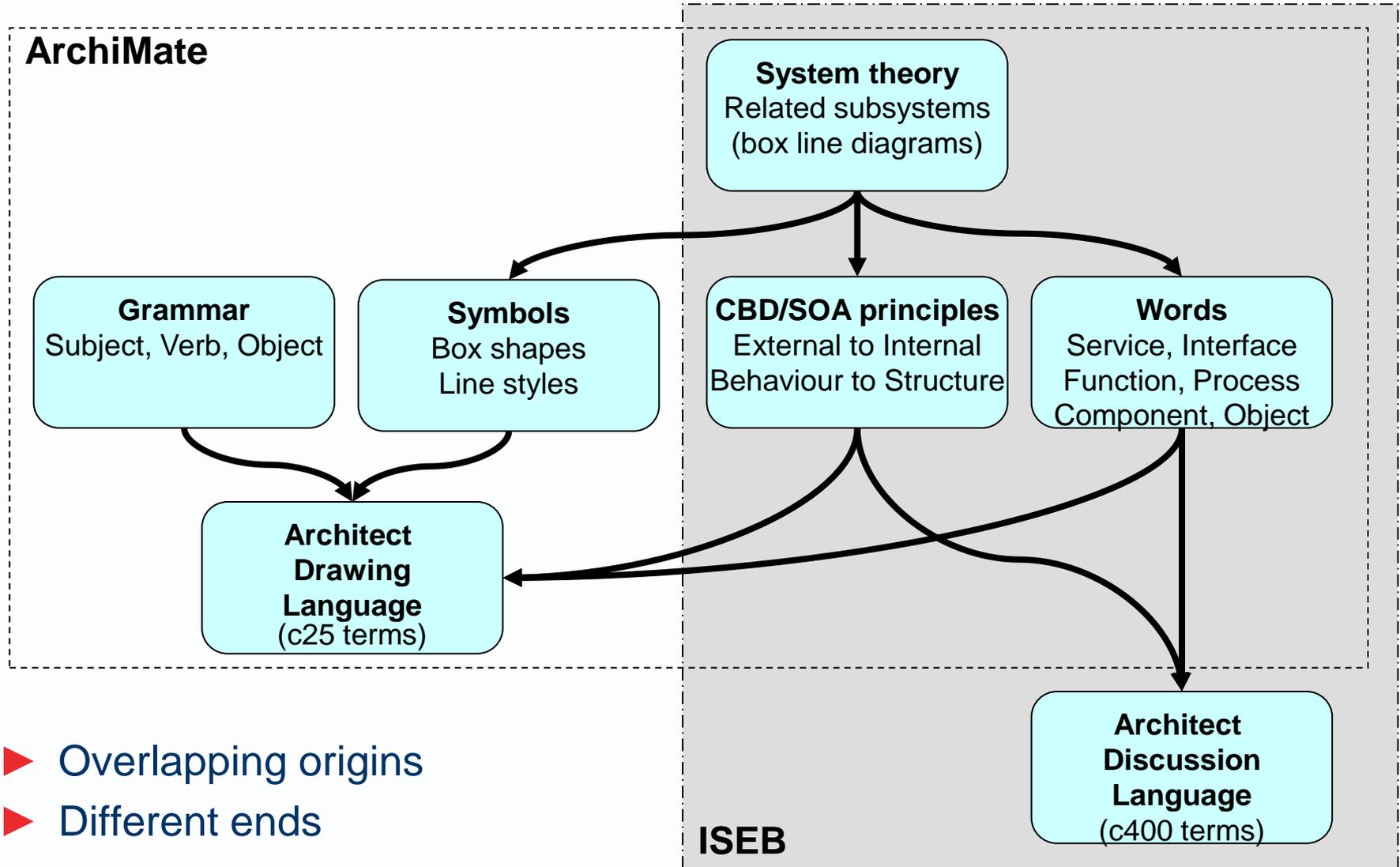


- ▶ ArchiMate does not explicitly separate logical and physical versions of structural components



- ▶ TOGAF explicitly separates logical and physical versions of structural components
- ▶ And maps behaviour to the logical components

# Note that ArchiMate and BCS have different goals



## A few conclusions

- ▶ Aligning architecture methods and standards isn't simple because they contain and/or imply meta models that:
  - Have different scopes
  - Interpret ideas about abstraction in different ways
  - Are based different principles and schema
  - Are correct only within the confines the meta modellers draw.

▶ **Logicity**

- Process threads you will find in various architecture frameworks

▶ **Modularity**

- Foundation concepts and strands in the modelling of human and computer activity systems

▶ **Granularity**

- The challenge of multi-level goals, plans and specifications

▶ **Generic meta model**

- A 4 cell schema for modelling systems, which helps you understand meta models

▶ **Functionality**

- Functions, Organisation Units and Processes in human activity systems

▶ **Architecture meta models**

- Comparing the meta models of industry standard architecture frameworks