

1. order and derivation
2. grouping
3. realisation

TOGAF® Framework and ArchiMate® Modeling Language Harmonization

Aligning core concepts

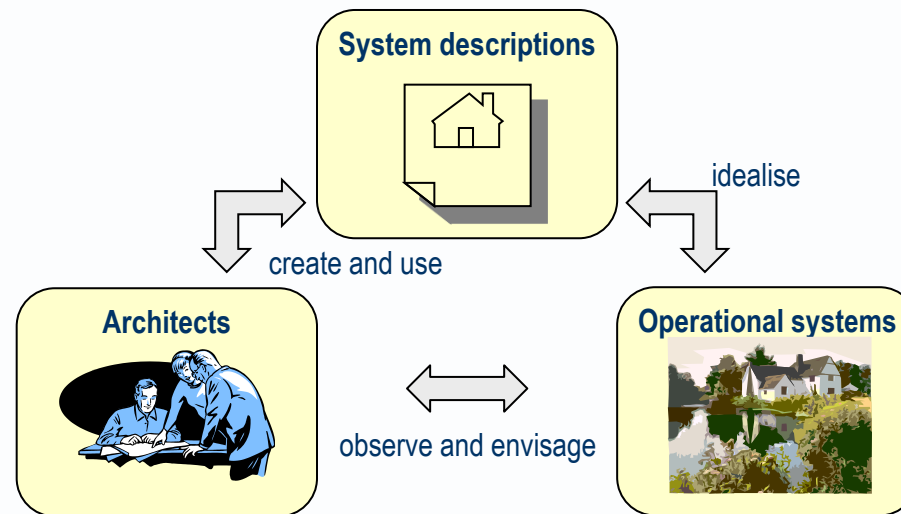
Terms and concepts in service-oriented system description according to TOGAF, ArchiMate and other approaches (DoDAF, WSDL, UML, System Dynamics and social system thinking).

The latest update on research done in 2008 for the British Computer Society

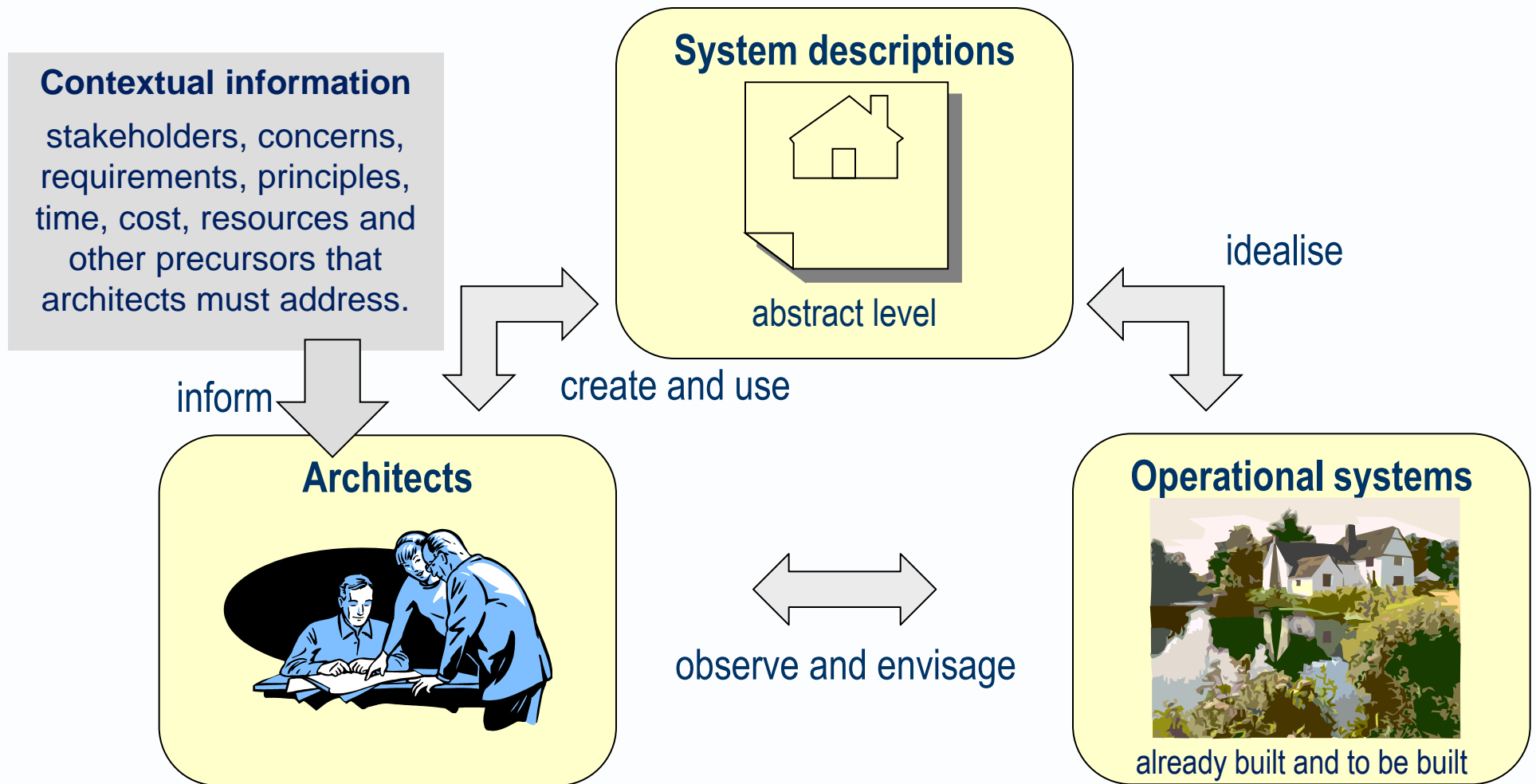
- ▶ Enterprise architecture as a layered system architecture
 - ▶ Service-orientation in ArchiMate and TOGAF
 - ▶ Component-based design
 - ▶ Structural and behavioural properties of a system
 - ▶ Other approaches (DoDAF, WSDL, UML, System Dynamics etc.)
 - ▶ Terminology issues
-
- ▶ This slide show introduces some research done for the BCS into “architecture” sources.
 - ▶ There are ambiguities in and discrepancies between the sources.
 - ▶ And the wider industry terminology is a mess.

TOGAF features c1,400 appearances of the word “system”, inc.

- ▶ “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...”

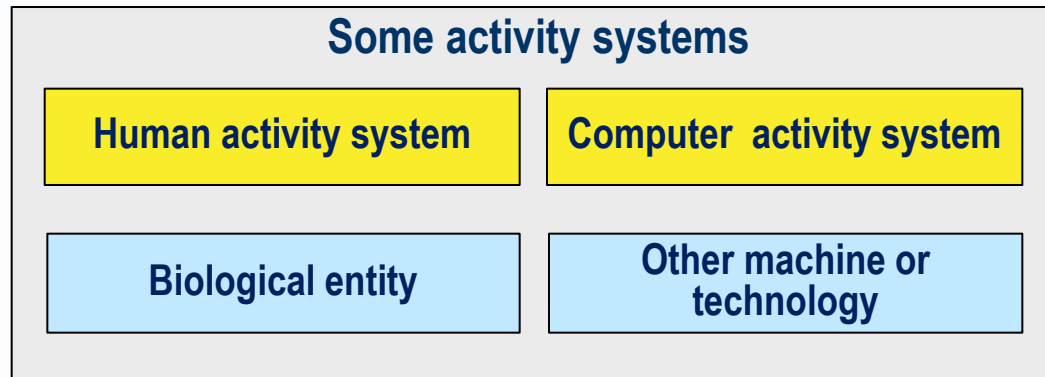


Architects are concerned with



What is a system?

- ▶ Generally speaking, an activity system is a set of
 - **components** that perform **roles** in **processes**
 - to produce **desired effects**
 - by maintaining **system state** and/or
 - producing **outputs** from **inputs**

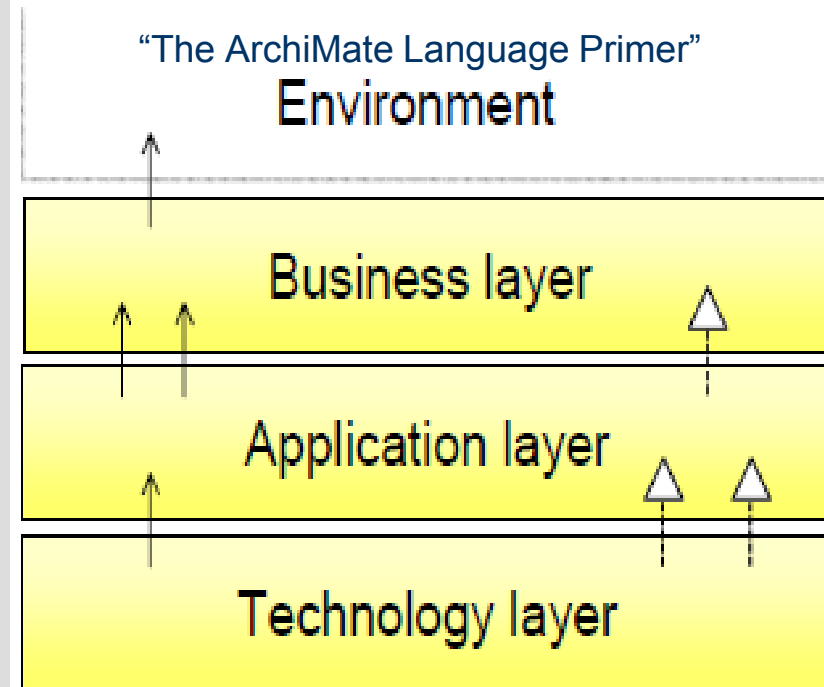


How do we model systems?

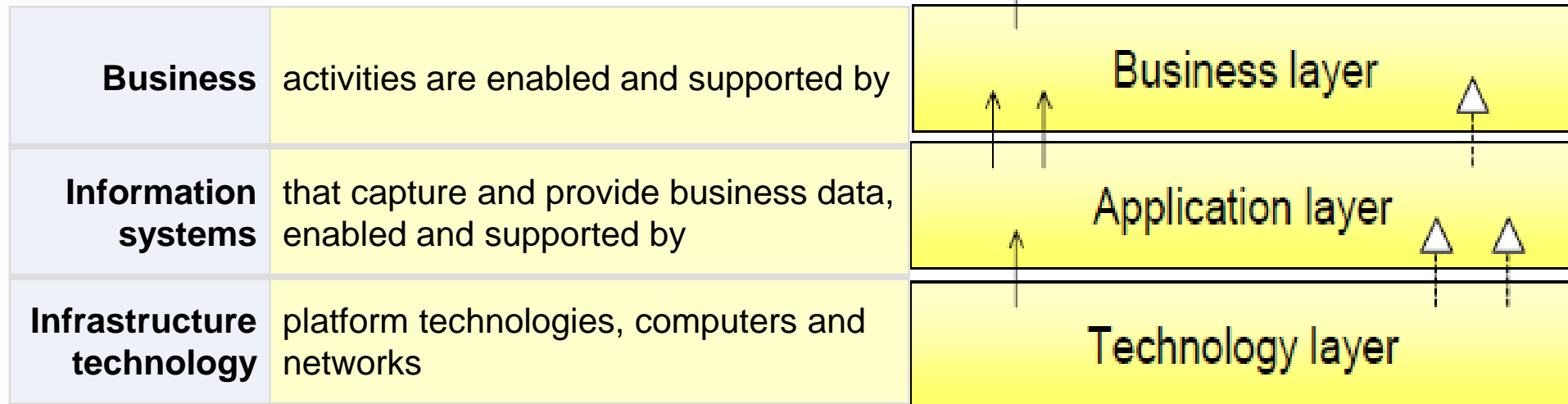
- ▶ Many and various approaches to modelling activity systems.
 - E.g. TOGAF, DoDAF, ArchiMate, UML, WSDL and System Dynamics.
- ▶ can be seen as applications of general system theory
- ▶ They model **human** and/or **computer activity systems**
 - Human activity system
 - Computer activity system
- ▶ as **discrete event-driven systems** in which
- ▶ **processes** are performed by **active components**.

EA is much about processes that create and use business data

- ▶ “the domain of **information-intensive organisations**...is the main focus of the language” (The ArchiMate modelling language standard v2.1)
- ▶ “EA is the determinant of survival in the **Information Age**.” (John Zachman)
- ▶ “Today’s CEOs know that the **effective management and exploitation of information** through IT is a key factor to business success.” (TOGAF 9.1)
- ▶ “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)

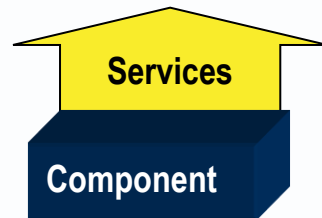


- ▶ “Architecture descriptions are **formal descriptions of a system.**”
 - of
- ▶ “**information-intensive organisations**”,
 - where

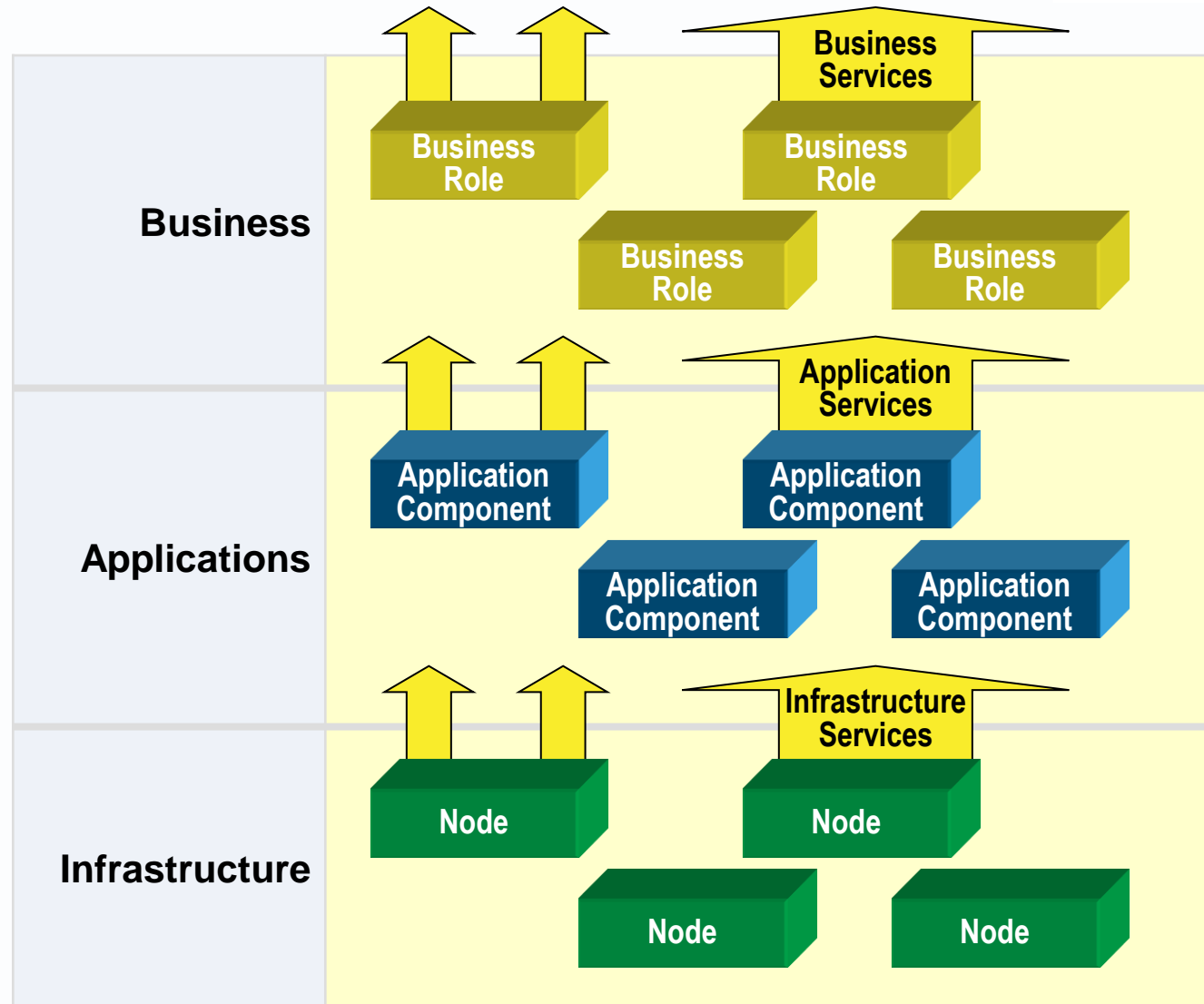


ArchiMate: architecture layers and system elements

- ▶ This naive 3-layer view is commonplace in architecture frameworks



- ▶ ArchiMate's system elements also include "passive structure" objects that are acted upon, including data objects.
- ▶ However, this slide show focuses on elements of active structure and behaviour.

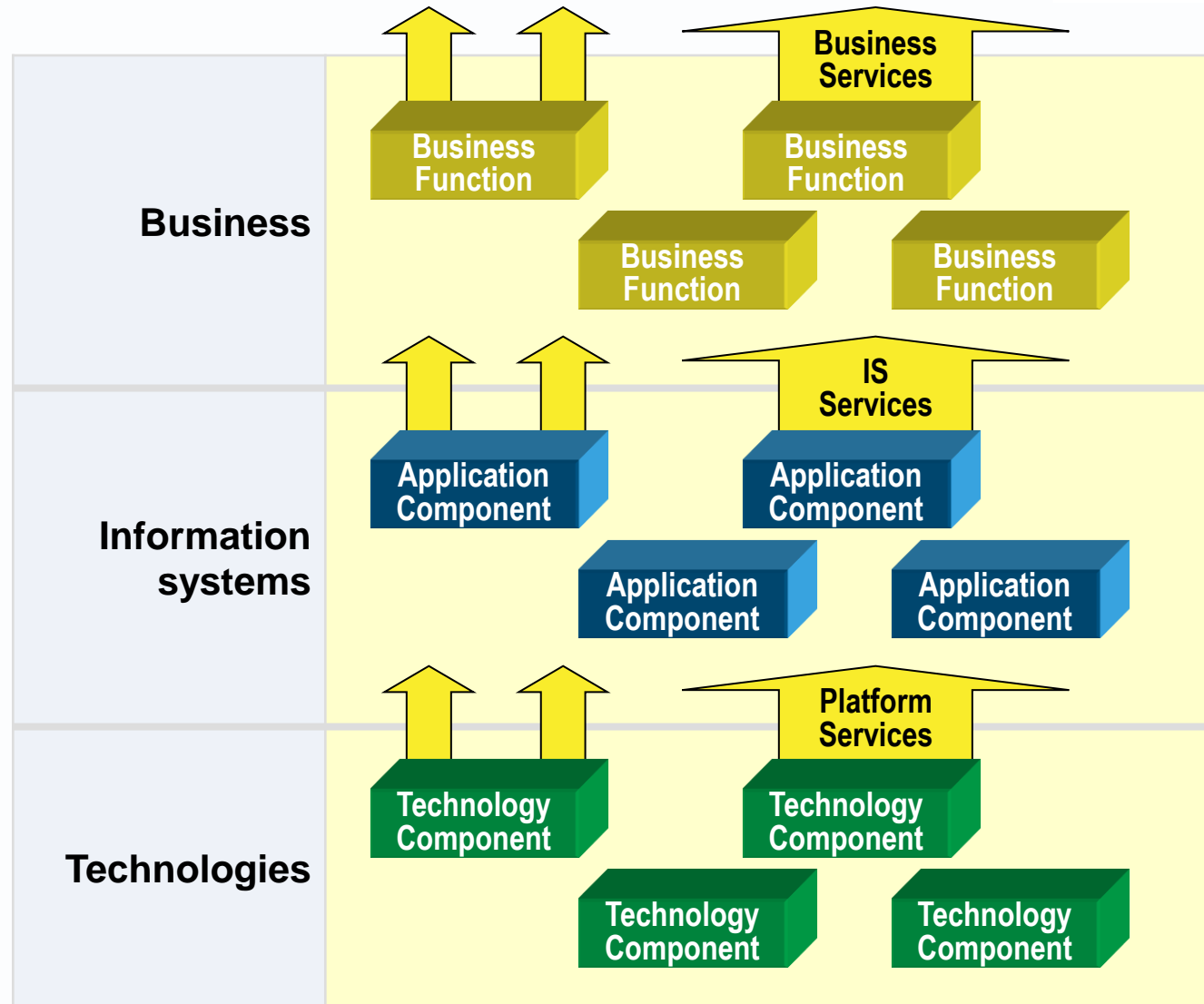


TOGAF: architecture domains and “Building Blocks”

- ▶ This naive 3-layer view is commonplace in architecture frameworks



- ▶ TOGAF has 4 domains; the information system layer is the sum of applications and data domains.
- ▶ However, this slide show focuses on the active structure and behaviour elements (rather than passive data structures).



Three architecture domains as layers

- ▶ In each layer, components deliver services to the layer above
 - (and to other components in the same layer)

	ArchiMate core concepts	TOGAF core building blocks
Business	<i>Business services</i>	<i>Business services</i>
	Business roles	Business functions & roles
Information systems	<i>Application services</i>	<i>Information system services</i>
	Application components	Application components
Infrastructure technology	<i>Infrastructure services</i>	<i>Platform services</i>
	Nodes (devices & system software)	Technology components

Service-orientation in ArchiMate and TOGAF

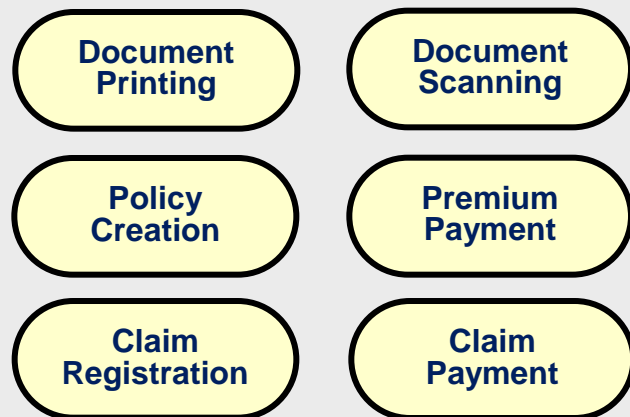
	Behavioural view	Structural view
External view	Event / Service	Interface
Internal view	Process	Component

A service is a discrete unit of behaviour

▶ ArchiMate definition

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

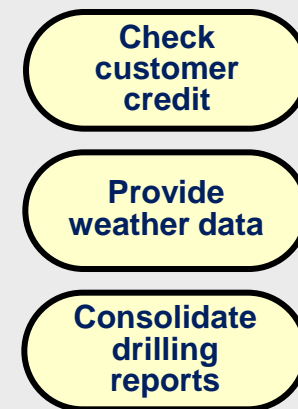
▶ ArchiMate examples



▶ 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



A service is a discrete unit of behaviour

- ▶ *is a discrete unit of behaviour that encapsulates processing and produces a result for its service requester/consumer.*
- ▶ is defined at whatever level of granularity an external entity (playing the role of service requester or consumer) recognises as a discrete operation.
- ▶ (perhaps an “epic” or “user story” in agile development)
- ▶ can be defined by a contract

“For the external users, only this **exposed functionality and value**, together with **non-functional aspects** such as the quality of service, costs, etc., are relevant. These [functional and non-functional aspects of a service] can be specified in a **contract**.”

ArchiMate 2.1

Check
customer
credit

Document
Printing

Premium
Payment

Consolidate
drilling
reports

Document
Scanning

Claim
Registration

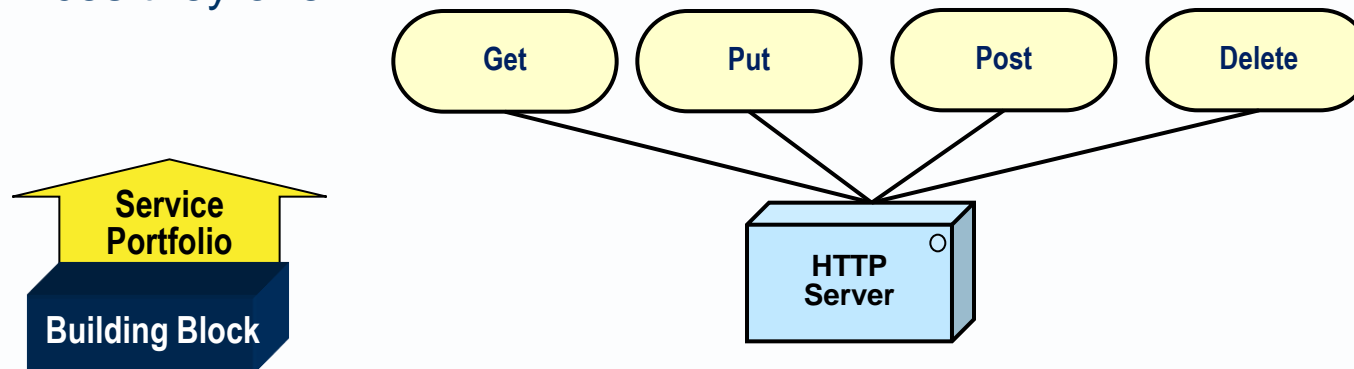
Provide
weather data

Policy
Creation

Claim
Payment

Services as the requirements for systems

- ▶ The Open Group (TOG) was created with the idea of standardising systems, to enable **portability** and **reuse**.
- ▶ Through the **open** development and publication of **vendor and technology-neutral specifications**. E.g. the Unix specification.
- ▶ They achieve this by specifying systems (and components thereof) by the services they offer.

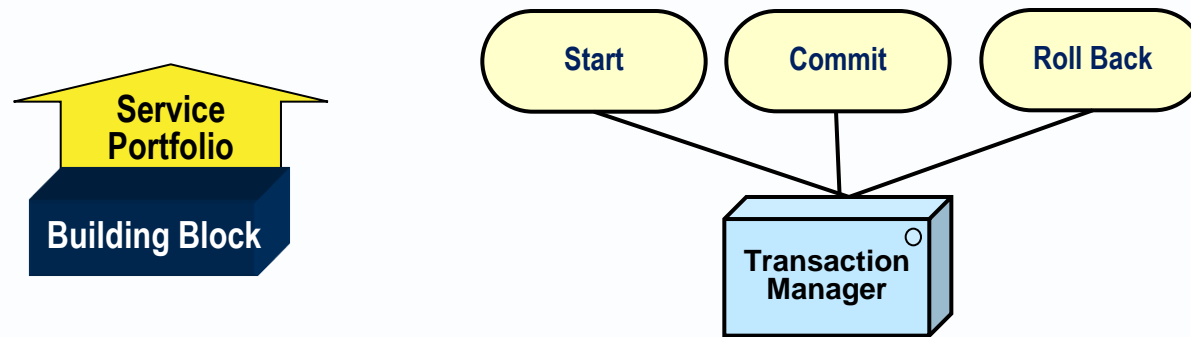


“For each building block, build up a service description portfolio as a set of non-conflicting services.” Phase D 12.4.1.6

TOGAF is based on defining services as the requirements for systems

- ▶ TOGAF 1 to 7 were centred on a Technical Reference Model (TRM)
- ▶ A TRM defines the enterprise's complete infrastructure/platform technology architecture by the services it delivers to business applications.

"For each building block, build up a service description portfolio as a set of non-conflicting services." Phase D 12.4.1.6



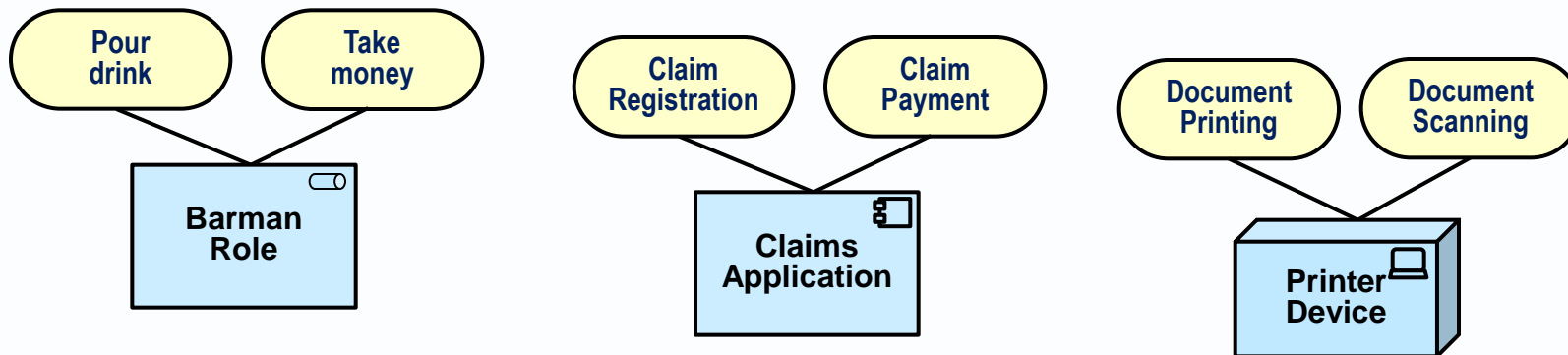
- ▶ TOGAF 8 and 9 extended the idea.
- ▶ The Architecture Requirements Specification contains the required services for building blocks each architecture domain/layer
- ▶ The required services are associated with "architecture building blocks".

Thus, TOGAF applies “component-based design” principles

- ▶ Defines an EA in terms of inter-related “**Building Blocks**”



- ▶ Defines a Building Block by the "**Service Portfolio**" (group of services) it provides.

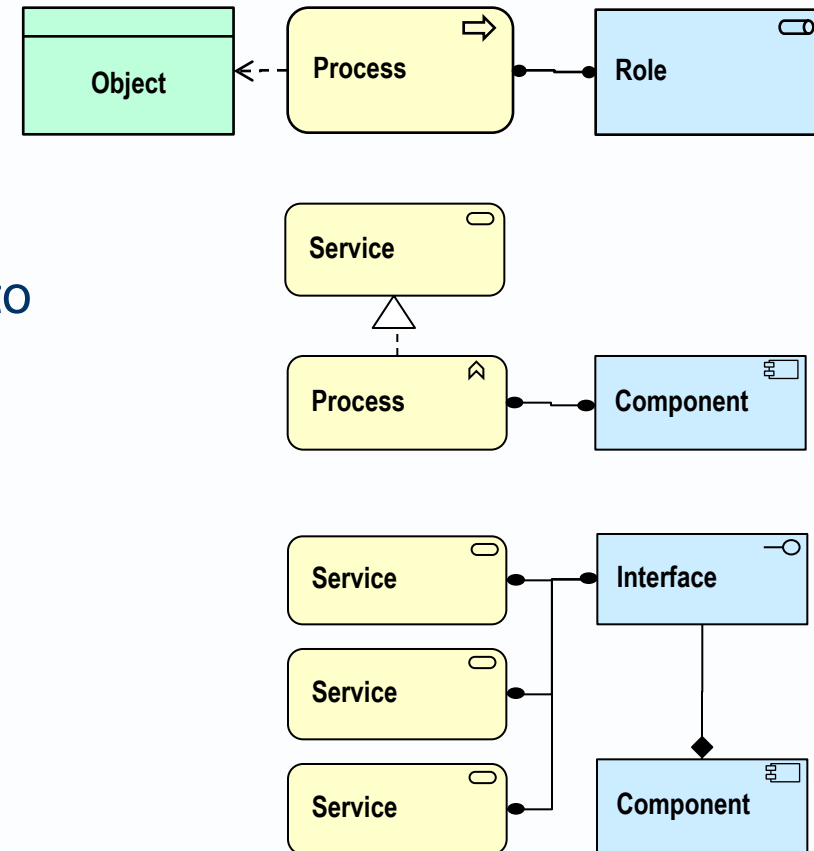


- ▶ Moreover, Building Blocks are defined at both **logical** and **physical** levels
- ▶ A "**logical component**"
 - an "**architecture building block**" (ABB)
 - defined by the "**service portfolio**" the component is required to provide.
 - E.g. the IETF standard FTP interface.
- ▶ A "**physical component**"
 - a "**solution building block**" (SBB)
 - you buy or build to realise the logical component's service portfolio.
 - E.g. the particular FTP server on your device(s).

Component-based design

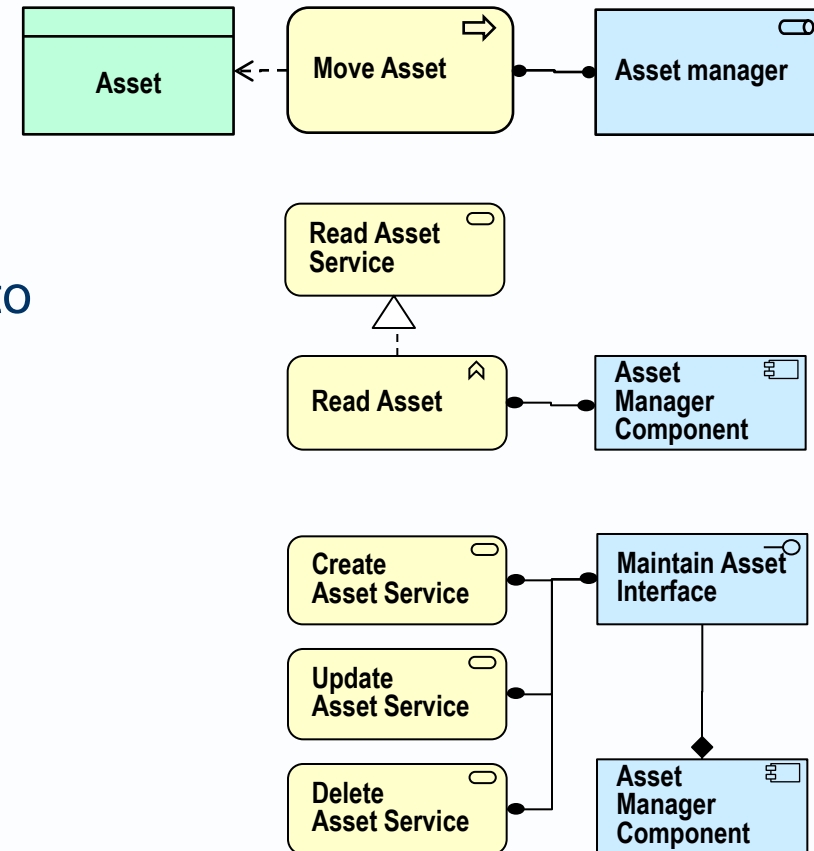
	Behavioural view	Structural view
External view	Event / Service	Interface
Internal view	Process	Component

- ▶ Much as general system theory suggests
- ▶ Components play Roles in Processes that create, move and change Objects.
- ▶ Components perform Functions/Processes to realise event-triggered Services.
- ▶ Components are encapsulated behind Interfaces through which they offer Services to other Components.



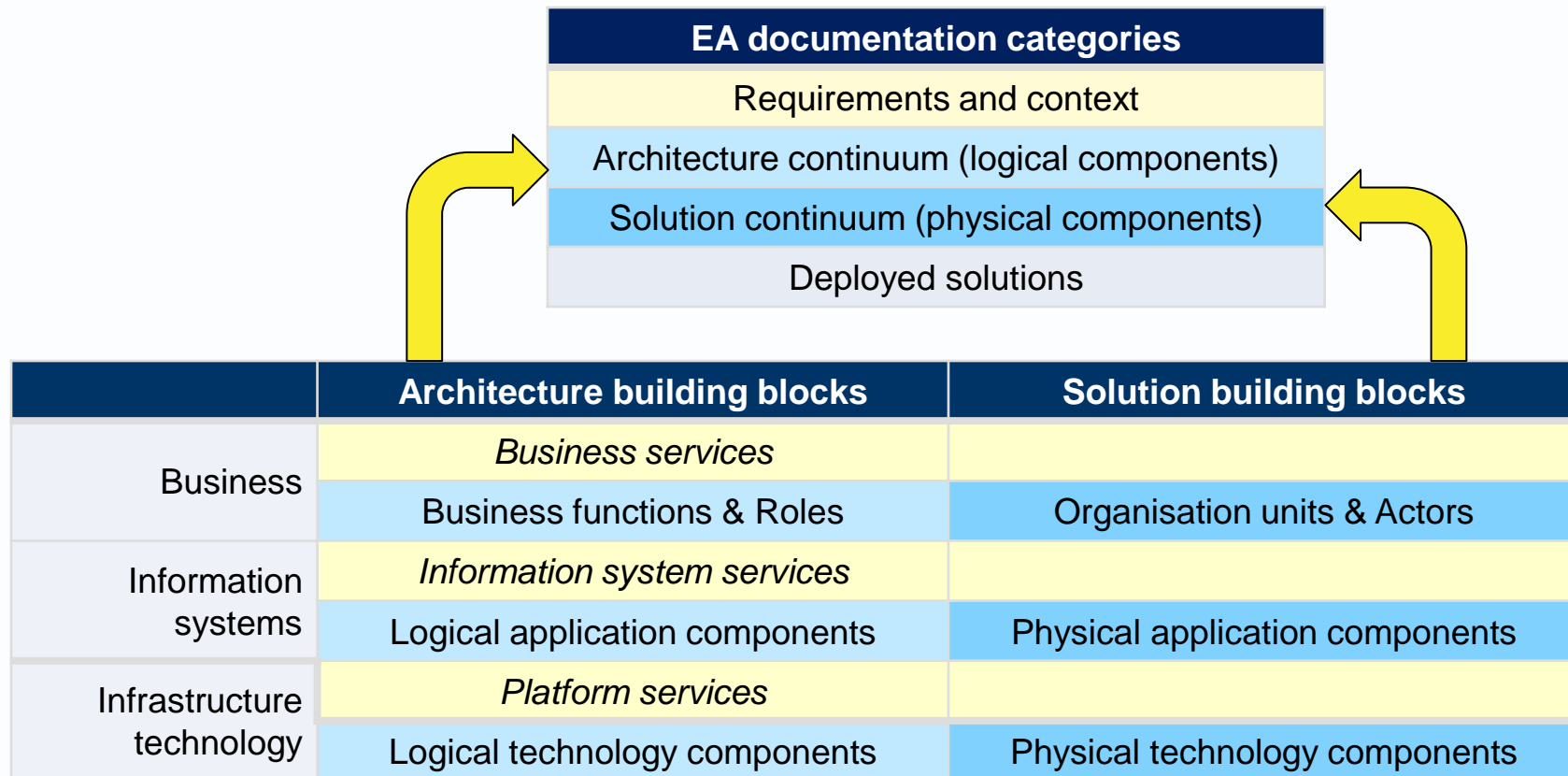
For example

- ▶ Much as general system theory suggests
- ▶ Components play Roles in Processes that create, move and change Objects.
- ▶ Components perform Functions/Processes to realise event-triggered Services.
- ▶ Components are encapsulated behind Interfaces through which they offer Services to other Components.



TOGAF's logical and physical component categories

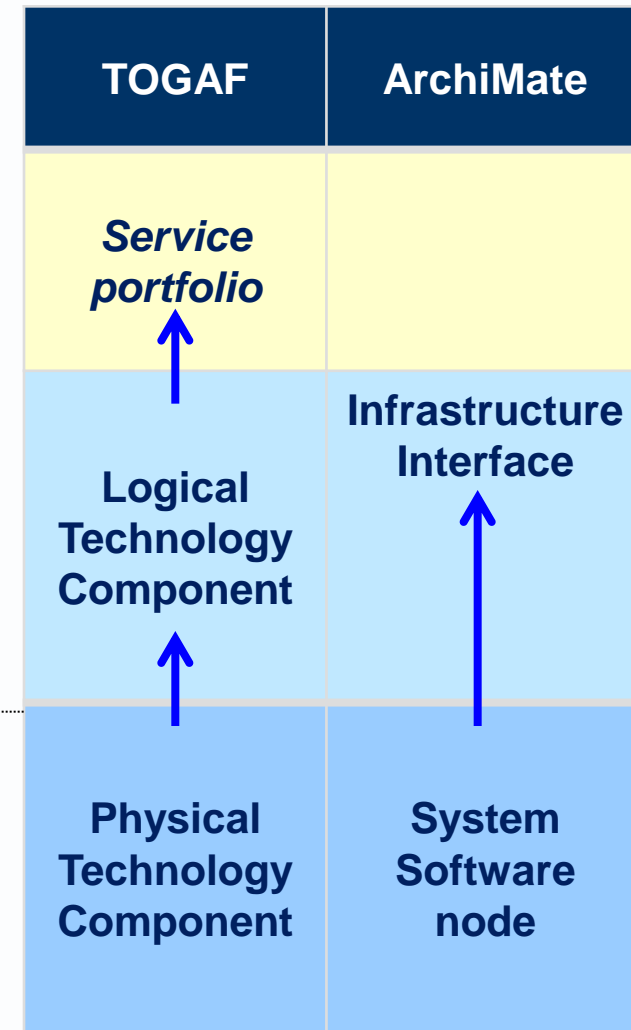
- ▶ The “Enterprise Continuum” classifies descriptive artefacts



- ▶ Each domain/layer is described as logical and physical components

Comparing TOGAF and ArchiMate terms

- ▶ A "**logical technology component**" is an "**architecture building block**" defined by the "**service portfolio**" the component is required to provide.
 - E.g. the IETF standard FTP interface.
- ▶ A "**physical technology component**" is a "**solution building block**" you buy or build to realise the logical component's service portfolio.
 - E.g. the particular FTP server on your device(s).



Logical component-based design

Phases B to D

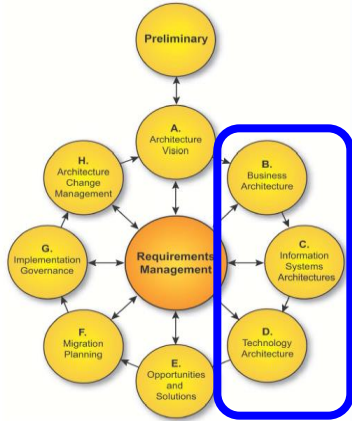
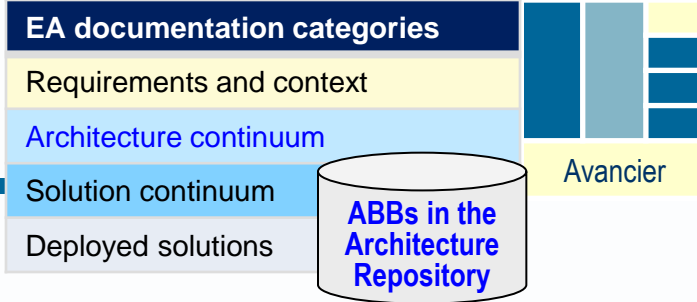


Figure 5-1 Architecture Development Cycle

TOGAF defines

- ▶ B: Logical Business Functions (e.g. "Sales") and Roles by the Business Services they offer.
- ▶ C: Logical App Components (e.g. "CRM") by the Information System Services (cf. use cases) they offer.
- ▶ D: Logical Technology Components (e.g. "DBMS") by the Platform Services (selected from the TRM) they offer.

	Architecture building blocks	Solution building blocks
Business	<i>Business services</i>	
	Business functions & Roles	
Information systems	<i>Information system services</i>	
	Logical application components	
Infrastructure technology	<i>Platform services</i>	
	Logical technology components	

Physical component-based design

Phases E to G

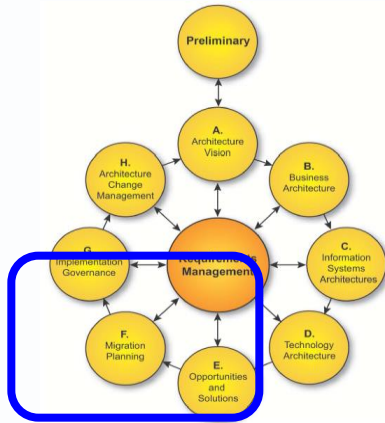
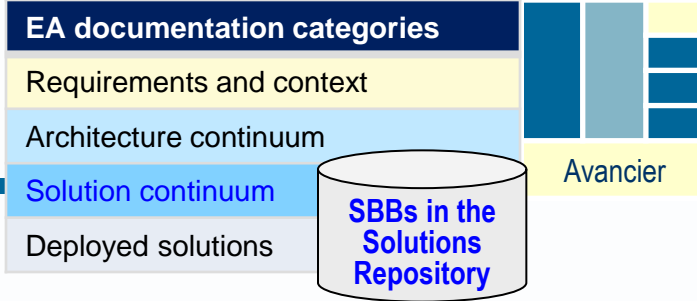


Figure 5-1 Architecture Development Cycle

- ▶ Buy or build one or more **Physical App Components** (say, Seibel) to **realise** each logical App Component.
- ▶ Select one or more **Physical Technology Components** (say, Oracle DBMS) to **realise** each logical Technology Component.

	Architecture building blocks	Solution building blocks
Business	<i>Business services</i>	
	Business functions & Roles	
Information systems	<i>Information system services</i>	
	Logical application components	Physical application components
Infrastructure technology	<i>Platform services</i>	
	Logical technology components	Physical technology components

The CBD and SOA principles underlying TOGAF

- ▶ In the Technology/Infrastructure Domain/Layer:
 - **Logical Technology Component = Platform Service portfolio**
 - **Physical Technology Component** (bought rather than built)

- ▶ In the IS/Application Domain/Layer:
 - **Logical Application Component = Information System Service portfolio**
 - **Physical Application Component** (bought or built)

- ▶ With luck, 1 Physical Component will realise 1 Logical Component.
- ▶ In general, the relation is many to many.
 - However, after you have chosen Physical Components in phase E, you could return to refactor the Logical Components so as to draw 1 to 1 correspondences, if you wanted to.

The same CBD and SOA principles in the Business domain/layer

- ▶ Applying the same reasoning to the business domain/layer:
 - **(Logical) Business Function = Business Service portfolio.**
 - **(Physical) Organisation Unit** (in-sourced or out-sourced).

- ▶ TOGAF refers to traditional structured analysis which “Identifies the key business functions..., and maps those functions onto the organizational units within the business” using a matrix

- ▶ The mapping can be done more precisely by assigning each service in each Function service portfolio to an Organisation unit (along with a manager and goals).

- ▶ The confusion of Actor with Role is widespread and profound, but one can make sense of it thus:
 - **(Logical) Role = (fine-grained) Business Service portfolio + skills required.**
 - **(Physical) Actor** (hired or managed).

So...

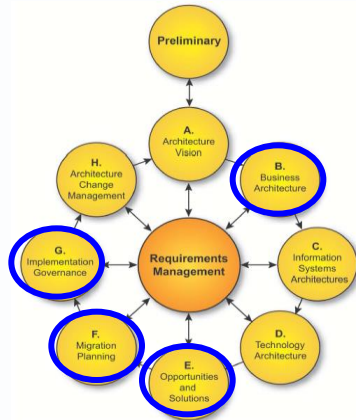
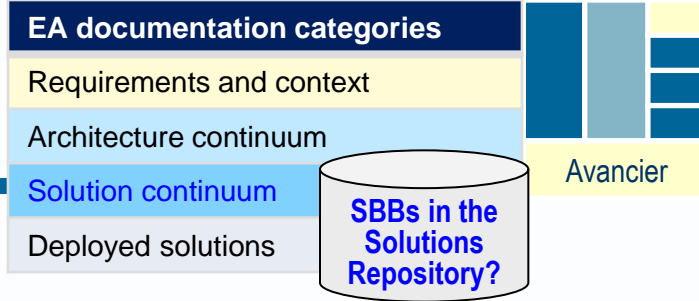
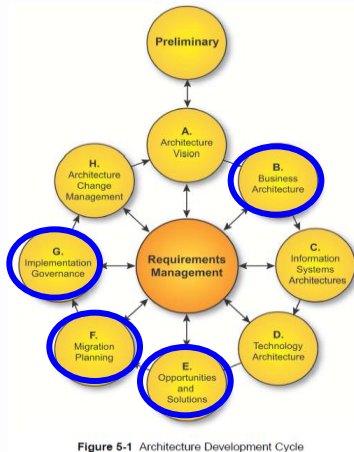


Figure 5-1 Architecture Development Cycle

- ▶ Does EA define physical **Organisation units** (e.g. "Sales"), in terms of goals, managers and resources,
 - to realise logical Business Functions?
- ▶ And then define physical **Actors**
 - to realise the logical Roles?



	Architecture building blocks	Solution building blocks
Business	<i>Business services</i>	
	Business functions & Roles	Organisation units & Actors
Information systems	<i>Information system services</i>	
	Logical application components	Physical application components
Infrastructure technology	<i>Platform services</i>	
	Logical technology components	Physical technology components



► Phase B:

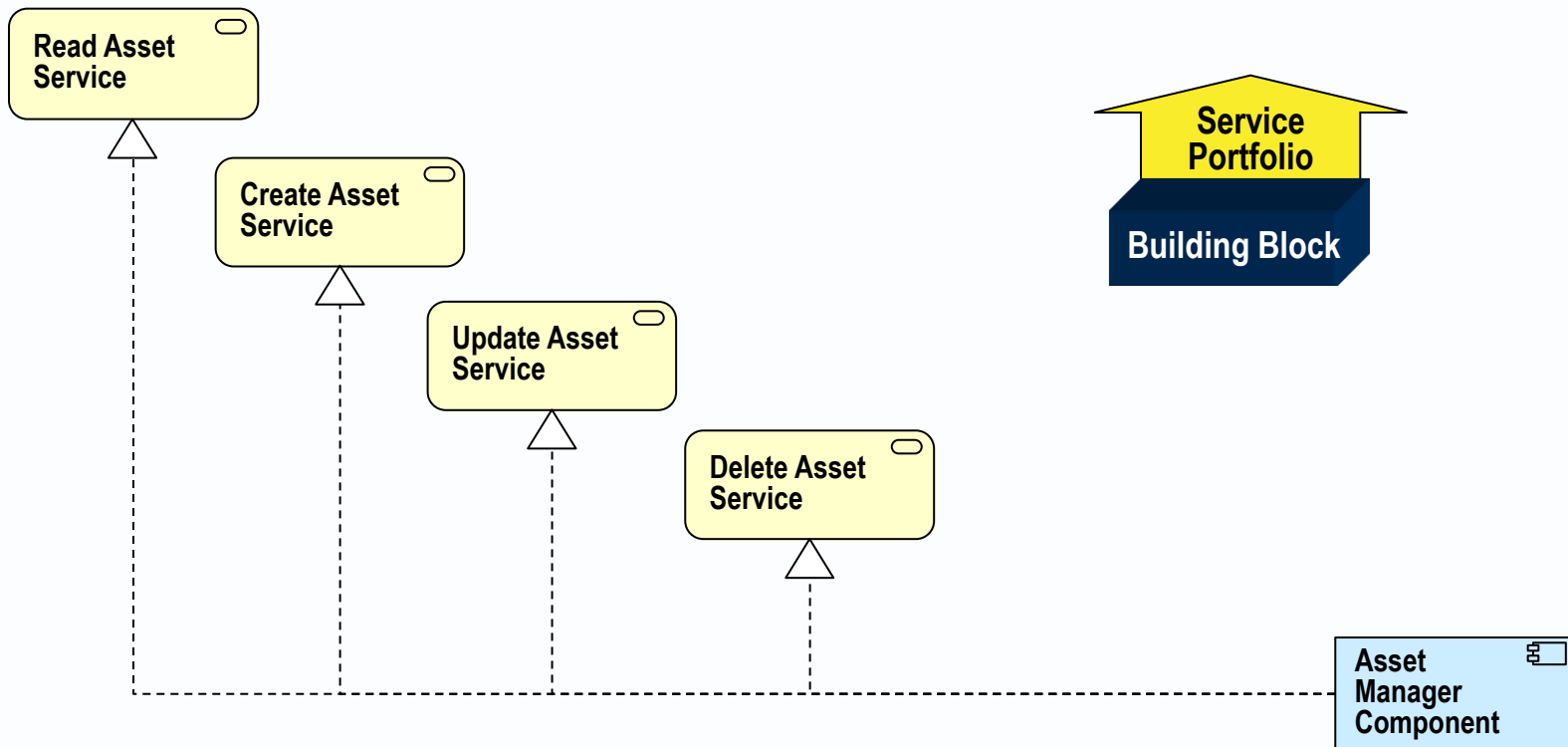
- TOGAF maps Business Functions to Organisation Units
- “Identifies the key business functions within the scope of the architecture, and maps those functions onto the organizational units within the business.” 8.4.1.1

► Phases E to G:

- Organization design, recruitment and other human matters (in the sphere of sociology, team managers, HR and Facilities Management) are usually better addressed by a “Business Change” team working in partnership with the EA team.

ArchiMate representation of services <realised by> a component

- ▶ A later slide show uses this example to illustrate “derived relations”



Structural and behavioural properties of a system

	Behavioural view	Structural view
External view	Event / Service	Interface
Internal view	Process	Component

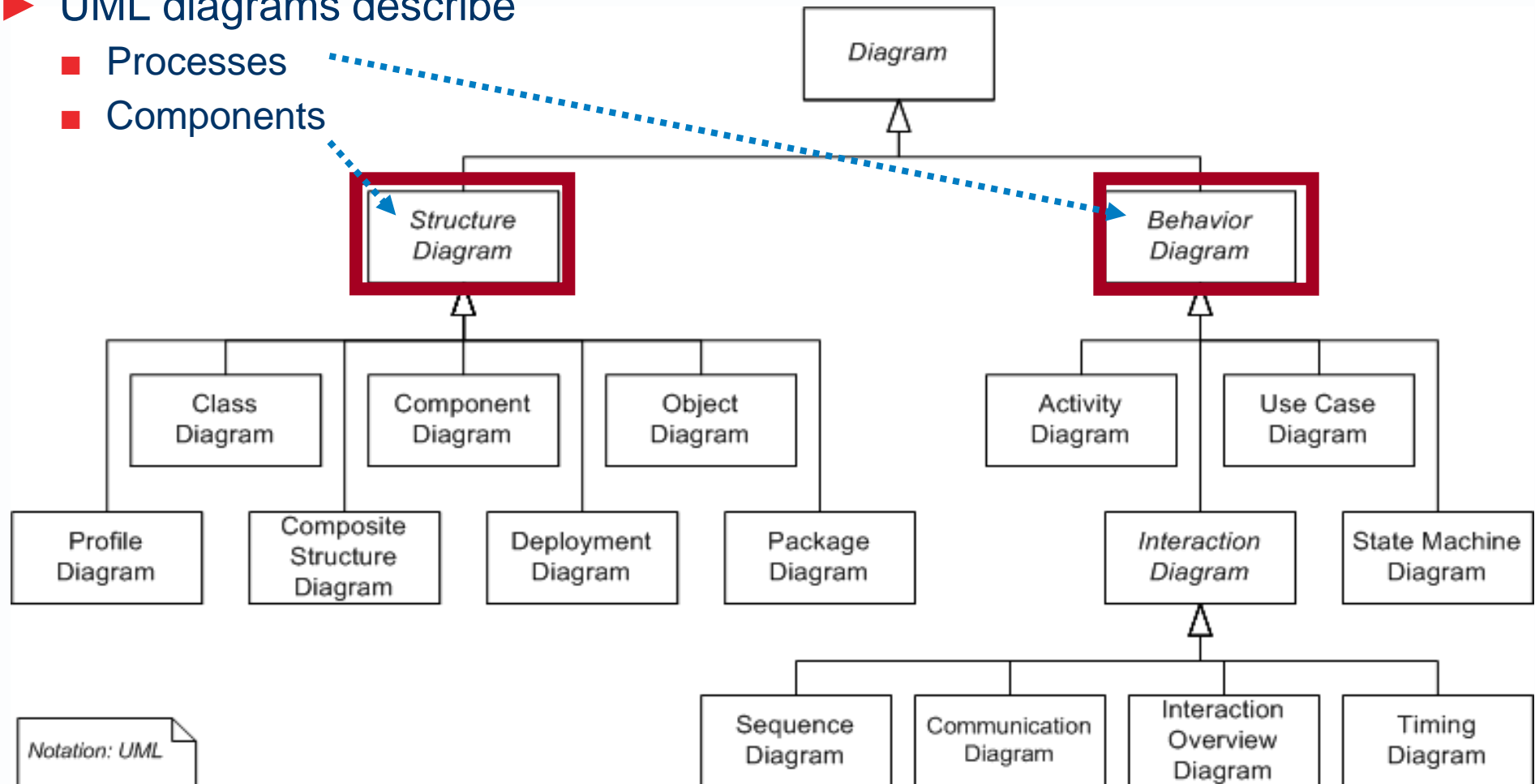
- ▶ “[Architecture descriptions are] organized in a way that supports reasoning about the **structural** and **behavioral** properties of the system and its evolution.” (ArchiMate® 2.1 Specification, The Open Group.)

- ▶ Traditionally
 - A behavioural view has start and end points, as in a flow chart, use case definition, state chart or interaction diagram in UML.
 - Behavioural elements describe what a system *does* - are typically called processes or activities – and have a time dimension.
 - Behavioural elements are usually repeated, and sometimes cyclical.
 - A structural view has no start and end points, as in a network diagram, a data model, or a class diagram in UML.
 - Structural elements describe what a system *is made of* – are typically called components, actors or roles – and are addressable.
 - Active components perform processes
 - Passive objects (on which behavior is performed) are not shown here.

UML also divides structural and behavioural properties

► UML diagrams describe

- Processes
- Components



An analogy – a restaurant

- ▶ It is unnecessary and impossible to prescribe human actions as far as computer actions must be prescribed.
- ▶ However, similar principles apply, and key system elements in a restaurant might be naively illustrated as below

ArchiMate/BCS grid	Behavioural view What a system does	Structural view What a system is made of
External view what external entities see	<i>Menu item services</i> <i>Bill requests</i>	<i>Menus</i>
Internal view the workers and workings	Ordering, Cooking, Serving, Paying	Waiters, Cooks, Ovens

- 14 system elements that ArchiMate provides symbols for

ArchiMate/BCS grid	Behavioural view What a system does	Structural view What a system is made of
	Menu items	Menus
External view what external entities see	Business Service Application Service Infrastructure Service.	Business Interface Application Interface Infrastructure Interface
Internal view the workings of the system	Business Process Application Function Infrastructure Function	Business Role, Actor Application Component System Software, Node
	Ordering, Cooking, Serving, Paying	Waiters, Cooks, Ovens

► 12 core building blocks in TOGAF's meta model

ArchiMate/BCS grid	Behavioural view What a system does	Structural view What a system is made of
	Services	Architecture Building Blocks
	<i>Business Service</i> <i>IS Service</i> <i>Platform Service</i>	<i>Function, Role</i> <i>Logical Application Component</i> <i>Logical Technology Component</i>
External view what external entities see		
Internal view the workings of the system	Business Process	Organisation, Actor Physical Application Component Physical Technology Component
		Solution Building Blocks

What is a Building Block (BB) in TOGAF?

- ▶ TOGAF is ambiguous
 1. BB = encapsulated component = replaceable subsystem = system inside another system, OR
 2. BB = *any* entity in the TOGAF meta model, including (1)
- ▶ If (1), which TOGAF entities are BBs? In what sense is a BB replaceable?
 - The previous slide lists 4 Architecture BBs and 4 Solution BBs.
 - Component A is replaceable by component B if they respond to the same events and deliver the same services that are listed and defined in *service contracts* – outlined on the next slide.
- ▶ If (2), in what sense is a BB replaceable?
 - Any architectural entity instance is replaceable if the replacement has the same attributes and relations to other entity instances in the architecture repository.
 - E.g. one location could replace another. E.g. One employee can replace another provided they have same attributes (knowledge and skills) defined as being required for a role.
- ▶ Since BBs are abstract definitions, real entities may qualify as replacements yet not be completely identical, and differences in *undefined qualities* damage the system. That is the risk of abstraction. And as every architect knows, it is often the non-functionals that kill you.

- ▶ A service **encapsulates behaviours** (aka processes, activities) that respond to an event or meet a service request.

“For the external users, only this exposed functionality and value, together with non-functional aspects such as the quality of service, costs, etc., are relevant. These [function and non-functional aspects of a service] can be specified in a contract.”

ArchiMate 2.1

- ▶ Service contract template
 - **Signature**: service name, inputs and outputs.
 - **Functional rules**: preconditions and post conditions
 - **Non-functional characteristics**: including performance and commercial conditions.
- ▶ May correspond to an “epic” or “story” in agile software development, a behaviour that delivers some value

Check
customer
credit

Document
Printing

Premium
Payment

Consolidate
drilling
reports

Document
Scanning

Claim
Registration

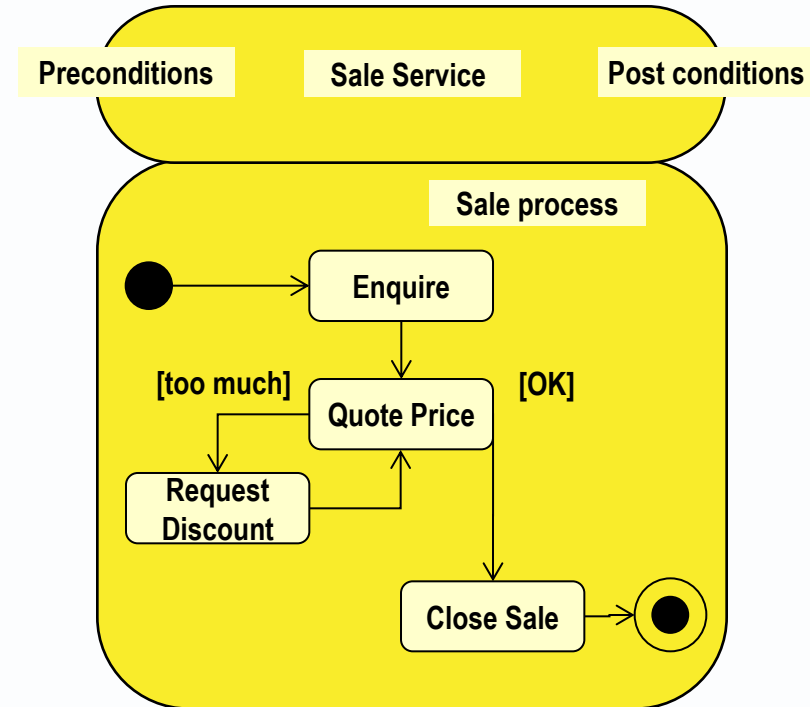
Provide
weather data

Policy
Creation

Claim
Payment

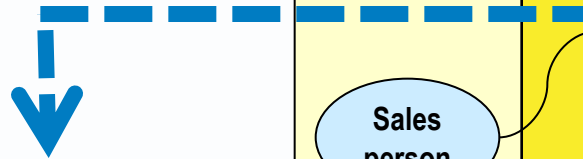
Services are <realised by> Processes

- ▶ There are
 - human processes,
 - computer processes, and
 - hybrid human-computer processes that may be called workflows or use cases.
- ▶ A process or activity is
 - specified by defining it at the level of detailed “actions” that actors/components can understand and perform.
 - deployed by publishing it where actors/components can read and perform it
 - performed when actors/components read and follow the published process.

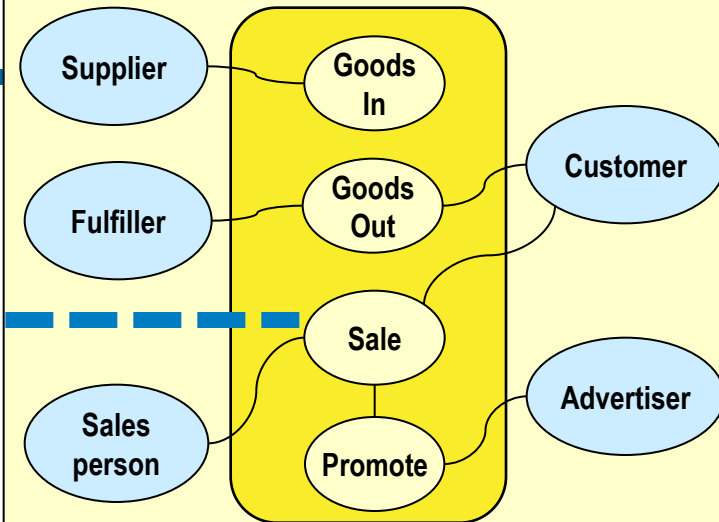


Process documentation

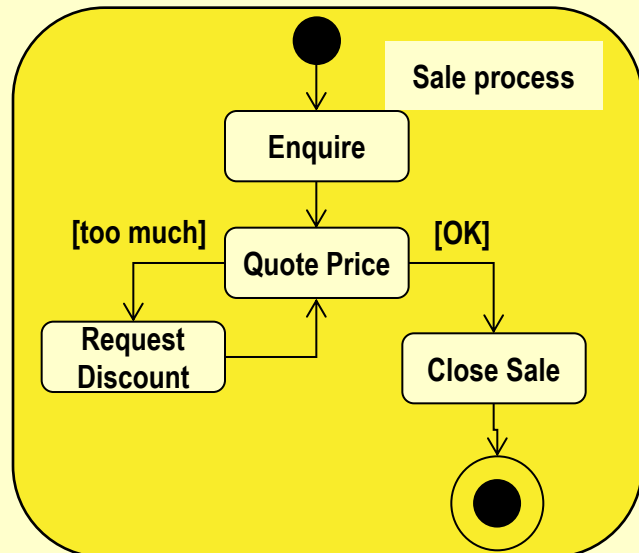
- ▶ A flow chart, use case definition, state chart or interaction diagram
- ▶ Has a time dimension
- ▶ Runs from start to end



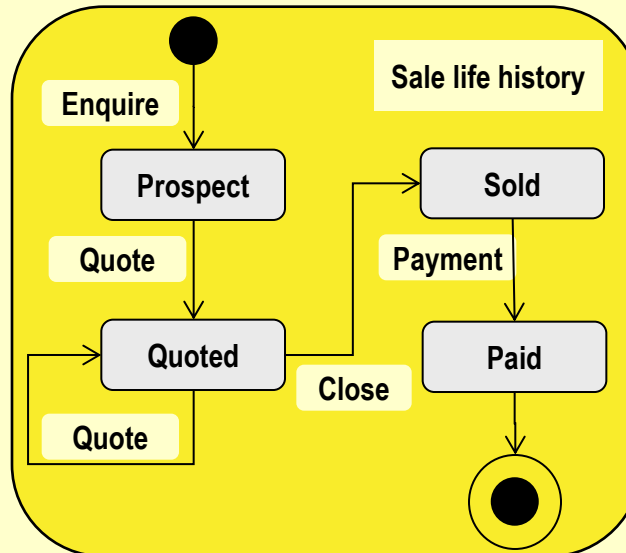
Scope – Process Map
UML Use Case Diagram



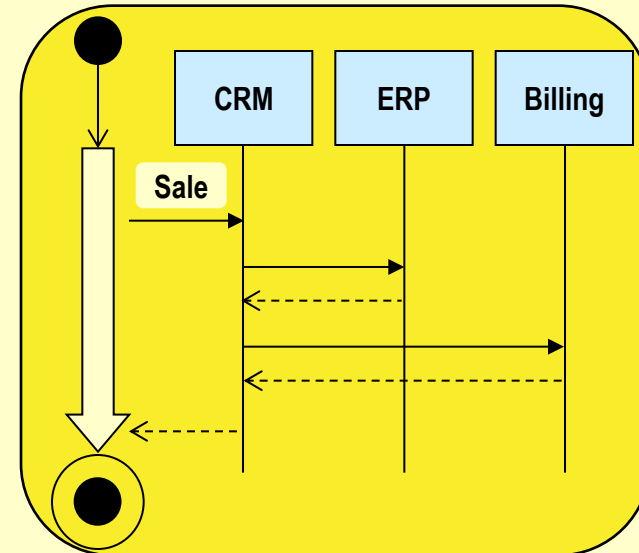
Process Flow Chart
UML Activity Diagram



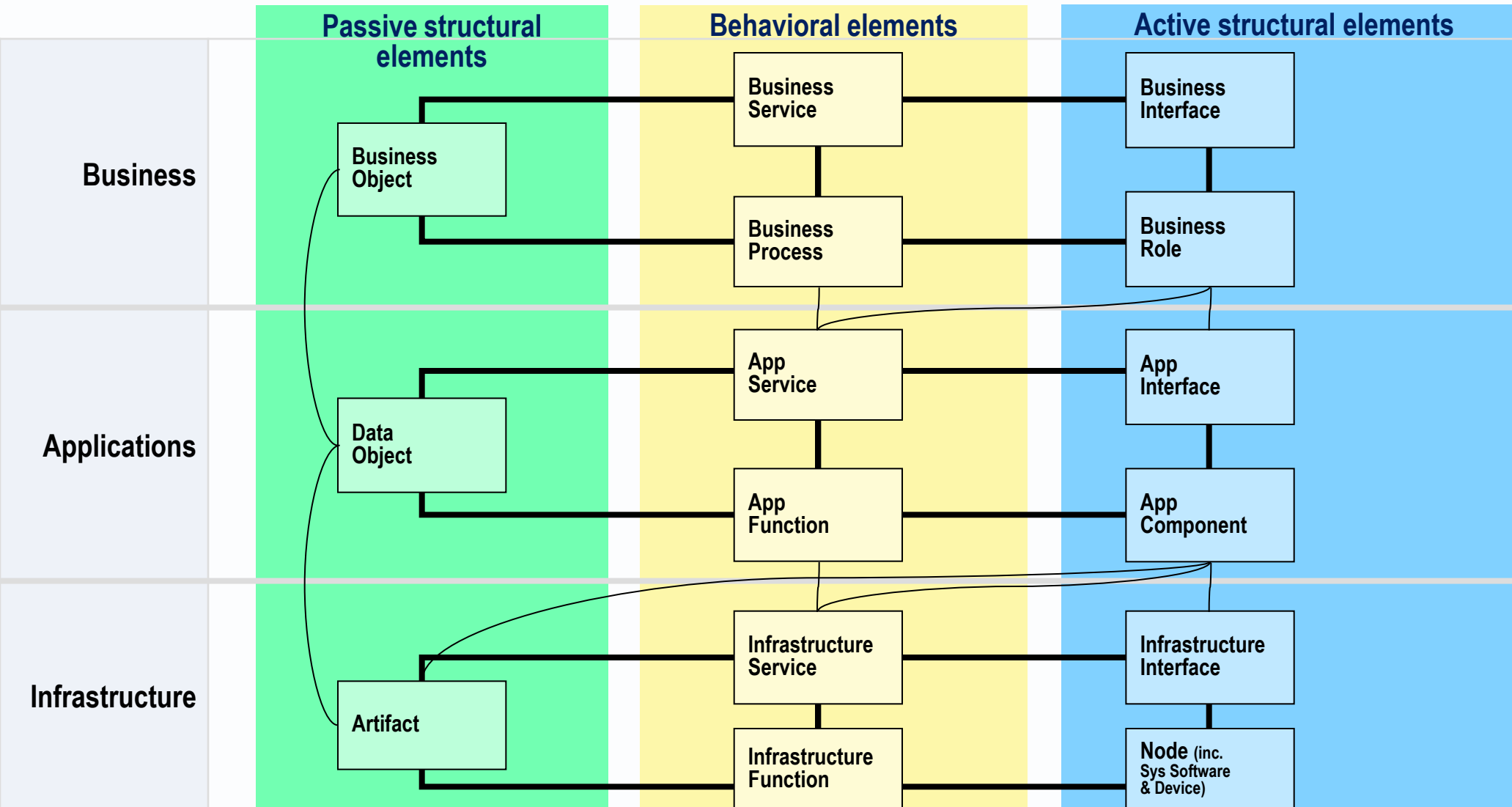
Entity Life Cycle
UML State Chart



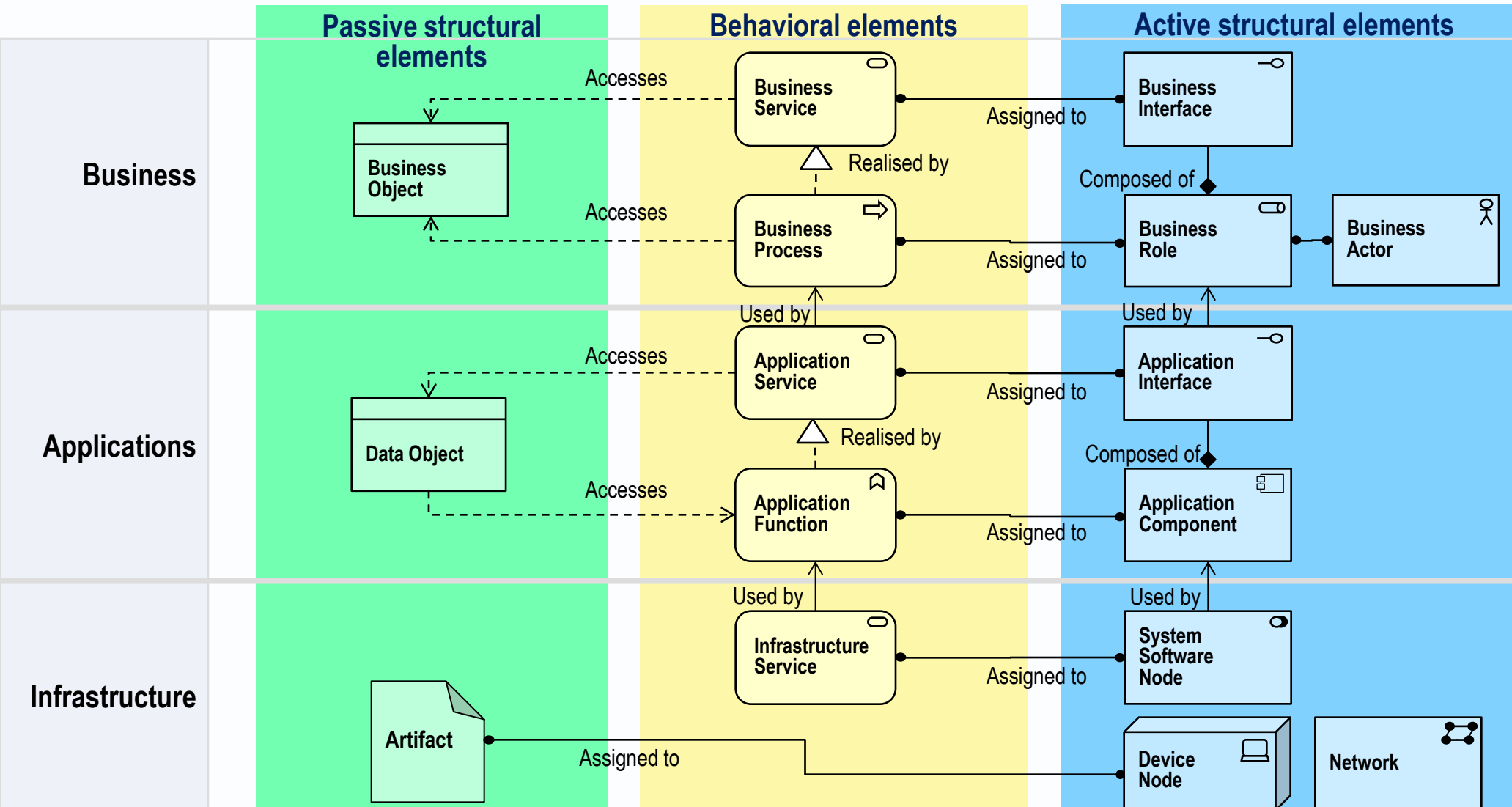
Applications Architecture Behaviour
UML Interaction/Sequence Diagram









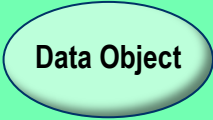








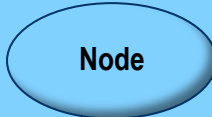
The ArchiMate generic meta model mapped to architecture layers



Examples of ArchiMate symbol uses

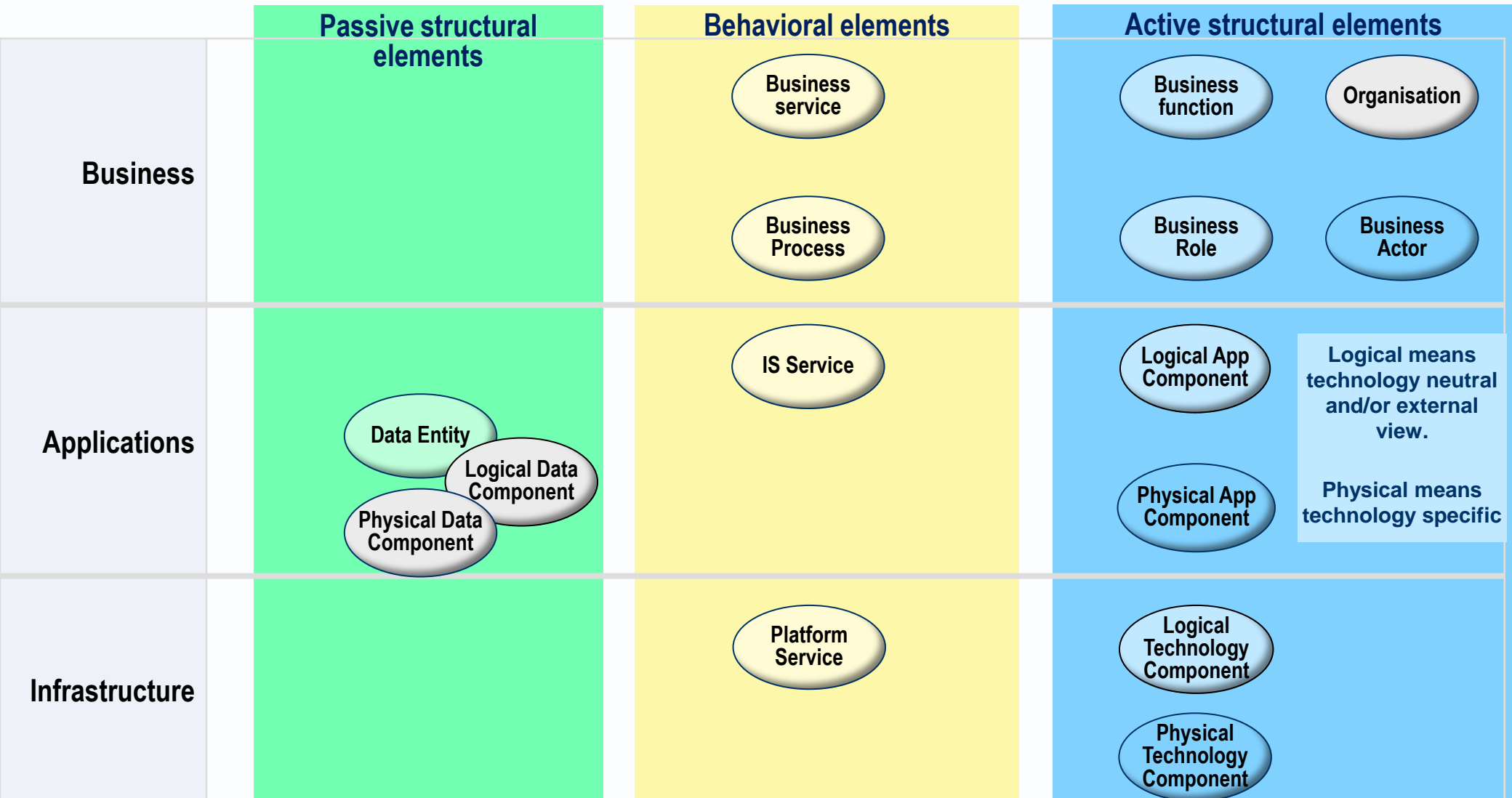


16 core concepts in ArchiMate

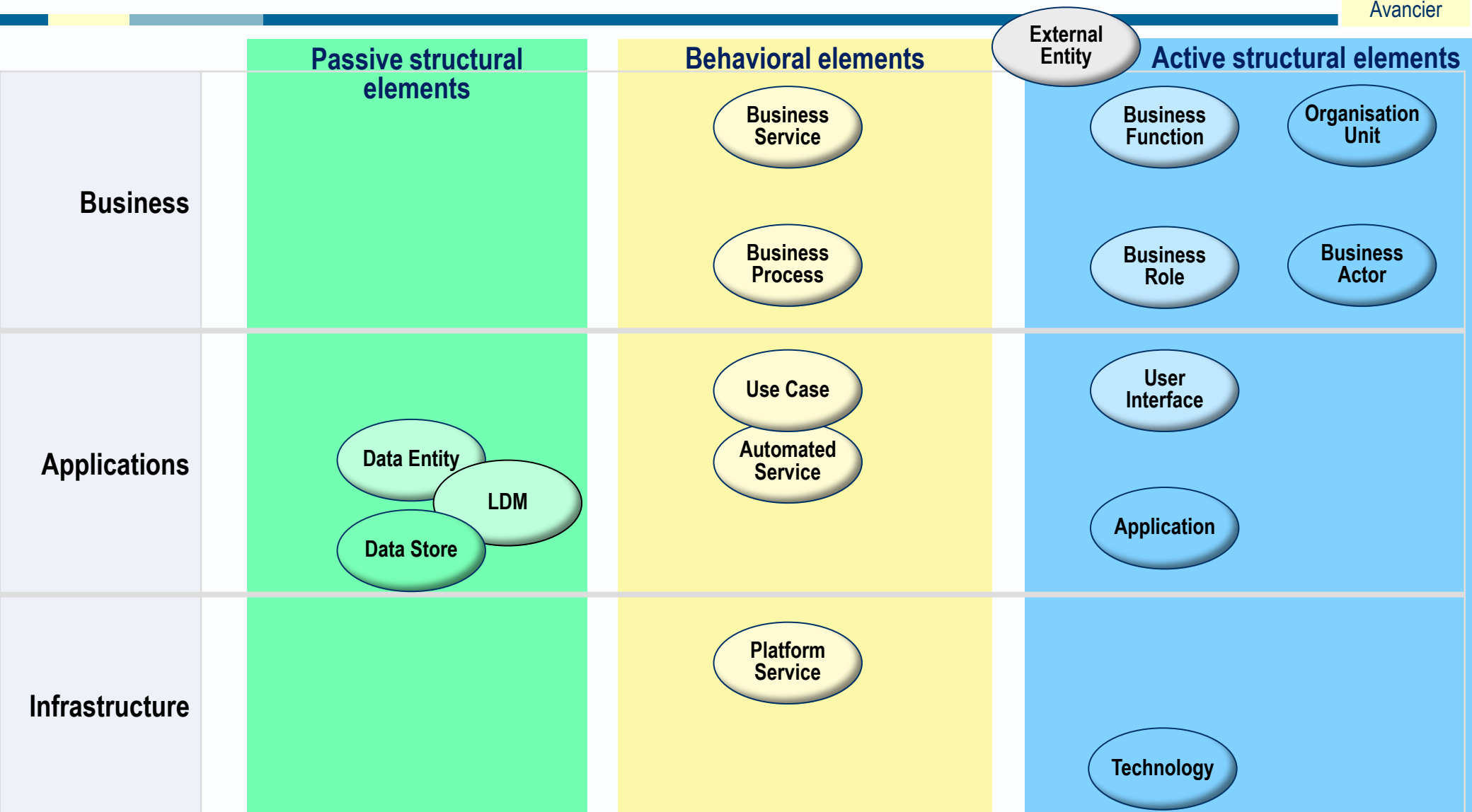
	Passive structural elements	Behavioral elements	Active structural elements
Business		 	  
Applications		 	 
Infrastructure		 	 

15 core Building Blocks in TOGAF

“Building Block” is ambiguous in TOGAF
Sometimes an encapsulated component
Below, any architectural entity



16 core concepts in the BCS reference model for E&SA



	Behavioural view	Structural view
External view	Event / Service	Interface
Internal view	Process	Component

Other approaches

DoDAF, WSDL, UML, System Dynamics and social system thinkers and others name the same or similar concepts differently

Mapping *DoDAF* concepts to the generic meta model

- ▶ DoDAF models systems in which Activities are performed by Performers

DoDAF	Behavioural view	Structural view
External view	? (service contract?)	Service (interface definition)
Internal view	Activity (process)	Performer (active component)

Mapping *WSDL* concepts to the generic meta model

- ▶ An *Interface* Definition Language (like WSDL) models the *external* view of a system, as its users see it

WSDL	Behavioural view	Structural view
External view	WS Operation (service contract)	Web Service (interface definition)
Internal view	hidden	Addresses of components that perform the operations named in the interface

Mapping *UML* concepts to the generic meta model

1. “all behavior in a modeled system is ultimately caused by actions executed by so-called “active” objects”
2. “behavioral semantics only deal with *event-driven*, or discrete, behaviors”

UML	Behavioural view	Structural view
External view	Events, Operations An operation is a behavioural feature of an interface that specifies the name, type, parameters, and constraints for invoking a behaviour (below)	Interface Specifies features to be realised by objects that provide a façade conforming to the interface (not an instantiable at run time)
Internal view	Activities, Interactions, State machines An activity is step-by-step algorithm, performed by active objects, decomposable into elementary performable actions.	Classes, Components, Nodes, Objects An active object performs behavior and does not cease until either the complete behavior is executed or the object is terminated

Mapping *System Dynamics* to the generic meta model

- ▶ A System Dynamics Model can be run to simulate system behaviour
 - (It is an abstraction of a Discrete Event Dynamics Model)

System Dynamics	Behavioural view	Structural view
External view	Time unit	Report (of stock levels over time)
Internal view	Flow (events per time unit and their effects on stock levels)	Stock (level of a stateful component)

Mapping *Social System* concepts to the generic meta model

- ▶ Boulding (1956) considered a social organisation as an ordered system, as a collection of **individuals** playing **roles** in repeatable **processes** to reach some goals. He said the concept of **information flow** must be added to the material and energy flows found in purely physical systems

Social systems	Behavioural view	Structural view
External view	Messages sent/received A social system depends on its individuals sending and receiving messages.	Roles? An individual interacts with its environment;
Internal view	Processes, Acts An individual actor acts in a society in a way that depends on the messages it receives from other actors, and its current state, a collection of remembered “mental images”.	Roles? Individuals The components of social systems are human roles rather than human beings. The individuals who play those roles exhibit behaviour.

A generalisation of the four system elements

- ▶ Structure and Behaviour views
- ▶ Internal and External views

ArchiMate/BCS grid	Behavioural view What a system does	Structural view What a system is made of
External view what external entities see	Event / Service encapsulates the processing of a discrete event or service request.	Interface a service portfolio accessible by clients
Internal view the workings of the system	Process one or more activities that respond to an event or meet a service request.	Component a subsystem that performs process steps.

How the four system elements are documented in practice

- ▶ The four elements are documented in various combinations.
- ▶ E.g. a use case wraps a process flow inside a service contract

ArchiMate/BCS grid	Behavioural view What a system does	Structural view What a system is made of
External view what external entities see	<div>Event / Service Found in interfaces The trigger / result of a process</div>	Interface Makes services accessible Provided and required by components
Internal view the workings of the system	<div><i>Use case definition</i> Wrapped in a service Performed by components Process</div>	Provides and requires interfaces Performs processes Component

Some conclusions

- ▶ All approaches can be seen as applying general system theory
- ▶ They model human and/or computer activity systems as discrete event-driven systems in which processes are performed by active components.
- ▶ The 2-by-2 grid classifies the terms used in different approaches
- ▶ The terms used in different approaches may be generalised as.

	Behavioural view	Structural view
External view	Event / Service	Interface
Internal view	Process	Component

Terminology issues

This slide show introduces some of the research done for the BCS into “architecture” sources. There are ambiguities in and discrepancies between the sources. And the wider industry terminology is a mess.

What does “data architecture” mean?

- ▶ An IS architect thinks of
 - data at rest (stores) and in motion (flows),
 - the data structures they contain (logical and canonical data models),
 - data qualities, data owners,

- ▶ An IT architect thinks of
 - disc arrays, NAS, SAN,
 - active/passive, synch/asynchronous replication,
 - availability and disaster recovery.

- ▶ Different architects use words with different meanings.

Is a “Function” structural or behavioural?

- ▶ A Business Function is clearly a unit of Business Capability in TOGAF.
 - a logical business component in a functional decomposition structure.
- ▶ But in other sources, a Function is a behaviour/process.

What we call	Others may call	Description	Think
Function	Capability	a component or chunk of business capability	TOGAF-style structured analysis.
Process	Function	a logical step-by-step sequence of actions	UML activity diagram, flow chart
Aim	Function	an outcome or purpose of a system	
Application	Process	an instance of a program running on a computer	in the “Task Manager” on your lap top

Other confusions include those between

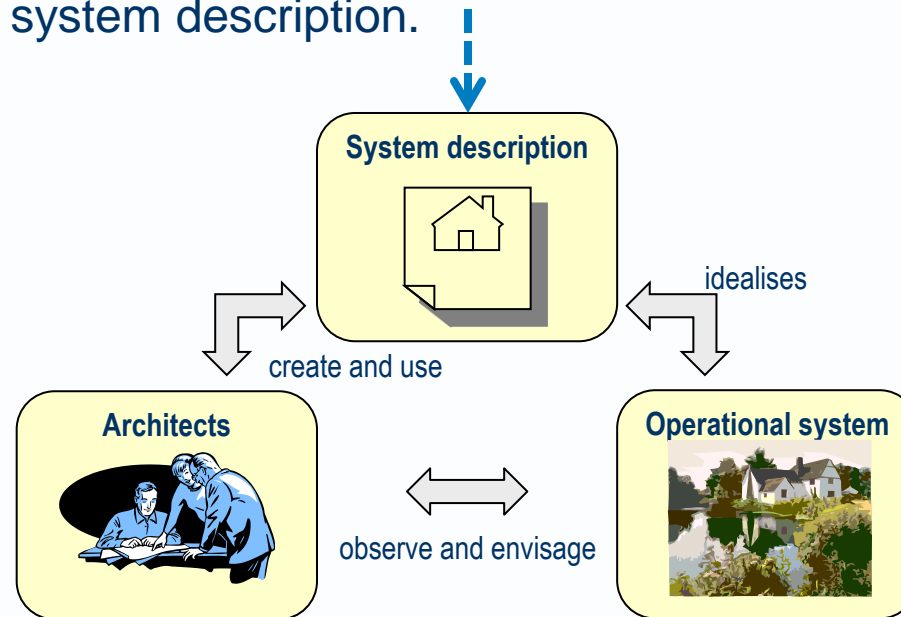
- ▶ Addressable interfaces, and invokable services.
- ▶ Interface definitions, and façade components.
 - Note that UML interfaces are not instantiated
- ▶ Types, instances and things.
 - Instantiation by design and creation of an instance (e.g. an OOP object) is one thing.
 - Instantiation by prompting a thing (e.g. a human actor) to instantiate a type (a role) is another.
 - We never model human actors, we model only the roles they play in our systems.
 - Human actors can do more for (and against) a system's purposes than any role defined in a system model for them.

“Service” and “Interface” are especially ambiguous terms

What we call a	Others may call a	Description	Think	For example in
Service	Operation	A contract for invoking processes to deliver a desired result	Menu item	ArchiMate and TOGAF examples above
Interface	Service	A service portfolio accessible by clients	Menu	A WSDL file
Façade	Interface	A structure that offers services of the first kind, a broker to service providers.	Waiter	
Data flow	Interface	A data structure passed from sender to receiver.	Message	TOGAF's application comm. diagram
(logical) Protocol	Interface	Rules used to convey data flows, over channels.	HTTP	File Transfer Protocol (FTP)
(physical) Channel	Interface	Medium used convey data flows, using protocols.	Wire	A cable connecting two computers
Component	Service	Any component that sits behind a published interface.		A "Microservice"
Available app	Service	The availability of an application to users, provided by IT operations.		
SLA contract	Service, Interface	Whatever services are listed in a contractual document.		

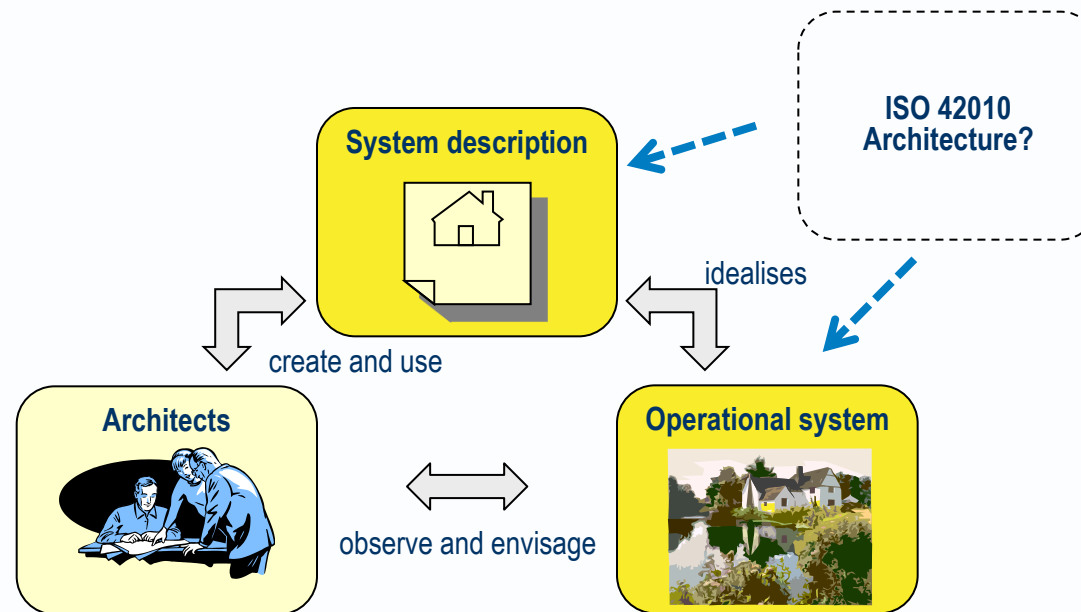
How do we govern or maintain "the architecture" of a system?

- ▶ The architecture of a system is an abstraction
- ▶ It exists only in a system description (be it mental or documented)
- ▶ Moreover, *infinite* different architectures can be abstracted from any operational system (or "real machine" as a system theorist might call it).
- ▶ The only definitive and tractable architecture is one found in an agreed architecture-level system description.



Contrary to what you might infer from ISO 42010

- ▶ To maintain "the architecture" of a system means ensuring alignment of
 - the properties of the operational system, which are testable
 - the properties recorded in an agreed architectural system description.
- ▶ There is no 3rd thing identifiable or recognisable as "the architecture".



1. “all behavior in a modeled system is ultimately caused by actions executed by so-called “active” objects”
2. “behavioral semantics only deal with *event-driven*, or discrete, behaviors” UML 2.1

- ▶ These fundamental premises are obscure in some sources.
- ▶ All architecture frameworks and modelling languages are about the design and description of human and computer activity systems.
- ▶ Systems are modelled as being
 - driven by discrete events
 - composed of structural components that
 - cooperate in behavioural processes
 - maintain information about the state of those processes.

An ontology of system elements

System element: a unit of an operational system that can be described.	Structural element: a unit of what a system is made of.	Active structural element: a structural element that can be prompted to do work, perform actions.	Component: a subsystem that can perform one or more process steps. It can be related to other components by requesting or delivering services. It can be replaced by any other component with the same interface(s). Aka building block.
		Passive structural element: a structural element that is acted upon.	Interface: a service portfolio accessible by clients.
	Behavioural element: a unit of what a system does.	Service: the external view of the processing that responds to a discrete event or service request.	Object: an item or structure that is used, moved or made. E.g. a data item, data structure, or other kind of structural component.
		Process: one or more activities, triggered by an event or service request, that ends by producing a desired effect. It is performed by one or more components. It is described or defined as a sequence of actions or instructions under a logical control flow.	Location: a place where components and interfaces are found and work is done.

- ▶ Our training and methods are useful with all architecture frameworks that share similar domains and entities
- ▶ <http://avancier.website>

