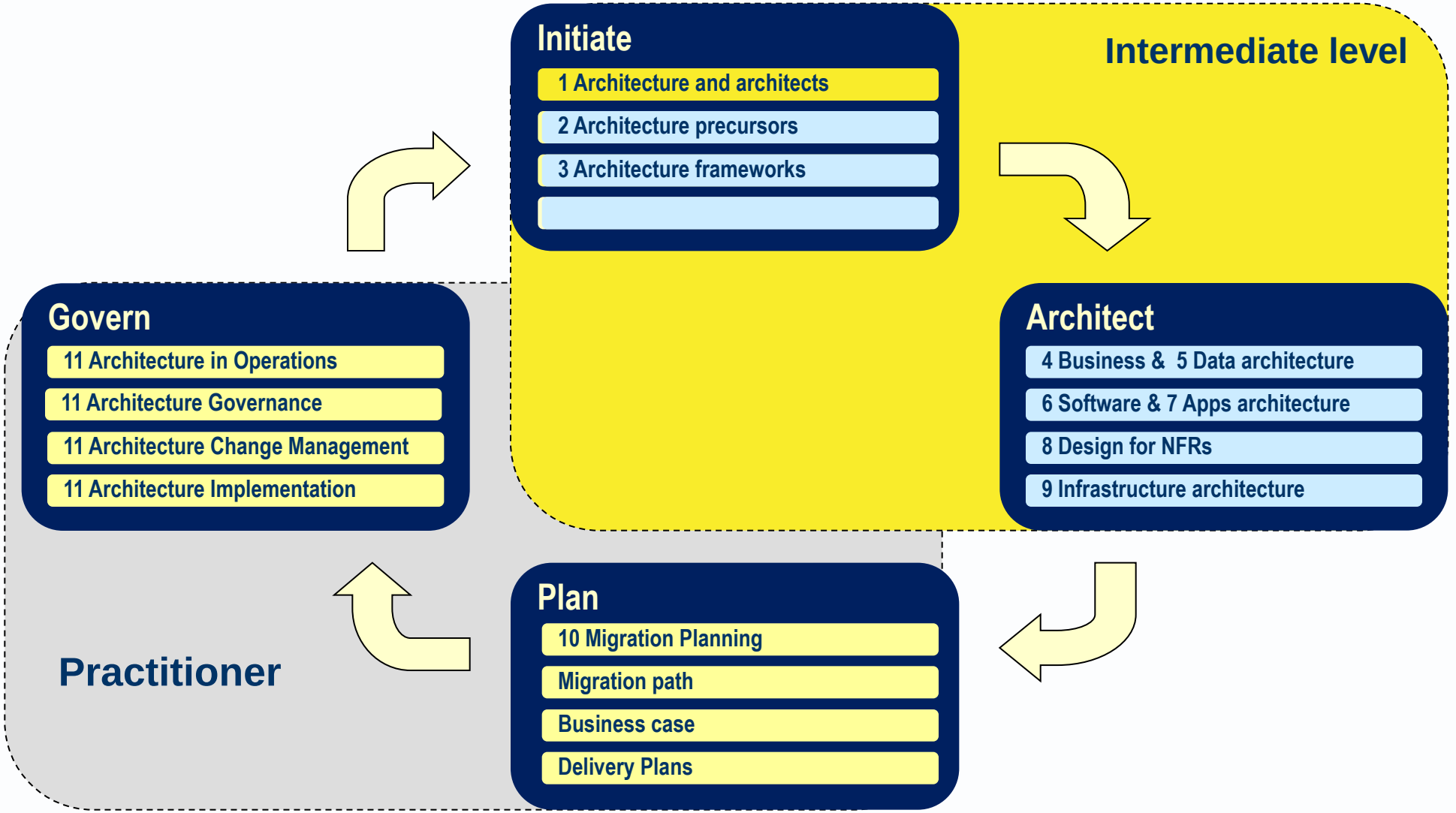
Avancier Reference Model

Architecture and Architects (ESA 1)

1. Architecture and architects

Avancier

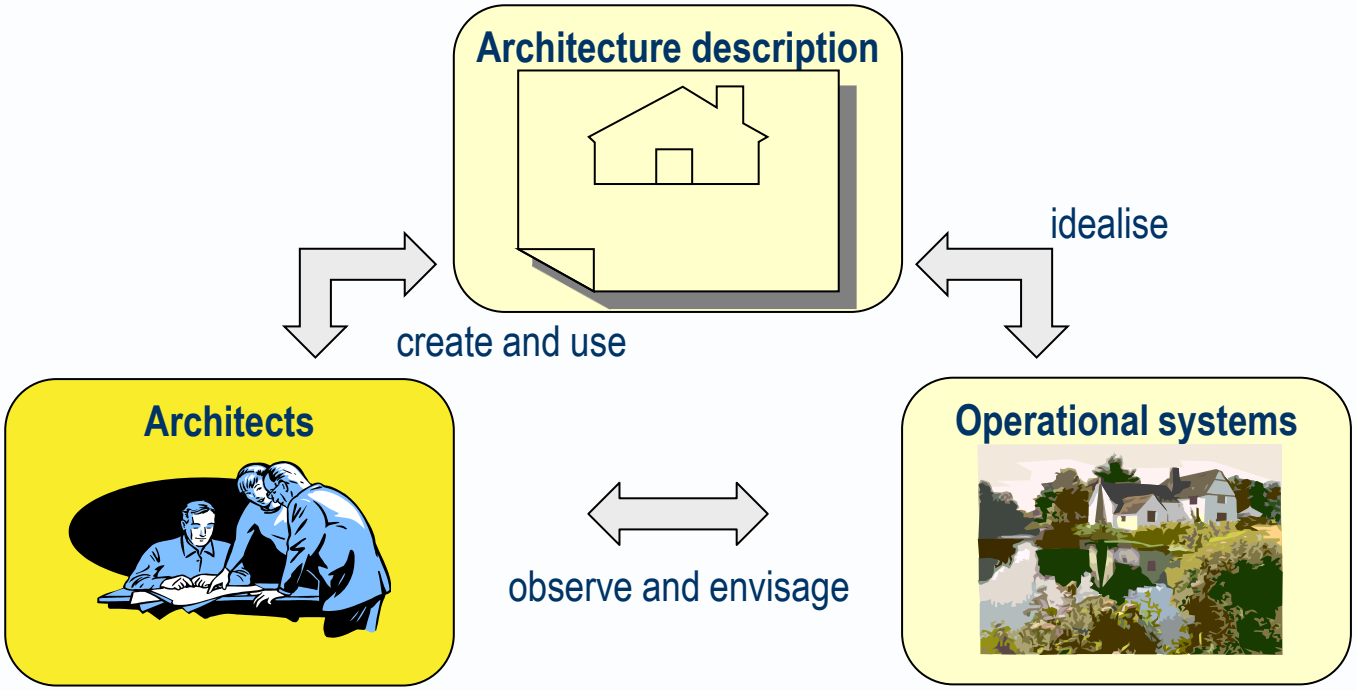


1.1: Foundation (rarely examined)

- Architect
- Architecture description (aka architecture)
- Architecting
- Architecture framework

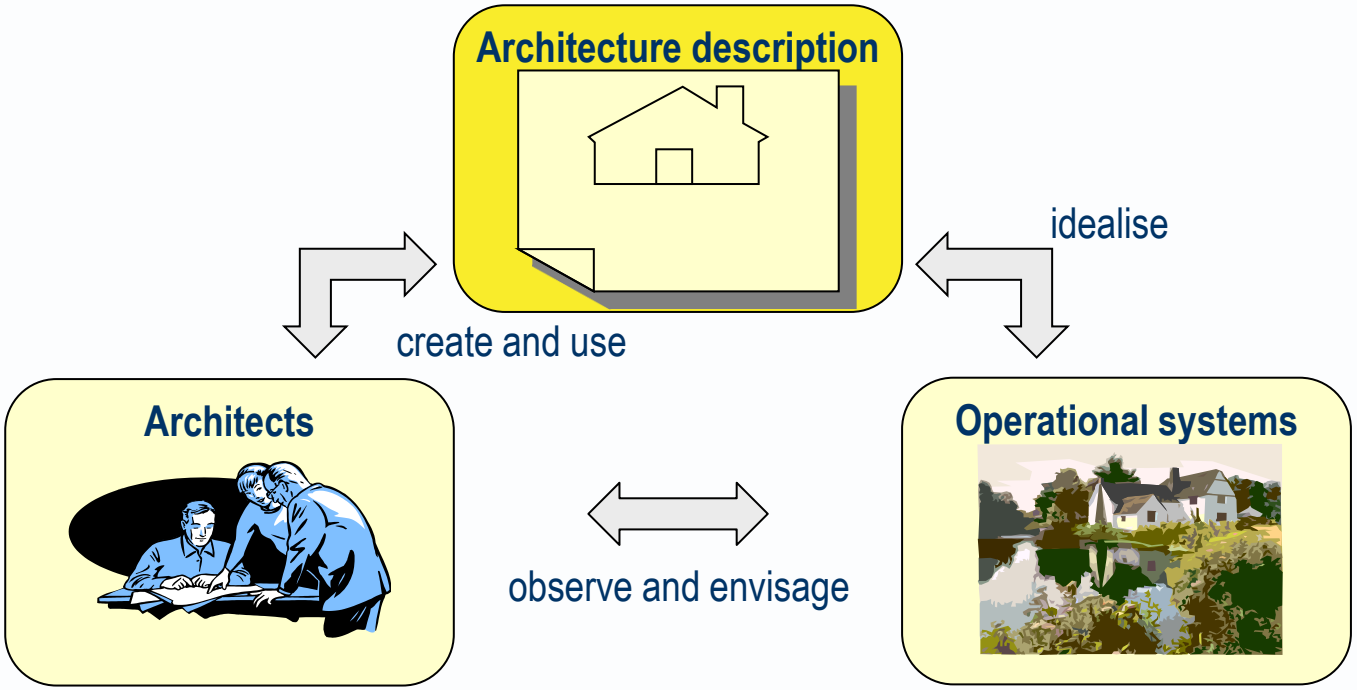
Architect:

- [a role] to design, plan and oversee the building of systems.
- It involves responding to requests for work, gathering contextual information, producing architecture descriptions and superintending lower-level design or construction.



Architecture description (aka architecture):

- [a work product] that describes the structures and behaviours of an operational system or capability.
- It may map system elements to motivations, constraints or work packages needed implement the system.



Architecting:

- [a work process] that creates an architecture description and a plan for building a system.
- It involves
  - analysis of the context,
  - engaging with stakeholders,
  - making decisions,
  - analysing and choosing between options,
  - defining system elements,
  - recording them in an architecture description and
  - ensuring agreement of what is described.
- It often involves
  - planning the move from the baseline state to a target state, and
  - governing that change to ensure conformance of what is implemented to architecture descriptions.

Architecture as higher level design

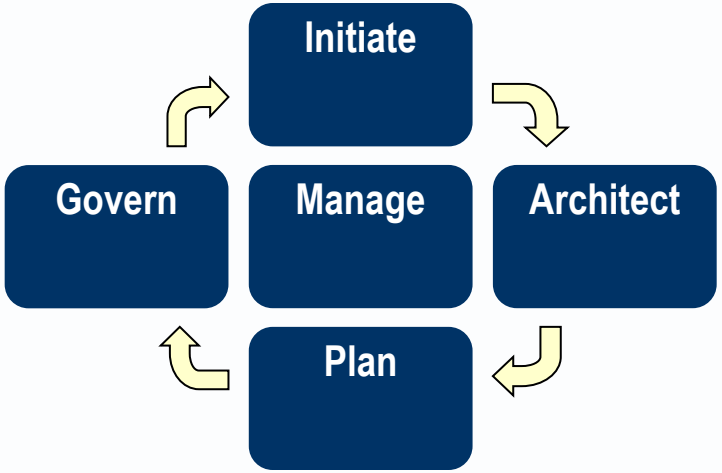
Avancier

Higher level design		Lower level design
Strategies and road maps	**Longer time** / **Shorter time**	Shorter term sprints and deadlines
Broader goals, longer processes and coarser-grained subsystems	**Composition** / **Decomposition**	Narrower requirements, shorter processes and finer-grained components
Standards, principles, patterns and reference models	**Generalisation** / **Specialisation**	Application of standards, principles, patterns and reference models
Business needs and idealised system descriptions	**Idealisation** / **Realisation**	Physical technology solutions
Encapsulation by services in interfaces	**External** / **Internal**	Realisation by internal roles and processes
Required services and processes	**Behaviour** / **Structure**	Designed roles and interfaces

Architecture framework:

- A comprehensive architecture framework contains advice on:
- Processes - for architecting
- Products - for architecture description
- People - architect roles.

- Processes - for architecting

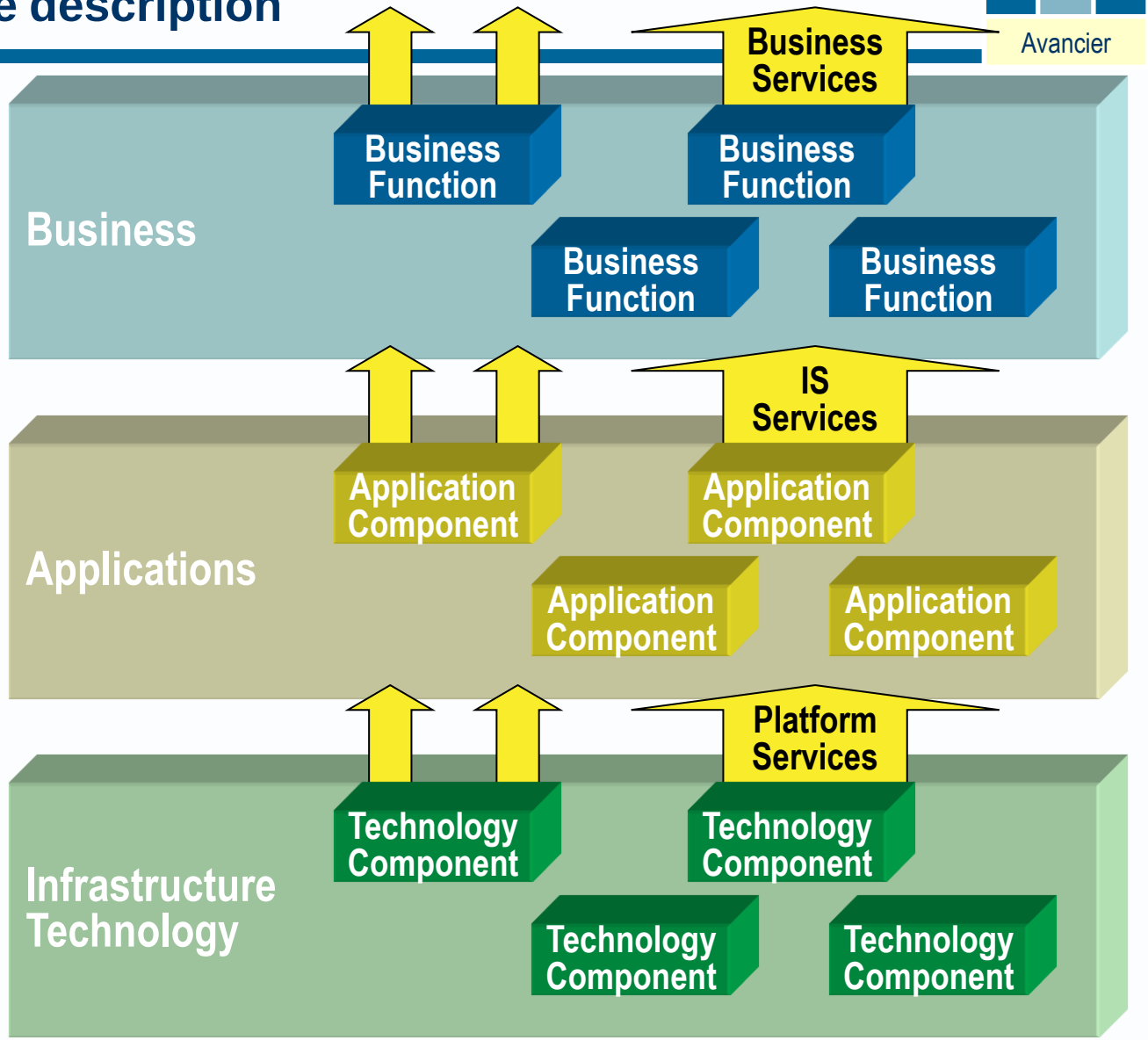


Products - for architecture description

Avancier

You will usually find three architecture domains arranged in layers.

Components (or building blocks) offer services.



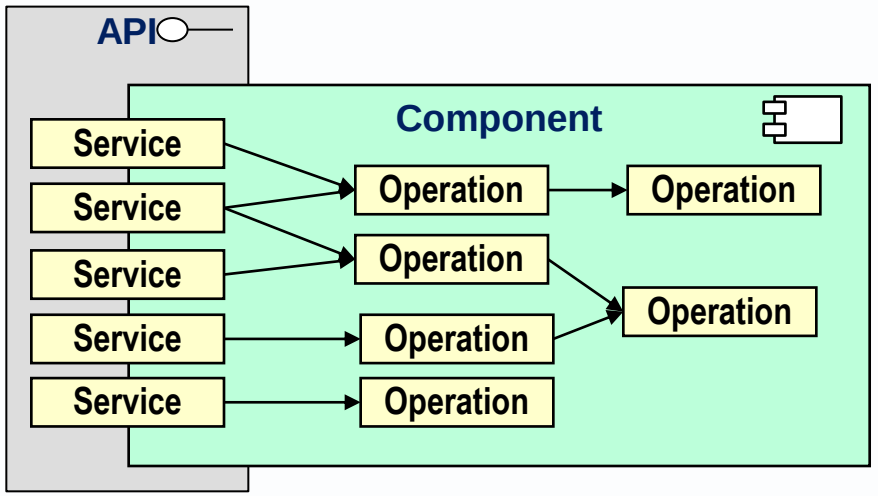
1.2: Architectural design patterns

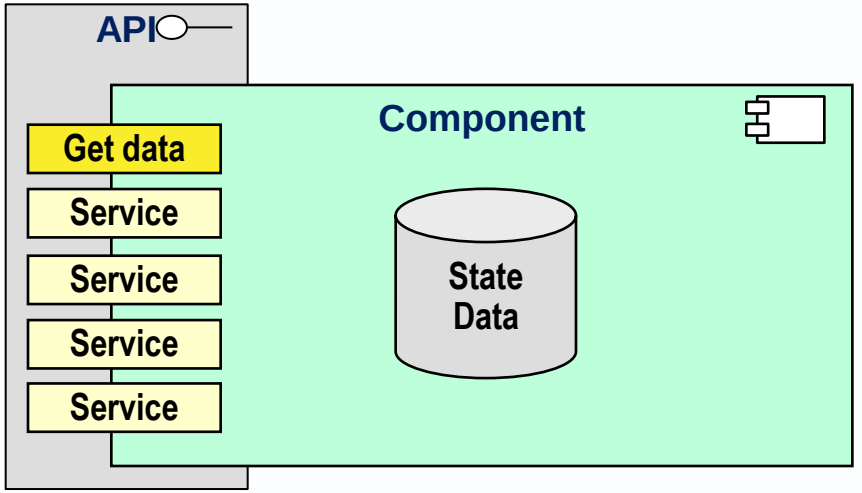
- Encapsulation
- Component-based design
- Design pattern

- Design pattern pair
  - Hierarchical or centralisation pattern / Anarchical or distribution pattern
  - Hierarchical structure / Network structure
  - Hub and spoke / Point to point
  - Fork/Orchestration / Chain/Choreography
  - Hierarchical layering / Peer-to-peer

Encapsulation

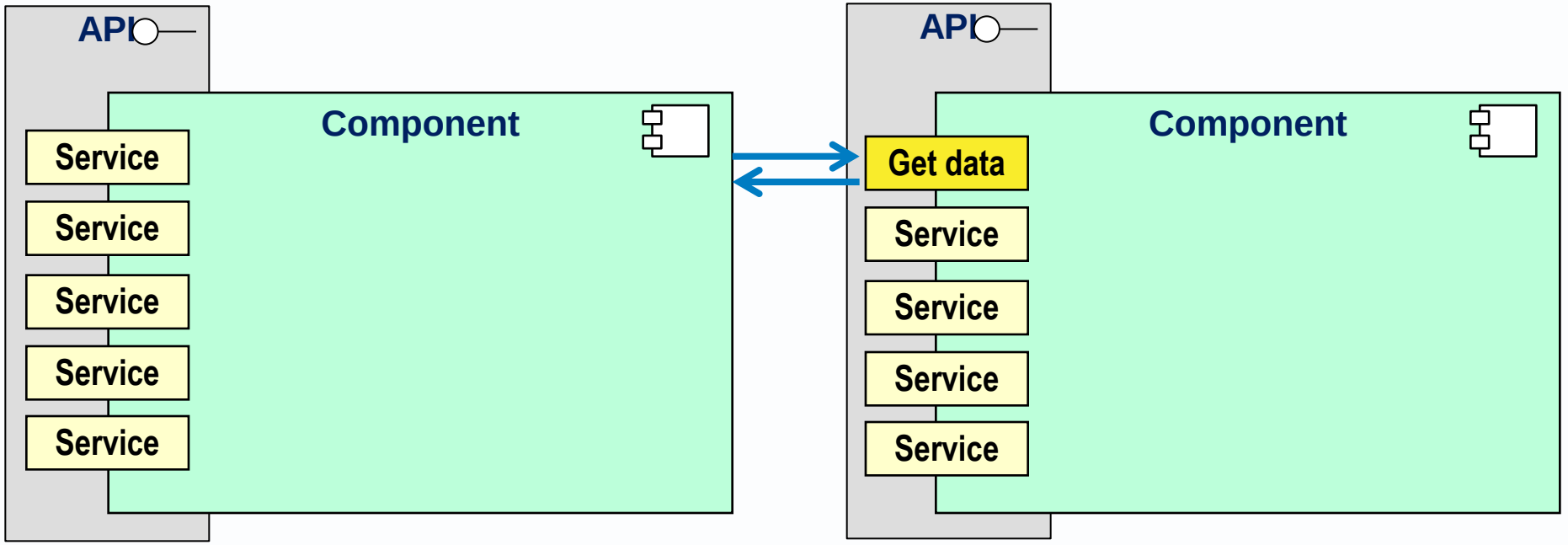
- **Encapsulation** [a technique] for defining a system or component by its input/output interface, by the discrete events it responds to and the services it can offer.
- It hides inner workings or processes from outsiders.
- It hides internal resources (e.g. data) from external entities, so the only way to access those resources is through the interface of the component.





Component-based design

- **Component-based design:** [a technique] for defining a system in terms of interacting components (or building blocks).
- The components are defined by interfaces that list the services they provide, and the services (or server components) they require.



Design pattern pairs

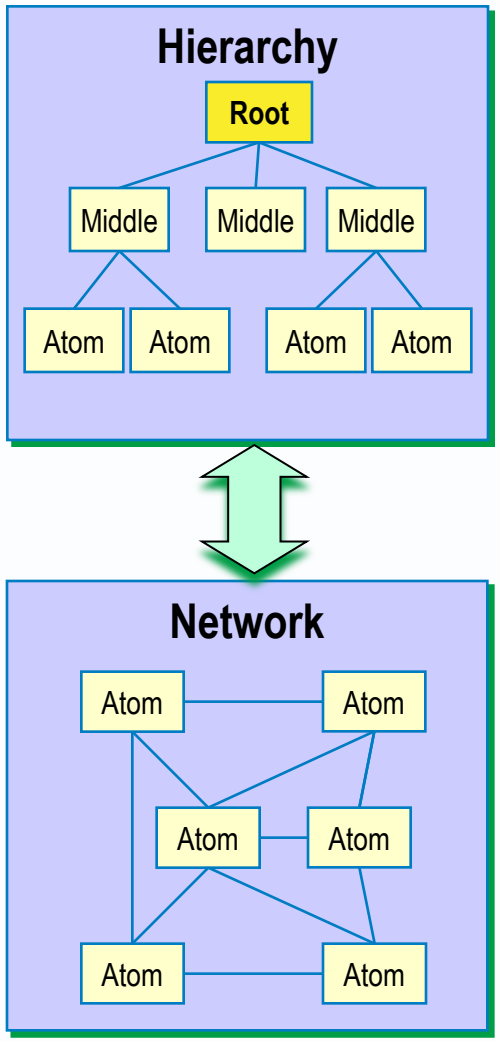
- **Design pattern:** a shape that commonly appears in solution designs, a tried and tested design that is tailored to address particular problems or requirements.

- **Design pattern pair:** a pair of contrasting patterns that each suit different situations. Architects choose between alternative patterns by trading off their pros and cons.

- **Hierarchical or centralisation pattern:** [a design pattern] that centralises control in one place or component.

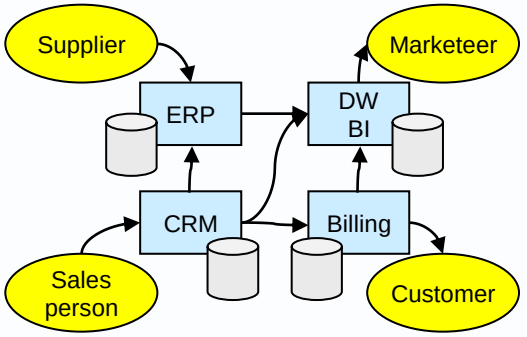
- **Anarchical or distribution pattern:** [a design pattern] that distributes control to many places or components.

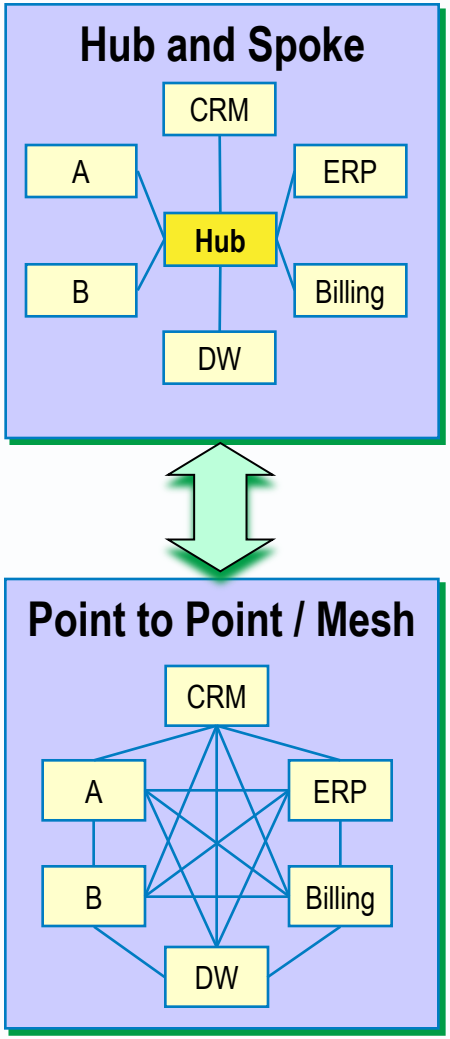
Centralisation of control	Distribution of control
in one place or component.	between many places or components.
Hierarchy	Anarchy or network
Hub and spoke	Point to point or mesh
Client-server	Peer-to-peer
Fork or Orchestration	Chain or Choreography



- **Hierarchical structure:** [a design pattern] in which a node is divided recursively into subordinate nodes.
- Rules of thumb include: divide a node into about seven subordinate nodes, stop decomposition at three or four levels (or 1,000 atomic nodes).

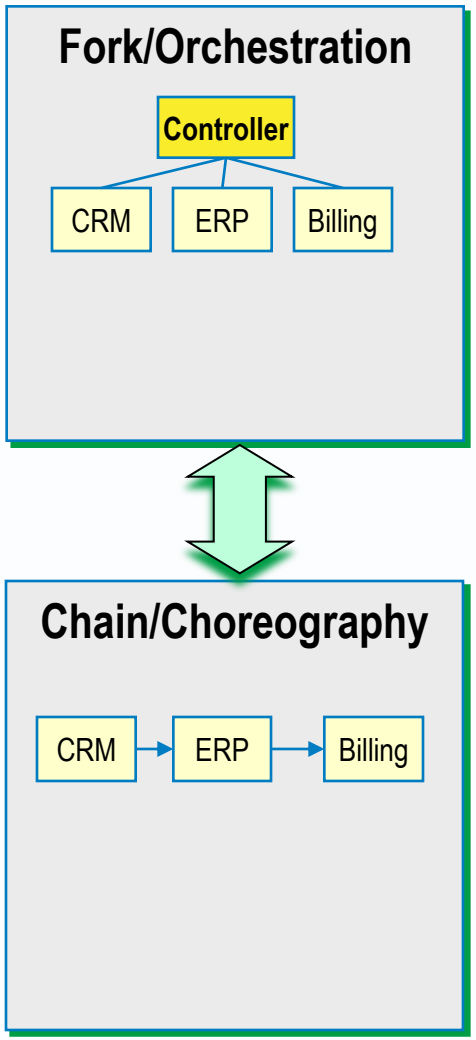
- **Network structure:** [a design pattern] in which a node can be connected to any other node.





- **Hub and spoke:** [a design pattern] in which components communicate via a mediator:
- good where inter-component communication is many to many, endpoints are volatile;
- can be more complex and slower.

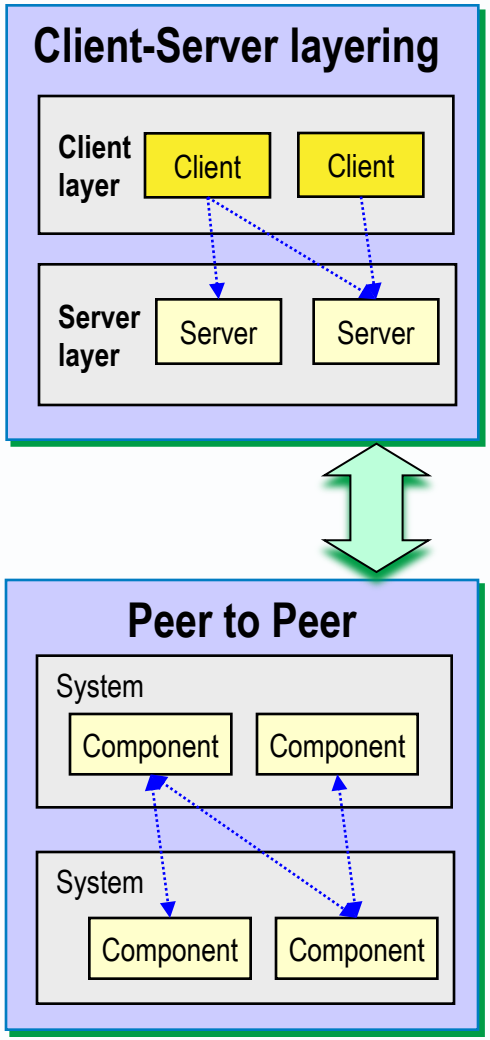
- **Point to point:** [a design pattern] in which components talk to each other directly:
- good where inter-component communication is 1 to 1, endpoints are stable;
- can be faster and simpler.



- **Fork/Orchestration:** [a design pattern] in which one component controls or schedules other components.
- It centralises intelligence about a process, a workflow that supervises and orchestrates the procedure.

- **Chain/Choreography:** [a design pattern] in which components interact and cooperate.
- It distributes intelligence about a process. Each component does part of the work, then calls the next component (cf. pipe and filter).

Client-Server v Peer to Peer



Client-server layering: [a design pattern] in which components in a client layer delegate work to components in a server layer, and so abstract from that layer**.**

This pattern is widely used to structure complicated systems (machines, networks, software applications and enterprise architecture).

For example, the domains of business, application and infrastructure can be seen as client-server layers.

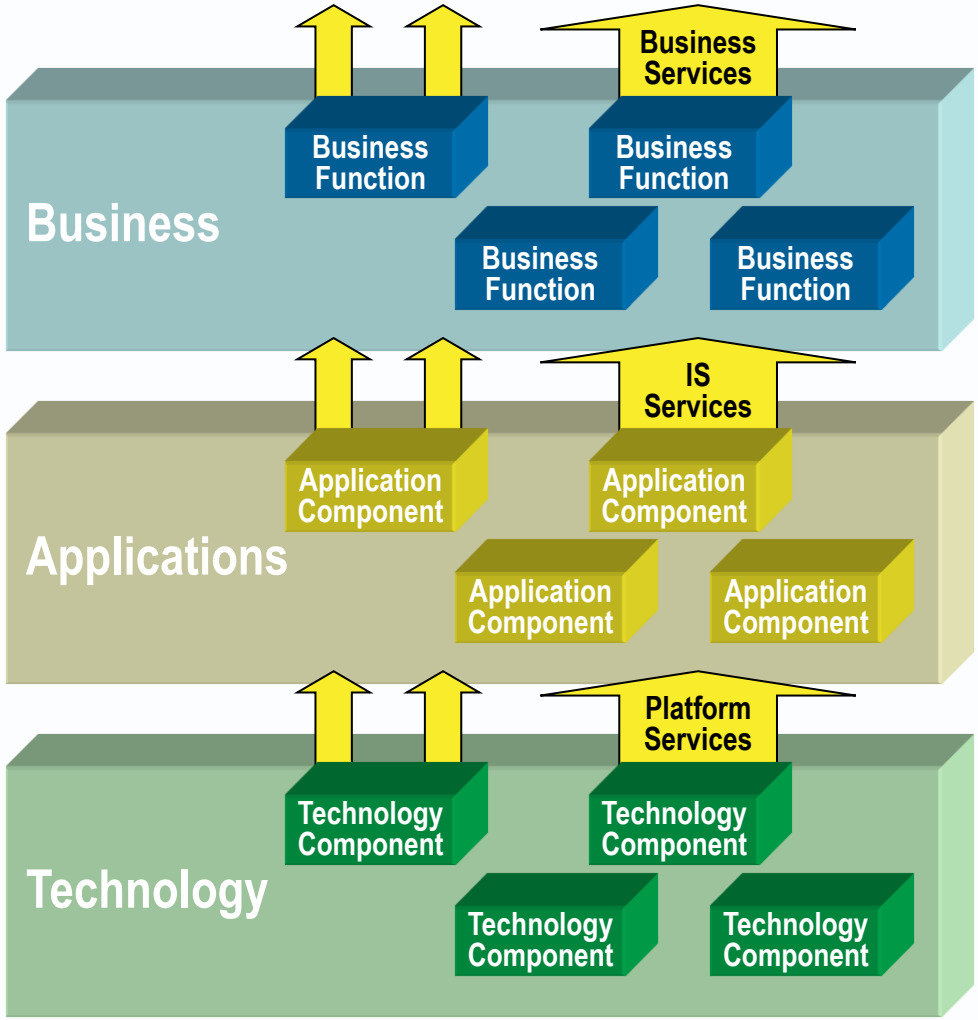
Peer-to-peer: [a design pattern] in which components can delegate to work to each other.
Sometimes said to be a bad thing, a fragile and unstable structure, difficult to understand and maintain.
Cyclic dependencies can undermine testability, parallel development, and reuse.

EA as a client-server hierarchy

Business, application and infrastructure architecture domains viewed as a hierarchically layered structure.

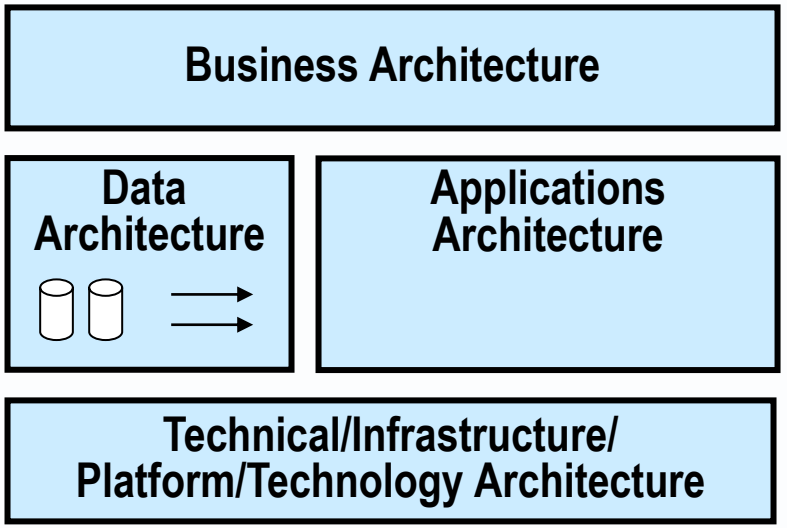
Components in one layer invoke services offered by components in the layer below.

A useful mental model for enterprise and solution architects alike.



1.3: Architecture domains/layers as views

- The four architecture domains below were established in
- the PRISM report (1986) and
- “EA Planning” (Stephen Spewak, 1993)
- and are now the basis of countless EA frameworks, including TOGAF.

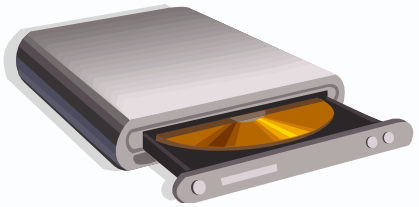


Business architecture

- [a view] that identifies and relates business elements:
  - business products and services
  - business processes (scenarios, value streams) that maintain resources and deliver services
  - business resources, roles and actors needed to perform processes.

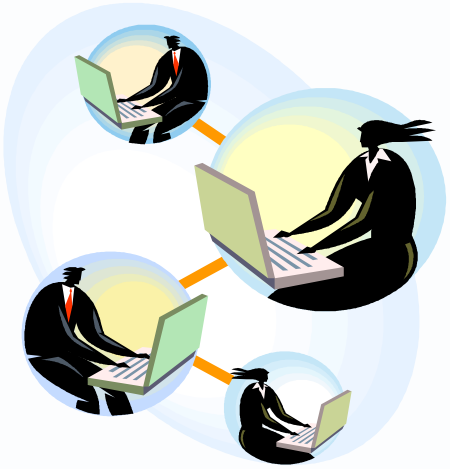
- Business elements may be mapped to goals and locations, to business data and applications.
- EA is concerned with standardisation and integration of business roles and processes.

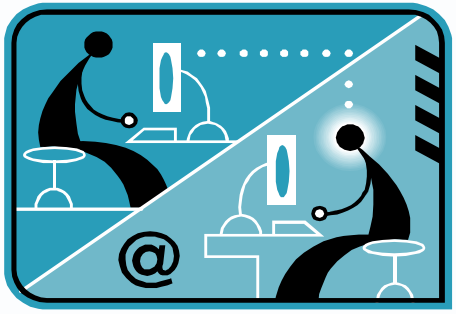
Data architecture



- [a view] that identifies and relates data elements:
  - identifies data stores and flows created and used by business activities
  - describes data stores and the data structures they contain
  - describes data flows and the data structures they contain
  - describes data qualities: data types, confidentiality, integrity and availability.
- Data elements may be mapped to business activities and to applications.
- EA is concerned with standardisation and integration of data between systems.

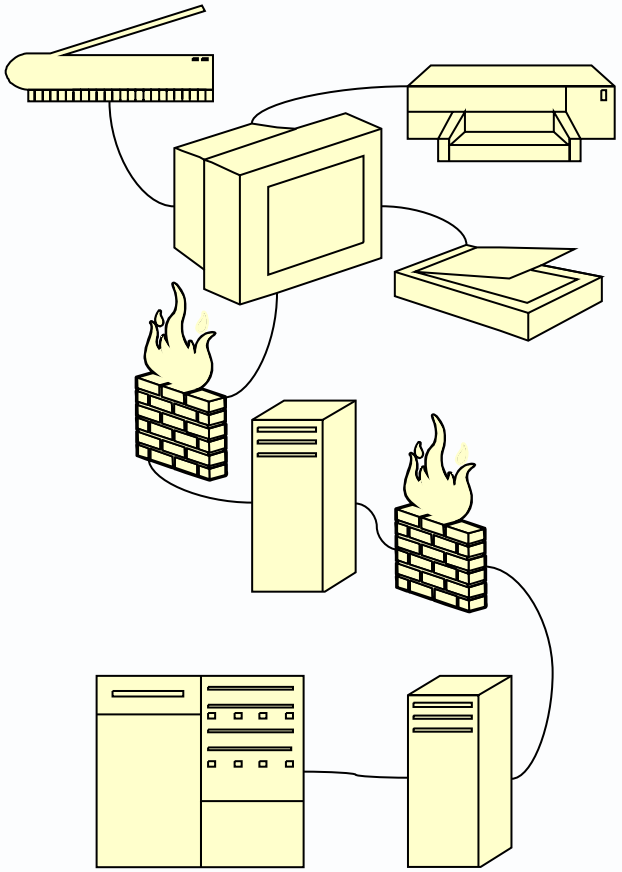
Applications architecture





- [a view] that identifies and relates application elements:
  - identifies business applications and their uses to support business activities.
  - describes coarse-grained applications rather than fine-grained software components.
  - identifies application use cases
  - identifies data flows consumed and produced by applications.
- Application elements may be mapped to business activities and to platform applications.
- EA is concerned with standardisation, integration and life cycles of business applications.

Infrastructure/platform architecture



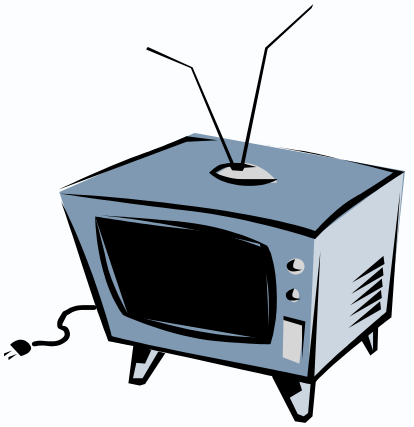
- [a view] that identifies and relates platform elements
  - identifies platform applications and the services they offer to business applications
  - relates business applications to the platform applications they need
  - defines the client and server nodes that applications are deployed on
  - defines the protocols and networks by which nodes are connected.
- Infrastructure elements may be mapped to business applications, data stores and data flows.
- EA is concerned with standardisation, integration and life cycles of platform applications.

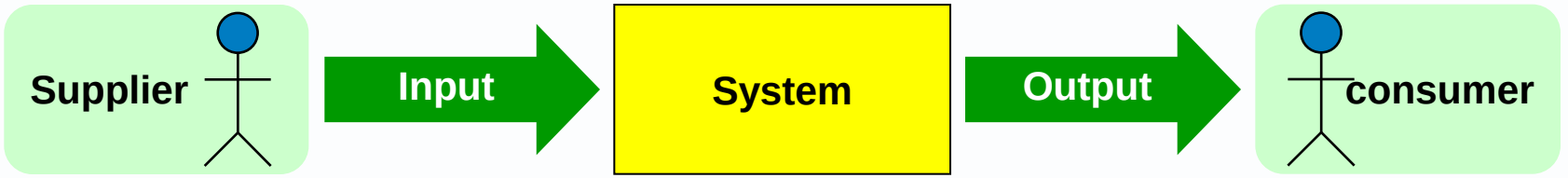
1.4 System elements

- System elements can be divided into structure elements and behaviour elements

System:

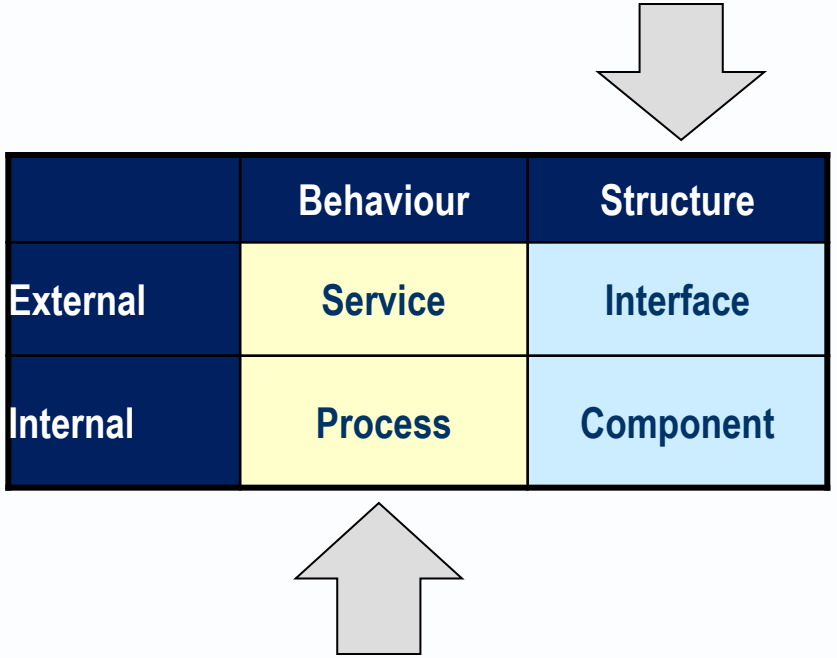
- a bounded collection of interrelated elements.
- An activity system is a network of actors/components playing roles in processes to produce desired effects by maintaining system state and/or transforming inputs into outputs.
- It can be encapsulated behind an interface.
- It is open, meaning it interacts with entities and events in its environment.





Structure v Behavior

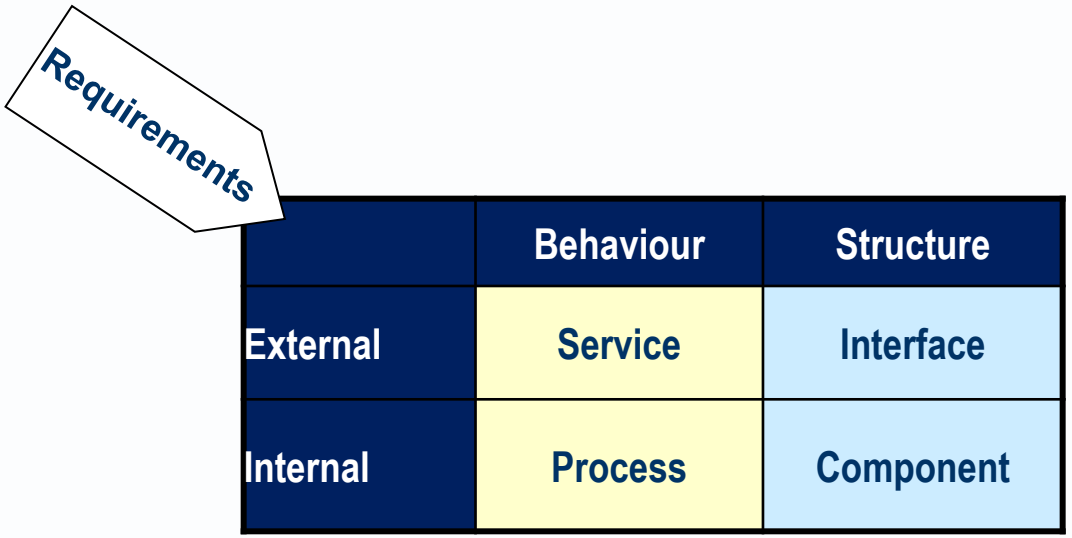
- **Structural view:** [a description] that shows what a system is made of. It shows actors/components and how they are connected in structures or accessed via interfaces.



- **Behavioural view:** [a description] that shows what a system does. It shows services and/or processes that run from start to end.

System element:

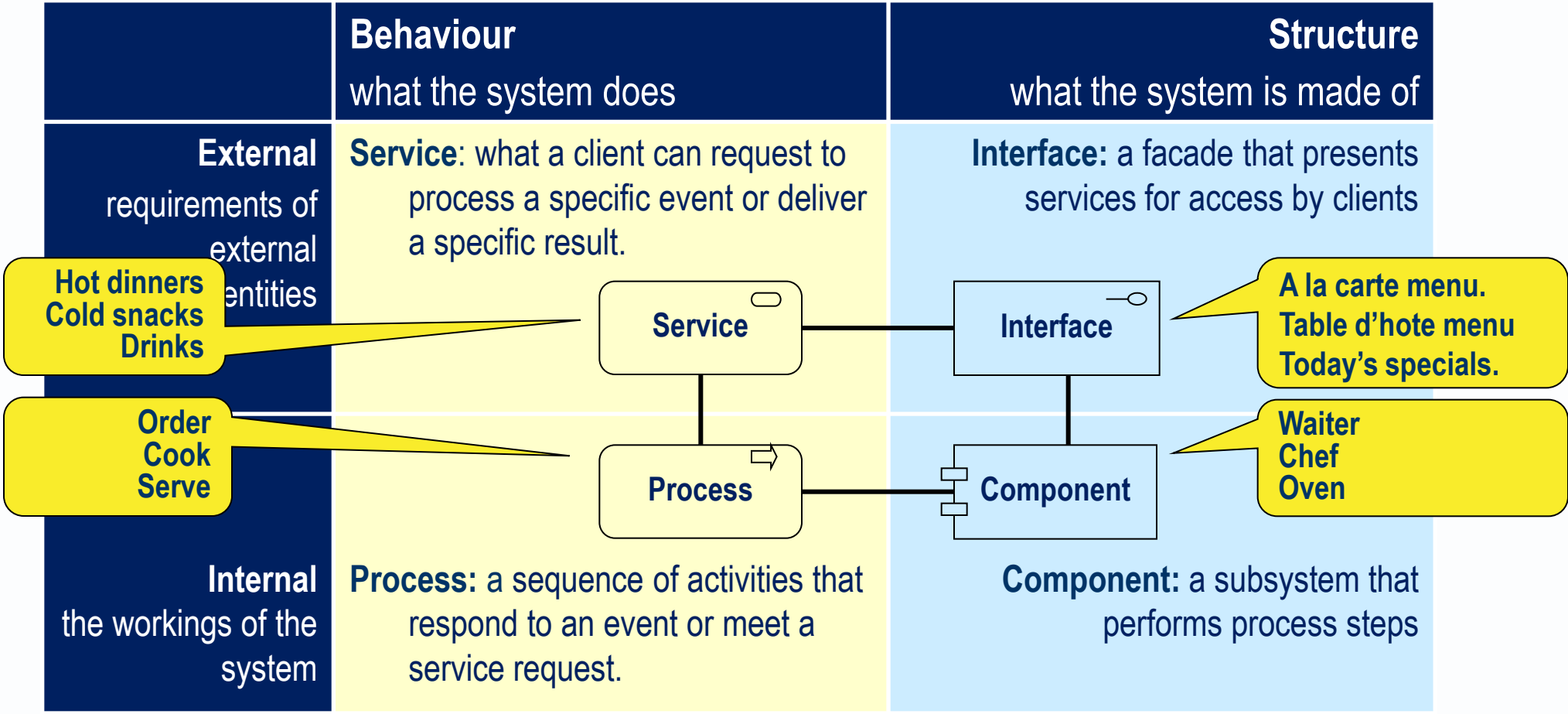
- a unit of an operational system that can be described.



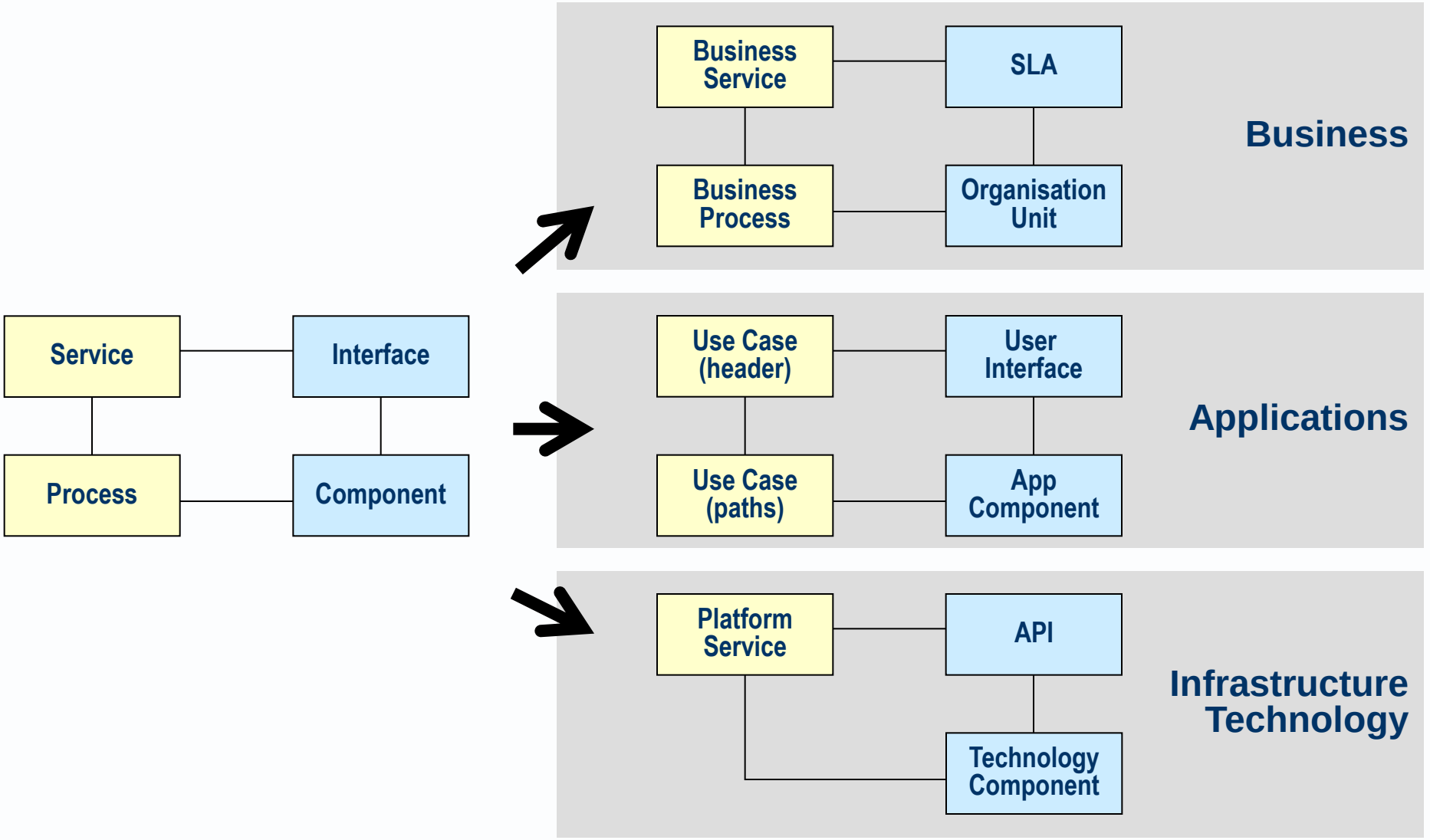
- Systems elements can be arranged in a grid as shown above.
- Since one person's system is another's subsystem, this model can be applied at any level of granularity.

BCS vocabulary for generic meta model of basic system elements

Avancier

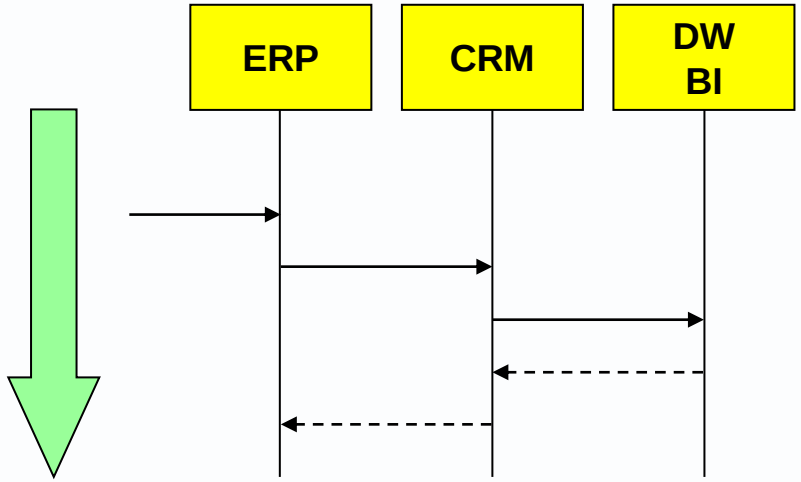


Combining sources



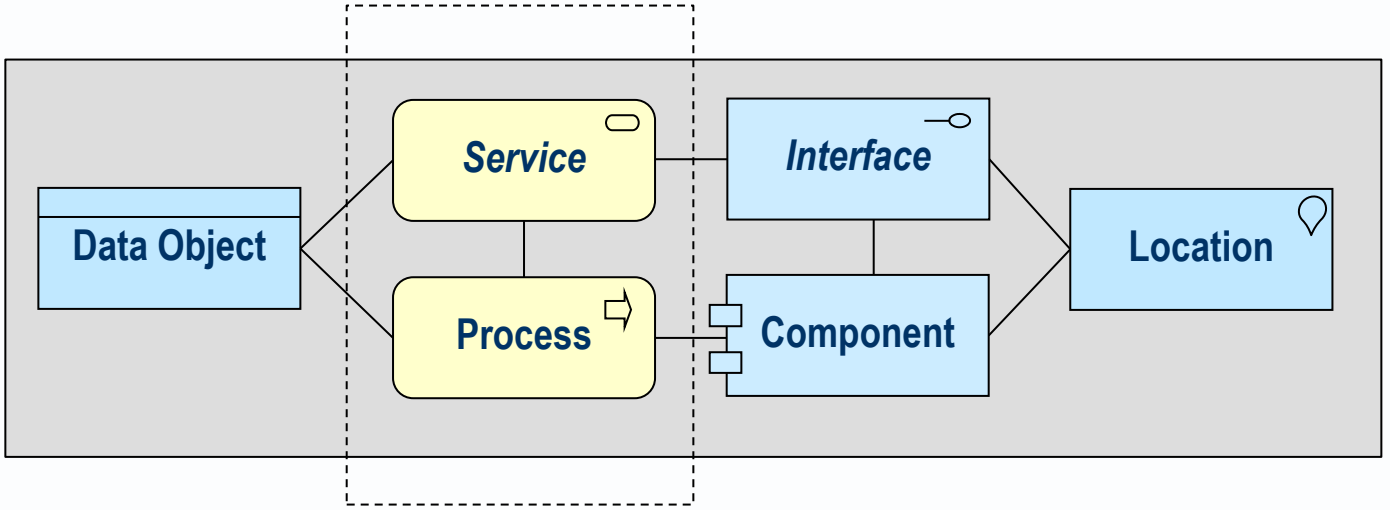
Behaviour element

- A thing that happens.
- A unit of activity triggered by an event and (providing its preconditions are met) terminating with the production of an output or other response.



Behaviors run over time

- Services and processes have a time-period



Service:

Avancier

- A discrete stimulus-response behaviour, specified in a service contract that encapsulates process(es) needed to realise the service.
- A service contract is divisible into three main parts.
  - The signature: the service name, inputs and outputs.
  - The rules: the preconditions and post conditions of the service.
  - The non-functional characteristics: including performance and commercial conditions.

Service Contract	**Business Service**	**999**
Signature	**Name**	**Cut hair**
	Input	**Hair length**
	Output	**Shorter hair (see also post conditions)**
Semantics or rules	**Preconditions**	**Barbershop open and barber ready**
	Post conditions	**Money received. Resource wear.**
Non-Functional Requirements	**Response time**	**45 minutes**
	Throughput	**6 per hour per shop**
	Availability	**90% waiting times less than 20 minutes from 09.00 to 17.00**

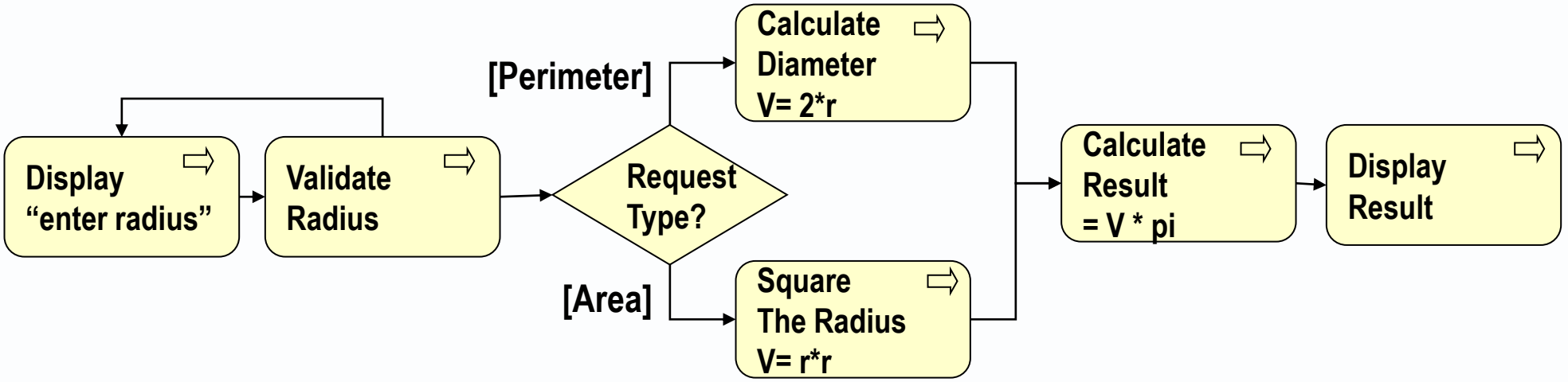
Infrastructure service contract example

Avancier

Service contract for FTP “get” operation		
Signature	Name	get
	Inputs	Remote file name Local file name
	Outputs or results	Reply = OK or Fail (see post conditions)
Semantics or rules	Preconditions - the state of the system in which the event is allowed	Remote computer can be reached. Remote file exists in the current remote directory.
	Post conditions - the state of the system after the event is complete	Remote file copied to (or on top of) local file current local directory.
Non-functionals	Response time	30 seconds
	Throughput	20 per minute
	Availability	99.99%
	Integrity	100% perfect file copy
	Scalability	Up to 100 per minute
	Security	No encryption
	Serviceability	
	Etc. Other non-functionals, dependencies and commercials.	

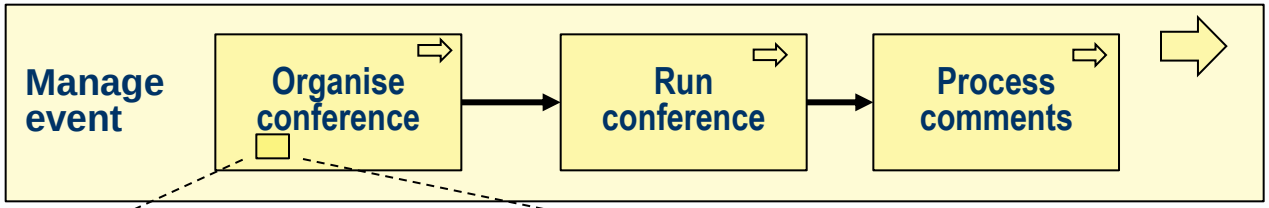
Process:

- A sequence of activities, performed by one or more components, that delivers a service or result of value.
- There are human processes, computer processes and hybrid human-computer processes (sometimes called workflows or use cases).
- A process may be documented using a flow chart or interaction diagram.

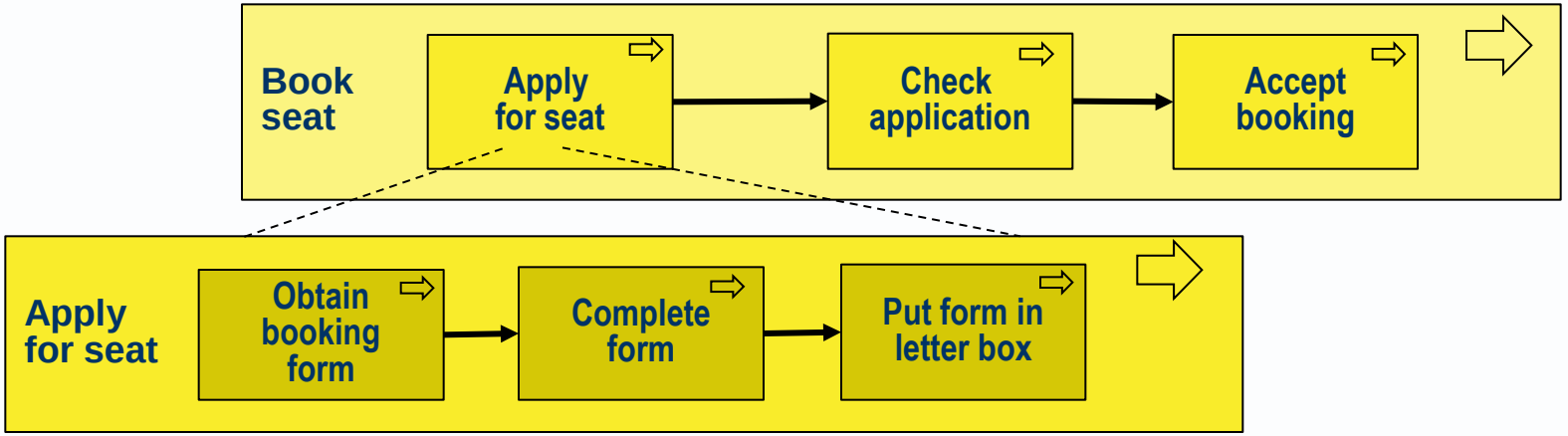


On process and service granularity

- A process may contain coarse-grained processes/steps/activities



- And/or fine-grained processes/steps/activities



- The same is true of services (which encapsulate processes)

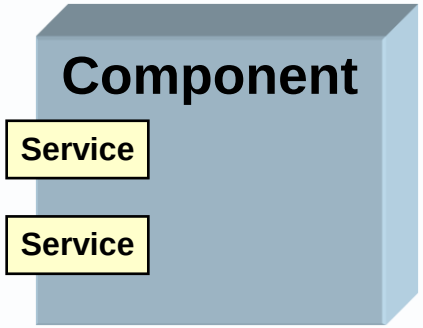
Structural element:

- what a system is made of, a unit which is addressable in space.

- **Active structure element:** A component responsible for performing behaviours, which may be specified at logical or physical levels.
- A subsystem that can be related to others by requesting or delivering services.
- It can be replaced by any other component with the same interface(s).

Components

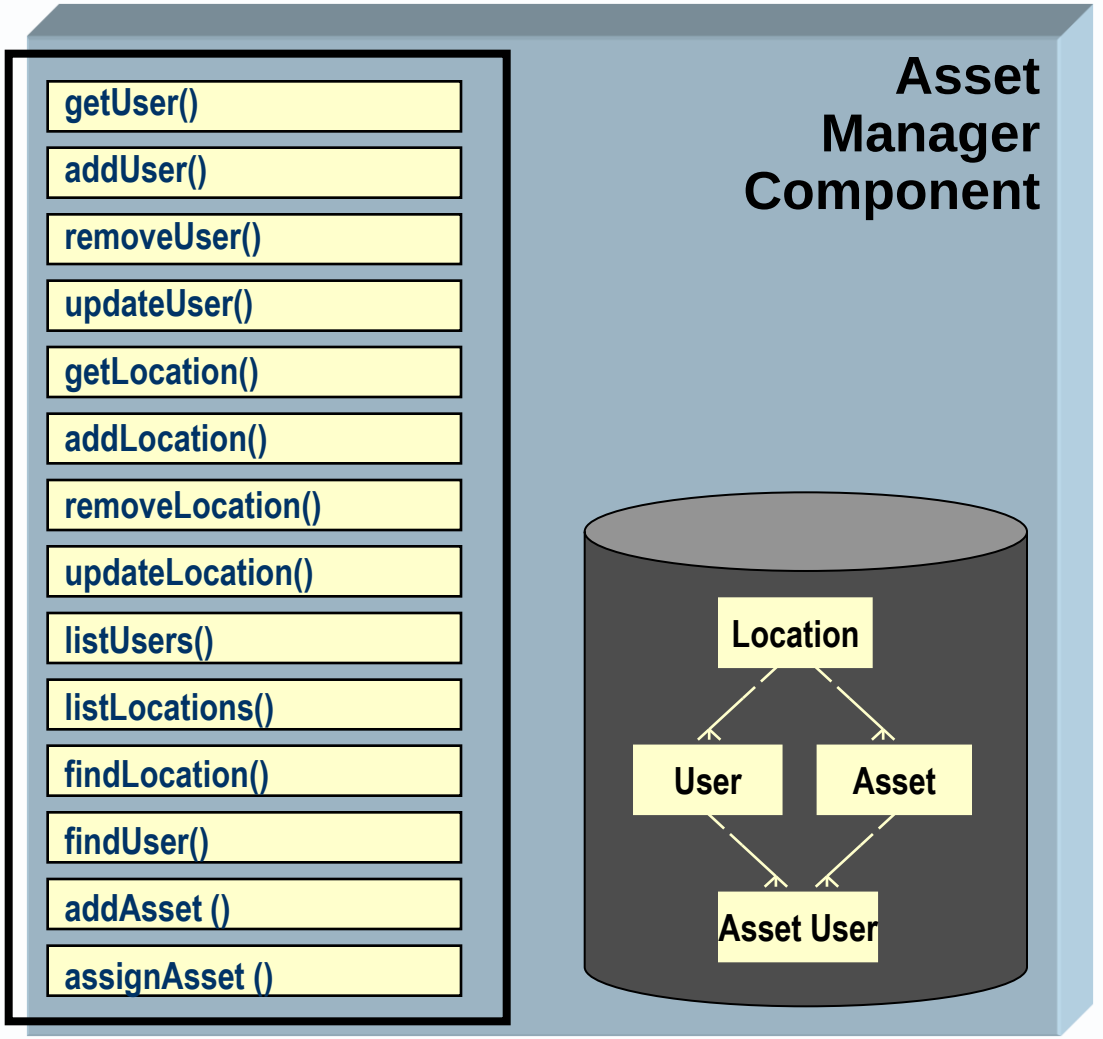
- **Logical component:** A structure element specified in terms of services provided and/or processes performed and abilities required.
  - Business component
  - Application component
  - Technology component



- **Physical component:** A structure element that can be hired, bought or built to realise a logical component and perform required behaviours.

Interface:

- a collection of services accessible by clients;
- it identifies services, may provide access to them, and hides what performs them.



File Transfer Protocol (FTP) – an interface offering many services

Avancier

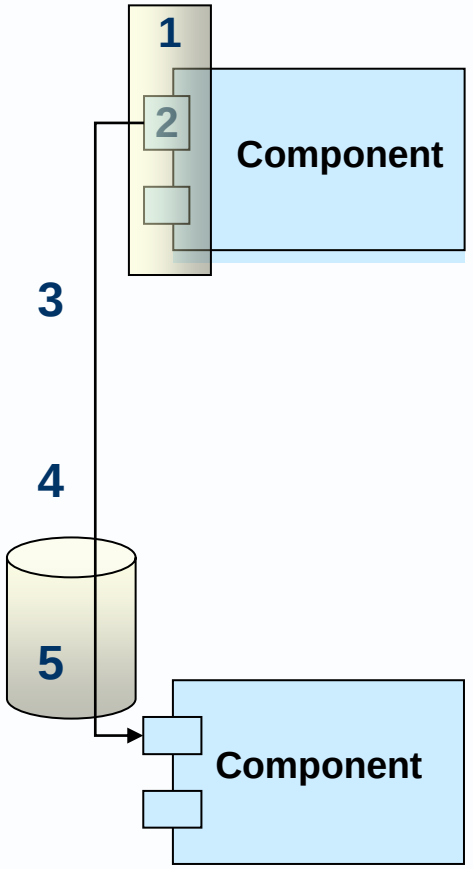
FTP	An interface implemented by a platform component whose role is to copy files to and from computers. Expressed below as in the common FTP utility program on a UNIX computer.
Service	**Summary description of service contract**
?	to request help or information about the FTP commands
ascii	set the mode of file transfer to ASCII
bye	exit the FTP environment (same as quit)
cd	change directory on the server computer
close	terminate a connection with another computer
delete	delete (remove) a file in the current remote directory (same as rm in UNIX)
get ABC DEF	copies file ABC in the current remote directory to a file named DEF in your current local directory.
get ABC	copies file ABC in the current remote directory to a file with the same name, in your current local directory.
help	request a list of all available FTP commands
mget	copy multiple files from the server computer to the client computer; you are prompted for a y/n answer before transferring each file
mput	copy multiple files from the client computer to the server computer; you are prompted for a y/n answer before transferring each file
open	open a connection with another computer
put	to copy one file from the client computer to the server computer
quit	exit the FTP environment (same as bye)
rmdir	to remove (delete) a directory in the current remote directory

Which do *you* mean by interface?

Avancier

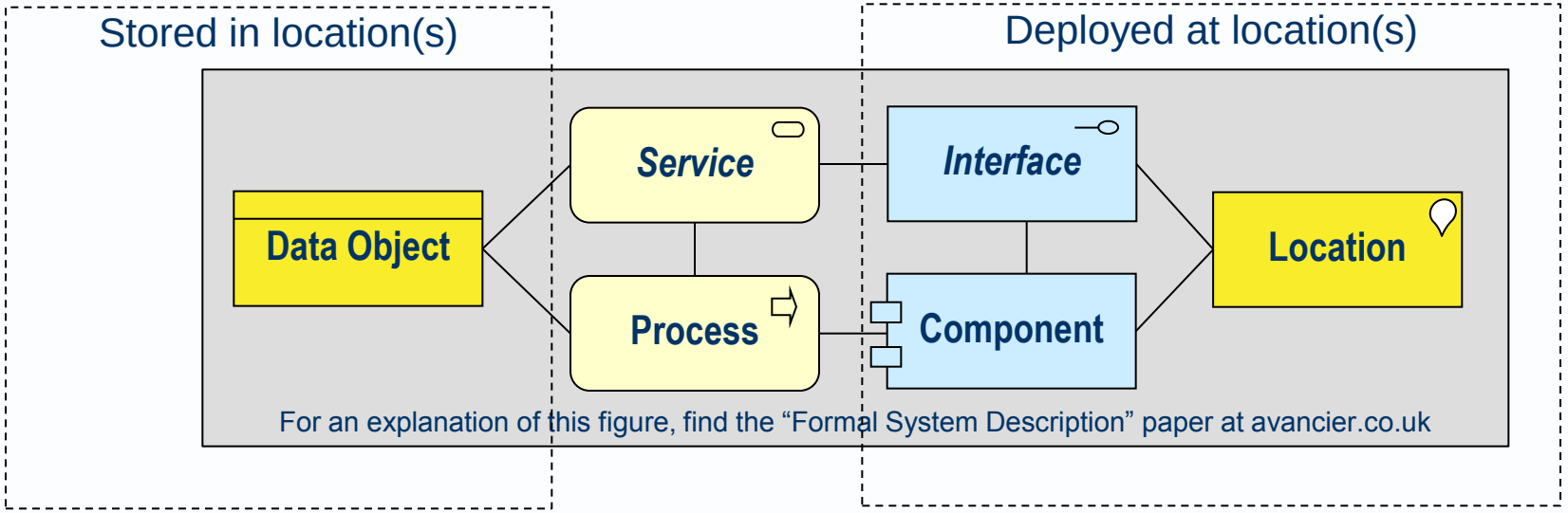
The means of connecting to a system, process or component.

1. A **list of services**, offered by one or more components.
2. The **signature** (name, inputs and outputs) of one service.
3. A **data flow** between sender and receiver components. (e.g. message, a document, a file of payment records.)
4. The **protocols** used to exchange data between components. (e.g. http, TCP, IP)
5. The **channel** via which data flows are passed. (e.g. telephone, HCI, internet).

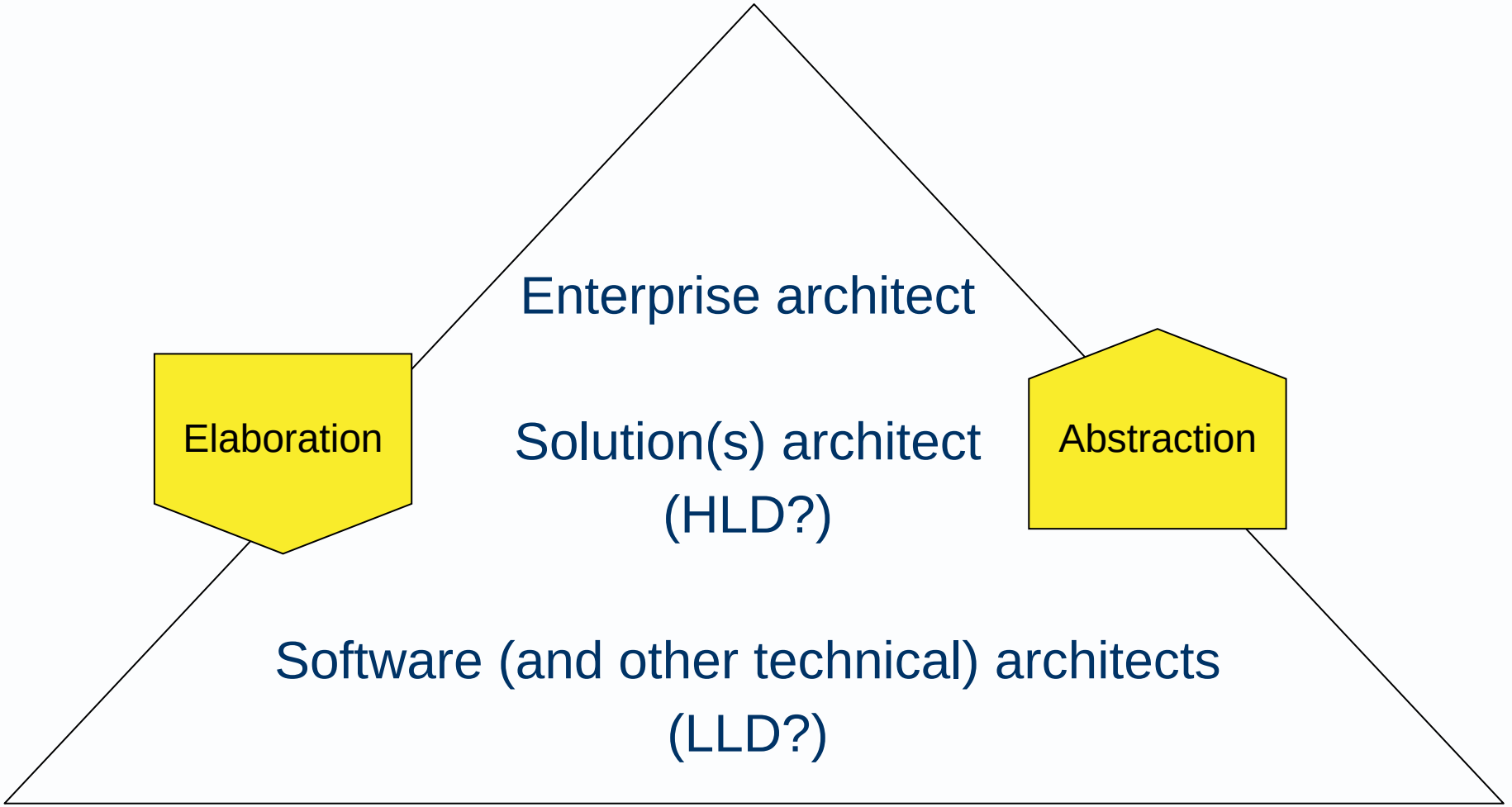


Passive structural element:

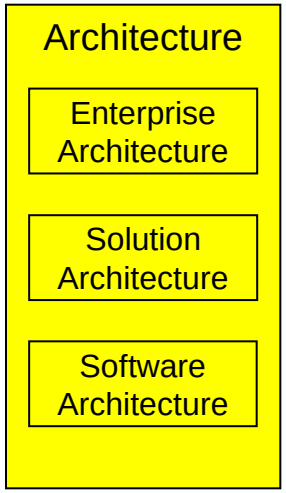
- a structural element that is acted upon or in.
- **Object:** an item or structure that is made, moved or modified. E.g. a data item, data structure, or any kind of structural component.
- **Location:** a place where actors/components and interfaces are found and work is done.

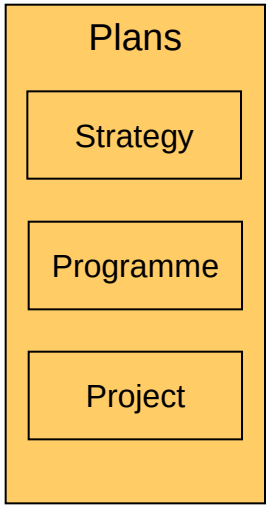


1.5: Architect responsibility level



Levels of architecture granularity





Architect level	Abstraction	Scope	Time	Target
Enterprise Architect	High-level	Wide	Far distant	Soft target
Solution Architect	Mid-level	Moderate	Medium time-frame	Flexible target
Software Architect	Low-level	Narrow	Short time-frame	Hard target

Enterprise architecture

Avancier

- [a responsibility level] that strives to:
  - take a long-term, strategic view
  - take an enterprise-wide view, treating the whole enterprise as a system
  - address cross-organisational concerns and goals
  - improve business processes by digitisation, standardisation and integration
  - exploit business data captured by business processes
  - identify potential innovations in business processes
  - understand the enterprise's estate and maintain enterprise architecture collateral
  - set out cross-organisational and/or strategic road maps and migration programmes.

Solution architecture

Avancier

- [a responsibility level] that
  - is relatively tactical: addresses specific problems and requirements, related to selected business processes and applications.
  - aims to ensure the quality of a particular solution delivery, in compliance with overarching goals, principles and standards where possible.
  - describe solutions and govern their delivery, usually at a project level
  - understands all domain/views well enough to work with others: e.g. a general solution architect may work in partnership with a lead business analyst and/or software architect
  - ensures the architecture of a system is sufficiently detailed for work to be planned and detailed design and building to proceed
  - shapes projects and then govern others according to agreed principles and plans
  - is responsible for delivery quality: focuses on critical success factors, especially non-functional qualities.

Software architecture

- [a responsibility level] that addresses the modularisation and integration of software components.
- It describes the fine-grained structure and behaviour of a business applications.
- It may follow principles and patterns for modularisation and integration.
-

- Aside: The principles and patterns of software architecture (e.g. encapsulation, cohesion and decoupling) are useful at higher levels. Architects should be familiar with such principles and able to apply them appropriately to their context.

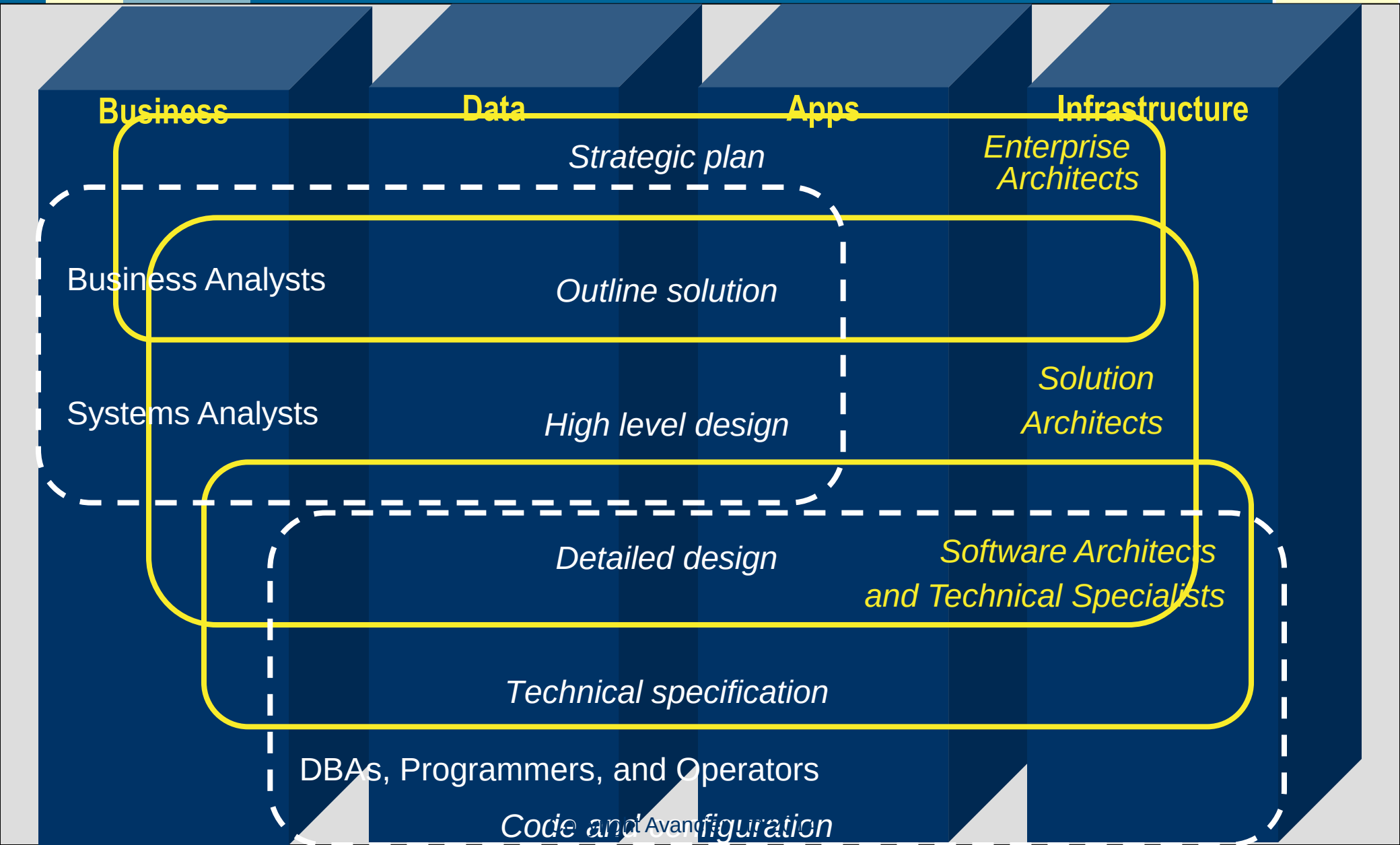
Architect Roles

- There is no industry standard!
- An EA team usually divides roles by **level** and/or by **domain**
- The power and the politics vary widely

The architects' working space				
Architecture facet Architecture level	Business Architecture	Data Architecture	Applications Architecture	Technology Architecture
Enterprise Architecture				
Solution Architecture				
Software Architecture & Technical Specialisms				

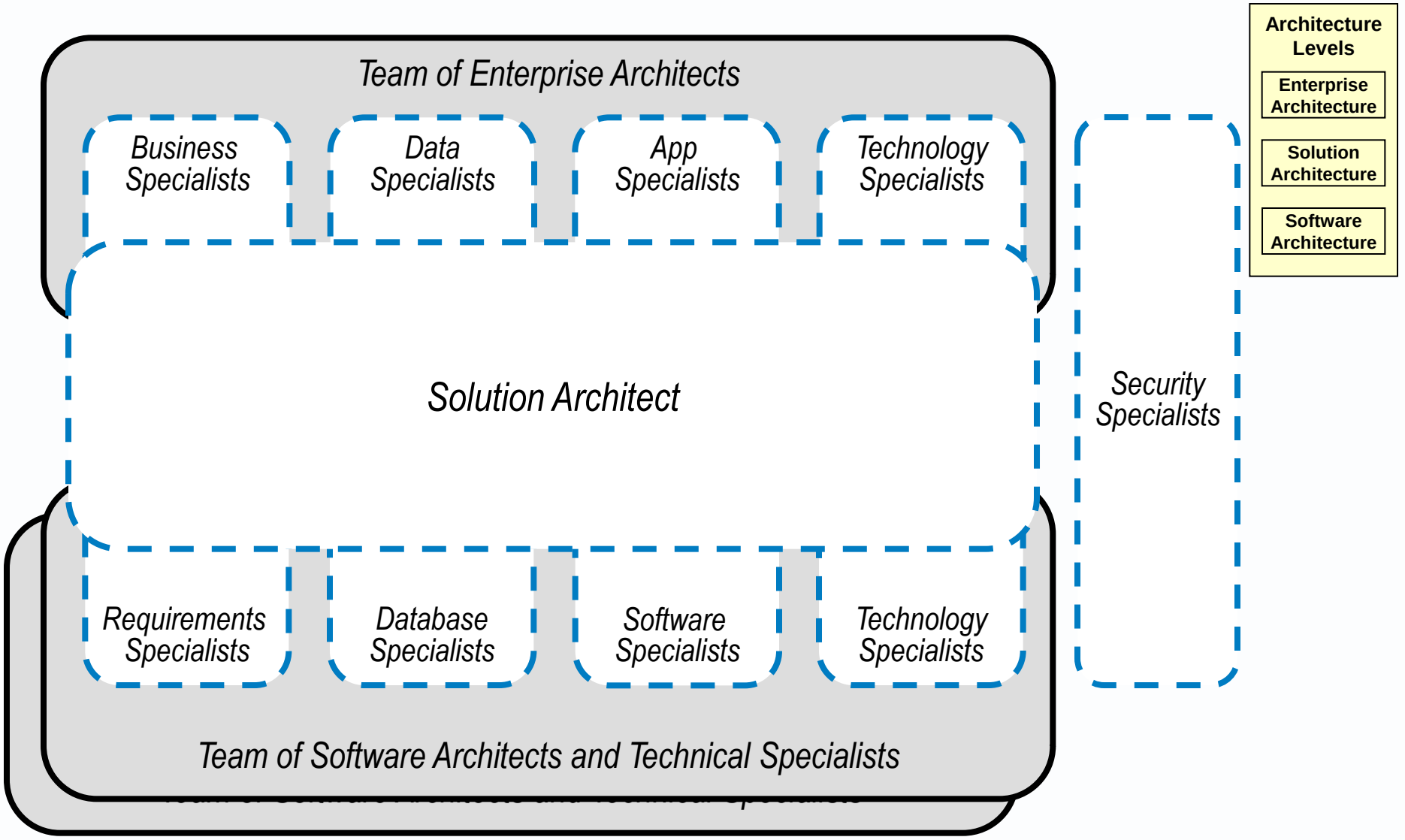
The four principal architecture domains

Mapping roles to a traditional specification hierarchy



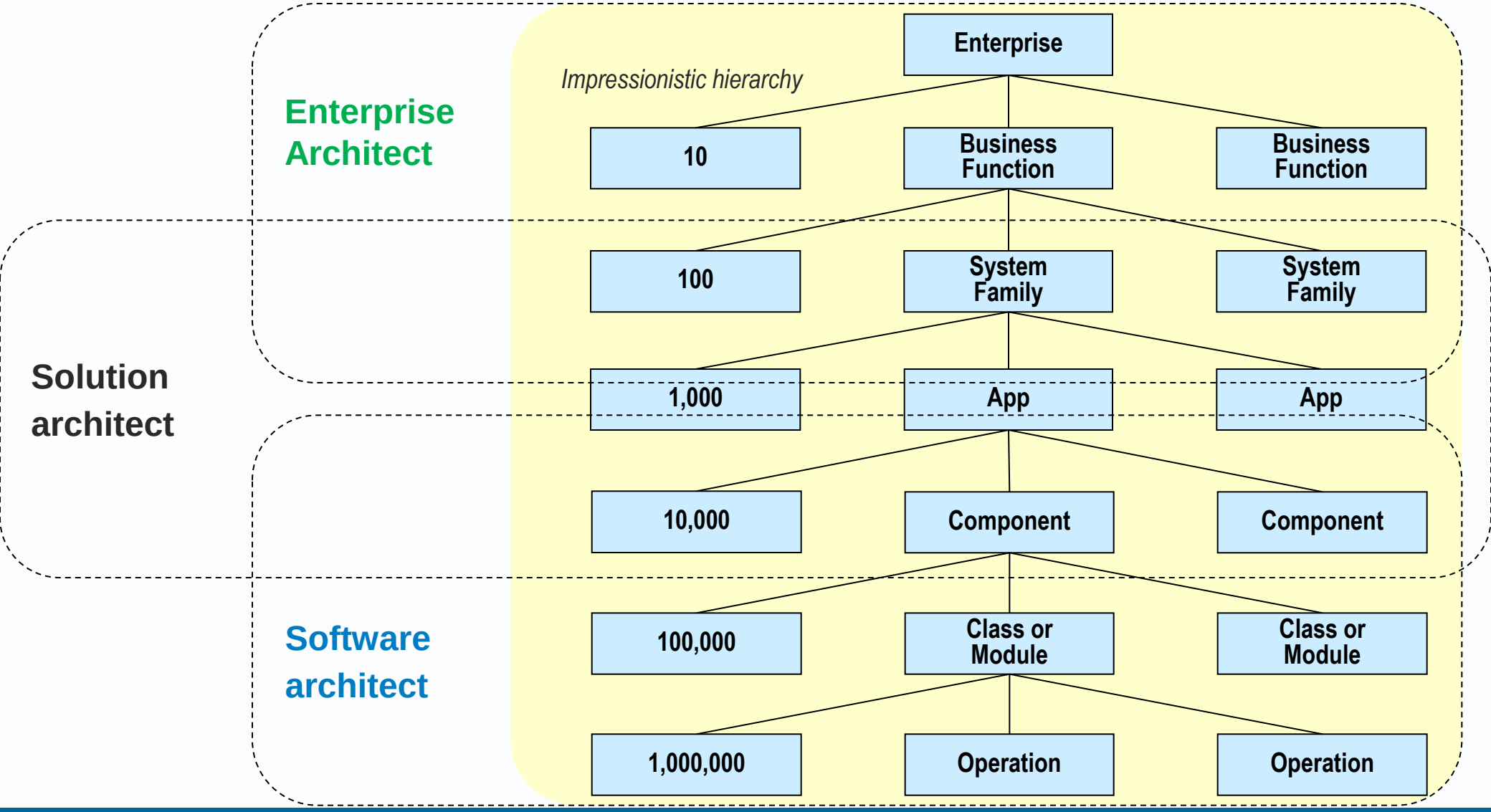
The central role of the solution architect

Avancier

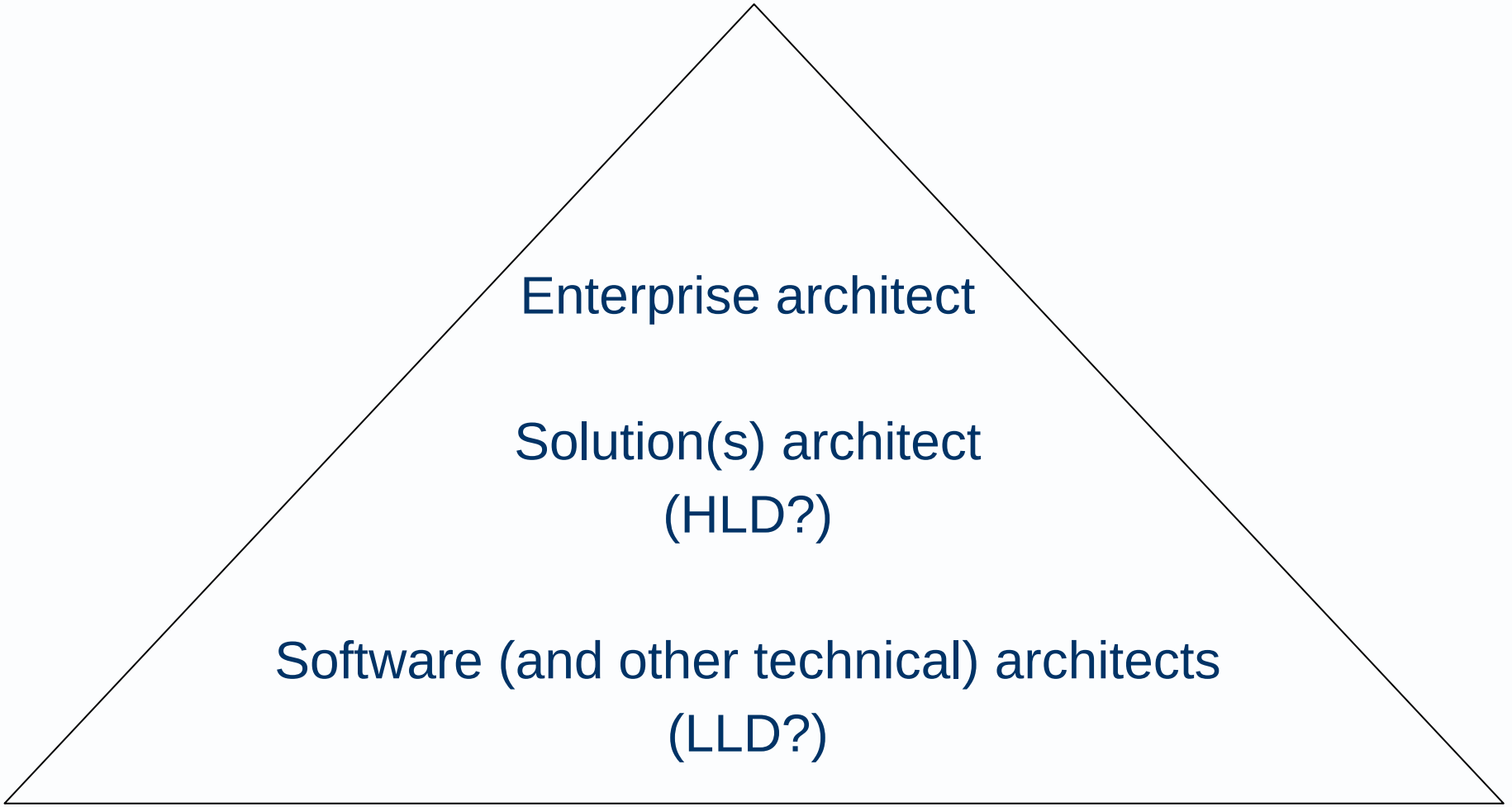


Mapping roles to system decomposition

Avancier



1.6: Architect goals and skills



Enterprise architect goals

1. Alignment of IS/IT to business processes, roles and goals.
2. Business agility and technical agility.
3. Standardisation: of processes, data, applications and technologies.
4. Integration: interoperation of processes, data, applications and technologies.
5. Enablement of strategically beneficial change through long-term planning.
6. Portability: vendor and technology independence.
7. Simpler systems and systems management.
8. Improved IS/IT procurement.
9. Improved IS/IT cost-effectiveness.

Solution architect goals

1. Support the goals of enterprise architects
2. Quality of IS/IT project deliverables.
3. Cost of IS/IT project deliverables (though a manager is usually *accountable*)
4. Timeliness of IS/IT project deliverables (though a manager is usually *accountable*)
5. Solution-level risk identification and mitigation.
6. Application integration and data integrity.
7. Conformance of solutions to non-functional and audit requirements.
8. Conformance of solutions to principles, standards, legislation.
9. Effective cooperation between managers and technicians.
10. Governance of detailed design to architecture principles and standards.

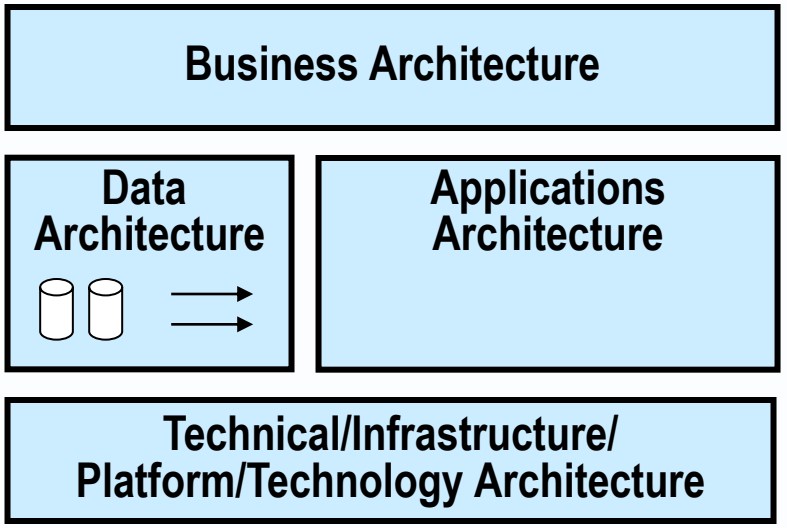
Architect knowledge and skills

1. Holistic understanding of business and technical goals.
2. Holistic understanding of business and technical environment
3. Broad technical knowledge – including current trends.
4. Broad methodology knowledge
5. Analysis of requirements and problems
6. Innovation.
7. Leadership.
8. Communication, political and soft skills (e.g. stakeholder management)
9. Awareness of project management and commercial risks and issues.

1.7: Other architecture domains

Avancier

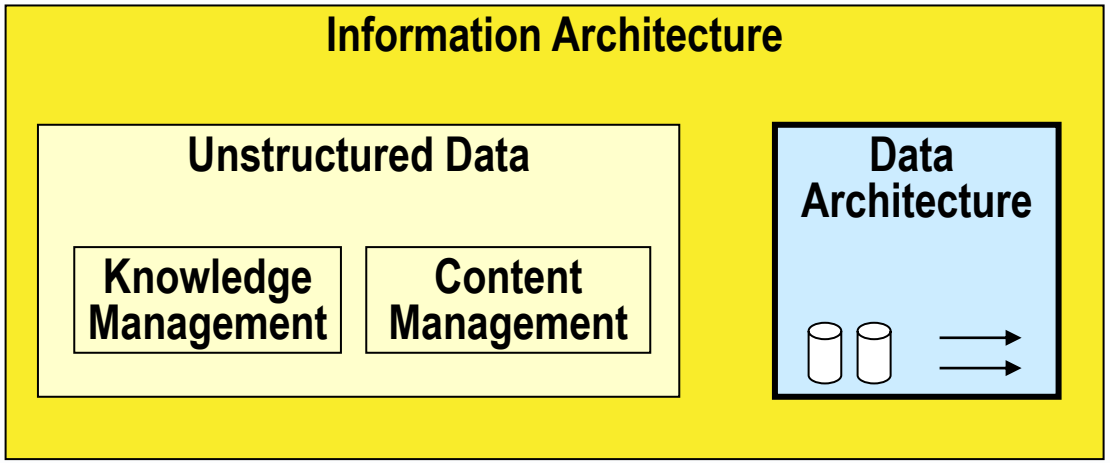
- In addition to the four primary architecture domains,



- four other architecture domains are named below.
  - Information architecture
  - Information systems architecture
  - Software architecture
  - Security architecture

Information architecture

- a term for the broad domain that includes structured data architecture, content management and knowledge management.

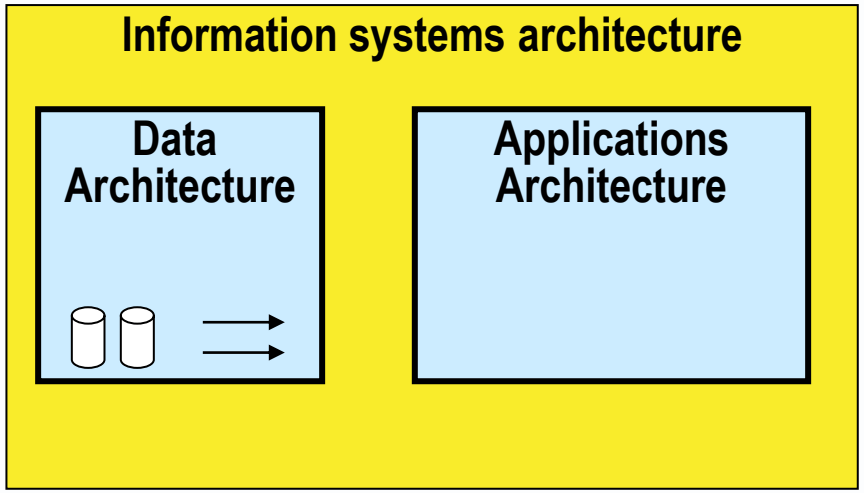


- EA frameworks (and this reference model) focus on structured data.

Information systems architecture

Avancier

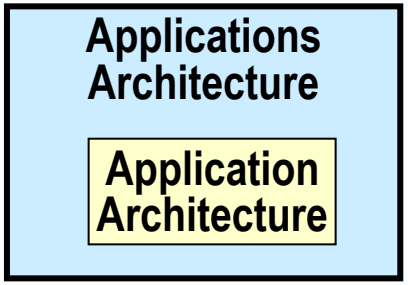
- a term often used to mean the combination of applications architecture and data architecture.



- It depends on but does not include the technology platform.

Software (aka application) architecture

- the internal structure or modularisation of a software application.



- This is software architecture at the lowest level of granularity (as discussed for example in “Patterns of enterprise application architecture” by Martin Fowler).
-
- It is usually below the level of modularity that enterprise and solution architects define. However, there is no rigid dividing line

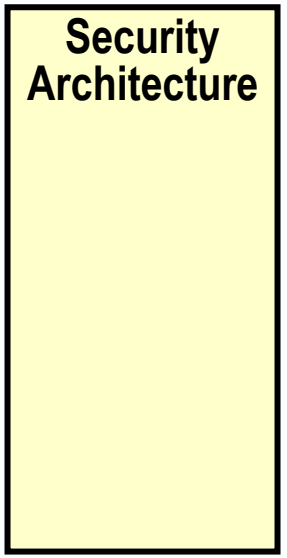
Security architecture

Avancier

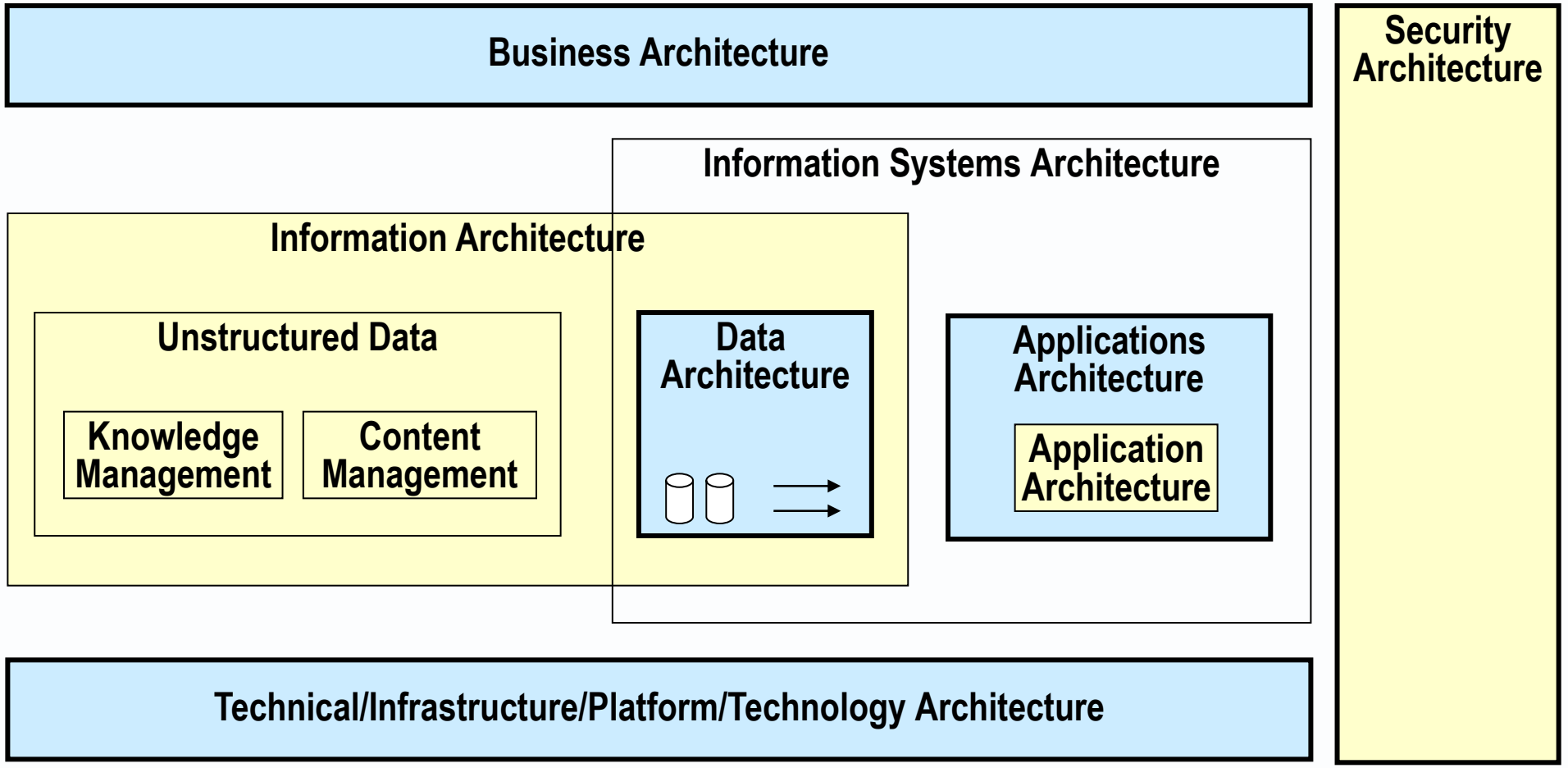
- a term covering the various design features designed to protect a system from unauthorised access.

- Not a cohesive architecture on its own so much as features of the other architecture domains – business, data, applications and infrastructure.

- Design considerations include
  - Human and organisational security considerations
    - E.g. gates, guards and guns
  - Data security considerations
    - E.g. encryption
  - Application security considerations
    - E.g. user roles, identities, passwords
  - Infrastructure security
    - E.g. firewalls



Put that all together



LEFT OVERS

Use case definition = process flow inside a service contract

Avancier

Service Contract	Business Service Id	999
Signature	Name	Calculate Area
	Input	Radius
	Output	Area
Trigger event		Service invocation
Functional rules or semantics	Preconditions	Radius is numeric
	Post conditions	Area = pi * Radius squared
Process flow		1. Variable = Radius * Radius 2. Area = pi * Variable
Non-functional requirements	Response time	0.1 second
	Throughput	one at a time
	Availability	100% of time the calculator is switched on
	Integrity:	100% accuracy
	Scalability:	n/a
	Security	n/a
	Other non-functionals, dependencies and commercials	

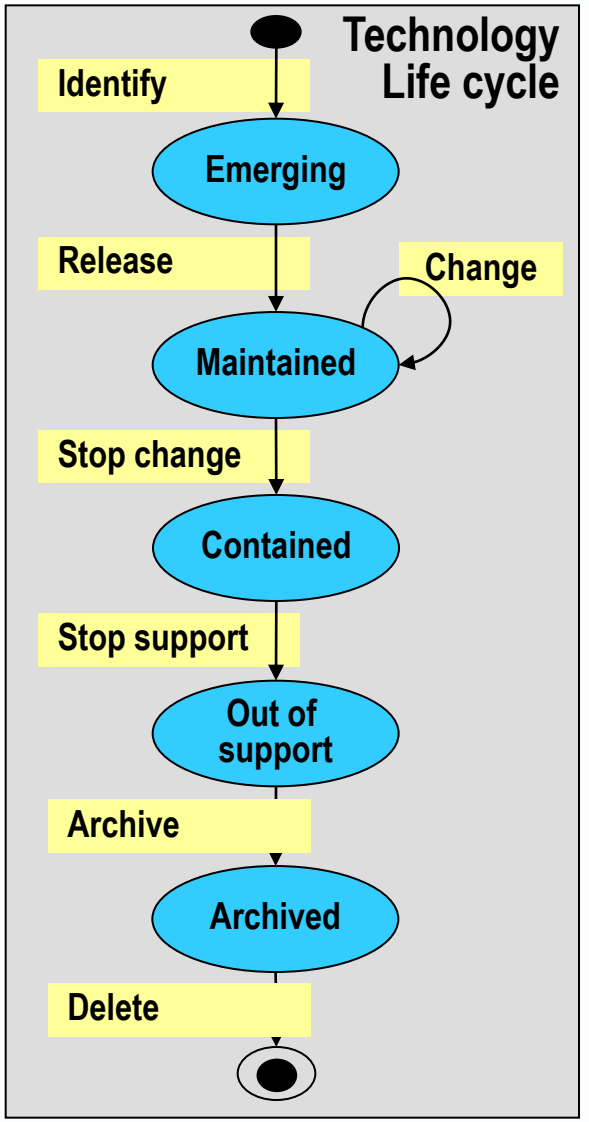
Service

Process

Foundation concepts

Avancier

Life cycle	A process from birth to death [of a thing] E.g. the life of: ▶ a system from conception through deployment and use to removal ▶ a project from conception through development to delivery ▶ an entity [See Data Lifecycle.]



- The life cycle of a thing is a process that can be defined in terms of:
  - states
  - events that trigger state transitions
  - actions that are performed on a state transition
  - actions that are performable while in a state.