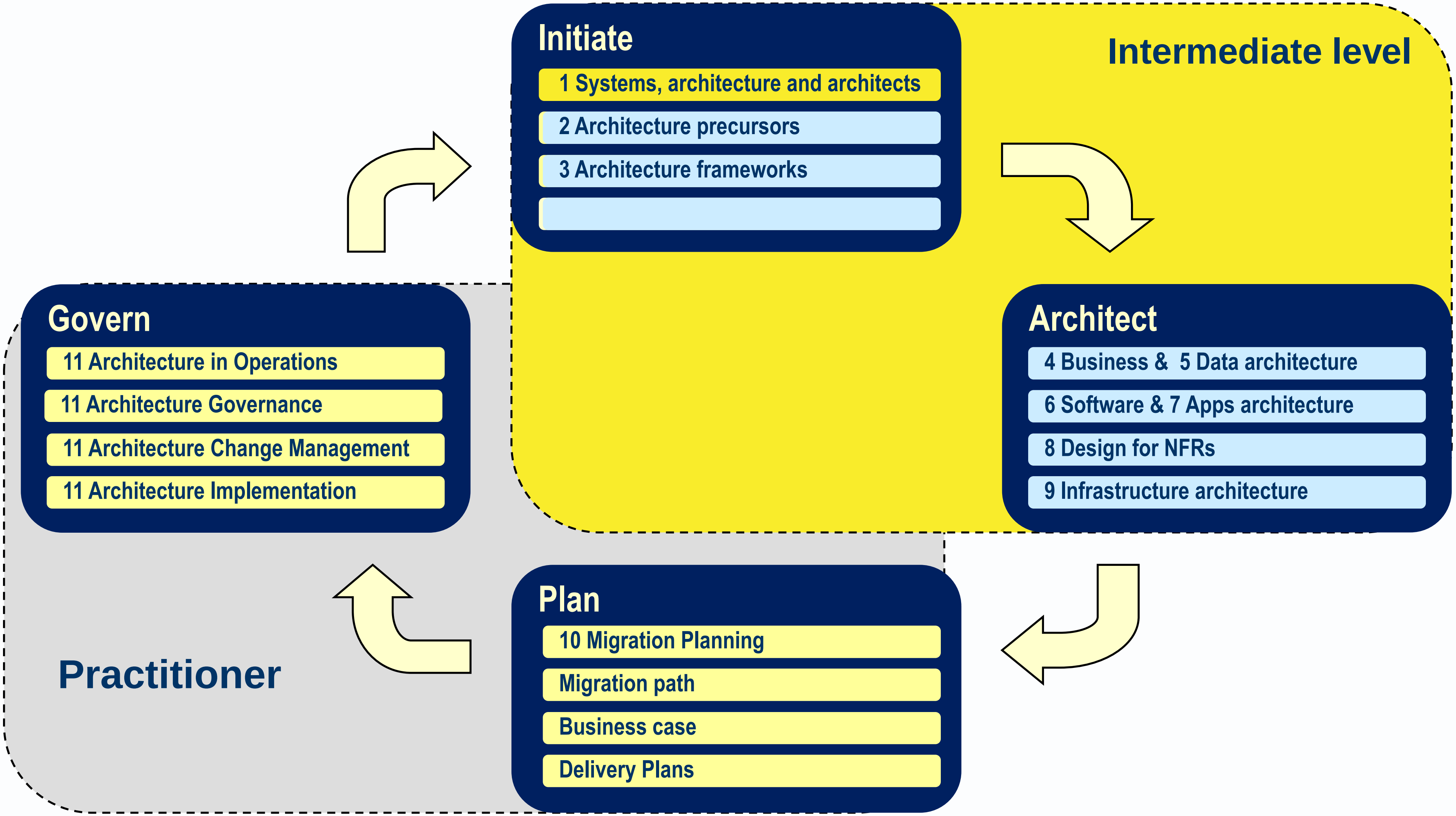
Avancier Reference Model

Architecture and Architects (ESA 1)

1. Architecture and architects



Intermediate level

Initiate

- 1 Systems, architecture and architects
- 2 Architecture precursors
- 3 Architecture frameworks

Architect

- 4 Business & 5 Data architecture
- 6 Software & 7 Apps architecture
- 8 Design for NFRs
- 9 Infrastructure architecture

Plan

- 10 Migration Planning
- Migration path
- Business case
- Delivery Plans

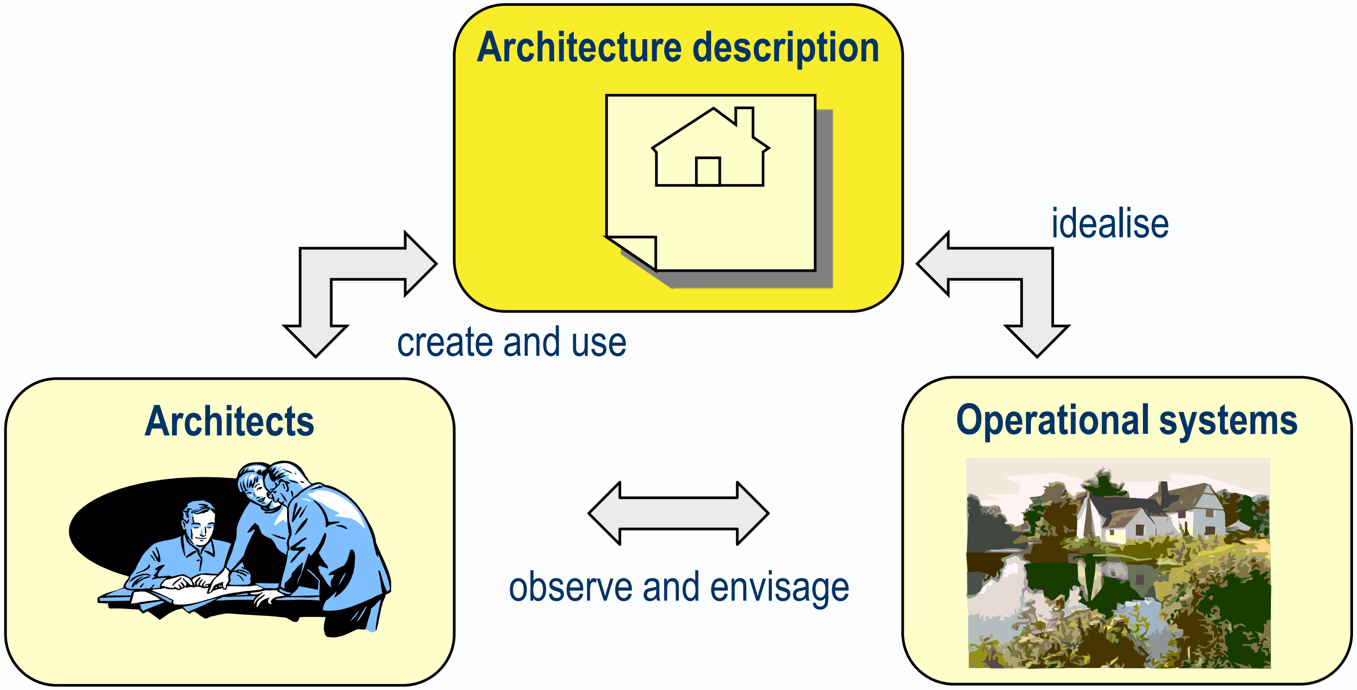
Govern

- 11 Architecture in Operations
- 11 Architecture Governance
- 11 Architecture Change Management
- 11 Architecture Implementation

Practitioner

1.1 Foundation (not to be examined)

- **Architecture** [A work product] that describes a system.
- An abstract description of the structural and behavioural elements of a system.
- It may map system elements to motivations, constraints
- It may map system elements to work packages needed implement the system.



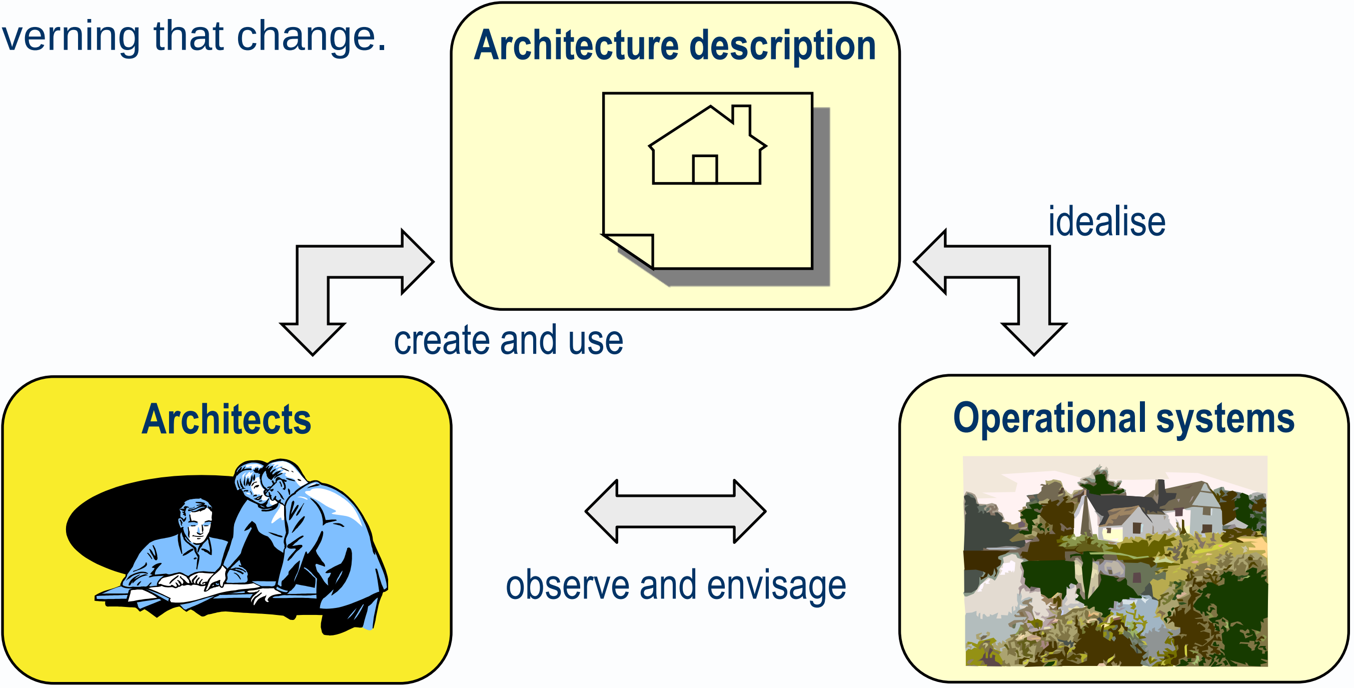
Enterprise and solution architecture

- **Enterprise architecture**
  - a cross-organisational and strategic view of business systems.
  - It includes business, data, application and technology component portfolios.
- **Solution architecture**
  - a solution to a problem or requirement.
  - It is usually developed within the context of a programme or project.
  - It may be done with or without reference to any higher or wider enterprise architecture.

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

Architecting

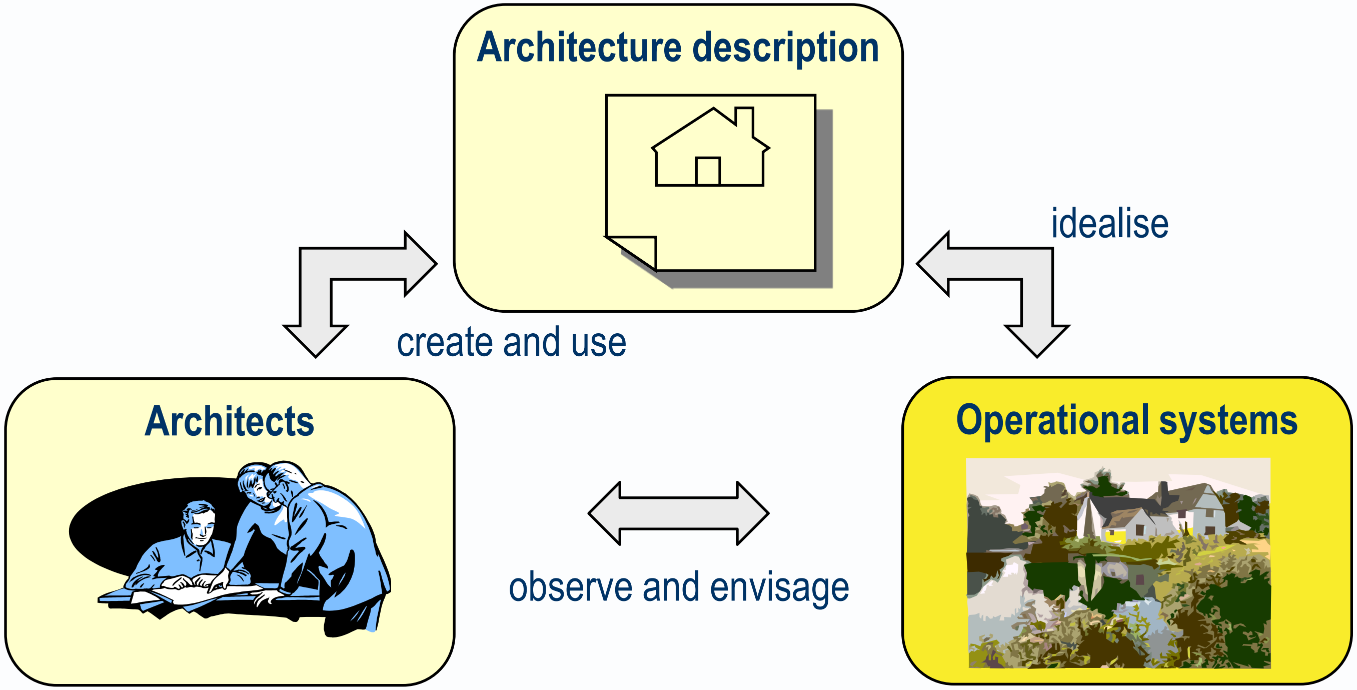
- [A work process] that involves
  - analysis of the context
  - analysing and choosing between options
  - defining an architecture (using principles and patterns)
  - ensuring agreement of what is described.
  - planning the move from the baseline state to a target state
  - governing that change.



“System”

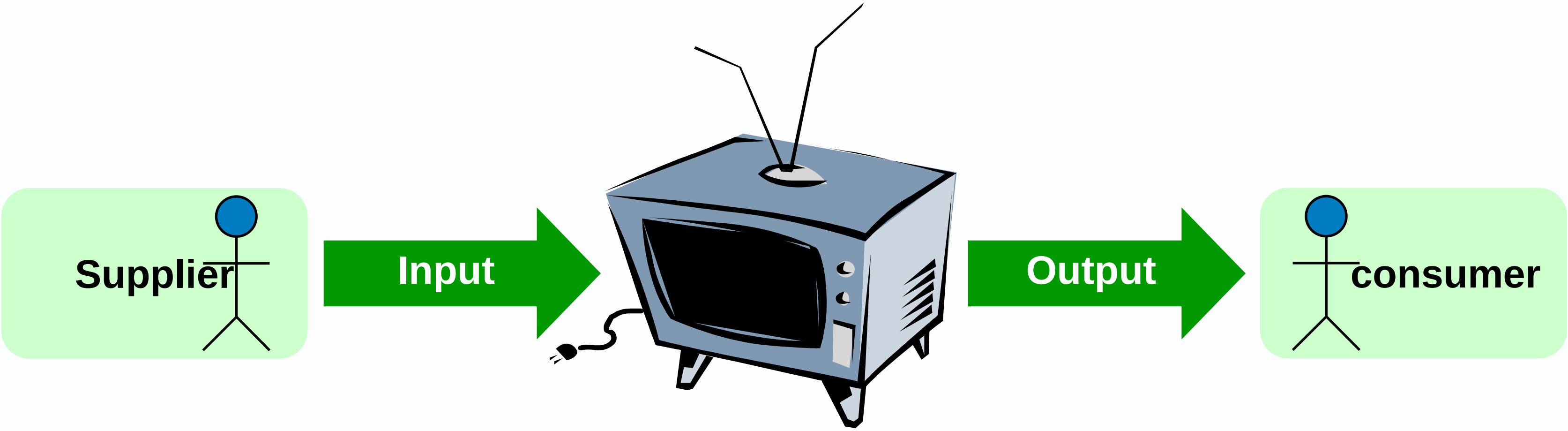
- “EA regards the enterprise as a system, or system of systems”
- “Architecture descriptions are formal descriptions of a system.”

TOGAF



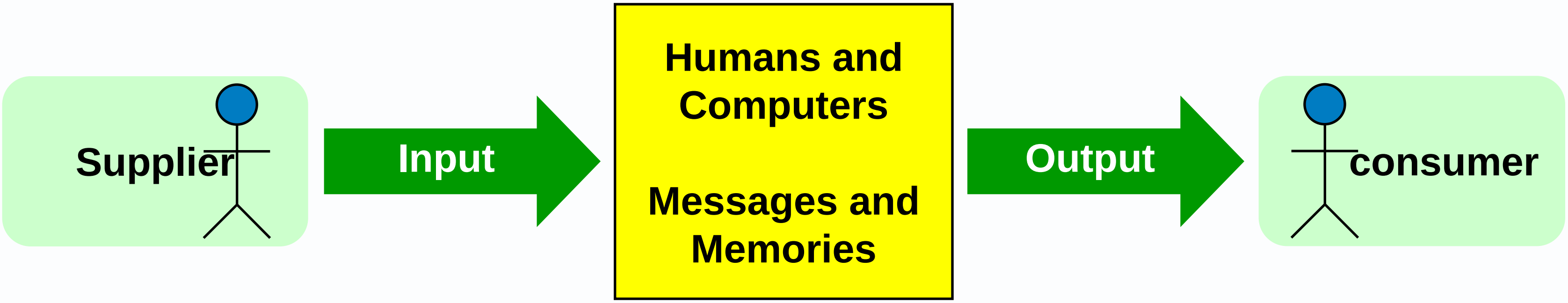
System (aka activity system)

- A system in which components interact in the performance of processes.
- A system may be encapsulated behind an interface.
- A system may be a subsystem or component of a larger system.
- The system boundary is chosen by the observer or designer who describes it.



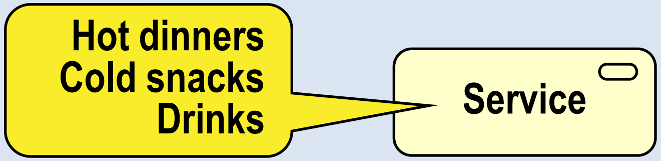
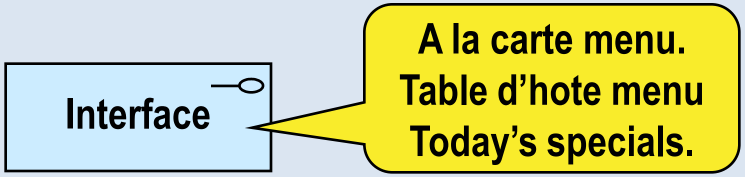
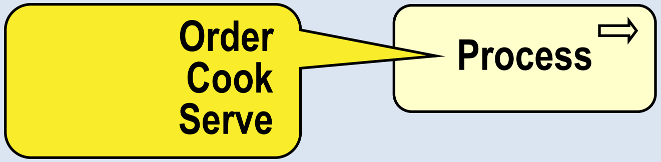
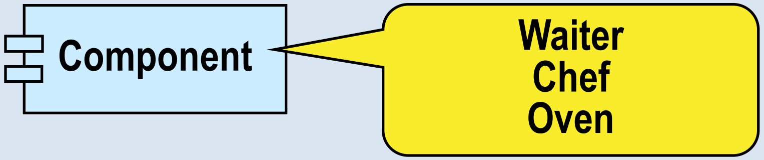
Business system

- [A system] in which processes are designed to achieve the aims of a business.
- The system components usually include human and computer actors.
- Components interact by creating and interpreting messages.
- Components respond to messages in the light of memories retained.
- Messages and memories contain data structures that capture information.



System element

- A discrete structural or behavioral element of a system.

	Behavioural view	Structural view
External view	**Service contract:** an end-to-end process defined as external entities see it.  	**Interface definition**: a declaration of available and accessible behaviours  Interface A la carte menu. Table d'hote menu Today's specials.
Internal view	 **Process:** a sequence of activities performed by one or more components.	 **Component:** a subsystem capable of performing one or more behaviours

- **Structural view**
  - [A view] that shows what a system is made of.
  - It connects system elements (components and objects) as nodes in a structure.
- **Behavioural view**
  - [A view] that shows what system does over time.
  - It shows how system elements (processes and services) progress from start to end.

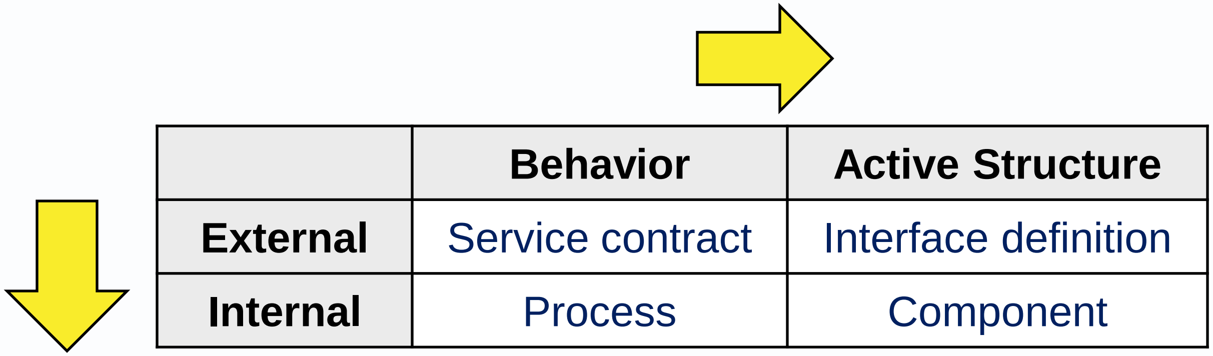
	Behavior	**Active Structure**
External	Service contract	Interface definition
Internal	Process	Component

- **External view**
  - [A view] that shows what an external entity sees of a system or process.
  - E.g. an interface definition or a service contract.
  - It may relate system elements to external entities
- **Internal view**
  - [A view] that shows the internal design of a system or process.
  - It shows the internal structures or behaviours of a system.
  - E.g. a network diagram or a process flow chart.

	Behavior	**Active Structure**
External	Service contract	Interface definition
Internal	Process	Component

System design principle

- A principle for how to approach the specification of a system.
- Architecture methods often feature the following principles.
  - External before internal
  - Behavior before structure
  - Business before technology
  - Logical before physical.



	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component

Structural elements (generic)

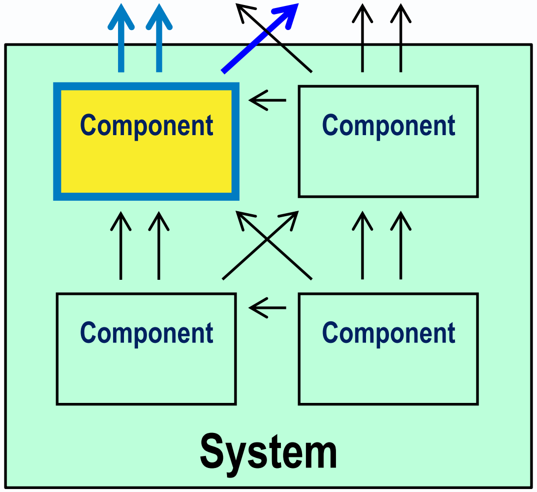
Structure

- Something a system is made of, an element that is addressable in space.
- It may be measured in terms of its size or space occupied.
- Active structures perform actions; passive structures are acted on.
- (Active structures are sometimes treated passive structures.)

	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component

Active structure elements

- **Component**
  - [An active structure] a subsystem capable of performing one or more behaviours.
  - Components interact to perform the processes and meet the aims of a larger system.
  - A component can be replaced by any other component with the same interface(s).



A large component may characterised as a «subsystem».UML

“A component represents a modular part of a system that encapsulates its contents.
A component is a self-contained unit that encapsulates [internal] state and behavior.“
UML

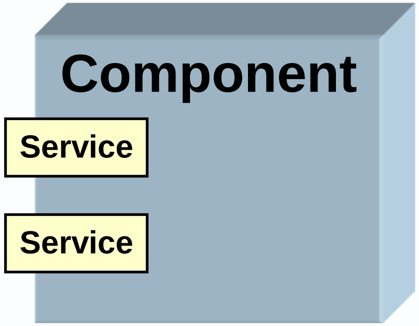
A building block is a package of functionality defined to meet the business needs across an organization.” TOGAF 9.1

	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component

Logical component

- [A component] that specifies what is required of one or more physical components.
- It is more abstract or supplier-independent than a physical component.

- It may specify services to be performed and data resources needed.
- E.g. a logical business, application or technology component.



	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component

Capability

- An ability to do or achieve something.
- E.g. to perform a process or role, realise function or achieve an aim.
- In practice, it is often 1 to 1 with a logical component.

	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component

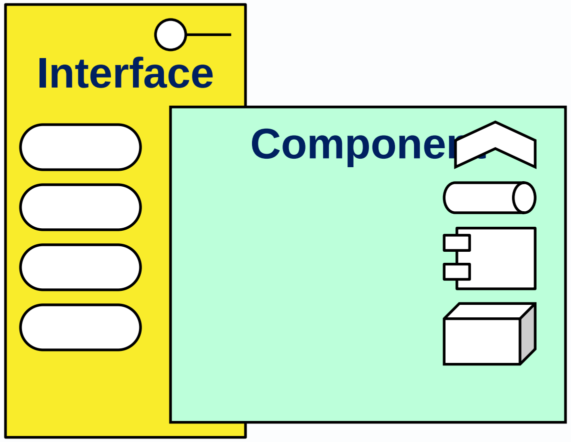
Physical component

- [A component] that is supplier-specific and/or can realise one or more logical components.
- E.g. a physical business, application or technology component.

	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component

Interface definition

- [An active structure definition] a declaration of available and accessible behaviours.
- **Logical interface elements:**
  - *Signatures* (name, inputs and outputs) for services.
  - *I/O flows* consumed and produced by services.
- **Physical interface elements:**
  - *Protocols* used to exchange data between components.
  - *Addresses* at which components can be found



“”A component has an *external or* “black-box” view by means of its externally visible services.”

“A component defines its behavior in terms of provided and required interfaces.” UML

“it is important that the interfaces to a building block are published and reasonably stable.” TOGAF 9.1

	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component

A logical interface definition

Avancier

FTP	An interface implemented by a platform component whose role is to copy files to and from computers. The services below are expressed as in the common FTP utility program on a UNIX computer.
Service name	**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

Passive structure elements

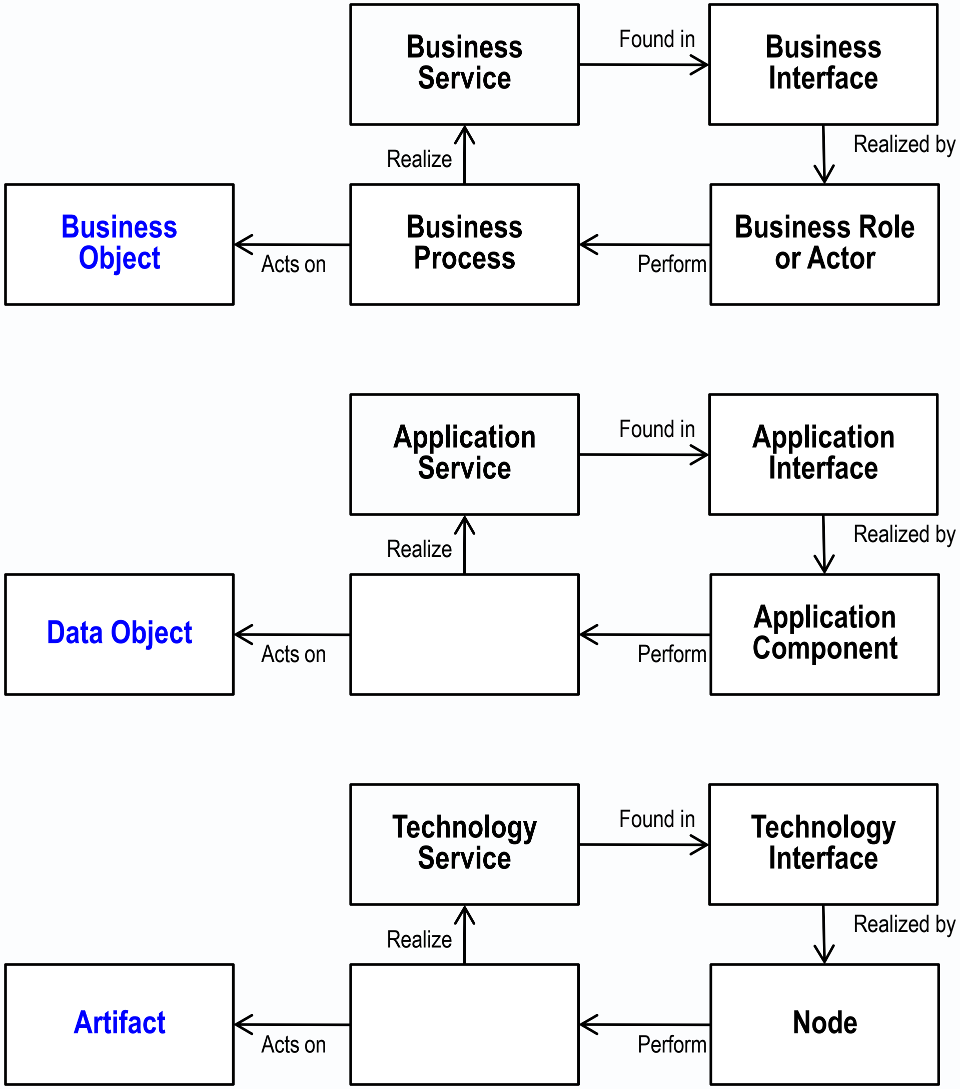
Object

- [A passive structure] a material or data structure that can be made, moved or modified.
- (NOT an object in the sense of object-oriented programming.)

Location

- [A passive structure] a place where active components or passive objects can be found and work can be done.

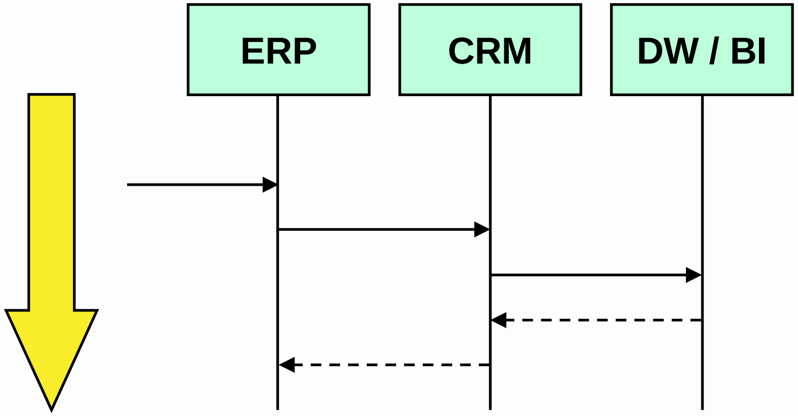
ArchiMate terms



Behaviour elements (generic): Behavior

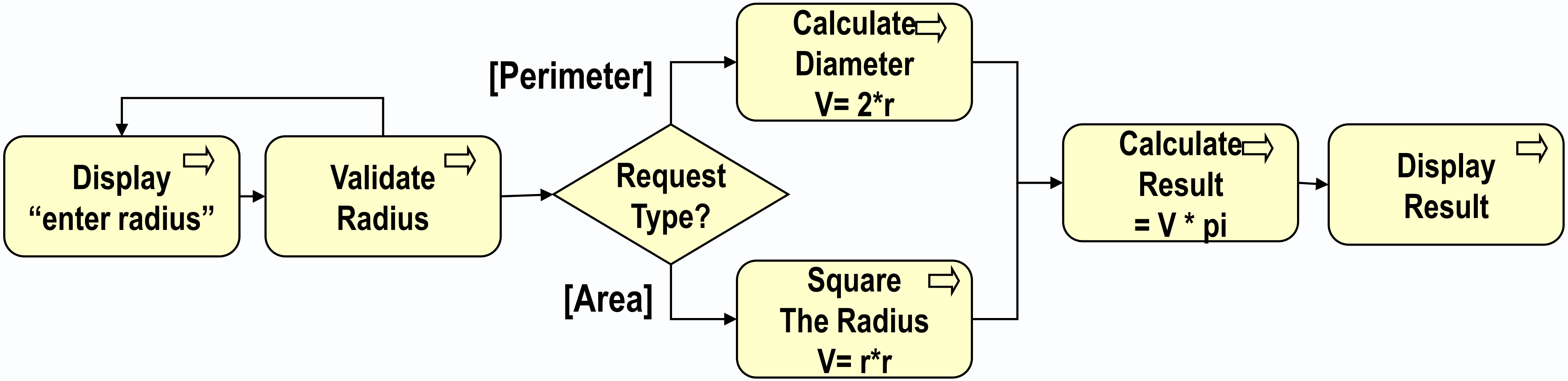
- Something a system does over time.
- It either changes the state of a system or reports its current state
- It is performed by one or more components.
- It may be measured in terms of its start and end times.

	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component



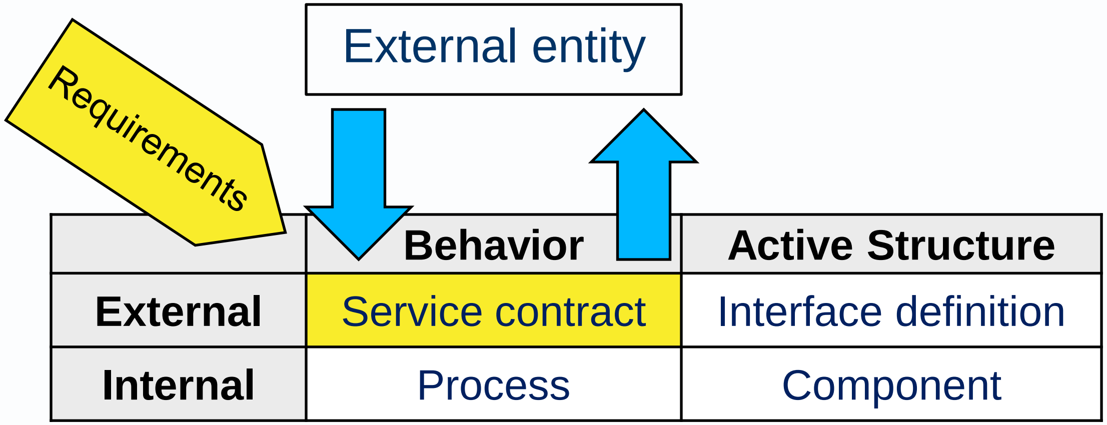
Process

- [A behavior] comprising one or more activities that run from a start to an end result of value. E.g. deliver package, check credit, provide weather data, and consolidate reports.
- It can be coarse-grained (build a house) or fine-grained (retrieve an address).
- It may be defined externally in a service contact.
- It may be defined internally as activities under logical control flow.
- It may orchestrate subordinate processes.



Service

- [A process] regarded as a service by an external entity.
- It supports or enables that entity by delivering one or more results.



Requirements

External entity

	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component

"The external [declarative] view of a unit of behaviour." ArchiMate
"a logical representation of a repeatable business activity... has a specified outcome". TOGAF

Service contract

- [A behavior definition] as external entities see it, in terms of:
  - Entry conditions
  - Exit conditions (aka results)
  - Qualities.

- It encapsulates all processes (human and/or computer) needed to complete the service.
- It may be documented along with other service contracts in a service portfolio, a Service-Level Agreement (SLA) or an interface definition.

"A component specifies a formal contract of the services that it provides to its clients and those that it requires from other components in the system." UML

"Ideally a building block is well specified."
"For each building block, build up a service description portfolio as a set of **non-conflicting services**." TOGAF 9.1

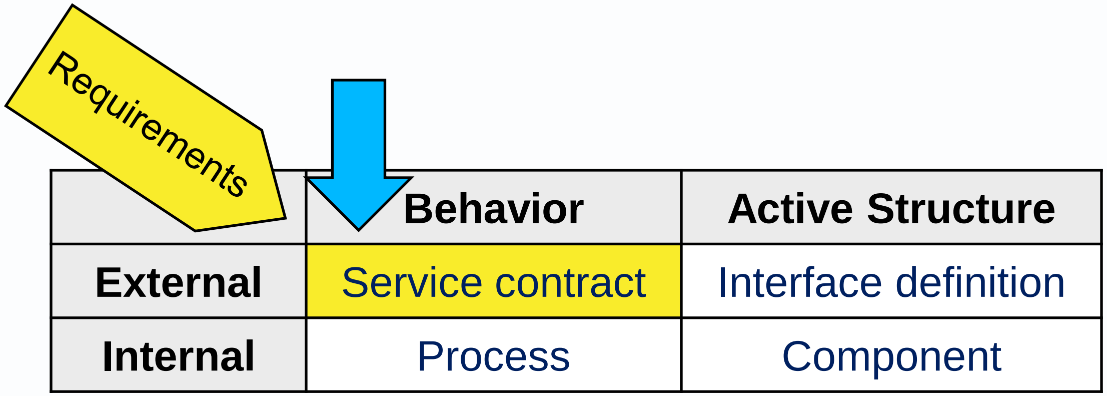
Service contract - Technology example

- A process defined as external entities see it, encapsulating all processes (human and/or computer) needed to complete the service

FTP service	Name	get
Entry conditions	Event	
	Input	Remote file name Local file name
	Preconditions	Remote computer can be reached. Remote file exists in the current remote directory.
Exit conditions or results	Output	Reply = OK or Fail (see post conditions)
	Post conditions	Remote file copied to (or on top of) local file current local directory.
Qualities	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

Entry conditions

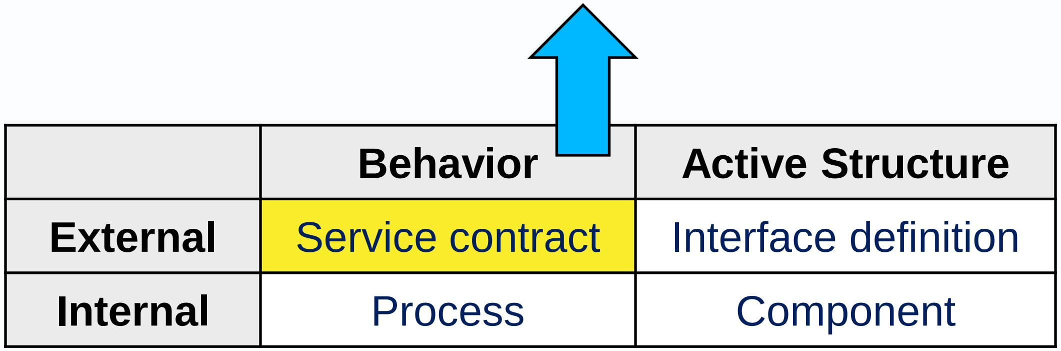
- Event
  - the trigger of a behaviour, which may be a flow, time event, or state change.
- Input flow
  - a material structure or message received by a process.
- Precondition
  - a prior state that must exist if the process is to complete successfully.



	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component

Exit conditions (aka results)

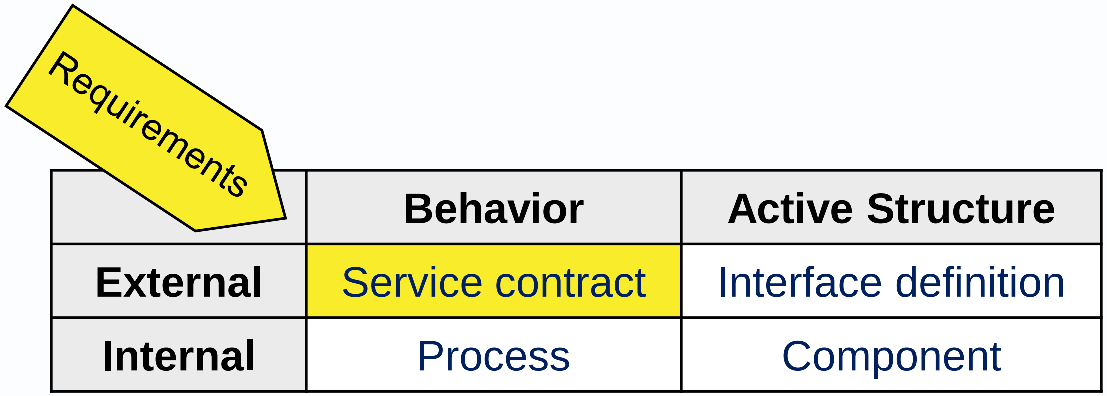
- Output flow
  - a material structure or message produced by a successful process.
- Post condition
  - a change of state resulting from a successful process; a change in the qualities of an object that is maintained, including any “added value”.
- Value
  - the worth of an output or state change to an owner or consumer.



	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component

Qualities

- Measurable attributes of a service.
- E.g. duration, cost, availability.



	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component

Remember: what service-orientation means here

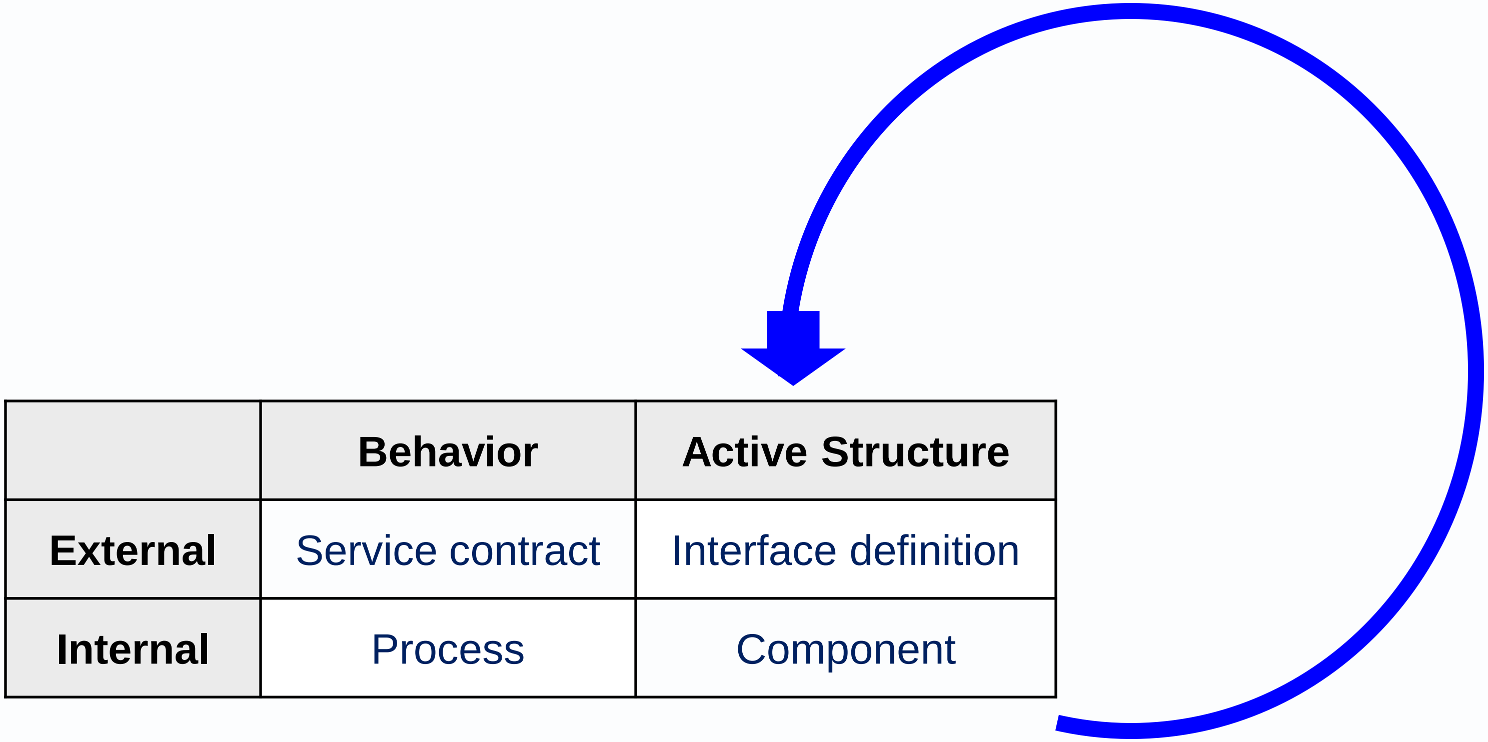
- A service is what is required – a discretely invokable behavior.
- A service can be detailed in a contract including these attributes:
  - Service name
  - Entry conditions
  - Exit conditions
  - Qualities

	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component

Component are encapsulated
by interfaces containing
the services they offer

1.2 Modularity

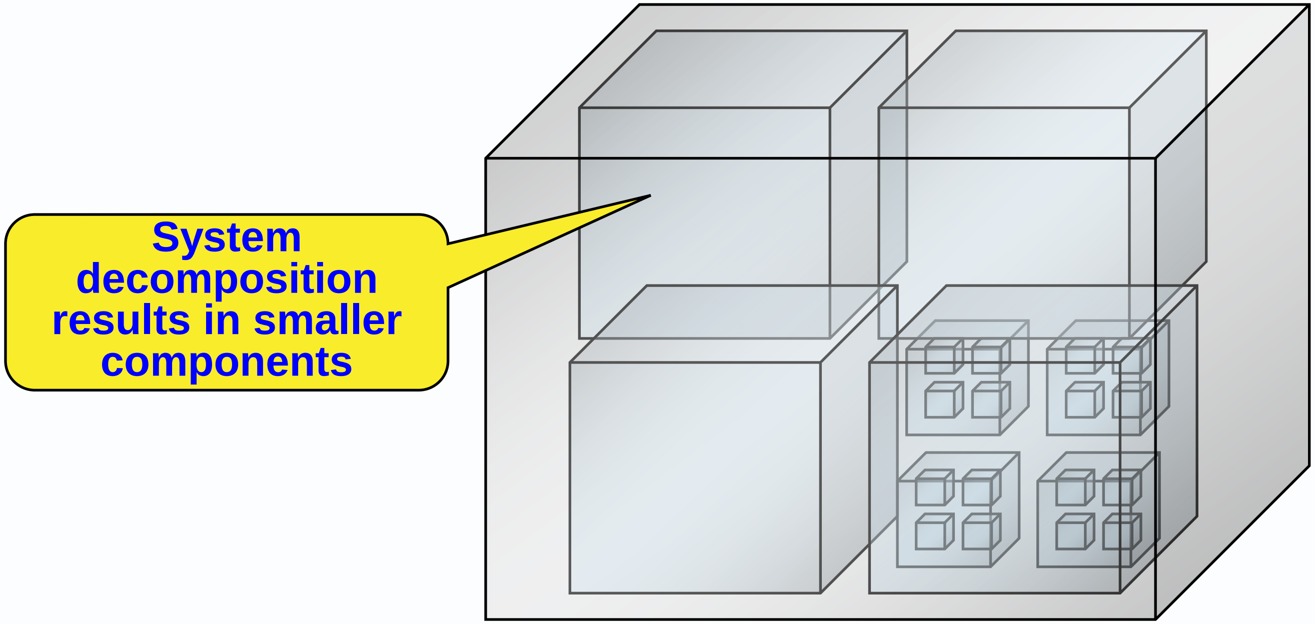
- Components are systems in their own right



	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component

System decomposition

- [A technique] that divides a larger system into smaller components. Each component can be seen as a system in its own right and further subdivided.

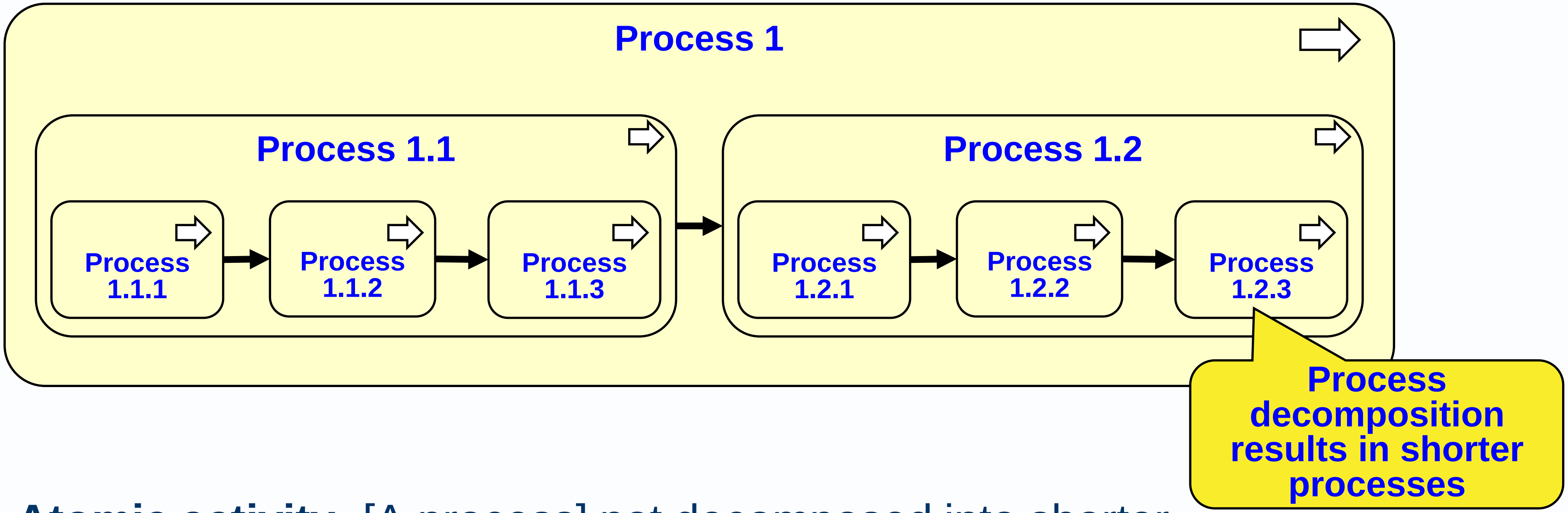


System decomposition results in smaller components

- **Atomic component:** [A component] not decomposed into smaller components (in a description of a system).

Behavioral decomposition

- [A technique] that divides a longer process into shorter activities
- Each activity can be seen as process in its own right and further subdivided.



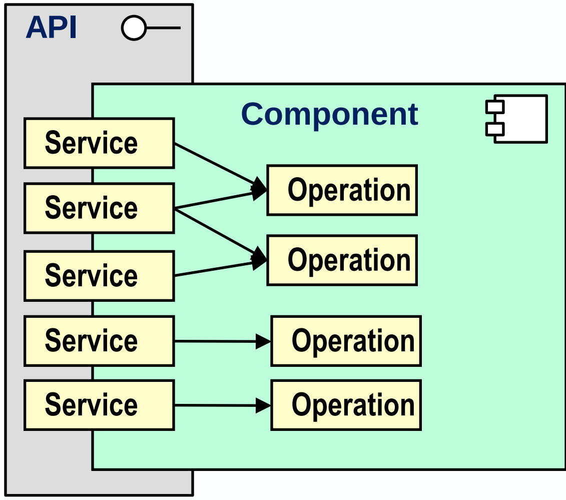
Process decomposition results in shorter processes

- **Atomic activity:** [A process] not decomposed into shorter processes or activities (in a description of a system).

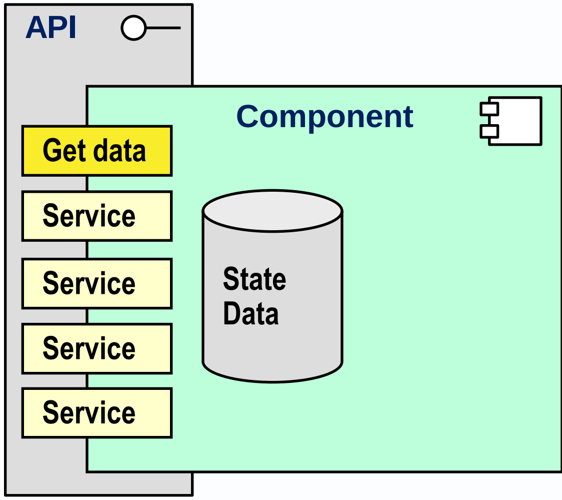
Encapsulation

- [A technique] that defines a thing by an interface it offers.

- It hides inner workings or processes from external entities.

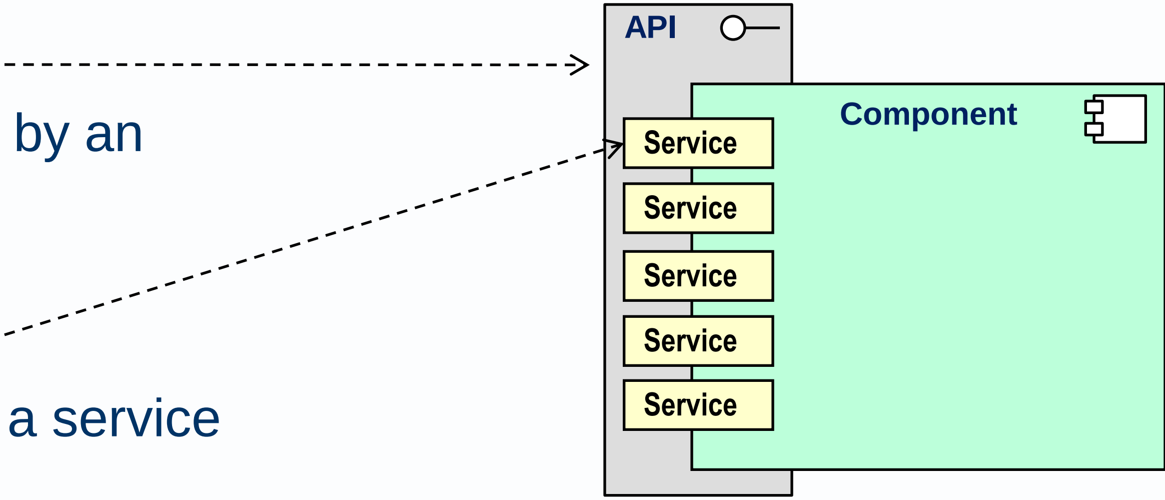


- It hides internal resources (notably data structures) from external entities.



Encapsulation varieties

- **Structural encapsulation**
  - Encapsulating a component by an interface it offers.
- **Behavioral encapsulation**
  - Encapsulating a process by a service contract.



	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component

Component-based design

- [A technique] for defining a system in terms of interacting components.
- The requirement for a component is defined as a provided interface definition.
- What a component needs is defined as required interface definitions or server component names.

"A component defines its behavior in terms of provided and required interfaces." UML

"it is important that the interfaces to a building block are published and reasonably stable." TOGAF 9.1

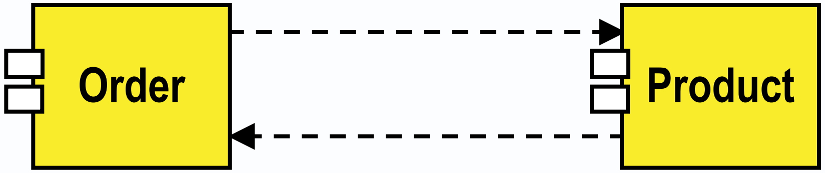
Modularity

- Central issues in architectural design include
  - The optimal **granularity** of components
  - The optimal number of **levels** of component decomposition
  - Avoiding unnecessary **dependencies** between components
  - Avoiding unnecessary **duplication** between components
  - **Distributing** components and **integrating** components

Dependency

- [A coupling] between two components that means a change to the depended-on component requires impact analysis of the dependent component.

Cyclical "dependency" arrows



- Note: it is said that strong cohesion within a component is good, and low coupling between components is good.
- But coupling is not a single or simple concept; there are *many* ways to be coupled or not (see section 6).

Granularity

- [A measure] of the size of a structure or the duration of a behavior.

- Note there is a granularity rule of thumb:
  - finer-grained components tend to be more tightly coupled;
  - coarser-grained components tend to be more loosely coupled, so they can be managed and run independently.

- This reference model does not fix the granularity of system elements.
- One interface definition may encapsulate several components and one service contract may encapsulate several (related) processes.
- So, it possible to re-engineer (refactor) internal components and processes without changing the external services and interfaces.

1.3 Architecture domains/views

- A division or view of an architecture that addresses a broad set of concerns.
- The four architecture domains below were established in the PRISM report (1986).
- They are now the basis of countless EA frameworks, including TOGAF.
- Other domains (such as motivation, security, governance) may span the primary domains.

The architects' working space				
	Business Architecture	Data Architecture	Applications Architecture	Technology Architecture

Business architecture

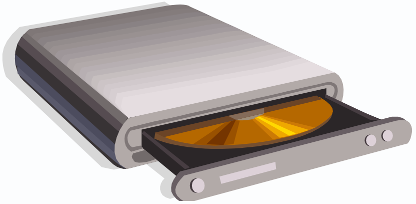
- [A view] that identifies and relates business elements:
- What the business delivers: business products and services
- How the business does it: business processes (scenarios, value streams)
- What the business needs to do it: components and objects needed to perform processes.

- 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

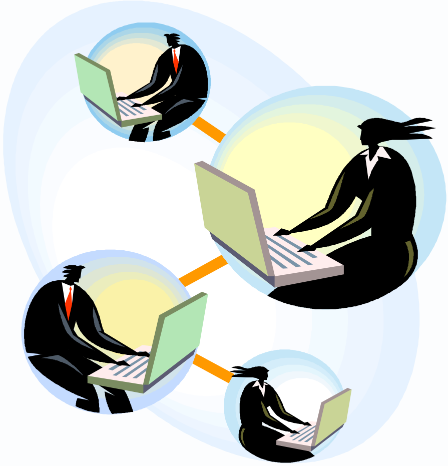
- [A view] that identifies and relates data elements:
- Data stores and flows used by business activities
- Data structures contained in the data stores and data flows
- Data qualities: data types, confidentiality, integrity and availability.

- Data elements may be mapped to business activities and to applications.
- EA is concerned with standardisation and integration of data between systems.

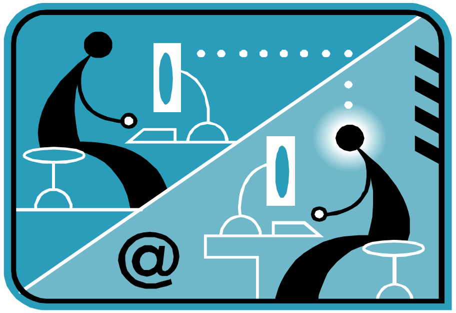


Applications architecture

- [A view] that identifies and relates application elements:
- Business applications needed to support business roles
- Data flows (messages) consumed and produced by applications
- Application use cases performed in the course of business activities.



- Application elements may be mapped to business activities and to platform technologies.
- EA is concerned with standardisation, integration and life cycles of business applications.

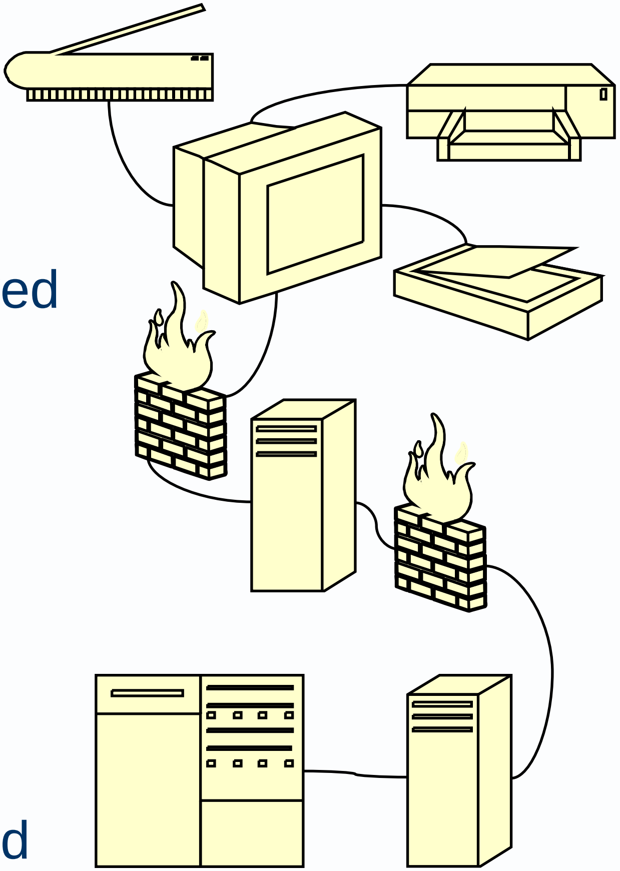


Technology architecture (aka infrastructure architecture)

- [A view] that identifies and relates technology components.
- Platform technologies and the services they offer to business applications/
- Client and server nodes that applications are deployed on.
- Protocols and networks by which nodes are connected.

- Technology elements may be mapped to business applications, data stores and data flows.

- EA is concerned with standardisation, integration and life cycles of platform technologies.

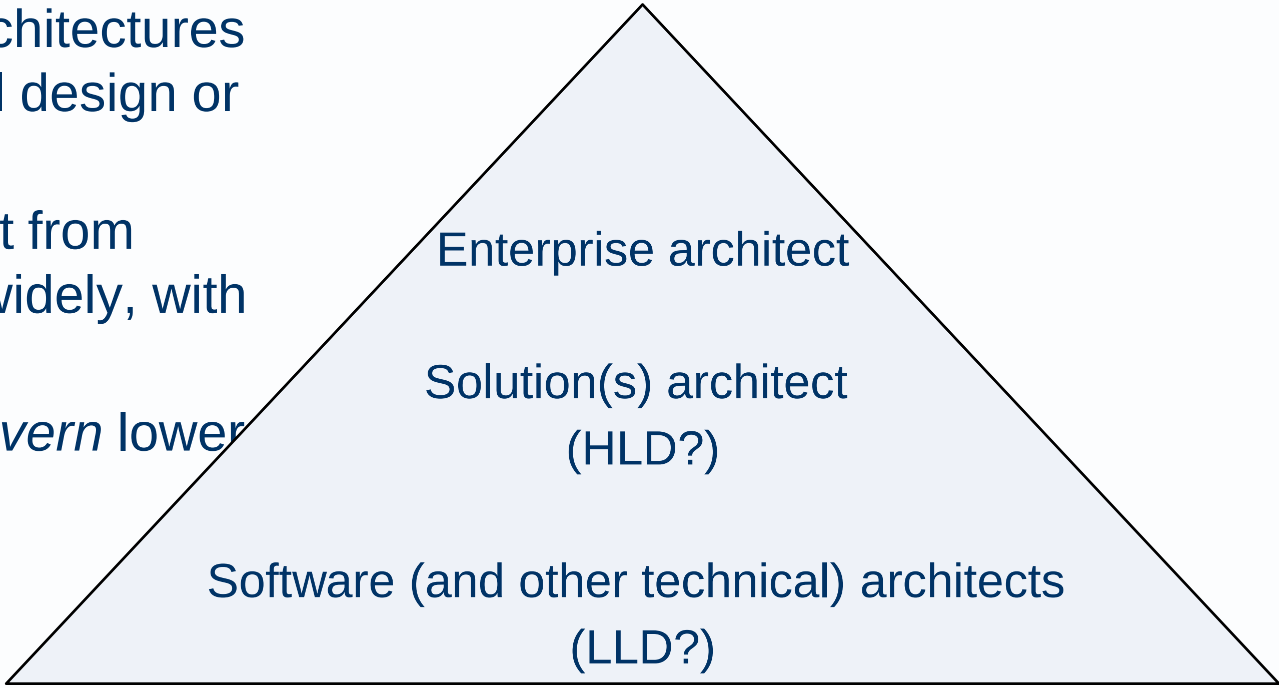


Core concepts in the TOGAF standard v9.2

Domain	Required behaviors	Logical structures	Physical structures
Business	Business service Process	Function Role	Organisation Unit Actor
Data		Data entity Logical data component	Physical data component
Applications	Application service	Logical application component	Physical technology component
Technology	Technology service	Logical technology component	Physical technology component

1.4 Architect roles

- **Architect:** [A work role] to design, plan and oversee the building of solutions and/or systems.
- It involves responding to requests for work, gathering contextual information, producing architectures and governing lower-level design or construction.
- Higher level roles abstract from detail and operate more widely, with a broader scope.
- Higher level roles may *govern* lower roles to some extent.



Enterprise architect

Solution(s) architect (HLD?)

Software (and other technical) architects (LLD?)

The architecture work space

- The table below is a simple way to view the architect roles implied in this reference model.
- Different organisations divide the work differently, and define other specialist architect roles.
- Any one architect may work at more than level and in more than one domain.

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

Enterprise architect

- [An architect] who takes a cross-organisational and strategic view of business systems, looking to.
  - optimise business systems by digitisation, standardisation and integration
  - exploit business data captured by business processes
  - identify potential innovations in business processes
  - maintain enterprise-wide business, data, application and technology component portfolios
  - maintain cross-organisational and/or strategic road maps
  - shape, steer and govern the work of solution architects.

Solution architect

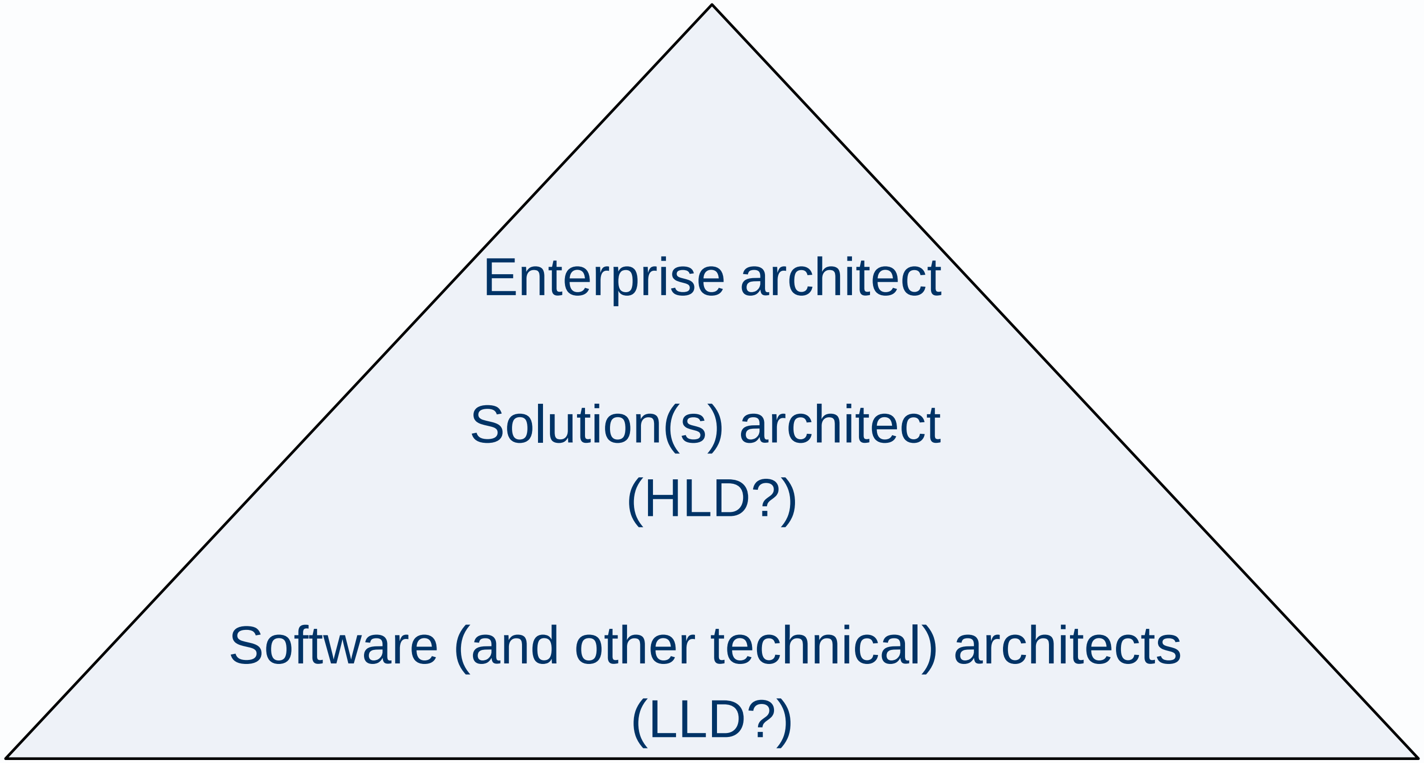
- [An architect] who focuses on individual solutions and systems.
  - addresses problems and requirements, related to specific processes and applications.
  - aims to ensure the quality of solution delivery, in compliance with overarching goals, principles and standards where possible.
  - describes solutions and govern their delivery, usually at a project level
  - understands all domain/views well enough to work with all analysts and designers
  - details a system architecture sufficiently for detailed design and building to proceed
  - focuses on critical success factors, especially non-functional qualities.
  - shapes, steers and governs the work of detail designers and implementers.

Software architect

- [An architect] who addresses the fine-grained modularisation and integration of software components inside business applications.
- He/she may follow principles and patterns for modularisation and integration.
-
- Aside: The principles and patterns of software architecture useful at higher levels.
- Architects should be familiar with (e.g.) encapsulation, cohesion and decoupling).
- And be able to apply them appropriately to their context.

1.5 Architect goals and skills

- This section is deliberately limited to short lists, sufficient for examination purposes.



Enterprise architect

Solution(s) architect (HLD?)

Software (and other technical) architects (LLD?)

Enterprise architect goal

- Alignment of IS/IT to business strategies and goals
- Business agility and technical agility.
- Standardisation: of processes, data, applications and technologies.
- Integration: interoperation of processes, data, applications and technologies.
- Enablement of strategically beneficial change through long-term planning.
- Portability: supplier and technology independence.
- Simpler systems and systems management.
- Improved IS/IT procurement.
- Improved IS/IT cost-effectiveness.

Solution architect goal

- Support the goals (above) of enterprise architects
- Alignment of IS/IT to business processes and roles
- Quality of IS/IT project deliverables.
- Cost of IS/IT project deliverables (though a manager is usually *accountable*)
- Timeliness of IS/IT project deliverables (though a manager is usually *accountable*)
- Solution-level risk identification and mitigation.
- Application integration and data integrity.
- Conformance of solutions to non-functional and audit requirements.
- Conformance of solutions to principles, standards, legislation.
- Effective cooperation between managers and technicians.
- Governance of detailed design to architecture principles and standards.

Architect knowledge or skill

- Holistic understanding of business and technical goals.
- Holistic understanding of business and technical environment
- Broad technical knowledge – including current trends.
- Broad methodology knowledge
- Analysis of requirements and problems
- Innovation.
- Leadership.
- Communication, political and soft skills (e.g. stakeholder management)
- Awareness of project management and commercial risks and issues.

FOOTNOTES

Avancier

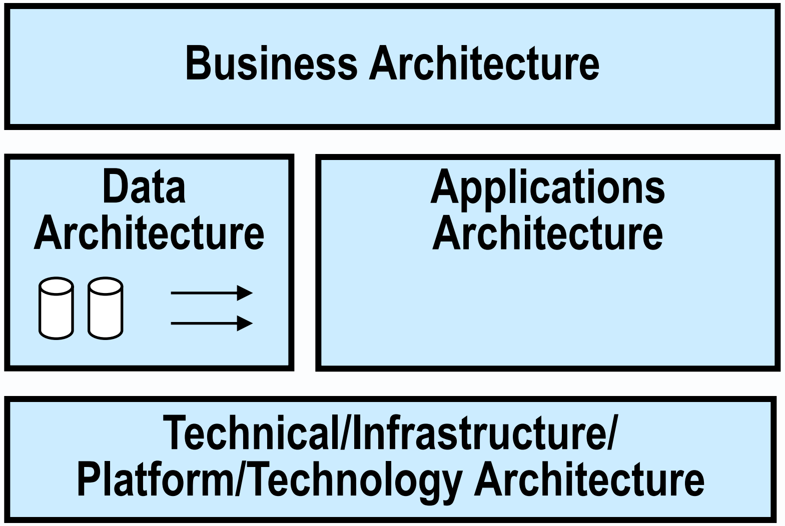
Architecture as higher level design

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

Other architecture domains

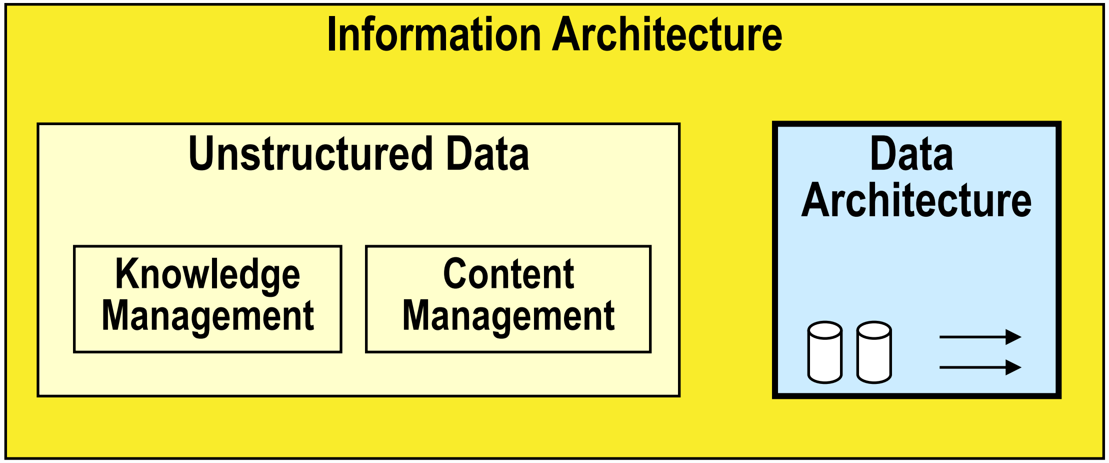
- In addition to the four primary architecture domains,


Business Architecture
Data Architecture
Applications Architecture
Technical/Infrastructure/ Platform/Technology Architecture

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

Information architecture

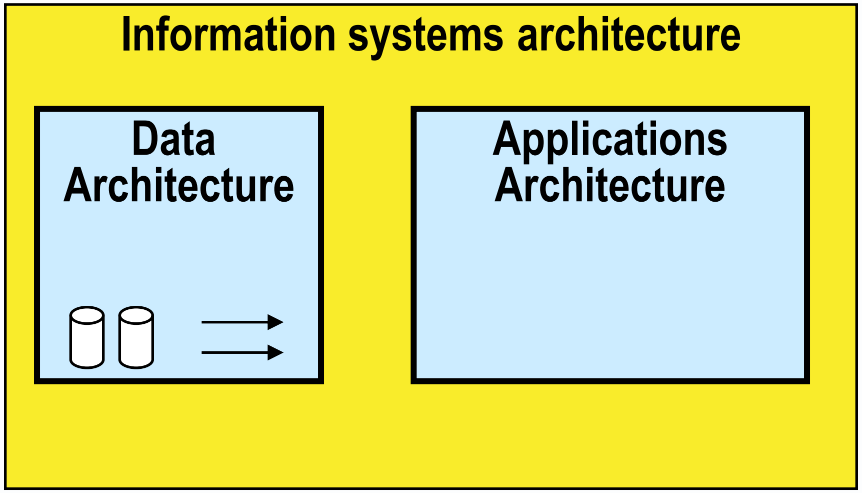
- a term for the broad domain that includes structured data architecture, content management and knowledge management.



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

Information systems architecture

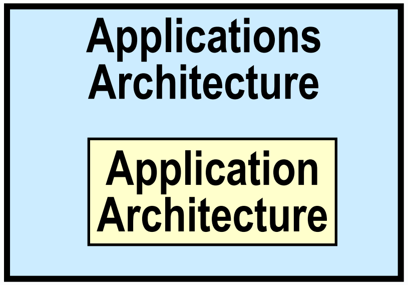
- a term often used to mean the combination of applications architecture and data architecture.



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

Software (aka application) architecture

- the internal structure or modularisation of a software application.

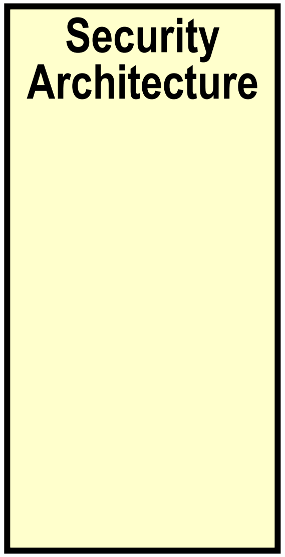


- This is software architecture at the lowest level of granularity (as discussed for example in “Patterns of enterprise application architecture” by Martin Fowler).
-
- It is usually below the level of modularity that enterprise and solution architects define. However, there is no rigid dividing line

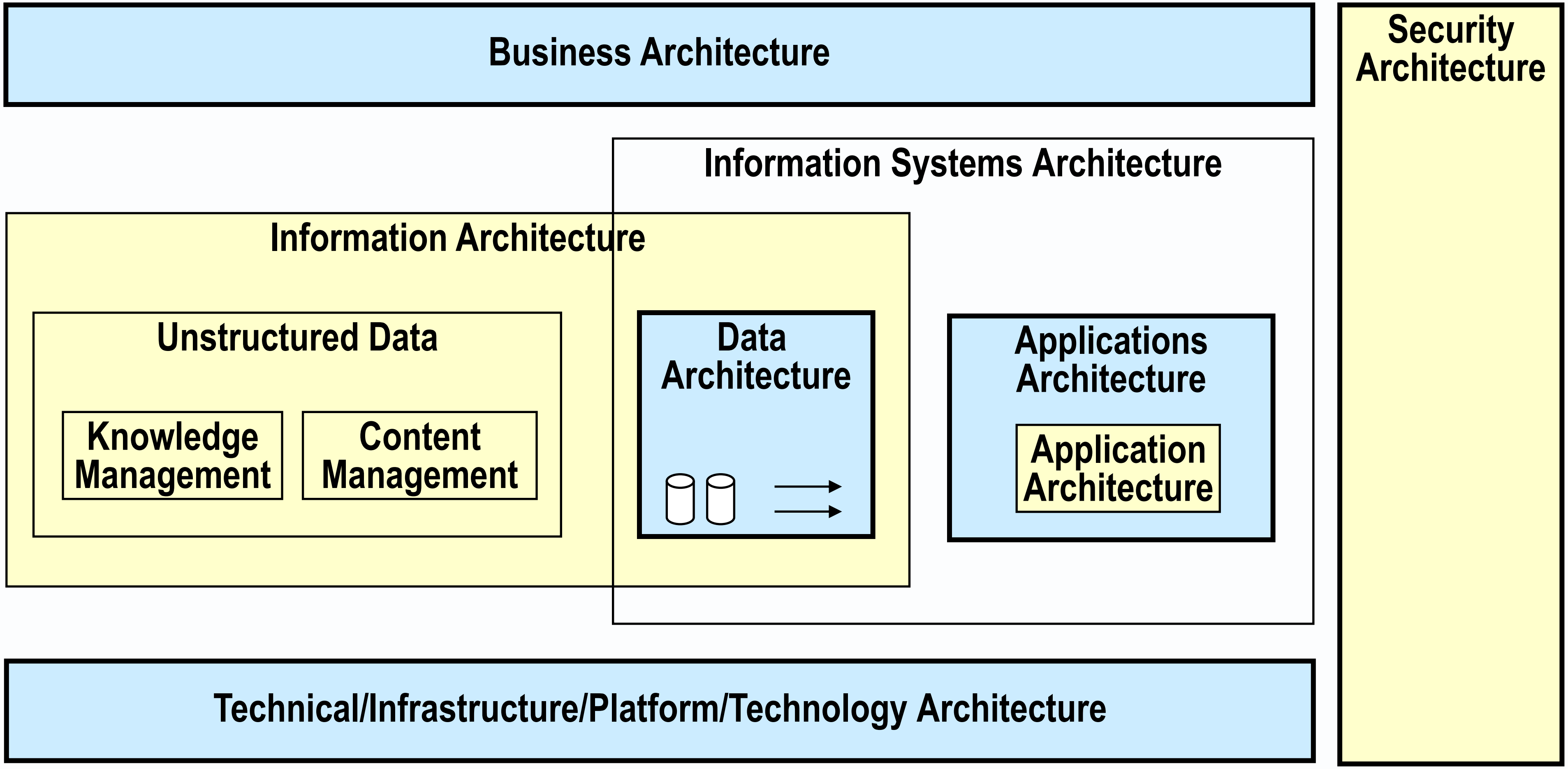
Security architecture

- 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



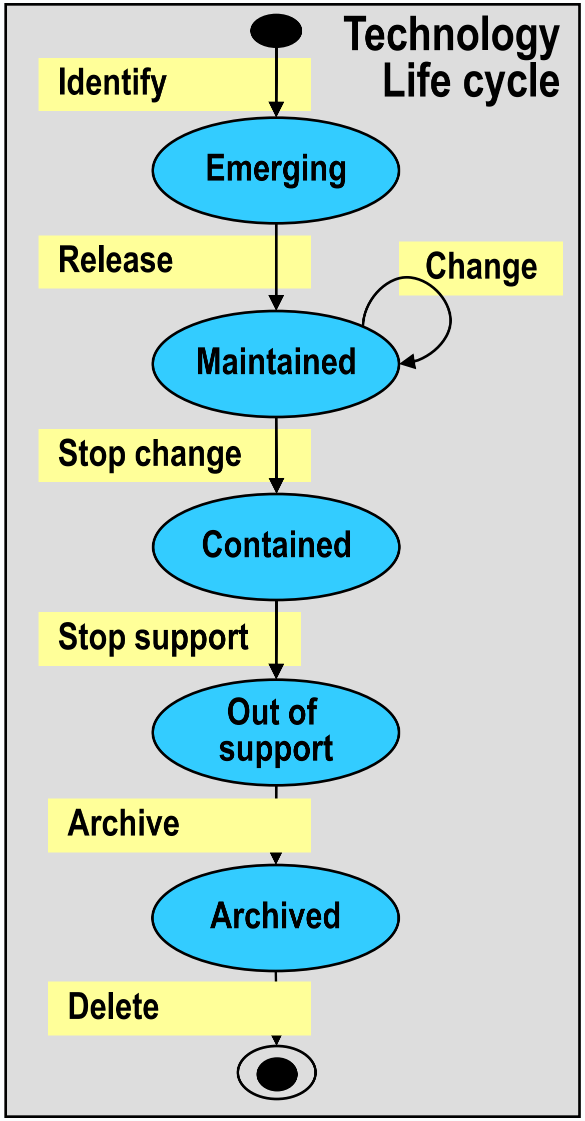
Put that all together



Foundation concepts

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.


Technology Life cycle
Identify
Emerging
Release
Change
Maintained
Stop change
Contained
Stop support
Out of support
Archive
Archived
Delete