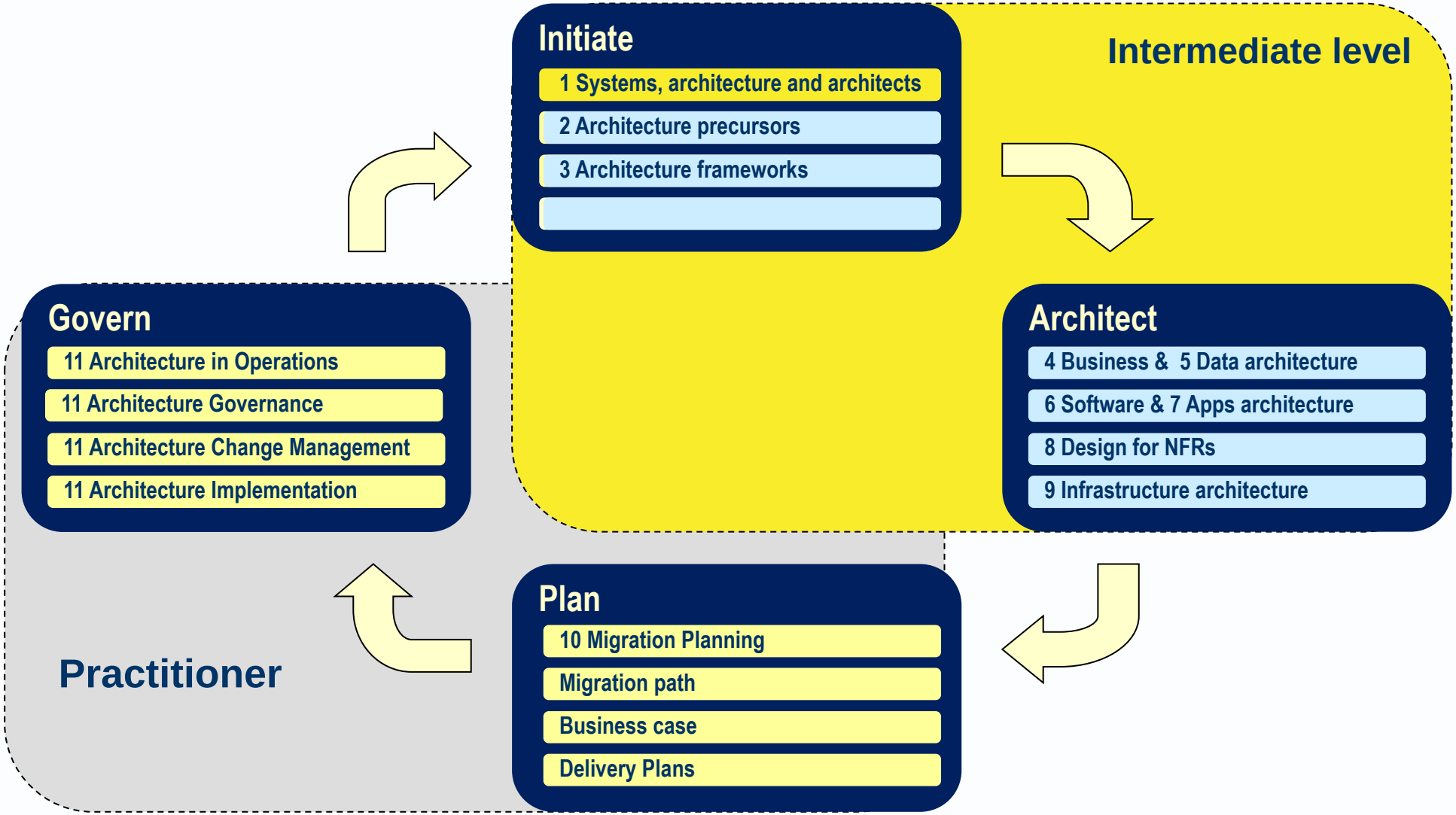
Avancier Reference Model

Architecture and Architects (ESA 1)

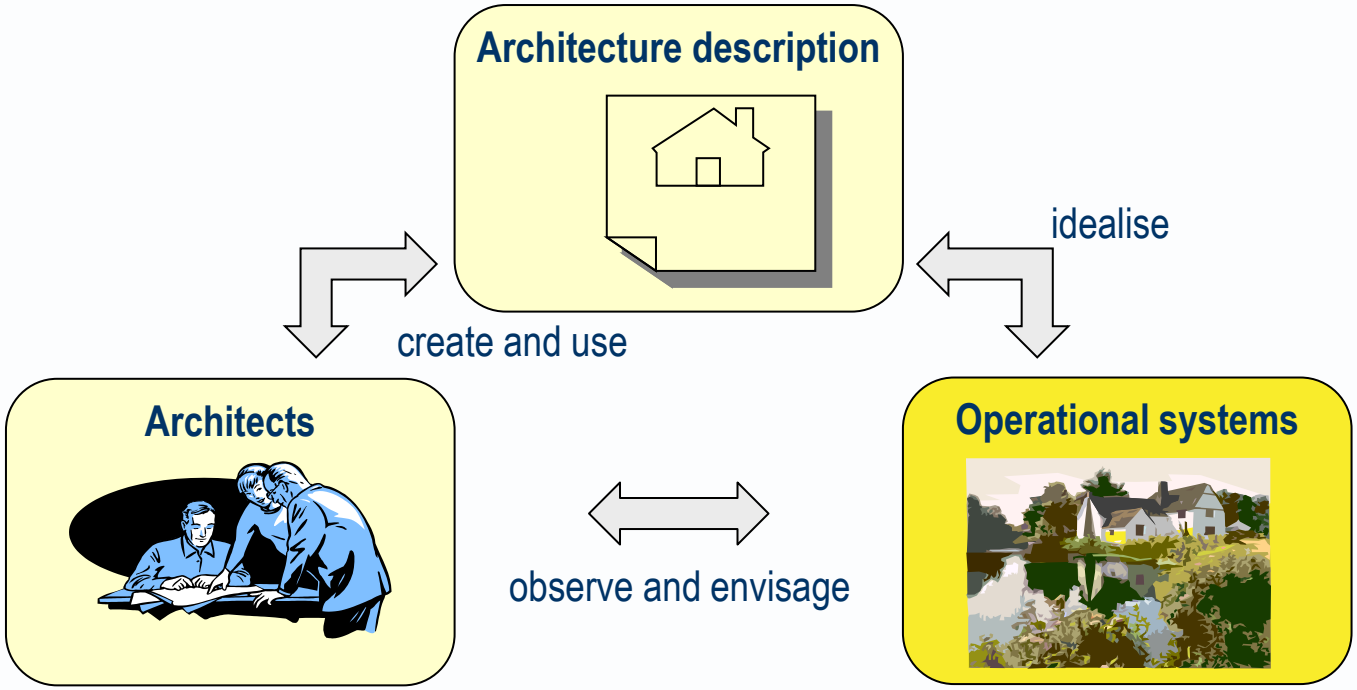
1. Architecture and architects



EA standards

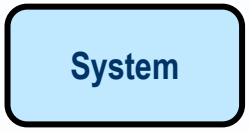
- "EA regards the enterprise as a system, or system of systems"
- "Architecture descriptions are formal descriptions of a system."

TOGAF

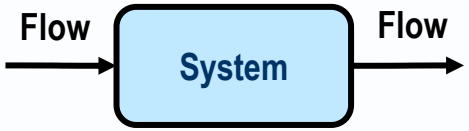


General activity system concepts

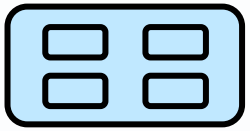
- System-environment boundary.



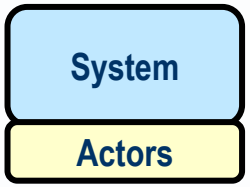
- Input and output flows of information and/or materials.



- Hierarchical composition/decomposition of systems, and resources used.



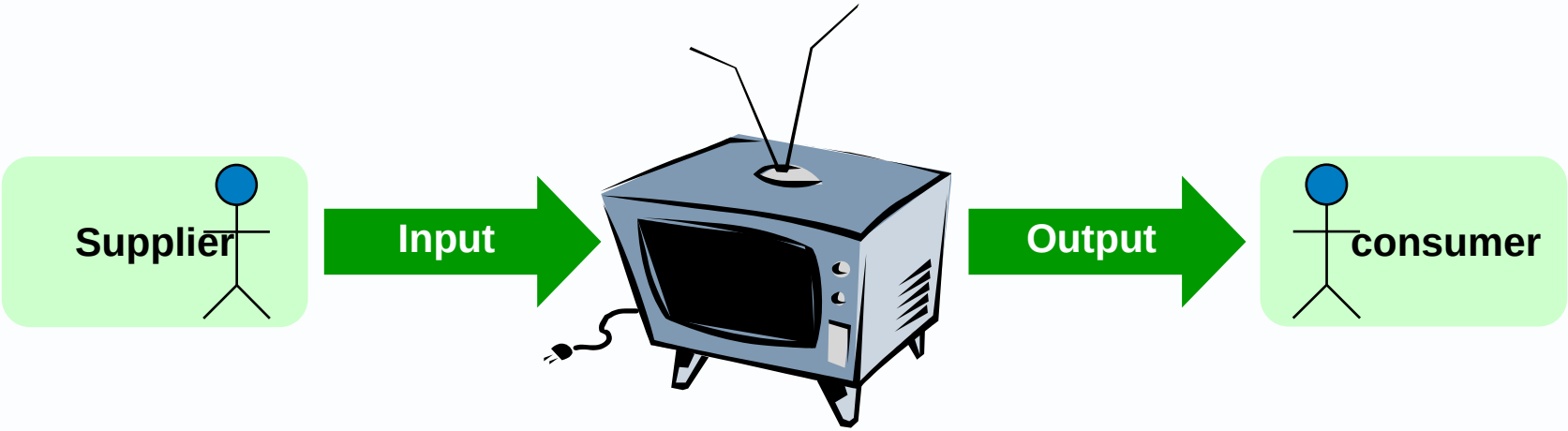
- Mappings of systems and subsystems to the actors (organizations or roles) that act in them.



- Processes

1.1 Encapsulation of systems

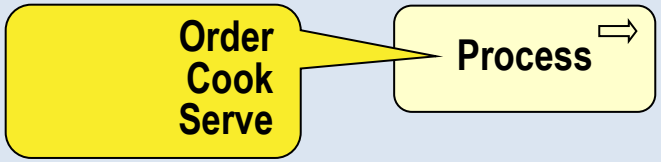
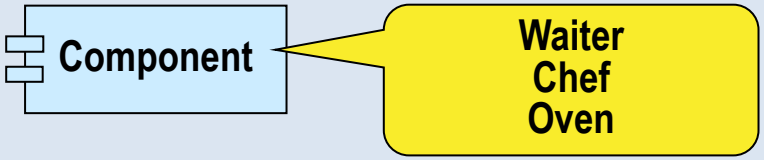
- A system transforms inputs into outputs in an orderly way.
- Its boundary is defined by the observer or designer who describes it.



- Business system is a system in which human and computer actors play roles in particular processes that create or use business data.

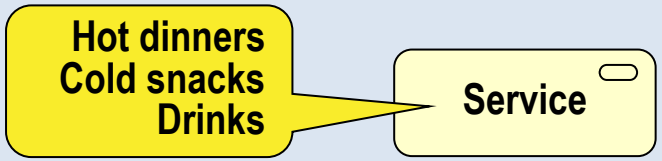
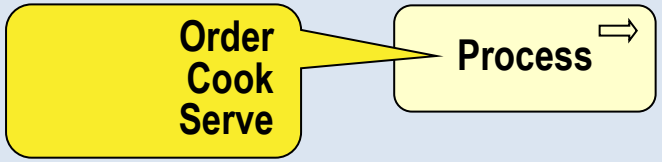
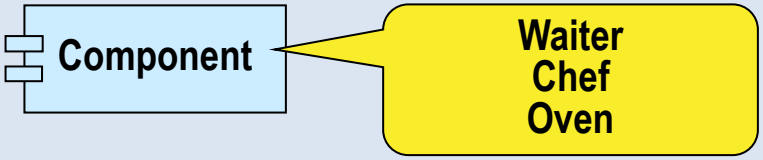
System elements

- A system contains actors or components (structures) that interact to perform particular processes (behaviors).

	Behavioural view	Structural view
Internal view	 **Process:** a sequence of activities performed by one or more components.	 **Component:** a subsystem capable of performing one or more behaviours

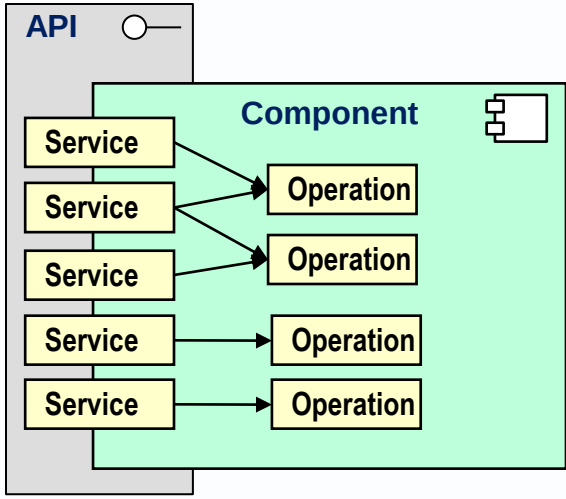
Encapsulation

- [A technique] that defines a system or component by its input/output interface.
- The interface hides inner workings or processes from external entities.
- It hides internal resources from external entities.

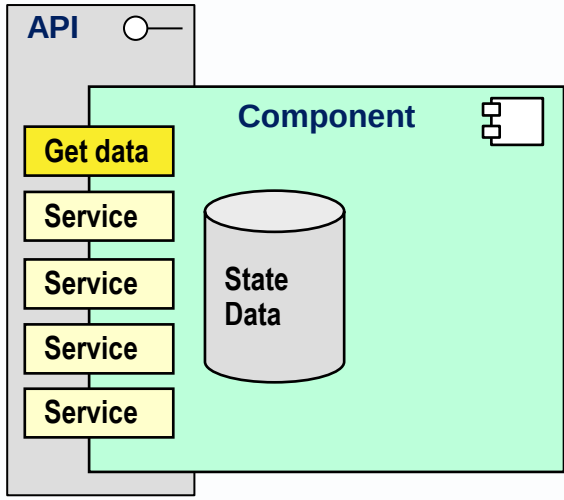
	Behavioural view	Structural view
External view	**Service contract:** an end-to-end process defined as external entities see it.	**Interface definition**: a declaration of available and accessible behaviours
		
		
Internal view	**Process:** a sequence of activities performed by one or more components.	**Component:** a subsystem capable of performing one or more behaviours

Encapsulation

- hides inner workings or processes from external entities.

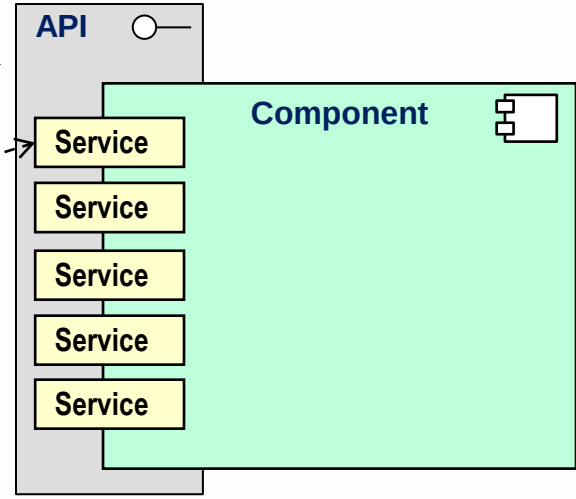


- hides internal resources (notably data structures) from external entities.



Encapsulation varieties

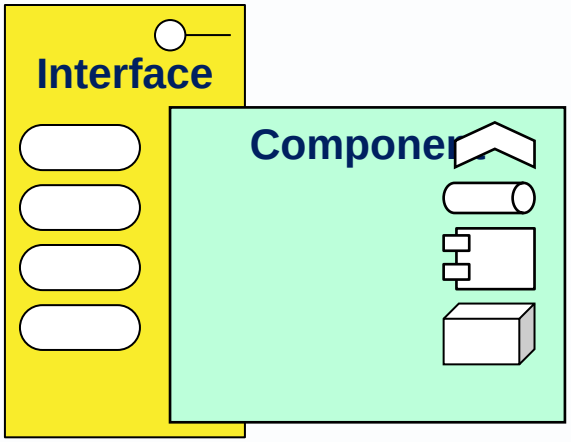
- **Structural encapsulation**
  - Encapsulating a component by an interface it offers.
- **Behavioral encapsulation**
  - Encapsulating a process by a service contract.



	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component

Interface definition

- [A structure] that declares what services a client can invoke.
- It encapsulates one or components, and may detail each service thus:
- **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

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

System design principles

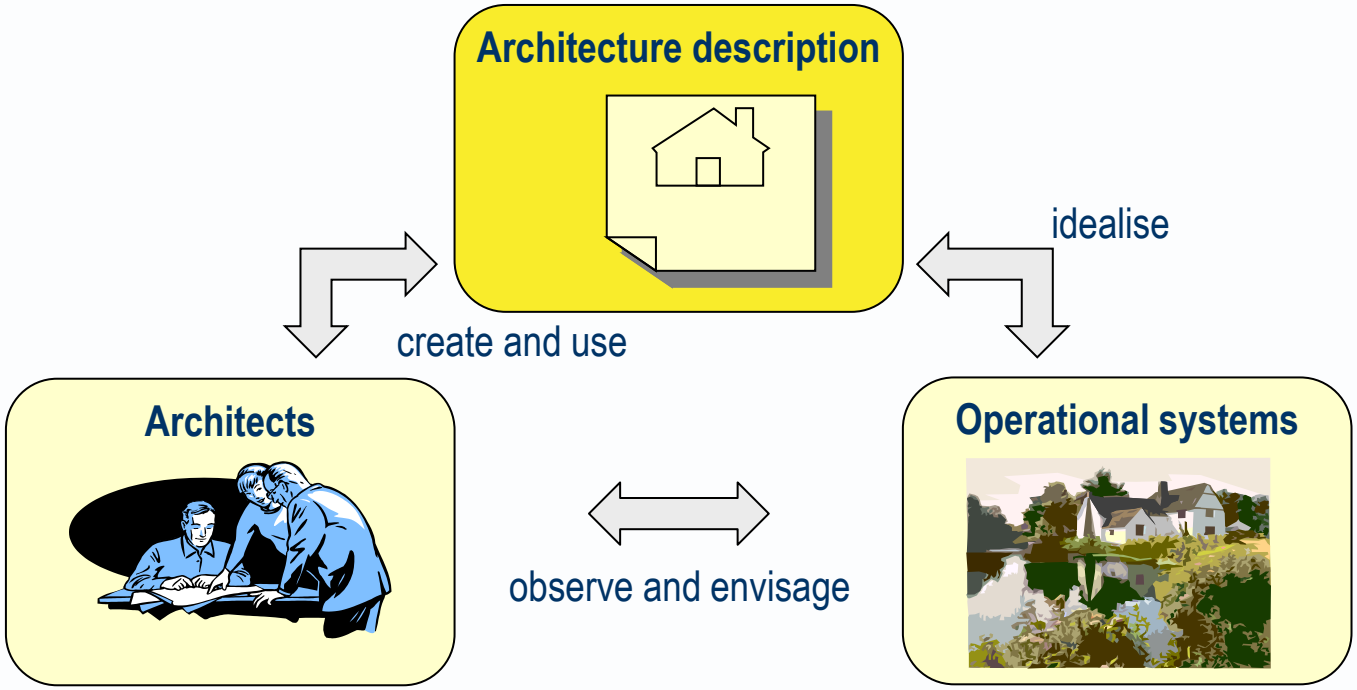
- Principles for the specification and modularisation of a system include:
  - external before internal,
  - behavior before structure,
  - business before technology,
  - logical before physical,
  - high cohesion within a component,
  - loose-coupling between components.

	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component

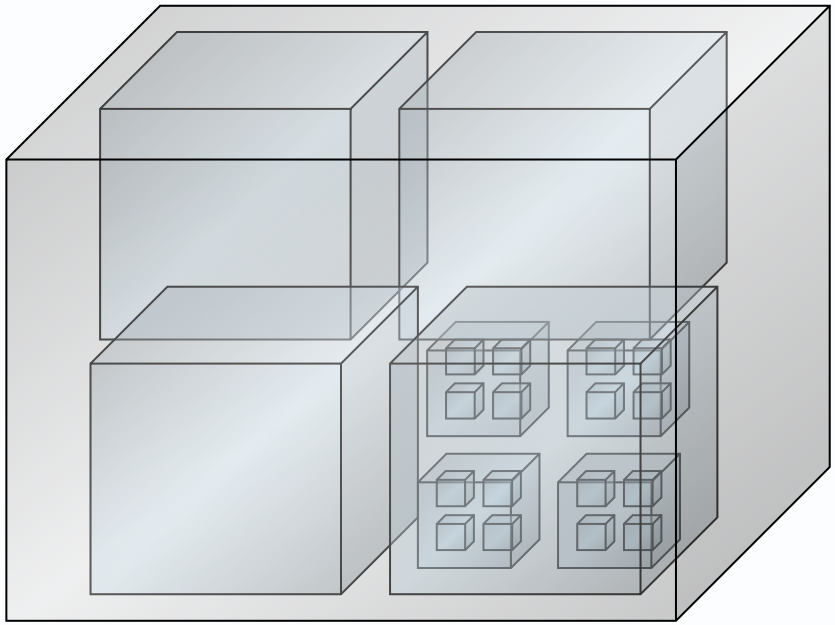
- Typically, finer-grained components are more tightly coupled, and coarser-grained components are more loosely coupled, so they can be managed and run independently.

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.

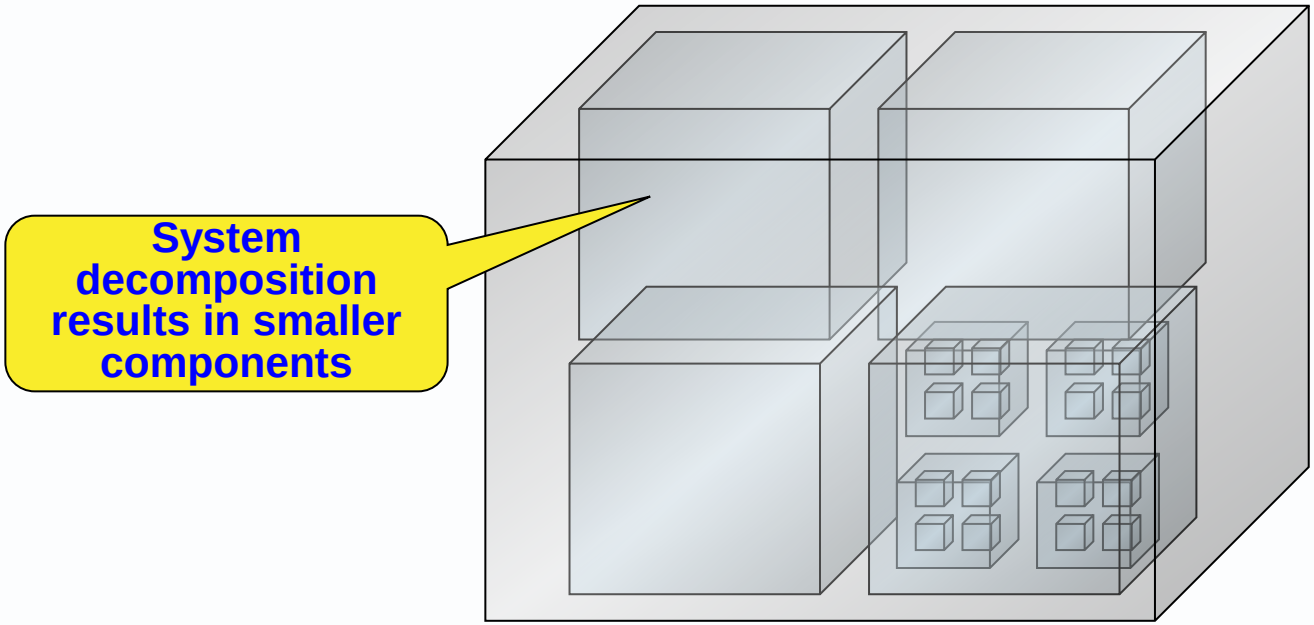


1.2 Structural decomposition



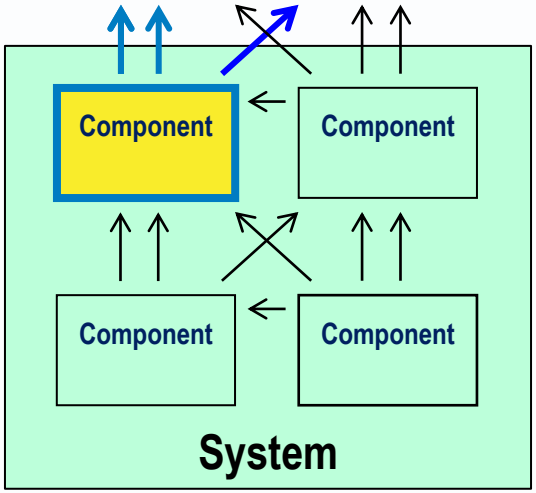
System decomposition

- [A technique] that divides a larger system into smaller components.
- You can see each component as a system in its own right and further subdivide it until you reach the level of atomic components you choose not to decompose.



Component

- [An active structure] or subsystem capable of performing some required behaviour.
- The required behavior can be defined in the form of one or more interface definitions.
- A component can be replaced by any other component with the same interface(s).
- EA frameworks address business, application and technology components.



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

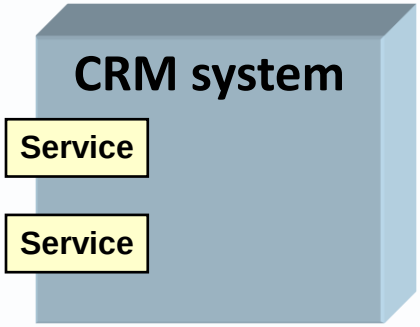
"A component represents a modular part of a system that encapsulates its contents. A component is a self-contained unit that encapsulates [internal] state and behavior." UML

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

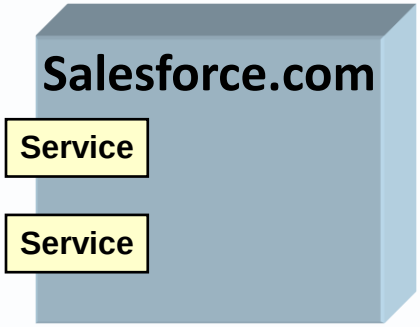
- [A component] that specifies the capability one or more physical components.
- It is abstract in the sense it is vendor and technology independent.
- It may be specified by services provided and resources (notably data) needed.



	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component

Physical component

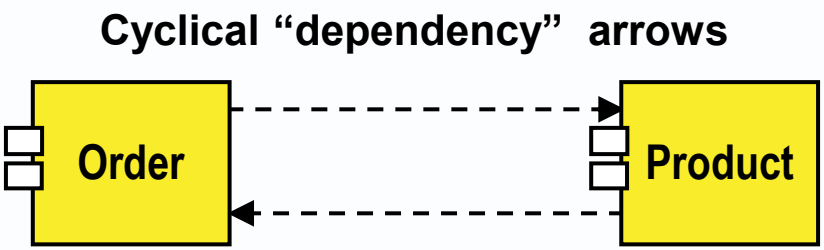
► [A component] that is vendor-specific and/or can realise a logical component.



	Behavior	Active Structure
External	Service contract	Interface definition
Internal	Process	Component

Dependency

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



- 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.

- Remember 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.

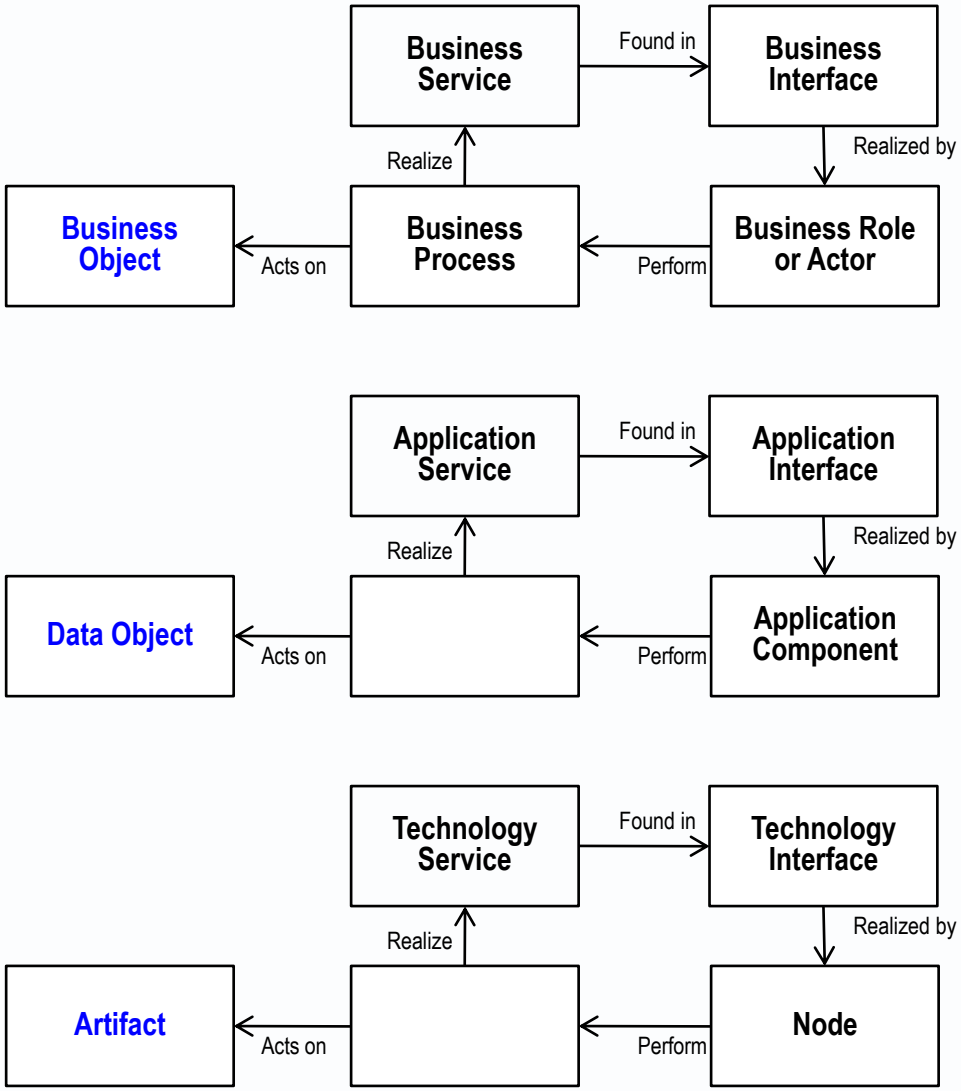
Modularity

- 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

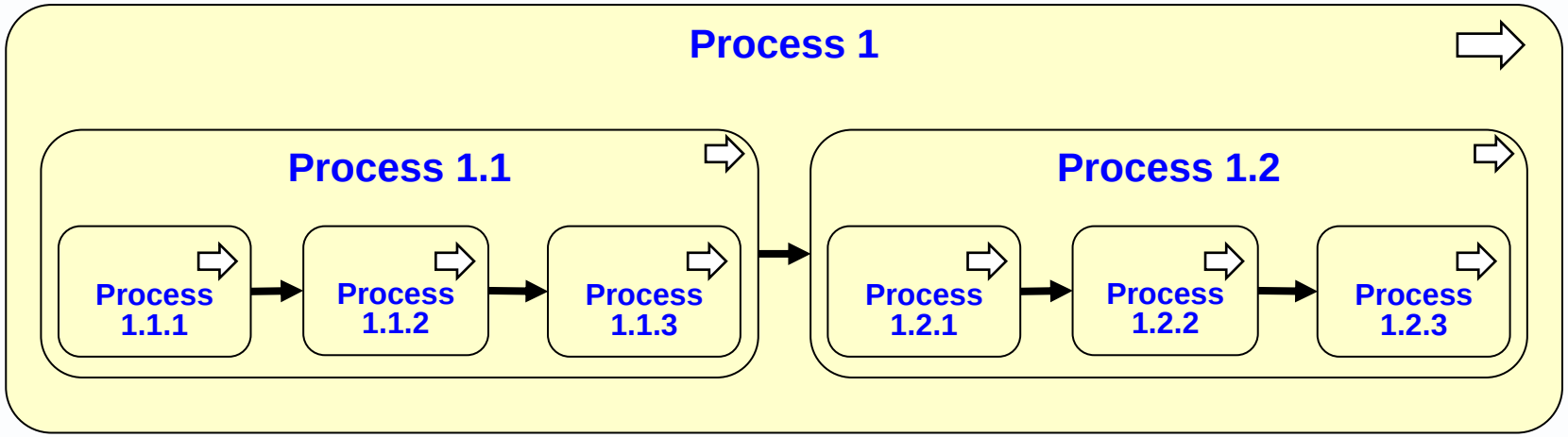
Avancier

Data component

- [A structure] in which some meaning or information is encoded.
- It can be created and used, moved and modified.

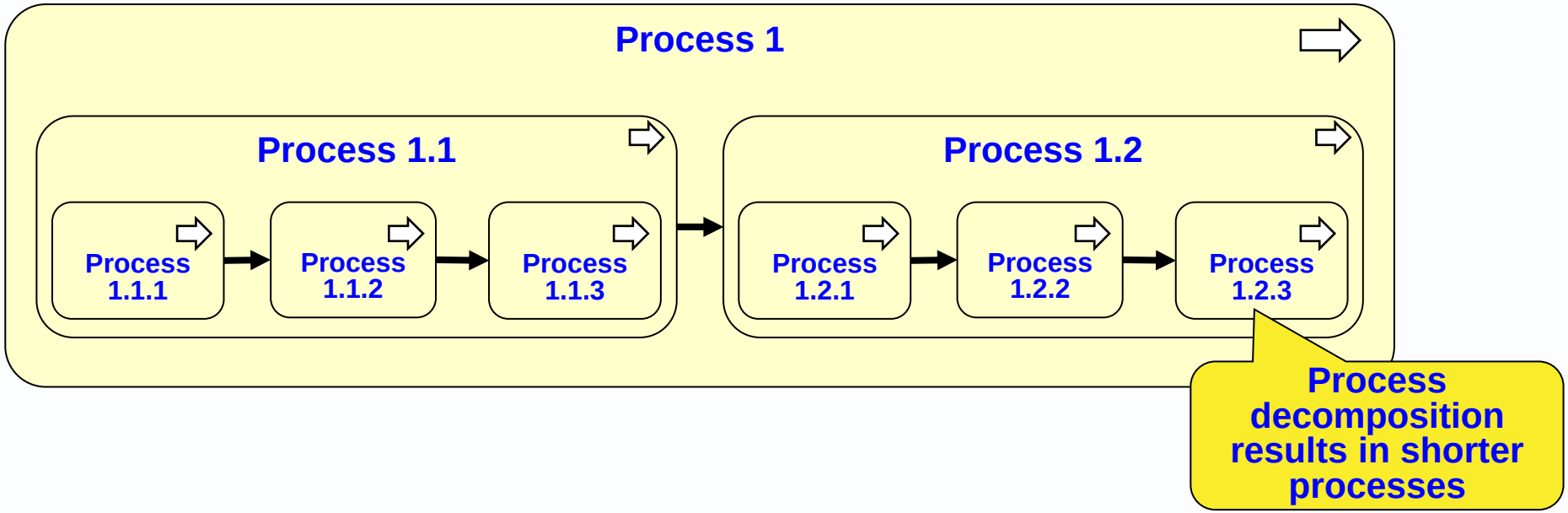


3 Behavioral decomposition



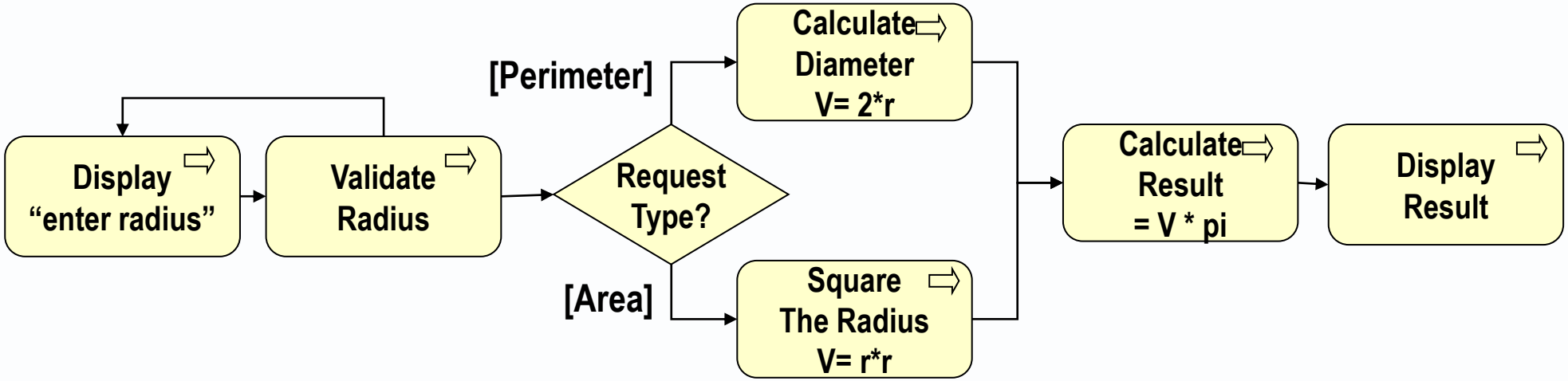
Process decomposition

- [A technique] that details an activity in a longer process as a shorter process of shorter activities.
- You can further subdivide it until reaching the level of atomic activities you choose not to decompose.



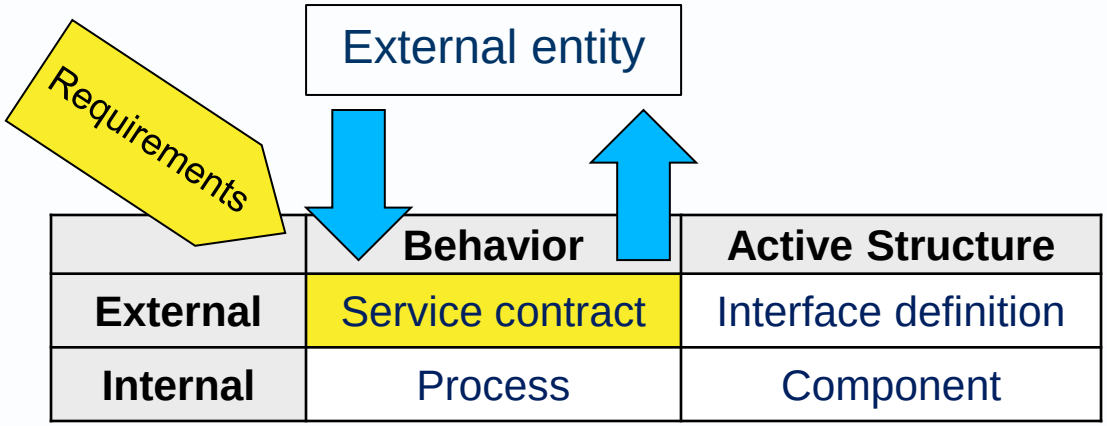
Process

- [A behavior] a sequence of activities that produce a result of value, meet an aim. Such as: 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 in sequence under logical control flow.
- It may orchestrate subordinate processes.



Service

- [A behavior] performed for a requester or consumer entity.
- It supports or enables that entity by delivering one or more results of value.



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

- Beware the term service is also used as a synonym for an interface definition, an application or an application component.

Service contract

- [A behavior definition] that encapsulate a process by defining it in terms of
  - entry conditions,
  - exit conditions (aka results) and
  - qualities,
- without reference what is needed to complete the service.
- It may be documented along with other service contracts in 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 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.

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.**

Service exit conditions (aka results)

Avancier

- 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.

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.

Service qualities

- Measurable attributes of a service, such as: duration, cost, availability.

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

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

1.4 Abstraction techniques

- To define architectures at a level that is more abstract than detailed design, architects use a mixture of abstraction techniques.

Composition	Delegation	Generalization	Idealization
Whole	Client	Universal	Conceptual
Part	Server/Client	Common	Logical
Atomic part	Server	Unique	Physical/Real
Decomposition	**Serving**	**Specialization**	**Realization**

Composition

- [A technique] that hides small things ones inside larger ones.
- A coarse-grained description features only large system elements.
- A fine-grained description features more detail by way of smaller system elements.

Composition
Whole Part Atomic part
Decomposition

Seen in EA standards as

Composition
Enterprise/Strategy Segment Solution/Capability
Decomposition

Delegation

- [A technique] that simplifies clients by hiding work they delegate to servers.
- A client component requests and/or consumes a service from a server.
- A server component performs (realises, completes) services for clients.

Delegation
Client Server/Client Server
Serving

Seen in EA standards as

Delegation
Business functions Applications Technologies
Serving

Generalisation

- [A technique] that simplifies by hiding differences between things.
- A *generic* description is more widely applicable or universal.
- A *specific* description is more narrowly applicable or unique; it may extend a more generic description with additional properties.

Generalization
Universal
Common
Unique
Specialization

Seen in EA standards as

Generalization
Foundation
Common system
Industry domain
Organization
Specialization

Design patterns

Reference models

Idealisation

- [A technique] that simplifies by hiding physical and/or vendor-specific details.
- The classic idealisation hierarchy is conceptual, logical, physical, real

Idealization
Conceptual
Logical
Physical/Real
Realization

Seen in EA standards as

Idealization
Requirements
Architecture
Solution
Deployed solutions
Realization

The classic idealization hierarchy

- **Conceptual model**
  - [A model] that defines terms and concepts in a business or problem domain without reference to any computer or software application.
- **Logical model**
  - [A model] that excludes details of a system's physical implementation.
  - It is vendor-independent and portable.
  - It may specify services or processes to be performed, and/or data or abilities needed.
  - It leaves open the choice of particular components and technological products.
- **Physical model**
  - [A model] that is vendor-specific or includes implementation details.
  - It specifies a particular entity that can be employed or deployed to realise a logical model.

Four kinds of abstraction found in EA

Composition	Delegation	Generalization	Idealization
Whole Part Atomic part	Client Server/Client Server	Universal Common Unique	Conceptual Logical Physical/Real
Decomposition	**Serving**	**Specialization**	**Realization**

Seen in EA standards as

"The enterprise continuum"

Composition	Delegation	Generalization	Idealization
Enterprise/Strategy Segment Solution/Capability	Business functions Applications Technologies	Foundation Common system Industry domain Organization	Requirements Architecture Solution Deployed solutions
Decomposition	**Serving**	**Specialization**	**Realization**

1.5 Architecture levels

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

- [A system] a business or organisation in which actors are directed to meet shared goals.
- It may be a segment of an organisation, or a collaboration between organisations

Enterprise architecture

- [An architecture] that gives a cross-organisational and strategic view of business systems.
- It includes business, data, application and technology component portfolios.

Solution architecture

- [An architecture] of a solution to a problem or requirement.
- It is usually developed within the context of a programme or project.
- It might be documented without reference to a higher or wider enterprise architecture.

Software architecture

- [An architecture] showing the modularisation and integration of software components.
- Some principles and patterns of software architecture are useful at higher levels.
- 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.

Strategic coordination view

- [A description] that shows the degree to which an organisation aims to standardise and/or integrate business processes and business data, presented as two-by-two grid of process standardisation against integration.

High process integration	**Coordination** **Shared** customers, products, **data.** Different organisations, functions, processes	**Unification** Globally integrated Standard processes and data Central IT
Low	**Diversification** Different customers, products, data Different organisations, functions, processes	**Replication** Different customers, products, data **Shared** organisations, functions, **processes.**
	Low	**High process standardisation**

Fig. 2.2 Strategic coordination view
(From “EA as Strategy” by Ross, Weill and Robertson)

1.6 Architecture domains

Avancier

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

Architecture domain

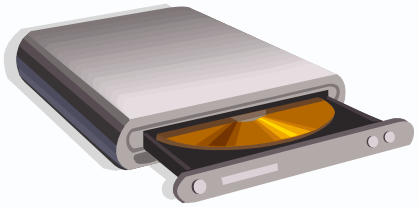
- A division or view of an architecture that addresses a broad set of concerns.
- The four architecture domains below were established in the PRISM report (1986).
- They are now the basis of countless EA frameworks, including TOGAF.

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 resources needed to perform processes.
- Business elements may be mapped to goals and locations, to business data and applications.

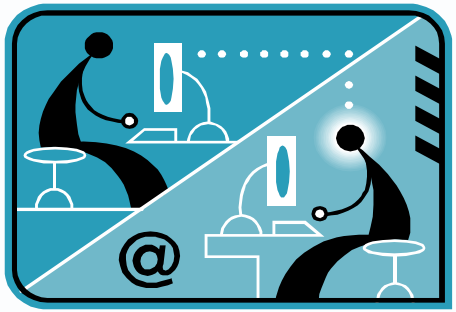
Data architecture

- [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.



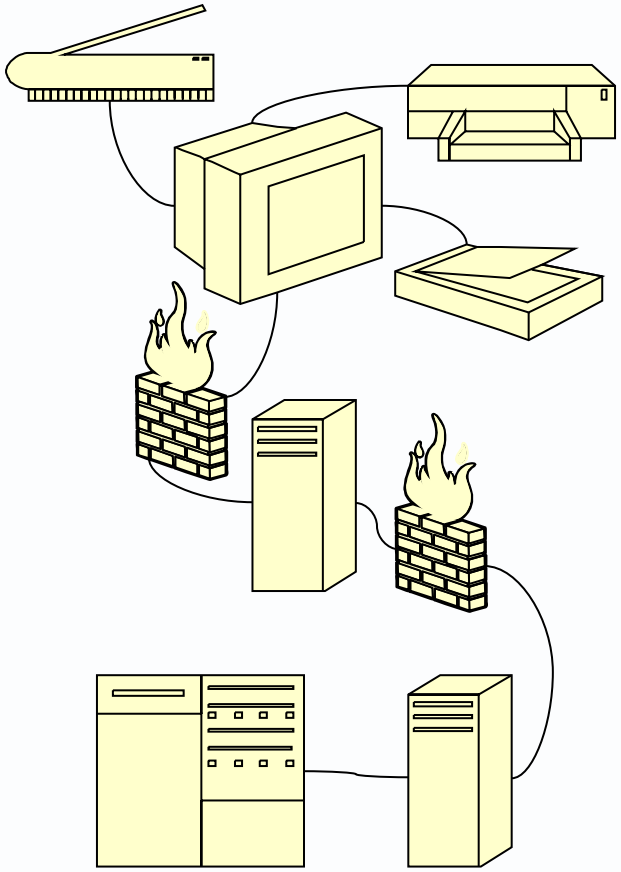
Applications architecture

- [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.



Infrastructure/technology 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.



Other domains

- Other “architecture domains” such as motivation, security, governance, may span the four primary domains above.

1.7 Architecture capability – roles, goals and skills

- **Architect**
- [A role] that involves
  - analysis of a context and requirements,
  - 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 the target state, and governing that change.

- Higher level architect roles abstract from detail, operate more widely, with a broader scope, and may *govern* lower roles to a greater or lesser extent.

Enterprise architect role

- [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 role

- [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.

Enterprise architect goals

- 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

- [A property] Such as:
- 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.