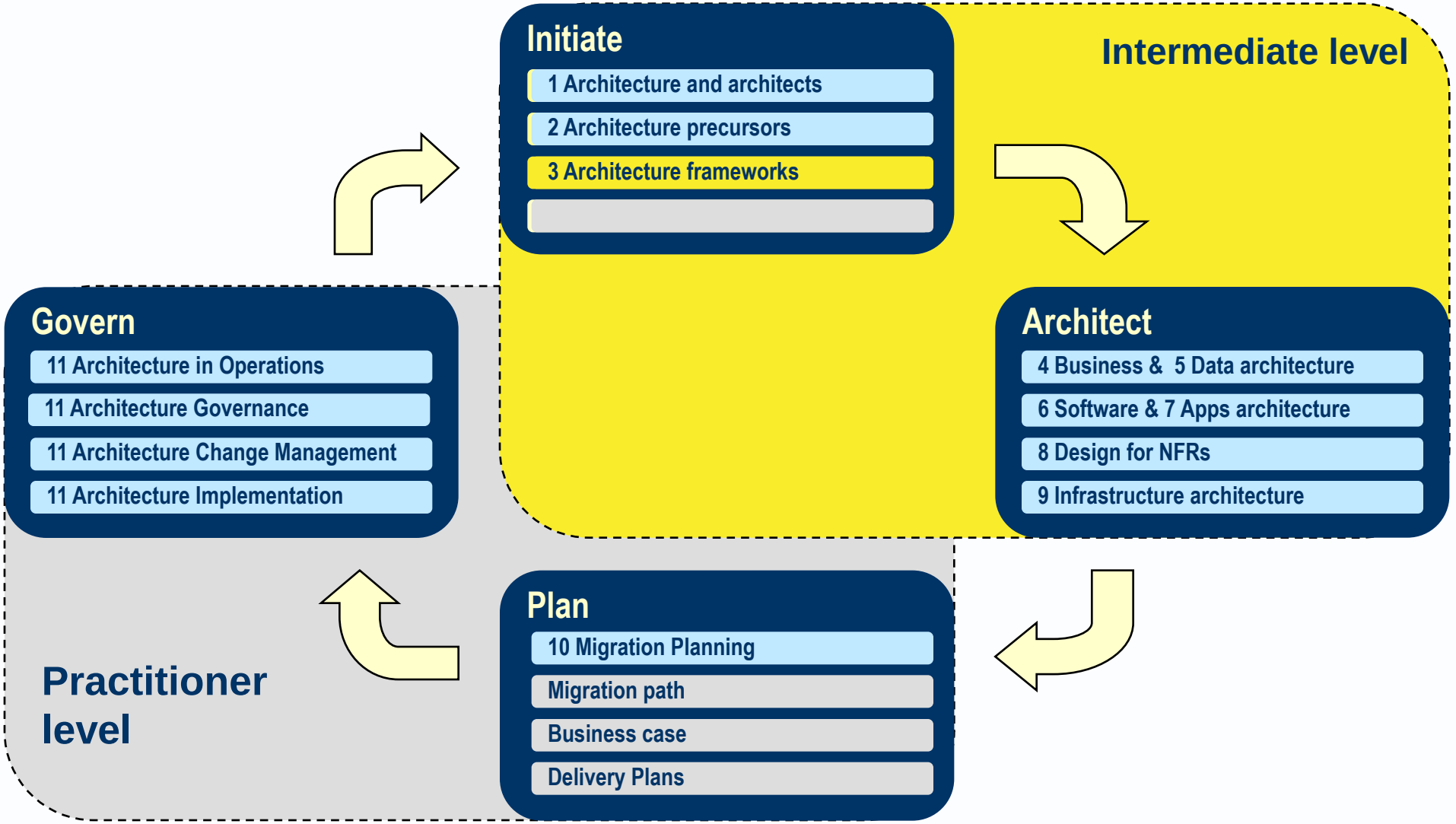
Avancier Reference Model

Architecture Frameworks (ESA 3)

Mapping the reference model to an architecture framework

Avancier



Architecture framework

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

Architecture state

- [an architecture description] at a point in time.
- A baseline architecture describes a system to be reviewed and/or revised.
- A target architecture describes a system to be created and implemented in the future.
- An intermediate or transition architecture defines a system between baseline and target.

Gap analysis reveals changes and work to be done

State	Baseline			Target
Domain				
Business	Process Organisation Locations			Process Organisation Locations
Information Systems	Data Applications			Data Applications
Technology	Infrastructure Technologies			Infrastructure Technologies

TOGAF: The Open Group Architecture Framework

- [An architecture framework] published on a free-to-read public web site.
- Its use by a commercial organisation is restricted by copyright conditions.
- It is a relatively abstract management framework for architecture work.
- It is centred on a process called the architecture development method.
- Originally focused on technology architecture; now focused more on business architecture.
- Originally intended for transforming an enterprise's business systems from a baseline state to a target state; now commonly used at a narrower and more tactical level.

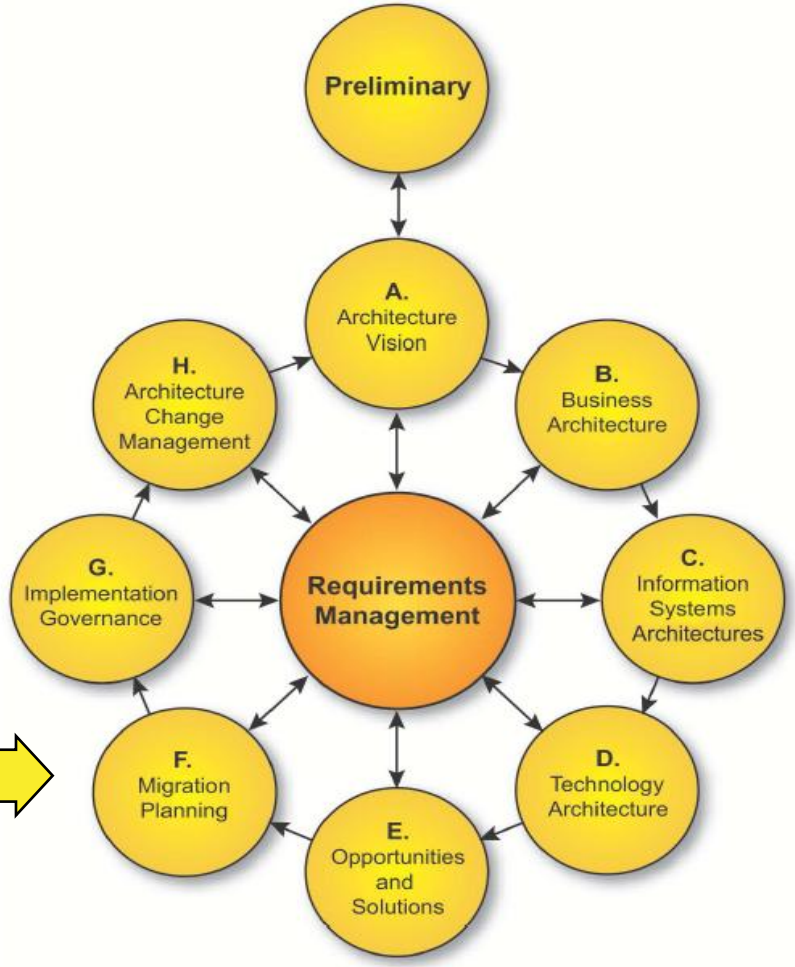


Figure 5-1 Architecture Development Cycle

ADM: Architecture Development Method

Avancier

- [The process] defined in TOGAF to develop and use an enterprise architecture.
- Users are expected to adapt the process and iterate through it.
- It is centered on a cycle of 8 phases.
  - A: Architecture Vision
  - B: Business Architecture
  - C: Information System Architecture (Data and Applications)
  - D: Technology Architecture
  - E: Opportunities and Solutions
  - F: Migration Planning
  - G: Implementation Governance
  - H: Architecture Change Management.

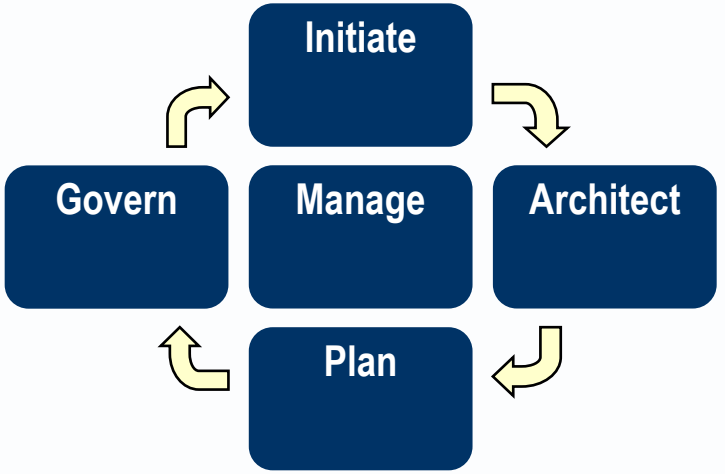
Process

Ex

Write Down Phase Names In ADM Circles

AM: Avancier Methods

- [an architecture framework] focused on solution architecture, though it also addresses enterprise architecture rationalisation.
- It is published on a free-to-read public web site, though its use by an organisation is limited by copyright conditions.
- It features a solution architecture process with four phases
- (Initiate, Architect, Plan and Govern),
- each subdivided into lower level processes.

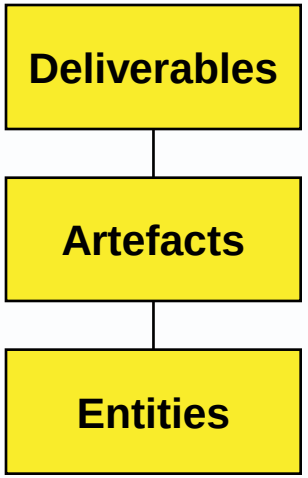


3.3 Architecture products

- Architecture content framework
- Architecture repository
- Mapping
- View
- Viewpoint
- Model

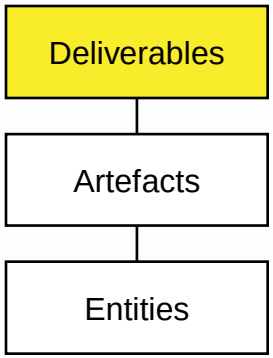
Architecture content framework

- [a passive structure] for organising an architecture description composed of deliverables, artefacts and entities.
  - Architecture deliverable
  - Architecture artefact
  - Architecture entity



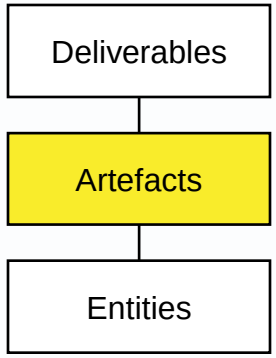
Architecture deliverable

- [a document] that architects produce or contribute to, for approval by sponsors if not all stakeholders.
- It should conform to a document type defined by a standard contents list.
- Deliverables often contain artefacts, which are in turn composed from architectural entities.



Architecture artefact

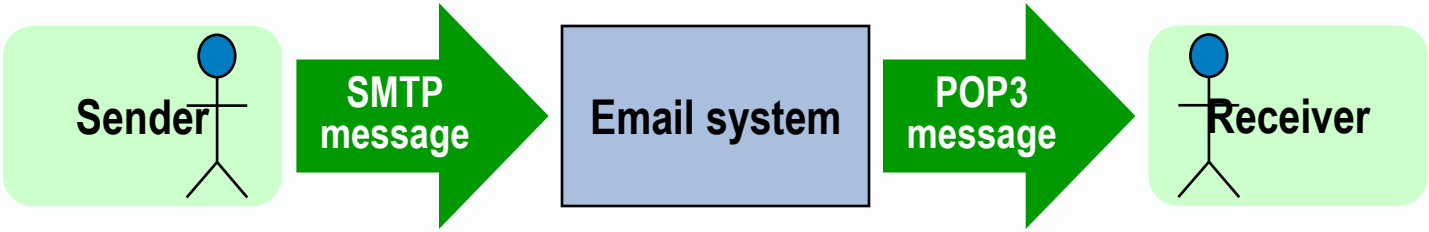
- [a model] that conforms to one of three artefact types:
- Catalogue



- Matrix

Location / Technology	Paris	New York	Hong Kong
PC	10,000	7,000	2,000
Printer	1,000	700	200
Photocopier	100	70	20

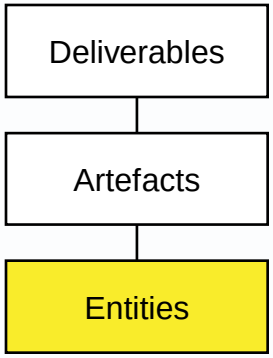
- Diagram.



Architecture entity

- [An entity] that appears in one or more artifacts.

- E.g.
  - Process,
  - Organisation,
  - Location,
  - Data entity,
  - Application,
  - Technology.



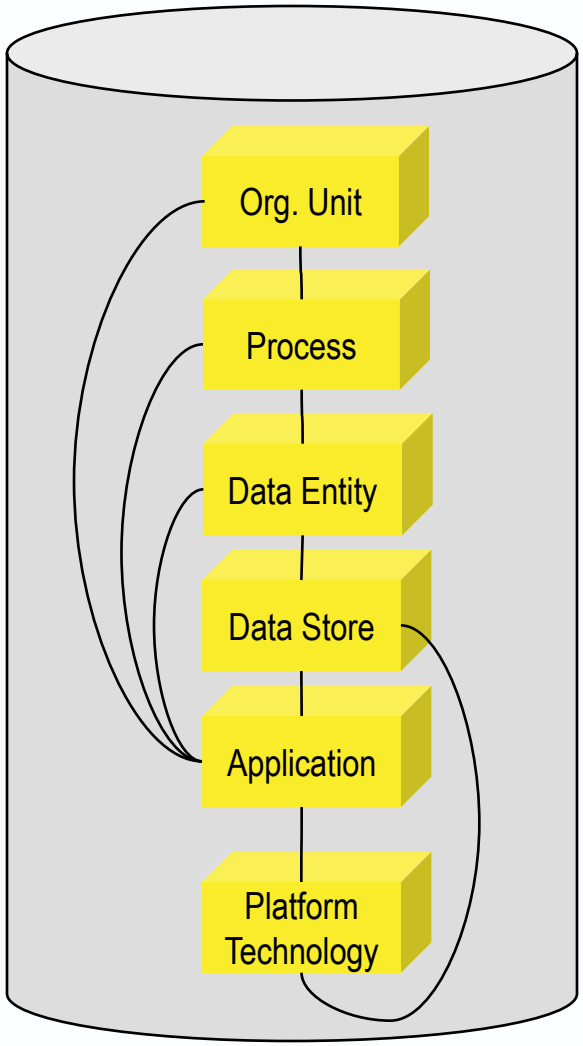
How architecture description develops

Avancier

1. You start off writing a deliverable as one document
2. You write sections covering business, apps and technology concerns, mentioning entities such human roles, business processes, apps and technologies
3. You insert various artefacts (tables and diagrams) to show the relationships between the Entities
4. The artefacts refer to the entities by name
5. You describe the entities more fully in catalogues in appendices
6. You divide the document into documents for different stakeholders with different concerns
7. Your overall description has now become so complex and distributed that (behind the scenes), you turn the appendices into a set of spreadsheets (a repository) from which you copy content into deliverables for stakeholders to read.

Architecture repository

- [A data store] an information base used by architects; a database that holds descriptions of an enterprise's business and its systems, and the meta data that describes those descriptions.
- Its structure is defined in some kind of schema or architecture meta model.
- The content of the repository can be categorised in many ways.
- For example, using the Zachman Framework or the Enterprise Continuum.



Mapping

- [a correspondence] that is drawn between elements of the same or different structures.

	Loc	Loc	Loc
Org	Works at		
Org		Works at	Works at
Org	Works at	Works at	

- Correspondences can be mapped for several purposes including:
  - gap analysis (see section 10),
  - impact analysis (see section 11),
  - requirements traceability analysis
  - cluster analysis (see section 4).

Mappings for gap analysis

- Gap analysis means looking for
  - items with no relationship,
  - loose ends, black holes and
  - gaps that may require attention.
- If the correspondence if close to one-to-one, then in a simple table will suffice:

Baseline Office Buildings	Target Office Buildings
London	London
Paris	Paris
New York to be closed	
	Mumbai to be opened

Mappings for gap analysis between states

- What elements in this architecture **state**
- don't match or relate to elements in another architecture **state?**

Target Apps / Baseline Apps	Billing	CRM	Business Intelligence	Baseline not in target
Billing	Port to new platform			
CRM		Leave as is		
Resourcing			**???**	**Decommission or reconsider**
Target not in baseline			**Buy or build**	

Mappings for gap analysis between domains

- Which elements in this architecture **domain**
- don't match or relate to elements in another architecture **domain**?

Technologies / Applications	DBMS	Messaging	ESB	Apps with no technology
CRM	Supported by	Supported by		
Broker App		Supported by		
Employee Portal			**???**	**Employee Portal**
Technologies not used			**ESB**	

Mappings for traceability analysis

- Why is this element needed? What objectives does it help to meet?
- What valued service or output does it help to produce?

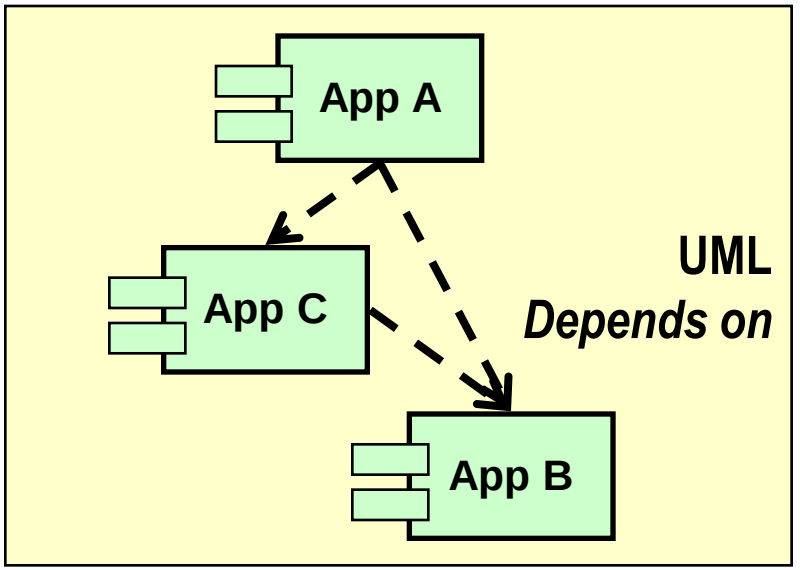
Solution items / Requirements	Business Intelligence	Sales Mobile device	CRM	Requirements with no solution
Faster Ordering		Satisfied by	Satisfied by	
Remote Working		Satisfied by		
Better Forecasts	Satisfied by			
Lower Sale Cost			???	No solution
Solution items with no requirement			No requirement	

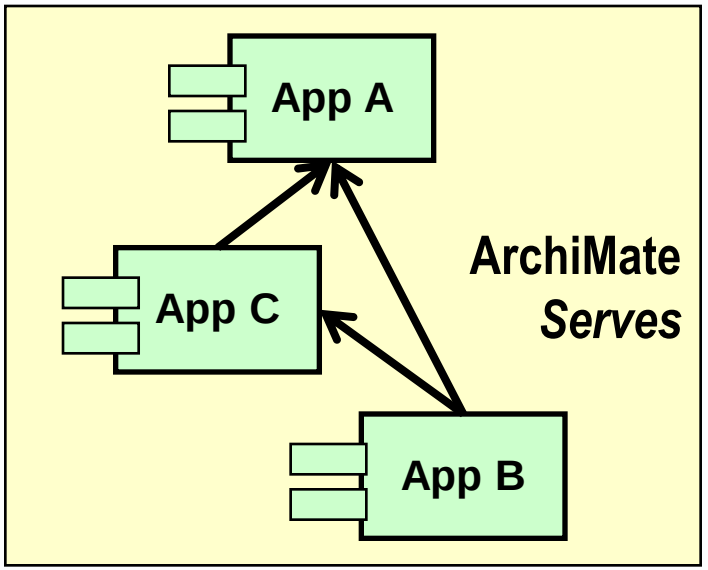
- Real life example: 40 solution elements
- 800 functional requirements (in a hierarchy) + 200 NFRs!

Mappings for change impact analysis

- If we change this element, what related elements may be affected?

	App A	App B	App C
App A		Depends on	Depends on
App B			
App C		Depends on	





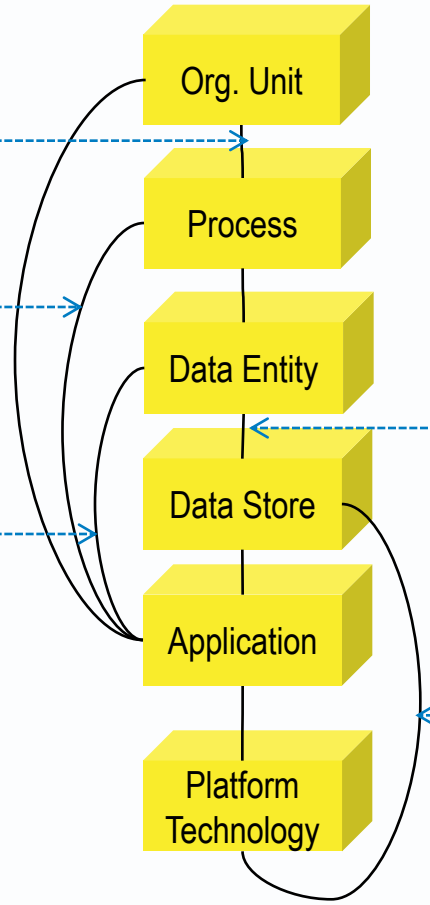
Mappings for change impact analysis

Avancier

	Process	Process	Process
Org	Performs		
Org		Performs	Performs
Org			Performs

	Process	Process	Process
App	supports		
App		supports	supports
App			supports

	App	App	App
Entity	cru	rud	
Product	crud	ru	r
Asset	r	crud	r



Store Data Entity	Data Store 1	Data Store 2	Data Store 3
Customer	x	x	
Product		x	x
Asset		x	x

	Tech	Tech	Tech
Data store	uses		
Data store		uses	uses
Data store			

Mappings for cluster analysis

- Which elements are closely enough related to be grouped in one component or work package?
- E.g. Which activities create the same data?

Actor or Activity Data element	Billing	Delivery	Sales	Reporting
Customer	Use	Use	Create	Use
Order	Use	Use	Create	Use
Delivery	Use	Create		Use
Invoice	Use	Create		Use
Payment	Create			Use
Report				Create

Actor or Activity Data element	Sales	Delivery	Billing	Reporting
Customer	Create	Use	Use	Use
Order	Create	Use	Use	Use
Delivery		Create	Use	Use
Invoice		Create	Use	Use
Payment			Create	Use
Report				Create

Mappings for cluster analysis

- Which elements are closely enough related to be grouped in one component or work package?
- E.g. Which activities create the same data?

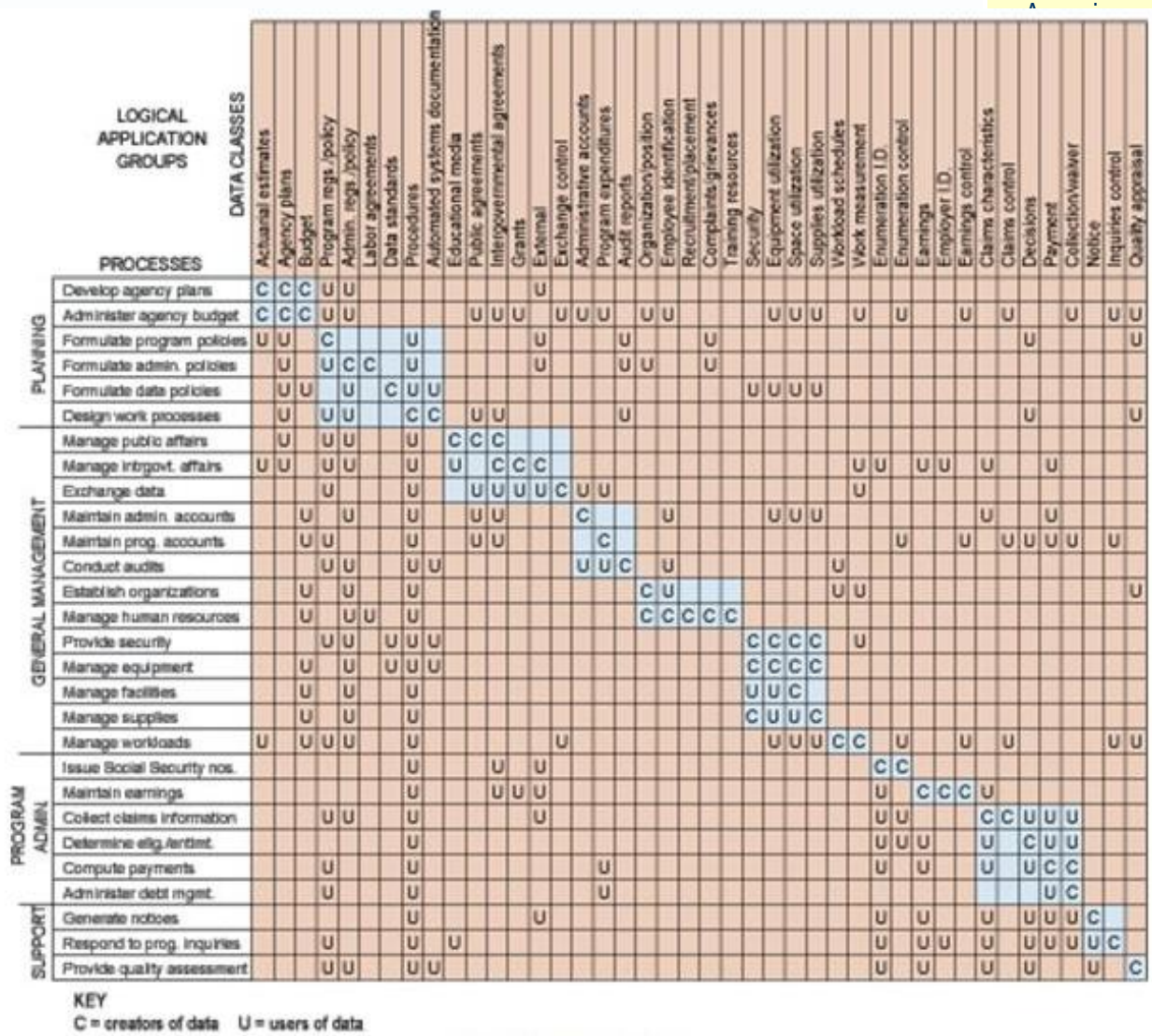
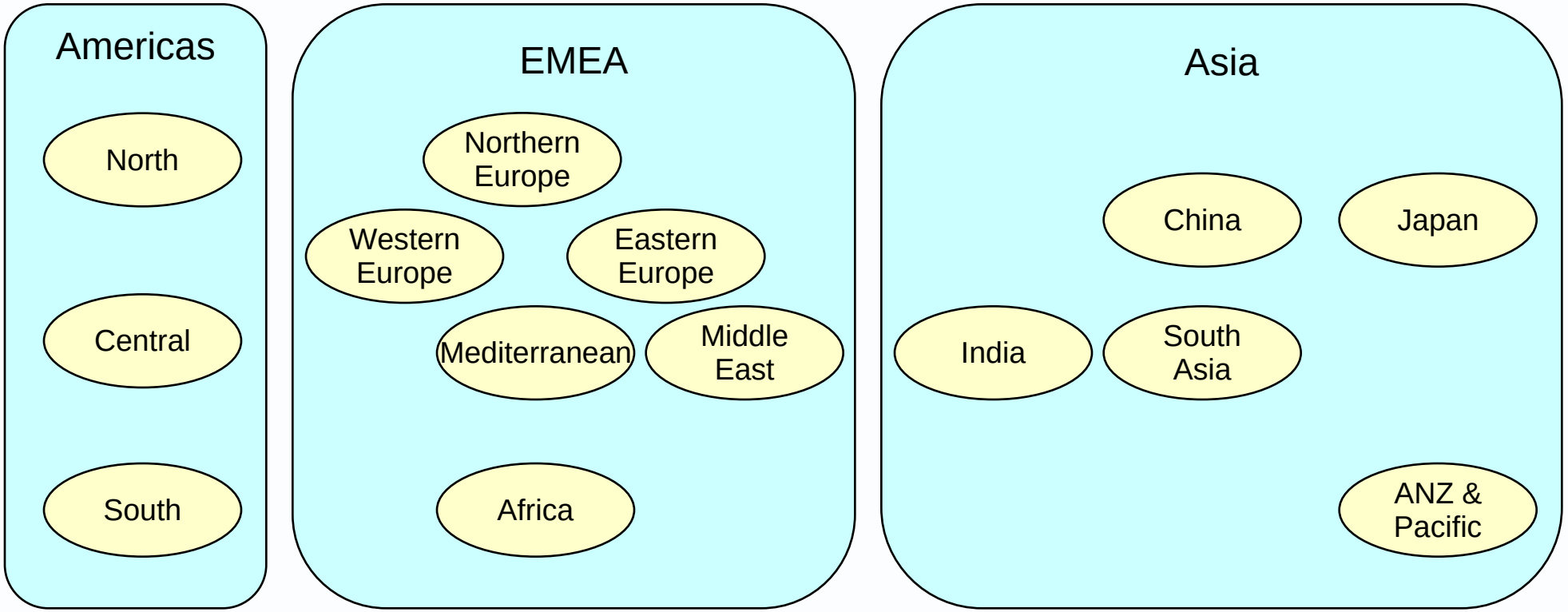


Figure 12-1

Enterprise A

Mappings for cluster analysis

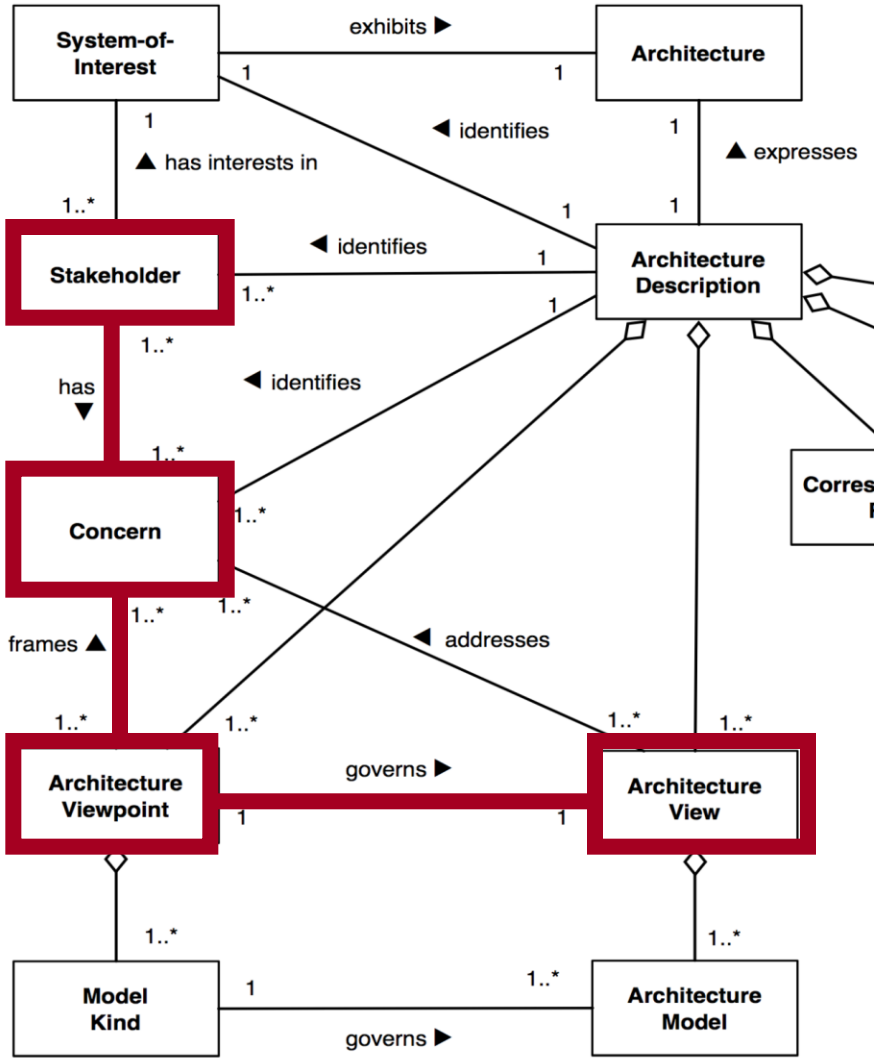
- Which elements are closely enough related to be grouped in one component or work package?
- E.g . Which entities are related by **location** and **time-frame?**



Four reasons to draw *mappings* between architectural entities

- Gap analysis
  - To find potentially missing items
- Traceability analysis
  - To check deliverables meet goals and solutions solve problems.
- Impact/Dependency analysis
  - To find the effects of a change
- Cluster analysis
  - To group closely-coupled items into encapsulated components

ISO/IEC 42010: Recommended Practice for Architecture Description of Software-Intensive Systems

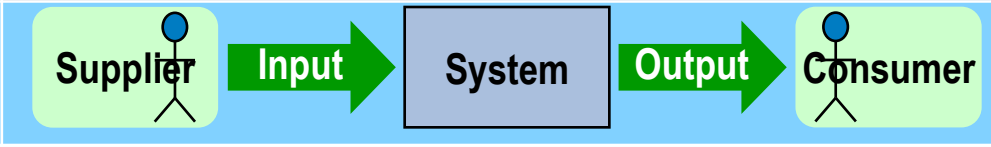


- A stakeholder can have several concerns.
- A concern can be shared by several stakeholders.
- A view is slice of the architecture, drawn to address *particular* concerns
- A viewpoint is the *type* of a view, a reusable *template* for drawing views to address *concerns in general*

Architecture viewpoints and views

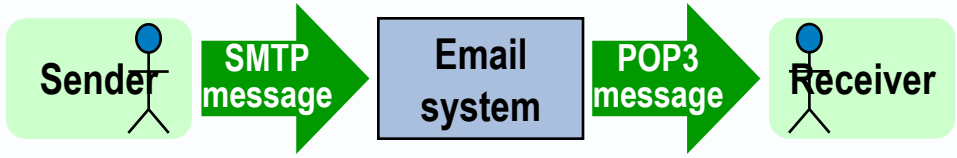
- Viewpoint

Template or type

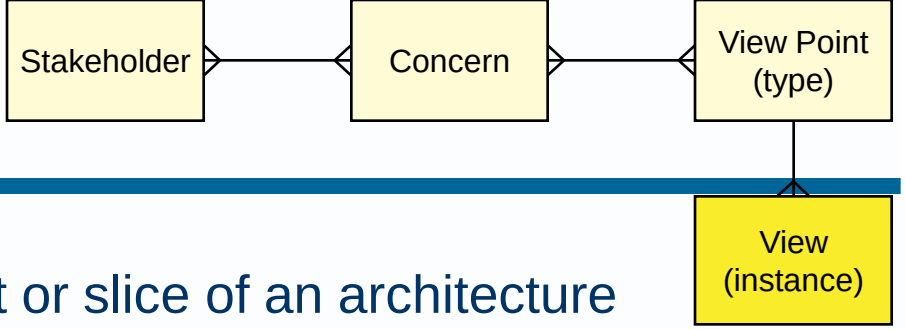
What	Why?	Who cares?	How to draw it?
Context diagram	The I/O scope of the system	Owners and designers	

- View

Example or instance



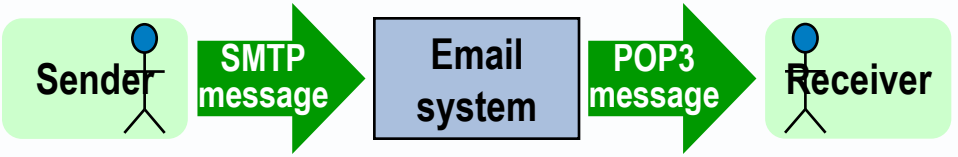
View



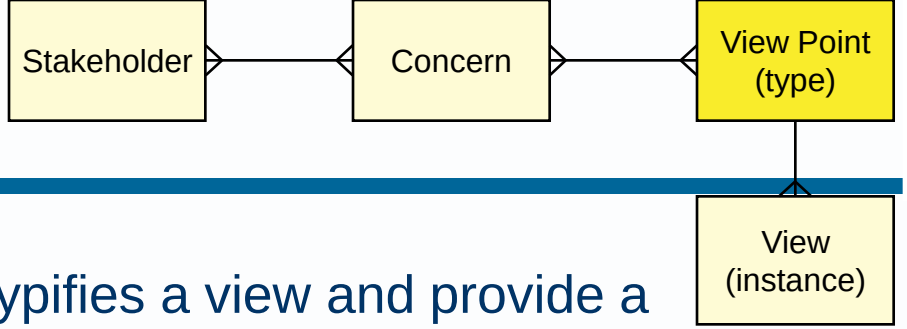
Avancier

- [a work product] that shows a part or slice of an architecture that addresses particular concerns.
- It can be visual, graphical or textual, and may contain one or more models.
- It can be an instance or example of a viewpoint, meaning that it conforms to the definition of that viewpoint.

Location Technology	Paris	New York	Hong Kong
PC	10,000	7,000	2,000
Printer	1,000	700	200
Photocopier	100	70	20



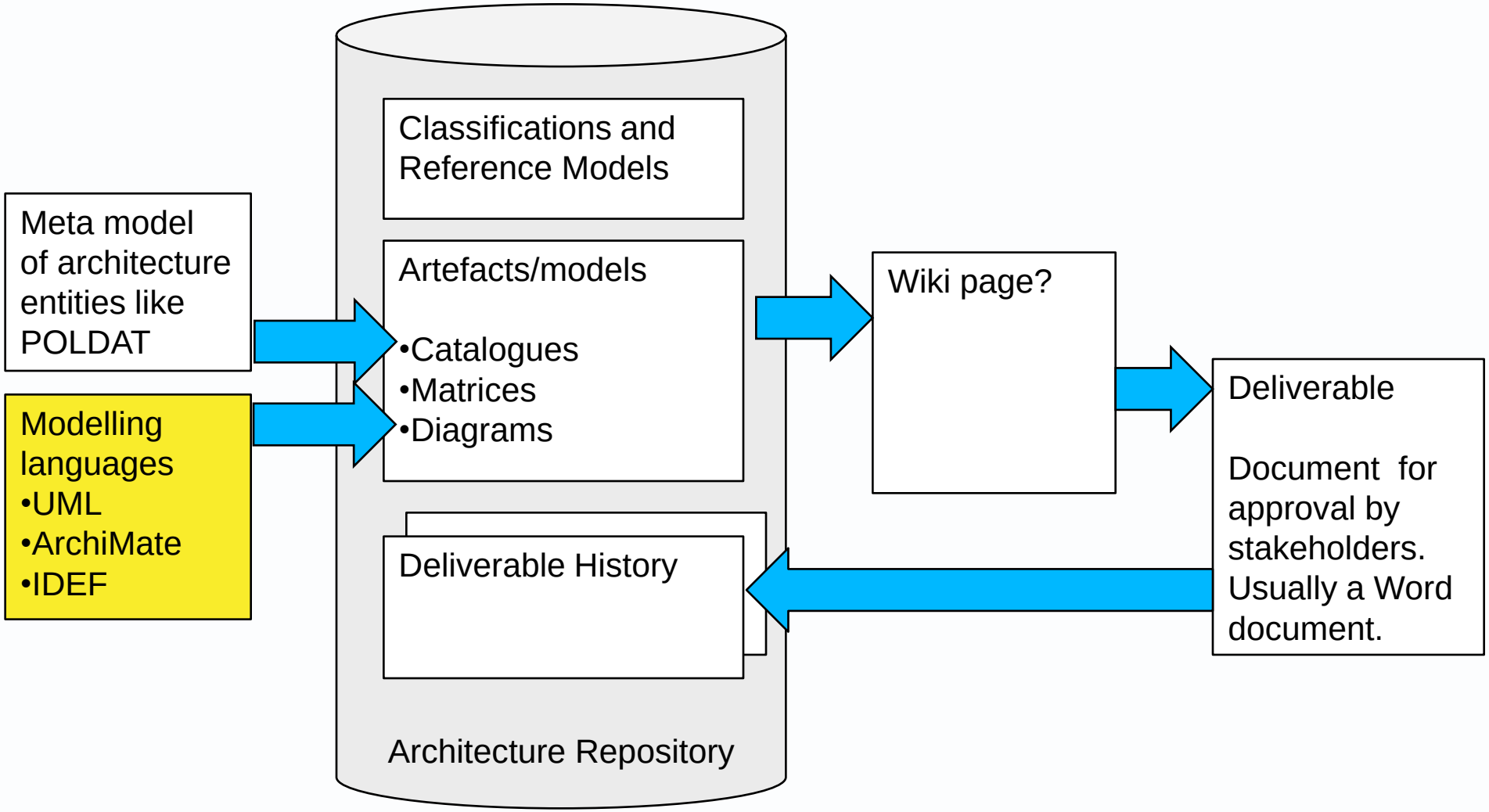
Viewpoint



Avancier

- [a work product description] that typifies a view and provide a template for it.
- It defines the conventions for creating and using views to address concerns about a system.
- It defines:
  - what – the name of the viewpoint
  - why - concern(s) that the viewpoint addresses
  - who - stakeholder(s) who have the concerns
  - how - model kind(s) used in the view.
- Within one architecture description, the viewpoint-to-view relationship is one-to-one.
- However, one viewpoint may be used as a template for views of many different systems.

Modelling languages



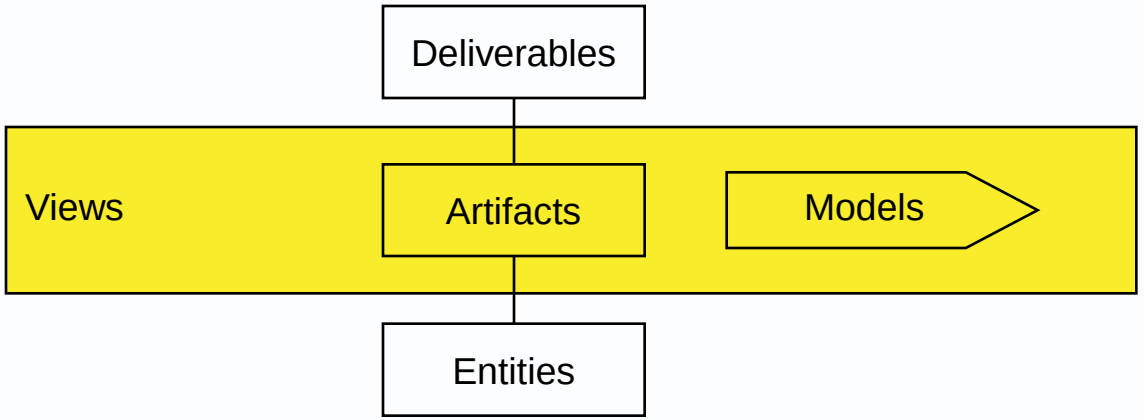
3.4 Architecture models

Model

- [A work product] that simplifies or abstracts from a thing or another description.
- It displays or records some properties of what is modelled.
- It enables some questions to about it to be answered.
- Architects build relatively abstract models of systems, and parts and views of them.

Model kind

- [A work product description] that typifies a model and provides a template for building a model.



Modelling language

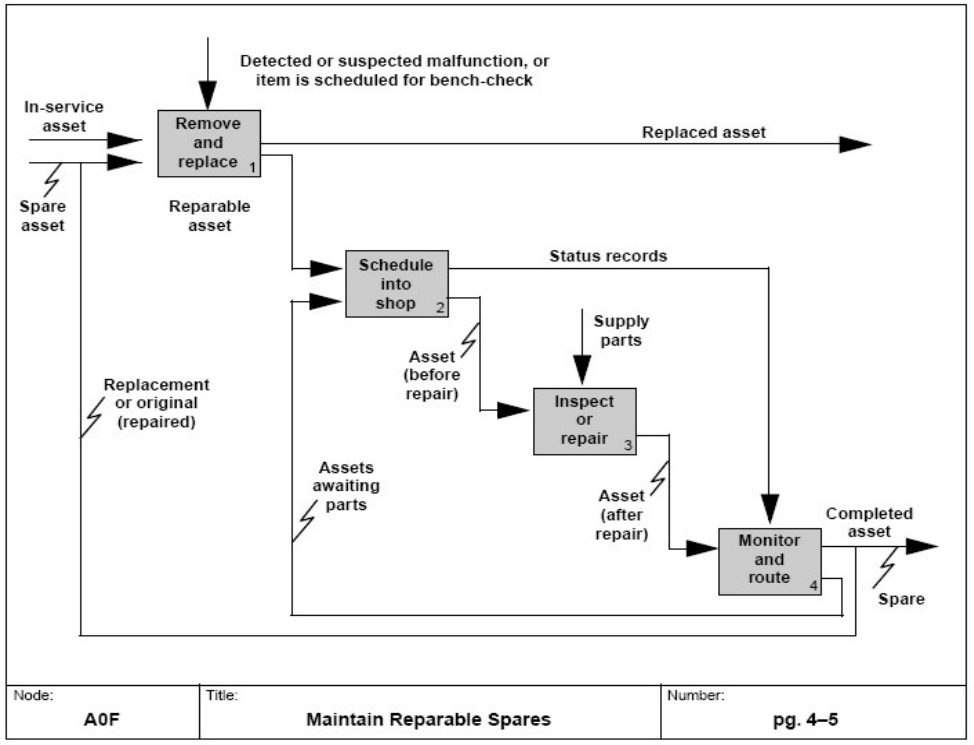
- [A standard] that defines shapes for representing architecture entities and arc/line styles for representing relationships between them.
- Three international varieties are IDEF, UML and ArchiMate.
- **IDEF: Integration DEFinition language**
  - [A modelling language] for systems and software engineering, originally funded by the US DoD.
  - Its includes IDEF0 (a process modeling language building on SADT) and IDEF1X for information models and database design.
- **UML: Unified Modelling Language**
  - [A modelling language] maintained by the Object Management Group.
  - Initially designed to help in OO software design, it is now used outside of that.
  - It includes structural models such as class diagrams and deployment diagrams.
  - It includes behavioural models such as use case, activity and sequence diagrams.
- **ArchiMate**
  - [A modelling language] maintained by the Open Group.
  - Components, interfaces and services are shown in distinct boxes.
  - It overlaps with UML, but is intended for more abstract architectural design.

Integration Definition language (IDEF)

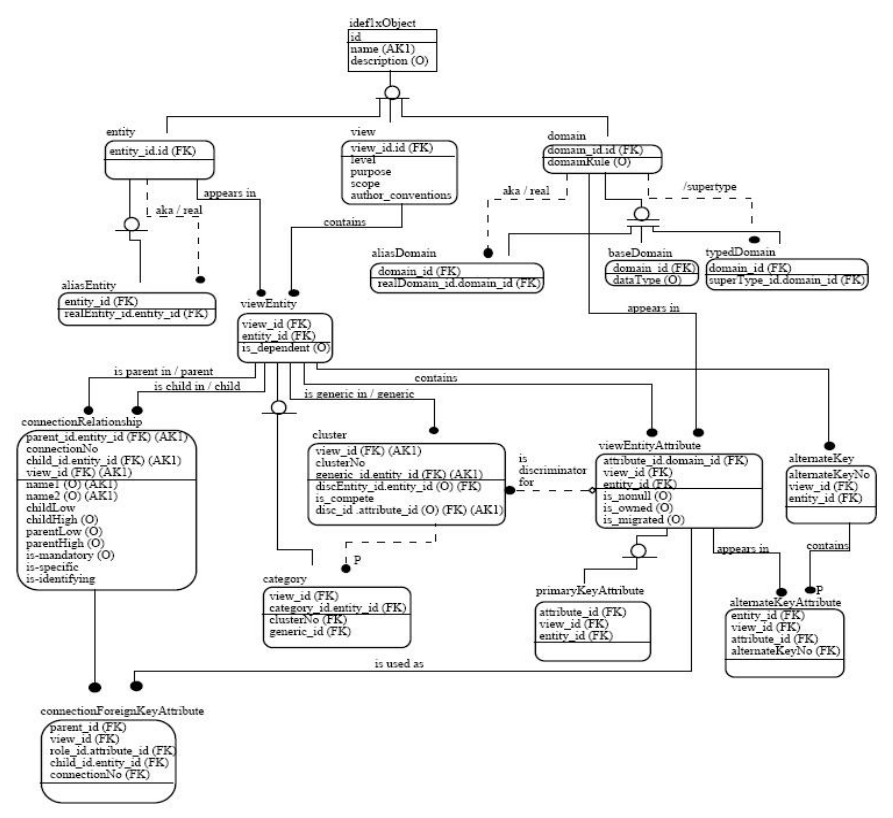
Avancier

- Grew out of 1970s USAF standards, best known for

- IDEF 0 Function Models



IDEF 1X Data Models (Wikipedia)



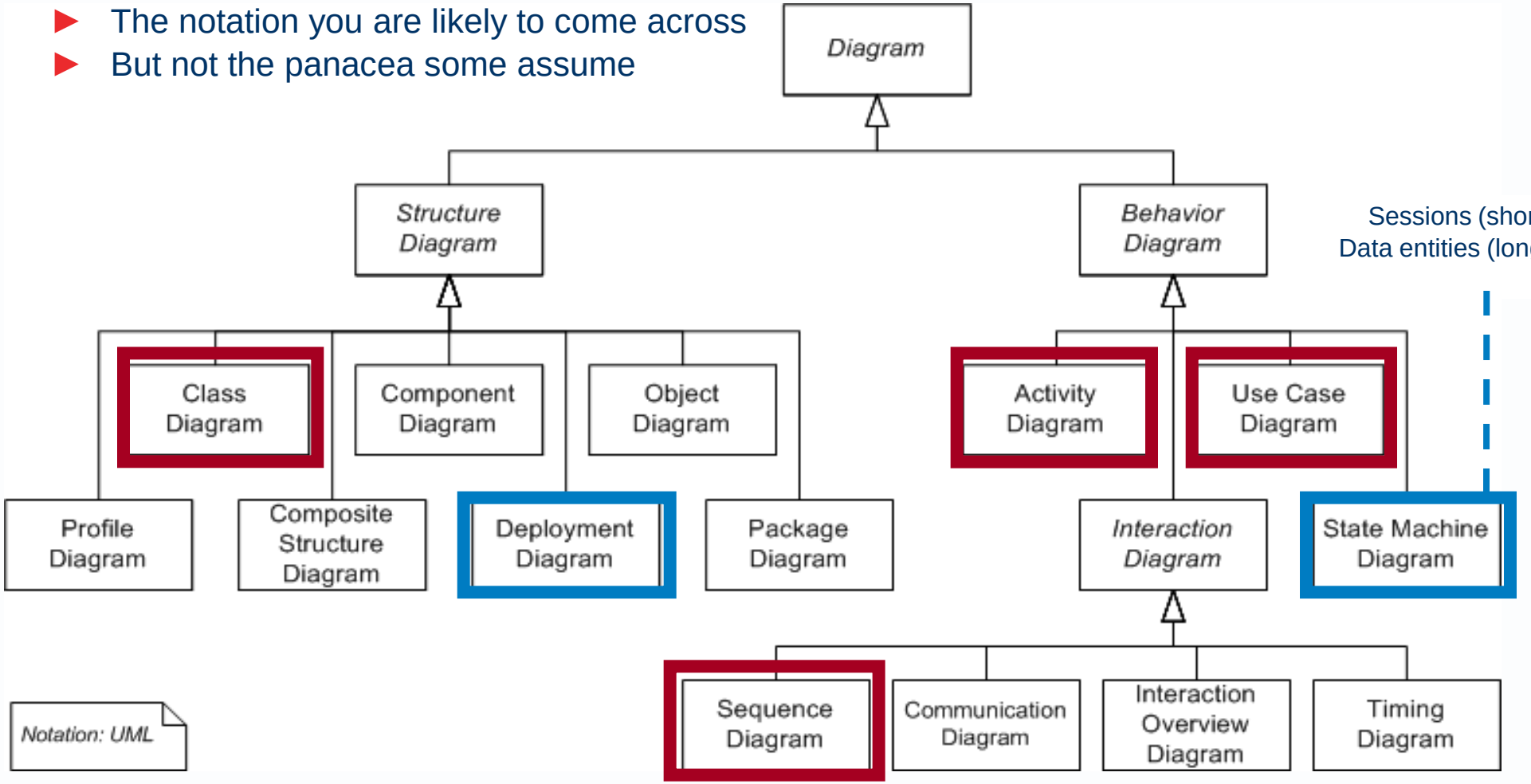
UML (OMG standard)

Most popular

Also

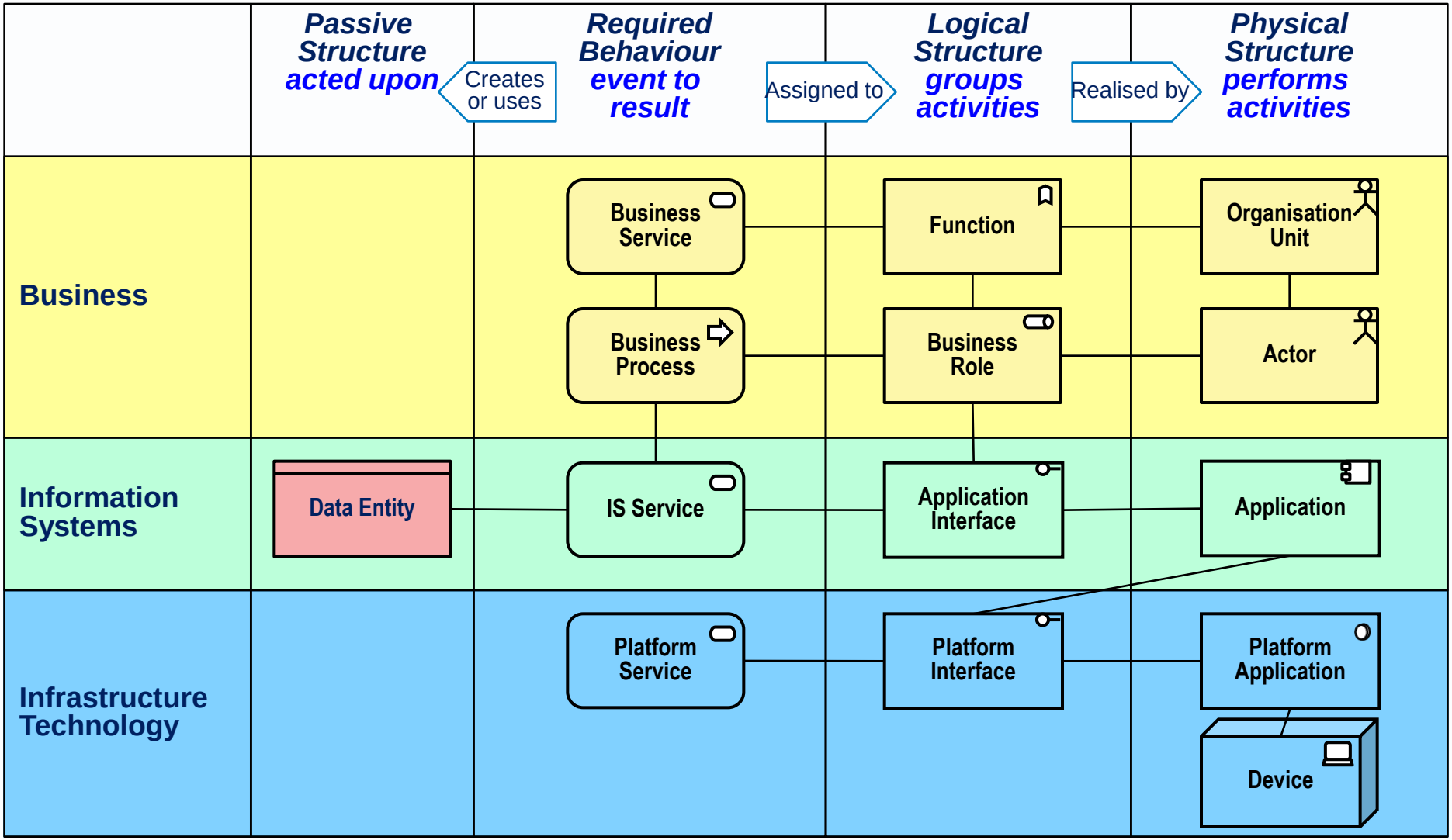
Avancier

- Created to assist in design of OO software
- The notation you are likely to come across
- But not the panacea some assume



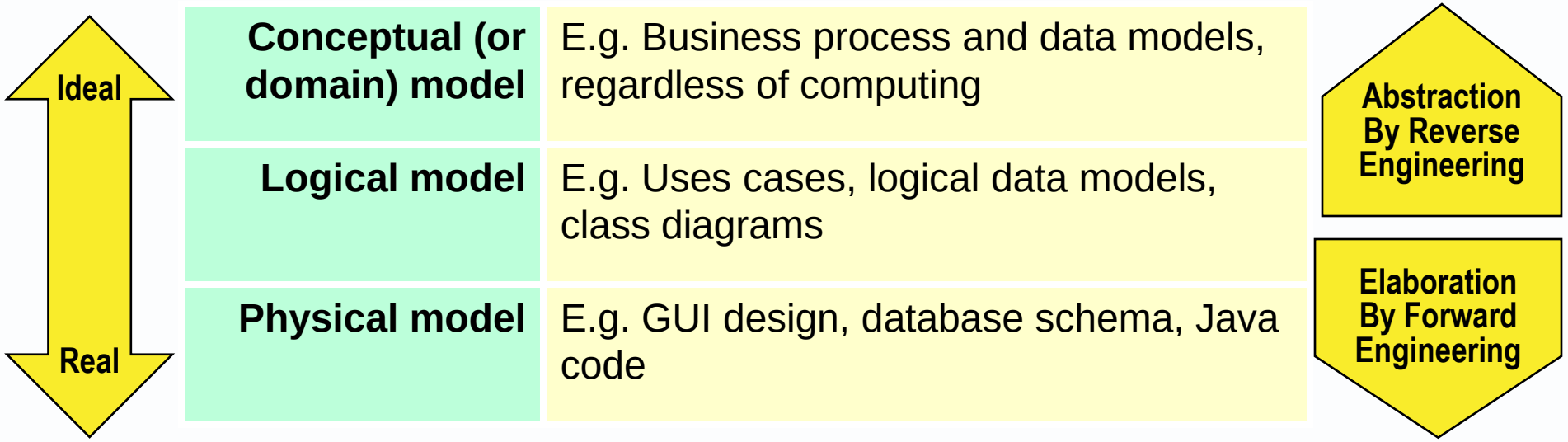
Sessions (short term)
Data entities (long term)

Core ArchiMate symbols



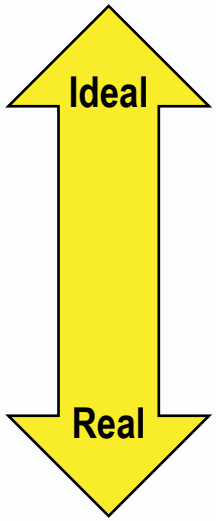
MDE: Model-Driven Engineering

- [A technique] used in methods and tools for forward engineering and reverse engineering.
- That is, the process of transforming a conceptual model to a logical model to a physical model, and the reverse of that process.

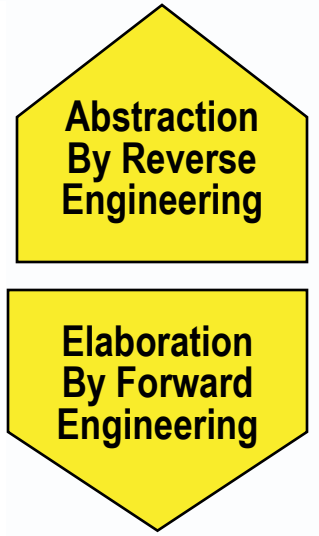


Model-Driven Architecture (MDA)

- a vision of the Object Management Group (OMG) that encourages suppliers to develop tools to standards defined by the OMG.
- The idealisation hierarchy is:
  - computation-independent model (CIM),
  - platform-independent model (PIM), unrelated to a specific technology
  - platform-specific model (PSM), related to specific technology.



CIM	Requirements in the form of business process and and data models, regardless of computing
PIM	Defines a system's functions, typically using a **UML class diagram**, or other language based on UML and/or the MOF.
PSM	Translation of a PIM into an implementable form, using a General Purpose Language like Java, C#, Python an OS and other platform technologies



Anecdotal evidence

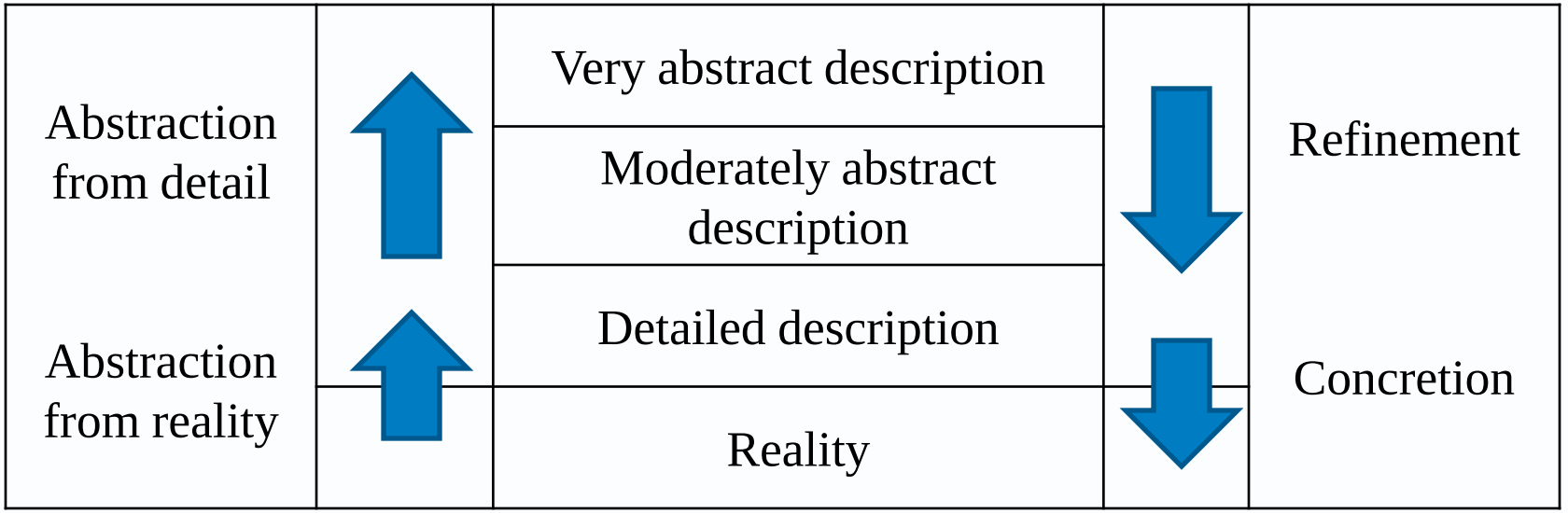
- People do like reverse engineering from code
  - because they can erase stuff from the model to leave a useful abstraction for discussion.

- People do not like forward engineering from model to code.
  1. The model has to be as detailed as the code – so there is no abstraction benefit – developers can work as readily with the code.
  2. Forward engineering leads to ugly, unreadable and sometimes inefficient code.
  3. Developers still have reasons to look at and work with the code anyway – which breaks the round-trip paradigm.
  4. People almost never have the requirement that justifies MDA – to port code from one platform to another
  5. If they do seek portability – they don't trust the tool vendor will keep step with all possible platforms and upgrades thereof.
  6. An enterprise is locked into a niche CASE tool, and developers who are trained in it.

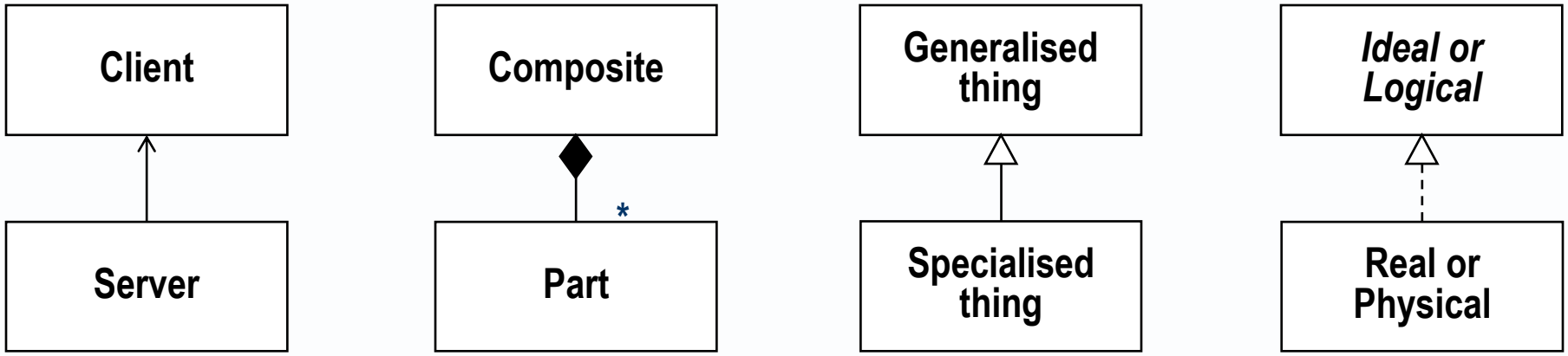
But see "Empirical Assessment of MDE in Industry" for a contrary view

3.3 Abstraction of architecture descriptions from systems

- Abstraction from reality
  - creating a description, the opposite of *concretion.*
- Abstraction from detail
  - creating a simpler description, the opposite of *refinement.*

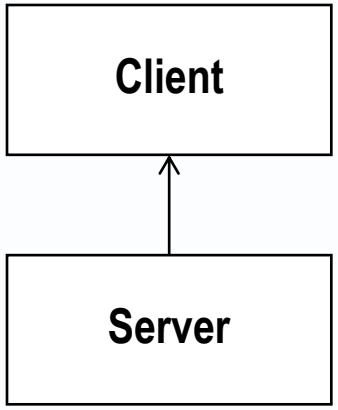


- Architects use a mixture of abstraction techniques to hide details and assist explanations.



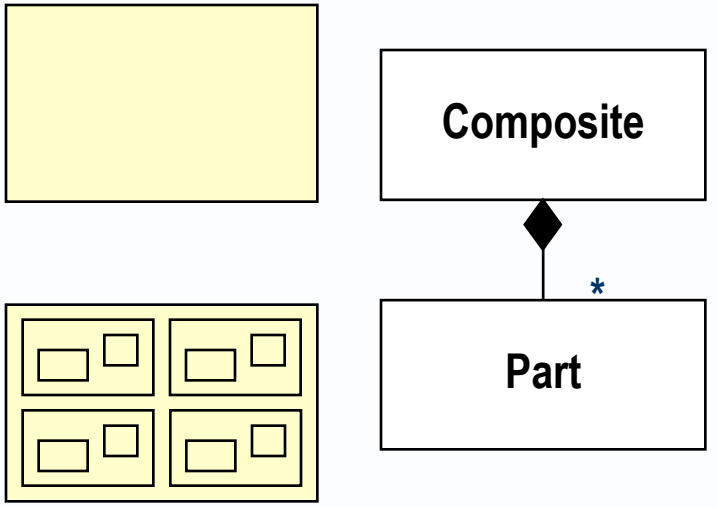
Abstraction by delegation (from server to client)

- [A technique] that simplifies clients by hiding work they delegate to servers.
- A *client* requests a service from a server; a *server* performs services requested by a client.



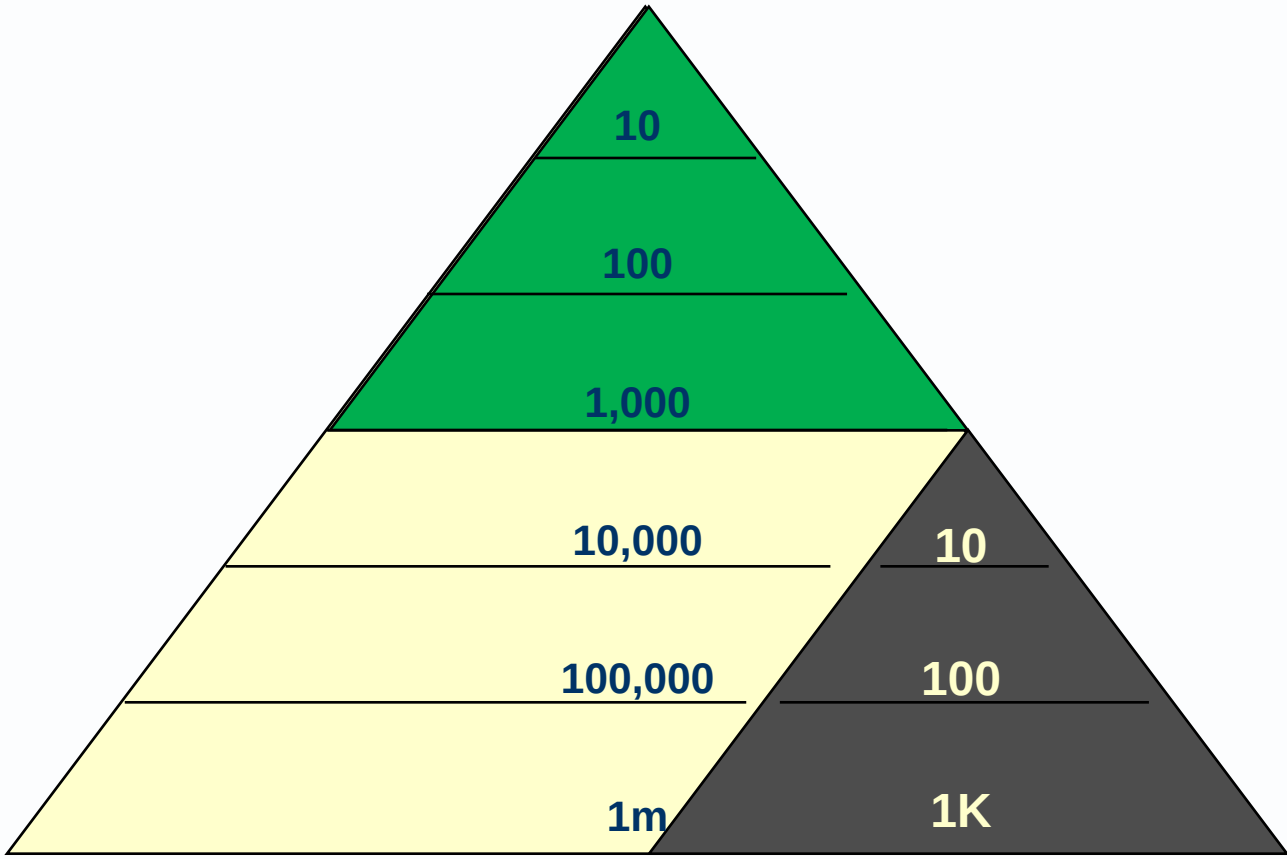
Abstraction by composition (from smaller to larger)

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



Composition – Coarse-grained views and models

- Enterprise architecture
- Applications portfolio

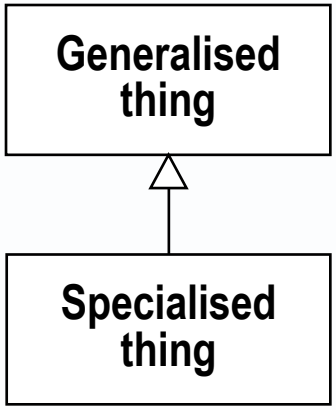


- Software architecture
- Application components

Abstraction by generalisation (from unique to universal)

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

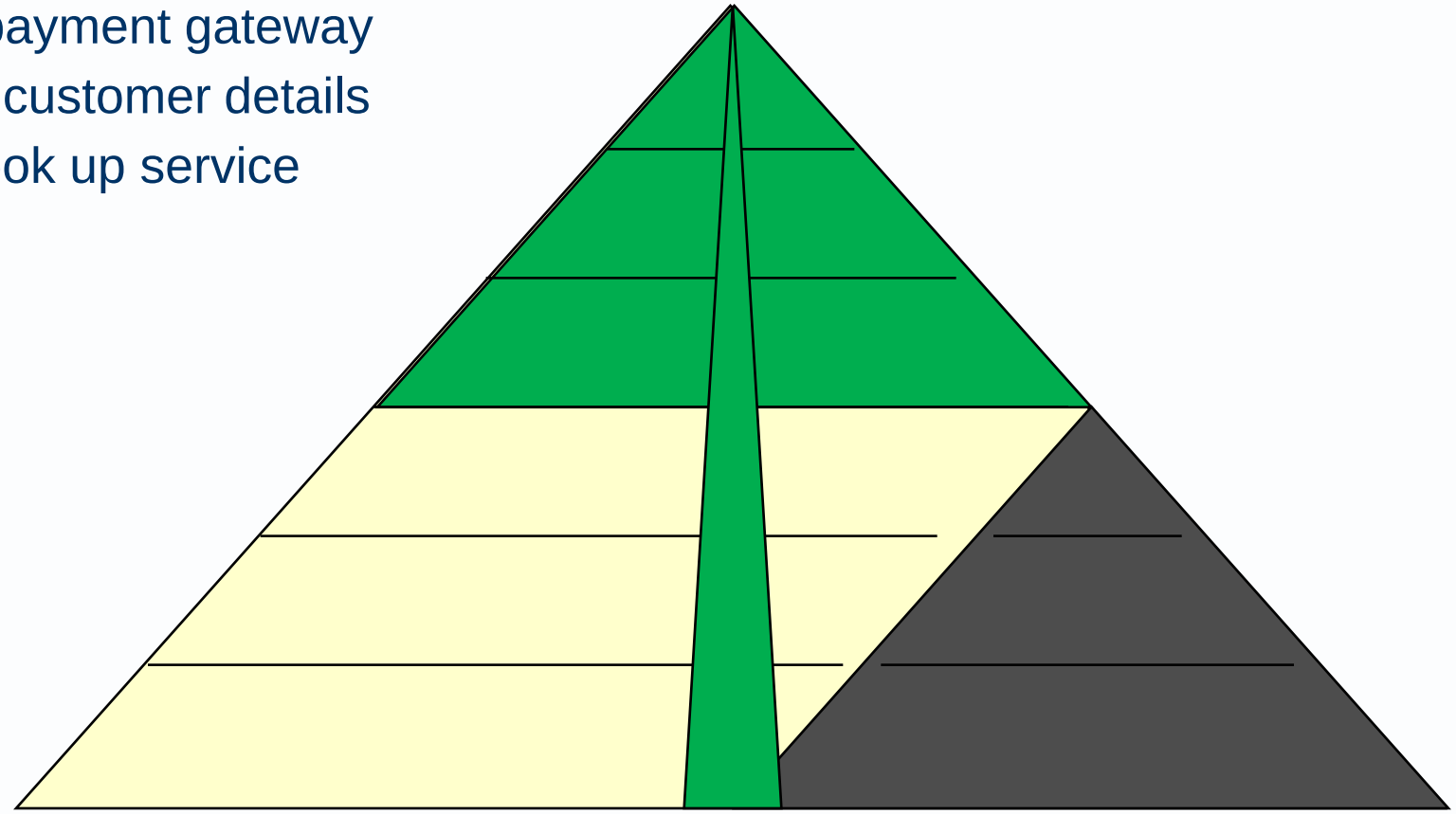
- **Class hierarchy**
- A hierarchy in which a generic description is incrementally specialised into successively more specific descriptions.



Generalisation - Common components and processes

Avancier

- E.g.
- Single sign on, across 1,000 applications
- In-bound payment gateway
- Canonical customer details
- Address look up service



Abstraction by idealisation (from reality to concept)

Avancier

- [A technique] that simplifies by hiding physical and/or supplier-specific details.
- The classic idealisation hierarchy “reverse engineers” from real to conceptual.
- **Conceptual (or domain) 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 that system’s physical implementation. It is supplier-independent and portable.
  - It may specify services or processes to be performed, and data or abilities needed.
  - It leaves open the choice of particular components, products and mechanisms.
- **Physical model**
  - a model that is supplier-specific or includes implementation details.
  - A description of particular products or mechanisms that can be employed or deployed to realise a logical model.

Elaboration by Forward Engineering

Abstraction by Reverse Engineering

How do you distinguish logical model from physical model?

Avancier

	Supplier dependence	Resource dependence	Encapsulation	Design
Logical HLD?	Supplier independent	Not dependent on a specific technology or material resource	Interfaces and service contracts that hide internal workings	Designed for simplicity and integrity
Physical LLD?	Supplier specific	Dependent on a specific technology or material resource	Internal processes and components	Designed for performance (speed and throughput)

What does a physical model add?

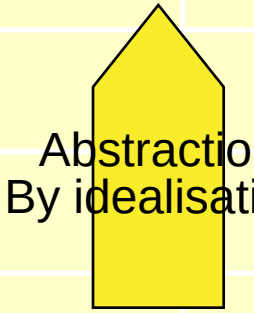
Avancier

Idealisation level	Data	Processes	Deployment
Conceptual regardless of computing	Business data model - describes business terms and facts	Business process models or use cases – describe workflows	Distribution of data processing
Logical specifies a computerised system – regardless of technologies	Logical data model - describes the content of a data store	Class diagram - describes logical modules and the operations they can perform State charts	Data flow diagram – shows application communications Logical deployment diagram – maps applications to logical nodes
Physical specific to a programming language, DBMS, OS or other platform technology	DB schema Indexes Sorting Clustering data entities into one block or page Next/prior/owner pointers Other features specific to a vendor's DBMS	Source code (Java, C++, …) Use of platform infrastructure (CICS, WebSphere…) Connection of distributed modules (via CORBA, DCOM, Web Services…) Etc.	Cache - for response time and throughput Load balancers and clustering - for availability Remote replication - for recoverability Firewalls - for security Server monitoring - for serviceability

Idealisation - in architecture frameworks

Avancier

Zachman Framework v2		Columns					
Rows - reification		What	How	Where	Who	When	Why
Idealisation-Reification	Stakeholders	Inventory sets	Process Transform'n	Network nodes	Organisation groups	Time periods	Motivation reasons
Scope Contexts	Strategists & theorists						
Business Concepts	Enterprise leaders & owners						
System Logic	Architects & designers						
Technology Physics	Engineers & builders						
Component assemblies	Technicians & implementers						
Operations Instance classes	Workers & participants						



Kinds of abstraction summary

Avancier

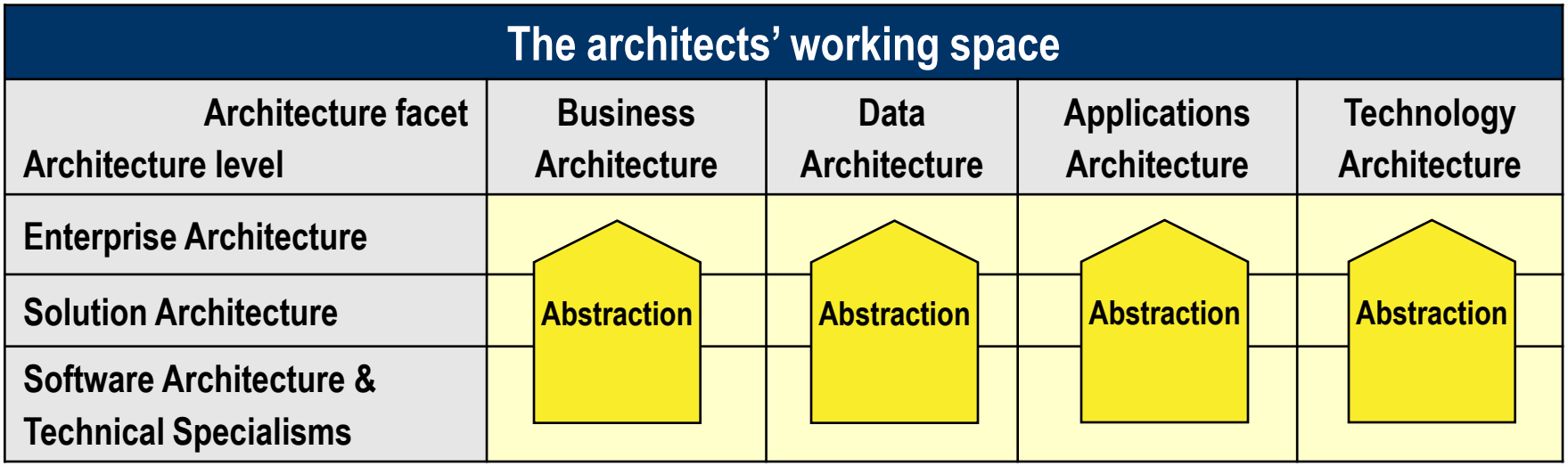
Omission leaving out details	**Composition** **packing smaller things inside bigger things**	**Generalisation** **removing differences between things**	**Idealisation** **removing differences between physical forms**
Vacuous	Coarse-grained composite	Universal	Concept
Sketchy	Mid-grained composite	Fairly generic	Logical Model
Elaborate	Fine-grained composite	Fairly specific	Physical Model
Complete	Elementary part	Uniquely configured	Physical Material
Elaboration	**Decomposition**	**Specialisation**	**Realisation**

EA tends to be more abstract in every possible way

- More generic - a higher level of generalisation
- More conceptual - a higher level of idealisation
- More coarse-grained - a higher level of granularity

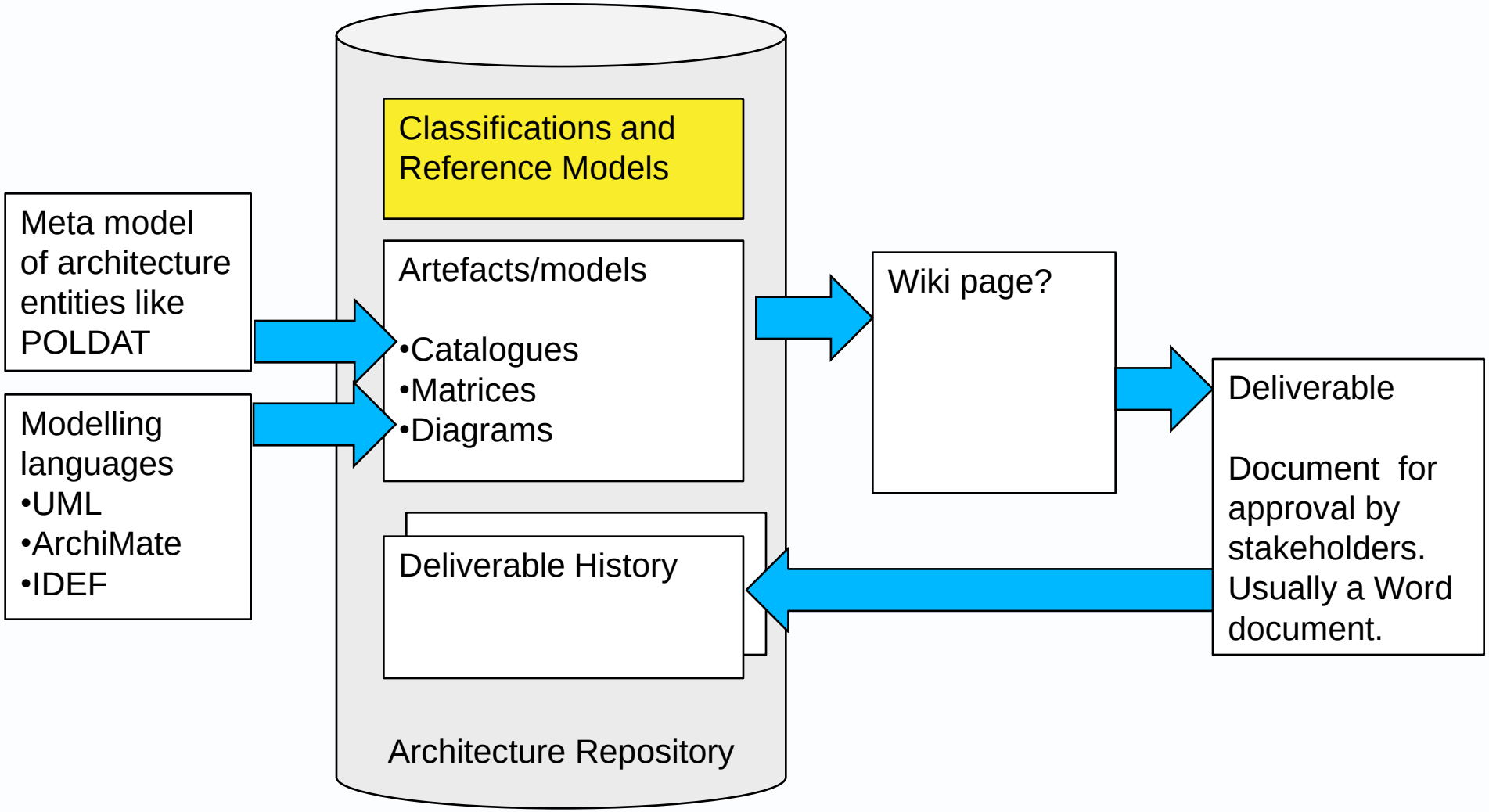
Who are you talking to?

Manager



Technician

3.5: Pre-defined classifications and reference models



3.5: Pre-defined classifications and reference models

- Documentation classification frameworks
  - Zachman framework
  - Enterprise continuum (TOGAF)
- Reference models

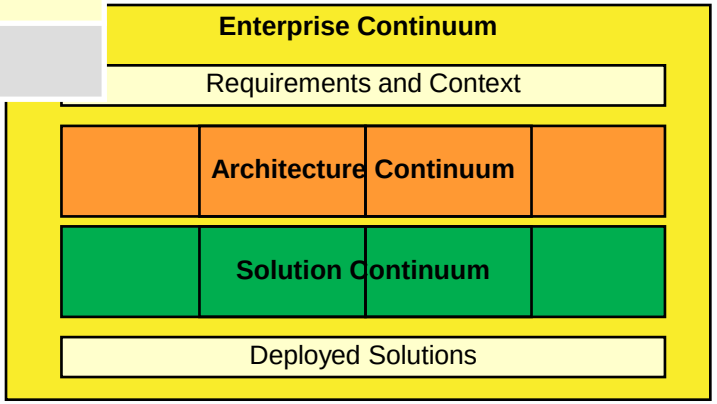
Documentation classification frameworks

Avancier

- **Nothing more or less than a set of pigeon holes for architecture description artefacts**

Zachman Framework	What	How	Where	Who	When	Why
Scope Contexts						
Business Concepts						
System Logic						
Technology Physics						
Tool components						
Operations – Instance classes						

TOGAF



- A *window* on to an architecture repository
- A *classification scheme* for reusable architecture assets

Zachman framework

Avancier

- [a pattern] “A logical structure for classifying and organising the descriptive representations of an Enterprise that are significant to managers and to developers of Enterprise systems.”

“Columns show “the primitive interrogatives”

Zachman Framework v3		What	How	Where	Who	When	Why
Level	Stakeholder perspective	Inventory sets	Process flows	Distribution networks	Responsibility assignments	Timing cycles	Motivation intentions
Scope Contexts	Executive						
Business Concepts	Business manag’t						
System Logic	Architect						
Technology Physics	Engineer						
Tool components	Technician						
Operations Instance classes	Enterprise						

“Rows show “reification - the transformation of an abstract idea into an instantiation… labeled
- Identification,
- Definition,
- Representation,
- Specification,
- Configuration &
- Instantiation.”

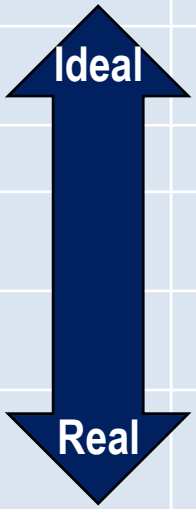
Remember the meanings of rows and columns

The Zachman Framework version 3

Avancier

- The 6 columns, though titled with interrogative questions, are mapped to architectural description elements.
- The 6 rows are levels of realisation from context to operational systems.
- But also mapped to stakeholder types and architecture domains or views.
- Zachman says the rows should not be interpreted as levels of decomposition.

Zachman Framework v3		What	How	Where	Who	When	Why
Level	Stakeholder perspective	Inventory sets	Process flows	Distribution networks	Responsibility assignments	Timing cycles	Motivation intentions
Scope Contexts	Executive						
Business Concepts	Business manag't						
System Logic	Architect						
Technology Physics	Engineer						
Tool components	Technician						
Operations Instance classes	Enterprise						



20011, essence of the Zachman Framework version 3

Avancier

Zachman Framework v3		What	How	Where	Who	When	Why
Idealisation	Stakeholder perspective	Inventory sets	Process flows	Distribution networks	Responsibility assignments	Timing cycles	Motivation intentions
Ideal to Real							
Scope Contexts	Executive	List inventory types	List process types	List distribution types	List responsibility types	List timing types	List motivation types
Business Concepts	Business management	Business entities & relationships	Business & input output	Business location & connection	Business role & work product	Business interval & moment	Business ends & means
System Logic	Architect	System entities & relationships	System & input output	System location & connection	System role & work product	System interval & moment	System ends & means
Technology Physics	Engineer	Technology entities & relationships	Technology input & output	Technology & location connection	Technology role & work product	Technology interval & moment	Technology ends & means
Tool components	Technician	Tool entities & relationships	Tool input & output	Tool location & connection	Tool role & work product	Tool interval & moment	Tool ends & means
Operations - Instance classes	Enterprise	Operations entities & relationships	Operations entities & relationships	Operations entities & relationships	Operations entities & relationships	Operations entities & relationships	Operations entities & relationships

Hmm... The ZF rows

Zachman Framework v3	
Idealisation	**Stakeholder perspective**
Scope Contexts	Executive
Business Concepts	Business management
System Logic	Architect
Technology Physics	Engineer
Tool components	Technician
Operations - Instance classes	Enterprise

- Conflate three different meanings
  - idealisation levels
  - stakeholders
  - architecture domains
- They unwisely force correspondences between them.
- People seem unclear whether the bottom row is
  - instances of things in real world system operation
  - instances of records of those things (as in CMDB?)

The ZF columns

What?	How?	Where?	Who?	When?	Why?
Inventory sets	Process flows	Distribution networks	Responsibility assignments	Timing cycles	Motivation intentions
Passive structure	**Process or behaviour**	**Position in space**	**Active structure**	**Position in time**	**Purpose or aim**

- People usually map what? to data.
- But Zachman changed his mind years and called it inventory.
- People seem unclear whether that means it is
  - objects in reality
  - data records describing objects in reality
  - data records describing objects and events in reality.
- Nobody I have in the last 20 years has used the ZF directly.

Enterprise continuum

Avancier

- [A pattern] a logical structure in TOGAF for classifying and organising architecture artifacts.
- It can be drawn as a table or grid.
- From top to bottom is ideal to real
- From left to right is general to specific.

Enterprise Continuum	Foundation	Common systems	Industry	Organisation
	Universal building blocks for system construction	Used in most business domains	E.g. Telecoms or Banking	Your unique business
Context and requirements				
Architecture continuum				
Solution continuum				
Deployed solutions				

The core of the enterprise continuum

Avancier

Generic to Specific

		Generic, horizontal, infrastructure building blocks and services	Patterns or structures of components and services	Business domain (Retail, Banking, Telecoms)	Enterprise (Tesco, HBOS, Orange)
		Foundation (Universal)	**Common Systems** (Fairly generic)	**Industry** (Fairly specific)	**Organisation** (Uniquely configured)
Ideal to Real	**Requirements and Context**				
	Architecture Continuum (Logical Models)	e.g. Open standards	e.g. application integration patterns.	e.g. Function, Data and Process models	e.g. Bespoke application use cases and data models
	Solution Continuum (Physical Models)	Strategic products Operating systems	Product assemblies (Email system, Security system)	e.g. COTS Packages	Bespoke solutions
	Deployed solutions				

Reference Models in the enterprise continuum

Avancier

Generic to Specific

		Generic, horizontal, infrastructure building blocks and services	Patterns or structures of components and services	Business domain (Retail, Banking, Telecoms)	Enterprise (Tesco, HBOS, Orange)
		Foundation (Universal)	**Common Systems** (Fairly generic)	**Industry** (Fairly specific)	**Organisation** (Uniquely configured)
Ideal to Real	**Requirements and Context**				
	Architecture Continuum (Logical Models)	**TRM** A hierarchical catalogue of platform services	**III-RM** An SOA design pattern for apps architecture.	**BIAN - banking TMF - telecoms ProAct - retail**	**Bespoke application use cases, data models…**
	Solution Continuum (Physical Models)				
	Deployed solutions				

Reference model

- [a pattern] a generic structure or classification used to create more specific models.
- It can be a structure of components, processes or data elements.
- It is sometimes applicable to a particular industry or business domain.
- It can act as a design pattern.

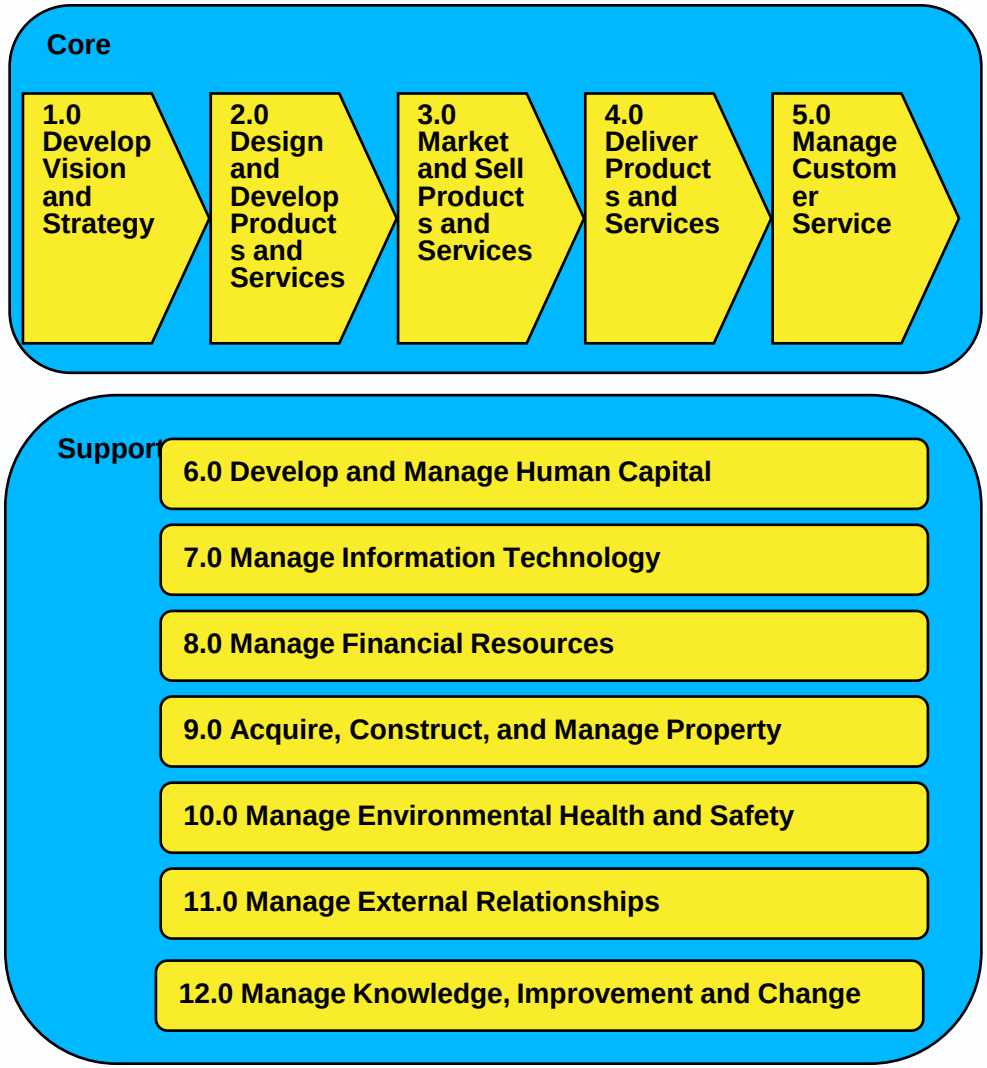
- E.g.
  - **APQC for a generic commercial organisation**
  - BIAN for banking (one of many banking reference models)
  - TMF for telecoms
    - eTOM – Business Architecture
    - SID – Data Architecture
    - TAM – Applications Architecture
  - SCOR for supply-chain businesses
  - ProAct for retailers
  - FEA for US federal government
  - A long list of industry-specific canonical data model

APQC process classification framework.

Avancier

This standard hierarchical classification of the functions in a commercial enterprise can provide you with a means to

- Structure baseline activities
- Identify and structure required activities.



APQC updated and limited to 3 levels

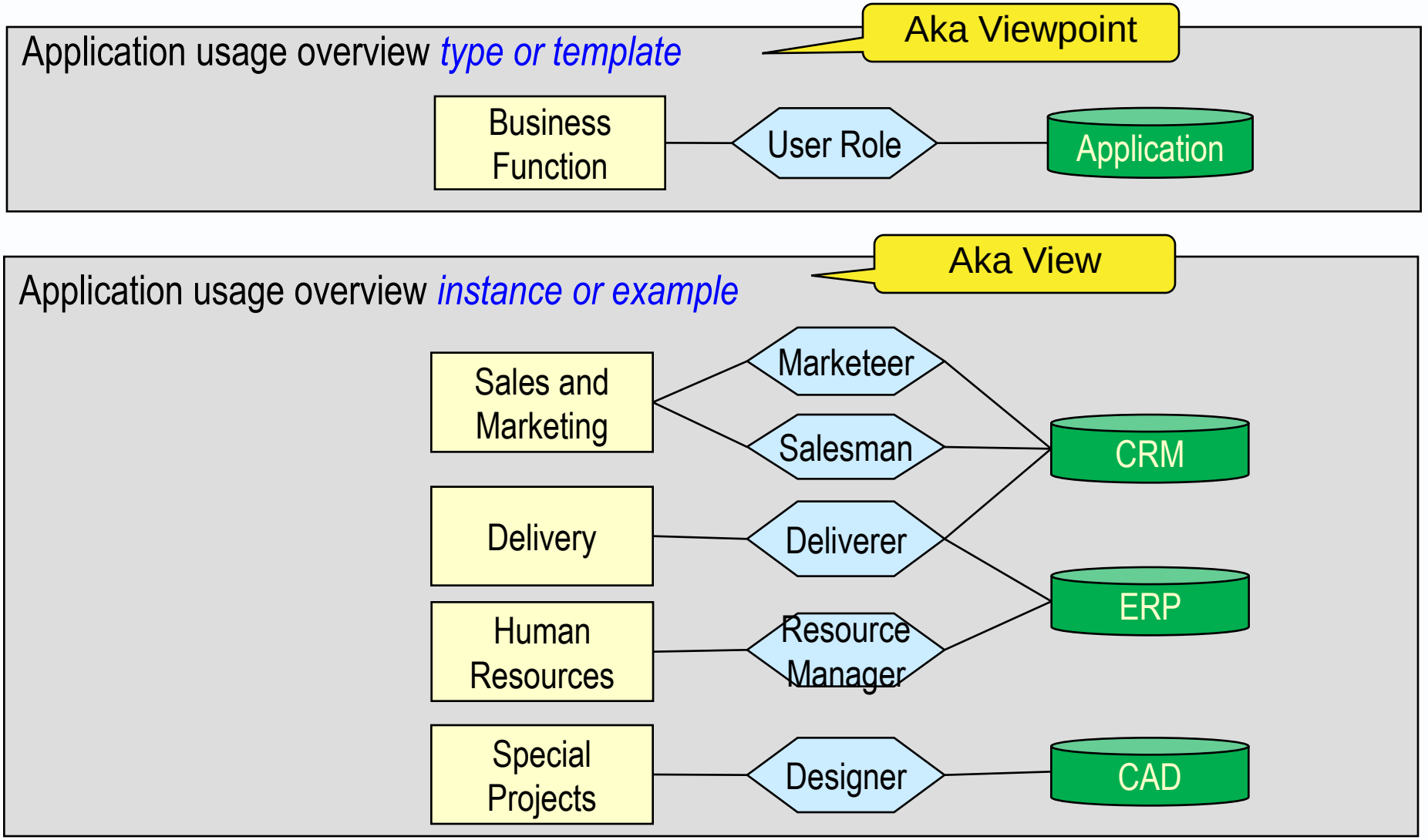
Avancier

- 1. UNDERSTAND MARKETS AND CUSTOMERS
  - 1.1 Determine customer needs and wants
    - 1.1.1 Conduct qualitative assessments
      - 1.1.1.1 Conduct customer interviews
      - 1.1.1.2 Conduct focus groups
    - 1.1.2 Conduct quantitative assessments
      - 1.1.2.1 Develop and implement surveys
    - 1.1.3 Predict customer purchasing behavior
  - 1.2 Measure customer satisfaction
    - 1.2.1 Monitor satisfaction with products and services
    - 1.2.2 Monitor satisfaction with complair
    - 1.2.3 Monitor satisfaction with commun
  - 1.3 Monitor changes in market or customer expecta
    - 1.3.1 Determine weaknesses of produc
    - 1.3.2 Identify new innovations that mee
    - 1.3.3 Determine customer reactions to
- 2. DEVELOP VISION AND STRATEGY
  - 2.1 Monitor the external environment
    - 2.1.1 Analyze and understand competition
    - 2.1.2 Identify economic trends
    - 2.1.3 Identify political and regulatory issues
    - 2.1.4 Assess new technology innovations
    - 2.1.5 Understand demographics
    - 2.1.6 Identify social and cultural changes
    - 2.1.7 Understand ecological concerns
  - 2.2 Define the business concept and organizational strategy
    - 2.2.1 Select relevant markets
    - 2.2.2 Develop long-term vision
    - 2.2.3 Formulate business unit strategy
    - 2.2.4 Develop overall mission statement
  - 2.3 Design the organizational structure and relationships between organizational units
  - 2.4 Develop and set organizational goals

More on ISO 42010

Architecture Frameworks (ESA 3)

Diagram types and instances



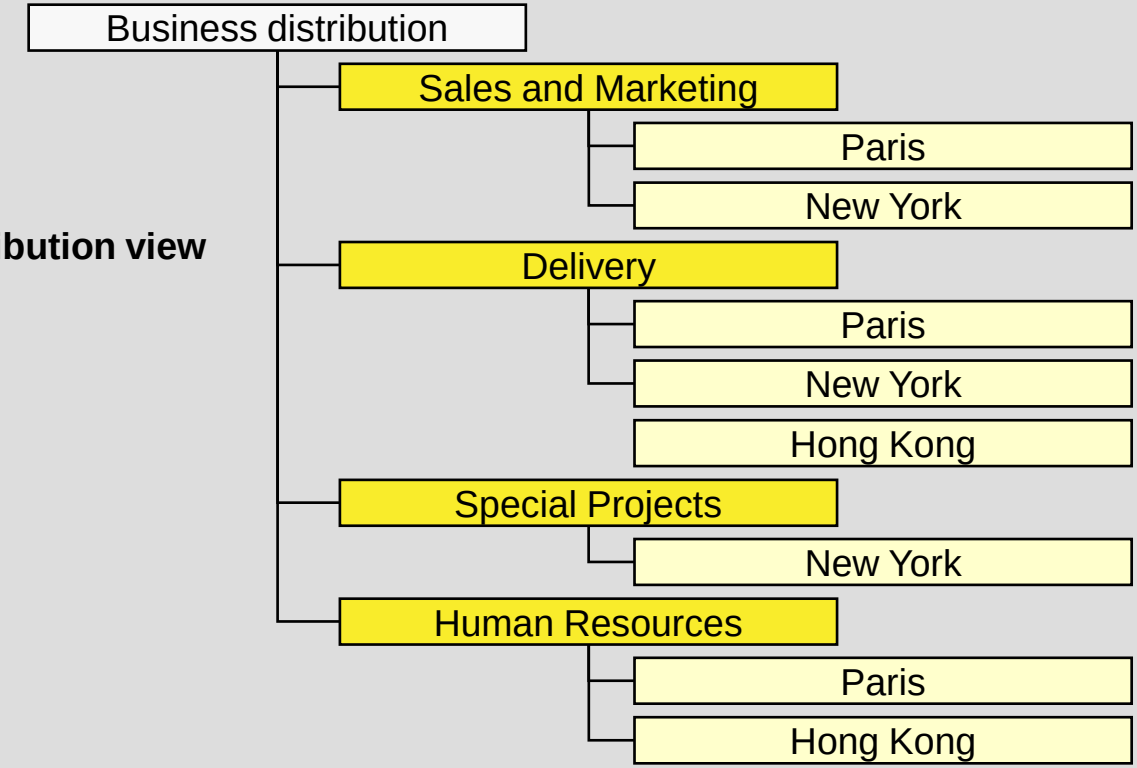
Viewpoint, view and architecture model (1)

Viewpoint

Viewpoint	Addresses these concerns	Of these stakeholders	Using this kind of model
Business distribution	Where business is done	CEO COO	Hierarchical tree showing locations of functions

View/Model

A business distribution view



Template or type

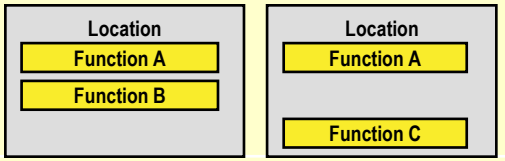
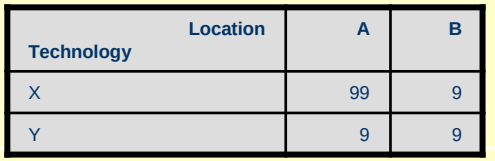
Example or instance

Viewpoints and Views

Avancier

- Any group of architects should share ways to address concerns

Viewpoint library

What	Why	Who	How
Viewpoint	**Addresses these concerns**	**Of these stakeholders**	**Using this kind of model**
Business distribution	Where we do our business	CEO COO	Hierarchical tree showing locations of functions
Location activity	What we do at each location	Facilities Manager	
Deployment overview	Where our kit is	IT Services Manager	
Data model	Data structure Data item types Potential access paths	Systems analysts Database designers Domain experts	IDEF1X standard
Application usage	Which applications used by which roles in which business functions	Application portfolio manager Systems analysts	In-house template

Template or type

Example or instance

Viewpoints and Views

Avancier

- Any group of architects should share ways to address concerns

Viewpoint library

What	Why	Who	How
Viewpoint	Addresses these concerns	Of these stakeholders	Using this kind of model
Networks	Bandwidth, protocols, resilience, performance	Network architect/consultant	Communication engineering diagram
Monitoring	SNMP, performance, probes, instrumentation	Operational monitor	Enterprise management diagram
Storage and back up	Data storage, movement, availability, recovery, replication	Storage architect/consultant	

ISO 42010: **Reuse of view points**

Avancier

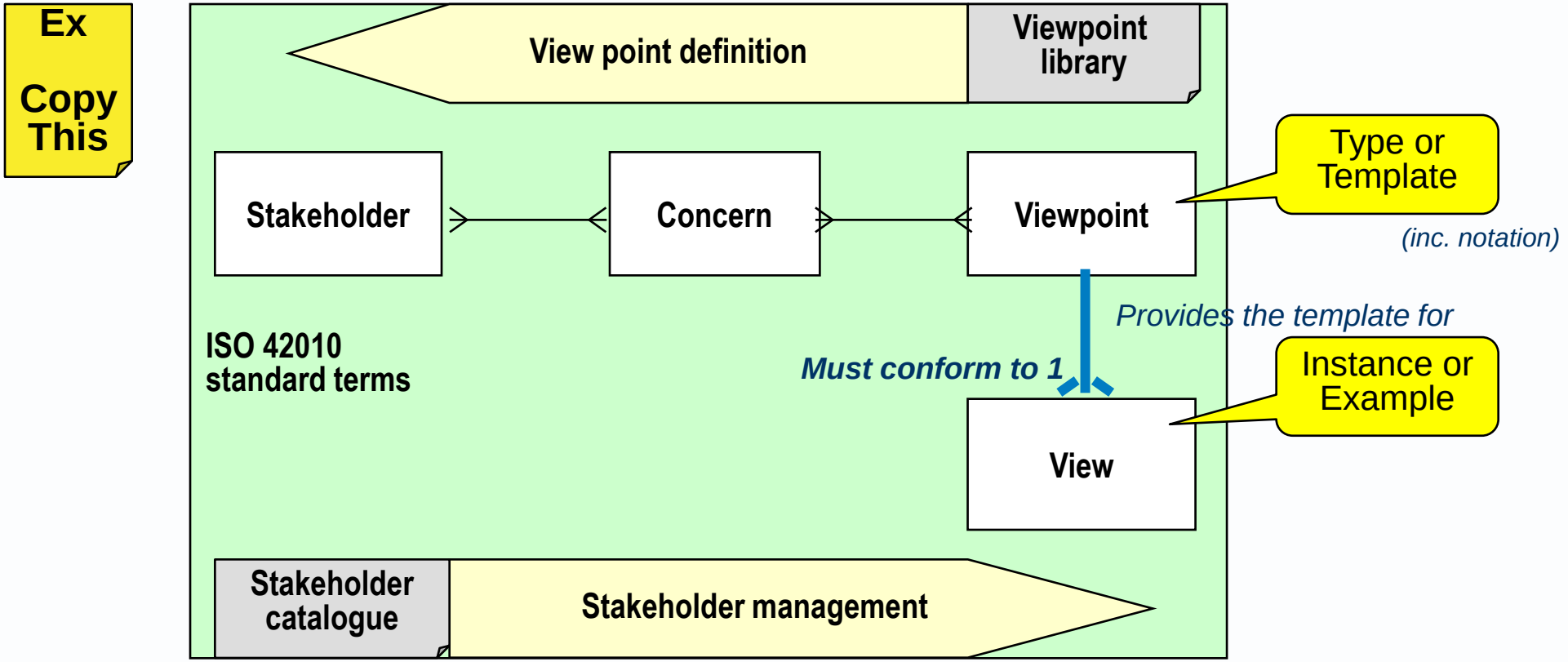
- The standard encourages you to define generic viewpoints and store them in libraries for re-use in different architectures.

Benefits:

- Less work for the architects
  - (because the viewpoints have already been defined and therefore the views can be created faster)
- Better comprehensibility for stakeholders
  - (because the viewpoints are already familiar)
- Greater confidence in the validity of the views
  - (because the viewpoints have a known track record)

The simple version implied by BCS reference model

- BCS exam may test your understanding of the relationships between these four essential concepts in ISO 42010 (aka ANSI 1471)



Imagine a spread sheet you can sort on any column/field

Avancier

Stakeholder catalogue

Sort on

Stakeholder	Concern	Viewpoint
CEO	Will the systems support business goals?	Business footprint diagram
Domain expert	How will the application support activities?	Use case definition
	What data can users query?	Logical data model
DBA	How to create the database schema?	Logical data model
Ops manager	What infrastructure is needed to run the apps?	Hardware configuration diagram

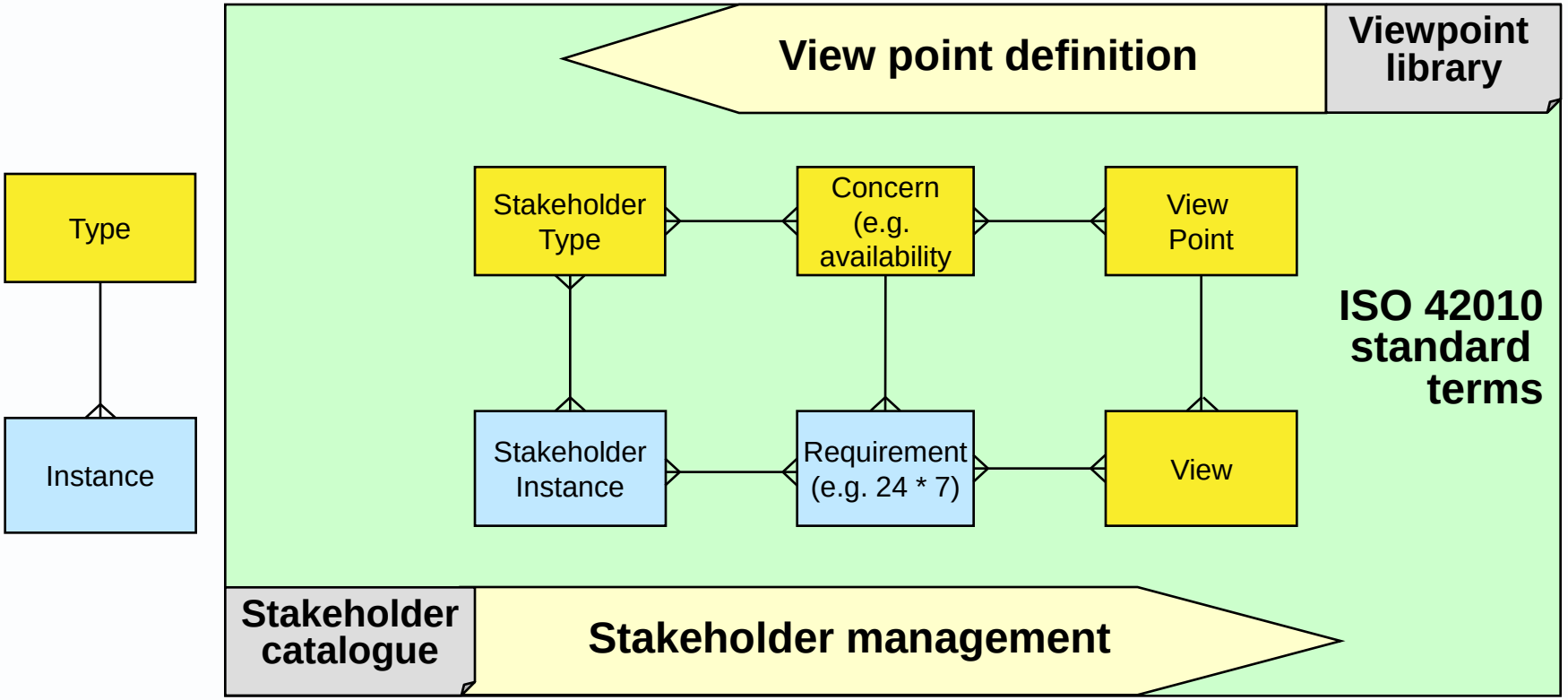
Viewpoint library

Sort on

Viewpoint	Concern	Stakeholder
Business footprint diagram	Will the systems support business goals?	CEO
Use case definition	How will the application support activities?	Domain expert
Logical data model	What data can users query?	Domain expert
	How to create the database schema?	DBA
Hardware configuration diagram	What infrastructure is needed to run the apps?	Ops manager

The simple version implied by BCS reference model

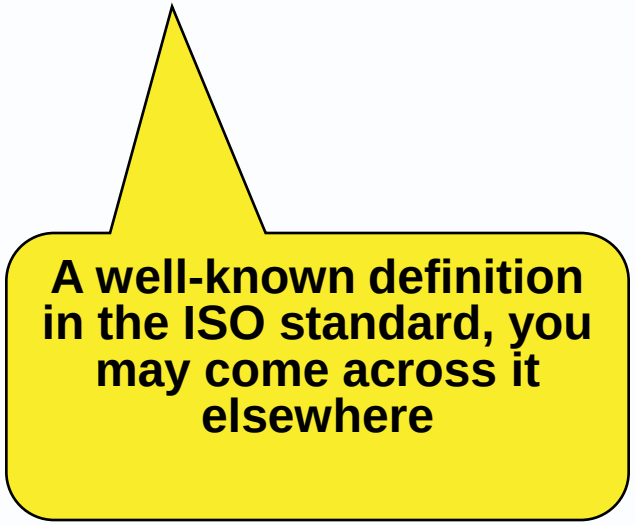
- BCS exam may test your understanding of the relationships between these four essential concepts in ISO 42010 (aka ANSI 1471)



ISO 42010: **"The** *architecture* **of a system**

Avancier

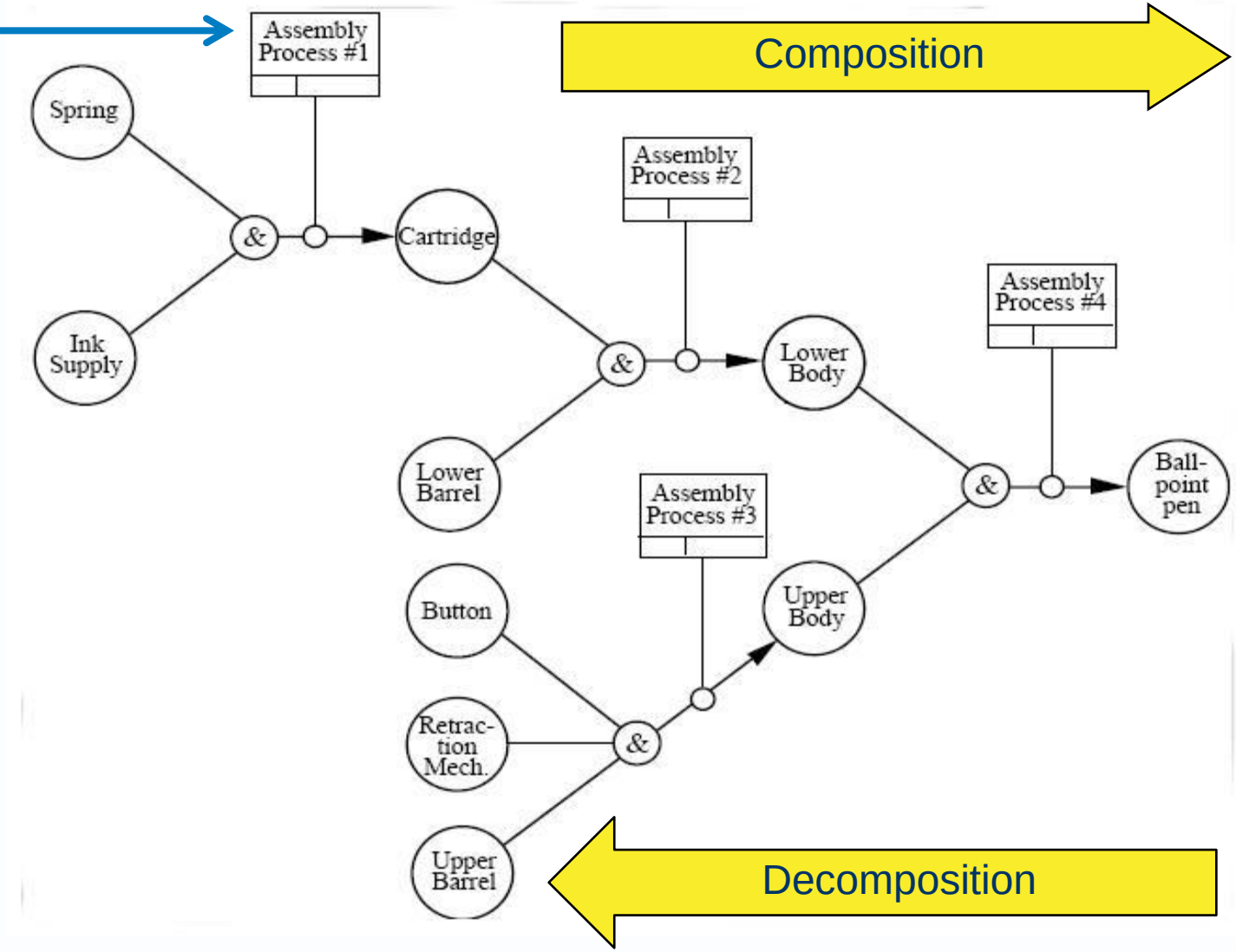
"fundamental concepts or properties of a system
in its environment embodied in
its **elements**, relationships, and
in the principles of its design and evolution"



- Note: the elements include components and processes

An obscure IDEF 5 notation for composition

- Showing processes that assemble parts into a whole



A vastly simplified meta model of UML entities

- Key entities in UML diagrams