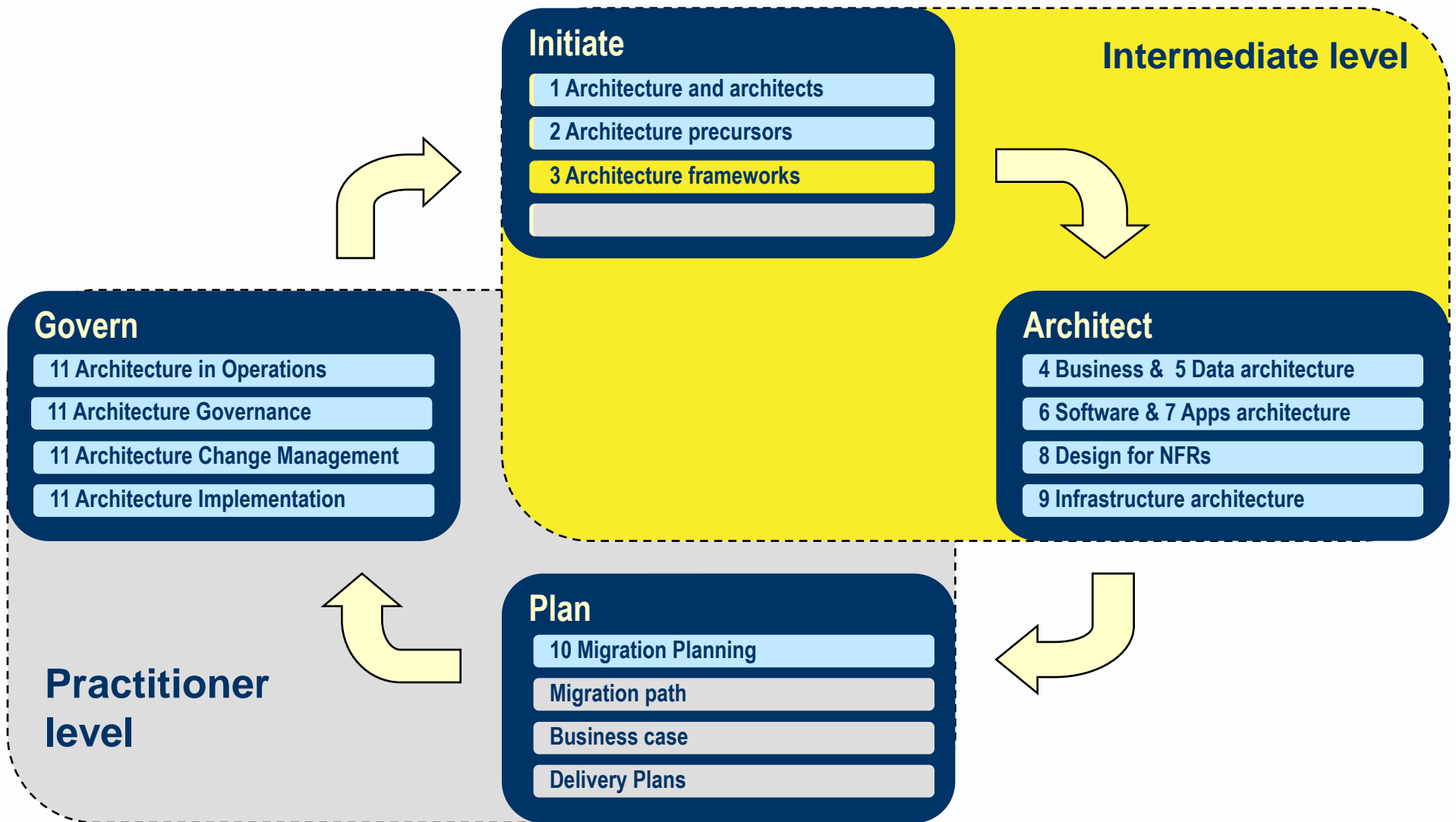


Avancier Reference Model

Architecture Frameworks (ESA 3)

It is illegal to copy, share or show this document
(or other document published at <http://avancier.co.uk>)
without the written permission of the copyright holder

Mapping the reference model to an architecture framework



3.1 Architecture frameworks (not to be examined)

Architecture framework

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

- ▶ [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.

State	Baseline	Gap analysis reveals changes and work to be done		Target
Domain				
Business	Process Organisation Locations			Process Organisation Locations
Information Systems	Data Applications			Data Applications
Technology	Infrastructure Technologies			Infrastructure Technologies

- ▶ [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.

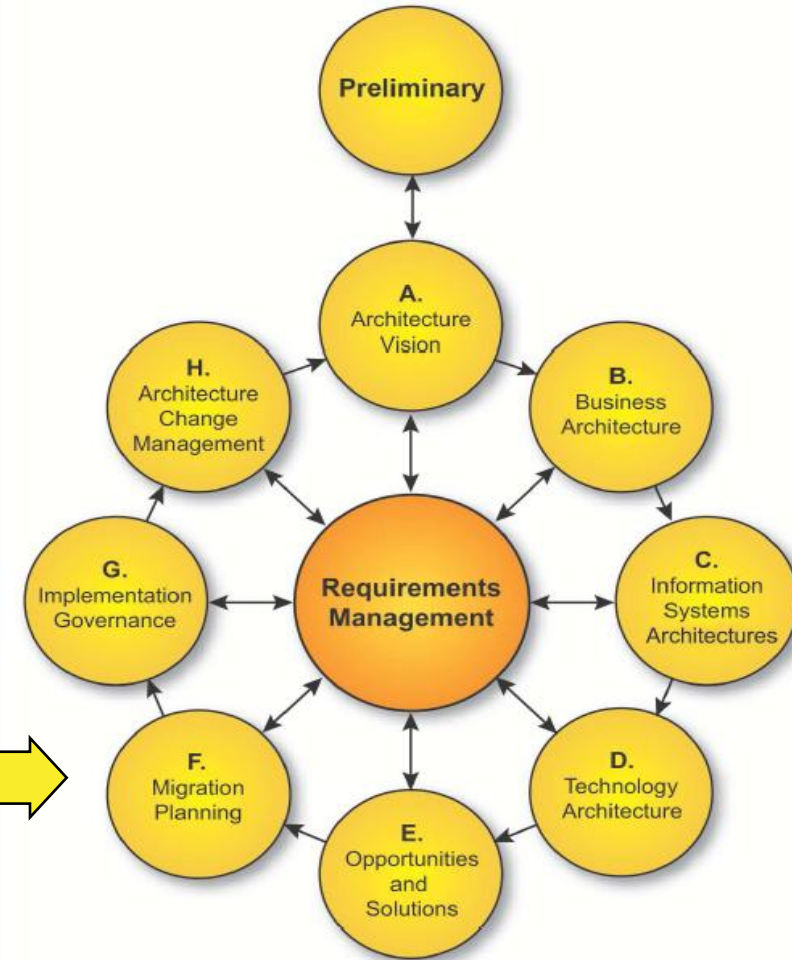
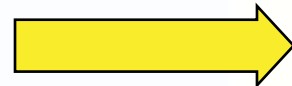


Figure 5-1 Architecture Development Cycle

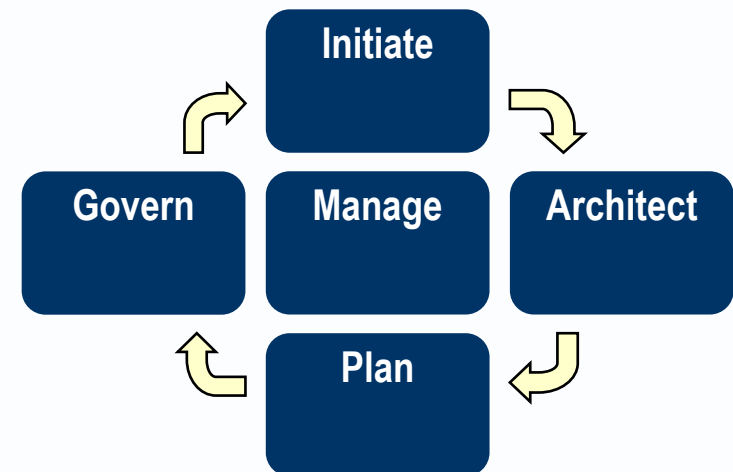
- ▶ [The process] defined in TOGAF to develop and use an enterprise architecture.
- ▶ Users are expected to adapt the process and iterate through it.
- ▶ It is centered on a cycle of 8 phases.
 - A: Architecture Vision
 - B: Business Architecture
 - C: Information System Architecture (Data and Applications)
 - 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**

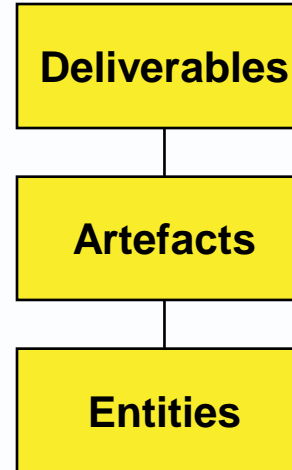
- ▶ [an architecture framework] focused on solution architecture, though it also addresses enterprise architecture rationalisation.
- ▶ It is published on a free-to-read public web site, though its use by an organisation is limited by copyright conditions.
- ▶ It features a solution architecture process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



3.3 Architecture products

▶ Architecture content framework

- Architecture deliverable
- Architecture artefact
- Architecture entity

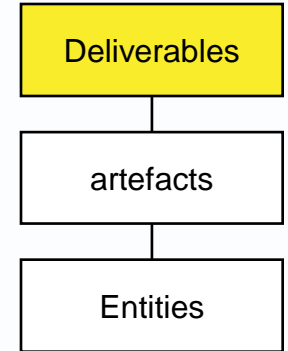


▶ Architecture repository

- ▶ Mapping
- ▶ View
- ▶ Viewpoint
- ▶ Model

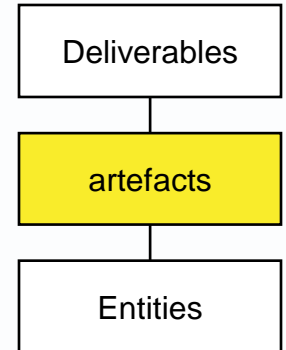
- ▶ [a passive structure] for organising an architecture description composed of deliverables, artefacts and entities.

- ▶ [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.



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

▶ Catalogue



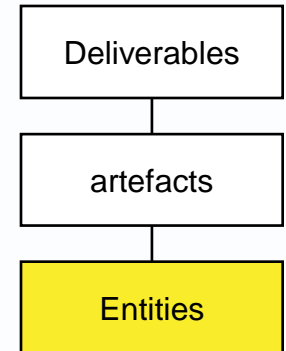
▶ Matrix

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

▶ Diagram.

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

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



How architecture description develops

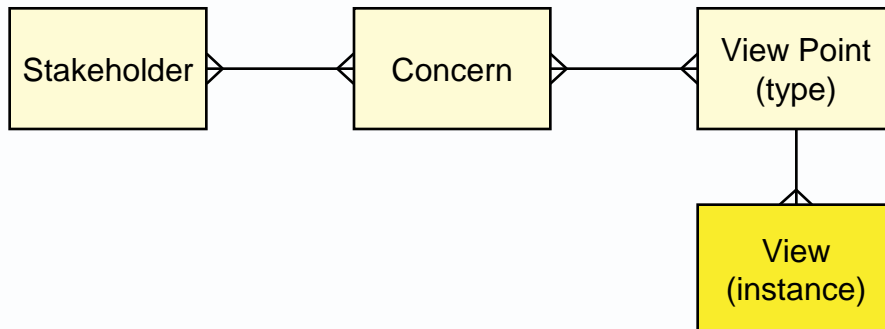
- ▶ You start off writing a deliverable as one document
- ▶ You write sections covering business, apps and technology concerns, mentioning entities such human roles, business processes, apps and technologies
- ▶ You insert various artefacts (tables and diagrams) to show the relationships between the Entities
- ▶ The artefacts refer to the entities by name
- ▶ You describe the entities more fully in catalogues in appendices
- ▶ You divide the document into documents for different stakeholders with different concerns
- ▶ 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.

- ▶ [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.

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

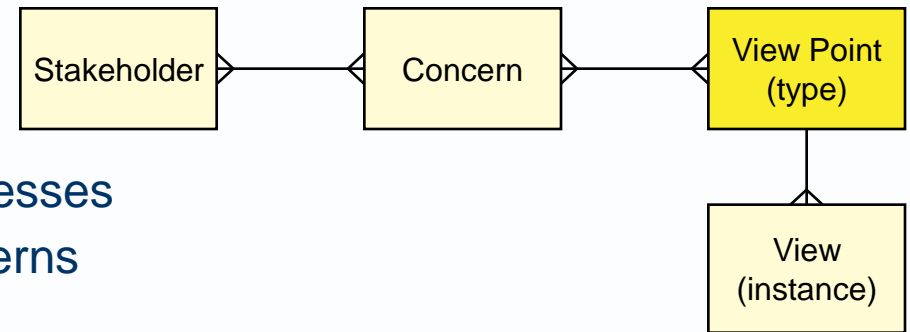
	Loc	Loc	Loc
Org	Works at		
Org		Works at	Works at
Org	Works at	Works at	

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

- ▶ [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.



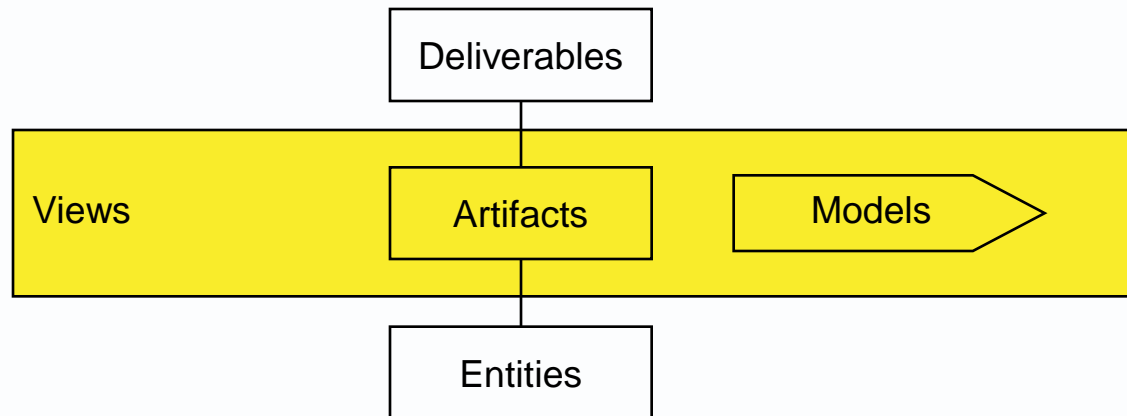
3.4 Architecture models

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

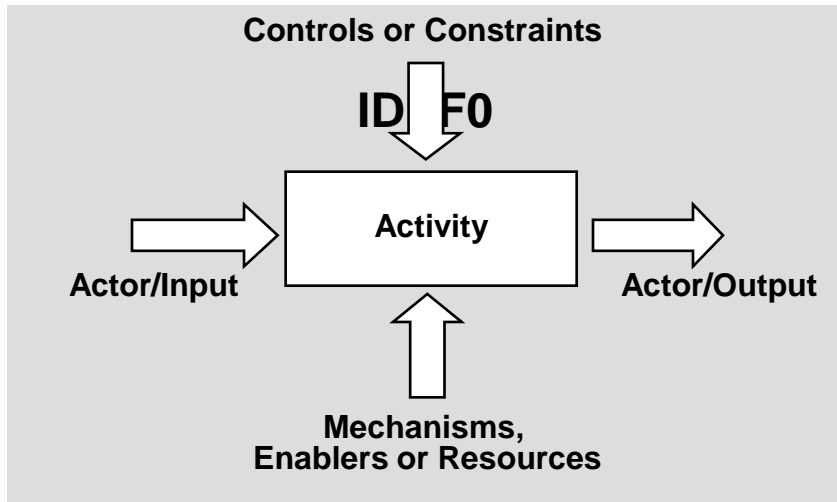
- ▶ [A work product description] that typifies a model and provides a template for building a model.



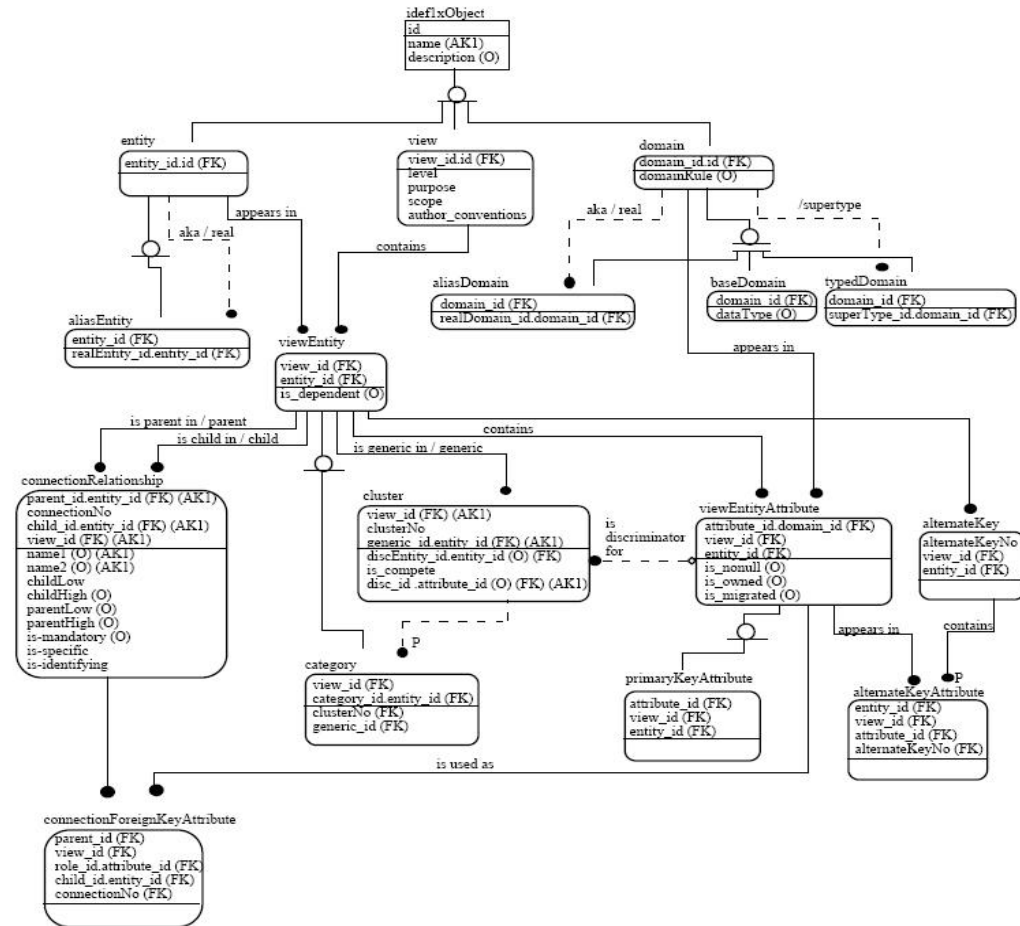
- [A standard] that defines shapes for representing architecture entities and arc/line styles for representing relationships between them.
- Three international varieties are IDEF, UML and ArchiMate.
- ▶ **IDEF: Integration DEFinition language**
 - [A modelling language] for systems and software engineering, originally funded by the US DoD.
 - It 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.

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

▶ IDEF 0 Function Models



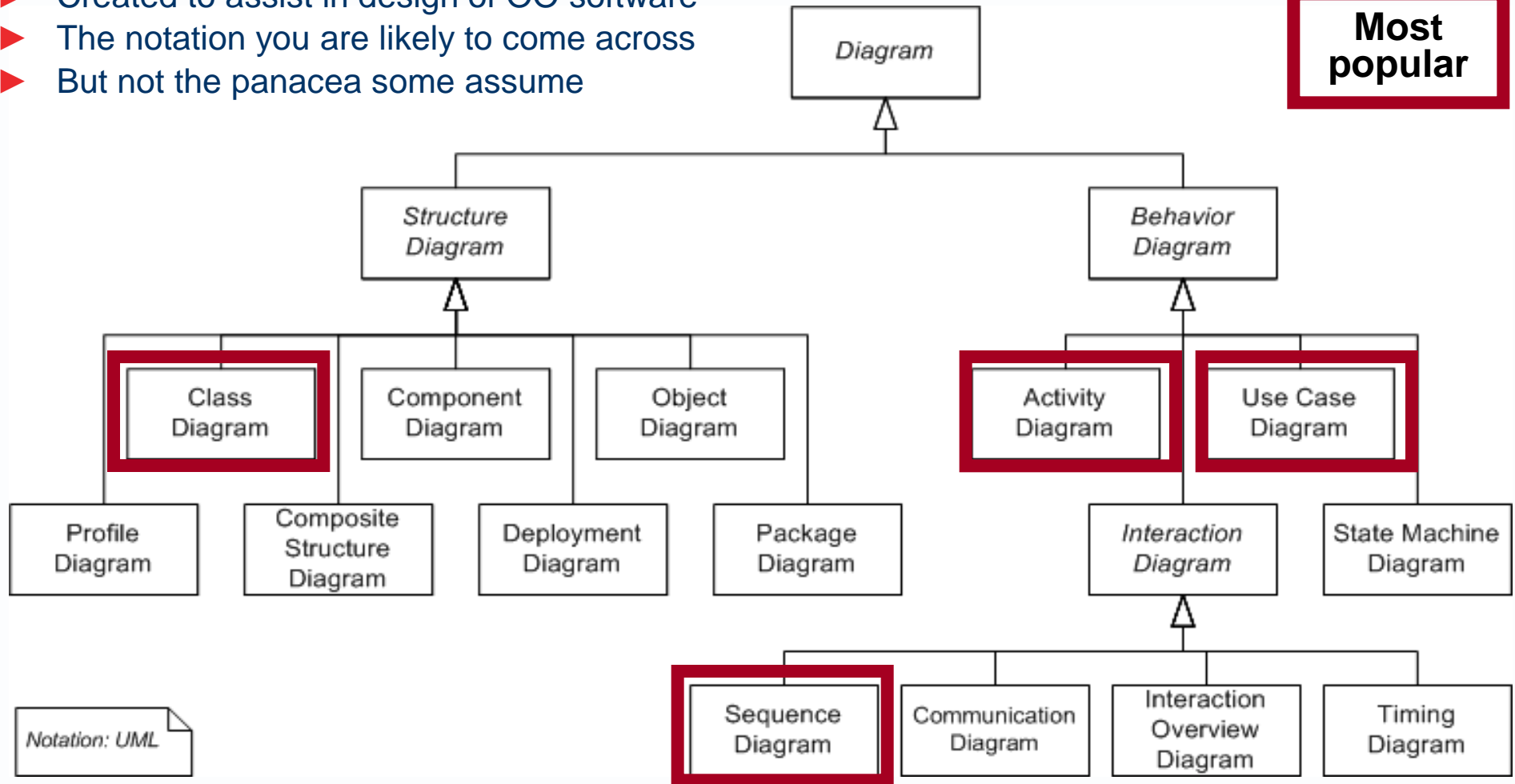
▶ IDEF 1X Data Models (Wikipedia)



UML (OMG standard)

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

Most popular



- ▶ [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 model-driven engineering technique] 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 infrastructure technology.

MDE: Model-Driven Engineering

- ▶ [a technique] used in methods and tools for transforming a conceptual model to a logical model, and a logical model to a physical model, and the reverse.
- ▶ It covers forward engineering and reverse engineering.
- ▶ For example, Model-Driven Architecture.

	Model-driven database design	Model-driven software design	
Elaboration by Forward Engineering	Business data model	Business process model	Abstraction by Reverse Engineering
	Logical data model	Platform-independent model	
	Physical data model	Platform-specific model	

- ▶ [a model-driven engineering technique] a vision of the Object Management Group (OMG) that encourages vendors to develop tools for Model-Driven Engineering to standards defined by the OMG.

- ▶ The idealisation hierarchy is:

**Elaboration
by
Forward
Engineering**

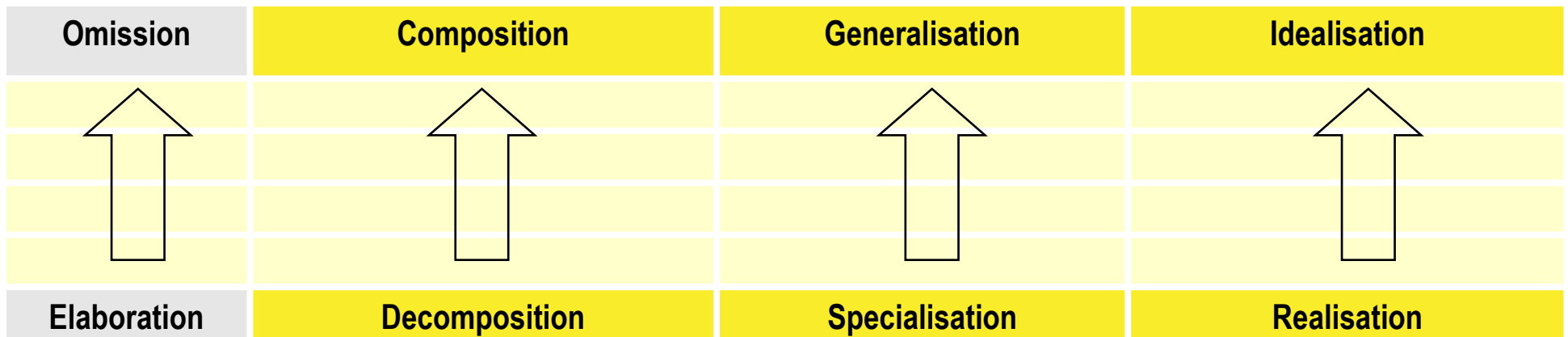
computation-independent model (CIM),
platform-independent model (PIM), unrelated to a specific technology
platform-specific model (PSM), related to specific infrastructure
technology.

**Abstraction
by
Reverse
Engineering**

CIM	Requirements in the form of business process and and data models, regardless of computing
PIM	Defines a system's functions, typically a UML class diagram , 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

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.



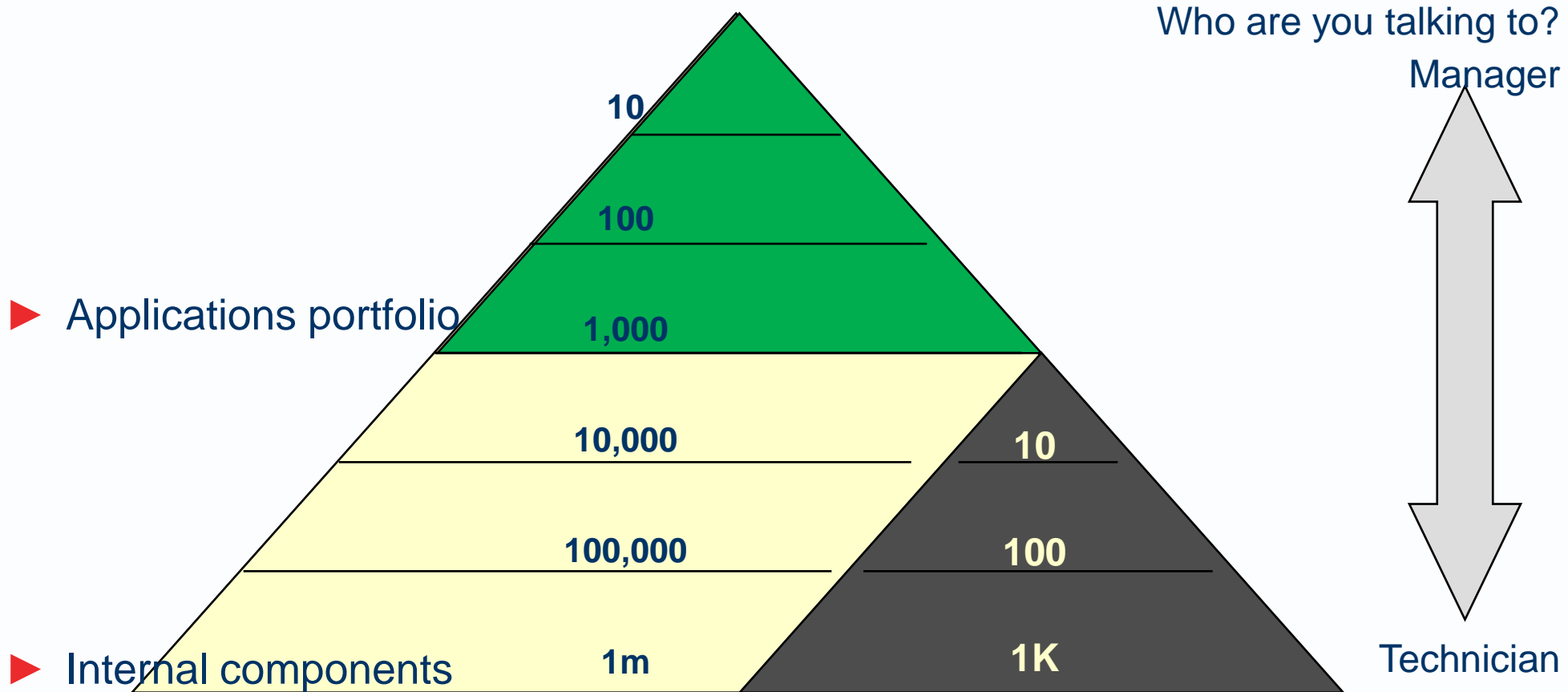
Abstraction by delegation (from server to client)

- ▶ [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.
- ▶ **Layered client-server hierarchy**
- ▶ See section 1.

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.
- ▶ **Hierarchical decomposition**
- ▶ See section 1.

Composition – Coarse-grained views and models

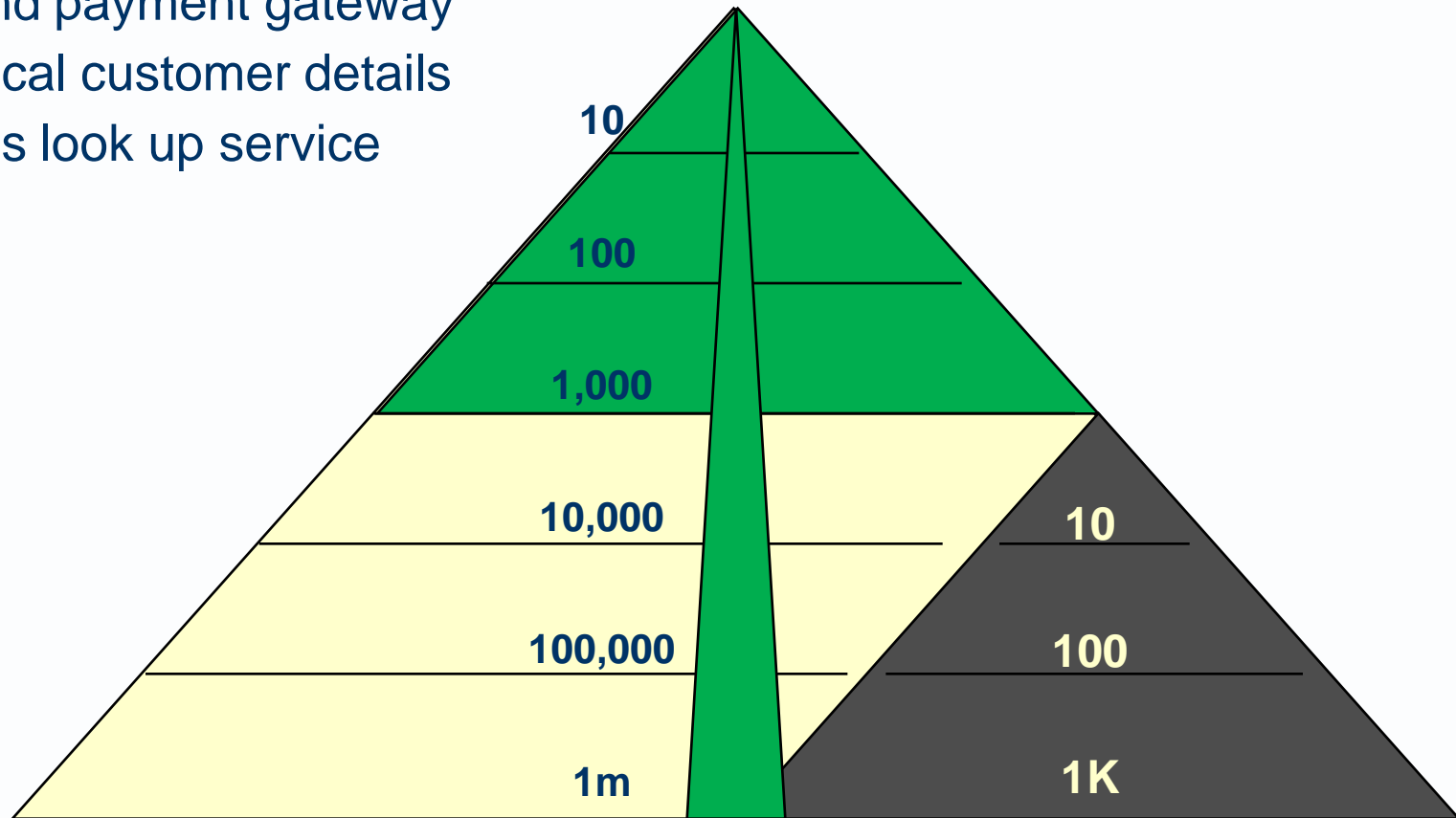


Abstraction by generalisation (from unique to universal)

- ▶ [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

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



- ▶ [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?

	Supplier dependence	Resource dependence	Encapsulation	Design
Logical 	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 	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?




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

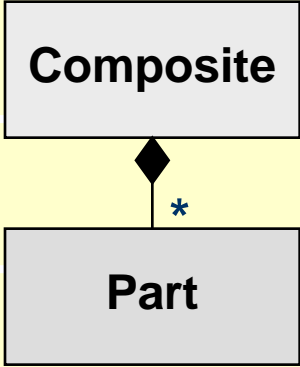
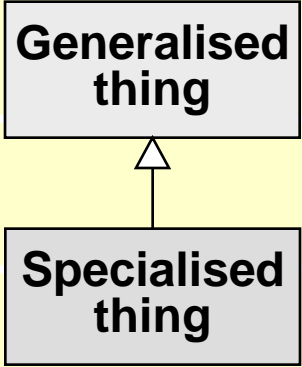
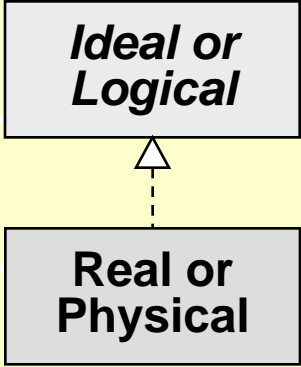
Idealisation - in architecture frameworks

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						

Abstraction
By idealisation



EA is more *abstract* than SA





Omission	Composition packing smaller things inside bigger things	Generalisation removing differences of detail	Idealisation removing differences between physical forms
	Coarse-grained views and models	Common components and processes	Conceptual/logical views and models
			
Elaboration	Decomposition	Specialisation	Realisation

Kinds of abstraction summary

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

The architects' working space

- ▶ 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

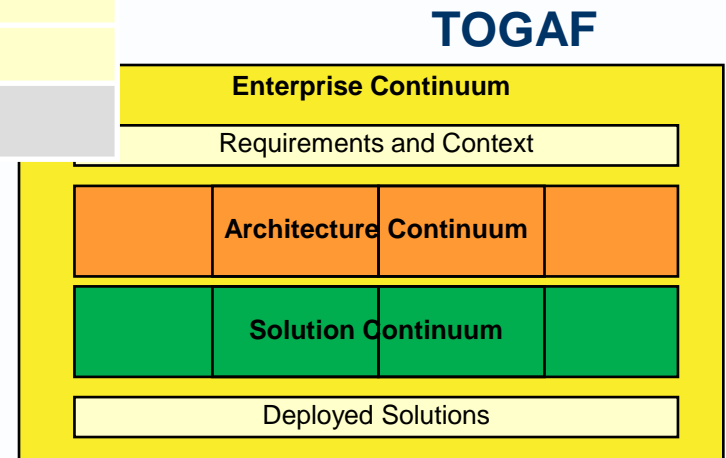
The architects' working space				
Architecture facet	Business Architecture	Data Architecture	Applications Architecture	Technology Architecture
Enterprise Architecture				
Solution Architecture				
Software Architecture & Technical Specialisms				

3.5: Pre-defined classifications and reference models

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

- ▶ 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						



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

- ▶ [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.”

- ▶ 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.

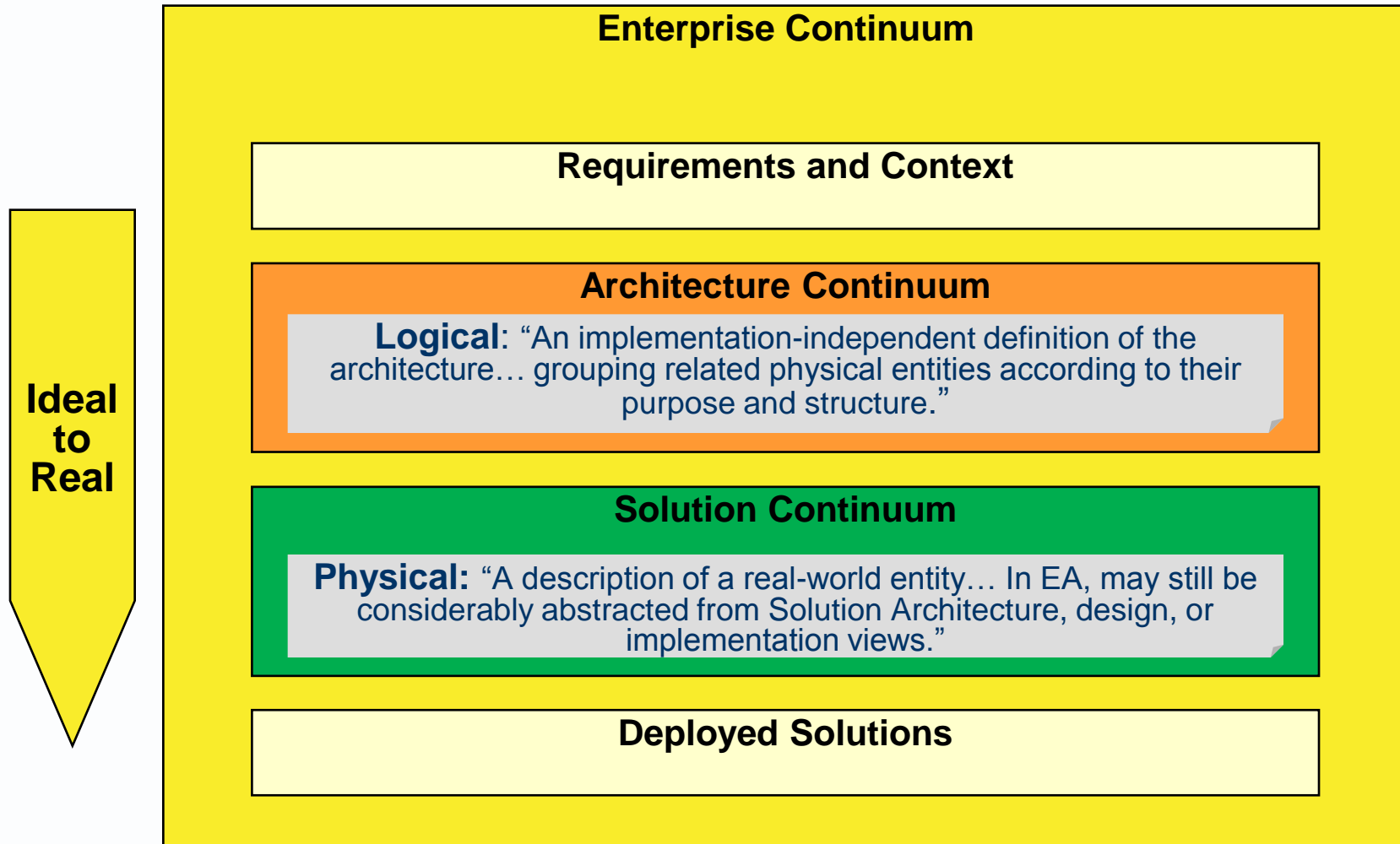
Zachman Framework v3		What	How	Where	Who	When	Why
Level	Stakeholder perspective	Inventory sets	Process flows	Distribution networks	Responsibility assignments	Timing cycles	Motivation intentions
Scope Contexts	Executive						
Business Concepts	Business manag't						
System Logic	Architect						
Technology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

20011, essence of the Zachman Framework version 3

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	Executive	List inventory types	List process types	List distribution types	List responsibility types	List timing types	List motivation types
	Business management	Business entities & relationships	Business & input output	Business location & connection	Business role & work product	Business interval & moment	Business ends & means
	Architect	System entities & relationships	System & input output	System location & connection	System role & work product	System interval & moment	System ends & means
	Engineer	Technology entities & relationships	Technology input & output	Technology & location connection	Technology role & work product	Technology interval & moment	Technology ends & means
	Technician	Tool entities & relationships	Tool input & output	Tool location & connection	Tool role & work product	Tool interval & moment	Tool ends & means
	Enterprise	Operations entities & relationships	Operations entities & relationships	Operations entities & relationships	Operations entities & relationships	Operations entities & relationships	Operations entities & relationships

- ▶ [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

Gen-Spec

	Enterprise Continuum	Foundation Generic, horizontal, infrastructure building blocks and services	Common System Patterns or structures for assembling building blocks and services	Industry Vertical business domain (Retail, Banking, Telecoms)	Organisation Enterprise-specific (Tesco, HBOS, Orange)
Architecture Continuum	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	Strategic products Operating systems	Product assemblies (Email system, Security system)	e.g. COTS Packages	Bespoke solutions	

Ideal to Real

TOGAF's two Reference Models

Gen-Spec

	Enterprise Continuum	Foundation Generic, horizontal, infrastructure building blocks and services	Common System Patterns or structures for assembling building blocks and services	Industry Vertical business domain (Retail, Banking, Telecoms)	Organisation Enterprise-specific (Tesco, HBOS, Orange)
Architecture Continuum		TRM A hierarchical list of platform services	III RM An SOA design pattern for apps architecture		
Solution Continuum					

Ideal to Real

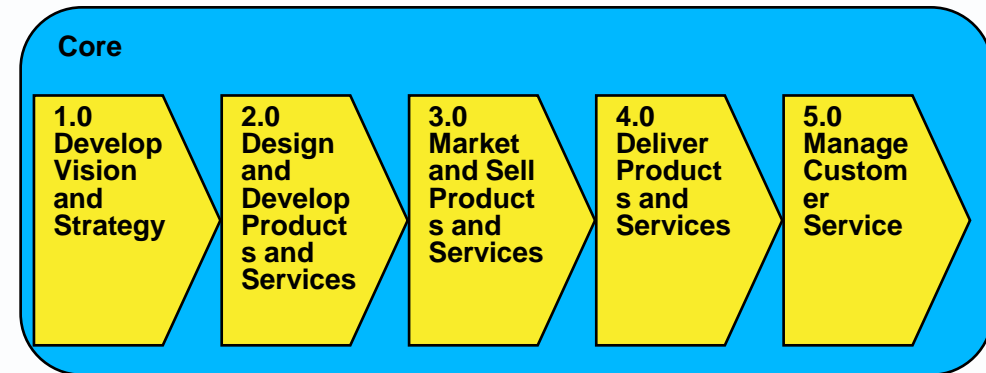
- ▶ [a pattern] a generic structure or classification used to create more specific models.
- ▶ It can be a structure of components, processes or data elements.
- ▶ It is sometimes applicable to a particular industry or business domain.
- ▶ It can act as a design pattern.

- ▶ The course includes several reference models
 - **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.

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

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 complain	
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	