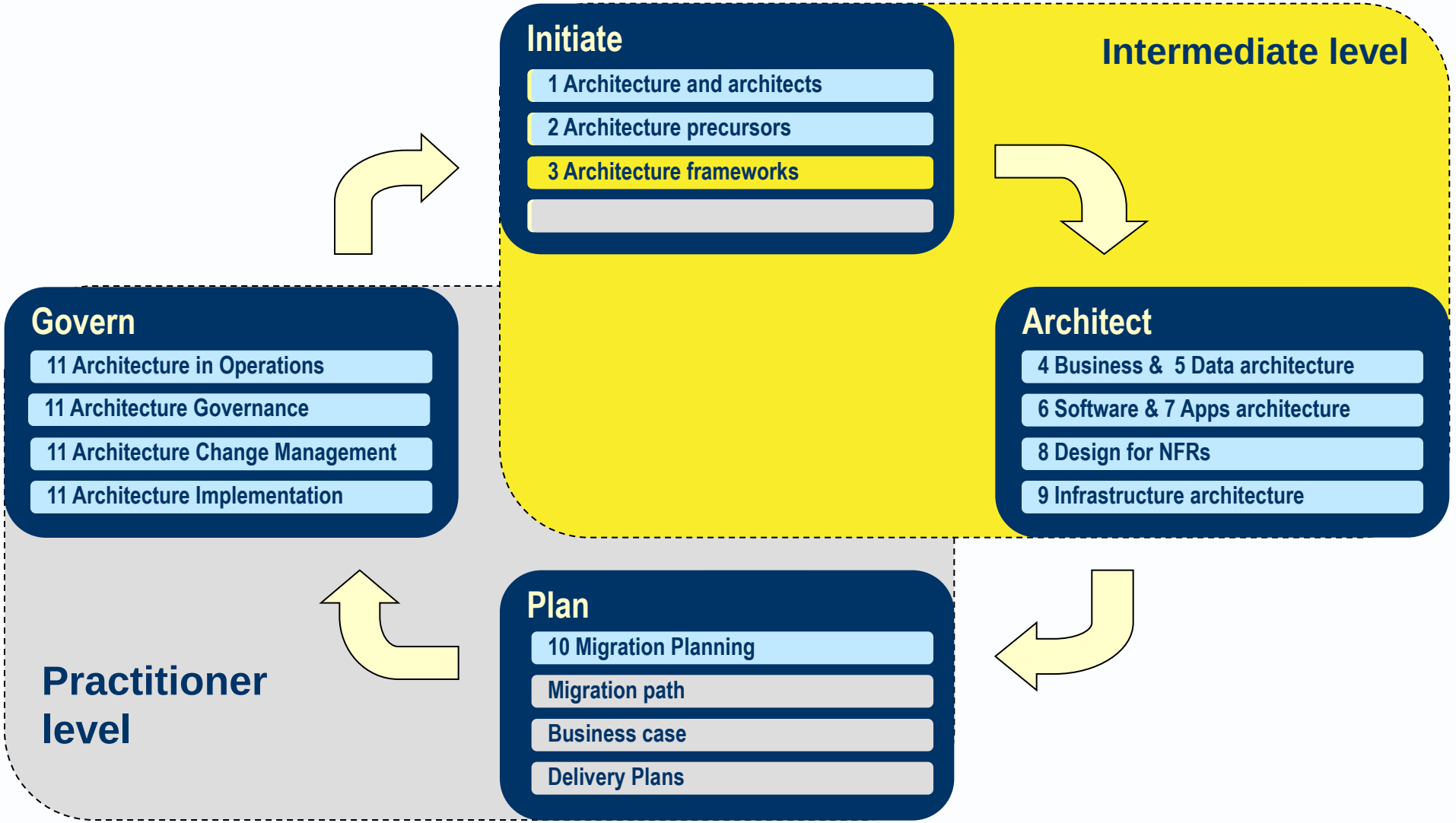
Avancier Reference Model

Architecture Frameworks (ESA 3)

Mapping the reference model to an architecture framework

Avancier



3.1: Foundation

- Architecture state
- TOGAF: The Open Group Architecture Framework
- ADM: Architecture Development Method
- AM: Avancier Methods

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

Avancier

- [an architecture framework] for transforming an enterprise architecture from a baseline state to a target state.
- It is published on a free-to-read public web site, though its use by a commercial organisation is restricted by copyright conditions.
- It is centred on a process called the architecture development method (ADM).

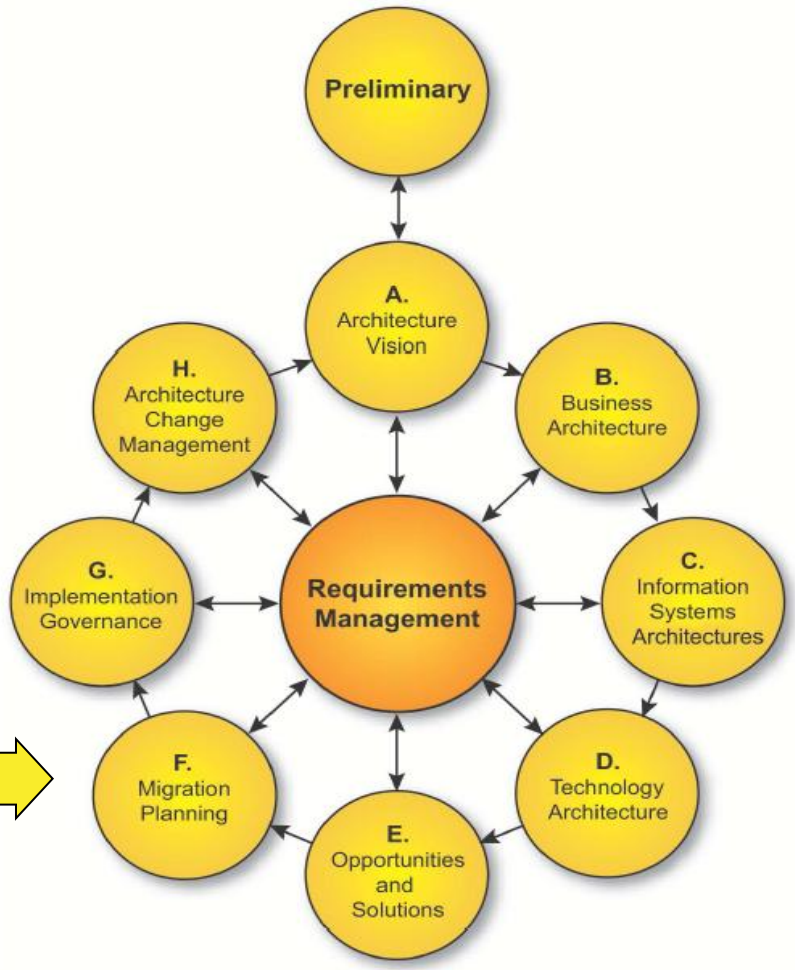


Figure 5-1 Architecture Development Cycle

ADM: Architecture Development Method

Avancier

- [a process] defined in TOGAF to develop and use an enterprise architecture.
- It involves 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.

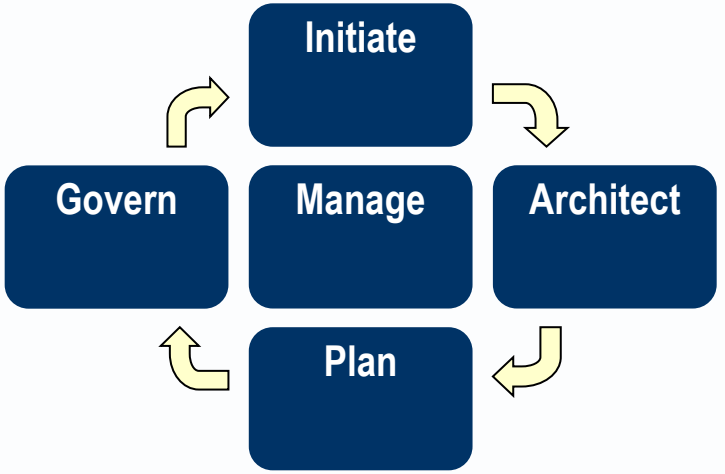
Process

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.



3.2: Architecture description

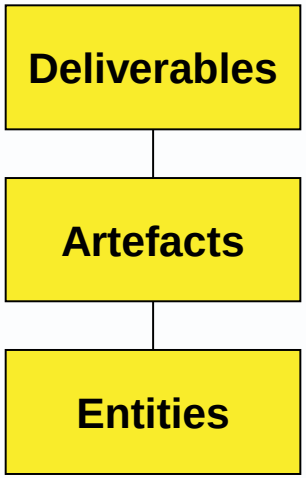
- Architecture content framework
- Architecture deliverable
- Architecture artefact
- Architecture entity
- Mapping
- View
- Viewpoint
- Model

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.

Architecture content framework

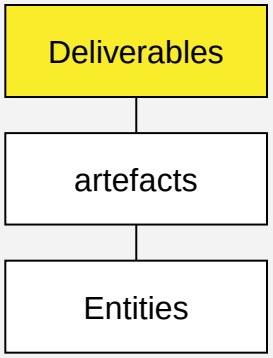
- [a passive structure] for organising an architecture description composed of deliverables, artefacts and entities.



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 deliverable



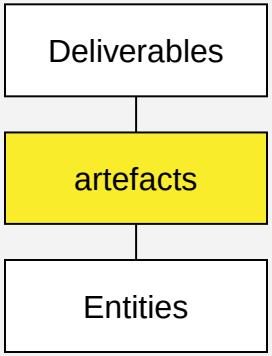
Deliverable examples:

- Request for Work,
- Statement of Work,
- Architecture Requirements,
- Architecture Definition,
- RAID Catalogue,
- Migration Plan.

Architecture artefact

- [a model] that conforms to one of three artefact types: catalogue, matrix and diagram.

Architecture artefact

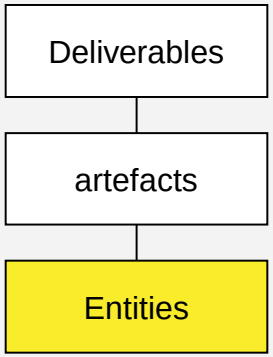


PRECURSORS: Goal or requirements hierarchy, Goal or requirements traceability, Process map, Context diagram.
BUSINESS: Business function structure, Business process model, Organisation structure, Location structure, Business function dependency matrix, Business data model.
DATA: Data model, Data lifecycle, Data structure, Business function-Data CRUD matrix, Application-Data CRUD matrix, Data dissemination matrix.
APPLICATIONS: Application portfolio, IS context diagram, Business-Applications matrix, Applications architecture diagram, Application decomposition diagram, Software layering diagram.
INFRASTRUCTURE: Technical Reference Model, Standards Information Base, Technical environments outline, Hardware configuration diagram.

Architecture entity

- [an entity] in an architecture description meta model.
- One entity instance can appear in several artefacts, which can appear in several deliverables.

Architectural entity



PRECURSORS: Stakeholder, Business goal/objective, Principle, Standard.
BUSINESS: Organisation unit, Business function, Business process, Role, Actor, Business service, Location, Time period.
DATA: Data entity, Data event, Data flow, Data quality, Data source, Data store.
APPLICATIONS: Application, Data flow, Use case, Automated service, Component.
INFRASTRUCTURE: Technology, Computer, Network

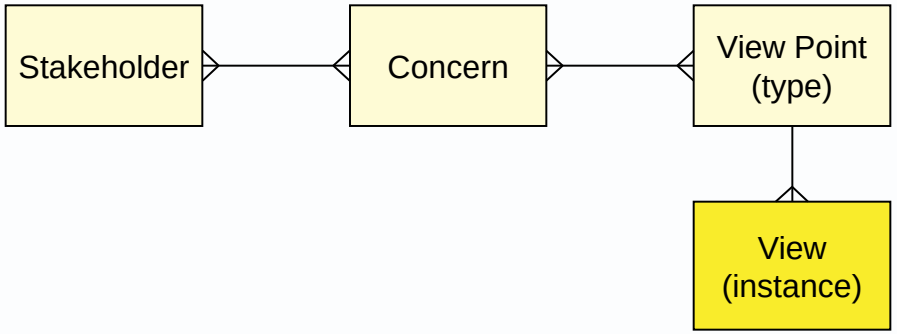
Mapping

- [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	

View

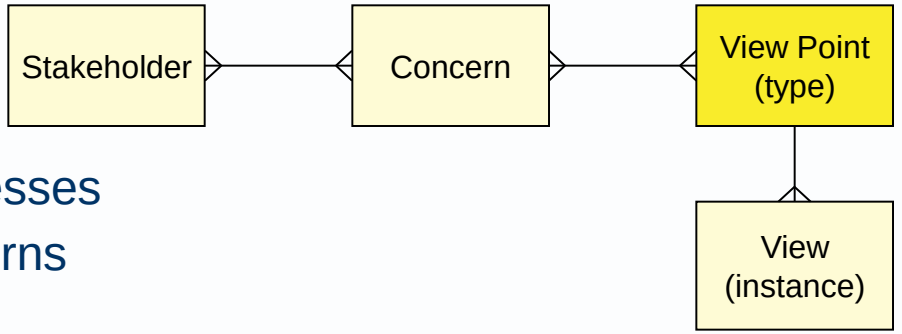
- [a work product] that shows a part or slice of an architecture that addresses particular concerns.
- It can be visual, graphical or textual, and may contain one or more models.
- It can be an instance or example of a viewpoint, meaning that it conforms to the definition of that viewpoint.



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

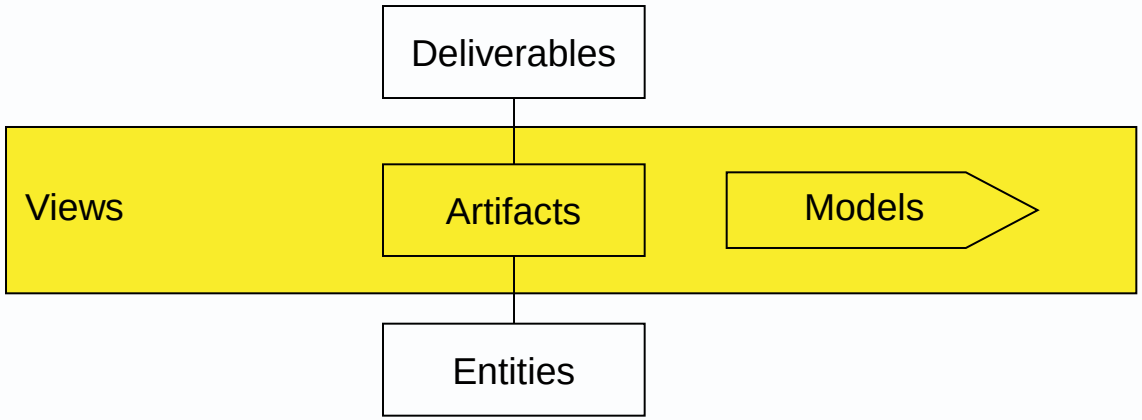
Viewpoint

- [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.



Model

- [a description] a description that simplifies or abstracts from a thing or another description.
- It displays or records some properties of what is modelled and enables some questions to about it to be answered.



3.3 Abstraction of architecture descriptions from systems

- **Abstraction**
- **Refinement**
- **Concretion**

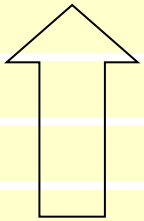
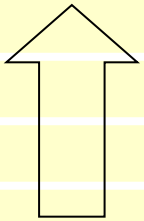
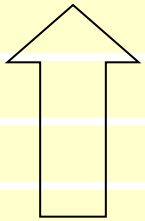
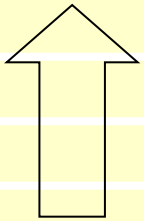
- **Omission**
- **Elaboration**

- **Composition**
- **Decomposition**

- **Generalisation**
- **Specialisation**

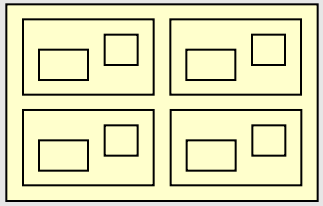
“EA is considerably abstracted from Solution Architecture, design, or implementation views.” (TOGAF).

Abstraction varieties

Omission	Composition	Generalisation	Idealisation
			
Elaboration	Decomposition	Specialisation	Realisation

Abstraction	[a technique] by which a simpler description is derived from other descriptions or from reality.
Refinement	[a technique] that yields a detailed description that conforms to a more abstract description. Everything in the abstraction holds, perhaps in a somewhat different form, in the refinement.
Concretion	[a technique] that instantiates a description as one or more real, active (or activatable) components.

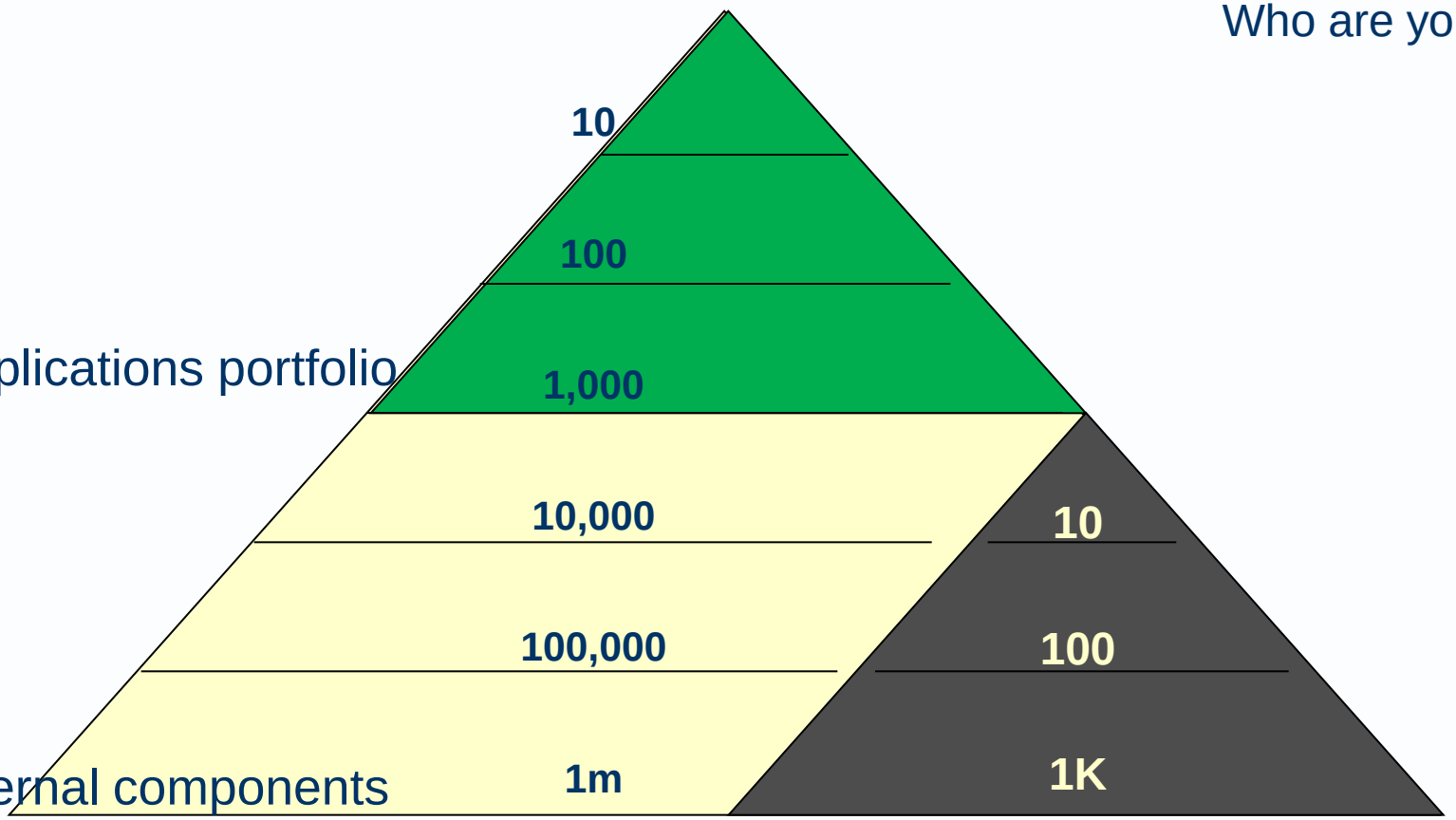
Foundation concepts

Composition 	[a technique] that assembles parts into a whole and/or hides components behind a façade. Architects describe systems in terms of coarse-grained components, processes and services.
Decomposition	[a technique] that divides a whole into parts and/or identifies components behind a facade. The conventional advice is that it is difficult to maintain the integrity of a hierarchical structure that is decomposed more the three or four levels (or more than a thousand elements) from the top.

Composition – Coarse-grained views and models

Avancier

- Applications portfolio
- Internal components



Who are you talking to?

Manager

Technician

Foundation concepts

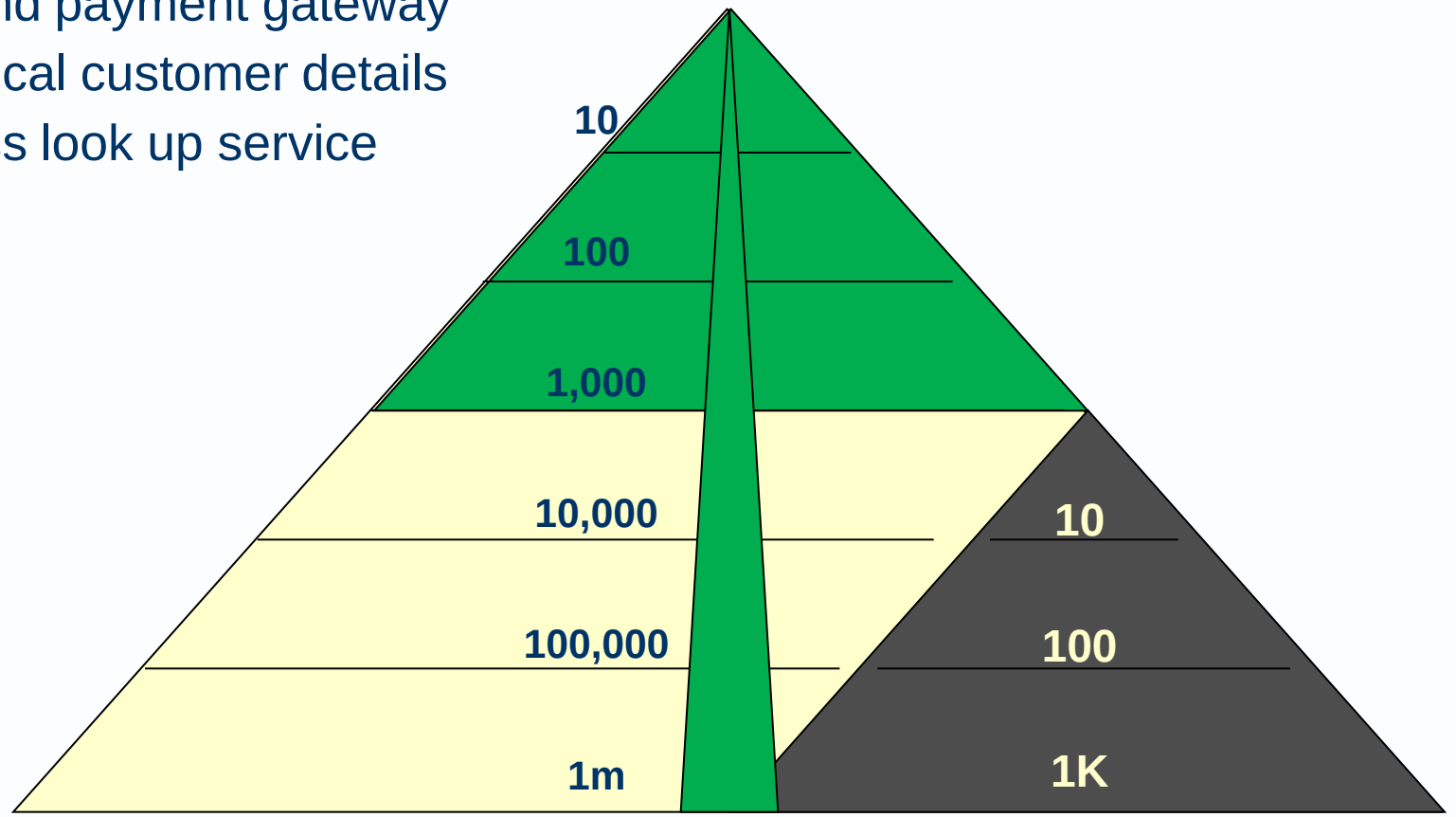
Avancier

Generalisation	[a technique] that defines properties shared by subtypes or entities. Architects look to maximise re-use of common components, processes and services across the enterprise.
Specialisation	[a technique] that extends or modifies generic properties to define a subtype or smaller population of entities. It can also mean configuring an instance by selecting variable values from a general range.

- Beware that generalisation of
  - architecture descriptions can yield abstractions of low practical benefit.
  - solution components for 'flexibility' can lead to performance problems.

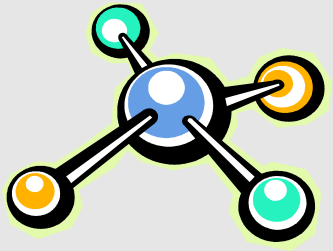
Generalisation - Common components and processes

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



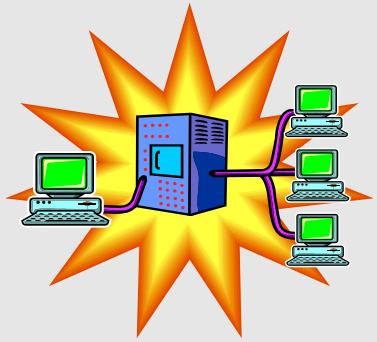
Architecture description: models and abstractions

Idealisation



[a technique**]** that reverse-engineers a more logical description by omitting some details relevant to a particular physical form of the thing described.
Architects produce and work with logical descriptions of components, processes and services.

Realisation



[a technique] that forward-engineers a more physical description. It adds details relevant to a particular physical form of the thing described.
OR concretion: the instantiation of a description as one or more active, run-time, components.

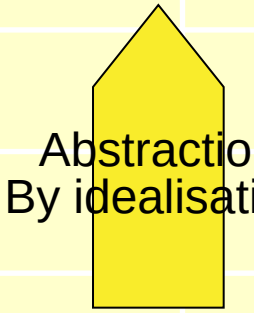
A kind of generalisation that produces a logical description of a physical component or process.

A kind of reverse engineering; the resulting description can be presented as the requirement for the real thing.

Idealisation - in architecture frameworks

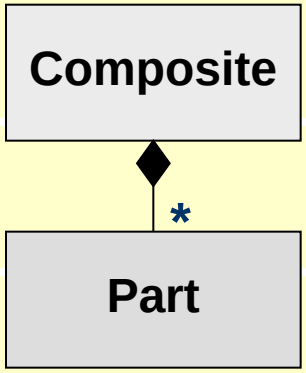
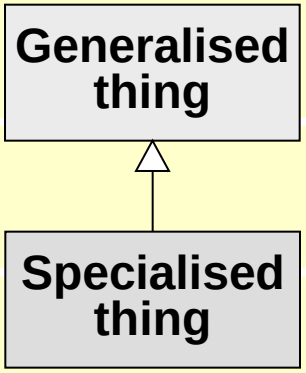
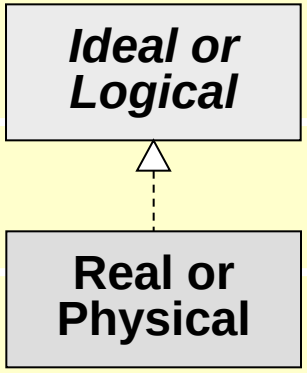
Avancier

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						



EA is more *abstract* than SA

Avancier

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/logica**l views and models
			
Elaboration	**Decomposition**	**Specialisation**	**Realisation**

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**

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 Architecture level	Business Architecture	Data Architecture	Applications Architecture	Technology Architecture
Enterprise Architecture				
Solution Architecture				
Software Architecture & Technical Specialisms				

3.4: Architecture models and languages

- Idealisation hierarchy
  - Conceptual (or domain) model
  - Logical model
  - Physical model

- MDE: Model-Driven Engineering
- Model-Driven Architecture (MDA)

- Modelling language
  - IDEF: Integration DEFinition language
  - UML: Unified Modelling Language
  - ArchiMate

Idealisation hierarchy

- the classic hierarchy of conceptual, logical and physical model.

  - **Conceptual (or domain) model** [an artefact] an abstract logical model that defines terms and concepts in a business or problem domain without reference to any computer application.
  - **Logical model** [an artefact] a model of a particular system that excludes details of that system's physical implementation.
  - **Physical model** [an artefact] a model of a particular system that is vendor or technology specific and/or includes details of its physical implementation.

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

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
Business data model	Business process model
Logical data model	Platform-independent model
Physical data model	Platform-specific model

Elaboration by Forward Engineering

Abstraction by Reverse Engineering

Model-Driven Architecture (MDA)

- [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

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] in the field of systems and software engineering, originally funded by the US DoD. Its most-well-known notations are 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. It was designed to help in OO software design, though often used outside of that. It includes structural models such as class diagrams and deployment diagrams, and behavioural models such as use case, activity and sequence diagrams.

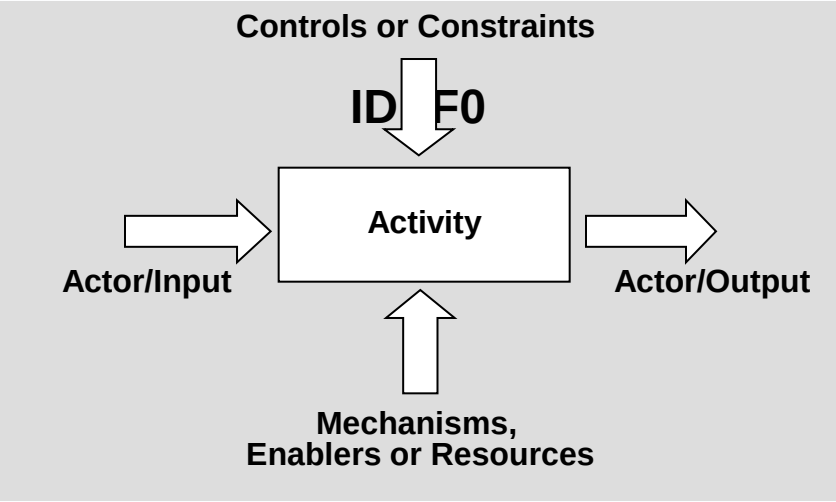
  - **ArchiMate** [a modelling language] maintained by the Open Group. It was designed to help in architecture description. Components, interfaces and services are shown in distinct boxes. It overlaps with UML but is intended for more abstract design.

Integration Definition language (IDEF)

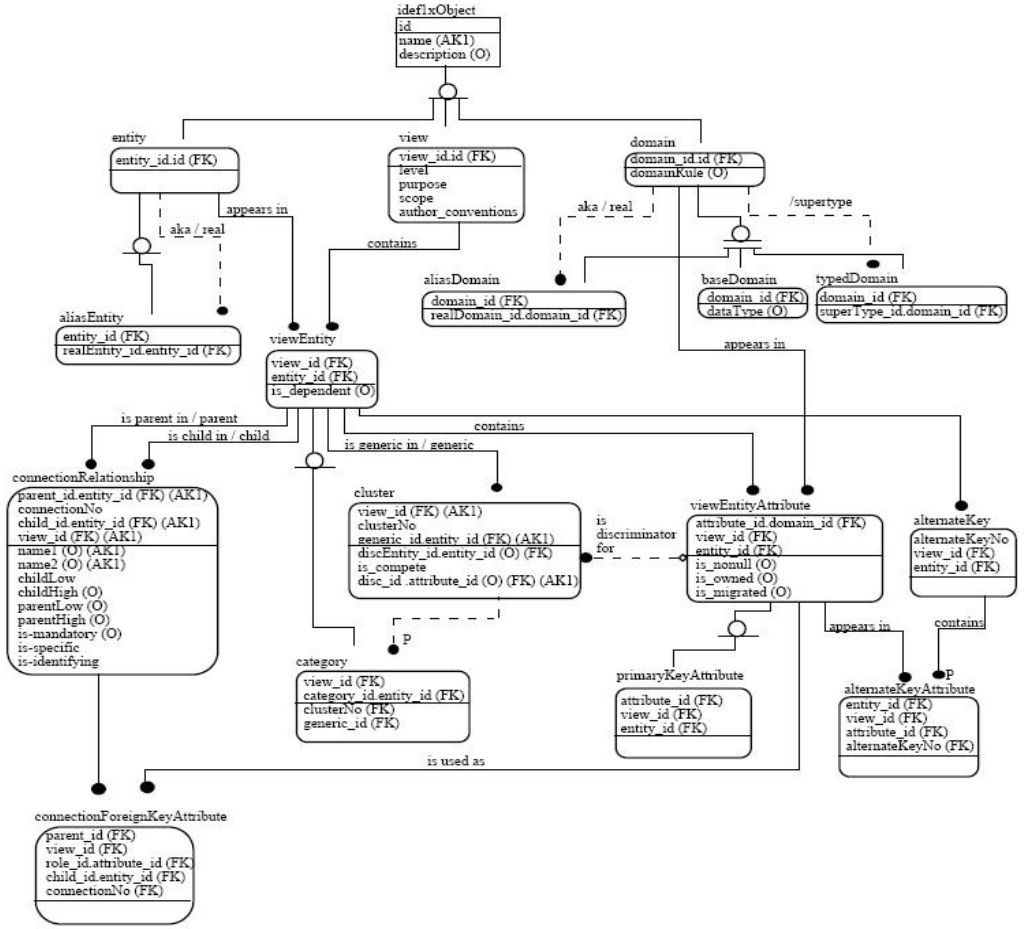
Avancier

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

- IDEF 0 Function Models

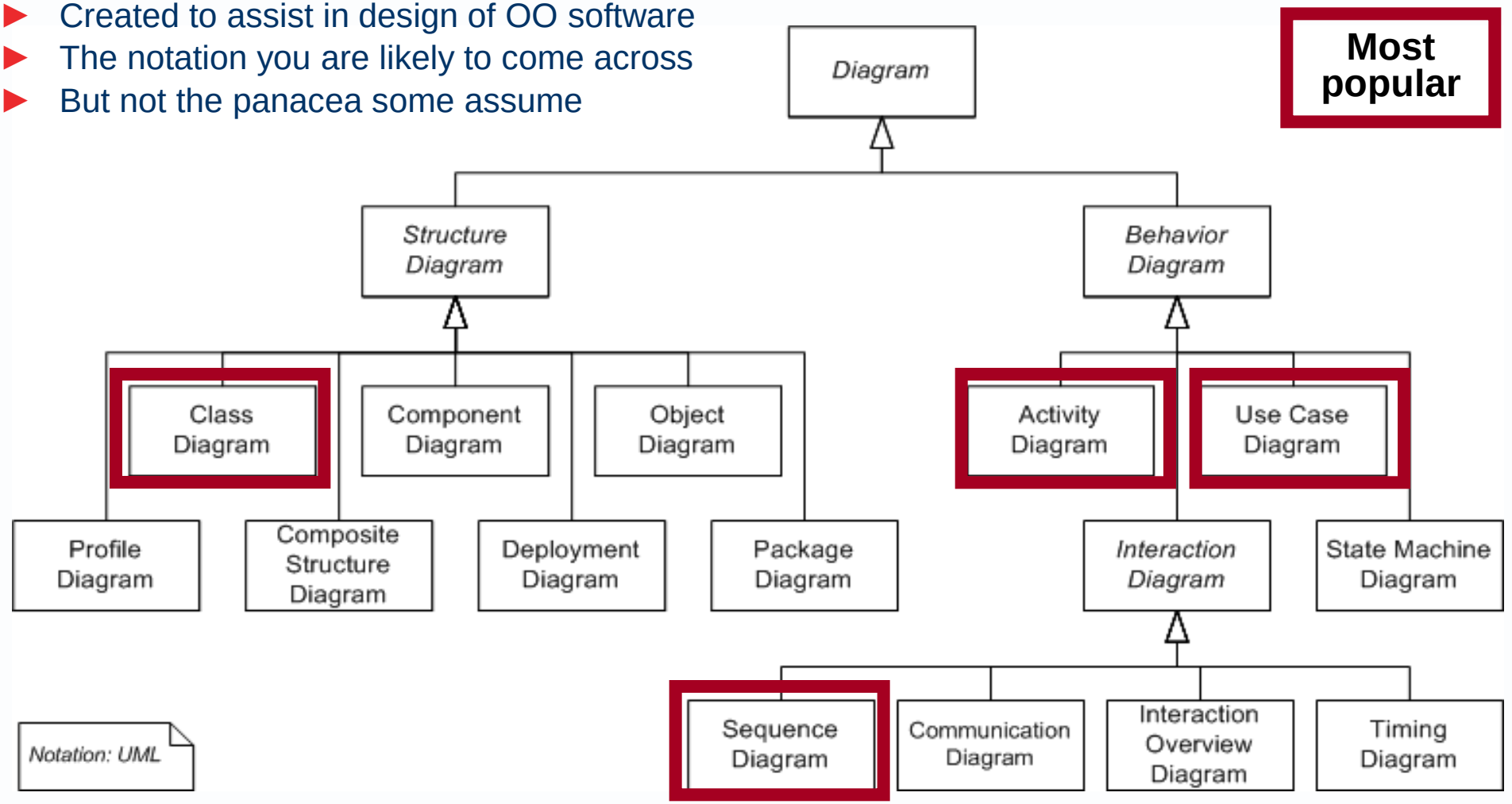


- IDEF 1X Data Models (Wikipedia)



UML (OMG standard)

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



3.5: Pre-defined classifications and reference models

- Architecture repository
- Zachman framework
- Enterprise continuum
- Reference model

Architecture repository

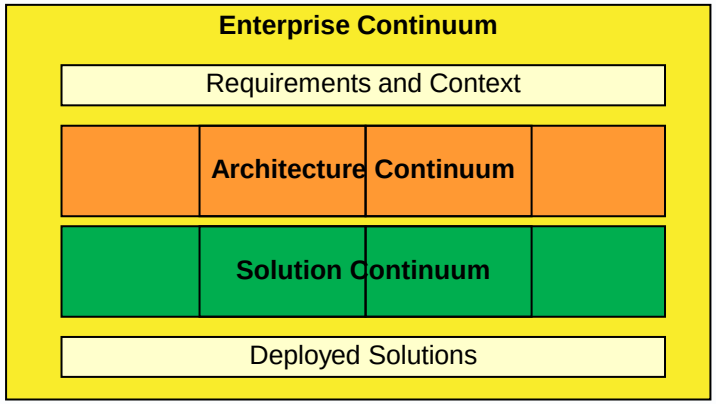
- [a data store] an information base used by architects; a system that holds and manages all the meta data that describes an enterprise and its information systems.
- Its structure is defined in some kind of schema or architecture meta model.

Description frameworks

- The content of the repository can be categorised using, for example, the Zachman Framework or Enterprise Continuum.

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

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.”

- The 6 columns, though titled with interrogative questions, are mapped to architectural description elements.
- The 6 rows are primarily levels of idealisation-realisation from context to operational systems, but are 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

Enterprise continuum

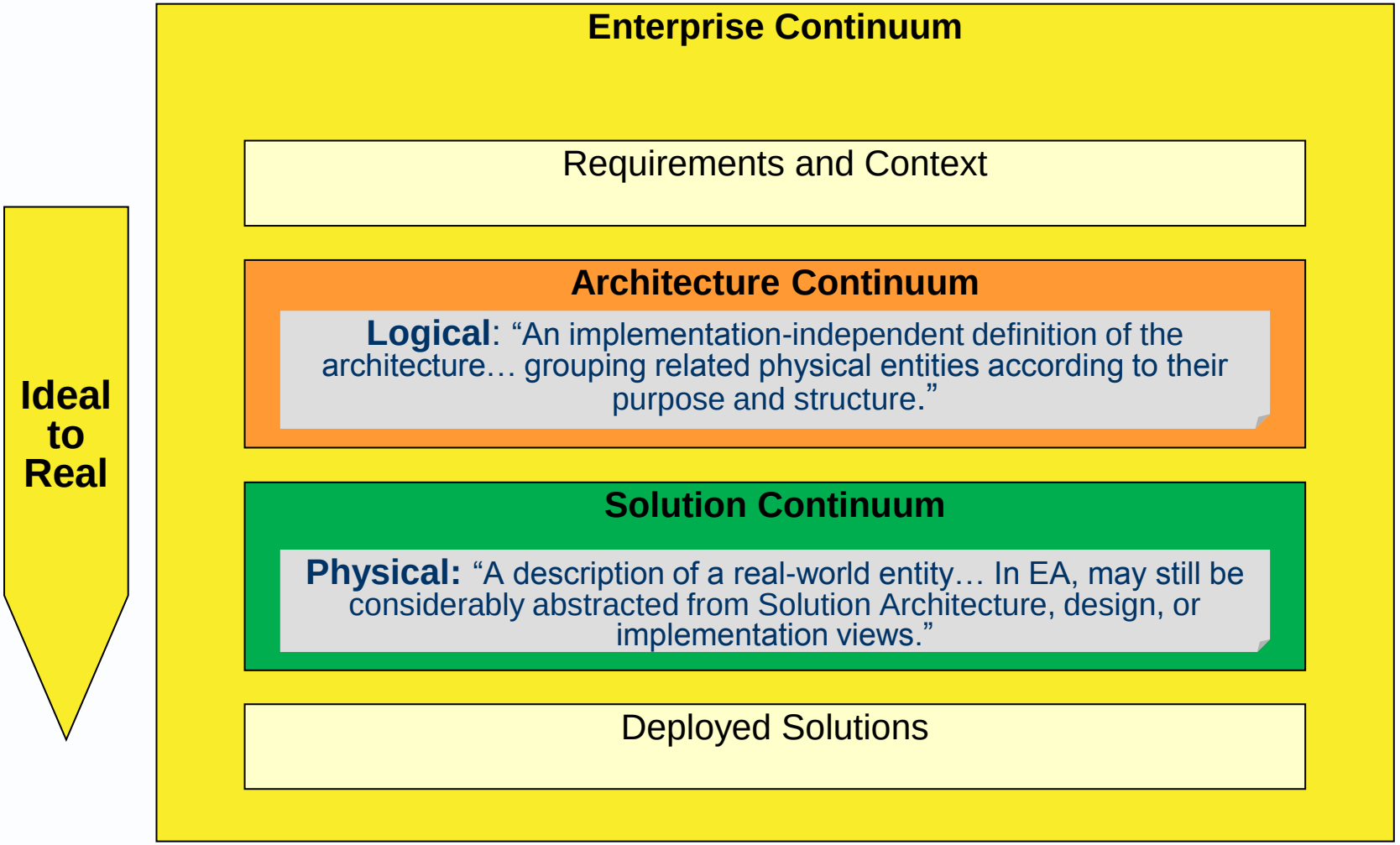
Avancier

- [a pattern] a logical structure for classifying and organising architecture description artefacts.
- It is a core part of TOGAF.
- 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				

TOGAF's Enterprise Continuum – top to bottom

Avancier



The core of the enterprise continuum

Avancier

Gen-Spec

Ideal to Real

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

TOGAF's two Reference Models

Avancier

Gen-Spec

Ideal to Real

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				

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.

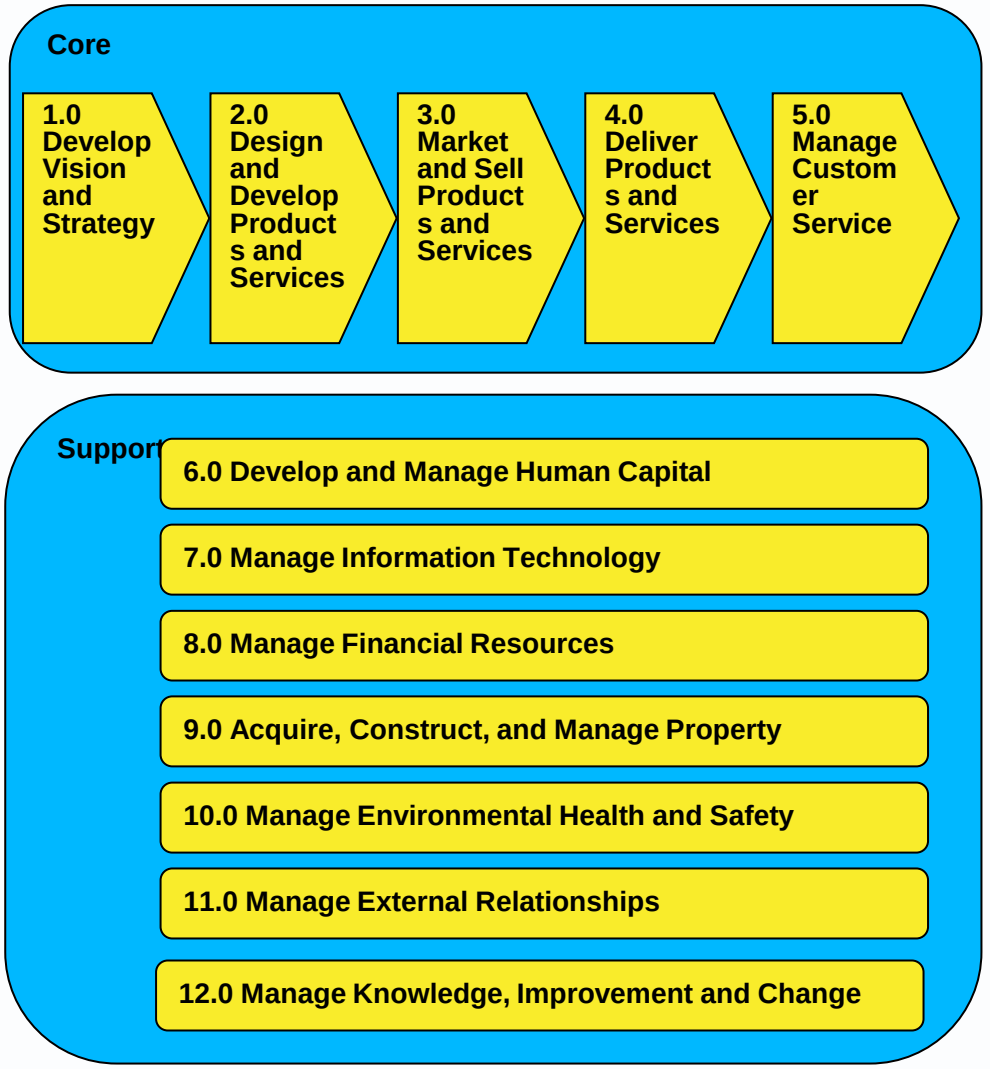
- The course includes several reference models
  - **APQC for a generic commercial organisation**
  - BIAN for banking (one of many banking reference models)
  - TMF for telecoms
    - eTOM – Business Architecture
    - SID – Data Architecture
    - TAM – Applications Architecture
  - SCOR for supply-chain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

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

- 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