IT Training and
Professional Development

Reference Model for BCS Certificates in Enterprise and Solution Architecture

Version 4.0

June 2012

Reference Model for BCS Certificates in Enterprise and Solution Architecture

This reference model follows the structure of the syllabus for BCS examinations in enterprise and solution architecture. It defines terms used in the syllabus. It is designed to help:

- Examiners to scope and phrase examination questions.
- Examinees to understand terms and concepts in examination questions.
- Training Providers to scope training courses that lead to the examinations.

Acknowledgements

BCS gratefully acknowledges that about half this reference model (at version 1) was based on the reference model published in 2008 by Avancier Ltd as an aid to architect training at http://avancier.co.uk. The two models are aligned at the time this model is published.

Many other public domain sources (such as the Object Management Group, Open Group and ITIL) have been used. Definitions which have been taken from another source are included in quotation marks. Definitions which have been tailored to ensure consistency between terms within this reference model are not attributed to any particular source.

Contents

1. Architecture and Architects

This section is about work and roles to describe the high-level design of business systems and the information systems that support them (not work and roles related to buildings).

1.1 Foundation Terms and Concepts

This reference model defines terms used in specifying the structure and behaviour of activity systems. Four essential architecture concepts can be classified as shown in this table.

	Behaviour	**Structure**
External	Service	Interface
Internal	Process	Component

This reference model is based the notion that these four concepts can be applied in each of three architecture domains, shown here as a layered architecture domain hierarchy.

Business	Business Services	Services
	Business Functions	Components
Information Systems	Information System Services	Services
	Applications	Components
Technology Infrastructure	Platform Services	Services
	Technologies	Components

The remainder of this reference model defines terms used in architecture. The definitions are generalisations that draw from several sources.

Architecture	1: documentation describing the structure (components) and behaviour (processes) of a system. A detailed plan of the system to guide its implementation. Or 2: the process for describing the architecture of a system to meet given requirements and under given constraints.
Structure	What a system is made of. A configuration of items. A collection of inter-related components or services. Possible structures include hierarchies (items arranged in a cascade of one-to-many relationships) and networks (items connected by many-to-many relationships).
System	A structure of interacting subsystems or components that perform activities. An encapsulated collection of processes that transform inputs into outputs (Output being the purpose of a man-made system).
Software system	A system in which computer programs execute the processes.

Human activity system	A system in which people execute some or all of processes. A rich system, much more complex and loosely-defined than a software system. It relies to a greater or lesser degree on human personalities, skills, ingenuity and judgements about what to do and how to do it.
Component	A subsystem that is encapsulated behind an interface - can be replaced by any other with the same interface - is related to other subsystems by requesting or delivering services. A component can have several interfaces.
Building block	A synonym in some architecture frameworks for a component and/or an architectural entity. To avoid this ambiguity, this reference model eschews the term.
Function	1: A component that offers several services. Or 2: A process that delivers a single service. Or 3: A purpose or goal of a component or process. To avoid this ambiguity, this reference model eschews the term, except within business function.
Interface	The means of connecting to a system, process or component. 1: A list of services, offered by one or more components. Or 2: The signature (name, inputs and outputs) of one service. Or 3: A data flow between sender and receiver components Or 4: The protocols used to exchange data between components. Or 5: The channel via which data flows are passed This reference model favours definition 1.
Location	A place where business is done, or where computers, applications or their users are. An identifiable point in space or geography. An architectural entity that appears in artefacts such as technology deployment diagrams.
Time Period	A slot in a schedule. An architectural entity that appears in artefacts such as application availability tables.
Behaviour	What a system does. What external entities observe a system as doing. The processes executed or supported by a system.
Service	1: The outcome of a process that is executed in response to a request from a client. Or 2: An interface of a component (such as a web service) that may offer many services of kind 1. This reference model favours definition 1, to limit confusion between components and services.
Service contract	The signature, semantics and non-functional characteristics of a service. The signature is what a client needs to invoke a service - composed of a name, inputs (arguments) and outputs. The semantics are what a client designer needs to know of what the service does - composed of its preconditions and post conditions. The non-functional characteristics are what a client designer needs to know of the conditions under which the service works, which includes both performance and commercial conditions.

Process	1: A procedure, started by an event, which terminates with the delivery of an output or service. A set of steps or activities arranged under a control flow in one or more sequential paths. Or 2: An encapsulated system or subsystem - as in the "active process" on a computer. This reference model favours definition 1.
Life cycle	A process from birth to death. E.g. the life of: ▪ a system from conception through deployment and use to removal ▪ a project from conception through development to delivery ▪ an entity [See Data Lifecycle.]
Abstraction	1: A process of composition, generalisation or idealisation by which shorter and more general descriptions are produced from longer, more detailed and more specific descriptions. Or 2: A simplified description of a system that is made by the process at definition 1. Abstraction is tool that every architect must be able to use. Enterprise architects work with the highest abstractions.
Composition	1: the assembly of parts into a whole; Or 2: the organisation of units under a manager or owner; Or 3: the encapsulation of content inside a shell. The result is some kind of aggregate or composite component or process. A composite contains its components in some sense. A description made with reference to a whole, manager or shell is more abstract than one that refers to its parts, units or content. Enterprise architects tend to work with coarse-grained components.
Decomposition	The opposite of composition. Division of one composite or aggregate component or process into several components or processes. 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.
Generalisation	Abstraction from several specific components or processes into a more generic component or process. A generalisation contains only the common features of its specialisations. Enterprise architects look to re-use generic components and services.
Specialisation	The opposite of generalisation; the elaboration or division of a general component or process into one or more specific components or processes. A specialisation contains its generalisation in some sense.
Idealisation	1: A kind of generalisation that produces a logical description of a physical component or process. Reverse engineering to produce a description that can be presented as the requirement for the real thing. Or 2: The separation of an interface from the component(s) that realise the interface. Enterprise architects define logical components and services.

Realisation	The opposite of idealisation; leads to the instantiation of physical materials. It can be viewed as forward engineering, as a progression from requirement to solution.

1.2 Intermediate Terms and Concepts

1.2.1 Architecture Granularity

Architecture granularity	A scale which ranges from coarse-grained or high level to fine-grained or detailed. Generally speaking, the narrower the scope of the system, the finer-grained the description that can be written in a given time. This reference model distinguishes three levels of architecture granularity.
Enterprise architecture	A strategic approach to architecture that addresses a whole enterprise. The highest level, widest scope, longest term kind of architecture. 1: documentation describing the structure and behaviour of an enterprise and its information systems. Or 2: a process for describing an enterprise and its information systems and planning changes to improve the integrity and flexibility of the enterprise.
Solution(s) architecture	A relatively tactical approach to architecture that addresses specific problems and requirements, related to selected information systems and business processes. 1: documentation describing the structure and behaviour of a solution to a problem. Or 2: a process for describing a solution and the work to deliver it.
Software architecture	A kind of architecture that addresses the scope and interaction of software components within an application. 1: Documentation describing the internal structure and behaviour of a software application. Or 2: Principles and patterns for software modularisation, for describing and building the internal structure of an application.

1.2.2 Architecture Domains

Architecture domain	A broad area architectural interest such as business, data, applications, and technology infrastructure. A facet or view of an architecture description. A description that hides other facets or views of the system described. A partial representation of a whole system that addresses several concerns of several stakeholders.

Primary Domains

Business architecture	The structure and behaviour of a business system (not necessarily related to computers). Covers business functions or capabilities, business processes and the roles of the actors involved. Business functions and business processes are mapped to the business goals and business services they support, and the applications and data they need.
Data architecture	The subset of information architecture that is focused on the definition, storage and movement of structured data. The data structures used by a business and/or its applications. Includes meta data: that is, descriptions of data in storage, data in motion, data structures and data items. Includes mappings of data objects to data qualities, applications, technologies etc.
Applications architecture	The structure and behaviour of applications used in a business, focused on how they interact with each other and with business users or actors. Focused on the data consumed and produced by applications rather than their internal structure. The applications architecture is shaped by where data is obtained from and where it is used. The applications are usually mapped to the business functions they support and the platform technologies they need. [See also application portfolio management.]
Infrastructure architecture	The structure and behaviour of the technology platform that underpins user applications. Covers the client and server nodes of the hardware configuration, the platform applications that run on them, the services they offer to applications, the protocols and networks that connect applications and nodes.

Other Domains

Information architecture	A broad domain that covers both structured and unstructured data, that is, content, document and knowledge management. (This reference model focuses on structured data.)
Information systems architecture	Often used to mean the combination of applications architecture and data architecture. It depends on but does not include the technology platform.
Software (aka application) architecture	The internal structure, the modularisation of software, within an application. Discussed for example in “Patterns of enterprise application architecture” by Martin Fowler. This is applications architecture at the lowest level of granularity - usually below the level of modularity that enterprise and solution architects define. However, there is no rigid dividing line.
Security architecture	The various design features designed to protect a system from unauthorised access. Not a cohesive architecture on its own so much as features of the other architecture domains – business, data, applications and infrastructure – as mentioned in other sections.

1.2.3 Hierarchical or Layered Architecture

Hierarchical or layered architecture	A structure in which components are organised into layers, such that components in one layer delegate work to components in the layer below, but not the layer above.
Platform	The layer that lies beneath and provides services to the layer under consideration. E.g. beneath the applications are application platform technologies including database management systems, and beneath the operating system is the hardware platform.
The architecture domain hierarchy	Business, application and infrastructure architecture domains viewed as a hierarchically layered structure. The components in one layer offer services to components in the layer above. This is a useful mental model for enterprise and solution architects alike. This reference model features several other hierarchical decomposition structures.

1.2.4 Architect Roles, Goals and Skills

Architect	One who designs buildings and superintends their construction. One who describes the architecture of system in sufficient detail for work to be planned and detailed design and building to proceed, then governs the building work.
Architect Role	The list of roles below has been agreed by BCS as sufficient for examination purposes. The focus of the following roles is evident from the definitions of architecture levels: ▪ Enterprise architect: defines principles, policies and plans that cover several solutions to several business problems. ▪ Solution architect: describes the structure of solution to a business problem, which may include several applications and technologies. ▪ Software architect: describes how a single application is built from software modules, at a fine-grained level. ▪ The focus of the following roles is evident from the definitions of architecture domains: ▪ Business architects: focus on business architecture; their principal concerns are the functions and processes of the business. ▪ Data architects: focus on data architecture; their principal concern is to ensure data quality in data stores and data flows, through the definition and maintenance of meta data. ▪ Applications architects: focus on applications architecture; their principal concern is the modularity of applications and data that flows between them. ▪ Technical/infrastructure architects: focus on technical/infrastructure architecture.

Enterprise architect goals	The list of goals below has been agreed by BCS as sufficient for examination purposes. ▪ Improved alignment of business and IT ▪ Improved IT cost-effectiveness ▪ Business agility ▪ Technical agility ▪ Long term planning: enablement of strategically beneficial IS/IT work. ▪ Vendor and technology independence (portability) ▪ De-duplication of applications and technologies ▪ Interoperability of applications and technologies ▪ Simpler systems and systems management. ▪ Improved procurement.
Solution architect goals	The list of goals below has been agreed by BCS as sufficient for examination purposes. The solution architect supports the goals of an enterprise architect, but focuses more on the following: ▪ Timeliness of IS/IT project deliverables ▪ Cost of IS/IT project deliverables ▪ Quality of IS/IT project deliverables ▪ Solution-level risk identification and mitigation ▪ Application integration and data integrity ▪ Conformance of solution to non-functional and audit requirements ▪ Conformance of solution to principles, standards, legislation. ▪ Effective interaction between managers and technicians. ▪ Governance of detailed design to architecture principles and standards.
Architect knowledge and skills	The list below has been agreed by BCS as sufficient for examination purposes. ▪ Holistic understanding of business and technical goals. ▪ Holistic understanding of business and technical environment ▪ Broad technical knowledge – including current trends. ▪ Broad methodology knowledge ▪ Analysis of requirements and problems ▪ Innovation. ▪ Leadership. ▪ Stakeholder management. ▪ Communication, political and soft skills. ▪ Awareness of project management and commercial risks and issues.

1.3 Practitioner Terms and Concepts

The table below suggests analogies between business systems and software systems. Practitioner architects should understand such analogies, and their limits.

Business (human activity) systems	Software (computer activity) systems
An enterprise offers services (business services) to its customers.	An application offers services (use cases) to its users.
An enterprise is divided into component divisions, which are further subdivided.	An application is divided into components, which are further subdivided.
Enterprise divisions offer business services to each other.	Application components offer automated services to each other.
Some divisions (e.g. Accounting, Procurement, HR) offer common services to many other business functions.	Some components (e.g., Address retrieval, Currency conversion, Date, Payment) offer common services to many other applications.

2. Architecture Precursors (Requirements & Context)

This section is about the various inputs, the requirements and constraints that guide an architect as to the nature and shape of the solution to be built. This information is needed to support a statement of architecture work. Note that management activities are addressed in sections 10 and 11.

2.1 Foundation Terms and Concepts

2.1.1 Stakeholders

Stakeholder	Person or role with power over and/or interest in work to be done or its deliverables. A person who has one or more concerns about the system to be built (because they own it, manage it, use it or other reason).
Stakeholder management	A technique based on analysis of stakeholders' positions in a power/interest grid, which determines a communication plan for each stakeholder.
Sponsor	A stakeholder who is willing to apportion money or other resources to some work.
Concern	A general kind of requirement (e.g. availability, usability) that is important to one or more stakeholders in the system, and determines the acceptability of the system to those stakeholders. A concern may be addressed in several view points. A view point may address several concerns.
Architect stakeholder	Enterprise and solution architecture stakeholders include: ▪ Owners: business and IT board members, customers. ▪ Managers: programme/project/change managers. ▪ Buyers: procurement/acquisition organisation. ▪ Suppliers: service and product providers. ▪ Designers, Builders, Testers: other project team members: ▪ Users: representatives and domain experts. ▪ Operators and Maintainers: IT Services Management.

2.1.2 Elaboration of Inputs to become Deliverables

Elaboration of inputs to become deliverables	The inputs to architecture definition include high level aims, directives, visions and strategies. During architecture definition, these are decomposed and elaborated. So the outputs include lower-level elaborations of the inputs. This reference model tends to distinguish different levels of decomposition by using different words.

2.2 Intermediate Terms and Concepts

2.2.1 Drivers, Aims and Directives

Driver	A pressure (internal or external) that helps to shape aims and plans. E.g. customer feedback, increased competition, staff retention.

Aim hierarchy	A hierarchy of goals, objectives and requirements, applicable to an enterprise, system or project.
Goal (business or technical)	An aim at the top of the aim hierarchy. It is often decomposed into lower level objectives. Sometimes qualitative; sometimes quantified using SMART Key Goal Indicators.
Objective	An aim in the middle of the aim hierarchy. It supports one or more higher-level goals. It should have SMART Key Performance Indicators.
Requirement	An aim at the bottom of the aim hierarchy. [See "requirements statement" for further definition.]
Balanced Score Card	A management tool in which top-level objectives are spread across four categories, then cascaded down the organisation and decomposed at each level.
SMART	The acronym for Specific, Measurable, Actionable, Realistic and Time-bound. The qualities of a good goal, objective or requirement.

The directive hierarchy below reconciles TOGAF 9 with the OMG's Business Motivation Model by placing Principles above Policies and Rules, and defines these terms so as to ensure consistency with other terms in this reference model.

Directive hierarchy	A hierarchy of principles, policies and business rules, applicable to an enterprise, system or project. Each is a statement that guides people along a path to reach an aim, and can be used as a tool of governance.
Principle	A directive at the top of the directive hierarchy. A strategic, abstract and not-directly-actionable directive that derives from high-level goals. A statement of an outcome that reflects goals. A tool used in governance of architecture work. Principles can be related to business, data, applications, infrastructure or security. E.g. ▪ Waste should be minimised. ▪ Data security is paramount.
Policy	A directive in middle of the directive hierarchy. A tactical directive that derives from objectives. A tool used in governance of day to day work. It guides behaviour that the company expects will lead to desired outcomes. e.g. Members of the public have minimal access to data. ▪ USB ports are disabled. ▪ Message data at security level 3 is encrypted. [The definition in the OMG Business Motivation Model "A rule that governs or guides the strategy" is closer to Principle above.]

Business Rule	A directive at the bottom of the directive hierarchy. It directs and constrains a procedure. It appears in specifications of automated data processing. A Term, Fact, Constraint or Derivation Rule used in definition of data processing. e.g. ▪ AccessLevel = Low if UserType = Public. [The definition in the OMG Business Motivation Model "An actionable rule derived from the policy" is comparable.]

2.2.2 Solution Descriptions and Plans

Business mission	What an organisation is about; its reasons for being; the essential products and services it offers customers.
Business vision	A high-level outline of an aspirational target state for an enterprise. "What an organisation wants to be or become." (Business Motivation Model.) The target state is supposed to meet the goals of one or more stakeholders. The state may be attainable and associated with specific objectives. The state may be unattainable, used only as a guiding principle for planning.
Mission statement	A declaration of mission, vision, main goals and values.
Target solution hierarchy	A solution defined at a vision level may be elaborated at increasing levels of detail. Solutions may be mapped to aims and directives.
Solution vision	An outline description of a target system, just enough to enable options to be compared and /or work to proceed. May be a response to a business problem or an elaboration of how to reach a business vision.
Solution outline	A high-level outline of a target system, produced after a first pass architecture definition, enough to pass risk assurance.
Solution to be built	A project-ready architecture, completed in sufficient detail for the project to be scheduled and resourced, and the building team to start work.
Plan	A document that defines the process to reach an aim. It should include timescales, costs and resources for each step. There can be a plan for an organisation, a project, or person, at any level of detail.
Plan hierarchy	Plans are often arranged in a hierarchy, increasing in number and in detail from strategic to tactical.
Strategy	"A plan to channel efforts towards achieving a goal". Business Motivation Model. A relatively high-level and/or long-term plan to reach a new state and so achieve some relatively high-level and/or long-term goals.
Programme plan	A plan for a programme of projects. In the context of architecture, the plan via which the enterprise architecture is developed and implemented.
Project plan	A plan for a project to develop and/or implement a solution. In the context of architecture, the plan via which a relatively self-contained solution architecture is developed and/or implemented.

2.2.3 Standards

Standard	A widely-accepted measure or set of qualities that is intended to increase uniformity between distinct systems and processes.
Standards Body	An enterprise with a mission to set standards and assess compliance to them. E.g. ▪ American National Standards Institute (ANSI). ▪ BCS The Chartered Institute for IT (BCS). ▪ Information Systems Examination Board (ISEB). ▪ Institute of Electrical and Electronic Engineers (IEEE). ▪ Information Systems Audit and Control Association (ISACA) ▪ International Standards Organisation (ISO). ▪ Office of Government Commerce (OGC). ▪ Open Applications Group Standards (OAGIS). ▪ Organisation for Advancement of Structured Information Standards (OASIS). ▪ The Object Management Group (OMG). ▪ The Open Group. ▪ US National Institute of Standards and technology (NIST). ▪ Software Engineering Institute (SEI).
Enterprise Standards Information Base	A repository contains standards recommended or used across the enterprise. Cf. The Open Group's SIB.
Profile	The standards (and perhaps options and parameters of those standards) necessary for a system, application or component to do its job.
Profiling	Selecting standards for a particular system, application or component.

2.2.4 Scope of Architecture Work

Scope of architecture work	The work needed to change a system may be divided between small changes handled through change management and big changes that require a substantial architecture effort. Architecture work has four dimensions of scope: ▪ Breadth: scope of the enterprise, system or solution. ▪ Focus: business, application or infrastructure change. ▪ Depth: the detail to which deliverables will be produced. ▪ Constraints on work.
Constraint (on work)	A factor that limits work to be done or potential solution options, such as time, budget and resources. (Not a constraint in the sense of a data type or business rule.)
Breadth of enterprise or system	This dimension of scope may be defined in several ways. ▪ Aim view: goal/objective/requirement catalogue ▪ Service view: a service catalogue. ▪ System view: a top-level context diagram. ▪ Process view: a top-level process map or use case diagram. ▪ Data view: a conceptual/domain/business data model.

Context Diagram (interfaces to external systems)	Shows a system as a 'black box', the inputs consumed and the outputs produced by that system, and the external entities (actors and/or roles) that send inputs and receive outputs.
External entity	An actor or role that inputs to and/or consumes outputs from a system or process. Defining external entities as actors tends to make a model more understandable. Defining external entities as roles tends to make a model more stable and flexible.
Actor	An identifiable individual external entity that plays one or more roles in relation to a system or process. May be a person, a human activity system or a software system. E.g. BACS, salesforce.com, a sales executive, a customer, an auditor.
Role	A part played by one or more actors in relation to a system or a process. With software systems, the part is often to enter or consume data and so named after the input or output data flow. E.g. loan applicant, expense claim approver, auditor.

2.2.5 Requirements

Requirement statement	A statement of need with which compliance must be demonstrated; this implies an acceptance test and an acceptance authority. Often expressed as an entry in a requirements catalogue or a use case description. Requirements attributes are likely to include: reference number, description, source, owner, type, priority, and deadline. A requirement should be SMART, which implies the definition of acceptance tests.
Functional requirement	A requirement related to data into or output from a system, and to processes and business rules for input and output.
Audit requirement	A requirement to do with ensuring an auditor can find the when/where/how/who of a process or stored data, and can replay events. (Considered by some to be a functional requirement and others to be non-functional.)
Non-functional requirement	A requirement about the ability of a system to perform its functions (whatever they are) effectively and efficiently. Usually quantitatively measurable.
Performance	Subdivides into two measures, often in opposition: ▪ Throughput: number of services executed in a time period. ▪ Response or cycle time (aka latency): time taken from request to response.
Availability	The amount or percentage of time that the services of a system are ready for use, excluding planned down time.
Recoverability	The ability of a system to be restored to live operations after a failure.
Reliability	The mean time between failures. Usually applied to the technologies in the Infrastructure, ignoring the more likely risk of application failure.

Integrity	[A term with several meanings defined under Data integrity and Data flow integrity.]
Scalability	The ability of a system to grow to accommodate increased work loads.
Security	The ability of a system to prevent unauthorised access to its contents.
Serviceability	The ability of operations team to monitor and manage a system in operation.
Usability	The ability of actors to use a system.
Maintainability	The ability of maintenance teams to revise or enhance a system.
Portability	The ability to move a component from one platform to another, or convert it to run on another platform. In practice, it can be difficult to set or estimate this quality metric realistically.
Interoperability	The ability for subsystems to exchange data at the technical level using shared protocols and networks. Sometimes embraces data integratability.
Integratability	The ability of interoperable subsystems to understand each other, which requires either common data types or brokers to translate between data types.
Extensibility	A synonym of maintainability.
Service Level Agreement (SLA)	"Written agreement between an IT service provider and customer (s) that documents agreed-to service levels." (ITIL) A document defining the requirements that services must meet, often focused on non-functional requirements.
Service Level Requirement (SLR)	"Criteria for level of service required to meet business objectives." ITIL

2.2.6 Regulatory Requirements

Regulatory requirement	A law or other regulation that adds to the requirements for or constraints on any solution to be developed or maintained.
IT accountability and procurement regulations	Regulation that makes public sector, IT directors and CIOs accountable for justifying investment in IT and for fair procurement from suppliers. US legislation of this kind was the stimulus for many early enterprise architecture initiatives: Government Performance Results act (GPRA) of 1993, P.L. 103-162. Federal Acquisition Streamlining act (FASA) of 1994, P.L. 103-355. Information Technology Management Reform act (ITMRA) of 1996, Division E, P.L. 104-106. Various EU directives have followed suit.
Data protection and freedom regulations	UK's data protection act: http://www.opsi.gov.uk/ACTS/acts1998/19980029.htm Freedom of Information act 2000: http://www.opsi.gov.uk/ACTS/acts2000/20000036.htm

Disability and accessibility regulations	UK Disability Discrimination act: http://www.opsi.gov.uk/acts/acts1995/1995050.htm. W3C Web Content Accessibility Guidelines: http://www.w3.org/TR/WAI-WEBCONTENT/ US Americans with Disability act.
Shareholder protection and audit regulations	US Sarbanes-Oxley act of 2002. Basel II.
Intellectual property rights regulations	International and national laws protect people and enterprises from theft of intellectual property.

2.2.7 Business Case [See Migration Planning for definitions]

Business case (before architecture)	Should be outlined at the start and updated as need be. It will be reviewed and refined several times while architecture work is done. It may decompose into business cases for specific options, stages or projects within the overall solution. [See the Migration Planning section for further definitions of this the supporting terms below. ▪ Return on Investment (ROI) ▪ Cost-benefit analysis ▪ Solution options ▪ Risk analysis ▪ Gap analysis (options) ▪ Trade-off analysis.]
Business scenario	A process, or story, to which are attached details of the actors, applications and technologies involved. A good way to create and present an architecture description. May be defined to support a solution vision or business case. May be defined during business architecture definition. May be presented as an example instance of a business process.

3. Architecture Frameworks

This section is about frameworks designed to help people create architecture descriptions and use them to good effect.

3.1 Foundation Terms and Concepts

Architecture Framework	A structured collection of guidance and techniques, a methodology, designed to help people create architecture descriptions and use them to good effect. A comprehensive framework contains: ▪ a development process (a process framework) ▪ a classification of architecture descriptions (a content or documentation framework) ▪ advice on organisation.

3.2 Intermediate Terms and Concepts

3.2.1 Architecture Process Frameworks

Architecture process framework	A description of a process that develops a target architecture to meet some requirements, under some constraints, plans the move from the baseline state to the target state, and governs that change.
Architecture state	A baseline architecture describes a system in a state, perhaps operational, ready to be reviewed and/or revised. A target architecture describes a system in a state that is to be created and implemented in the future. An intermediate or transitional architecture defines a state of a system between baseline and target.
The Open Group Architecture Framework (TOGAF)	A well-known framework for enterprise architecture, centred on a process called the architecture development method (ADM).
Architecture Development Method (ADM)	The core of TOGAF. A step-by-step process to develop and use an enterprise architecture. Designed more for enterprise architecture than solution architecture. 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.

Avancier Methodology (AM)	An architecture process framework designed more for solution architecture than enterprise architecture. The process is presented in 9 phases, though iteration and parallelism is expected. 1. Define precursors 2. Scope the work 3. Understand the baseline 4. Review non-functional criteria 5. Outline the target 6. Select and manage suppliers 7. Plan the migration 8. Hand over 9. Govern the migration.

3.2.2 Architecture Descriptions

Architecture description hierarchy	In describing the structure and behaviour of an enterprise or system, it is common to build a three level architecture description. Architecture deliverables contain artefacts, which are in turn composed from architectural entities. Conversely: an entity can appear in several artefacts; an artefact can appear in several architecture deliverables.
Architecture deliverable	A report that an architect is required to deliver. Deliverables include management and project documents as well as architecture descriptions that contain architecture artefacts. Deliverable examples: Request for Work, Statement of Work, Architecture Requirements, Architecture Definition, RAID Catalogue, Migration Plan.
Architecture artefact	A list, hierarchy, table, diagram or model that names and relates architectural entities. It describes the entities to some extent, though they are usually documented separately. Artefact types (aka view points) include: 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.
Architectural entity	A discrete architectural element, an object in an architecture repository that is reusable in different artefacts, and is definable using a standard template. An entity instance may be decomposed into finer-grained instances of the same type.

	Most architectural entity types can be classified as belonging to an architecture domain. For example: 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	An artefact, view or model that relates items in different structures, made for the purposes of ▪ gap analysis, ▪ impact analysis, ▪ requirements traceability or ▪ cluster/affinity analysis. Artefacts that take the form of mappings between architectural entities include: ▪ organisation unit to business function, ▪ business function to application, ▪ application to platform technology. ▪ data entity to business function, ▪ data entity to data store, ▪ data entity to data quality.
ISO/IEC 42010	Recommended Practice for Architecture Description of Software-Intensive Systems. A standard for software architecture or system architecture. It focuses on the description of an architecture as the concrete artefact representing the abstraction that is software architecture or system architecture. Commonly known by its original identity ANSI 1471.
View	The term used in ISO/IEC 42010 for an instance of a broad architecture domain description or narrower architecture artefact. “a representation of a whole system from the perspective of a related set of concerns, to demonstrate to one or more stakeholders that their concerns are addressed in the design of the system.” Views are decomposable. Views can share content. Views can contain parts of other views. An instance of a view point. A description that hides irrelevant details or facets of the system described. E.g. A logical data model shows the scope and structure of data stored by an application, but shows nothing of processes or technologies.

View point	The term used in ISO/IEC 42010 for a type of architecture domain or narrower architecture artefact. "what views of the same kind look like… a schema or template… describing the purpose and intended audience of the view" Defines a view's scope (the concerns addressed) and style (documentation conventions). Should be stored for reuse by architects in the same organisation. E.g. Logical data models drawn using the IDEF1X standard can address concerns shared by systems analysts and database designers.

3.2.3 Architecture Models and Languages

Model	A description that hides some details of the things described. Often, an architecture artefact or view that is drawn in the form of a diagram. A limited representation of entities and events in the world that is monitored by a system.
Idealisation hierarchy	The idealisation hierarchy of conceptual model, logical model and physical model.
Conceptual (or domain) model	A logical model that defines terms and concepts in a business or problem domain without reference to any computer application. In MDA, a computation-independent model (CIM).
Logical model	A model that excludes details of physical implementation. In MDA, a platform-independent model (PIM) that is unrelated to a specific technology (though may be related to an industry standard).
Physical model	A model that includes details of physical implementation. In MDA, a platform-specific model (PSM) that is related to a specific technology, and may include specific infrastructure functions.
Model-Driven Architecture (MDA)	A vision of the Object Management Group that encourages vendors to develop tools that help transform a conceptual model to a logical model, and a logical model to a physical model, and the reverse. A transformation may be forward engineering or reverse engineering.
System modelling techniques	Types of diagram used to model a system's structure or behaviour. Most of the diagrams and notations used by architects emerged out of ways to define software system architectures. Divided by BCS into structured and UML variants. Models commonly used by architects include process models, data models, context diagrams, use case diagrams, data flow diagrams, and interaction/sequence diagrams.

Modelling language	A standard that defines box shapes and line symbols for drawing node-arc diagrams to represent relationships between things.
Integration DEFinition (IDEF) language	A modelling language in the field of systems and software engineering. Includes IDEF0 for system or process structures and IDEF1X for data structures.
Unified Modelling Language (UML)	A modelling language maintained by the Object Management Group. Designed to help in object-oriented software design, though often used outside of that domain. Includes use case, activity, class and sequence diagrams.
ArchiMate	A modelling language maintained by the Open Group. Designed to help in architecture description. Components, interfaces and services are shown in distinct boxes. Overlaps with UML but mostly more abstract.

3.2.4 Architecture Description Structures

Architecture repository	An information base used by architects. A system that holds and manages all the meta data that describes an enterprise and its information systems. The content of the repository can be categorised in many ways, including tabular classifications such as the Zachman Framework and the Enterprise Continuum.
Zachman framework	"A logical structure for classifying and organising the descriptive representations of an Enterprise that are significant to managers and to developers of Enterprise systems." Drawn as table or grid: the 6 columns are primarily analysis questions. But they are also interpreted as architecture domains (data, process, network etc.). The 6 rows are primarily levels of idealisation-realisation from highest level context to operational systems, but they are also interpreted as stakeholder groups and architecture viewpoints. Zachman says the rows should not be interpreted as levels of decomposition.
Enterprise continuum	A structure for architecture description documentation used in TOGAF. A classification scheme for the contents of an architecture repository. Drawn as a table or grid, the rows are similar to those in the Zachman Framework. The columns are a spectrum from universal to unique. The 4 columns represent generalisation-specialisation, ranging from universal to bespoke or uniquely configured. ▪ Foundation: Structures and items that are universal. ▪ Common systems: Structures (composed of foundation items) that are used across most business domains. ▪ Industry: Structures and items used by enterprises in one business domain (say Telecoms or Banking). ▪ Organisation: Structures and items specific or bespoke to a single enterprise. The 4 rows represent levels of idealisation. From top to bottom:

	▪ Requirements and context. [See architecture precursors.] ▪ The architecture continuum. [Defined below.] ▪ The solution continuum. [Defined below.] ▪ Deployed solutions (architecture implementations).
Architecture continuum	A higher-level spectrum in the enterprise continuum, which contains logical or vendor-neutral specifications of requirements. It corresponds to the logical model level of the idealisation hierarchy.
Solutions continuum	A lower-level spectrum in the enterprise continuum, which contains specifications of products and services that implement the logical specifications. It corresponds to the physical model level of the idealisation hierarchy.
Reference model	A relatively abstract model people use as a guide in creating their own more specific model. Typically, a structure of components, services, processes or data entities. Sometimes for a specific industry or business domain. A kind of design pattern.

4. Business Architecture

This section is about one of the primary Architecture Domains defined in the first section of this reference model.

4.1 Foundation Terms and Concepts

Business goal	A goal of an enterprise or organisation. [See Goal.]
Business objective	An objective of an enterprise or organisation. [See Objective.]
Enterprise	A business or organisation with common goals and budget, in the public or private sector. A part of the real world that is directed and controlled by a management board to some human purposes, in which players cooperate to meet goals and objectives. Usually the highest level of an organisation, spanning several relatively distinct organisation units. Usually a human activity system that involves people, processes and technologies. Usually maintains resources and properties. Usually governed according to principles and policies.
Business	An enterprise or organisation that offers products and services to its customers in return for payment or funding.
Organisation	A synonym for the management structure of an enterprise or business.
Organisation unit	A physical subdivision of an enterprise's organisation. Usually given a manager, goals, budget and employees.
Management structure	The structure of organisation units defined by its managers. Usually a hierarchy decomposed to bottom level (elementary) organisation units. Usually shows the reporting line up from each bottom-level manager to the management board. Sometimes a matrix.

4.2 Intermediate Terms and Concepts

4.2.1 Business Architecture Structure and Behaviour

Business function catalogue or portfolio	A list of business functions, usually arranged in hierarchical structure. Functions may be arranged in two or more hierarchies. Functions may arranged in a matrix in which the rows or columns may titled function, capability or domain. To validate and complete the business process and business function structures, some decompose them to the same level so that every elementary business process step is also an elementary business function.

Business function	An idealised or logical subdivision an enterprise's capability. An encapsulation of the activities and resources needed to produce a cohesive set of services and/or products, for internal or external consumption. It may map to one or more physical organisation units. E.g. Marketing, sales, customer services, security, emergency response. (The examples in this section overlap because the concepts overlap.) Bottom level (elementary) business functions are commonly mapped to business process steps, to data entities and to organisation units.
Core business functions	A business function that is focused on the development, marketing, sales, creation and delivery of business products and services (as opposed to a support business function).
Core competency	A business function or capability that differentiates one business from another, if only in the manner that the activities are carried out. It should be difficult for competitors to imitate.
Support business function	A business function that serves core business functions. E.g. personnel, procurement or finance. Often similar in different businesses, so an obvious candidate to be out-sourced and/or delegated to a "shared service". (Analogous at this level of definition to a reusable application component, but very different in practice.)
Business capability	A business function whose performance is the subject of management attention - as for example in Capability Maturity Models or in Capability-Based Planning. It is usually a high-level and cross-organisational business function. E.g. legal compliance, customer service, security, emergency response. (The examples in this section overlap because the concepts overlap.)
Capability-based planning	Planning in which the focus of managers is to establish or improve a business function regardless of current organisation unit boundaries. The end result of capability-based planning may be that a business capability is given a manager and becomes an organisation unit.
Business domain	The primary business function of an enterprise or organisation unit. Classifies an enterprise or organisation unit by the services it offers and/or the expertise it has. Sometimes a market segment. Sometimes a public sector domain (say tax collection) that recurs only in different nations. E.g. law, employment law, telesales, insurance, airline operation, airline maintenance, security, emergency response. (The examples in this section overlap because the concepts overlap.)
Business process	A process that is important to an enterprise, leads to a goal of the enterprise, or involves people in the enterprise.

Value stream	A chain of activities; an end-to-end business process that produces a result valued by the customer. Products pass through the activities, gaining value at each step. A key concept in the Six Sigma framework, which is designed to remove wasteful activities and optimise processes in product manufacturing industries. Often shown as a naïve end-to-end business process model, lacking the formality required for it to be automated.
Value chain	A particular kind of value stream featuring core business functions (inbound, operations, outbound, marketing, sales and service) and support business functions (HR, R&D, Procurement). Introduced in "Competitive Advantage: Creating and Sustaining Superior Performance" Michael Porter.
Business service (business sense)	A service or product provided by an organisation unit or business function to its customers, internal or external. Defined for its consumers by an interface or service contract, rather than by internal processes needed to deliver it. [Analogous in SOA to a software business service, but at a completely different level of different analysis and design.]
Service-oriented architecture (business sense):	An approach that divides a business into distributed components that offer business services to each other and to customers. Each component (business function or organisation unit) may be defined by a Service Level Agreement. [Analogous to SOA in software systems, but very different in practice.]
Business data model	A conceptual model of business terms and facts, independent of any specific computer application. The entities in the model represent things in the real world. May also be known as a domain model. It usually takes the form of a data structure that defines business terms and concepts in the form of entities, attributes and relationships.
Business semantics	A definition of business rules: that is, terms, facts, constraints (on data values and process steps) and derivation rules (for data items). May be recorded in a data dictionary, a business data model and/or the pre and post conditions of processes.
Business model	A term used by business managers to mean various things other than a business architecture diagram. 1. The way an enterprise delivers products and/or services to its customers, determined by its business strategy. 2. The operating model, sometimes expressed in terms of how far the business processes are or should be standardised and integrated (after Ross, Weill and Robertson). 3. A what-if model of business operations, perhaps in spreadsheets or animated workflow models.

4.2.2 Business Process Decomposition and Automation

Process map	A top-level picture of a business in which its processes are named. The processes are not connected, or connected by dependencies rather than control flow. Might be drawn as a use case diagram. Might be arranged in swim lanes.
Business process decomposition	The hierarchical decomposition of a process into lower-level processes. Every business process step may be defined as a process in its own right. To validate and complete the business process and business function structures, some decompose them to the same level so that every elementary business process step is also an elementary business function.
OPOPOT	One person, one place, one time. A rule of thumb used to define the bottom level of business process decomposition.
Process automation hierarchy	The hierarchy within an enterprise, system or project of business processes use cases automated services. [See applications architecture for further discussion of process decomposition.]
Information system service	A use case or automated service provided by one application to another, or to an end user. [See applications architecture for further definitions.]
Workflow	1: the assignment of business process steps to actors in an organisation. Or 2: the logic or control flow of a process. Or 3: a technology that helps people to define or change one or both of the above.

4.2.3 Design for Business Security

Design for human and organisational security	Definition of all the things that can be done outside of software systems to secure business information, such as security guards, locks, definition and roll out of policies and procedures.

5. Data Architecture

This section is about one of the primary Architecture Domains defined in the first section of this reference model.

5.1 Foundation Terms and Concepts

Entity	A structural thing that persists. A thing that is created, affected and ultimately destroyed by events.
Event	A behavioural thing that happens. May create or destroy an entity, or move an entity from one state to another in its lifecycle. May affect several entities.
Data	Can be divided into structured and unstructured data. [This reference model focuses on structured data.]
Information	1: Data at the point of use by an actor in a business system. Or 2: unstructured data as opposed to structured data.
Structured data	One or more items (atomic facts) that describe one or more entities or events. Contained in data flows and data stores.
Data item	An elementary unit of information, a fact about an entity or an event. An attribute of an entity in a data model or an event in a data flow structure. A variable containing a value that is constrained by a type.
Data structure	A structure that arranges data items in one or more groups. May be defined in logical model or at a physical level in an XML schema or a database schema.
Data entity	The representation of an entity as a data structure that persists in a data processing system. Often records the state of a business process.
Data event	The representation of an event as a data structure in a data flow input to one or more data processing systems.
Data lifecycle	The life of a data entity expressed in terms of the states it passes through from creation to deletion, and data events that cause state transitions.
Type	A generalisation; a form or structure common to several things; a constraint on an instance shared by all instance of that kind.
Data type	A type that defines the properties shared by instances of a data item or larger data structure. It constrains the values of the data. It defines the processes that can legitimately be performed on the data.
Data type (primitive)	A data type defined in a programming language. e.g. alphanumeric string, integer, floating-point number (decimal), and boolean.
Data type (user-defined)	A data type defined by systems analysts that is bespoke to the business at hand. e.g. customer, order, product.
Constraint (rule)	A business rule that limits the values of a data type.
Derivation rule	A business rule that defines how the value of a data item is derived from the value of one or more other items (a special kind of constraint).

Meta data	Data that describes data. Includes data structures, data types, business rules, data locations and data qualities.
Data dictionary	A catalogue of data item types, which may include business rules.
File	A synonym of data store; any identified collection of data stored in the computer. May be used in a data flow. May be saved for future use. Sometimes means more specifically a "flat file", a data store in which data is stored and accessed in sequence, starting at one end.
Data source	Any data store, actor or component from which data is received by an application.

5.2 Intermediate Terms and Concepts

5.2.1 Unstructured data management

This entry is out of scope for examination purposes. The choice of content, document and knowledge management applications is as much or more a matter for applications architects than data architects.

Content management	The organisation, processes and tools for producing, storing, editing, sharing and searching any unstructured data. Roles can include creator, editor, publisher, administrator (managing access permissions etc.) and consumer, viewer or guest.
Document management	A subtype of content management focused on electronic documents and document images. Usually includes processes for: ▪ Capture, indexing, attaching meta data ▪ Storage, security ▪ Searching, retrieval ▪ Distribution, publishing ▪ Collaborative, configuration management Often associated with workflow systems.
Knowledge management	A subtype of content management focused on the knowledge of an enterprise and on capturing lessons learned. Typically supported by collaborative applications.

5.2.2 Data Architecture

Data in storage	Those aspects of data architecture relating to data that persists in a location.
Data store	A data structure that is held in persistent memory. Any file or database from which data can be extracted by an application. You can define the state of any data store (be it a database, cache or component) in a data model (though it may contain only a flat list of attributes).

Data model	A schema that groups data items into a data structure and defines the type of each data item. A structure that defines the attributes of entities and the relationships between them. It may include derivation rules for some data items. A business data or conceptual or domain model is a vehicle for documenting business semantics. A logical data model is a definition of the data that must persist for the processes of an application to work.
State	The data structure maintained inside the memory of a process or component.
Database	A persistent data structure that can be accessed by applications. Usually accessed via a database management system that enables direct (rather than serial) access to any part of the data structure. [See section 12: Enterprise Technology Classification for definition of database management system.]
Cache	A local store of data that has been copied from a master data store, usually for the purpose of speeding up response or cycle time.
Data in motion	Those aspects of data architecture relating to the movement of data.
Data flow	A data structure that is transported from sender to receiver. It is carried from data source to destination in a message, file, report or other data transport vehicle.
Regular expression	A hierarchical structure of elements arranged so that every element is part of a sequence, or is an option of a selection or is an occurrence of an iteration. You can define the every data flow structure as a regular expression (after Kleene's theorem). Though many messages contain no more than a list of data items. [You can also define a process structure as a regular expression, under a logical control flow in which loops and alternative paths are governed by conditions.]
Data format	A format or language for presenting data flow structures. E.g. Comma Separated Values (CSV), Extensible Mark Up Language (XML).
Data format standard	A standard for the content of data flow structures. E.g. EDIFACT, domain-specific XML Schema
Canonical data model	The "one true definition" of data types (e.g. customer address, order value, tax reference number) used by an enterprise. A logical data model that defines the data types that appear in messages between applications, and in the signatures of automated business services. Usually applied to data in data flows, as a standard for integration of applications, but could apply to data in data stores as well. A canonical data may be defined at a physical level using XML schema and other data format standards.

5.2.3 Data Qualities and Integration

Data quality	A characteristic of a data item, data structure or data store. Notably: Confidentiality, Integrity and Availability (CIA).
Data integrity	Data integrity (1): A data item has the same value in every part of a distributed system. A fact (e.g. customer name) has the same value in all locations that data item is stored. Data integrity (2): A data item obeys relevant business rules, sometimes in relation to another data item. The value of a data item is consistent with all invariant business rules e.g. an order must be for a known customer. Data integrity (3): A data item accurately represents a fact about an entity or Event. The value of a data item in a data processing system is consistent with a fact in the real world. Data disintegrity is a problem. Data integrity solutions can involve one-off data quality improvement exercises, data warehouses and master data management.
Data flow (or message) integrity	The requirement that a data flow has the same data content when it reaches its destination as it did when it left its source.
Data dissemination view	A view showing the dispersal (and perhaps duplication) of data between data stores or locations. Useful in analysis of change impacts, data mastering and security vulnerabilities.
Data warehouse	A kind of database system designed to hold a non-normalised data structure that is optimised for the production of management information reports.
Master data management	The systems and processes that enable an enterprise to maintain and/or find one “master” version of any data item or data structure, typically customer or product data. Supported by a range of approaches and technologies, including middleware technologies that hide the reality of multiple disparate data sources from data consumers.

5.2.4 Design for Data Security

Data security	1: Confidentiality alone. Or 2: a combination of Confidentiality, Integrity and Availability. Tom Peltier suggests rating the security level of a data item, data structure or data store as equal to the highest of the individual ratings (high, medium, low) awarded for Confidentiality, Integrity and Availability.
Security protection	Prevention of access to data designed to maintain the required data qualities of confidentiality, availability, and integrity.
Security feature	A feature of a system that enables its data and processes to be protected. E.g. Encryption, Checksum, https.
Security policy	A policy that defines which actors have (or do not have) Access rights to objects in a given domain - along with any other protections.

Information domain	A uniquely identified set of objects with a common security policy. Access to any data within the domain limited and constrained by the same rules.
Identity	One or more data items (or attributes) that uniquely label an entity or actor instance. E.g. passport number or user name.
Encryption	A process to encode data items (in a data store or data flow) so that they are meaningless to any actor who cannot decode them.
Checksum	A redundant data item added to a message that is the result of adding up the bits or bytes in the message and applying a formula. This enables the receiver to detect if the message content has been changed. It protects against accidental data corruption, but does not guarantee data flow integrity, since it relies on the formula being known only to sender and receiver.
Digital signature	A cryptographic scheme that simulates the security properties of a handwritten signature. More secure than a check sum, it is said to guarantee the data flow integrity of a message, since the signature is corrupted if the message content is changed.

6. Software Architecture

This section is about the internal structure of applications defined in applications architecture.

6.1 Foundation Terms and Concepts

Modular design	The design-time organisation of a system into components, or a process into sub-processes. You can divide a system or process into smaller modules by top-down decomposition of its structure or behaviour. You can compose entities or processes into larger modules by bottom-up cluster or affinity analysis.
Client	An actor, computer or software component that requests one or more services from a server. In software, the request is usually called an invocation.
Server	An actor, computer or software component that can provide one or more services in response to requests from a client.
Design time structure and runtime behaviour	Design time is when a structure of components is defined. Run time is when those components work and interoperate. An architect should have a good idea of how a design-time structure will behave at run time, since that is critical to meeting non-functional requirements.
Encapsulation	The enclosure within a component of processes, meaning that the inner workings are invisible to outsiders. Also the enclosure within a component of data, so the only way to access that data is by using the interface of the component. (Components whose state is persisted outside of the component often turn out to be pseudo components; they have an interface but do not encapsulate state, so external processes may read or update that data).
Cluster or affinity analysis	Techniques for finding and grouping items based on characteristics they share. An aim is to encapsulate cohesive items in one component, and minimise the couplings between components. Cohesive means closely related, inter-dependent or tightly-coupled by time, location, acquaintance, protocol or other ways. [See tight-coupling for more detail.]
Stateless	Characterises a component or service that has only a transient state. Its working storage is discarded at end of the process. It does not persist between invocations (or at least, does not remember any data it worked on last time).
Façade	A frontage to a building. An interface component that sits in front of a system and shields clients from some kinds of change to that system. Often, stateless. Used to reduce the coupling between client and server components. Used to aggregate services into a coarser-grained component.
Transactional	Characterises a service that can be completely and automatically rolled back if anything goes wrong. A desirable property in that it saves design effort and preserves integrity.

Delegation	One process or component calls on another to do the work, invoking it by passing a message (rather than by inheritance). May call a stateless façade.
Dependency	A relationship between two parties in which any change to the depended-on party implies an impact analysis of the dependent party. Often reflects a client-server invocation, or delegation to a server. Often represents a run-time relationship.
Cyclic dependency	A relationship in which two or more components depend on each other. Considered to be an architectural flaw - fragile and unstable - difficult to understand and maintain. Software with many cyclic dependencies is said to undermine testability, parallel development, and reuse. However, large-scale business components (e.g. Customer, Order, Product) are inevitably co-dependent.
Hierarchical (non-cyclic) dependency	A relationship in which higher-level components depend on lower-level ones, but not vice-versa. E.g. client components depend on server components, and application components depend on infrastructure components, but not vice-versa.
Service quality	A characteristic of a service in a well-designed architecture. A service conforming to Web Service standards has four such qualities: it is an abstraction, composable, loosely-coupled and defined by a contract. Beyond that, a well-designed service should be reusable, autonomous, discoverable and stateless. These eight qualities were suggested by Thomas Erl; and for simplicity, a service should be transactional as well.
Service-oriented design	A methodology that matches required services to available services. Required services are discovered though decomposition of high-level business process and use cases. Available services are discovered in some kind of services catalogue and invokable across a network via some kind of services directory.
Service-oriented design challenges	Obstacles to successful service-oriented design. Notably: service ownership, maintenance of shared services, versioning strategy. Governance of the service catalogue, avoiding vendor lock in, management of service-oriented design methodology across the enterprise. As a service consumer, you don't want service owner to change the interface unless you ask, or refuse to change it when you do ask.

6.2 Intermediate Terms and Concepts

6.2.1 Component Interfaces

Application Programming Interface (API)	An interface used to invoke the services of an application, or more usually, an application platform technology.
Interface Description Language (IDL)	A language for defining an API that is, ideally, independent of the technology used to implement the component behind the interface. Enables communication between components in different languages and running on different operating system. An interface generator (pre-compiler) reads the IDL to create a header file for use by client and server, and client and server stubs.
Realisation (software sense)	The act of providing an implementation for an interface. A service is an abstract thing until there is a system or component to implement it. One component may realise (publish or offer) more than one interface e.g. older and newer versions, or full and restricted list of services. One service can appear in several interfaces. Components require interfaces as well as offer them. A component with a required interface desires to meet a component with a matching offered interface.
Synchronicity	How work, divided between two components, is scheduled. The two components are called client and server below, but could be called sender and receiver, or prior and successor.
Synchronous	1: A request-reply style: a client must wait for a server to reply before continuing. (This is the usual invocation from one COBOL module to another or one Java object to another.) Or 2: A blocking style: a server serves one client at a time, turning away any other client who attempts to request a service. The caller and responder hold a channel open, blocking others from using it. (This is the usual invocation style in CORBA.)
Asynchronous	1: A fire-and-forget style in which a client does not wait for a server to reply to a request, carries on to do something else. A call-back mechanism may be needed. Or 2: A non-blocking style in which a server can accept requests from several clients before responding to the first request. The channel is released after a message is received. This implies the responder holds a queue of messages received. (This is the usual style in Web Services.)
Loosely-coupled	Communicating parties are not related by: ▪ Time: Parties communicate asynchronously and need not be deployed together. ▪ Location: Parties do not know each others' Locations Acquaintance: Parties do not know each others' names, or know only logical rather than physical names. ▪ Data types: The request or reply data has simple or weak data types. ▪ Operation types: Clients use the same names for different services. Interaction style: Messages rather than objects ▪ Control logic: Distributed (chain) logic

	▪ Transaction management: The service cannot be rolled back, compensating transactions may be needed. ▪ Version: Clients are not needlessly affected by upgrades to services. ▪ Protocol: A client can use any of several protocols to call a service.
Tightly-coupled	Communicating parties are related by some of these cohesion factors: ▪ Time: parties communicate synchronously and are deployed together. ▪ Location: parties know each others' locations. ▪ Acquaintance: parties know each others' names, or depend on physical names ▪ Data types: The request or reply data has complex or strong types. ▪ Operation types: clients use many different names for different services ▪ Interaction style: objects (perhaps requiring knowledge of an inheritance structure) rather than messages. ▪ Control logic: centralised (fork) logic. ▪ Transaction management: The service is roll-backable. ▪ Version: clients are needlessly affected by upgrades to services. ▪ Protocol: a client must use one protocol to call a service. ▪ (Combining lists published separately by Thomas Erl and Nicolai Josuttis).

6.2.2 Component Structures and Patterns

Component structure or pattern	A shape in which components are organised and interoperate to complete a higher level process.
Design pattern	A form or shape you can reuse in the design of different systems, to address similar issues. A common modular structure, or a pattern in how components communicate. Patterns often come in pairs; and there are trade offs between a pattern and its obverse. The use of design patterns encourages consistency and reduces the risk of reliance on an individual designer.
Hierarchical and peer-to-peer structures	Two contrasting ways to organise a structure of modules. A hierarchical design is one-way structure in which a client of party X cannot also be a server of party X. A peer-to-peer design has no hierarchy, meaning that any party can be both a client and server of any other.
Fork and chain structures	Two contrasting ways to divide a process between components, so they transform input into output in a series of relatively simple incremental steps. The fork pattern centralises intelligence about a process sequence. One controller supervises and orchestrates the procedure. It manages the sequence of activities by invoking components in turn. (Analogous to, if not the same as, a pattern known as process manager.) The chain pattern distributes intelligence about a process sequence. Each component does part of the work, then calls

	the next component. The system works by choreography rather than orchestration. (Analogous to, if not the same as, a pattern called pipe and filter.)
Model-View Controller (MVC)	A pattern for software modularity that separates user interface "views" from the state of the data entities that "model" the world monitored by the system. Either, controllers pass state-change events from the model to view-handlers that update the user interface (MCV). Or, controllers pass state-change events from the user interface views to the model that holds the state (VCM).
Controller	1. The manager or orchestrator component in a Fork pattern 2. A central component in the MVC pattern, the intermediary between the user interface and the data.
OO Design pattern	A pattern used in the design of object-oriented (OO) software. Usually solves a common design problem by placing a broker component between client and server components. A pattern for the modularisation of OO classes that is usually presented as a class diagram. This may be supplemented with an object diagram or interaction diagram. The most famous reference is "Design Patterns: Elements of Reusable Object-Oriented Software" [Gamma et al. 1995]. Some of their patterns are generalisable and useful outside of OO software, including Façade and the patterns below.
Singleton	A component (or class) with only one instance (or object).
Proxy	A surrogate for a distributed component. Used in distribution of code between different name spaces. Simplifies the interoperation of distributed components in DO and SOA interoperation styles, and in CORBA, DCOM and Web Services technologies.
Adapter	A wrapper that converts a provided service into a required service. Facilitates the reuse of existing technologies.
Observer	A component that monitors the state of another component. Curiously, the subject of observation may be an object.

6.2.3 Component Interoperation Styles

Component interoperation style	A paradigm, exemplar or pattern for how components interact. Four styles, varying from tightly-coupled to loosely-coupled, are DO, SOA, REST and EDA.
Distributed Objects style (DO)	A component interoperation paradigm in which an object on one computer invokes an operation on another object on another computer. In the classic distributed objects style: ▪ the interaction is request-reply ▪ the invocation is an object-method pair ▪ the server component is a stateful object (persists between operations). ▪ the server is synchronous (blocking) ▪ a connection is maintained while the service request is processed (The idea is to scale up object-oriented program design from one computer to many - the wider system looks and behaves

	like one object-oriented program - the programmer writes code as though all the objects are local, and call each other directly).
Service-Oriented Architecture style (SOA)	A component interoperation paradigm in which a process on one computer invokes a process on a remote computer by passing a message. This style is a contrast to the DO style, meaning that: ▪ the interaction is request-reply or fire-and-forget ▪ the invocation can be a plain operation call (rather than an object-method pair) ▪ the server component is stateless (rather than stateful) ▪ the server is asynchronous (rather than blocking) ▪ no connection is maintained while the service request is processed SOA is commonly associated with the use of web services, though these can be used to implement other styles.
Representational State Transfer style (REST)	A component interoperation paradigm in which a client identifies resources using a URI and invokes a process on that resource using only the generic operations in an internet protocol API – usually HTTP. A resource is anything can be found at a URL, including text and executable code. A client does not need to know the extent or structure of the resource set. A resource representation usually contains URLs that enable the client to navigate to related resources. A client does not need to know the names of any operations specific to any resource. A client can retrieve a representation of any resource using the GET operation, and update that resource using PUT, POST and DELETE operations.
Event-Driven Architecture style (EDA)	A component interoperation paradigm in which any component can read, or subscribe to receive, an event/message published by any other component. This is a fire-forget style. It means that senders and receivers are very loosely-coupled. It implies a mediator or shared memory communication style.

6.2.4 Component Communication Styles

Component communication style	The manner in which components interact (as client and server, or sender and receiver), which can be direct or via intermediaries. There are three broad categories: ▪ Point to point ▪ Introduction agent (aka direct broker) ▪ Mediator (aka indirect broker).
Point-to-point communication	A sender (or client) is coupled to a receiver (or server). More precisely defined by two features: 1) The sender is responsible for knowing or determining both the location of the receiver, and a protocol and data format the receiver understands. 2) A message is sent by one sender and received by one receiver. Strength: simple to implement. Weaknesses: potential duplication of data transformation and routing code, high

	configuration cost of receiver address changes. [See section 12: Enterprise Technology Classification for tools that implement this style of communication.]
Introduction agent (direct broker) communication	A direct broker helps parties to communicate. It decouples clients and servers, at least to the extent that the two parties can work in different places. It hides some complexities of the communication transport. The broker must register parties willing to communicate (end point registration). The broker can then establish initial connection when a client requests a service from a server. Subsequently, the parties communicate point-to-point via client-side and server-side proxies, which is faster than via a mediator broker. [See section 12: Enterprise Technology Classification for tools that implement this style of communication.]
Mediator communication	An indirect broker helps parties to communicate. It decouples clients and servers by sitting between them. It means the parties can work at different places and times (asynchronously). It can shield one party from some effects of some changes to the other. It does for components what email infrastructure does for people, that is, enable them to communicate asynchronously via messages - rather than talk directly over an end-to-end network connection kept open for that conversation.
Message broker (indirect broker)	An active mediator that brokers communication by forwarding messages from senders to receivers. It: ▪ provides a shared infrastructure for sending messages to recipients. ▪ hides some complexities of the communication transport. ▪ offers common command messages. ▪ registers end-point parties willing to communicate. ▪ enables routing: locates parties and sends them messages. ▪ enables transformation: converts data formats.
Message router	A message broker that sends a message where the sender directs. The client communicates the logical name of the server to the broker. The broker looks up the server that is registered under the logical name and passes the communication to the server. [See section 12: Enterprise Technology Classification for tools that implement this style of communication.]
Message bus	A message broker that is schema-based. It uses a common data format to reduce the cost of adding and removing communicating parties. It provides a set of agreed-upon message schemas, in a common data model.
Passive mediator	Any memory space or data store that is shared by two or more applications. Communicating parties post messages to it and/or read messages from it. Examples include serial files, databases and the “blackboard” OO design pattern.
Publish & Subscribe Distribution	Subscribers register their interest in receiving a message type with a message broker. Divides into topic-based and content-based

Topic-based publish and subscribe	Distributes messages depending on their subject line; this subdivides into broadcast and list based.
Broadcast-based publish and subscribe	A topic-based distribution. An event publisher creates a message and broadcasts it to the local area network. Each listening node has a service node that inspects the subject line. If the subject line matches a subject the node subscribes to, then the node processes the message. Else, the listening node ignores the message.
List-based publish and subscribe	A topic-based distribution. You identify a subject and maintain a list of subscribers for that subject. When an event occurs, the subject notifies each subscriber on the subscription list.
Content-based publish and subscribe	Distributes messages depending on the content of the message. Highly configurable, since any combination of information items can be used to direct messages, and this gives an exponential enlargement of logical routing possibilities.

7. Applications Architecture

This section is about one of the primary Architecture Domains defined in the first section of this reference model.

7.1 Foundation Terms and Concepts

Information system (IS)	A kind of system that produces information we humans can process. A data processing system. Can be a paper-based, mechanical or electronic system. Usually a computer application that processes data for use by people in a business system.
User	A role or actor outside the boundary of the system being used. A role or individual who uses an application. Usually human. "Person that daily uses IT services." ITIL
Application	A system of software components that supports a business function and/or maintains a data store. Supported by application platform technologies. Characteristics include size, owner, cost, value.
Business application	An application that helps its users to perform business-specific tasks. E.g. order entry system, management information system. Business applications may be divided into front-office or Business Support Systems (BSS) and back office or Operational Support Systems (OSS).
Generic application	An application that helps its users to perform generic office and administrative tasks. E.g. a browser, word processor, spreadsheet.
Platform application	An application that supports business applications – a component of the application platform. E.g. a message broker, DBMS or operating system.
Enterprise Resource Planning (ERP)	ERP is the planning of how enterprise resources (materials, employees, customers, etc.) are acquired, moved from one state to another. An ERP system is a operational and business support system that maintains (ideally in a single database) the data needed for some or all of Manufacturing, Supply Chain Management, Financials, Projects, Human Resources, Customer Relationship Management, Data Warehouse and Management Information.
Customer relationship management (CRM)	CRM is the development and maintenance of mutually beneficial long-term relationships with customers. It helps with some or all of the following: attracting customers, transacting business with customers, servicing and supporting customer, enhancing customer relationships. A CRM system is a business support system that helps this relationship management.

7.2 Intermediate Terms and Concepts

7.2.1 Applications Architecture Structure

Application catalogue or portfolio	A list of applications, usually arranged in hierarchical structure that reflects the business function hierarchy.
Application portfolio management	An organisation and processes designed to catalogue, describe, and value the applications of an enterprise, with a view to rationalisation or optimisation of those applications.
Applications architecture structural model	A diagram that shows applications and the data flows that pass between them. Typically drawn using some kind of data flow diagram. Where there are too many data flows, they may be abstracted into dependencies in some kind of dependency diagram.

7.2.2 Applications Architecture Behaviour

Applications architecture behavioural model	A diagram that shows how a process works through the interaction of users and applications. Often drawn using some kind of interaction diagram. Often used to examine where time is lost in or between application processing steps.
Information system service (repeat)	A use case or automated service provided by one application to another, or to an end user.
Use case	A process in which an actor uses a system: a sequence of transactions. Usually supports an OPOPOT business process step. Usually has one main path and several alternative (or exception) paths. The details of each step (including any automated services invoked) may be documented separately from the use case itself.
Automated service (SOA sense)	A software process that a client can invoke. May be classified as a business service or data service. Often a transaction.
Business service (SOA sense)	An automated service whose input and output data is defined in a canonical data model. Can be provided either by a broker application or an application that encapsulates a data source, since it is the interface that matters, not the deployment location. (Usually at a very much lower level of design than “business” business service.)
Data service (SOA sense)	An automated service whose input and output data items are defined according to the parochial physical data model of a specific data source.
Transaction	1: An exchange between a user and a computer in which the user inputs a command and receives result. Or 2: A process or unit of work that can be rolled back, for example if what was specified as a precondition is violated.
ACID	Atomic, Consistent, Isolated and Durable. The properties of a transaction in definition 2.

Compensating transaction	A backtracking, undo or correction process. It may undo updates committed to databases, remove messages placed in message queues, send follow-up correction messages, report cases of data disintegrity. A process to handle the side effects of regular process (or workflow) that started but could not complete successfully – where that process cannot be implemented as an ACID transaction.

7.2.3 Applications Integration

Application integration	The ends or means of connecting one application to another such that they contribute to the same output data flow, or store the same data. Often underpinned by one or more of the component interoperation and communication styles defined in the Software Architecture section.
Batch process	A kind of process that processes a collection of messages that have been accumulated in advance in a file or queue. (Opposite of on-line or transactional process.)
ETL	A pattern and/or set of tools for extracting (E) data from data sources, transforming (T) data items from one format to another, and loading (L) the reformatted data into data stores. Often requires data to be cleaned up before or after the transformation stage,
Application consolidation	The integration of applications by merger of distinct databases into one, so that the previously distinct applications can also be viewed as one. Not always a good idea, since a hub application can become a bottleneck in change management.
Point-to-point application integration	A pattern in which subsystems talk to each other directly (rather than via a central hub component, broker or bus).
Hub and spoke application integration	A pattern in which subsystems communicate via a central hub component, broker or bus (rather than talk to each other directly).
Boundaryless Information Flow	A trademark of the Open Group intended to express the vision of delivering any data, any time, any place to anybody who is authorised to view the data.
Integrated Information Infrastructure Reference model (III-RM)	A pattern in TOGAF for a service-oriented architecture. User applications invoke services provided by broker applications, which in turn invoke services provided by applications that encapsulate data sources.

7.2.4 Design for Applications Security

Identification	A process via which an entity or actor reveals their Identity. Usually followed by authentication.
Authentication	A process to confirm or deny that an actor is trusted - is the entity to which an identity was given. E.g. A password check. Usually followed by authorisation. Authentication of an actor produces one of four results: true positive, true negative, false negative (which leads to wrongly-denied access) or false positive (which leads to unauthorised access).
Three-factor authentication	Authentication that involves checking three facts about an identified actor. Factors can include something they: ▪ remember (e.g. password, mother's name), ▪ carry (e.g. credit card or key) ▪ are (e.g. biometric data.).
Authorisation	A process giving access to a trusted actor, based on that actor's known access rights. Usually followed by Access.
Access	A process to look inside a system to find data (or processes) of interest. Data can include files containing executable processes.

8. Design for NFRS

This section mentions only a selection of common techniques that could be relevant in answers to exam questions.

8.1 Foundation Terms and Concepts

None

8.2 Intermediate Terms and Concepts

Design for performance (response time and throughput)	The general advice is to look for bottlenecks and tackle them one by one. Poor performance is very often the result of poor design, such as needless distribution of processes, wasteful use of a network or accesses to a database, delays and failures caused by message queues filling up. Given a reasonable design, further optimisation often involves running processes in parallel.
Database optimisation	Techniques for optimising processes by eliminating needless database access. Four of many techniques are listed below.
Normalisation	Relational data analysis. A technique for defining a data store structure that assists data integrity by storing each fact once. It also optimises update processes by minimising redundant data storage.
Denormalisation	A design technique that optimises input and/or output processes by structuring a data store structure to reflect the most important input or output data flow structures, at the expense of duplicating some stored data.
Index	A list of pointers to elements of stored data. Usually used to optimise output processes. May be temporarily disabled to optimise on-line update processes (and updated later off-line).
Access path analysis	Study of the route a process takes through a data store structure. A very common source of performance problems is that an SQL programmer does not know the access path their procedure takes through a database. So it is advisable to use access path analysis and/or employ highly skilled SQL resources for critical database access programs.
Caching	Holding data in a temporary storage area - usually frequently-accessed data. Placing copies of persistent data in a location nearer to the user than the original data source. Generally good for response or cycle time. Can raise concerns about data integrity and security.
Scale up	Increase the power of one processing node. Usually means add resources (processors or memory) to a node on a computer network. Generally good for response time and throughput.

Scale out (aka clustering)	Increase the number of parallel processing nodes. Usually means add more nodes to a cluster. Usually requires some kind of load balancer to sit in front of the cluster and distribute service requests between them. Generally good for throughput. Not necessarily good for response time.
Design for resilience (availability and reliability)	The primary technique is to build redundancy into the system. E.g. to scale out, or add one to the number of servers in cluster that calculation or prototyping suggests is needed. Designers can also provide failover capability and/or defensive design and programming techniques.
Fail over	Automatic switch over to a redundant or standby computer server, system, or network upon the failure or abnormal termination of the previously active server, system, or network. Failover happens suddenly and generally without warning.
Defensive design	Designing a client component so that it does not fall over if a server component does not work properly, which implies asynchronous invocation. Or designing a server component so that it does not depend on its input data being valid, which implies testing all data types and business rules before processing. The opposite of "design by contract".
Design for recoverability	The principal technique is to back up and provide some kind of switch-over or fail-over procedure. Procedures must address also "fail back", to return operations from a disaster recovery site to the normal production site.
Back up	A copy of data that may be used to restore the original after data loss. Used in disaster recovery. Also used to restore individual files that have been deleted or corrupted. Backups are typically the last line of defence, coarse-grained and can be inconvenient to use.
Backup site	A location where systems are or can be duplicated. A cold site has no equipment. A warm site has infrastructure but no up-to-date data or software. A hot site has up-to-date software and more or less up to date copies of data.
Design for integrity	The two general techniques are to reduce data replication and ensure updates are made via ACID transactions. More specific techniques are: normalise stored data, switch on automated referential integrity checks, apply transaction management to update processes, remove caches, and consolidate distributed databases.
Design for serviceability	The principal technique is to instrument applications so that they report on what they are doing, and how well they are doing it.
Design for security	Design for security is addressed elsewhere in this reference model under the headings of business, data, applications and infrastructure architecture.
ISO/IEC 17799	Information technology: Code of practice for information security management.
ISO/IEC 24762:2008	Information technology — Security techniques — Guidelines for information.
ISO/IEC 27001	Information technology — Security techniques — Information security management systems — Requirements.

9. Infrastructure Architecture

This section is about one of the primary Architecture Domains defined in the first section of this reference model.

9.1 Foundation Terms and Concepts

9.1.1 Basic Infrastructure Components

Node	A node on a computing network. A physical node is a computer, switch, router, bridge, repeater or any other device attached to a network. A logical node is a web server, app server, database server or other platform for application software.
Computer	The primary component of IT systems. A system that can execute stored programs. A hardware device that contains a processor. May take the role of a client and/or server.
Processor (CPU)	The part at the heart of a computer that reads and executes elementary software instructions. The power or speed of a processor may be increased by various kinds of parallel processing.
Operating System (OS)	The bottom level platform application, which enables other applications to execute instructions on the hardware.

9.1.2 Network scopes

Network	1) Generally: a structure of links in which arcs connect nodes, lines connect boxes, relationships connect entities, or channels connect inter-communicating components. Or 2) More specifically: a set of communications links that enables one computer or electro-mechanical device to receive data sent by another. There are computer networks and phone networks.
PAN, LAN, MAN, WAN	Personal, Local, Metropolitan or Wide Area Network.
Virtual Private Network (VPN)	A network in which communication between nodes is carried by connections within some larger network (usually the Internet) instead of by physical wires. Uses a WAN but feels like a LAN. The link-layer protocols of the virtual network are said to be tunnelled through the wider network. Can reduce costs. Can raise security concerns.
Cloud computing	The provision of computer power, data storage and application services over the internet. Remote hosting of enterprise applications in a “virtual data centre” managed by a service provider.

9.1.3 Network topologies

Topology	The shape of a network or communication routes over a network.
Topology shape	A shape that connects nodes or constrains communications routes over a network. The four classic IT network shapes are point-to-point, bus, hub and ring. Another shape is a grid.

9.1.4 Network layers

Network Layer	A level in a hierarchy of communication layers. Five typical levels are outlined below.
Application or component connection level	The top level - where application components communicate. [See other sections for discussion of topologies at this level.]
Transport level	Manages the end-to-end control of message delivery. For example, determines whether all packets have arrived. May check for errors and ensure complete data transfer.
Network level	Handles the routing and forwarding of data at the packet level. Sends outgoing transmissions in the right direction to the right destination. Receives incoming transmissions.
Data transport level	The level at which data is transported around the physical network by network communications software. Provides synchronisation for the physical level. Does bit-stuffing for strings of 1's in excess of 5. Furnishes transmission protocol knowledge and management A topology at this level describes sequence and protocol that physical nodes use to communicate. E.g. Ethernet is based on a bus topology. First designed so all packets were sent to all nodes on the same network segment. Each node listens to all packets and filter out unwanted ones. Token passing (IEEE 802.5) is based on a ring topology. Each node connected to the next node. Nodes pass a message around in a circular fashion until it arrives at the intended destination.
Physical level	The bottom level, where nodes connect to a physical network medium. Conveys the bit stream through the network at the electrical and mechanical level - provides the hardware means of sending and receiving data on a carrier. A topology at this level reflects how wires connect nodes.

9.1.5 Network protocols

Protocol	The rules used by message senders and receivers when they exchange messages via transport mechanisms, by end points in a telecommunication exchange when they communicate. Rules may cover a standard format for the header that precedes the message, the footer than follows the message, and the sequence in which messages are exchanged.
Protocol Stack	Protocols are arranged in layers, corresponding to layers of platform technologies. The best known protocol stacks are probably OSI and TCP/IP.
OSI 7-layer Stack	A classification (not a formal standard) that divides telecommunication into seven layers, which technology vendors use to explain their products. **7: application layer:** identifies communication partners and quality of service, authenticates users, considers privacy, identifies constraints on data syntax. **6: presentation layer:** usually part of an OS that converts incoming and outgoing data from one presentation format to another (for example, from a text stream into a popup window with the newly arrived text). **5: session layer:** sets up, coordinates, and terminates conversations, exchanges, and dialogs between the applications at each end. It deals with session and connection coordination. **4: transport layer:** manages the end-to-end control. **3: network layer:** handles the routing and forwarding of data at the packet level. **2: data-link layer:** provides synchronisation for the physical level. **1: physical layer:** conveys the bit stream through the network at the electrical and mechanical level. (after Whatis.com)
TCP/IP 5 layer stack	An alternative to the OSI model which collapses the upper three layers and uses the same names for slightly different layers. E.g. **5: application layer:** DNS · FTP · HTTP · IMAP4 · POP3 · SIP · SMTP · RPC · TLS (and SSL) · SOAP· (more) **4: transport layer:** TCP · UDP · (more) **3: network/internet layer:** IP (IPv4 · IPv6) · OSPF · (more) **2: data link layer:** 802.11 (WLAN) · 802.16 · Wi-Fi · WiMAX · Token ring · Ethernet (more) **1: physical layer:** Ethernet physical layer · Modems · Optical fibre · Coaxial cable · Twisted pair · (more) (after Wikipedia)

9.1.6 The internet

Internet Protocol (IP)	A protocol to send data across a packet-switched internetwork.
IP address	An IP4 address is made from four numbers. (E.g. Binary: 10010110.11010111.00010001.00001001, or Decimal: 150.215.017.009.) The 4 numbers are used for two addresses: the network address (first 1, 2 or 3 numbers) and the host computer address (last 1, 2 or 3 numbers. IP6 numbers are not discussed here.
Subnet	The network administrator can divide the host part of the IP address so as to identify both a subnet and a host.
Convergence (of telecom. media)	The enabling of one operating platform to supply many media. The merger of telecom, data processing and imaging technologies. So fixed, mobile, and IP service providers can offer content and media services. Enables equipment providers to combine voice, data and images in services offered to the user.
Voice Over IP (VoIP)	Also known as IP Telephony (IPT). Promoted as offering lower network installation and management costs, lower voice phone tariffs and mobility of phone numbers.

9.2 Intermediate Terms and Concepts

9.2.1 Infrastructure services and components

Infrastructure service	A term that can be interpreted at several levels and in different ways. For example: A basic service such as computing power or memory, provided by real or virtual computers. A platform service. [See below.] An operational service such provided by an IT services management tool or organisation. [See “IT Service”.]
Platform service	A service such as transaction management or user access control, provided by one or more platform technology components to applications. [See TRM.]
Technology catalogue or portfolio	A list of technology components types in a baseline or target architecture, usually arranged in the hierarchical structure of an enterprise technology classification. [See ETC.]

9.2.2 Enterprise technology rationalisation

Technology rationalisation	A process for studying the services provided by a baseline technology infrastructure and defining a de-duplicated target architecture: 1. Classify baseline platform technologies [See ETC.] 2. Catalogue baseline technologies 3. Classify baseline platform services [See TRM.] 4. Catalogue baseline platform services 5. Define target platform services 6. Define target technology components 7. Plan baseline-to-target migration 8. Govern delivery of the change.
Enterprise technology classification (ETC)	A structure for a catalogue of technology components, with headings such as those below. ▪ Client (user access) devices ▪ Generic user applications ▪ Application platform ▪ Software development ▪ Integration tools ▪ Data management ▪ Servers ▪ Data storage ▪ Networks ▪ IT Services Management / operations ▪ Environment ▪ Security [See section 12 of this reference model for further detail.]
Technical Reference model (TRM)	A logical and hierarchical classification of the platform services provided by infrastructure technologies to applications. Can provide a requirement specification for technology rationalisation. For example, the service categories in the TOGAF Technical Reference Model include: ▪ Data Interchange Services ▪ Data Management Services ▪ Graphics and Imaging Services ▪ International Operation Services ▪ Location and Directory Services ▪ Network Services ▪ Operating System Services ▪ Security Services ▪ Software Engineering Services ▪ System and Network Management Services ▪ Transaction Processing Services ▪ User Interface Services ▪ Quality of Service
Virtual machine	Software that enables application programs to run above – decoupled from - the underlying operating system and/or hardware processor. Enables applications to be moved between different operating systems and/or processors; and enables server consolidation.

Server consolidation	A programme of work to deploy current or baseline applications to fewer servers, usually involving virtualisation.

9.2.3 Solution technology definition

Solution technology definition	A process that progresses through stages from a logical application-information view through progressively more physical views up to a hardware configuration diagram. A process for defining the technologies that will support and run an application that includes the activities such as: 1. Clarify the precursors, requirements and context 2. Establish baseline opportunities and target constraints 3. Define client-end devices 4. Define data servers 5. Define intermediate servers 6. Map software layers to platform and hardware tiers 7. Define the network 8. Iteratively refine to handle non-functional requirements 9. Define the environment strategy.
Hardware configuration view	1: A physical system; a structure of nodes connected via one or more networks. Or 2: A logical model of 1.

9.2.4 Connecting Applications to Networks

Network address	The address of a computer on a network; can be any of a variety of address types, mostly notably MAC address and IP address.
MAC address	A quasi-unique identifier (physical address) assigned to most network adapters or network interface cards (NICs) by the manufacturer for identification. If assigned by the manufacturer, a MAC address usually encodes the manufacturer's registered identification number.
IP address	A numerical identification (logical address) assigned to a node in a computer network that uses the Internet Protocol for communication between its nodes.
Service Type	A protocol for computer I/O (e.g. file transfer, web access, or email)
Port	A computer's network address has thousands of logical ports for sending and receiving data. Each port sends or receives data using one protocol or service type. An international standard defines default port numbers. E.g. An http: (unsecured) URL typically uses port 80. An https: (secured) URL typically uses port 443. An SMTP server typically uses port 25. A POP3 server typically uses port 110. However, the choice of port number is an architectural design decision. E.g. A security architect might ban the use of port 80 for http.
Socket	A socket (a software thing) is the use of a port to input/output a service type via a network. A socket is identified by a logical network address and a port number (which is used for a

	service type). E.g. Socket = port 80 for http: at IP address nnnn.nnnn.nnnn.nnnn.
Active process	A computer can run several applications (e.g. browser instance, email software) at once; each has a process number. A process can use several sockets, for different kinds of input and output. Several processes can use the same socket.

9.2.5 Design for Infrastructure Security

Design for infrastructure security	Techniques for protecting client and server computers from malicious access.
Client security	Features that protect client-end computers from malicious access.
Server security	Features that protecting server computers and databases from malicious clients.
Firewall	Software at the boundary of a network that is used to detect, filter out and report messages that are unauthorised and/or not from a trusted source.
De-Militarised Zone (DMZ)	An area of a network, usually between the public internet and the enterprise network. It uses firewalls to filters out messages that fail security checks. It contains servers that respond to internet protocols like HTTP and FTP.
https	The combination of a normal HTTP interaction over an encrypted Secure Sockets Layer (SSL) or Transport Layer Security (TLS) connection. This ensures reasonable protection of data content from those who intercept the data flow in transit. An https: URL may specify a TCP port. If it does not, the connection uses port 443 (whereas unsecured HTTP typically uses port 80).
Web site security	Usually, a process whereby a web browser checks the public key certificate of a web server at the other end of an https connection. The aims are to check the web server is authentic (who it claims to be) and that messages to/from with the web server cannot be read by eavesdroppers.
Public key certificate	An electronic document that enables a web server to accept https connections, or to verify that a public key belongs to an individual. It incorporates a digital signature to bind together a public key with an identity (the name of a person or an organisation, their address, and so forth).
Certificate authority	The web site administrator must get a public key certificate signed by a certificate authority. This signature certifies (authenticates) that the certificate holder is the entity it claims to be. Web browsers are generally distributed with the signing certificates of major certificate authorities, so that they can verify web-server certificates.

10. Migration Planning

This section addresses the process of turning baseline and target architecture descriptions into a plan for a programme or project, and the contributions made by architects to programme/project planning.

(Note that this section does not set out to compete with established management methods. The architect should integrate the activities below programme/project management offices approaches such as MSP, PRINCE2 and PMI).

10.1 Foundation and Intermediate Terms and Concepts

Business case	A rationale and business justification for spending time and money. Generally speaking, the essential elements are ▪ ROI (benefits – costs), ▪ Options (business or technical), ▪ Impacts (work to be done and changes to be made) ▪ Risks. These terms are defined separately in this reference model. There should be a business case for work to describe an architecture and/or to implement an architecture as operational systems. An outline business case is needed before architecture definition starts in earnest. It will be reviewed and refined several times later in the process, and perhaps decomposed into business cases for specific options or projects within the overall solution.
Return on Investment (ROI)	A statement of benefits gained minus costs spent. Costs must cover development, implementation, operation and maintenance. Benefits may include money made, money saved, regulations complied and the resolution of specific problems. E.g. the benefit of data integrity is to save the cost of data disintegrity.
Cost-benefit analysis	An assessment of the costs and the benefits of a course of action and/or a proposed system.
Solution Options	Alternative designs. It is usual, at least at the solution vision stage, to describe two or more alternatives. They may be compared at several stages and at several levels of design. The choice can be guided by: ▪ cost-benefit analysis, ▪ risk analysis, ▪ gap analysis and ▪ trade-off analysis.
Risk analysis	Analysis of vulnerabilities that threaten the ability of a target system to meet requirements, especially non-functional requirements, including security. Risk analysis is needed before architecture definition starts in earnest, and then several times later in the process, and at several levels of design.
Gap analysis (options)	Generally, a technique for comparing two similar lists or structures, to find potentially missing items. It can be used to compare two optional solutions, and identify gaps in one or

	both. It helps if the two options are presented under the same structure as each other, or a more general structure.
Trade-off analysis	A process in which a consultant leads analysis of target system options and the trade offs between them. Published and promoted by the Software Engineering Institute of Carnegie Mellon University.
Business scenario	[See the definition in section 2.]

10.2 Practitioner Terms and Concepts

Gap analysis (baseline-target)	Generally, a technique for comparing two similar lists or structures, to find potentially missing items. Often, it means comparing the components or services of a baseline system with those of a target system; and the result of this analysis informs programme planning.
Migration path or plan	Sometimes a synonym for a roadmap, but better used to mean less - only progressive series of architectures describing different states of an enterprise or system, from baseline to target.
Roadmap	A migration path/plan with timescales, and perhaps some idea of costs and resources. Half-way between a migration path and a project plan.
Critical path analysis	A technique to construct a model of the project that includes (i) a list of all activities required to complete the project (also known as work breakdown structure) (ii) the duration of each activity, and (iii) the dependencies between the activities.
Program Evaluation and Review Technique (PERT)	A method to analyse the tasks involved in completing a given project, especially the time needed to complete each task, and identifying the minimum time needed to complete the total project.
RAID catalogue	A catalogue of risks, assumptions, issues and dependencies, maintained separately from requirements and from solution documentation. Cf. Risk Register in PRINCE2.
Risk	A potential problem; an event that will cause an issue if it occurs.
Assumption	Statement that, if not true, could turn into a risk or issue that threatens the success of a project.
Issue	A problem that needs resolution. Sometimes the realisation of a pre-identified risk, or an assumption that turned out to be false.
Dependency (risk sense)	A dependency of a project upon an external actor or deliverable, not under the management of the project manager.
Management methodology	A collection of processes and deliverables designed to guide people in how to complete a programme, project or service
Programme	A set of projects that are related by a common goal or shared budget, usually under one manager.
Managing Successful Programmes (MSP)	A methodology for managing programmes, maintained and published by the OGC. Applicable at the level of enterprise architecture.
Project	A process that consumes time and resources to deliver a required outcome, usually under one manager.
PRINCE2	A project management method. A well-known methodology maintained and published by the OGC. Applicable at the level of an application development project.

11. Architecture Management

This section addresses the organisations and processes needed to govern and implement an architecture description, in development and operation, including the management of changes.

11.1 Foundation and Intermediate Terms and Concepts

None

11.2 Practitioner Terms and Concepts

11.2.1 Architecture Implementation

Architecture implementation	The realisation of an architecture description as a system, through development and deployment. This requires programme and project management organisations and processes. It uses tools (e.g. for source code management, unit testing load testing, regression testing, security testing, and compliance testing).
Software Development Life Cycle (SDLC)	A solution development process centred on software engineering. There are agile, iterative and waterfall variants.
Waterfall	A development process that is sequential: usually analysis, design, build, test and roll out. Engineers proceed from one kind of work to the next without significant iteration or parallelism between stages.
Iterative Development (aka Incremental Development in DSDM)	A development process that proceeds by increments, meaning that a working subset of the full solution is delivered as early as possible. Not necessarily agile. E.g. The Unified Process is iterative, but not fully agile. It is loosely associated with UML. (RUP is a commercial variant embodied in CASE tools from IBM/Rational.)
Agile Development	A solution development process that is not only iterative, but also flexible about the requirements, the solution and the process being followed. The many varieties are characterised by short-cycle iterative development, early testing for usability and performance, and flexible requirements. User involvement and feedback is a mandatory prerequisite in agile development.
Transition	Once the architecture has been realised in the form of an operational system, that system is usually handed over to two organisations. Transition into Operations. The production or run-time system is handed over to be run by some kind of managed operations organisation. Transition into Maintenance: The design or compile-time system is handed over to be maintained and perhaps enhanced by some kind of maintenance organisation.

ISO9001	A standard in the ISO 9000 family for quality management systems; which includes: a set of procedures that cover all key processes in the business; monitoring processes to ensure they are effective; keeping adequate records; checking output for defects, with appropriate and corrective action where necessary; regularly reviewing individual processes and the quality system itself for effectiveness; and facilitating continual improvement.

11.2.2 Architecture Change Management

Architecture change management	The organisation and processes that are needed to manage changes to architecture descriptions, mostly stemming from changes to requirements or constraints, or operational systems.
Baseline configuration	A specification or product that has been formally reviewed and agreed upon. The basis for further development. Can be changed only through formal change management. E.g. a contract, a requirements catalogue, architecture documentation, or a hardware configuration.
Configuration Item	An item in a baseline configuration. Could be a requirement, a source code component or a hardware device. Can be at any level of granularity. "Component of an Infrastructure under the control of configuration management. A configuration item can range from an entire system (hardware, software, documentation) to a single hardware component." ITIL
Agile	Willing and able to speedily respond to change.
Change management	The organisation and processes needed to both exercise change control to a baseline, and perform configuration management.
Change Control	The organisation and processes needed within change management to: ▪ Monitor the potential sources of change ▪ Record change requests ▪ Perform impact analysis ▪ Decide which changes should be made.
Request for Change (RFC)	"Form used to record details of a request for a change to any Configuration Item within an Infrastructure or to procedures and items associated with the Infrastructure." ITIL
Impact analysis	Analysis of the effects of a change (perhaps a new requirement or deliverable) to find the effects of that change. How does it impact what has been done so far? How does it constrain what is planned for the future? Leads to an impact analysis report.

Configuration management	The organisation and processes needed within change management to establish a baseline configuration and apply changes to that baseline configuration. Involves work to: ▪ Identify and document the characteristics of each item. ▪ Define dependencies between items. ▪ Control the introduction of new versions of items. ▪ Report the status of configuration items and changes to them.

11.2.3 Architecture Governance

Governance	That facet of management concerned with ensuring an enterprise does what it is supposed to do - that is, achieves goals, follows rules and delivers what its stakeholders expect. It requires measurement and control of performance. May be subdivided into ▪ Corporate governance and principles: the responsibility of the enterprise's executive board. ▪ IT governance and principles: the responsibility of an IT board. ▪ (Enterprise) architecture governance and principles – for strategic design: the responsibility an architecture board.
Architecture governance	The management of an architecture (in development or operation) so as to ensure it conforms to pre-defined architectural requirements, principles, policies and models.
Architecture board	The group of people who maintain architecture principles and governance processes, and appoint governing architects.
Architecture contract	A document that defines those architectural requirements, principles, policies and models that a system should conform to as it is built and when it runs. Also defines any architecture stakeholder rights and interests that must be met.
Governing architect	The architect who has been nominated by the governance organisation to ensure a system is built and/or run in accord with its architecture contract, to manage risk and to ensure the value of the system to its stakeholders. Aka chief architect or design authority.
Architecture compliance review	A process for monitoring the compliance of work done to architecture principles, policies and models. Reviews of various kinds may be carried out at various points in the specification and development of a system. Only some of these reviews require a governing architect or use an architecture review checklists.
Architecture review checklist	A standard checklist of questions to be asked in an architecture compliance review. The questions are general ones, not necessarily mentioned in the architecture contract.
Architecture conformance level	How well or how much of an architecture contract is met by a system, or an architecture description is realised in a system.
Architecture compliance level	How well or how much of a system corresponds to its architecture contract and/or description.

Dispensation	A time-bound waiver from the terms of an architecture contract, granted by a governing architect, and to be reviewed after the specified time.
Capability maturity model	A reference model for evaluating the maturity of an organisation and its processes. First and best known is the maturity model is the CMM for software processes, from the Carnegie Mellon University Software Engineering Institute. There are now maturity models for architecture organisations and processes, such as those included in the list of references.

11.2.4 Architecture in Operations

Architecture in operations	The organisation and processes that are needed to manage the architecture description of an operational system.
COBIT	Control Objectives for Information and related technology, controlled by Information Systems Audit and Control Association (ISACA).
IT service	A service provided an IT operations department. E.g. ▪ management of user roles and identities, ▪ client device configuration, ▪ storage administration, ▪ network provision, monitoring and analysis, ▪ server provision, monitoring and analysis, ▪ business activity monitoring, ▪ virtualisation, ▪ back up & restore, ▪ incident and problem management.
IT services management (ITSM)	The organisation and processes for managing IT infrastructure and the services it provides.
Information technology Infrastructure Library (ITIL)	A large and globally recognised body of advice from the UK government Office of Government and Commerce on how to manage an IT services organisation.
ISO/IEC 20000	An international standard for ITSM (based on the earlier British Standard, BS 15000). It promotes integration of processes to deliver managed services to meet the business and customer requirements. Processes include Planning & Implementing New or Changed Services, Service Delivery Process, Relationship Processes, Control Processes, Resolution Processes and Release Process. It was originally developed to reflect best practice guidance contained within the ITIL framework, although it equally supports other IT Service Management frameworks and approaches (after Wikipedia).
IT configuration management database (CMDB)	"A database of record of configuration item specifications including relationships among configuration items." (ITIL). The authorised configuration of the significant IT components - vital to a configuration management process. Should relate to any enterprise architecture repository.

Asset management system	A record of IT assets. Sometimes focused on end user devices. Should relate to any CMDB.
Common Information Model (CIM)	A standard that defines how the elements in an IT environment can be represented as a common set of objects and relationships in a CMDB. "A common definition of management information for systems, networks, applications and services" (Distributed Management Task Force, Inc.).

12. Enterprise Technology Classification

This section contains a classification of the computing and information technologies that are commonly used to implement the logical components and services described in earlier sections of the reference model. This classification may be used as the structure of a technology catalogue or portfolio.

This section is NOT EXAMINABLE. But training providers and students may find it helpful when learning other sections of the reference model.

Client/user access	Technology components used by end users:
Client device	Desktop PCs, lap tops, mobile devices, client-side operating systems.
Peripheral	A hardware device attached to a computer for the input or output of data. Has its own operating system.
Generic user applications	End user applications that are not specific to a business domain: browsers, portals, office tools, scanners, etc.

Application platform	Intermediate server components and services that support the running of applications, in the tiers of the platform between user interfaces and databases.
Web server	Receives and responds to HTTP request from clients Delegates the request to a suitable server-side program. Responds to client with an HTTP response: usually an HTML page or image for viewing in a Web browser which contains pre-existing (static) content) or generated on-the-fly (dynamic) content. Could be any type of file, Could redirect to another server.
FTP server	Receives and responds to FTP requests from clients.
App server	A server that provides client applications with business services, operations or functions: Makes these business services accessible through various protocols (including HTTP). Provides any kind of data by way of response (including display mark up). Manages its own resources (security, transaction processing, resource pooling messaging) to various qualities of service.
Transaction Manager	A platform application that enables an application to start, commit and rollback transactions. Often provided by a DBMS or an app server.
Distributed Transaction manager	A transaction manager that can commit or rollback a transaction that places data in several distinct data resources, including databases and message queues.
Web services stack	A protocol stack used to define, locate, implement, and make Web services interact with each other. (Service) Transport Protocol: for transporting messages between network applications; includes HTTP, SMTP, FTP. (XML) Messaging Protocol: for encoding messages in a common XML format understood at either end of a network

	connection; includes such protocols as XML-RPC, WS-Addressing, and SOAP. (Service) Description Protocol: describes the public interface to a web service; usually WSDL interface format. (Service) Discovery Protocol: centralizes services into a common registry. UDDI is yet to be widely adopted.
Remote Database Access (RDA)	A standard API used by applications to access a remote DBMS. Defines e.g. how a program should send SQL queries to a DBMS and how record sets will be returned. E.g. ODBC: Open Database Connectivity, hundreds of drivers for many platforms. JDBC: Java Database Connectivity, for Java clients. ADO.NET: Part of the base class library in the Microsoft .NET Framework.

Software development	Tools used to develop applications: languages, Integrated Development Environments, testing tools, etc.
Screen scraper	A tool that extracts data messages passing to and from the displayed output of a legacy application. It enables other applications, including new user interfaces, to enter data into a legacy application, and redirect data originally destined for display on the legacy screen. Most often used to interface to a legacy system that has no defined API for its automated services.

Integration tools	Tools that support point-to-point communication, directory-based distribution and mediator-based distribution, such as file transfer, remote procedure call, and message oriented middleware. It is impossible to draw clean lines between integration tools. Most middleware technologies can be used to implement most component interoperation styles - to support distributed objects, SOA, EDA etc..
Middleware	A platform technology that helps application components to communicate and interoperate across a network. “A confusing mess of messaging, gateways, interfaces, request brokers, queue managers and transaction monitors - often embedded in each other and in other things. “ (Loosely and Douglas).
Point-to-point integration tool	A technology that enables a programmer to write distributed software, in which client/sender and server/receiver modules run on different computers. The theory is to make a remote invocation appear to a programmer as though it is a local invocation of a process within the same name space on the same computer. In practice, remote invocation is more complex. The server module is slower to respond, less certainly available, less reliable and distribution raises security concerns. On the other hand, the server is more scalable, since it can be scaled out to parallel servers. Technologies that can be used to enable point-to-point communication (as well as other kinds of communication) include RPC and Web Services.

Remote Procedure Call (RPC)	A technology that enables a client to find an automated service on a remote machine, and invoke a named procedure in this different name space. The client sends a fixed set of parameters and waits for the answer to be returned in message using the same interface. (E.g. XML-RPC uses XML to encode its calls and HTTP as a transport mechanism.) RPC can be used to implement both DO and SOA interoperation styles. In DO, since it invokes a procedure as an object-method pair, it is often called Remote Method Invocation. (And RMI is the name of Java's Java Remote Method Invocation API.)
Web services	A technology that enables a client to find an automated service and invoke it using the SOAP protocol to exchange XML-based messages. Usually over a network. Usually via HTTP/HTTPS - but also SMTP. Web services can be stateless or stateful, and invoked synchronously of asynchronously. So though they are commonly associated with SOA, they can also be used to implement a Distributed Object style of interoperation.
Introduction agent (direct broker) integration tool	Technologies that can be used to implement the introduction agent kind of distribution. They add overheads on top of RPC.
Object Request Broker (ORB)	A technology that enables the objects in an object-oriented program to be distributed between computers. A technology programmers need to get one distributed object to call an operation on another. E.g. Microsoft ORBs include DCOM and .Net remoting.
Common Object Request Broker Architecture (CORBA)	Standards for an object request broker to comply with, defined by the OMG. It uses CORBA IDL and IIOP.
Internet Inter-ORB Protocol (IIOP)	A standard for technologies used to send objects (rather than messages) over TCP/IP.
Web service broker	A technology that uses Universal Description Discovery and Integration (UDDI) as the API for registration and location of web services. It offers a SOAP-based web service for finding and registering web services.
Message broker (indirect broker) integration tool	A Message-Oriented Middleware (MOM) technology that sits between communicating components; it stores and forwards messages according to the required styles of interoperation and communication. MOM technologies may be used to implement any communication style. They may use RPC or P2P technology under the covers.
Message Queuing technology (MQ)	A message broker that stores and forwards messages. It smoothes demand for server operations, since the server works at its own pace, but can reduce client response time.
Publish and subscribe integration tool	Publish and subscribe distribution is possible via any kind of component communication technology, but is most usually associated with a mediator or message broker.

Data management tools	Tools for the storage and retrieval of structured data.
Database management system (DBMS)	A platform application that manages the storage and retrieval of data in/from a data store. A broker that enables applications to create, read, update and delete data in a database. The main types are: Hierarchical Network (aka CODASYL) Relational (supports SQL) NoSQL (used in document and web applications)
Distributed database	A database that is physically located in two or more locations, under the control of one DBMS or a distributed transaction manager.
Data warehouse	A database that is designed to support analysis of stored data and management information reports.

Unstructured data management tools	Tools for content, document and knowledge management
Content management tools	Tools for producing, storing, editing, sharing and searching any unstructured data.
Document management	Content management tools for producing, storing, editing, sharing and searching electronic documents and document images. Often associated with workflow systems.
Knowledge management tools	Typically some kind of collaborative application.

Server devices	Server hardware, server OS, clustering, etc.

Data storage	On-line data store, DAS, NAS and SAN, Connect and switch , etc.

Networks	Technologies that enable applications to communicate over a network: LAN devices, WAN devices, load balancers, convergent technologies.
Router	A computer that ties two or more networks together, with routing and forwarding capabilities.
Hub	A computer that connects devices (computers, printers, servers) on the same network within a building or campus, and enables connected devices to talk to each other.
Switch	A hub with additional routing and forwarding capabilities.

IT Services Management/Operations	Tools used to support service level management, performance monitoring, event and fault monitoring , etc.

Environment	Technologies used in data centres: server racks, control systems, power systems, etc.