

Fundamentals Of EA - Viewpoints, Metamodel, Change/Development

Enterprise Architecture

Small and relatively simple organisations can often afford to ignore the architecture and manage in a gut feel, holistic fashion.

As organisations become larger and more complex the interactions, dependencies and connections between their internal and external resources create many different effects and need to be managed at a number of levels and from a number of different perspectives.

Enterprise architecture provides the basis for defining and then managing an organisation's resources, their interactions and their outcomes. An enterprise architecture is a model of an organisation's business assets that reflects:

- The current state of those assets and their interaction
- The target state of those assets and their interaction
- The transitional states as transformation and change programmes and projects are implemented

Having effective documentation of these perspectives enables an organisation to achieve enterprise alignment and integration in order to manage change, optimise cost and value, improve quality and satisfaction, and reduce "time-to-market".

Enterprise architecture is the highest-level, most strategic type of architecture and its directives, such as principles, policies and business rules, may apply everywhere within the enterprise, including within solution architecture and the resulting solutions.

Solution Architecture

A solution addresses a problem, risk or opportunity facing a business.

Over time many solutions are produced, even if they are not named as such, and once established they become indistinguishable from each other and part of the fabric of the enterprise.

Solutions may address newly arising situations or seek improvements to the business by modifying existing provision in a specific area, in other words replacing an existing solution.

Solution architecture, is a structured way to address a problem, risk or opportunity and produce an effective solution. It is called solution architecture because of the comprehensive and systematic approach that ensures the solution design truly addresses the problem and will work within the larger structure of the business.

Solutions are not the same as IT systems. Plenty of problems can be solved, risks avoided, and opportunities seized by changes in staff behaviour or making better use of existing resources, for example. Many solutions do involve new or modified IT systems, but these are rarely able to solve the problem on their own.

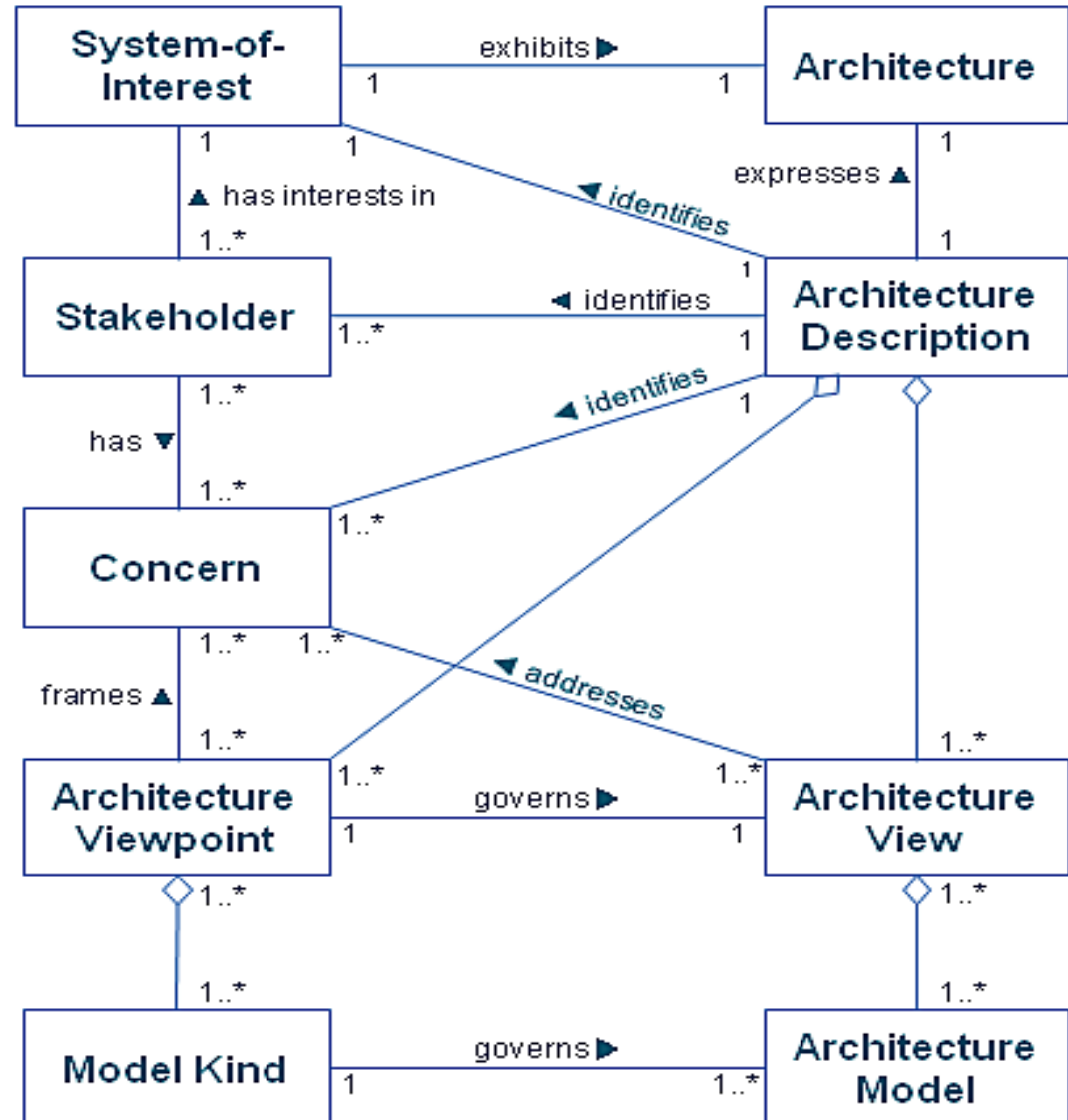
Solution architecture treats a solution as a system with component parts that interact to produce the required behaviour. The components may be new or existing and may be shared with other solutions. The interactions occur via interfaces between the components where information is exchanged.

Elements Of An Architecture Description

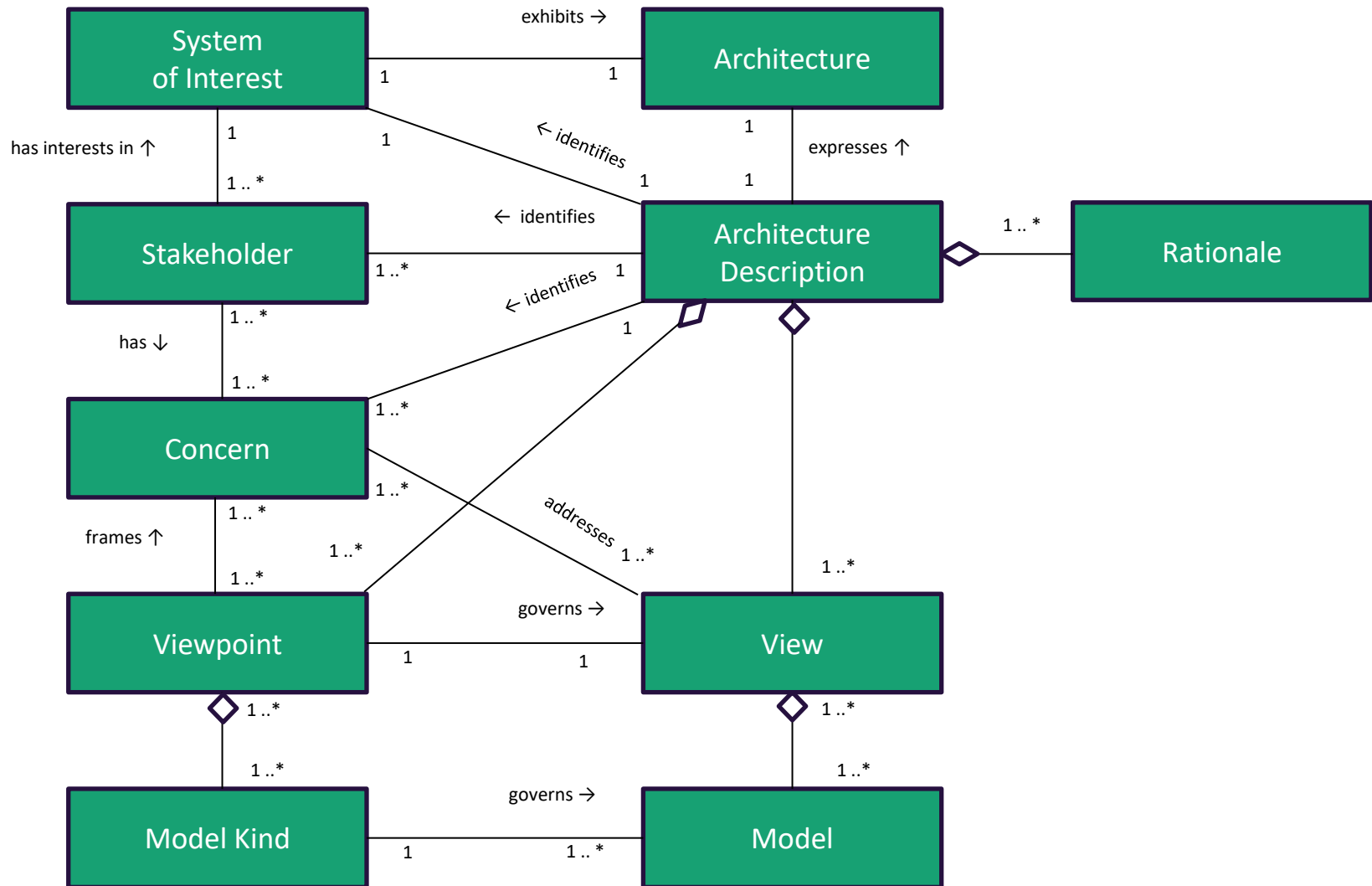
A **system** is a collection of building blocks / components organized to accomplish a specific capability or set of functions.

A **concern** is a key interest that is crucially important to stakeholders, and may determine the acceptability of the system

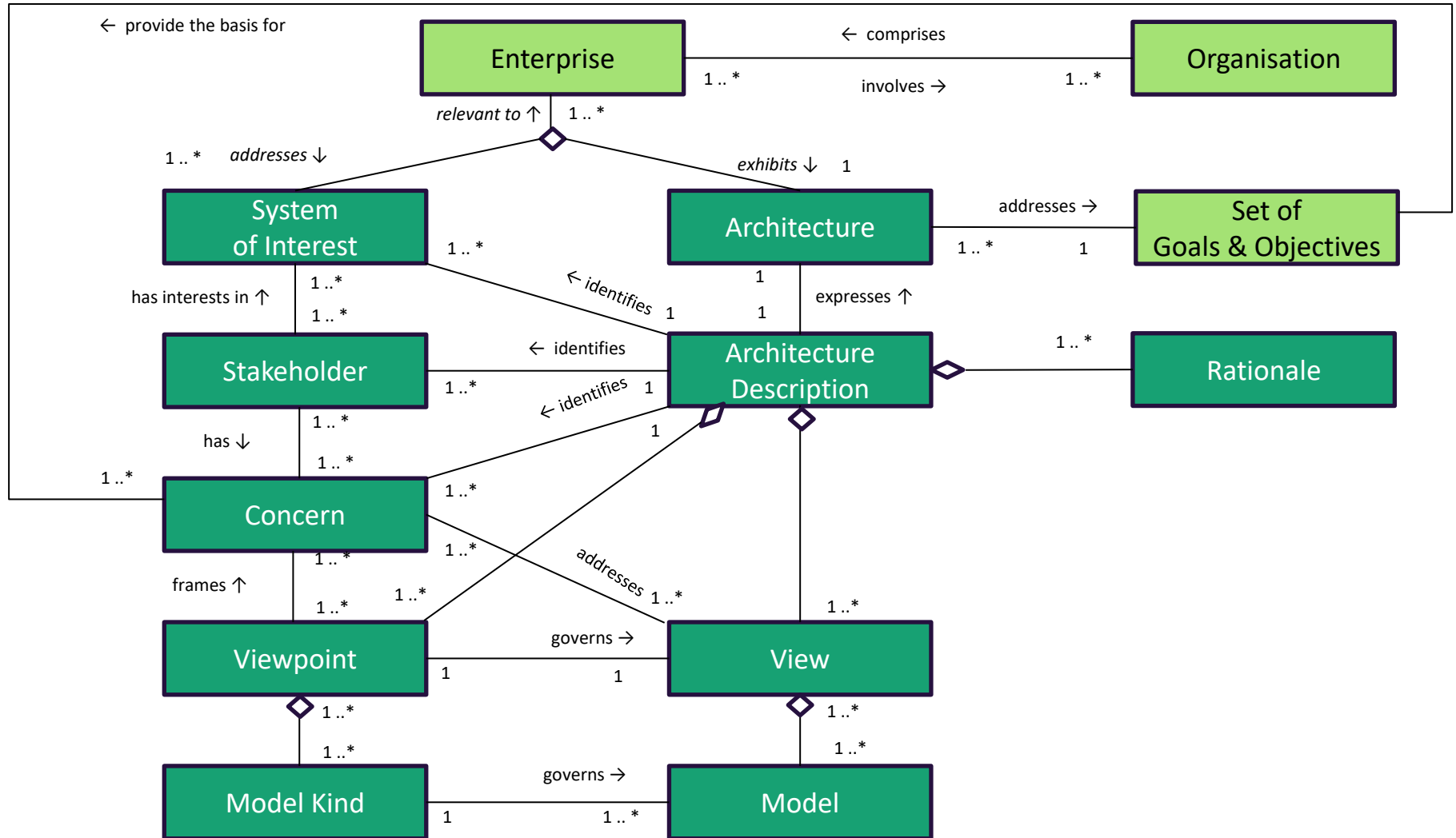
Adapted from the
ISO /IEC/IEE 42010:2011
ISO /IEC/IEE 15288:2015
Standards



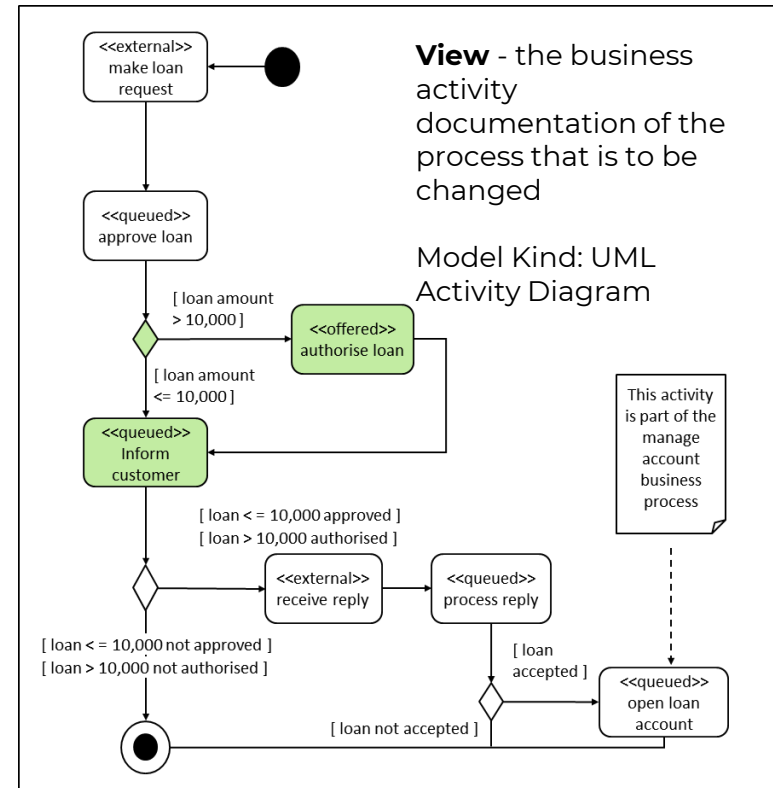
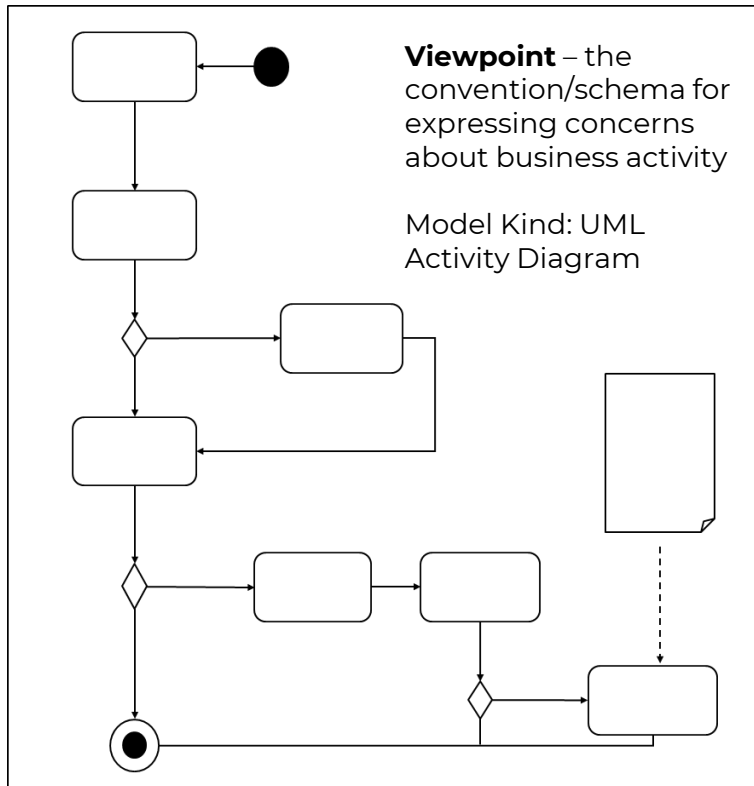
ISO /IEC/IEE 42010:2011 - Single System - Viewpoints & Views



Extending the ISO 42010 Standard To Include The Enterprise



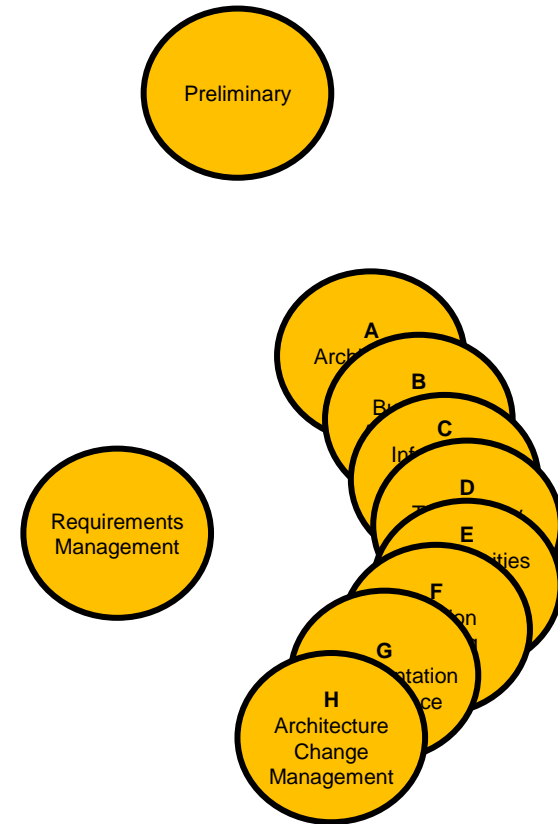
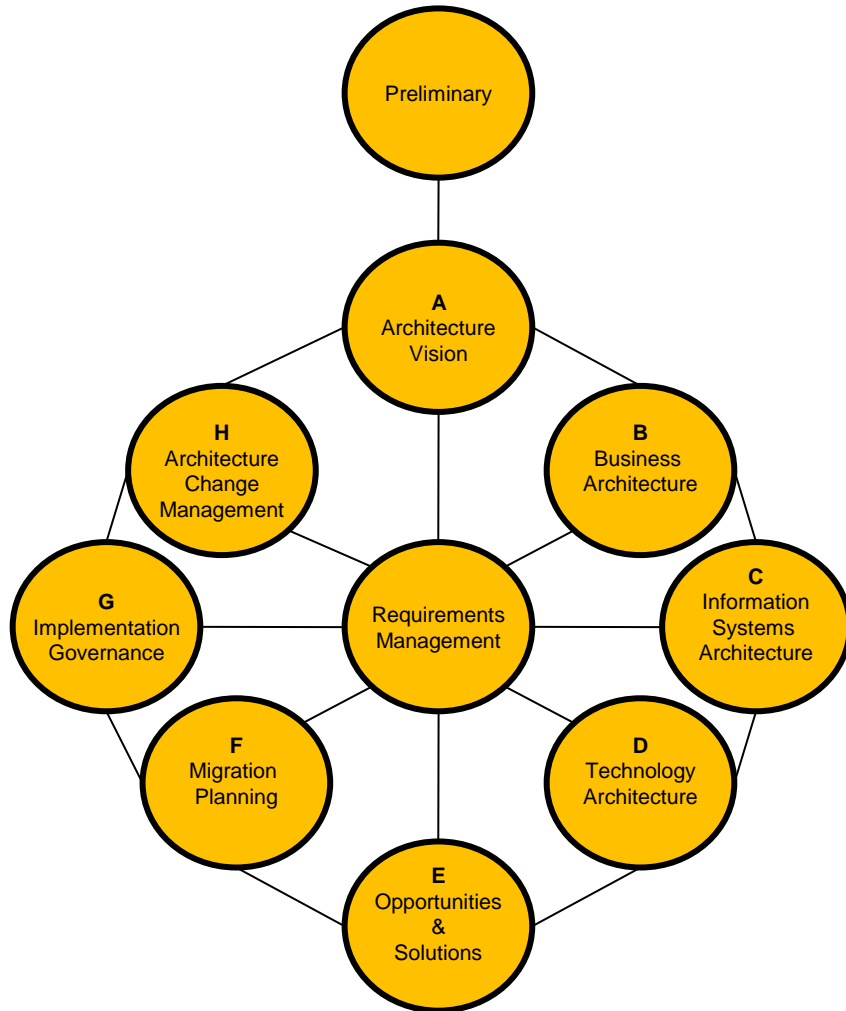
Viewpoint & View Example



L1 / L2



Predecessor & Successor Phases

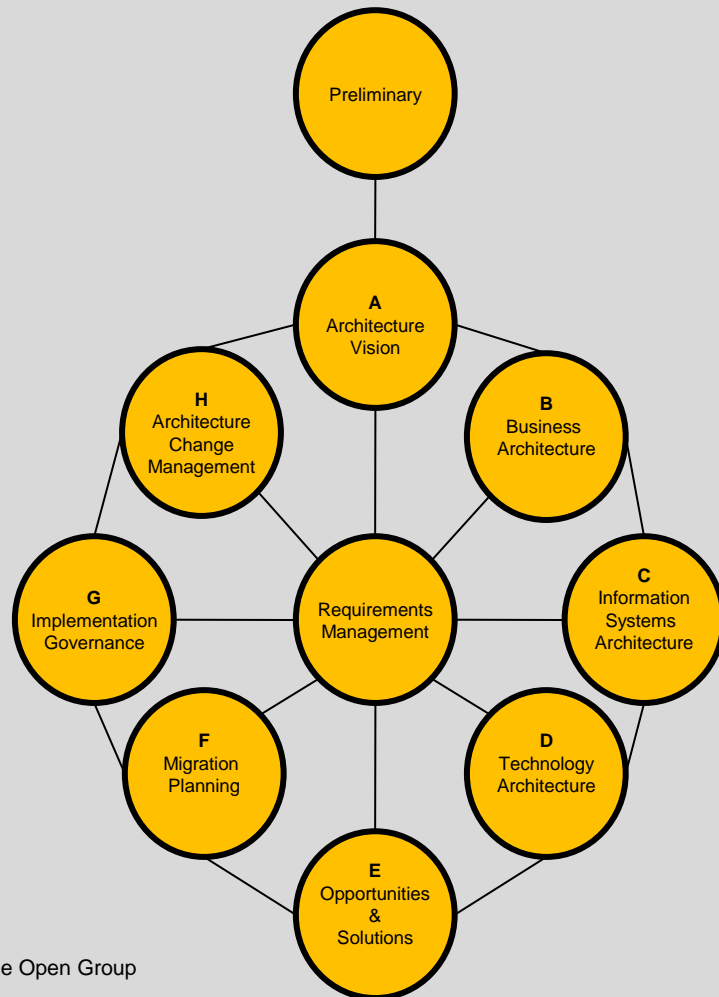


Predecessor and Successor Phases

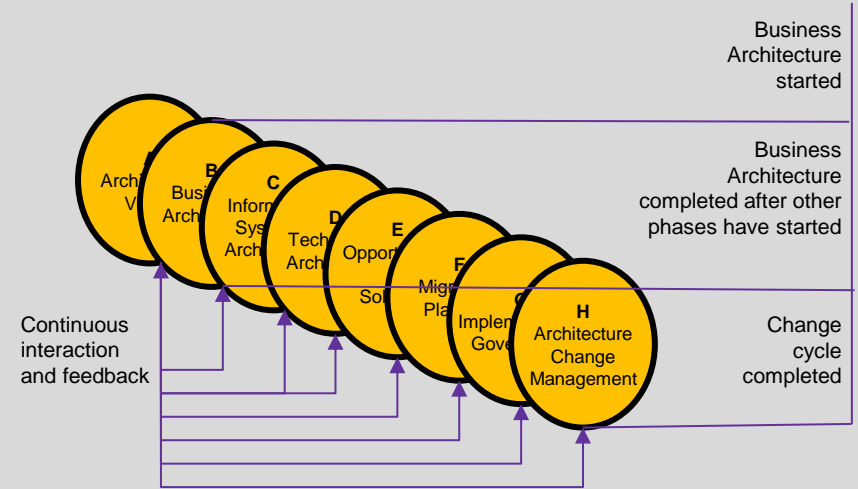
The previous phase must finish before the successor phase completes, but the successor phase can start once there is some requirement or other business reason for it to do so.

Predecessor & Successor Phases

Logical Sequence



Actual Sequence

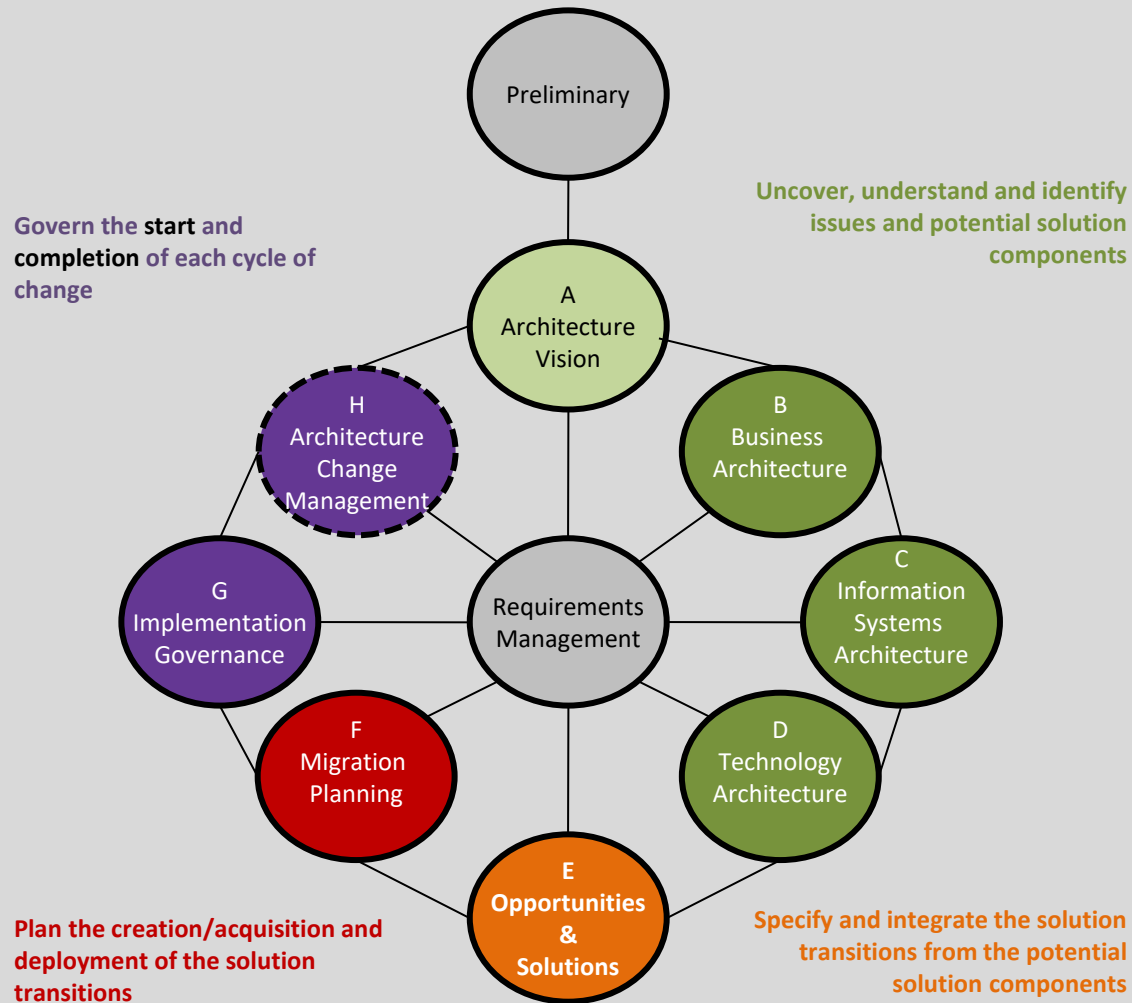


Predecessor and Successor Phases

The amount of work done in each phase is bounded by the scope agreed by the stakeholders for each complete cycle of change during the architecture vision phase (equivalent to defining a backlog of sprints for a product when using agile methods).

The previous phase must finish before the successor phase completes, otherwise something is being missed. However, each successor phase can start once there is some requirement or other business reason for it to do so.

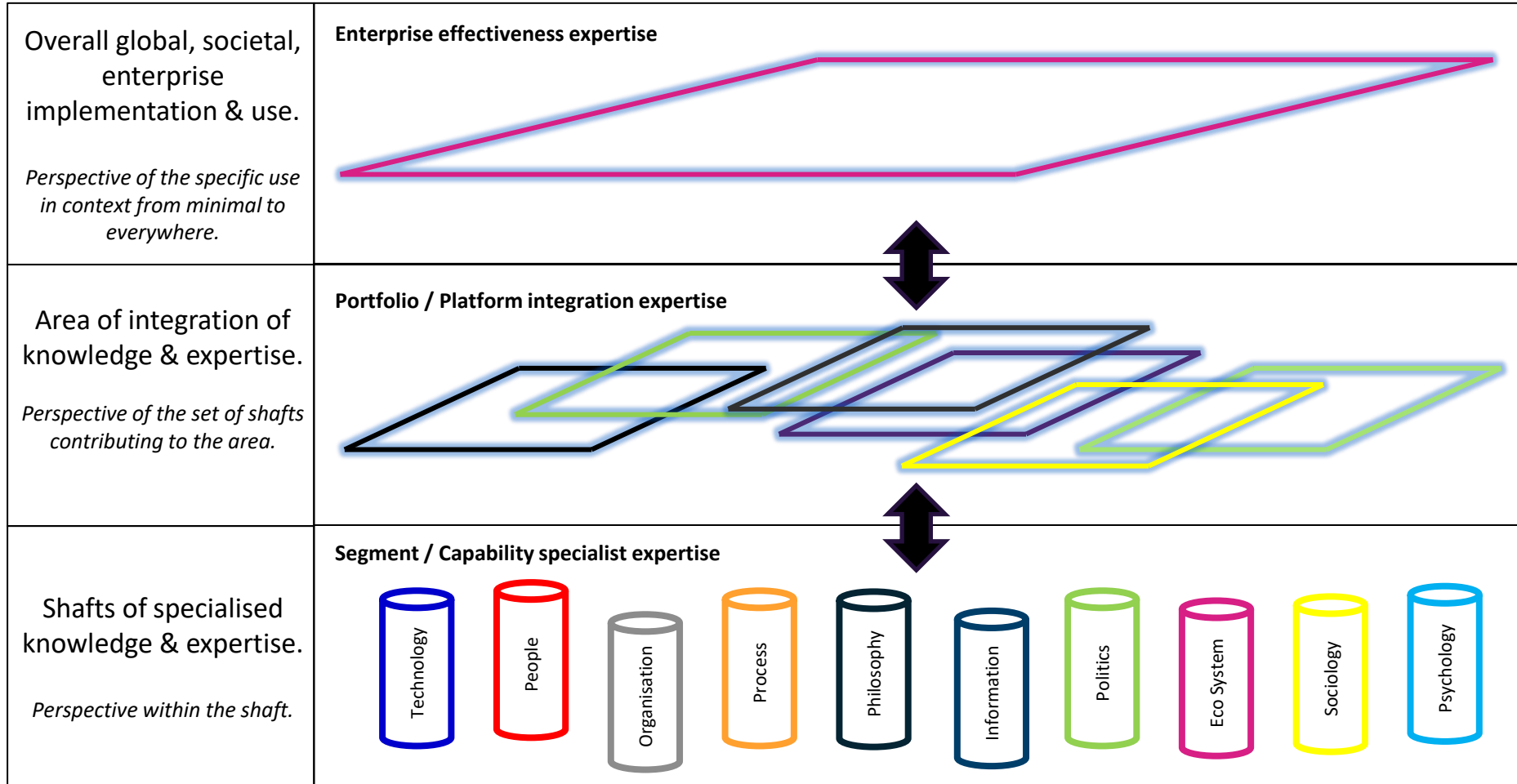
The Basic Steps In The ADM



Experts and Expertise:

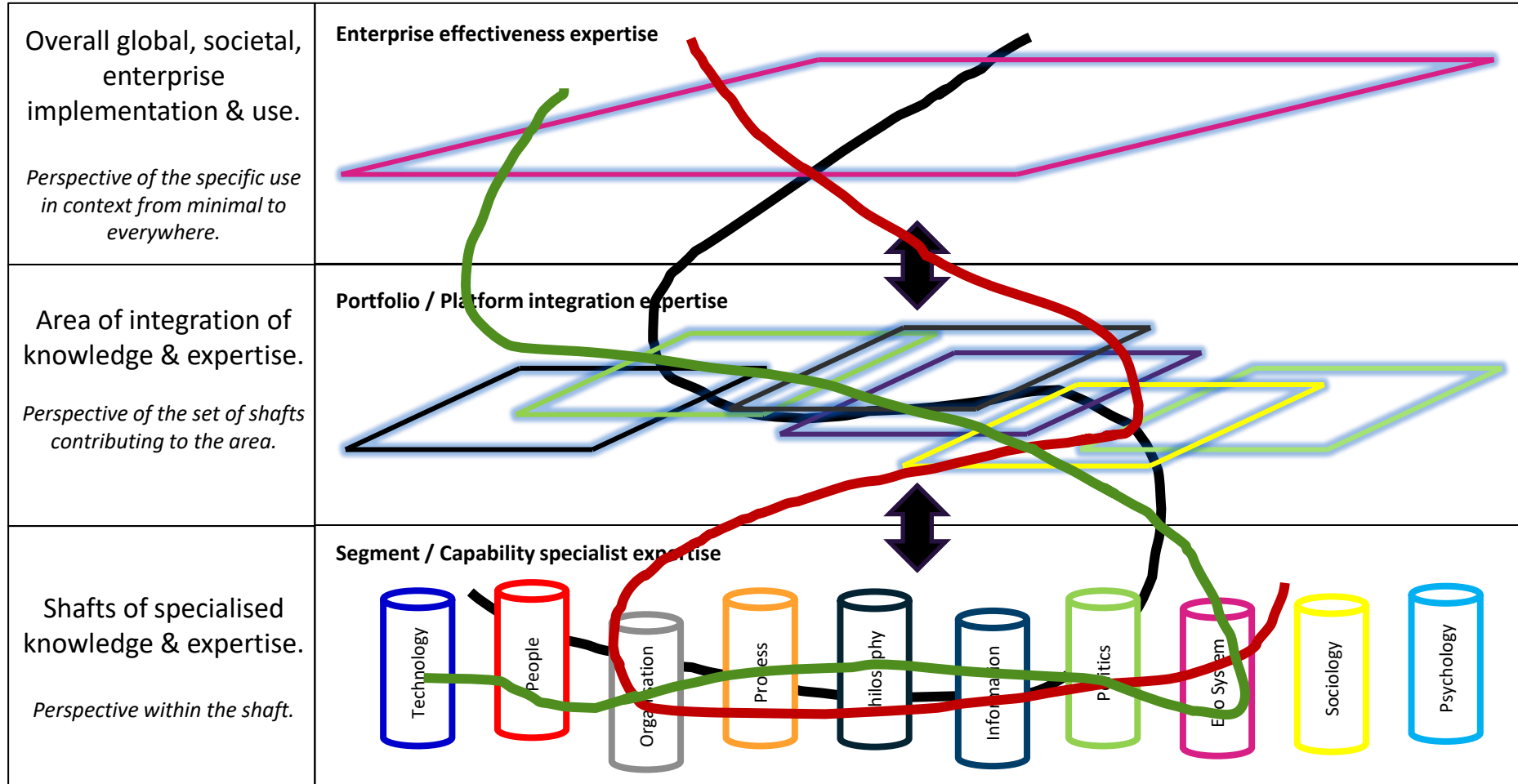
A situation of continual conflict and necessary resolution

Expertise occurs at all levels and is often in conflict. A solution involves some set of often (unique) expertise.



Solutions Use Different Sets Of Capabilities Across Many Platforms To Deliver Enterprise Effectiveness

Expertise occurs at all levels and is often in conflict. A solution involves some set of often (unique) expertise.



Architecture Levels & Roles

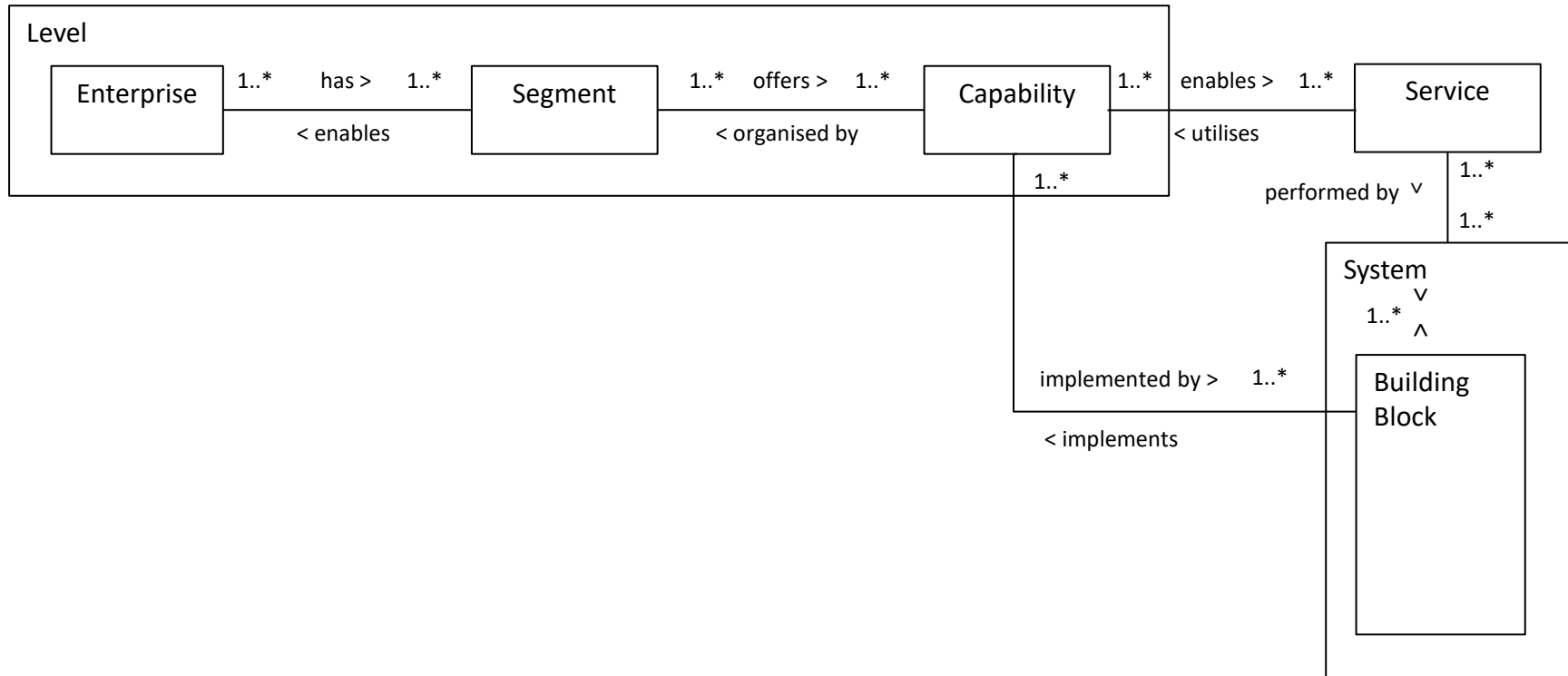
Segments and Domains

Enterprise architects evolve and govern aspects across and between segments to achieve goals and objectives

Segment architects evolve and govern capabilities within segments to achieve goals and objectives

Capability (specialist) designers and builders create and change capability in building blocks to satisfy requirements

Solution architects organise capabilities into new or changed solutions delivering services to meet requirements



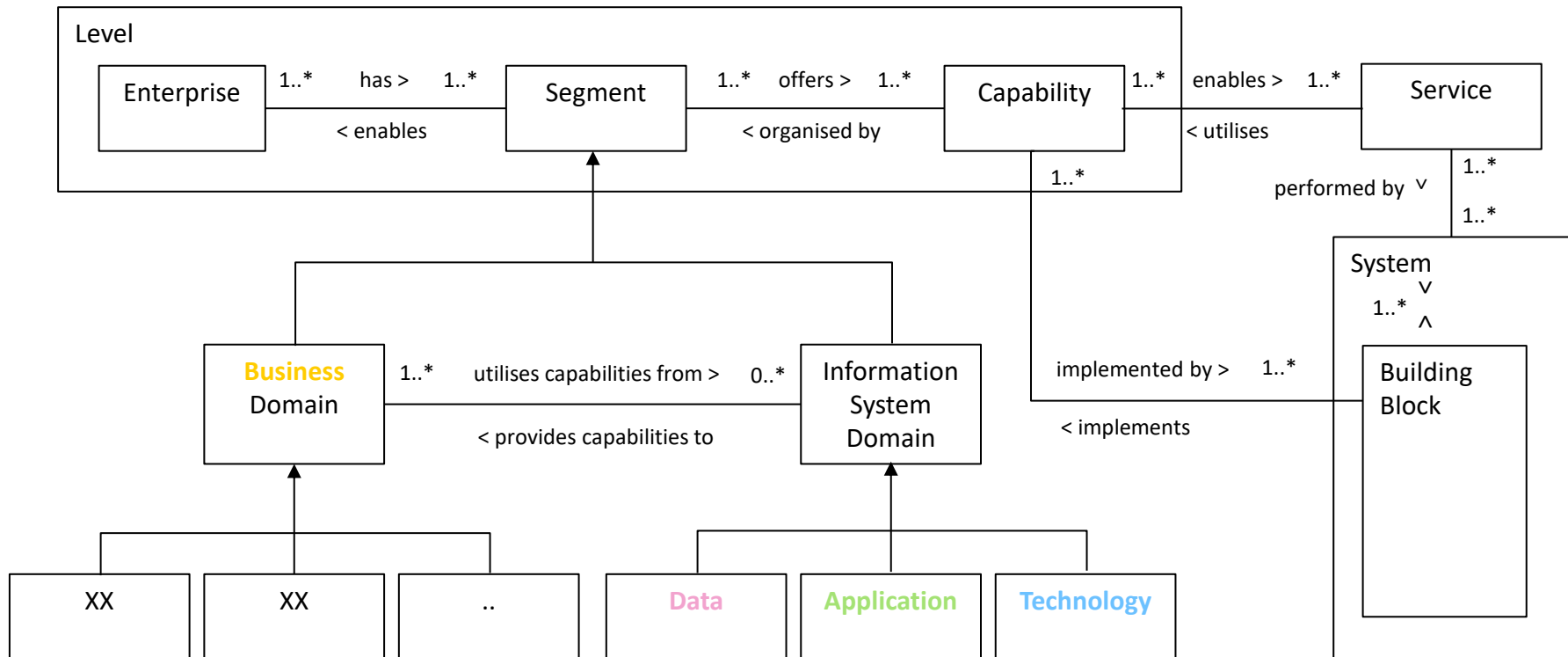
Segments and Domains

Enterprise architects evolve and govern aspects across and between segments to achieve goals and objectives

Segment architects evolve and govern capabilities within segments to achieve goals and objectives

Capability (specialist) designers and builders create and change capability in building blocks to satisfy requirements

Solution architects organise capabilities into new or changed solutions delivering services to meet requirements



TOGAF identifies one overall business domain that may be broken down further e.g. business function / process portfolios depending on what a business actually does e.g. Finance, Manufacturing, Legal, HR, Customer Management, Service & Support

TOGAF identifies three starting information system domains which may be broken down further e.g. (Technology) into Software, Servers, User Devices, Execution Environments, Networks, Buildings & Mechanical Devices



The Enterprise Architect

- has the responsibility for architectural design and documentation at a landscape and technical reference model level. The Enterprise Architect often leads a group of the Segment Architects and/or Solution Architects related to a given program. The focus of the Enterprise Architect is on enterprise-level business functions required.

The Segment Architect

- has the responsibility for architectural design and documentation of specific business problems or organizations. A Segment Architect re-uses the output from all other architects, joining detailed technical solutions to the overall architectural landscape. The focus of the Segment Architect is on enterprise-level business solutions in a given domain, such as finance, human resources, sales, etc.

The Solution Architect

- has the responsibility for architectural design and documentation at a system or subsystem level, such as management or security for a specific solution.

TOGAF Comments On Architect Roles

“An architect is involved in the entire process; beginning with working with the customer to understand real needs, as opposed to wants, and then throughout the process to translate those needs into capabilities verified to meet the needs.

Additionally, the architect may present different models to the customer that communicate how those needs may be met and is therefore an essential participant in the consultative selling process.

However, the architect is not the builder and must remain at a level of abstraction necessary to ensure that they do not get in the way of practical implementation.

The following excerpt from The Art of Systems Architecting (Rechtin and Maier, 2000) depicts this notion: "It is the responsibility of the architect to know and concentrate on the critical few details and interfaces that really matter, and not to become overloaded with the rest."

The architect's focus is on understanding what it takes to satisfy the client, where qualitative worth is used more than quantitative measures. The architect uses more inductive skills than the deductive skills of the builder. The architect deals more with guidelines, rather than rules that builders use as a necessity."

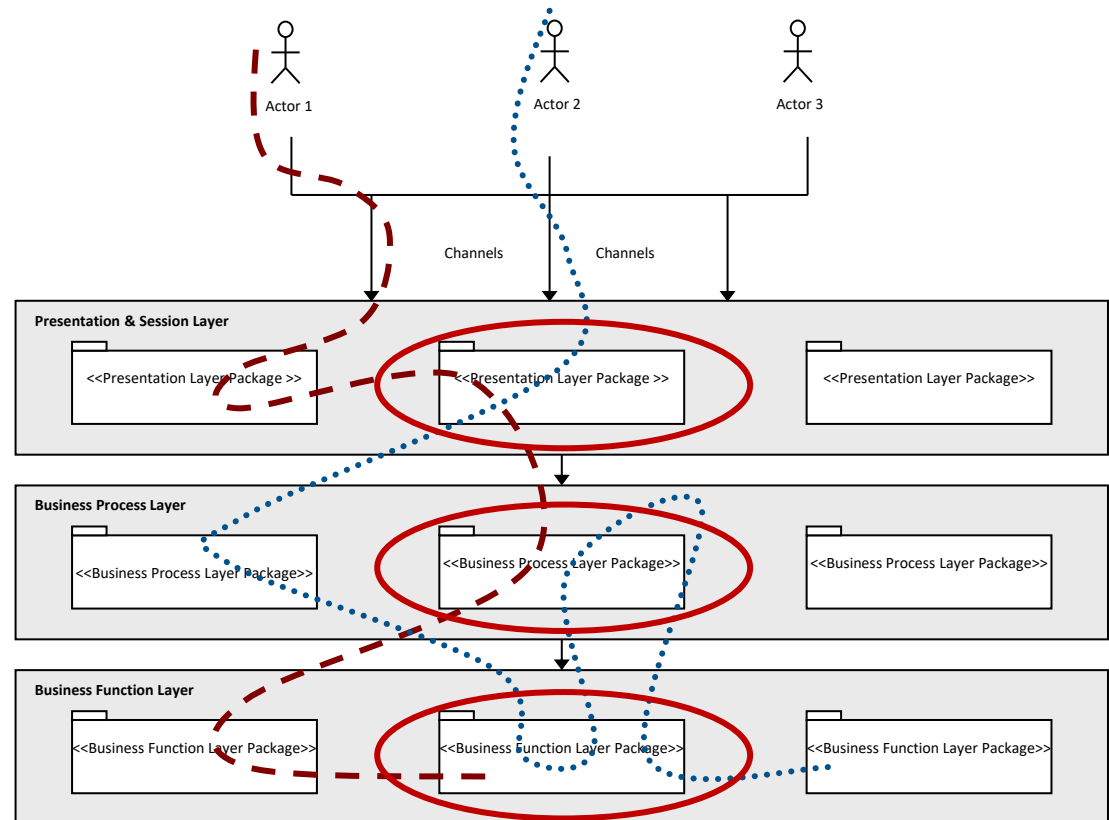
The Foundational Problem Of The Architect's Role

Global vs Local Optimisation

Any solution (at the strategic, segment or capability level) will have to balance its specific impact for a particular actor and on a specific module with the wider impact on all of the other actors and modules within the enterprise.

It is almost never the case that the optimum outcome for a one local viewpoint is the same for other local viewpoints or the global viewpoint.

One of an architect's unique responsibilities is to bring out this conflict, explain the issues, and work with others in the business the shape the delivered solution to do the best combination of greatest benefits and least harms.



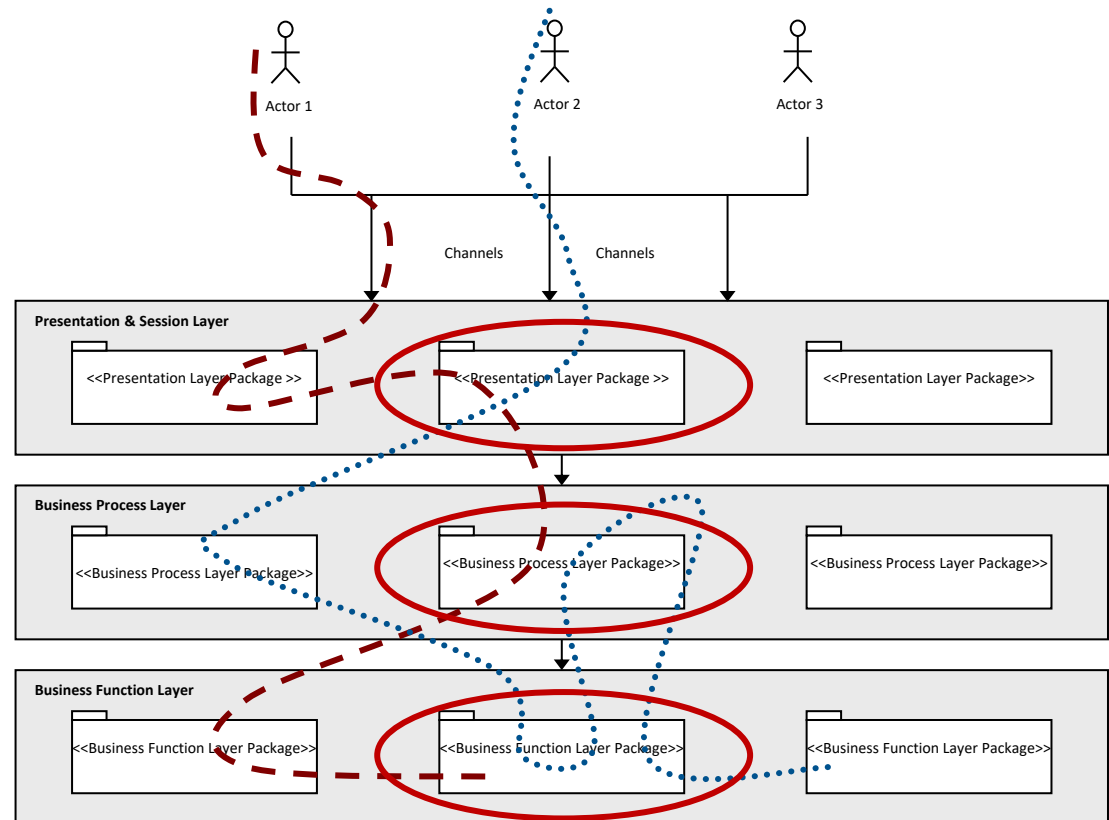
Example of Global vs Local Clashes

Sales in a regulated environment such as the drug industry.

Actor 1 is in sales and wishes to change a module, removing what they see as redundant steps and checks to speed up the sales process and increase yearly revenue.

Actor 2 is in compliance and wishes to add new steps and checks to ensure compliance to new traceability regulations, required to operate in the market without penalties.

Each individual set of requirements requires different changes to business process, applications and data. The enterprise, solution and specialist architects' role is to recognise the conflict and help to broker a solution that achieves the right balance.



Architecture / Architects vs Design / Designers

The difference between the practice and roles of architecture/architects and design/designers is not always clear. In reality there are many overlaps, and many people perform both or some of both roles

Architecture without design (specific implementation and practice) is slide ware, while design without architecture (context and scope) delivers cul-de-sacs and closed off local optimisations.

Architects

Architects focus more on the abstract and logical; strategy, purpose and structure; and the integration of components (ABBs)

Designers

Designers focus more on the concrete and physical; implementation and practice; and the internals of components (SBBs)

So, who is Enterprise Architecture and TOGAF for:

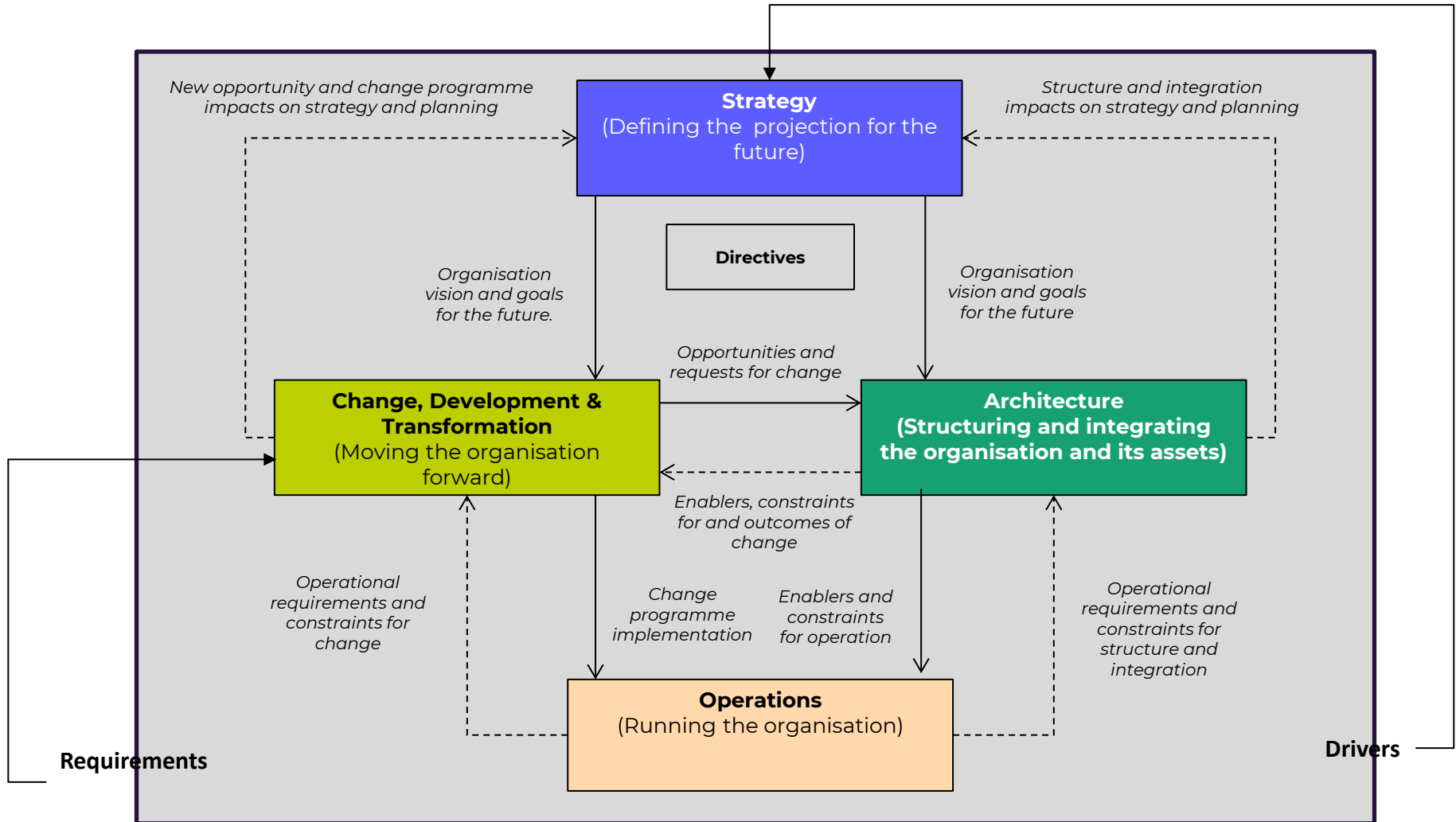
Enterprise, Solution and Segment/Specialist Architects at all levels who construct and govern architecture building blocks (ABBs) to enable the creation of effective solution building blocks.

Enterprise, Solution and Specialist Designers at all levels who need to design solution building blocks (SBBs) and must work within defined architectures.

Programme and Project Managers who lead projects that develop architecture and solution building blocks.

Positioning Architecture - Four Main Views Of An Organisation

Organisations are driven from four main views, that reflect the goals, change, structure, actions of and outcomes for that organisation.

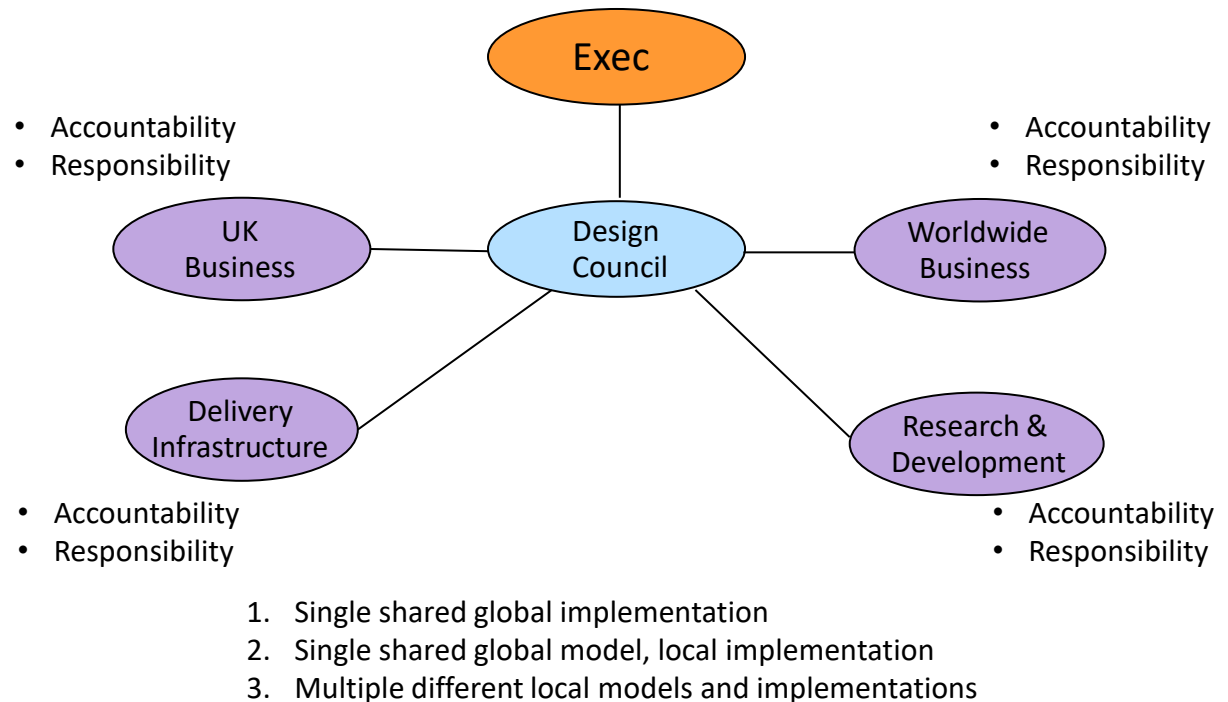


The Operating Model and Interoperability Requirements

Key to establishing interoperability is the determination of the corporate operating model, where the operating model is “the necessary level of integration and standardisation services to customers.

An operating model describes how a company wants to thrive and grow.

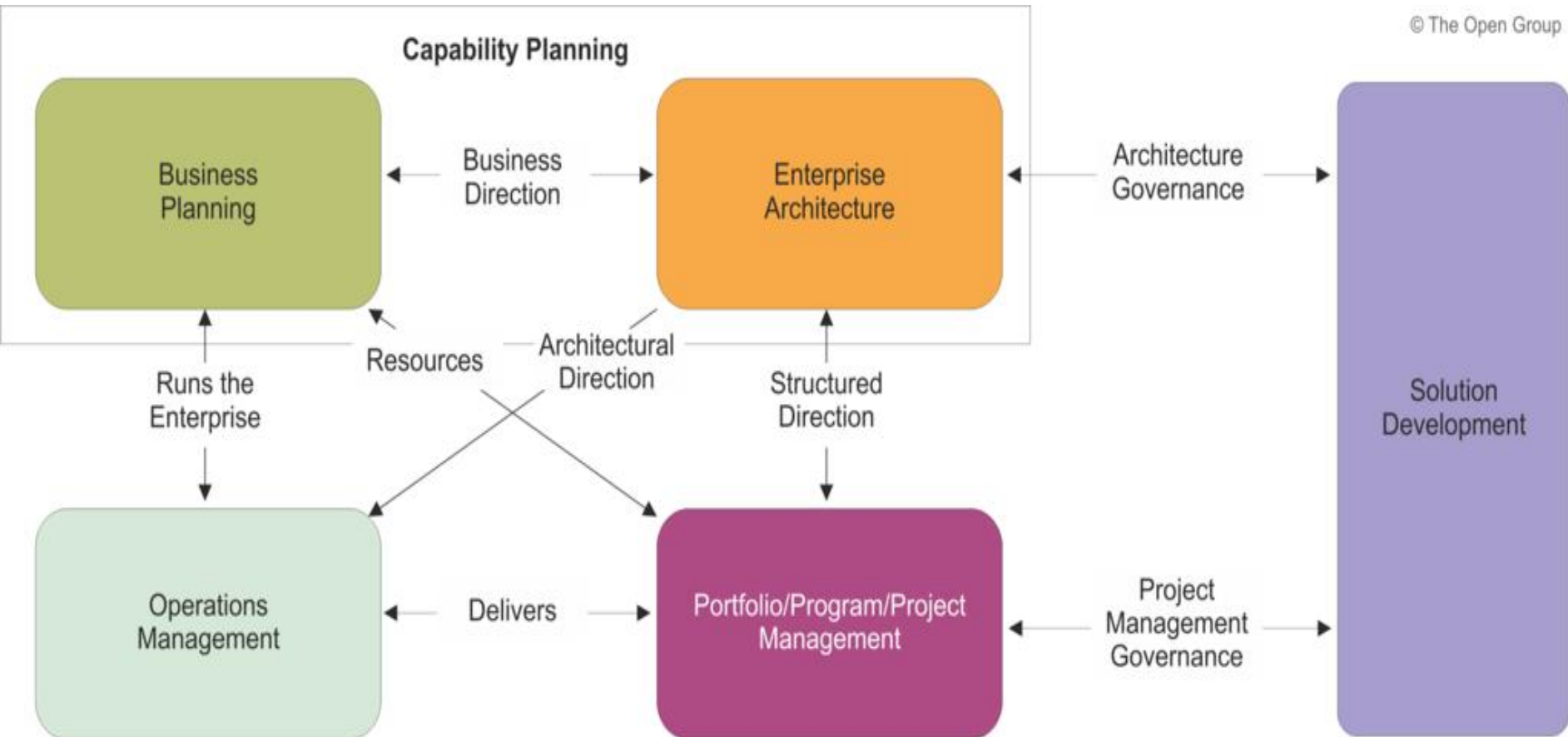
The corporate operating model will normally indicate what type of interoperability approach will be appropriate. This model should be is usually provided by the enterprise's strategy team.



Example interoperability/integration patterns

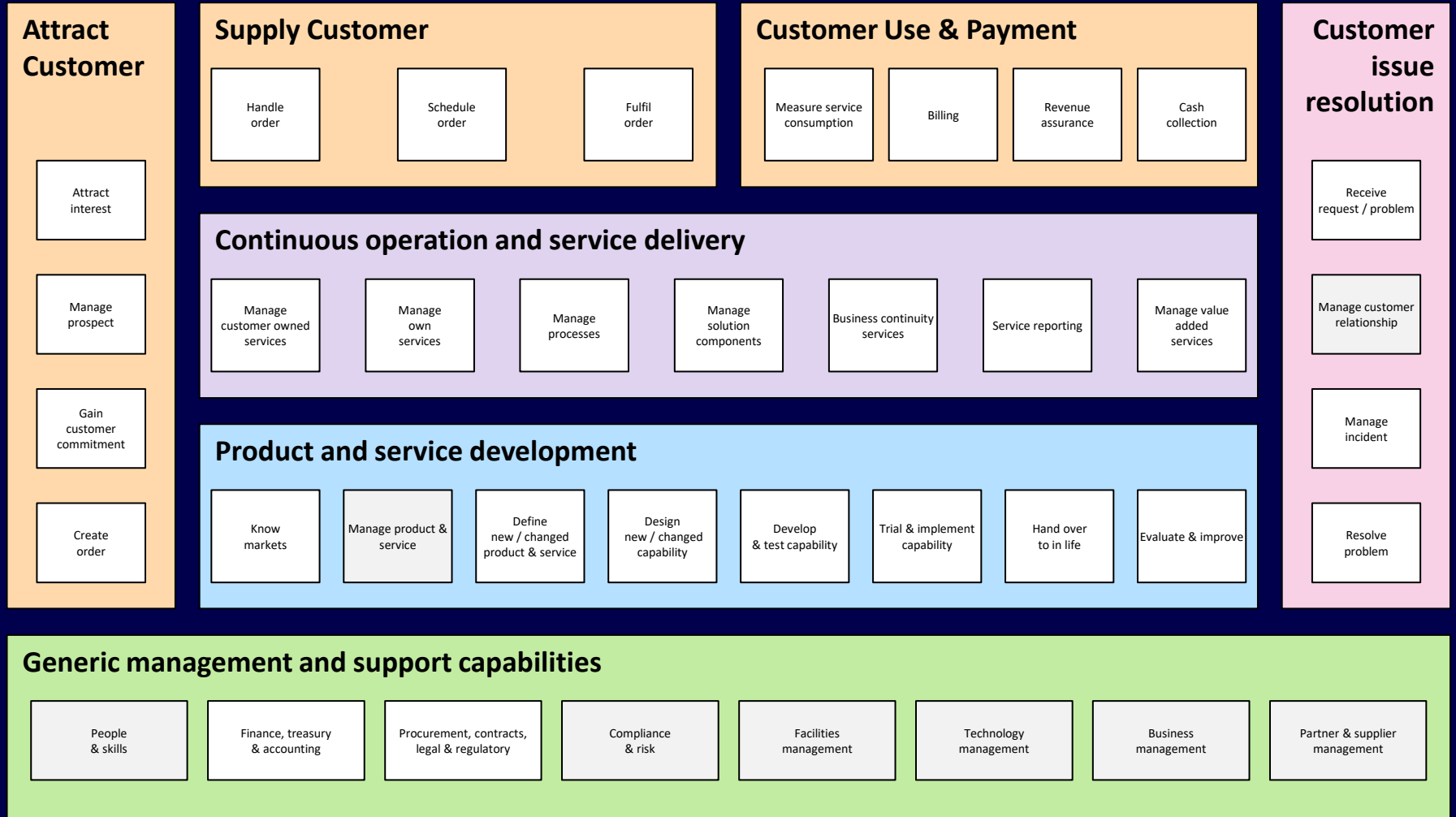
Capability Planning Context (TOGAF Approach)

Interaction between planning and control framework that reflect the goals, change, structure, actions of and outcomes for an enterprise.



Value Streams & Business Capabilities

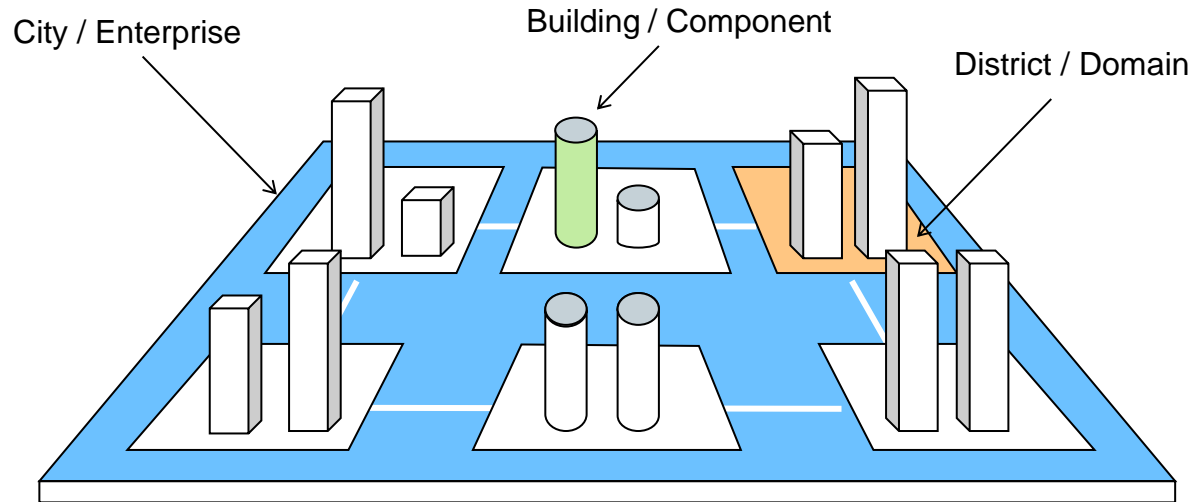
Example Business Capability (38) Map - Structured By Value Streams (7)



Architecture Contexts

City planning is an often-used analogy for enterprise architecture. In simple terms a city can be considered to have at least three organising contexts.

- **City (Enterprise)**
The structures and integration needed across the whole city (sewers, roads etc.).
- **District (Domain/Segment)**
The structures and integration needed to manage the district and its buildings and integrate back into the city-wide structure.
- **Building (Component/Capability)**
The structures needed to deliver the specific properties of the building and integrate back into the district structure.

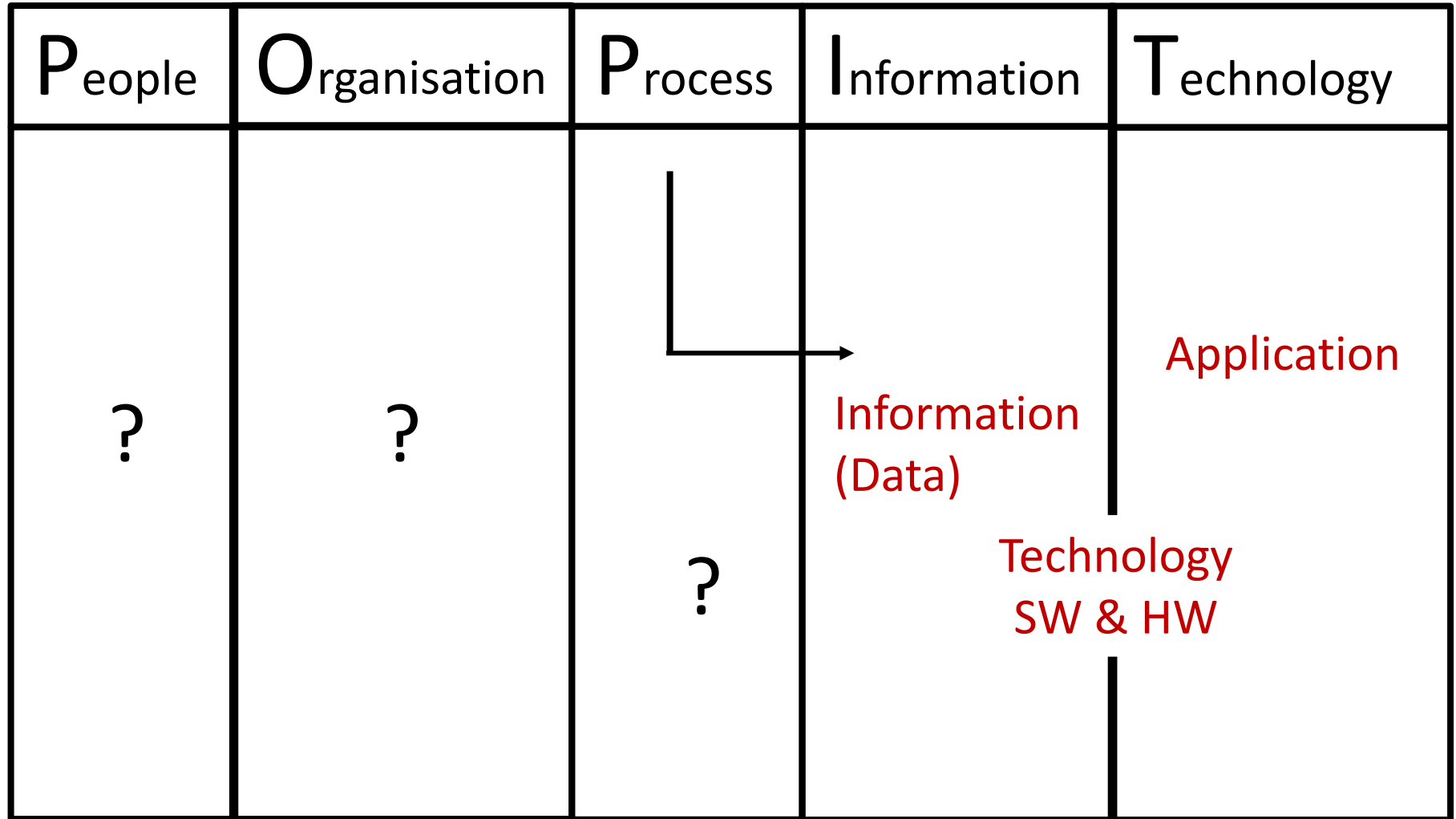


The City Planning Analogy

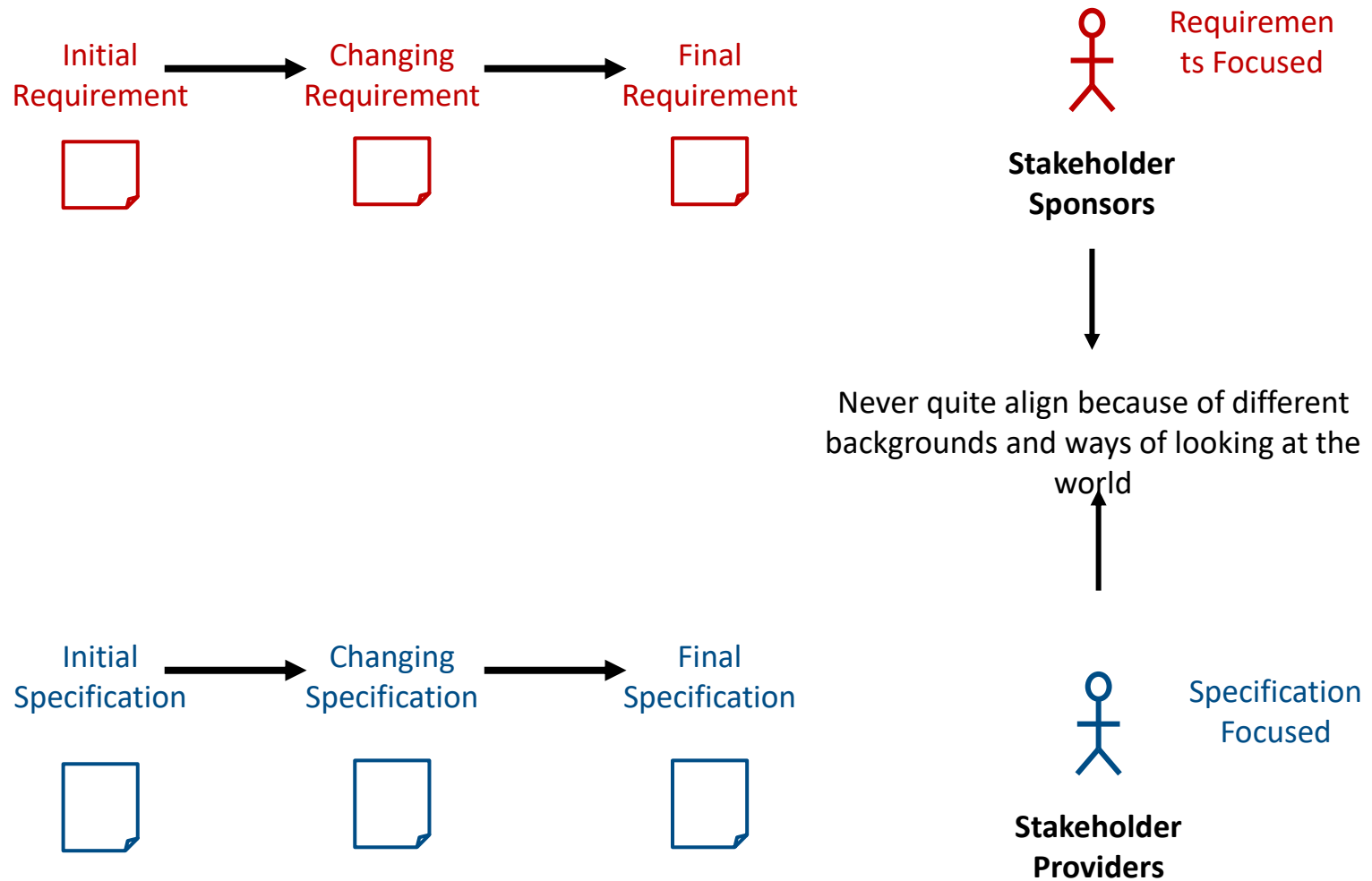
Different capabilities are planned and managed at the different levels .

- Roads, sewers, water pipe, electricity grids tend to be planned across the city
- Types of activity, (business, leisure, housing) are often “zoned” into specific districts.
- Individual buildings are then designed within these contexts and to meet their specific requirements.

City planning sets the context and the basis for integration for district and then building planning.



Requirements & Specification Viewpoints



The Nature Of Change

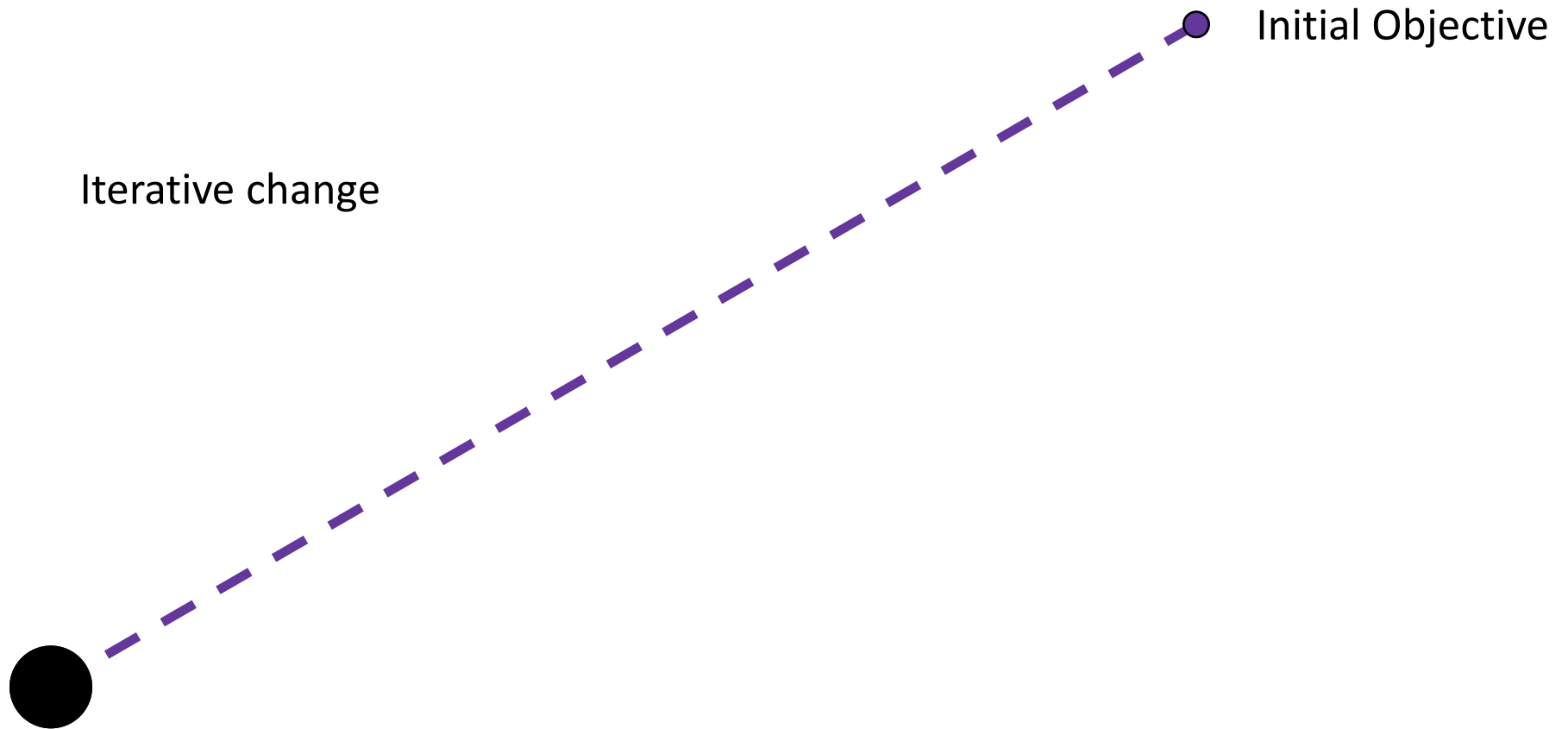
- All architecture is intentional.
- The question is what is the nature of the intention?
- Specific truisms:
 - Doing nothing is a plan.
 - All change is planned (**even if the plan is empty and pure discovery**)
 - All change generates discovery
 - All change generates learning
 - All change is founded on previous change (or existing elements)

All Change Is A Direction Of Travel

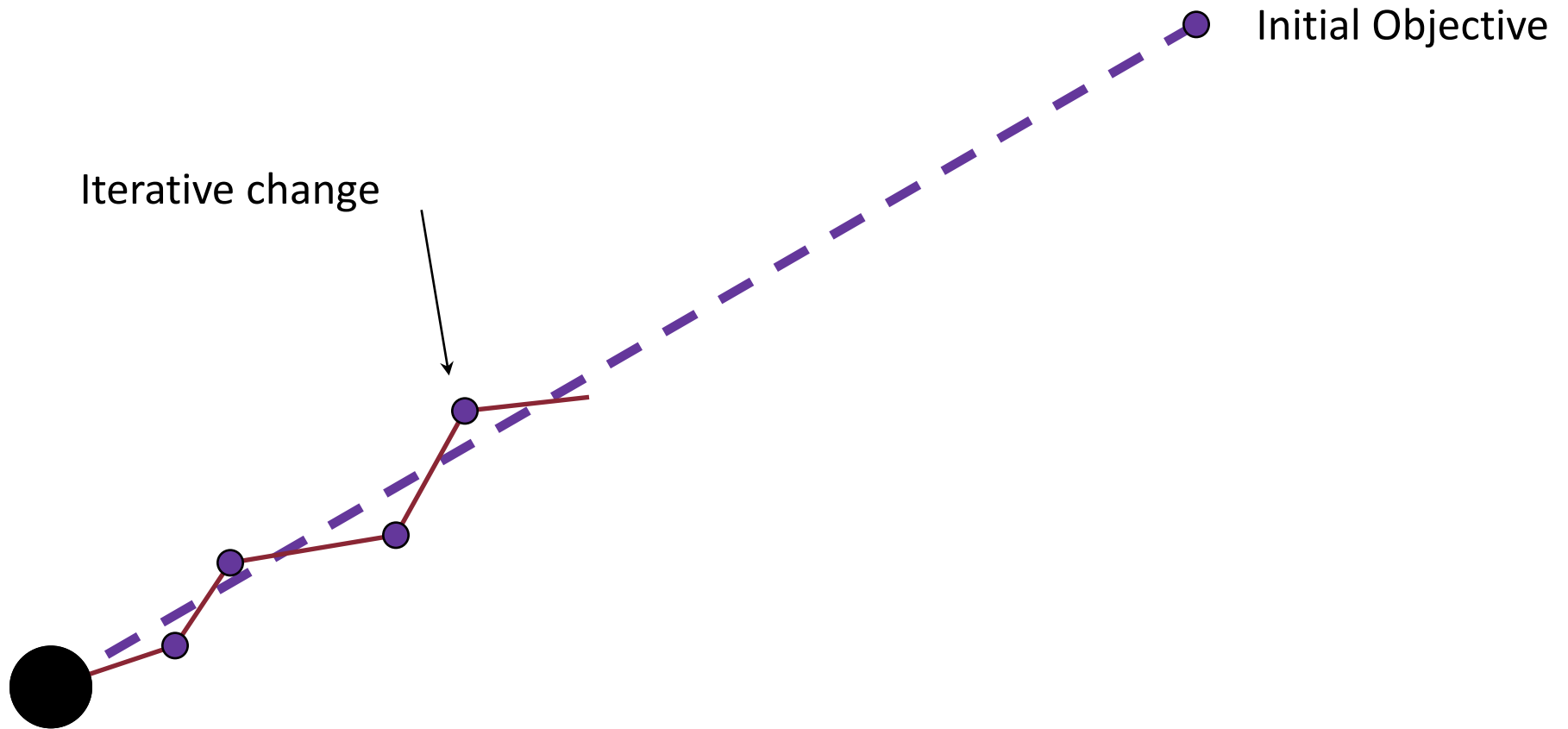
Intentional architecture is essentially planning. A plan is an expectation for the future and organising the move towards that future with more or less detail.

- All futures are **aspirational**, that is they are **attractive targets** that are moved towards. The future is something we cannot fully know and therefore all plans to move into the future have to deal with change.
- All methods for business and information systems development include change and an expectation that the future will need adjustment. Even in heavily contractually driven work **there is always a mechanism to deal with change**.
- There is no such thing (and never has been) as a fixed waterfall lifecycle for delivery of any complex system change at scale.
 - Although there are more or less specific statements of requirements and solution specifications that reflect the need-to-know beforehand verses the need to discover as capability is developed.

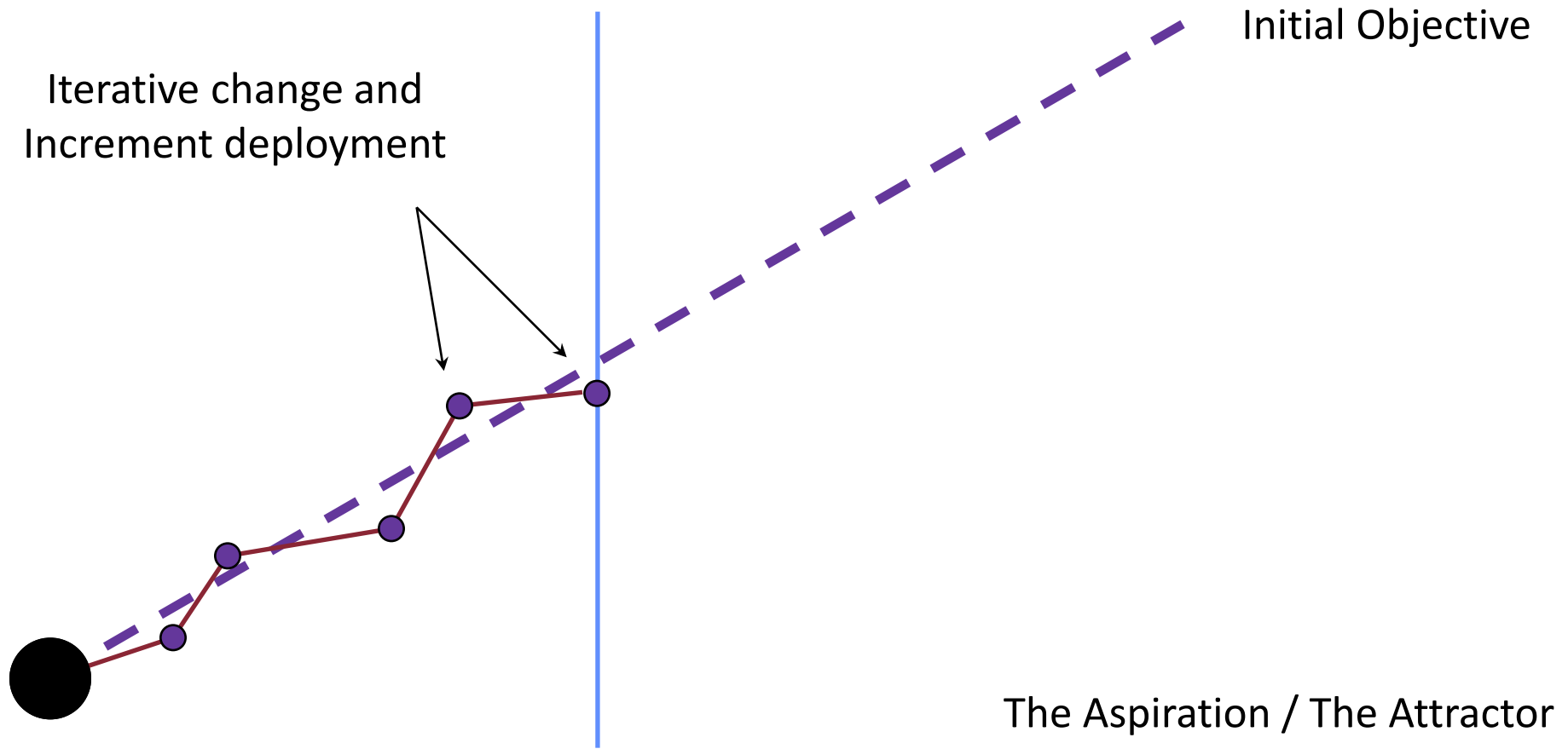
The Characteristic Flow of Change



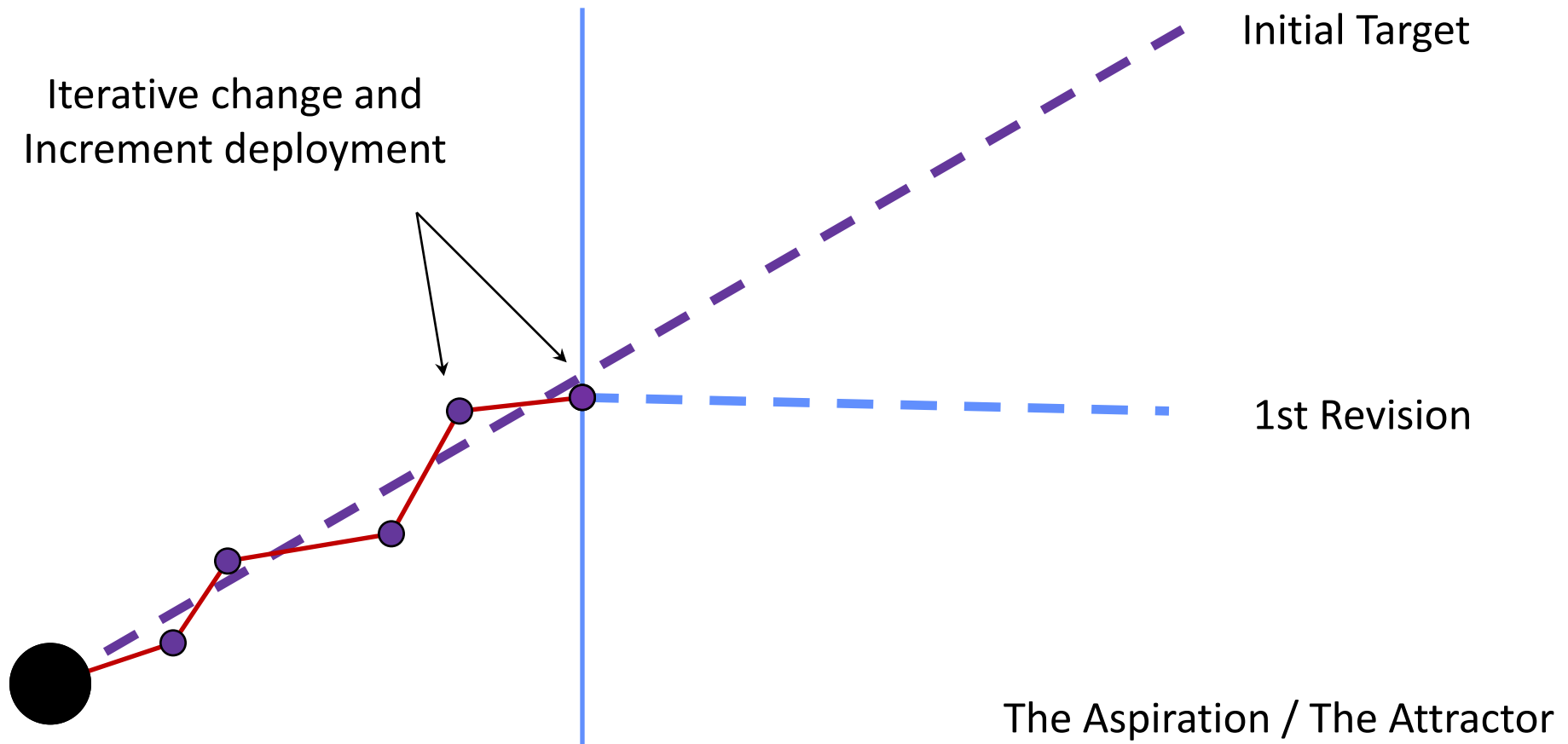
The Characteristic Flow of Change



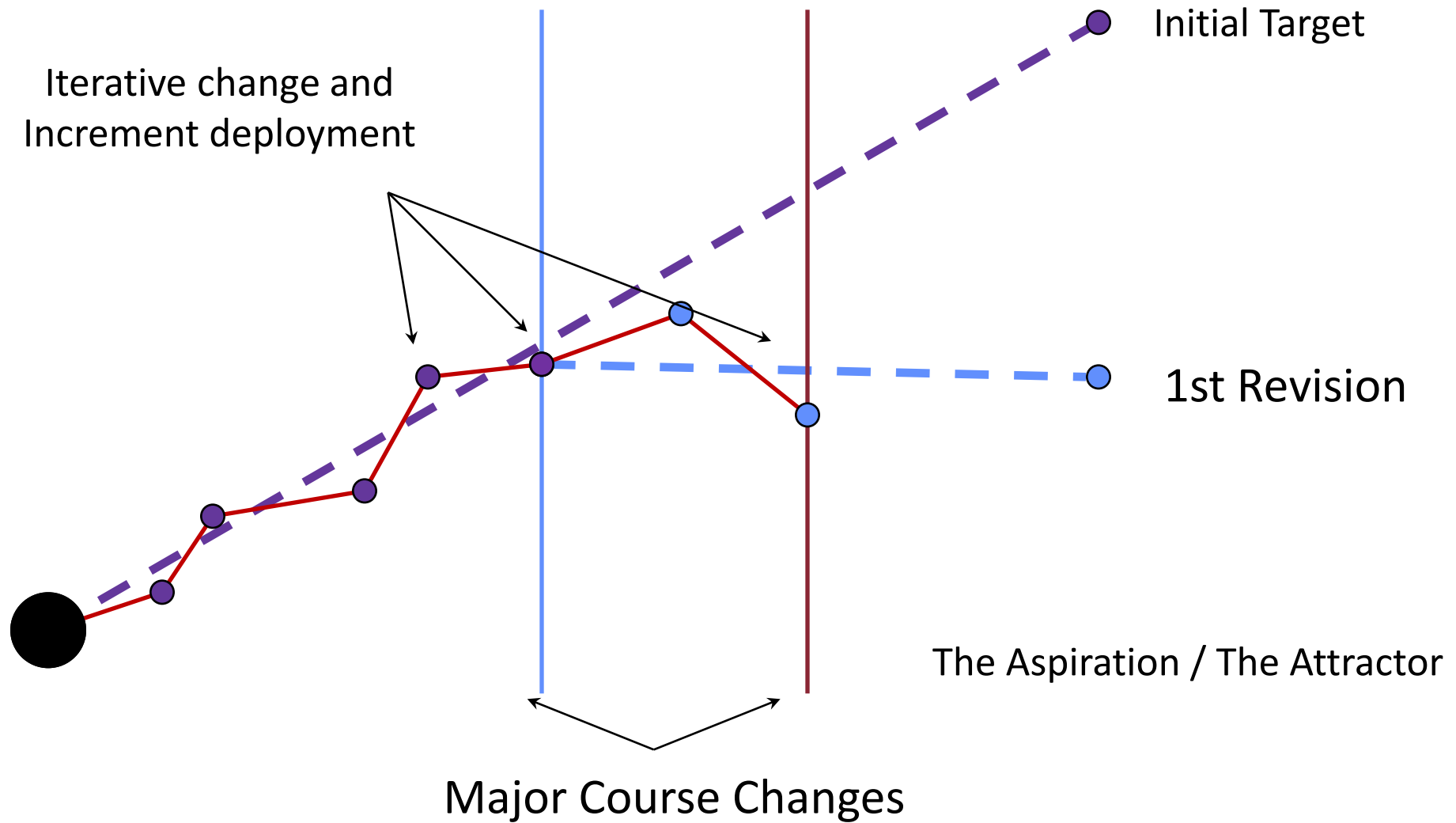
The Characteristic Flow of Change



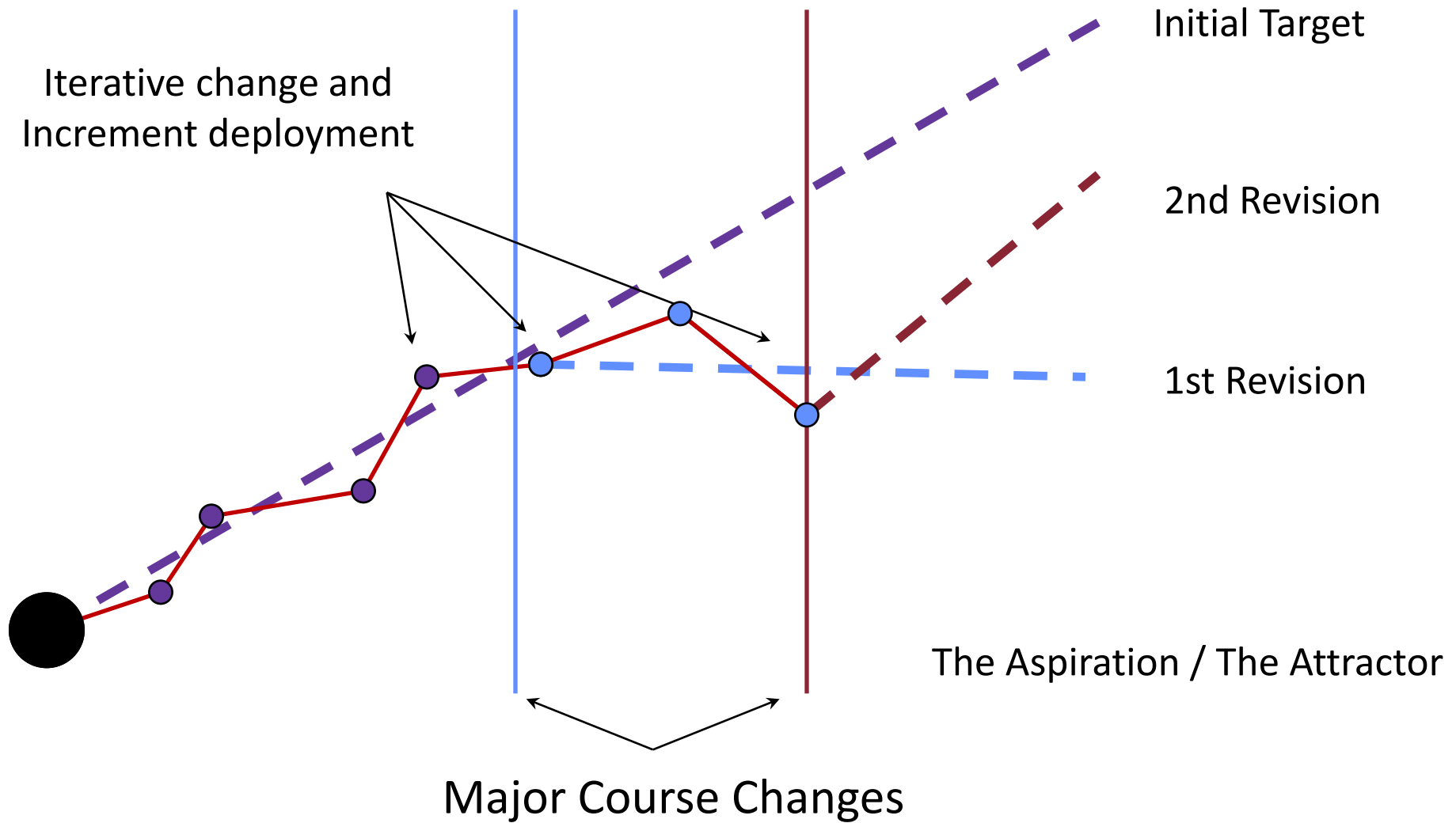
The Characteristic Flow of Change



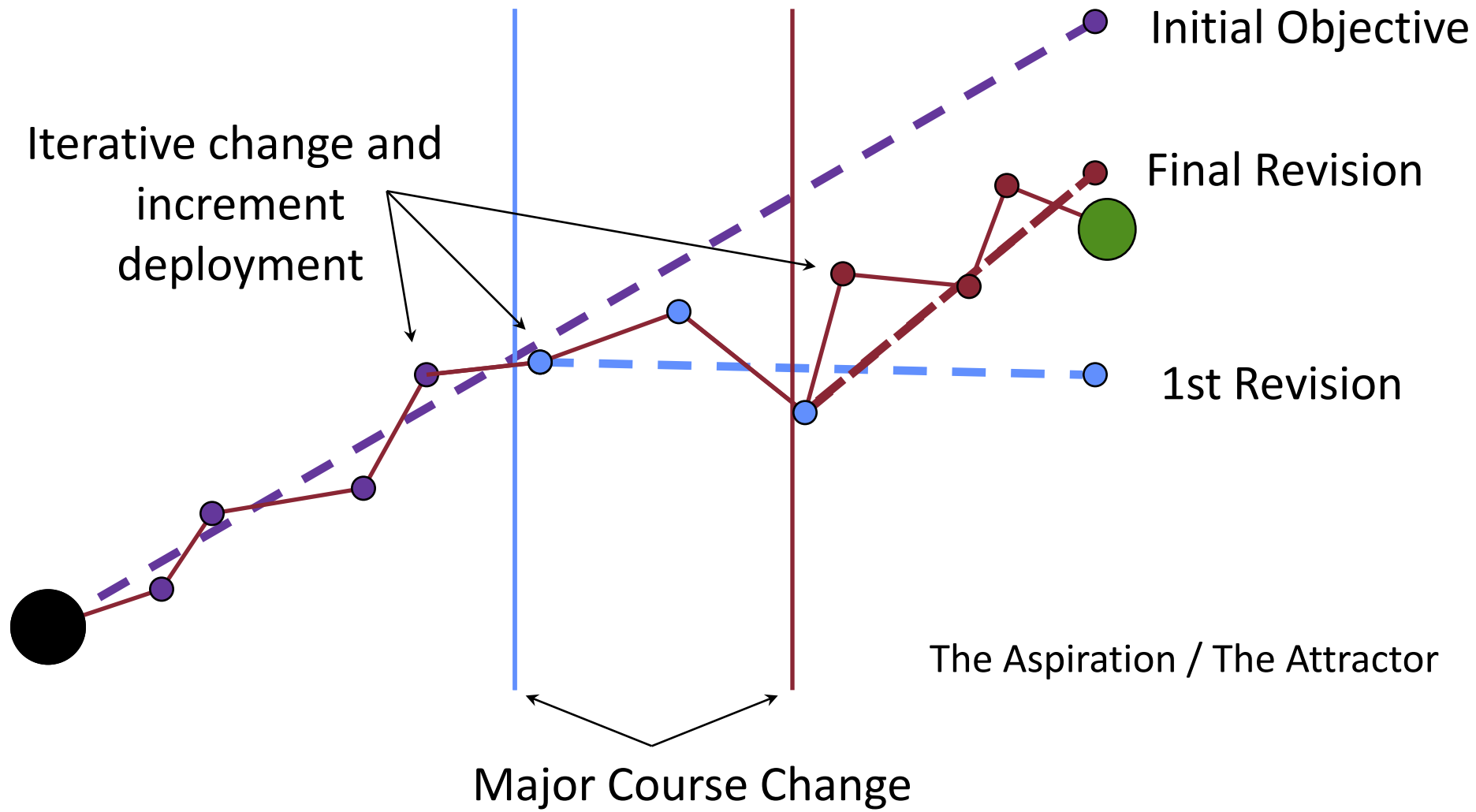
The Characteristic Flow of Change



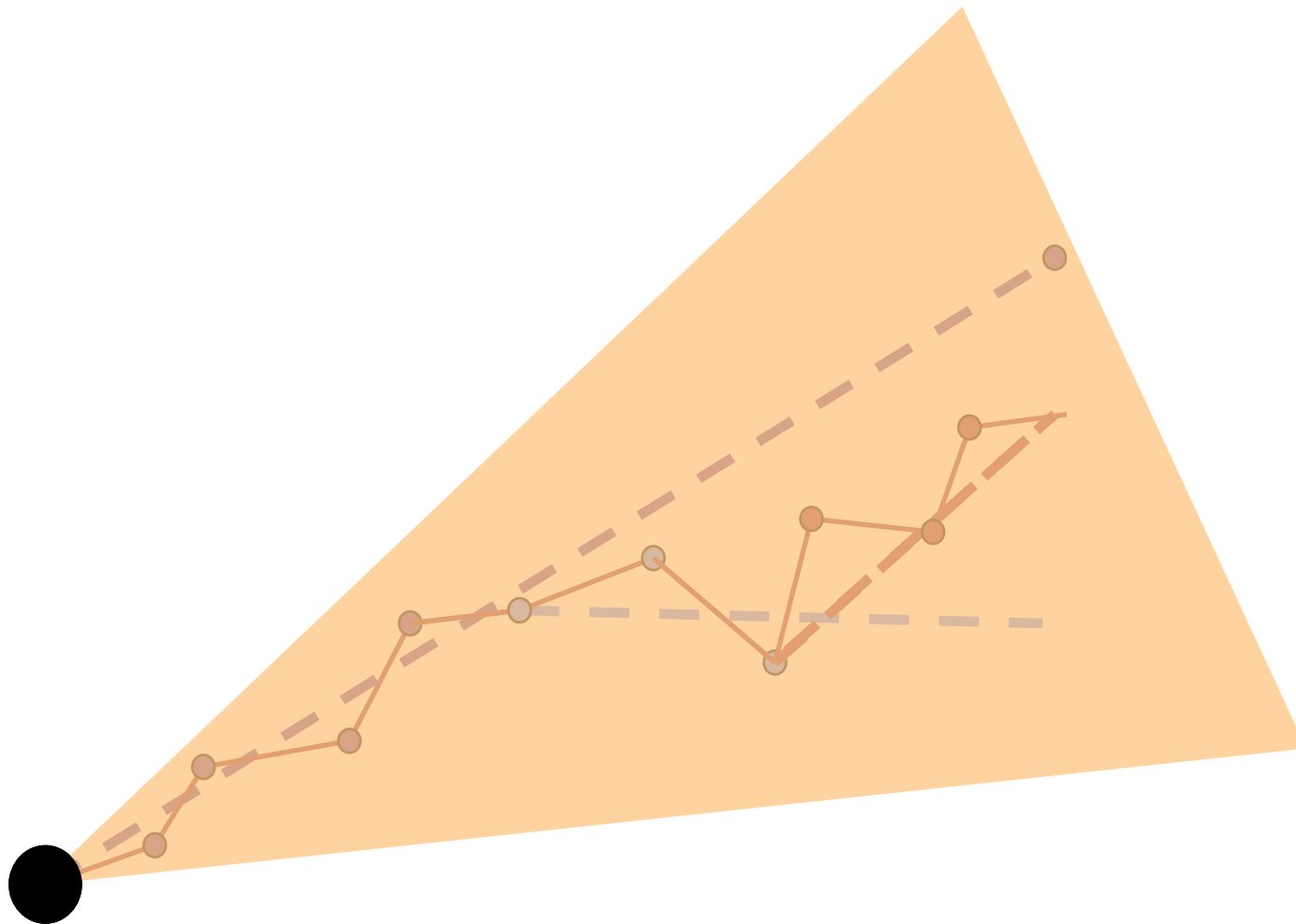
The Characteristic Flow of Change



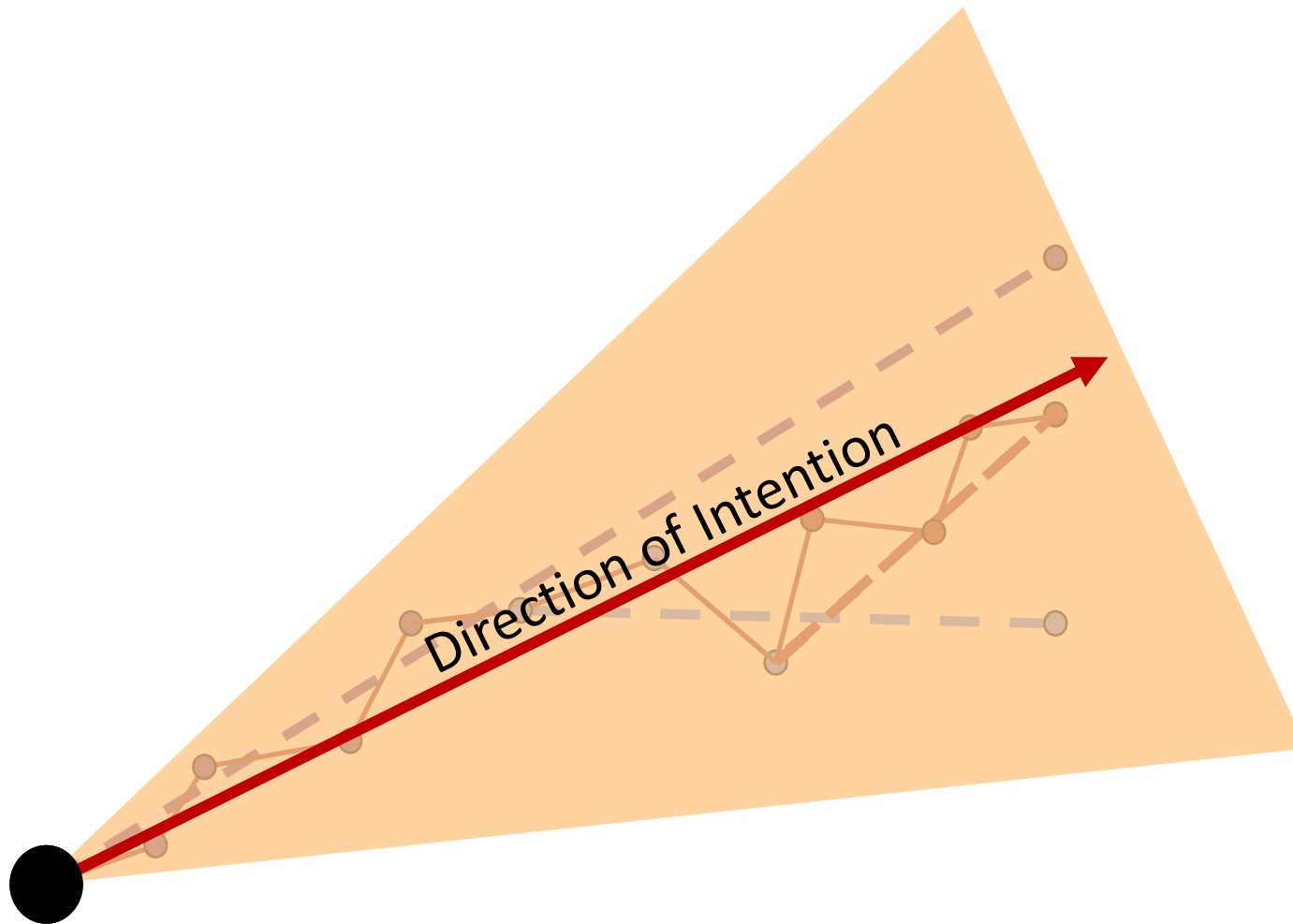
The Characteristic Flow of Change



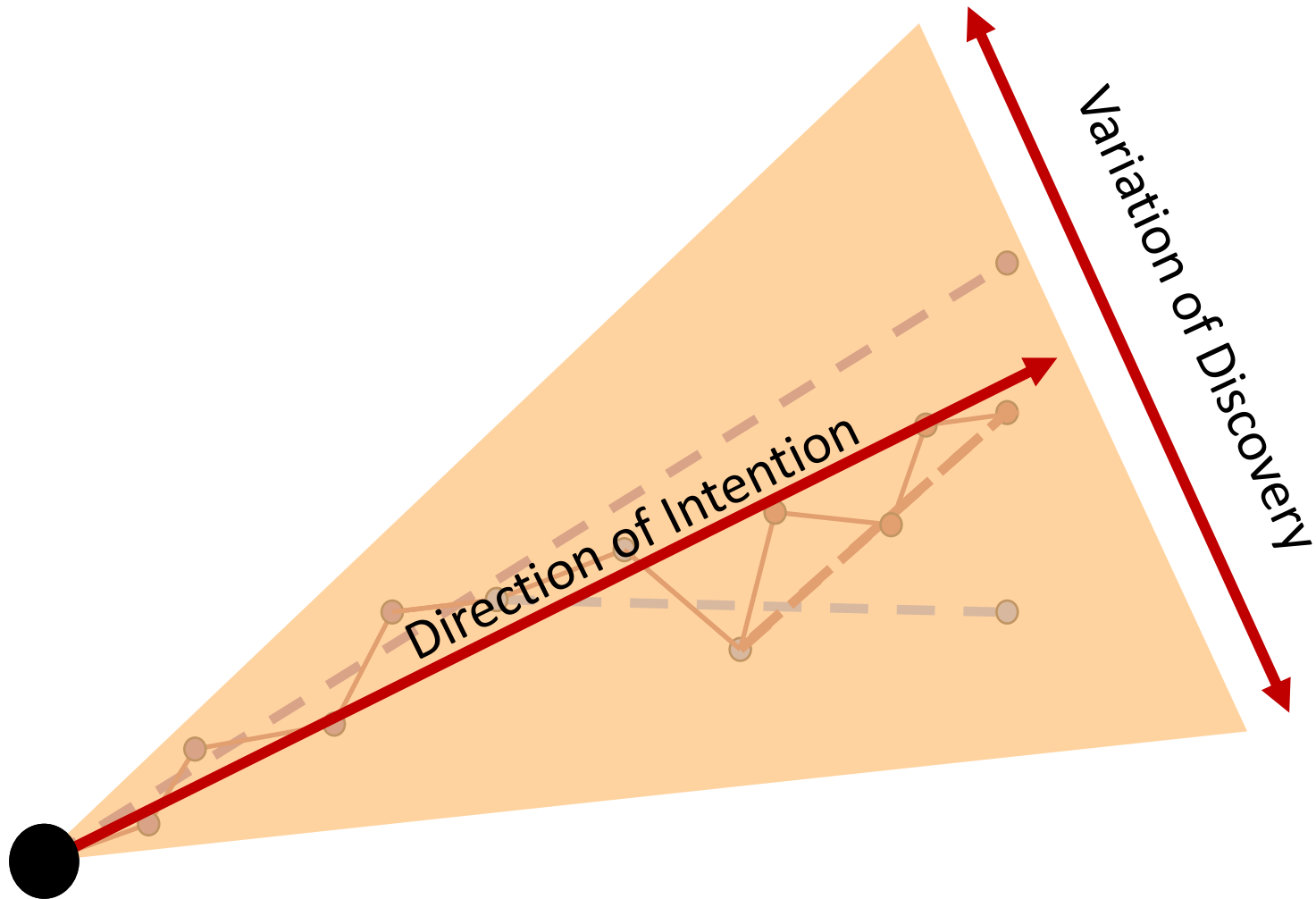
The Change Space



The Change Space

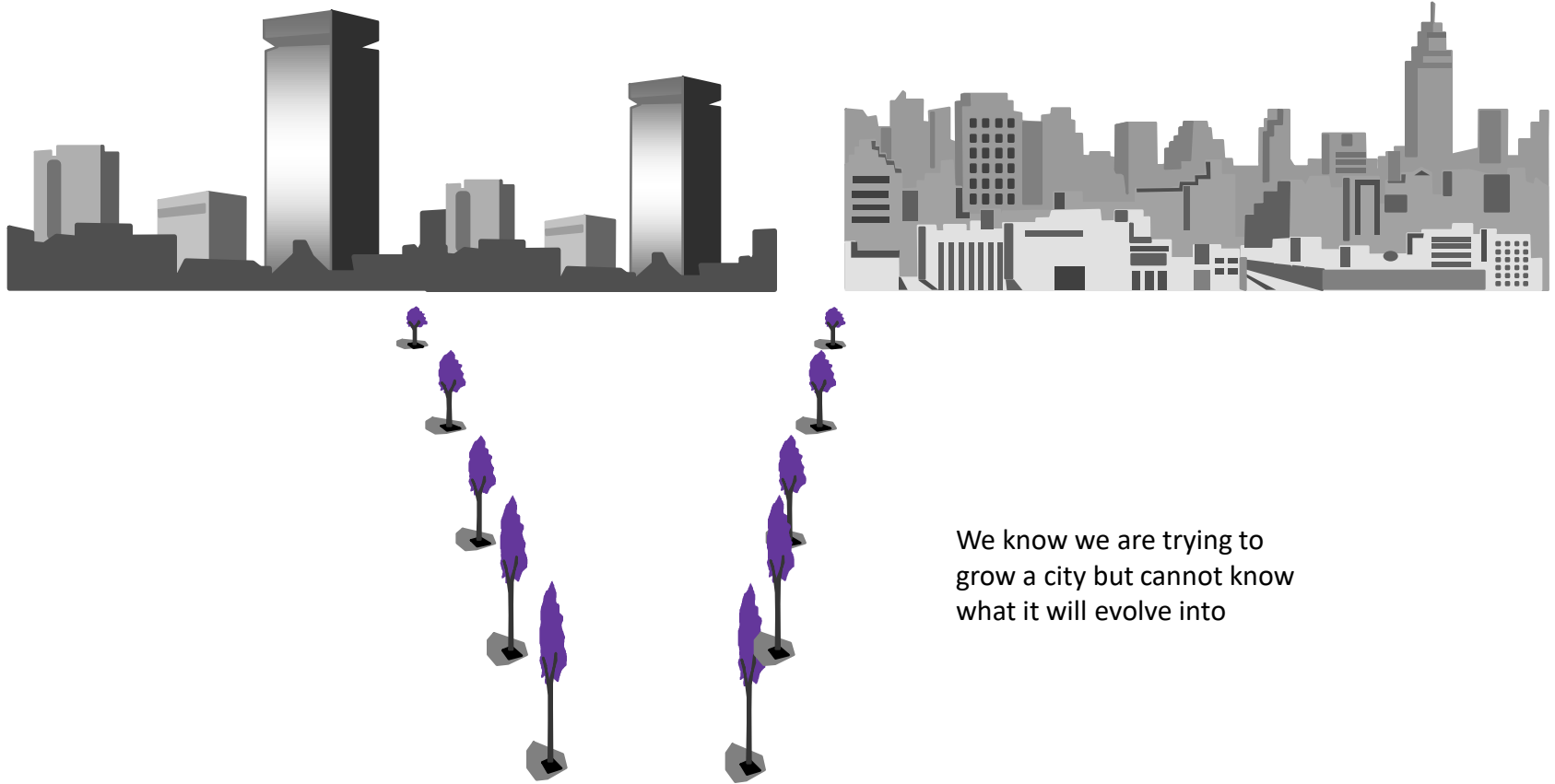


The Change Space



The Need For A Paradigm Shift

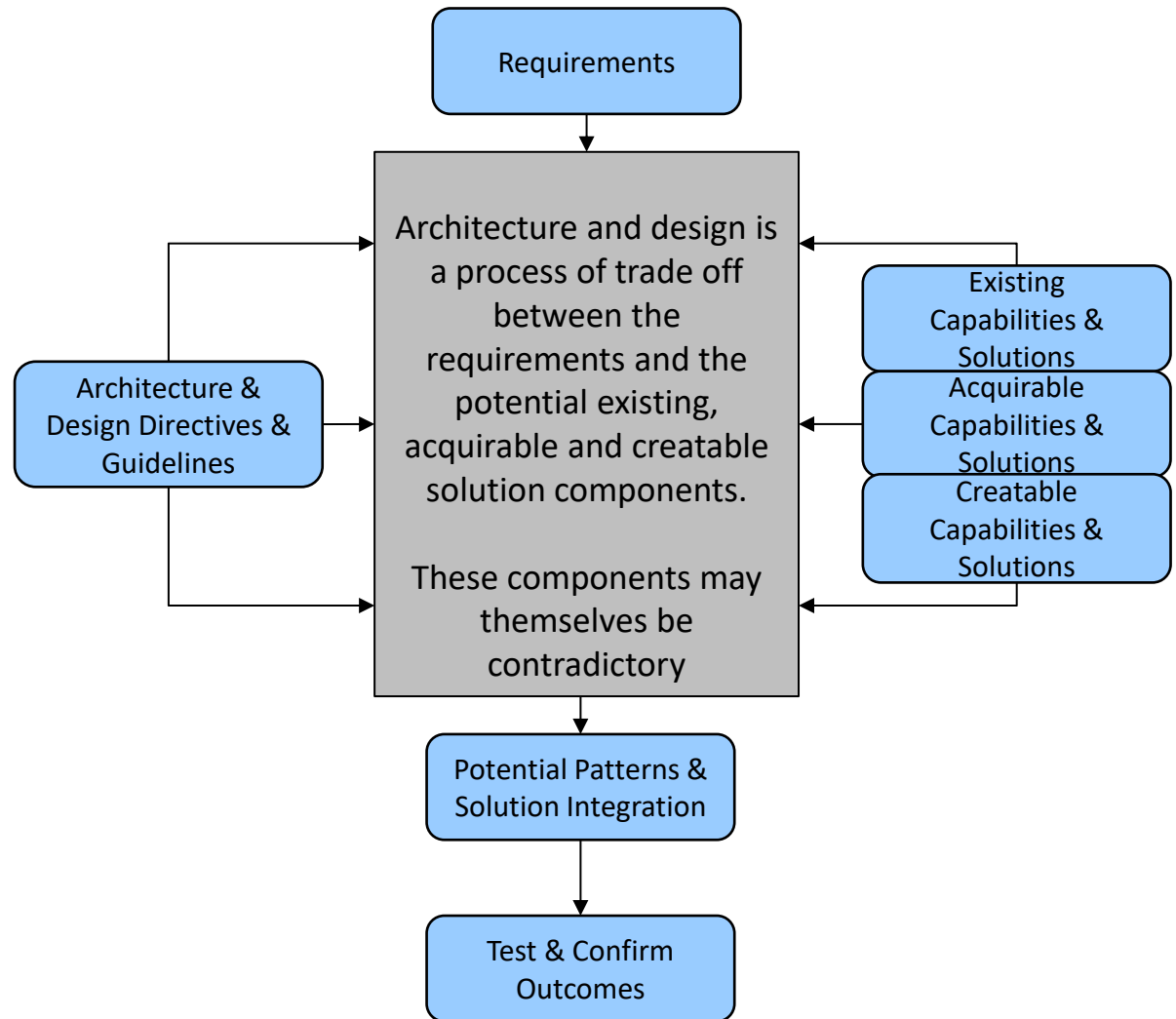
General Destination Known - Specific Destination Unknown



Starting Point

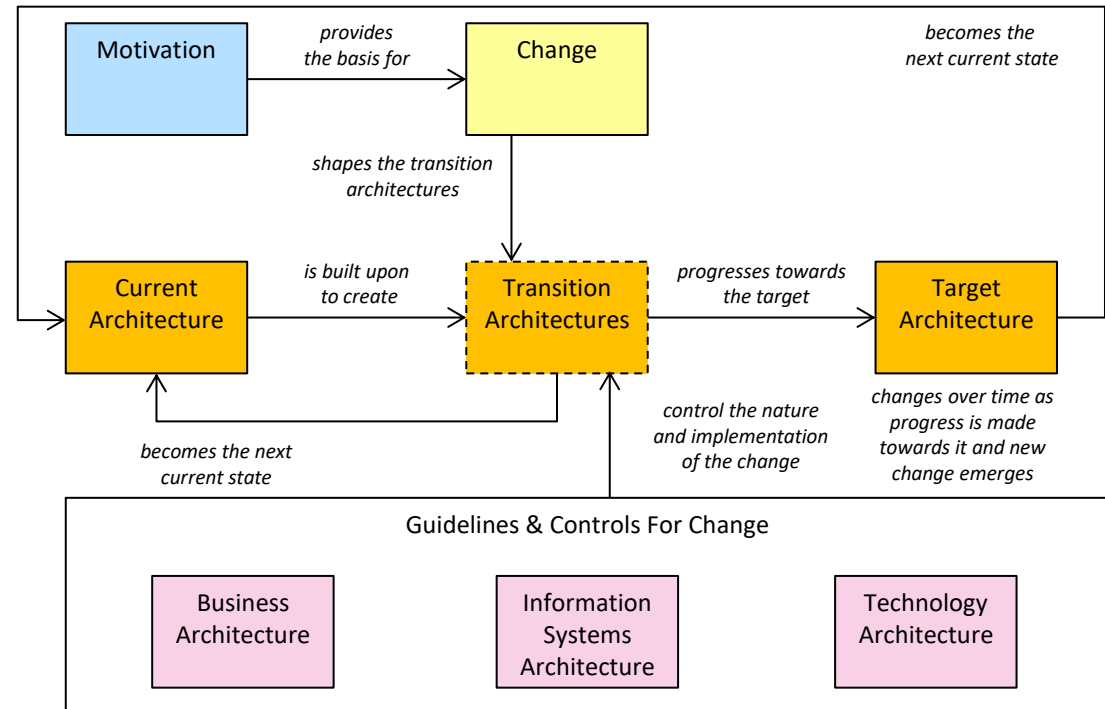
As architecture and design progresses the differences and contradictions between the desired and available properties emerge.

Architecture and design is often a process of trading off between different requirements and the ability of different potential solutions to best meet those requirements.



An enterprise architecture is not static but changes over time. This can be viewed as a series of state definitions representing:

- A current/baseline architecture (the current state)
- Transition architectures (steps on the path towards the target architecture)
- A target architecture (the future desired state)



Each organisation decides which components to formally manage and may create a series of change roadmaps. The most commonly found roadmaps tend to be for:

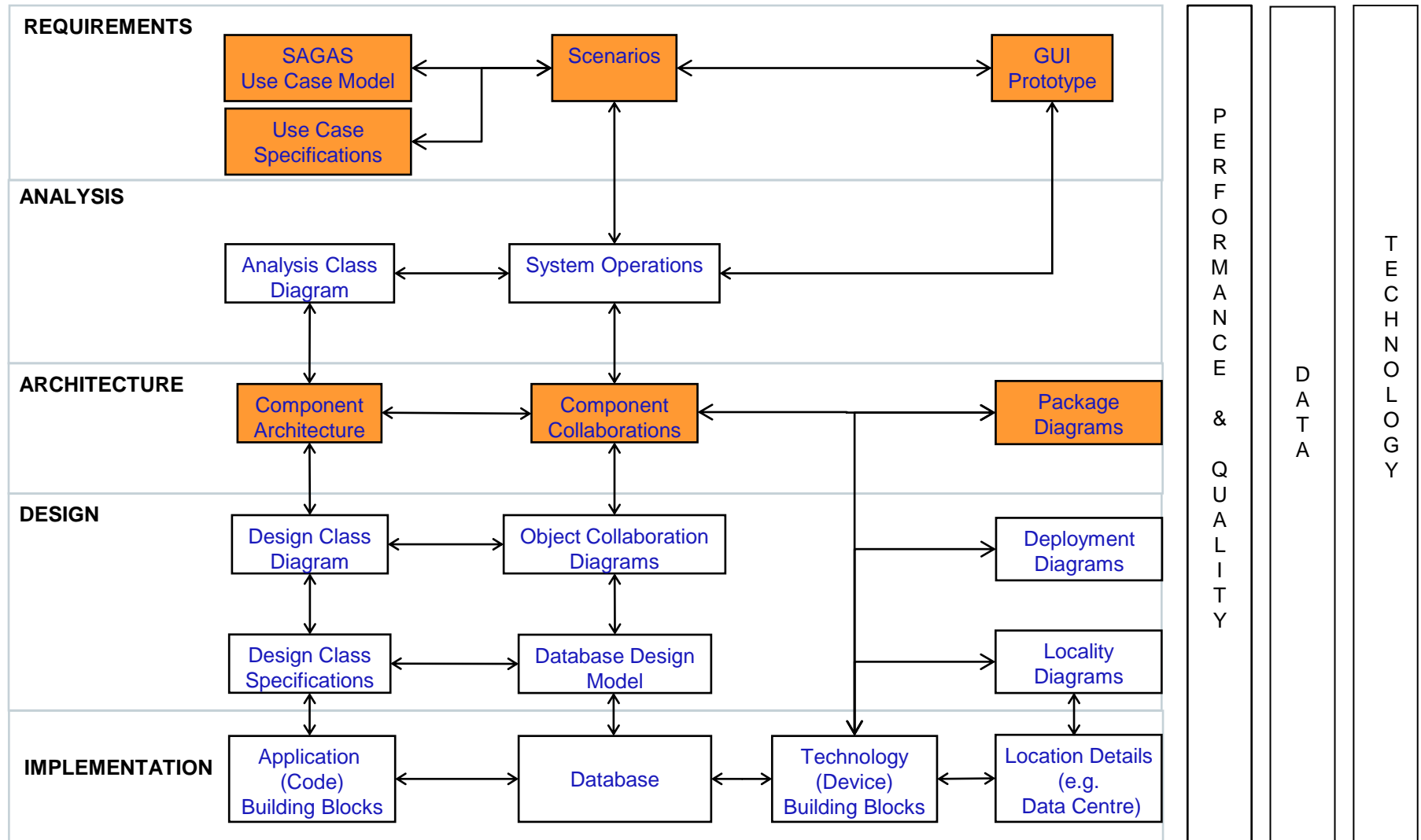
- Business service and process evolution and management (sometimes combined as business portfolios)
- Technology and application evolution (sometimes combined as information system service platforms)
- Information and data evolution and management

EA Tailored For Different Implementation Approaches

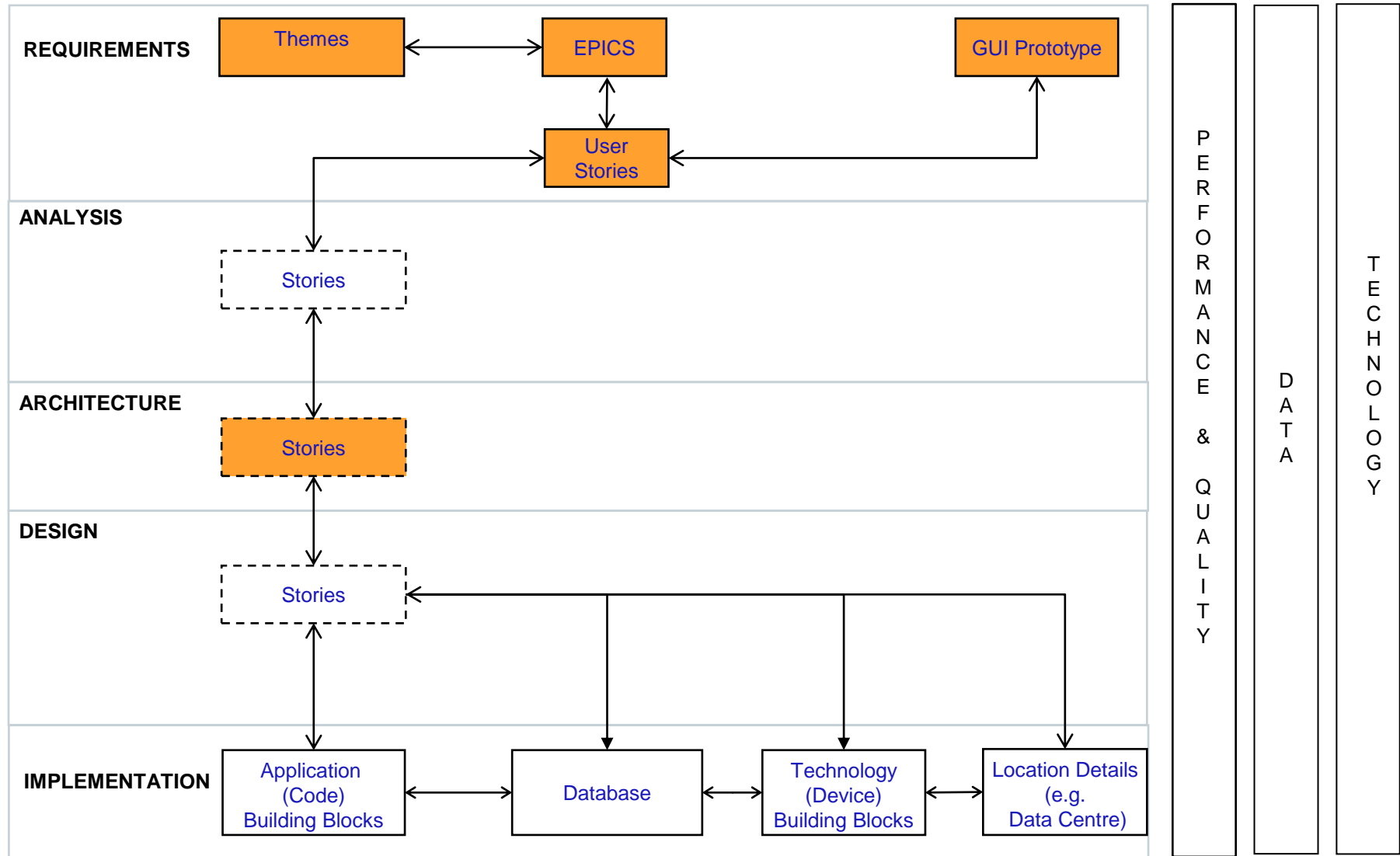
TOGAF And Development and Implementation Methods

	User Story Led Generic Agile	Prototyping Led DSDM	Design Led Unified Method	Architecture Led TOGAF	
Scoping	Theme EPIC / Stories	Business Case Feasibility	Business Case SAGA / Use Case Context	Business Case Vision	Goals & Objectives
Understanding	Stories Acceptance Criteria	Requirements Foundations Models Prototypes	Use Case Descriptions Class/Object Diagrams Package Diagrams Sequence Diagrams Activity Diagrams State Machines	Business Capabilities Architecture Building Blocks Level Across Landscape	+ Requirements
Specification	More Detailed Stories Acceptance Criteria	Solution Architecture Evolutionary Development	Design Class Diagrams Communication Diagrams Package Diagrams Component Diagrams Deployment Diagrams	Transition Architecture Architecture Contract Solution Building Blocks	+ Constraints
Implementation	Build Acceptance Criteria	Engineering Review & Benefits Assessment	Implementation Test	Implemented Solution Building Bocks Implementation Governance	KPIs + Deployment Environment

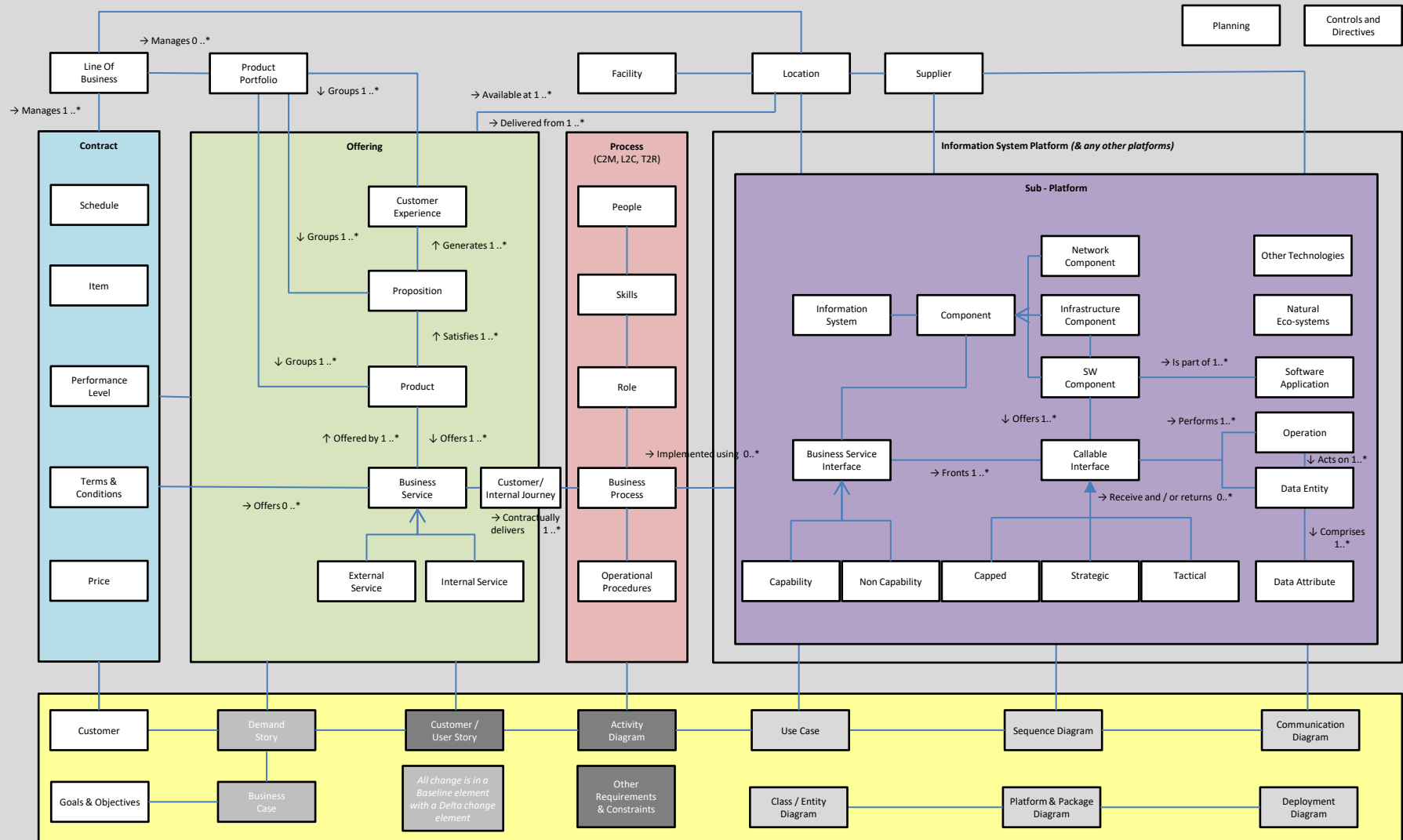
Example Deliverables Of Full Software Engineering Approach



Example Deliverables Of More Limited Agile Engineering Approach

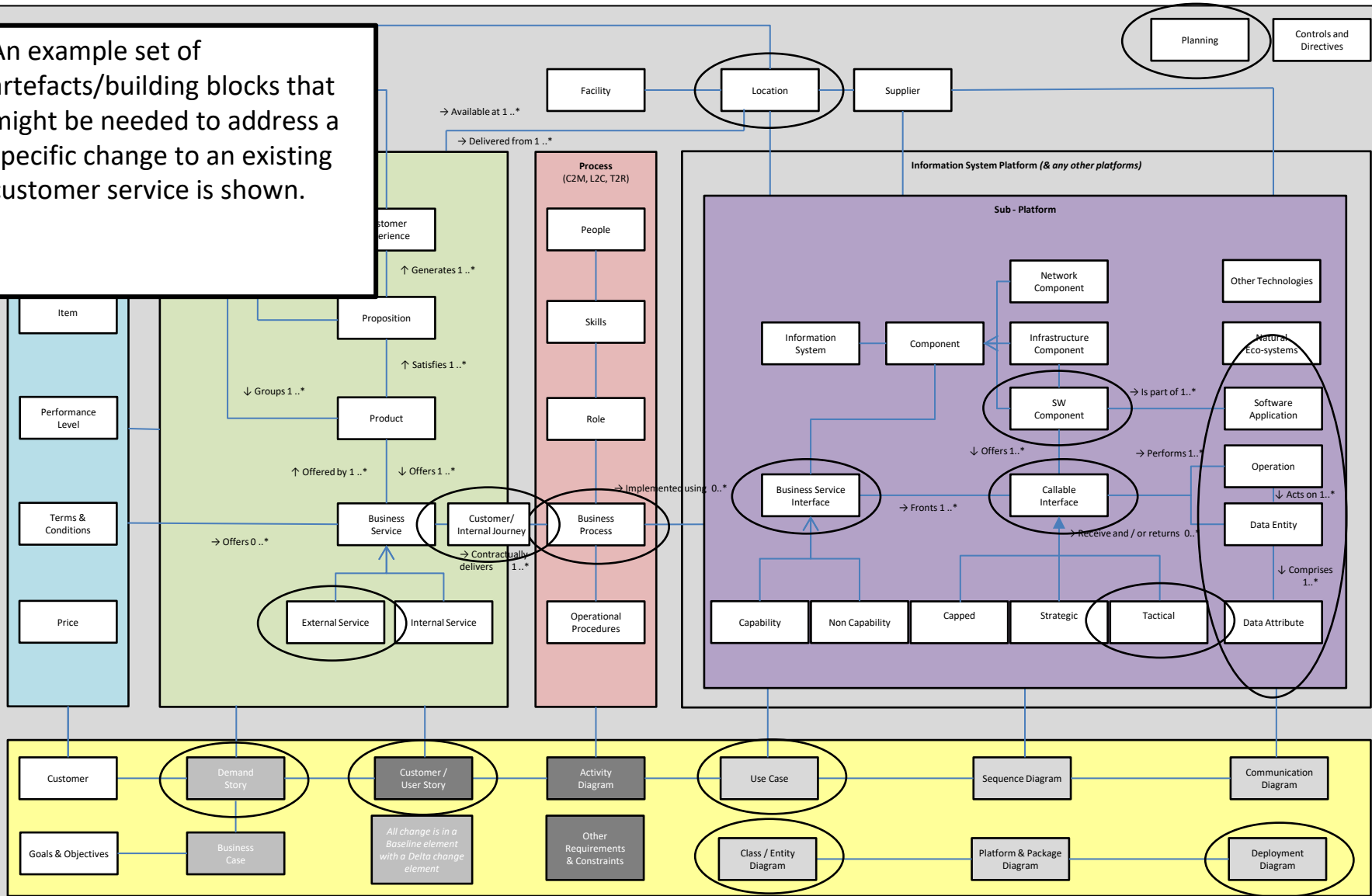


Example Aligning EA, Stories and Use Cases

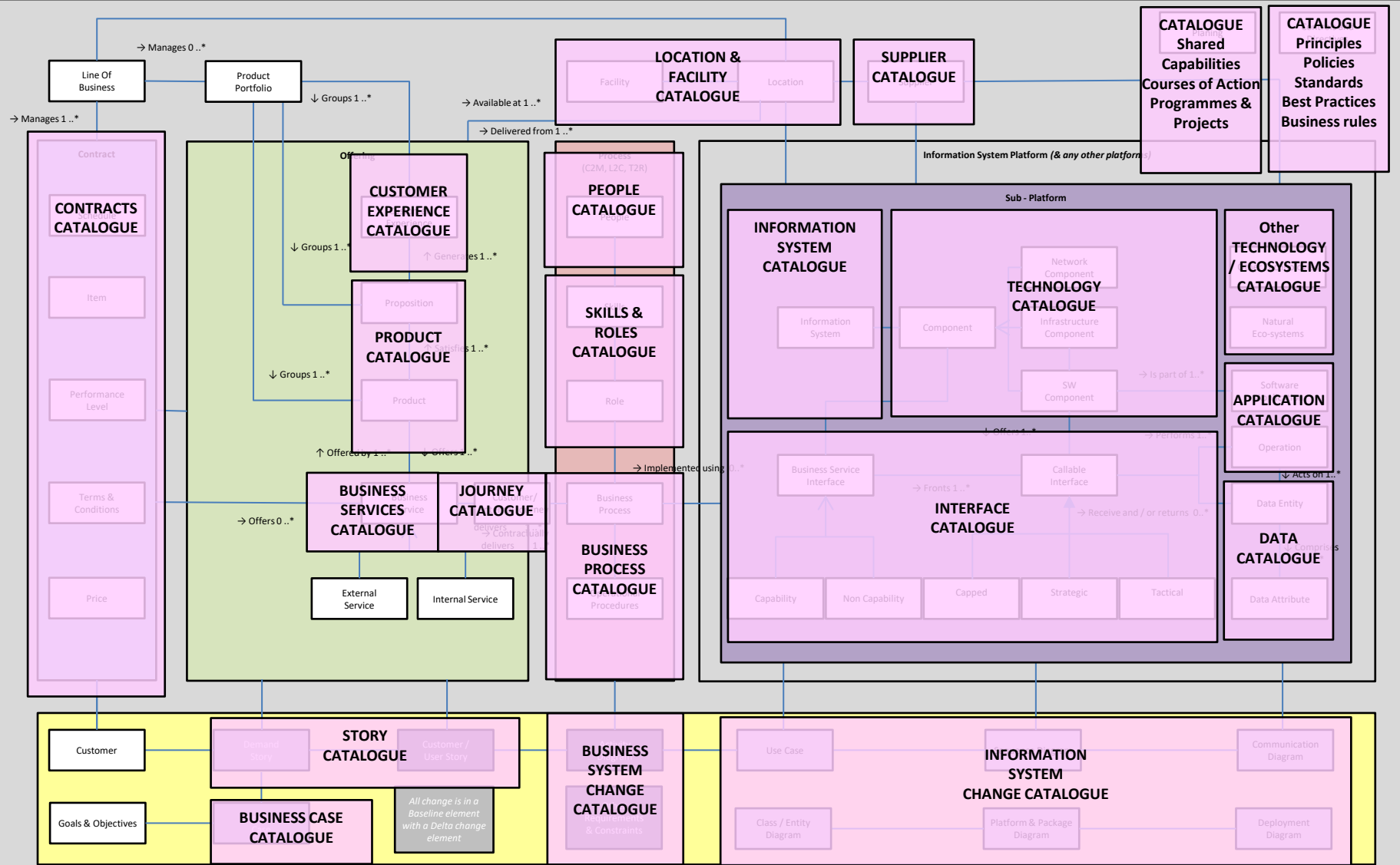


Example Solution Building Block Set

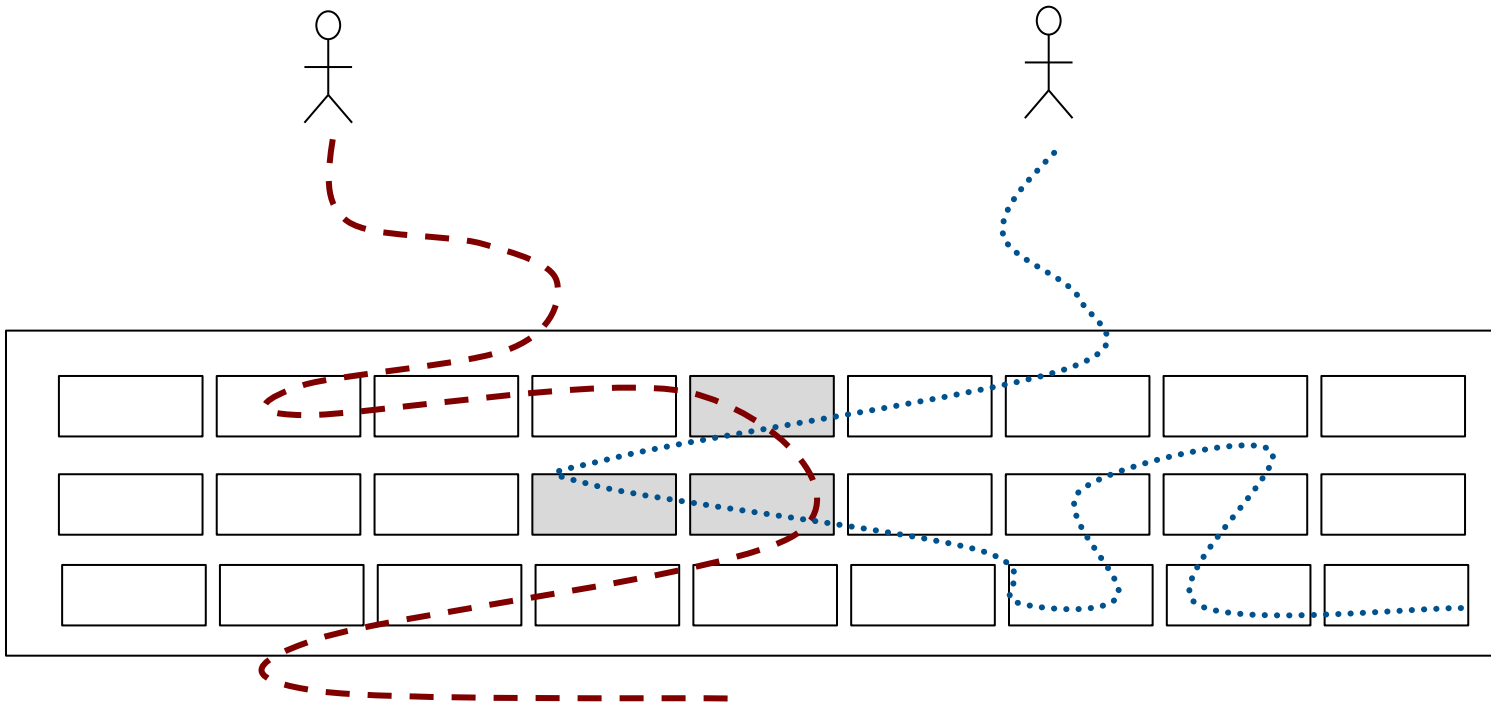
An example set of artefacts/building blocks that might be needed to address a specific change to an existing customer service is shown.



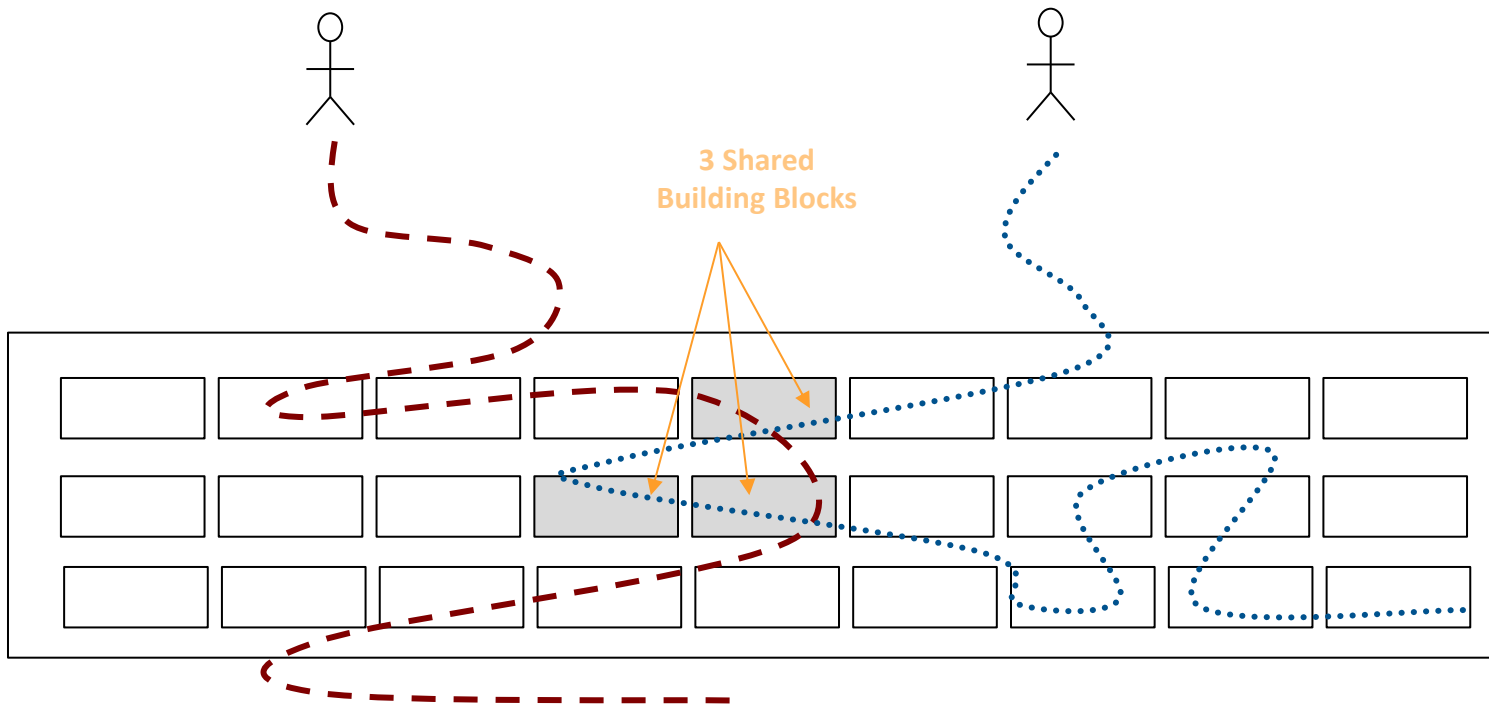
Example Catalogues



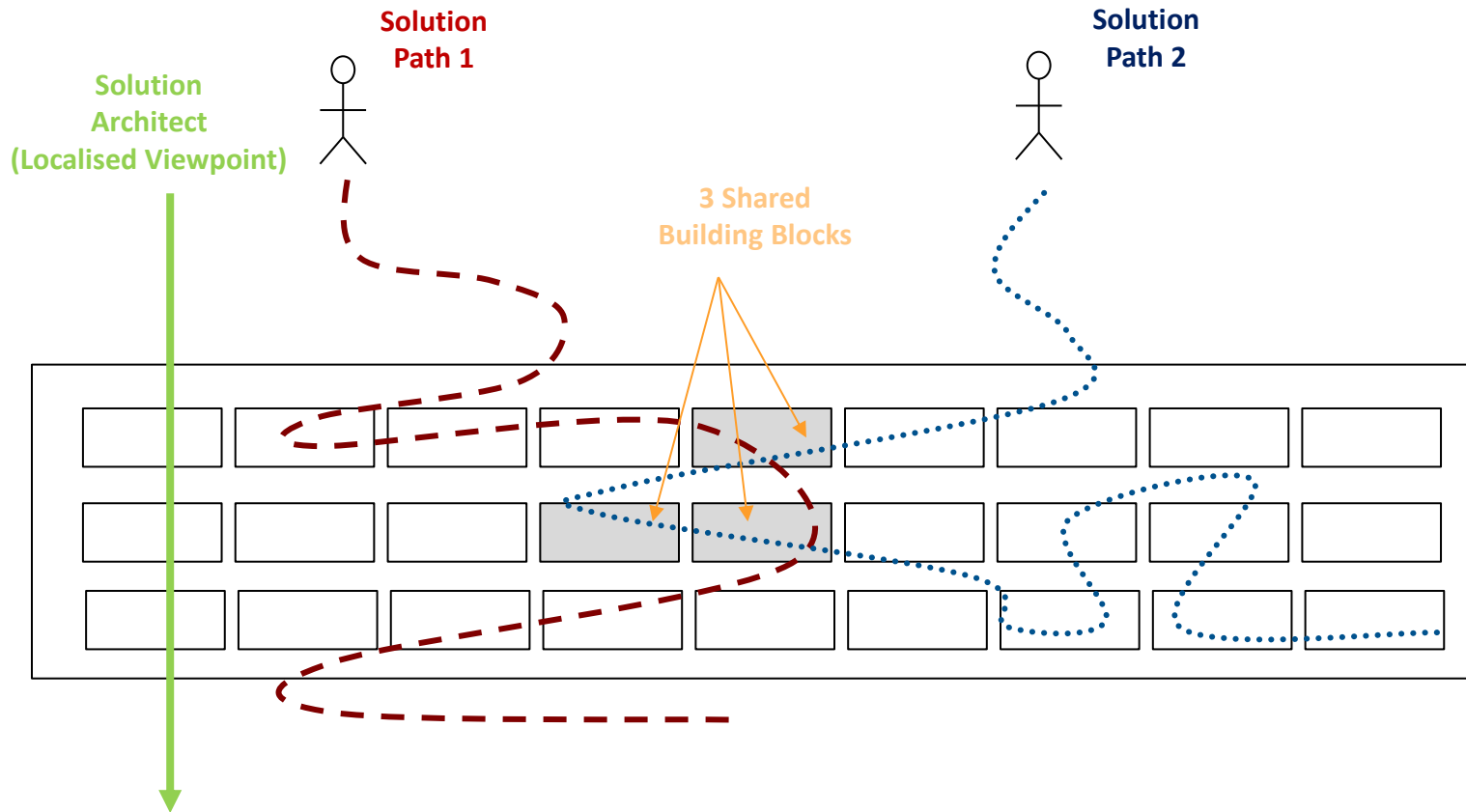
Global vs Local vs Specific Viewpoints



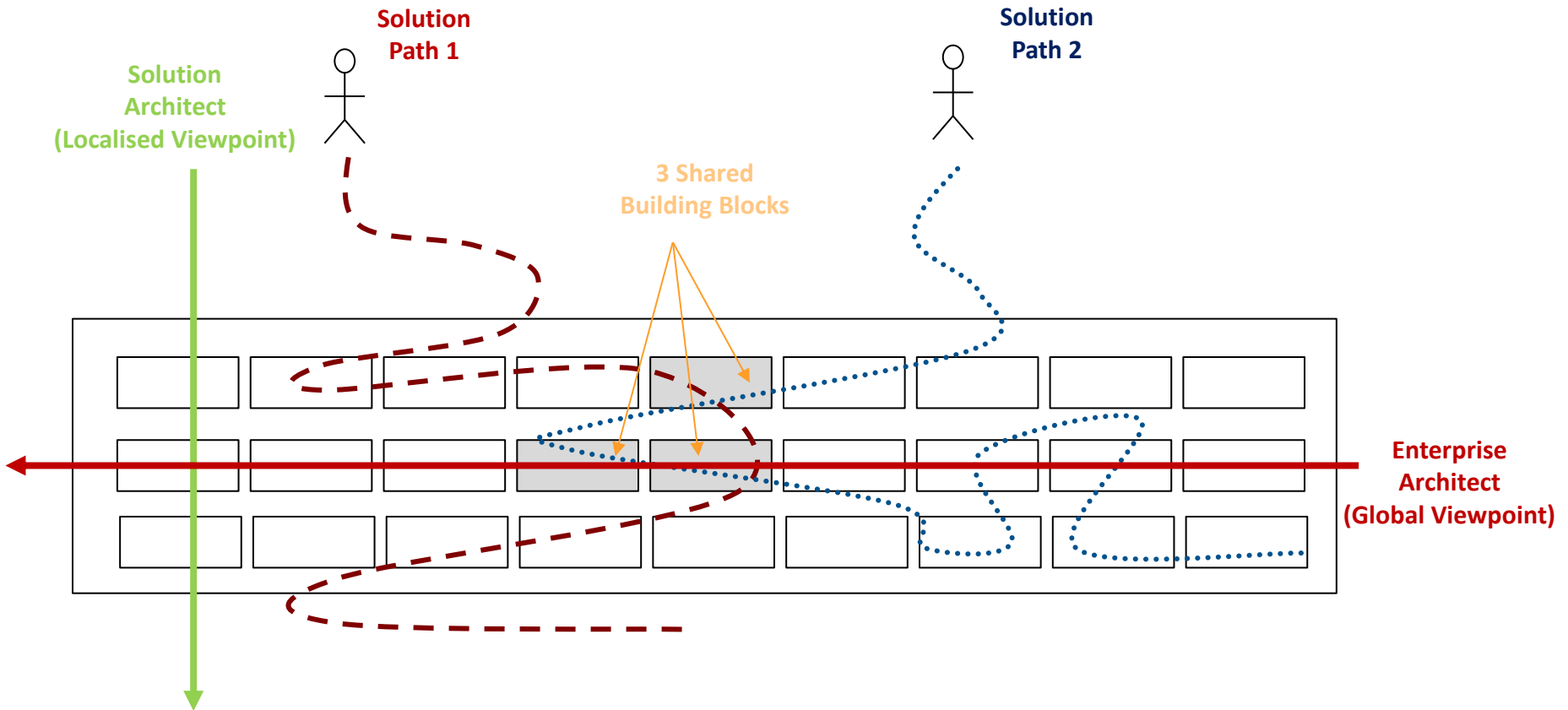
Global vs Local vs Specific Viewpoints



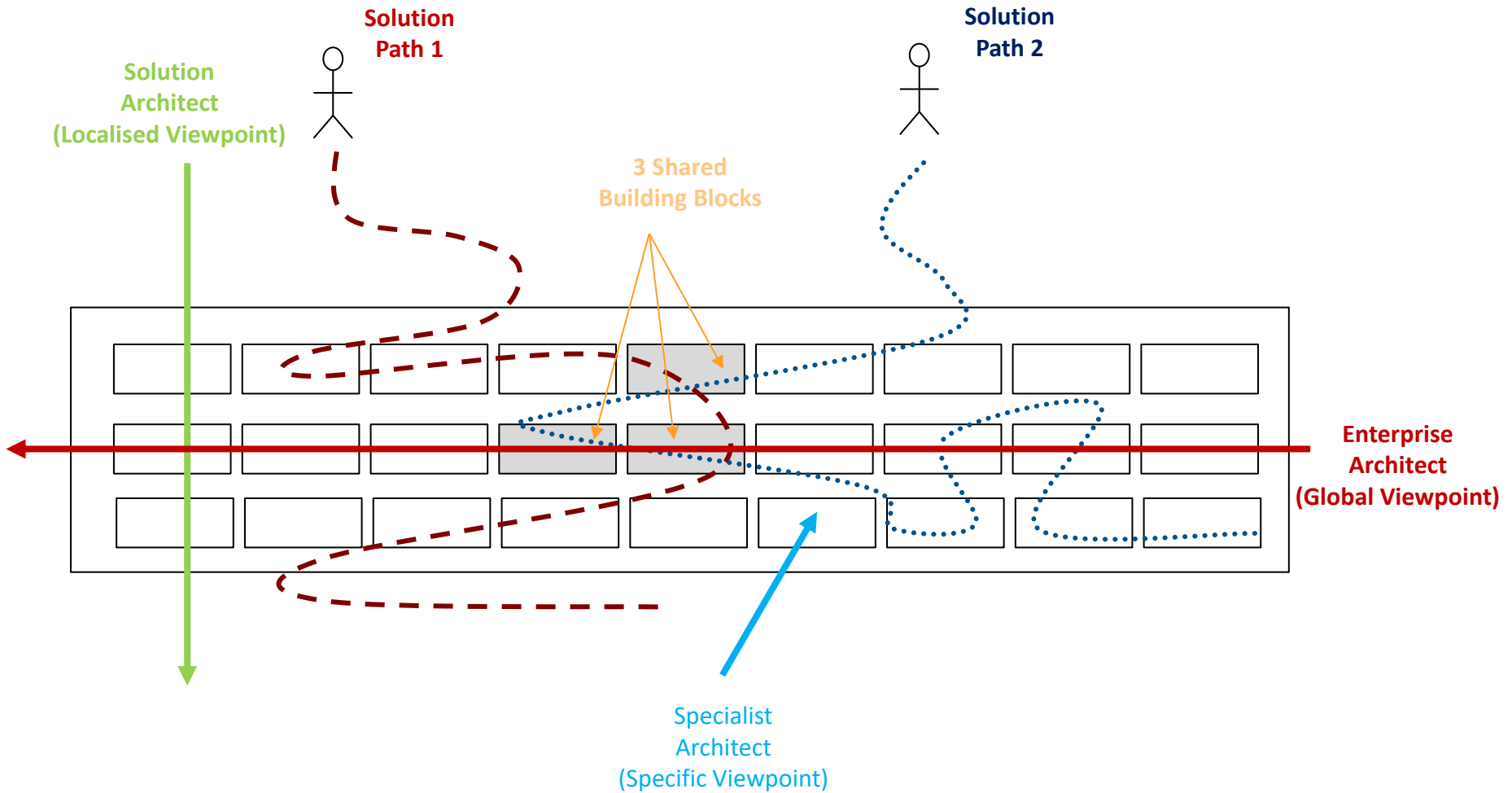
Global vs Local vs Specific Viewpoints



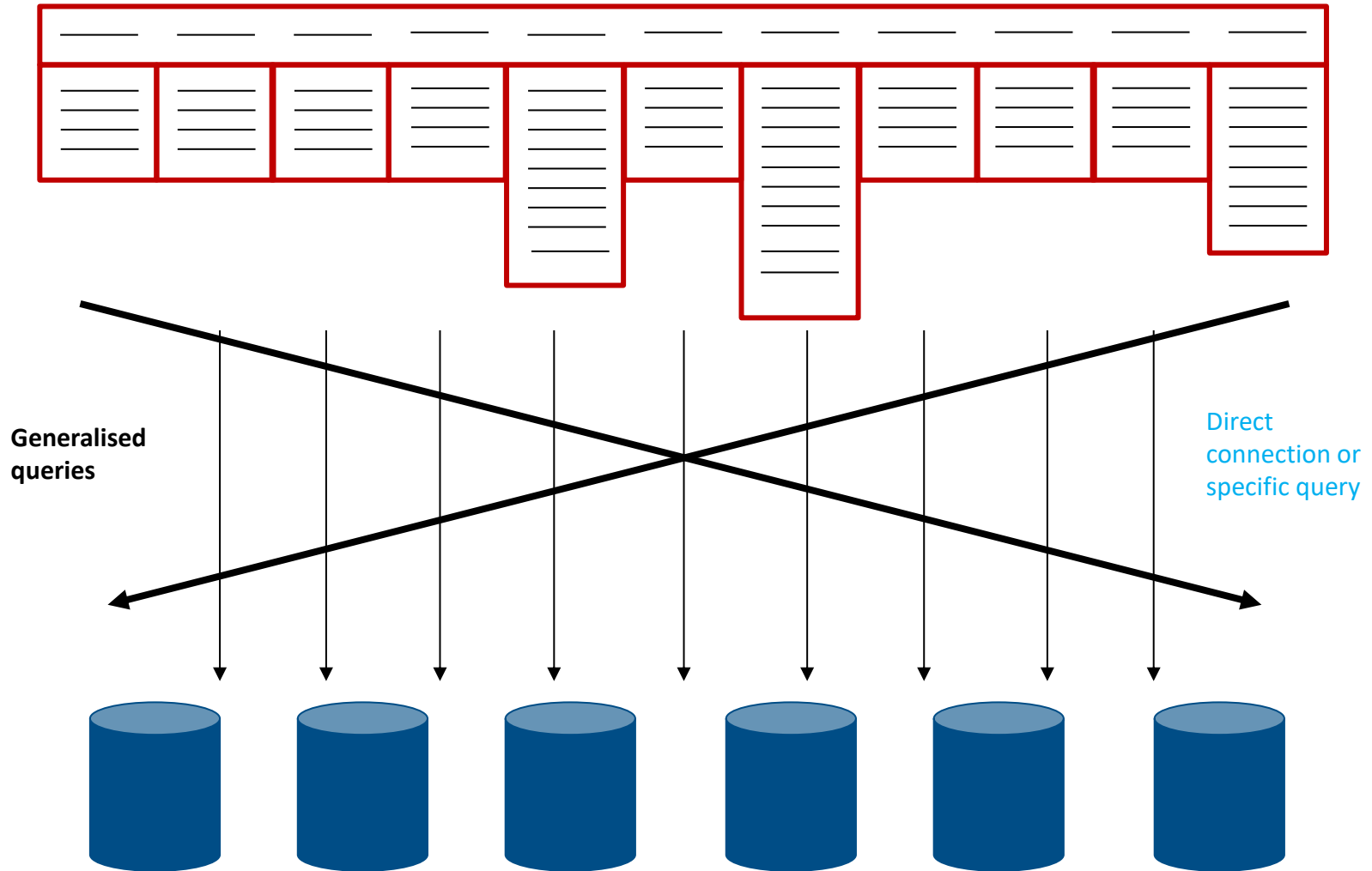
Global vs Local vs Specific Viewpoints



Global vs Local vs Specific Viewpoints

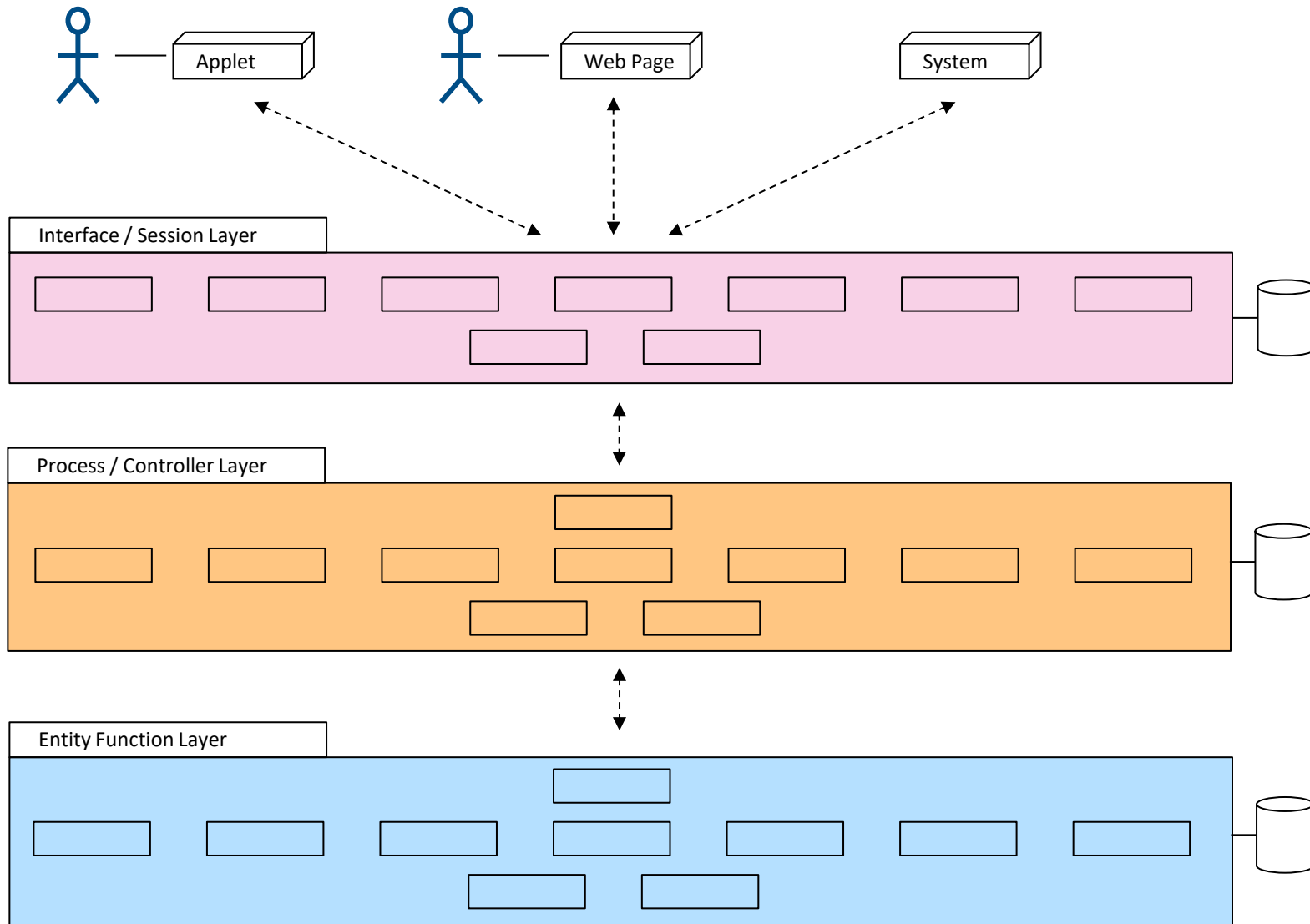


Portals & Repositories

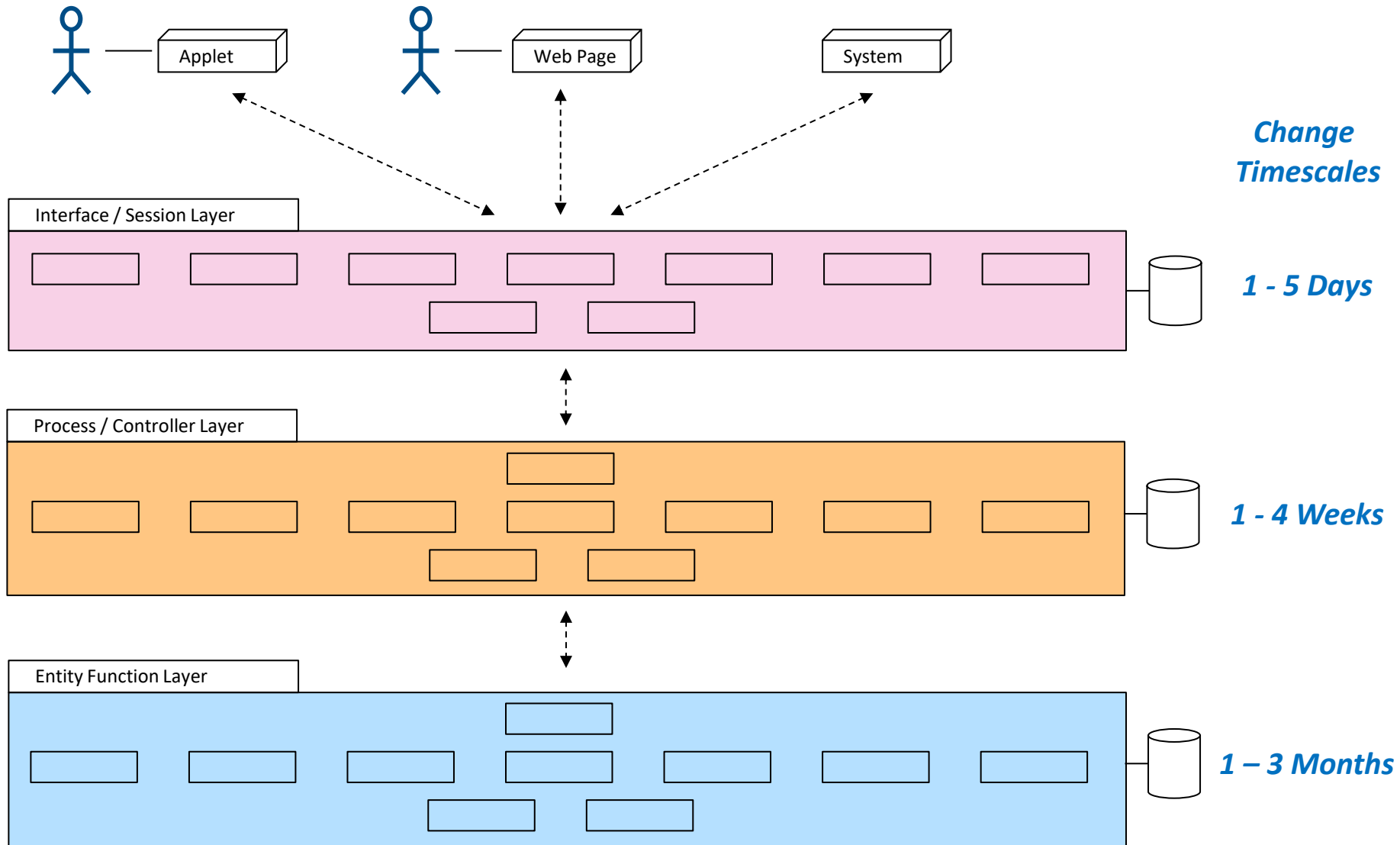


Layering Information System Solution Components & Scope

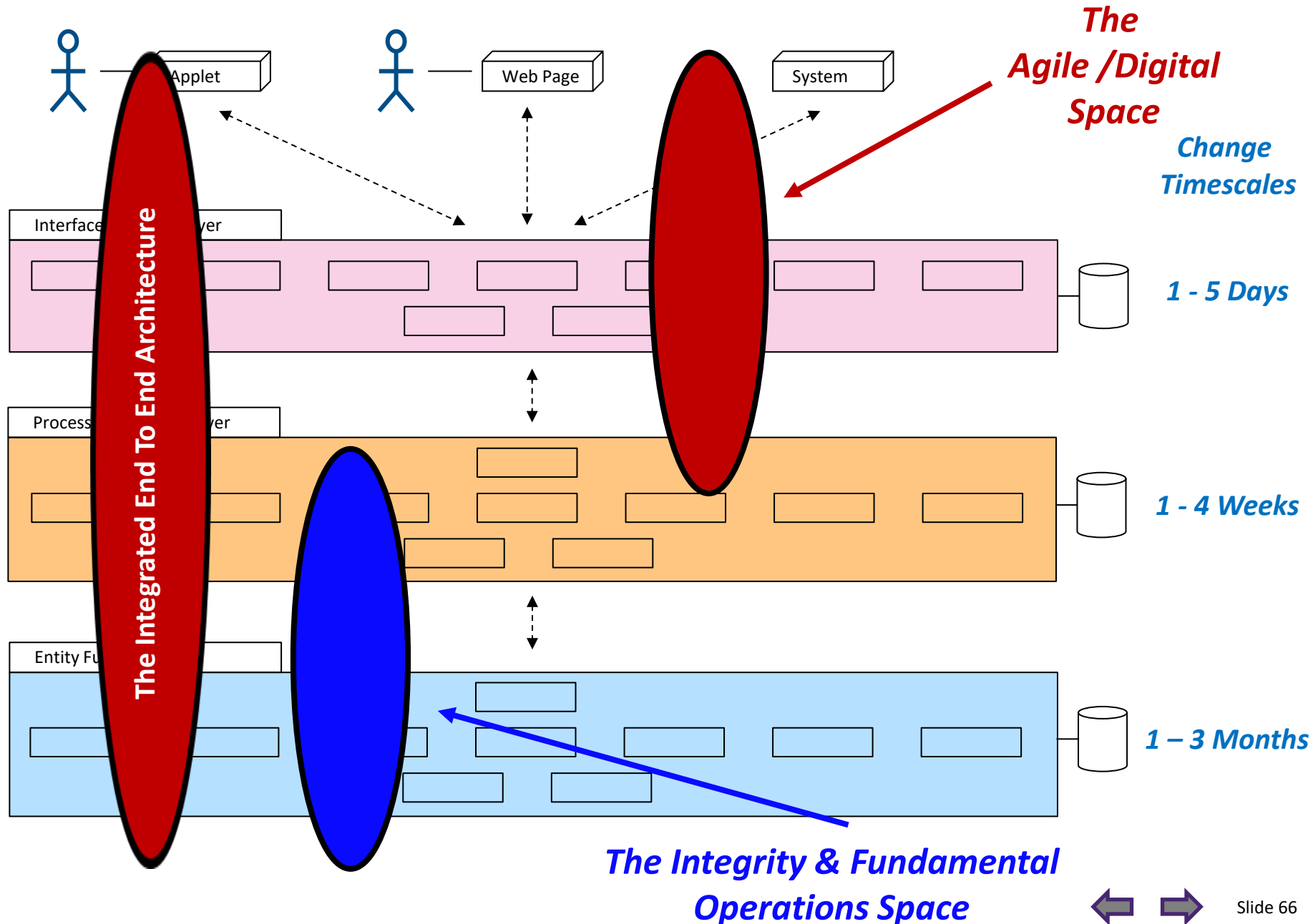
UML Object Types (Interface / Controller / Entity Function)



Change Timescales



Characteristic Viewpoints For Management & Control



High Level Building Blocks - Services & Modules

Service Oriented Architecture

Architecture and design is about bringing different components together to deliver a set of outcomes.

Service Oriented Architecture (SOA) is an approach to both business and technical services based on a set of principles about how components can be defined and interact in a managed and visible manner (based on ideas developed about contracts).

SOA principles provide the basis for defining and implementing services that integrate components to deliver specified business outcomes.

A **service**

represents the delivery of an agreed outcome, by a provider within the bounds of defined performance, quality and service levels; in response to a request by a client.

A **business service** delivers a specific outcome to a customer using the relevant resources of a business (people, organisation, process, information and technology)

In some cases, the business service can be delivered by just using none, one or more software services, in other situations additional elements such as visiting buildings, talking to people, delivering or installing physical devices; may also be needed

A **software service** delivers a specific outcome in terms of an internal information system state change and/or a return of information; in response to an identified system request/event

The General Properties Of A Service

The "SOA Manifesto Working Group", have defined a SOA Manifesto.

Its aim is to provide clarity about the basic properties of a good service oriented architecture.

The manifesto provides set of overall general tendencies for a good service oriented architecture that value:

- Business value over technical strategy.
- Strategic goals over project-specific benefits.
- Intrinsic interoperability over custom integration.
- Shared services over specific-purpose implementations.
- Flexibility over optimization.
- Evolutionary refinement over pursuit of initial perfection.

These are underpinned by a set of basic properties that should apply to the components within a SOA:

- Reuse
- Granularity
- Modularity
- Composability
- Componentization
- Interoperability
- Standards-compliance (both common and industry-specific)
- Service identification and categorization.

The Specific Properties Of Effective Services

SOA tends to focus on the properties of software services. The properties listed show key aspects of well formed SW services.

These aspects are equally important when applied to business services (within which the software services operate).

As an architect or designer, you should always be aware of these aspects when designing the integration of components for contracts, products and services. The more that each of these compositions support these aspects the greater the potential for future reuse.

- **Standardized Service Contract** – Services adhere to an agreed contract, as defined collectively by one or more service-description documents.
- **Service Loose Coupling** – Services maintain a relationship that minimizes dependencies and only requires that they maintain an awareness of each other.
- **Service Abstraction** – Beyond descriptions in the service contract, services hide logic from the outside world.
- **Service Reusability** – Logic is divided into services with the intention of promoting reuse.
- **Service Autonomy** – Services have control over the logic they encapsulate.
- **Service Granularity** – A design consideration to provide optimal scope and the right granular level of the business functionality in a service operation.
- **Service Statelessness** – Services minimize resource consumption by deferring the management of state information when necessary
- **Service Discoverability** – Services are supplemented with communicative meta data by which they can be effectively discovered and interpreted.
- **Service Composability** – Services are effective composition participants, regardless of the size and complexity of the composition.
- **Service Normalization** – Services are decomposed and/or consolidated to a level of normal form to minimize redundancy. In some cases, services are denormalised for specific purposes, such as performance optimization, access, and aggregation.
- **Service Optimization** – All else equal, high-quality services are generally preferable to low-quality ones.
- **Service Relevance** – Functionality is presented at a granularity recognized by the user as a meaningful service.
- **Service Encapsulation** – Many services are consolidated for use under the SOA. Often such services were not planned to be under SOA.
- **Service Location Transparency** – This refers to the ability of a service consumer to invoke a service regardless of its actual location in the network

Heuristics For Good Architecture And Design

A heuristic is :

A distillation of experience about a way of working that has been found to be of value.

By following good heuristics, you improve our overall architecture and design practice.

The core heuristics for good architecture and design are:

- Understand your context
- Produces satisficing (good enough) designs
- Manage dependent outcomes
- Prioritise your activities and deliverables
- Actively manage risks
- Simplify, simplify (but don't oversimplify)
- Create modules
- Build solutions from available modules
- Focus e2e solution design on the interfaces
- Use agile / iterative / evolutionary development lifecycles
- Deliver frequent deployments based on short sprints / cycles
- Develop stable intermediates
- Design for testability and test often
- Do not optimize too early

Interoperability is the ability to share information and services.

Defining the degree to which the information and services are to be shared is a very useful architectural requirement, especially in a complex organisation and/or extended enterprise.

- **Operational or Business Interoperability** defines how business processes are to be shared.
- **Information Interoperability** defines how information is to be shared
- **Technical Interoperability** defines how technical services are to be shared or at least connect to one another.

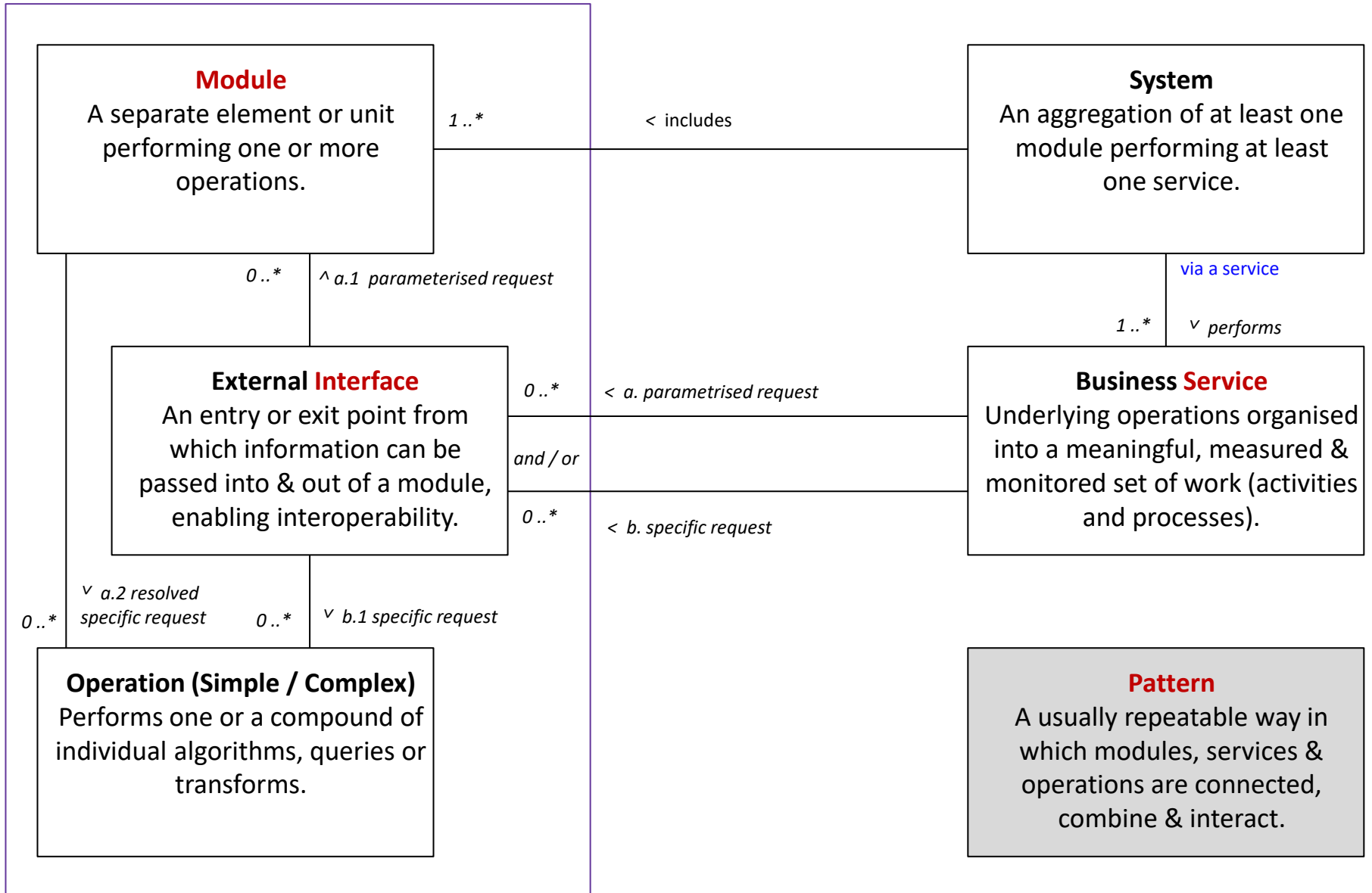
Key to establishing interoperability is the determination of the corporate operating model, where the operating model is “the necessary level of business process integration and standardisation for delivering goods and services to customers. An operating model describes how a company wants to thrive and grow. By providing a more stable and actionable view of the company than strategy, the operating model drives the design of the foundation for execution.

For example, if lines of business or business units only need to share documents, then the Architecture and Solution Building Blocks (ABBs and SBBs) may be simpler than if there is a need to share structured transaction data. Similarly, if the Architecture Vision includes a shared services environment, then it is useful to define the level the services are to be shared.

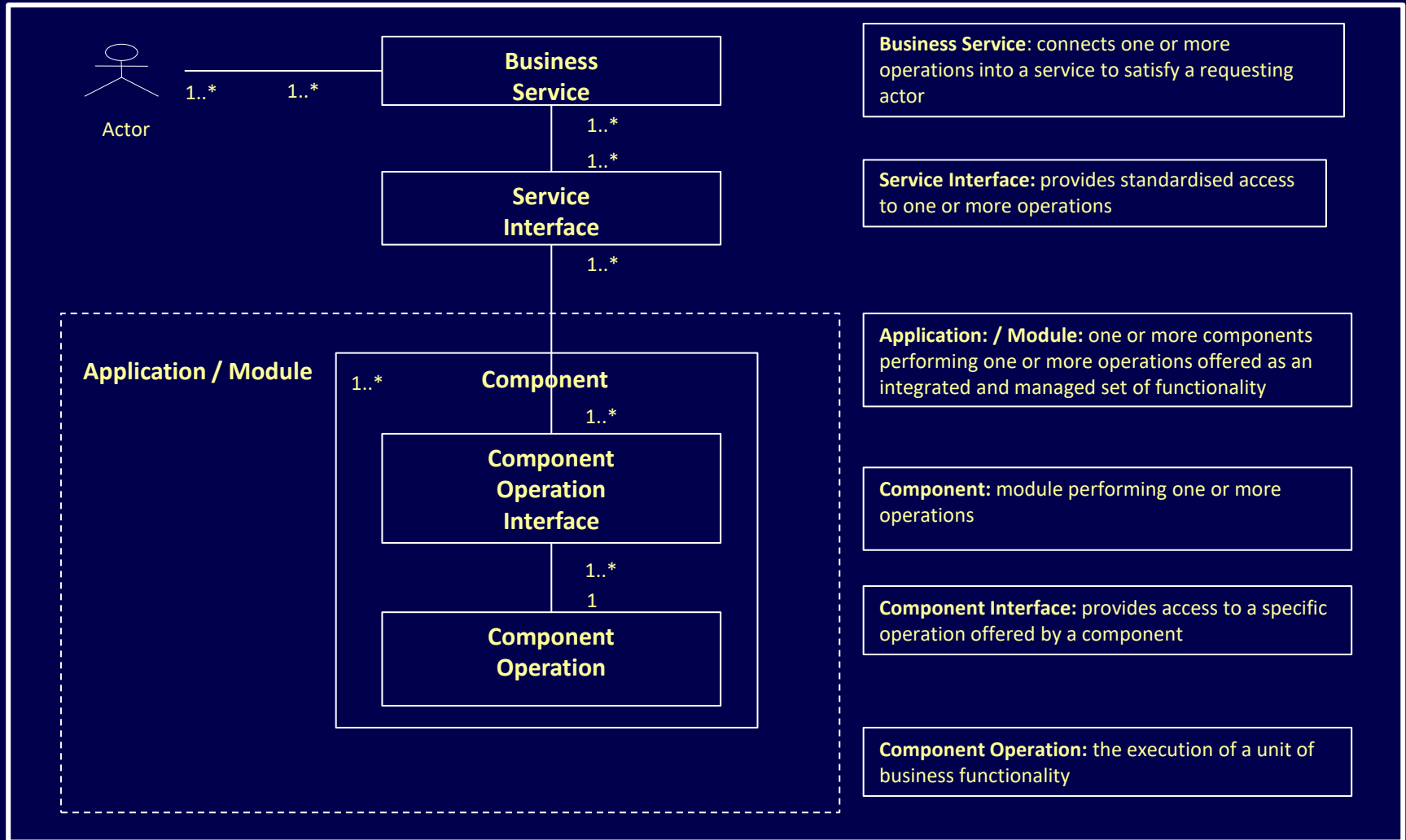
The corporate operating model will normally indicate what type of interoperability approach will be appropriate. This model should be determined in Phase A (Architecture Vision) if not in Phase B (Business Architecture), and definitely by Phase E (Opportunities & Solutions).

Complex enterprises and/or extended enterprises (e.g. across a supply chain) may have more than one type of operating model. For example, it is common for the internal operating model (and supporting interoperability model) to differ from the one used for the extended enterprise.

Systems, Modules, Interfaces, Patterns and Services



Generic Business Service – Application Pattern

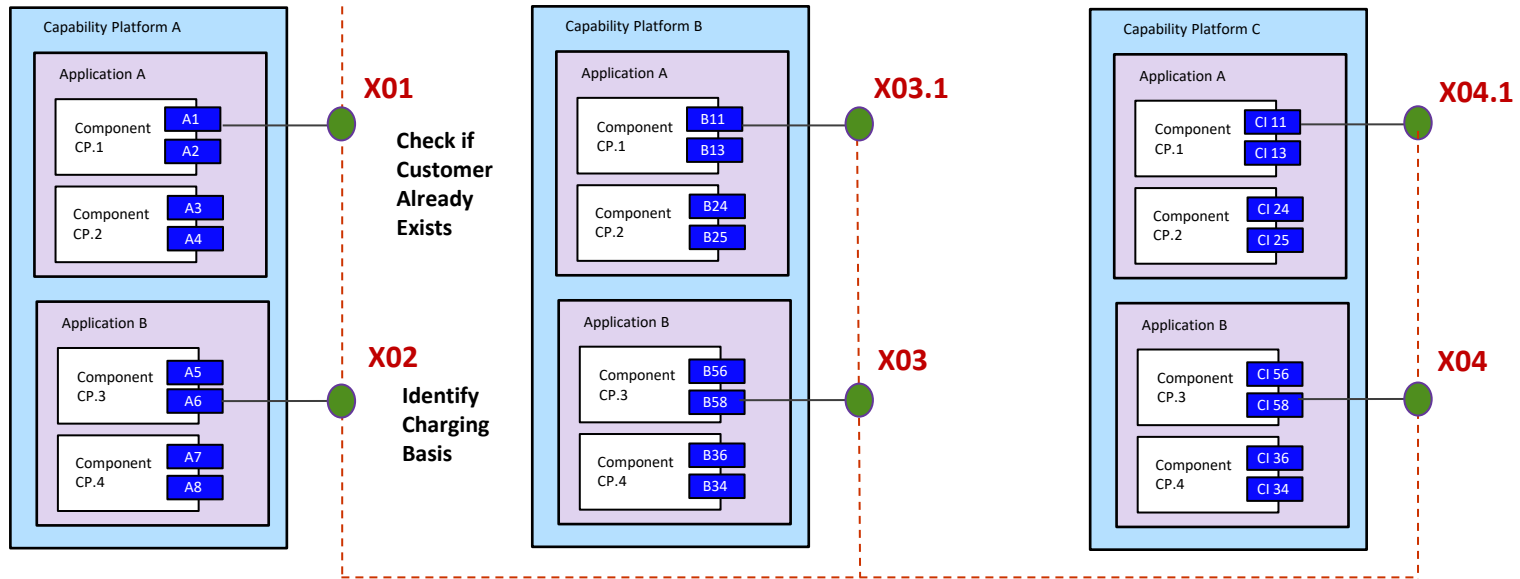


Business Service (with interfaces & operations)

Segment / Domain Platform
Commercial Account Services



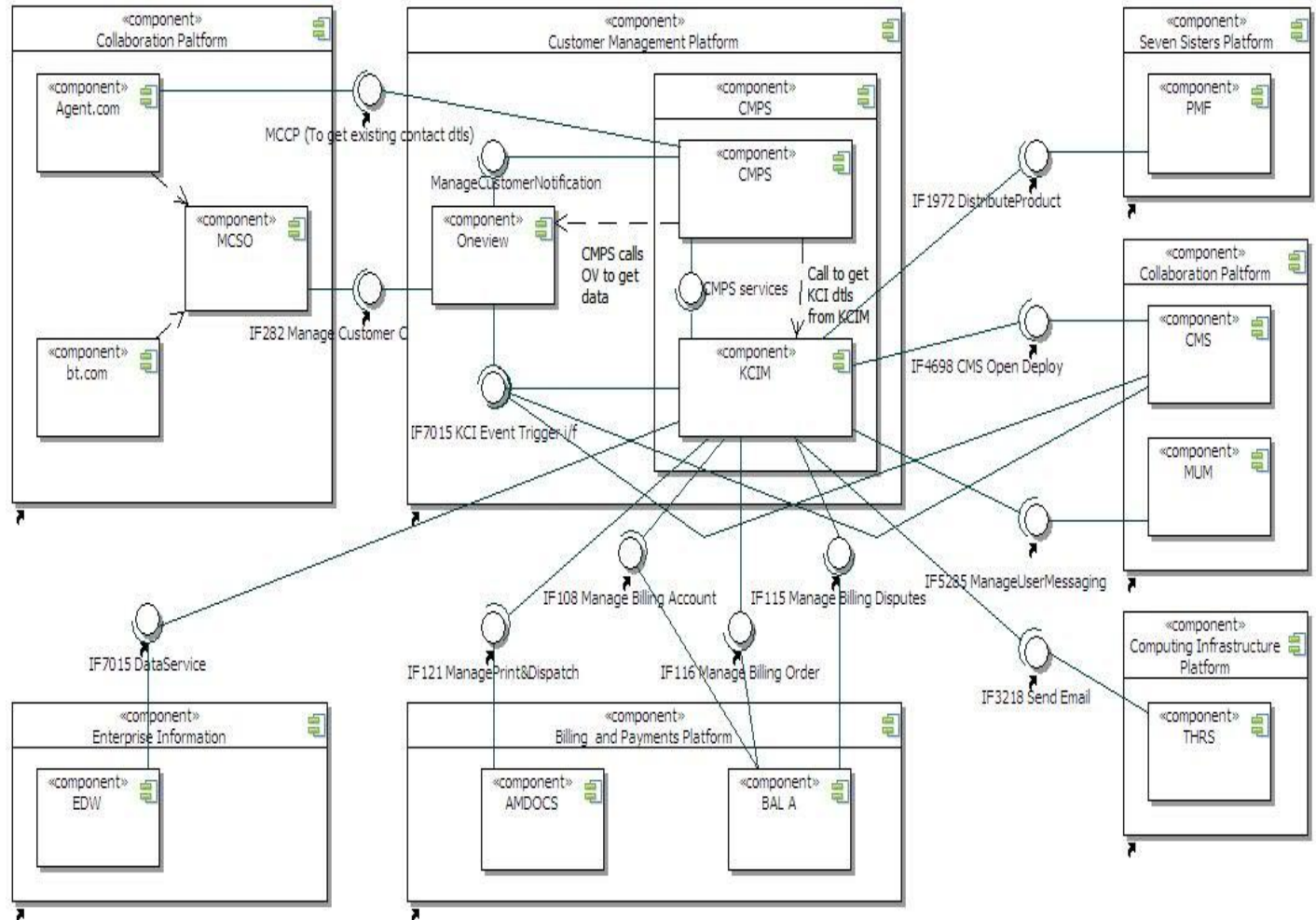
Business Service
Create New Account



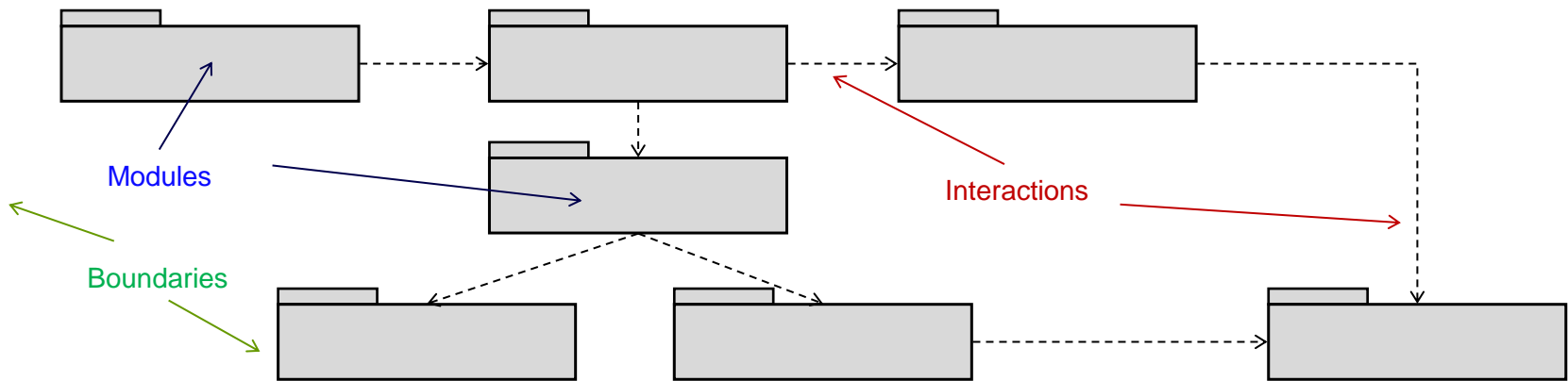
Example Real World Business Service & Application Platform View

An Enterprise Level Platform Architecture Example.

Application Platform And Business Service Interaction Diagram



Modularity and Interactions



Enterprise architecture provides the blueprint, plan or model and the management practices to more actively and effectively manage the complexities within an organisation by facilitating the:

- Definition of the resources within a business
- Definition of systems, modules and patterns
- Definition of the boundaries and relationships between these resources
- Negotiation and management across these boundaries
- Management of the shared processes, information, objects and systems across these boundaries
- Establishment of trust across these boundaries
- Establishment of responsibilities for resources within the business
- Governance for those resources from the perspective of the complete business ecosystem.

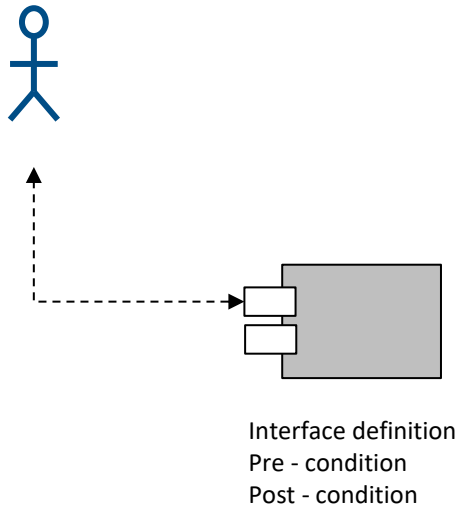
The term design pattern is often used to refer to any pattern which addresses issues of software architecture, design, or programming implementation.

We often also need to work with higher level architecture patterns.

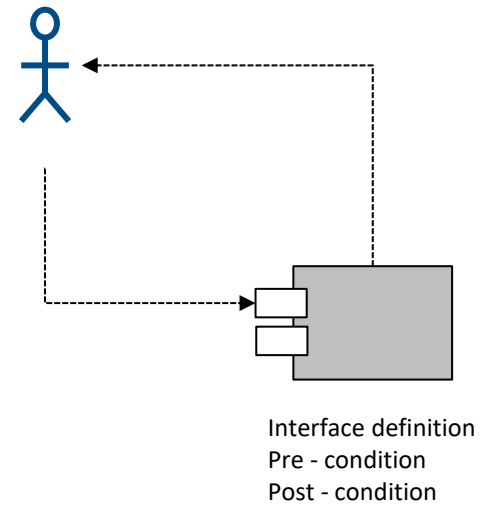
- An **Architecture Pattern** expresses a fundamental structural organization or schema for solutions (systems and services of all types). It provides a set of predefined structures and interactions, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them.
- A **Design Pattern** provides a scheme for refining the subsystems or components of a specific type of system component such as a software system, or the relationships between them. It describes a commonly recurring structure of communicating components that solves a general design problem within a particular context.
- An **Idiom** is a low-level pattern specific to an implementation context (such as a programming language). An idiom describes how to implement particular aspects of components or the relationships between them using the features of the given implementation description / language.

Component Interaction Styles

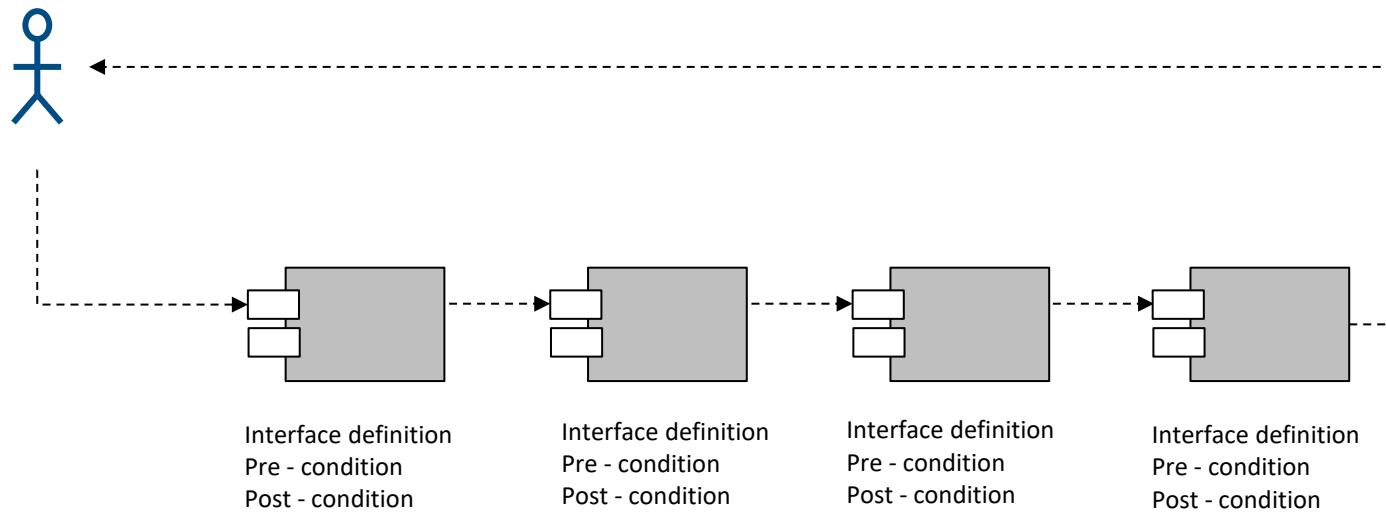
Synchronous / Blocking (Wait)



Asynchronous (Fire & Forget)

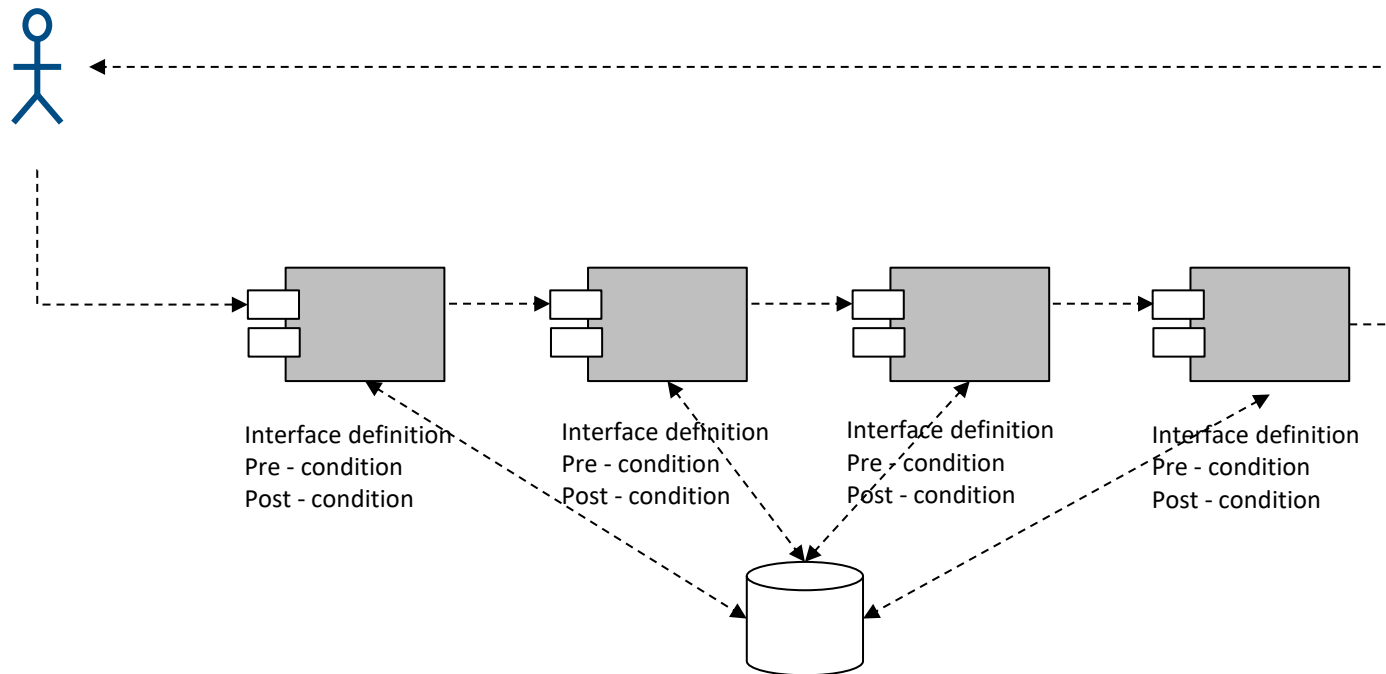


Serial / Stateless (Pass By Value)

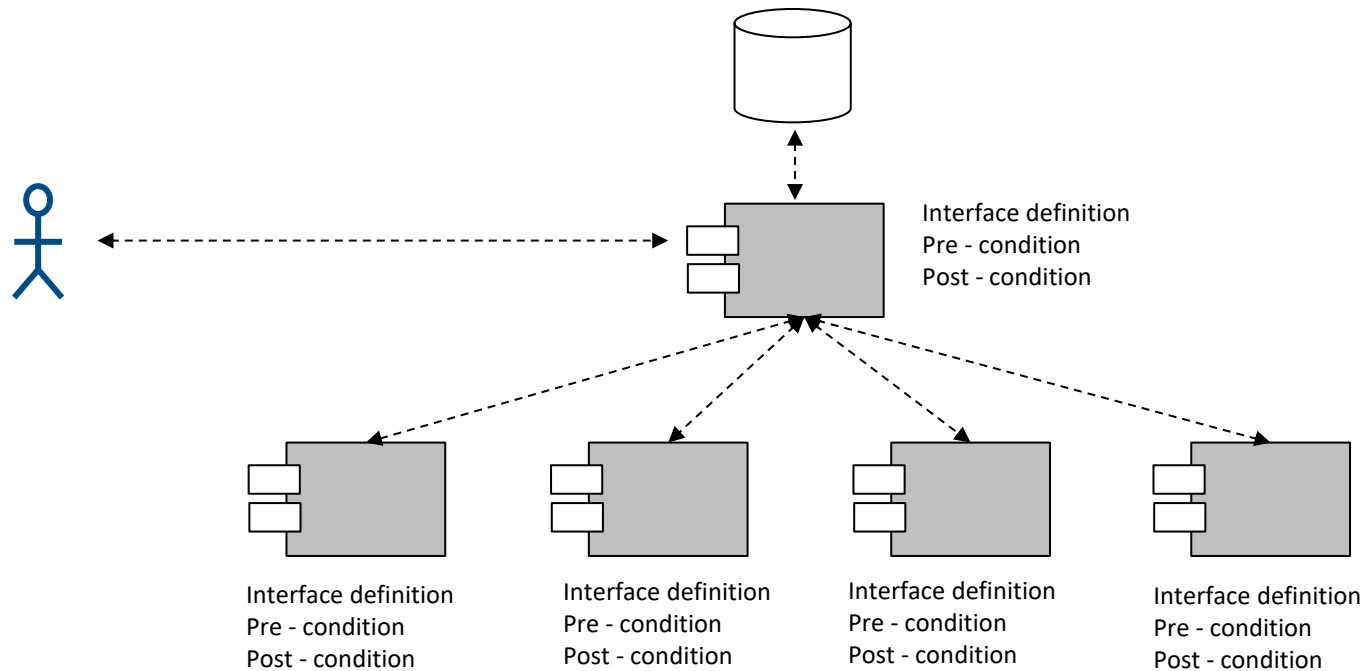


Component Interaction Styles

Serial / Stateful (Pass By Reference)



Orchestration / Controller



Pessimistic and Optimistic Patterns Of Interoperability

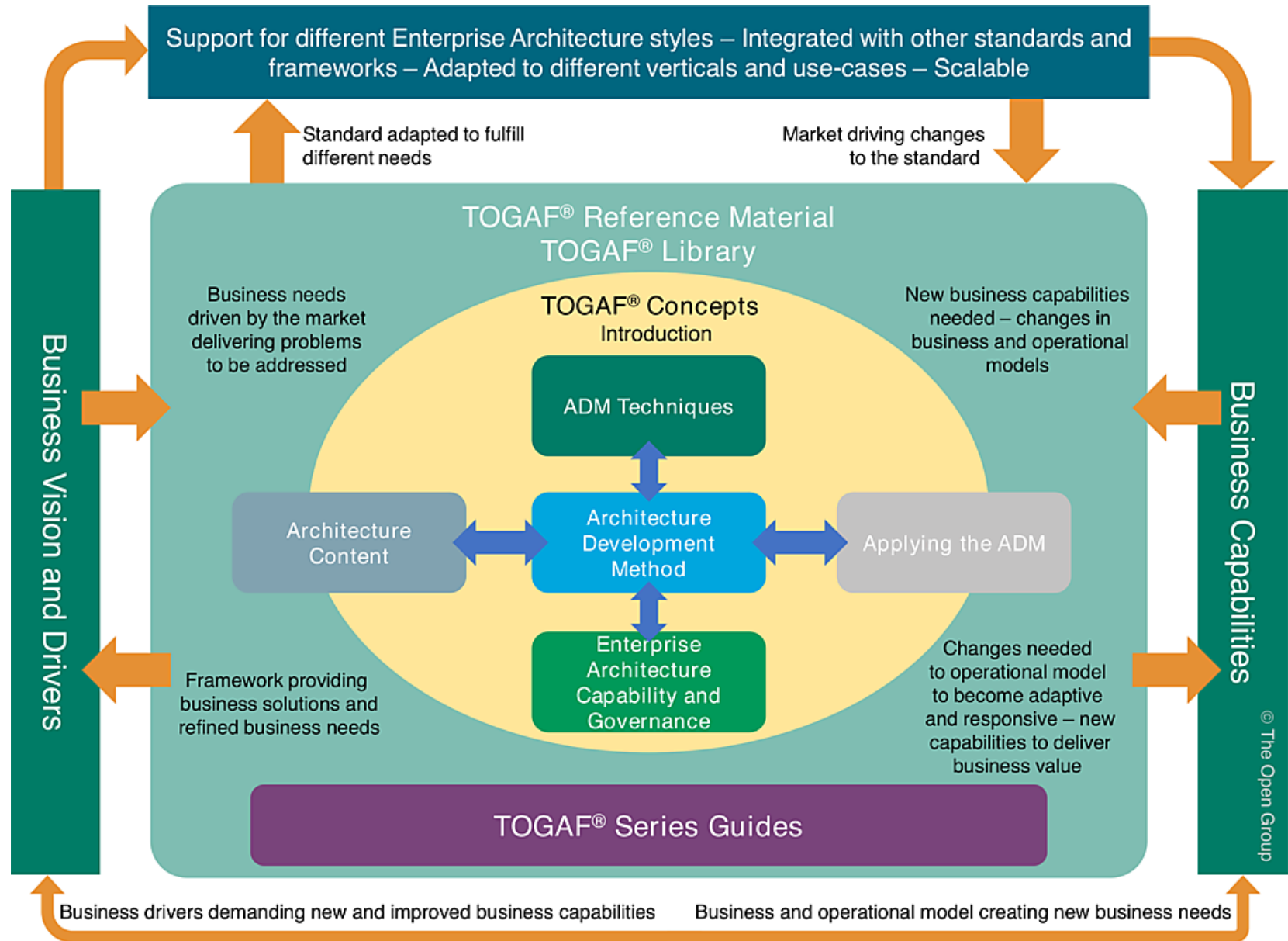
ACID (**A**tomic / **C**onsistent / **I**solated / **D**urable)

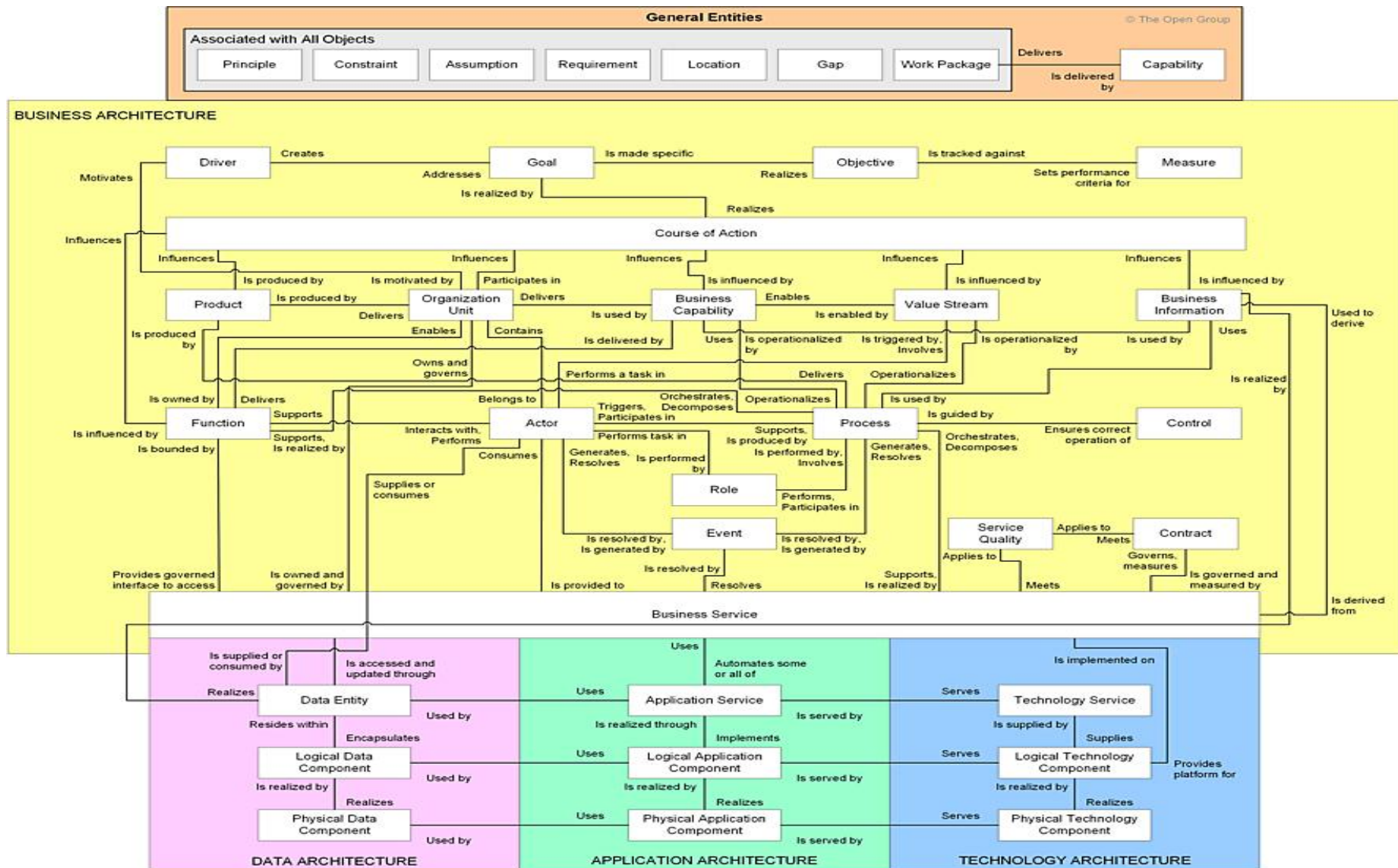
- Control / predictability / consistency
- Consistency is created beforehand ensuring the operations follow the rules.
- Planning across resource managers and explicit management of dependencies.

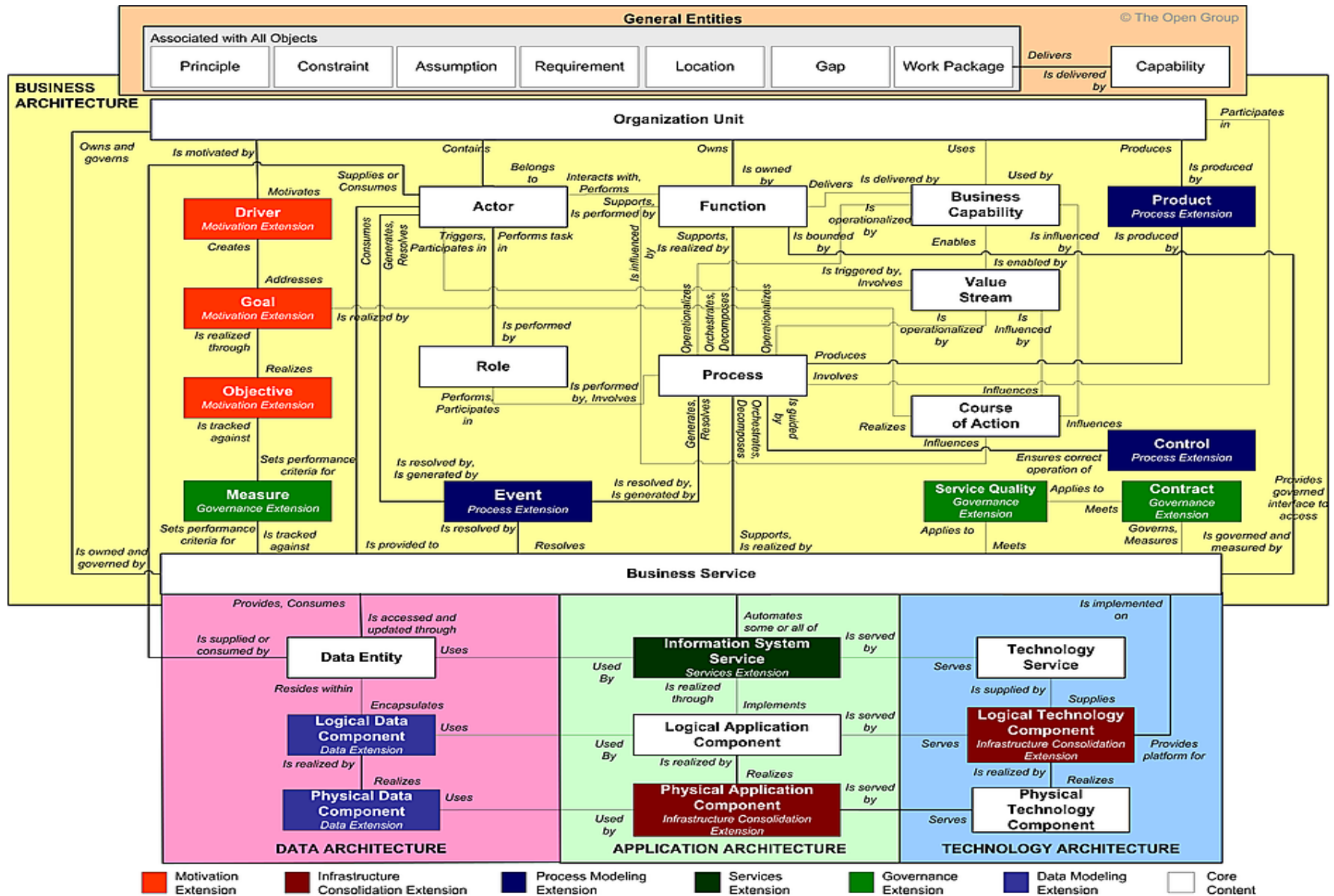
BASE (**B**asic **A**vailability / **S**oft State / **E**ventual Consistency)

- Flexibility / simplicity / speed
- Consistency created afterwards as a result of the best efforts if any compensation is required (which may not fully address any problems).
- Real time discovery of dependencies and limited planning across resources managers.

Tailoring The TOGAF Content Metamodel For Each Organisation





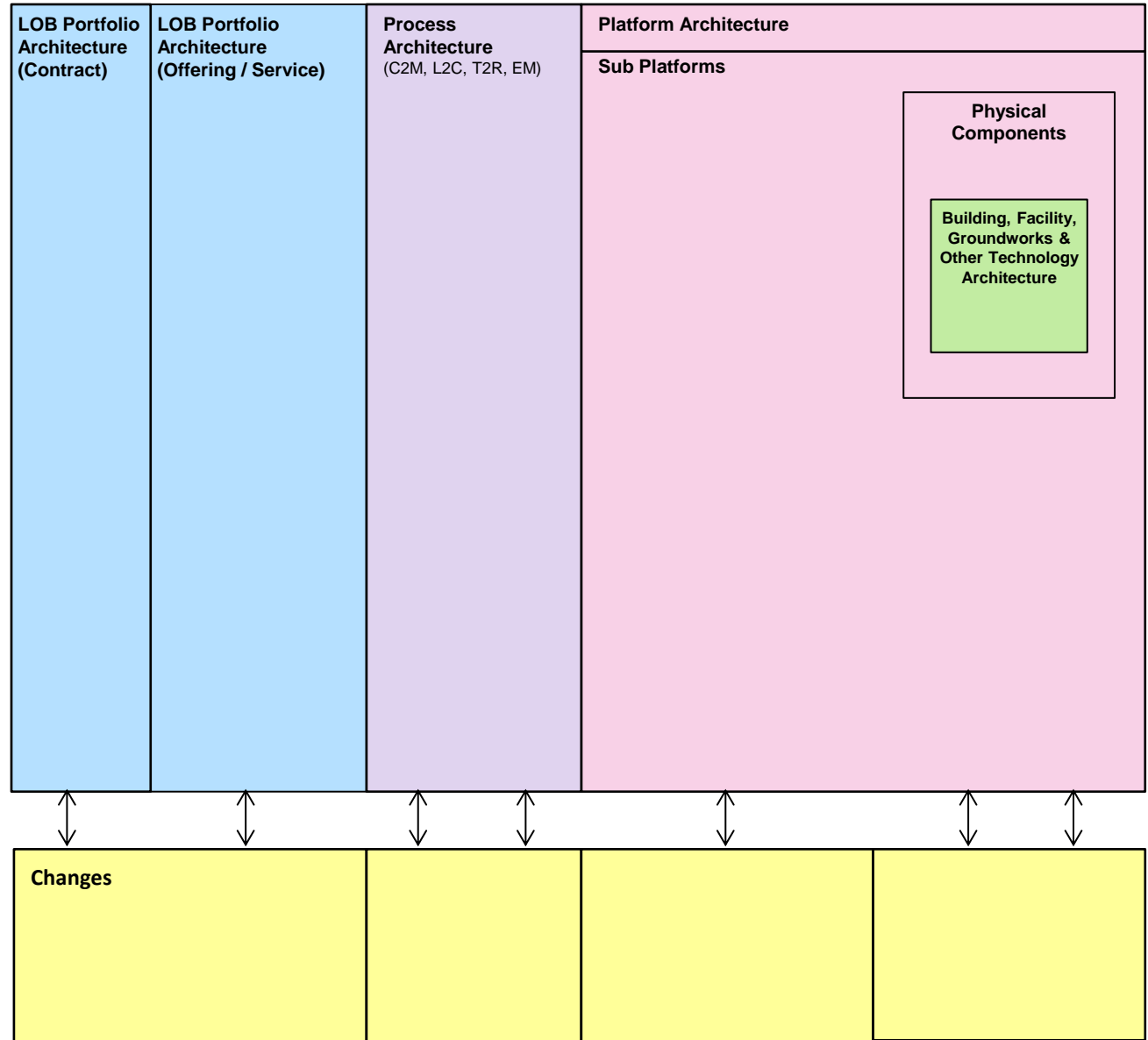


Using the architecture views as a frame

The architecture views can be presented as a frame onto which we place the solution building blocks.

This view enables us to see the connections between the solution building blocks across all of the standard framework views.

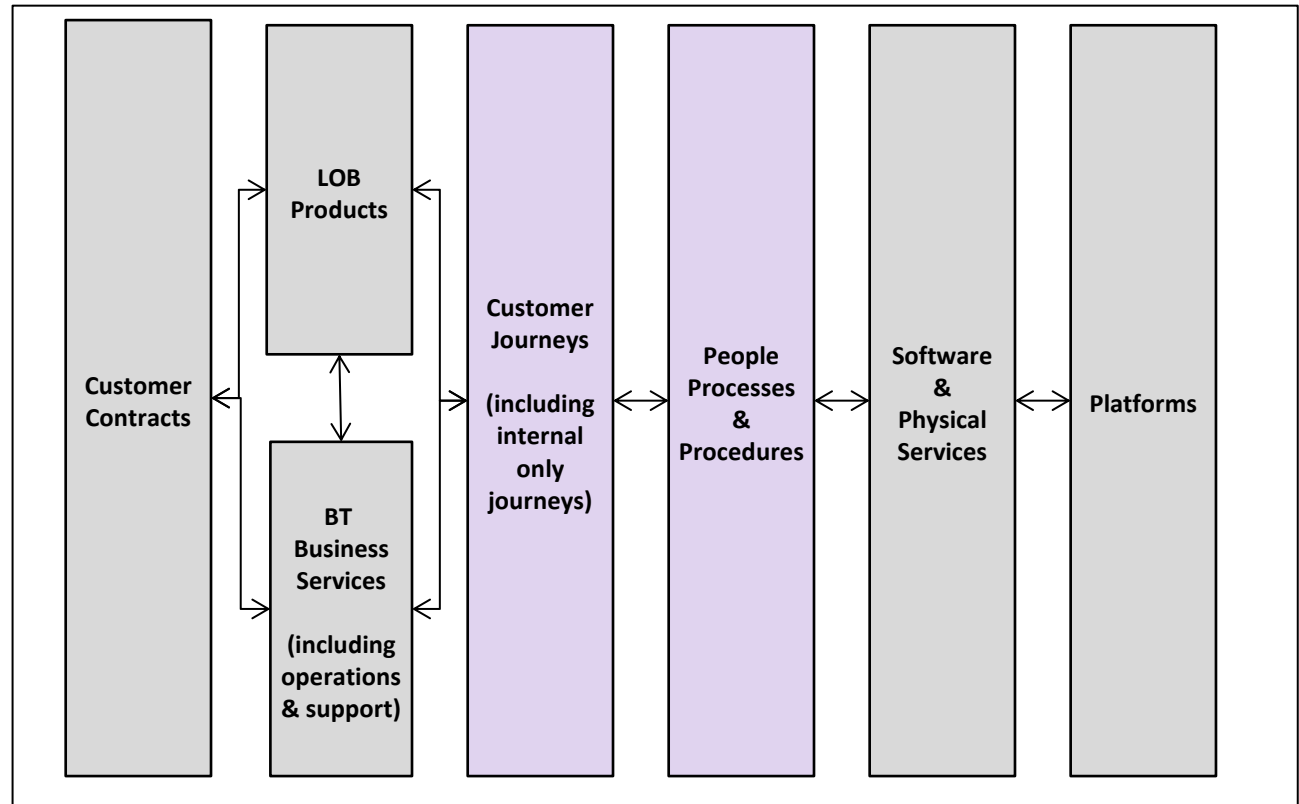
We can also relate the specific solution building block types to requests for change in the form of user story types.



The top level solution building blocks

We group the solution building blocks into a set of major elements that reflect the “assemblies” that we use within the business. These cover:

- Customer contracts
- LOB propositions & products
- Business services
- Customer journeys
- Processes & procedures
- Software & Physical services organised into Platforms.



These represent the first level of reuse from which we can select elements to create new solutions.

The more that we can find existing solution capabilities in the left hand side of the diagram (i.e. existing contracts or already defined products) the greater the reuse; as they already encompass the more detailed solution building blocks.

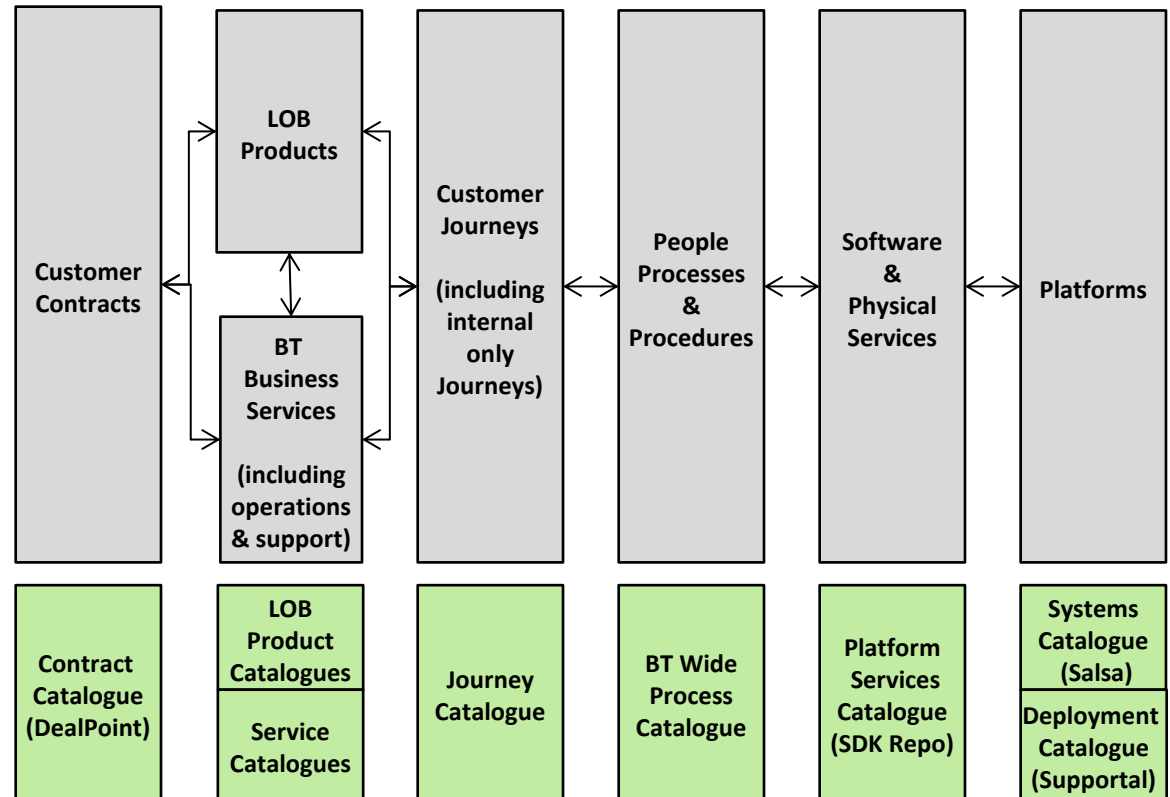
If we have to start looking for more granular capabilities at the right hand side of the diagram (i.e. individual software services or physical devices) the amount of change, integration and testing that will be needed to deliver new business services, products or contracts.

Finding and reusing components for delta designs

As we define the high level design we should initially consider using components from the existing solution building blocks (shown in grey).

As the solution stories are decomposed we develop the design by selecting the components with the highest level of composition first (i.e. reusing an existing customer contract or LOB product automatically reuses the more detailed components to its right).

We look for the existing components in the various catalogues (shown in green) and only consider creating new components if we can't find existing ones that can be reused.



As the high level solution design is developed and building blocks are selected from each level; a set of interaction diagrams showing how they work together should be developed, detailing the:

- The impacted journeys and their processes and people.
- Platform service changes; identifying the components involved, the cross platform flows and responsibilities.

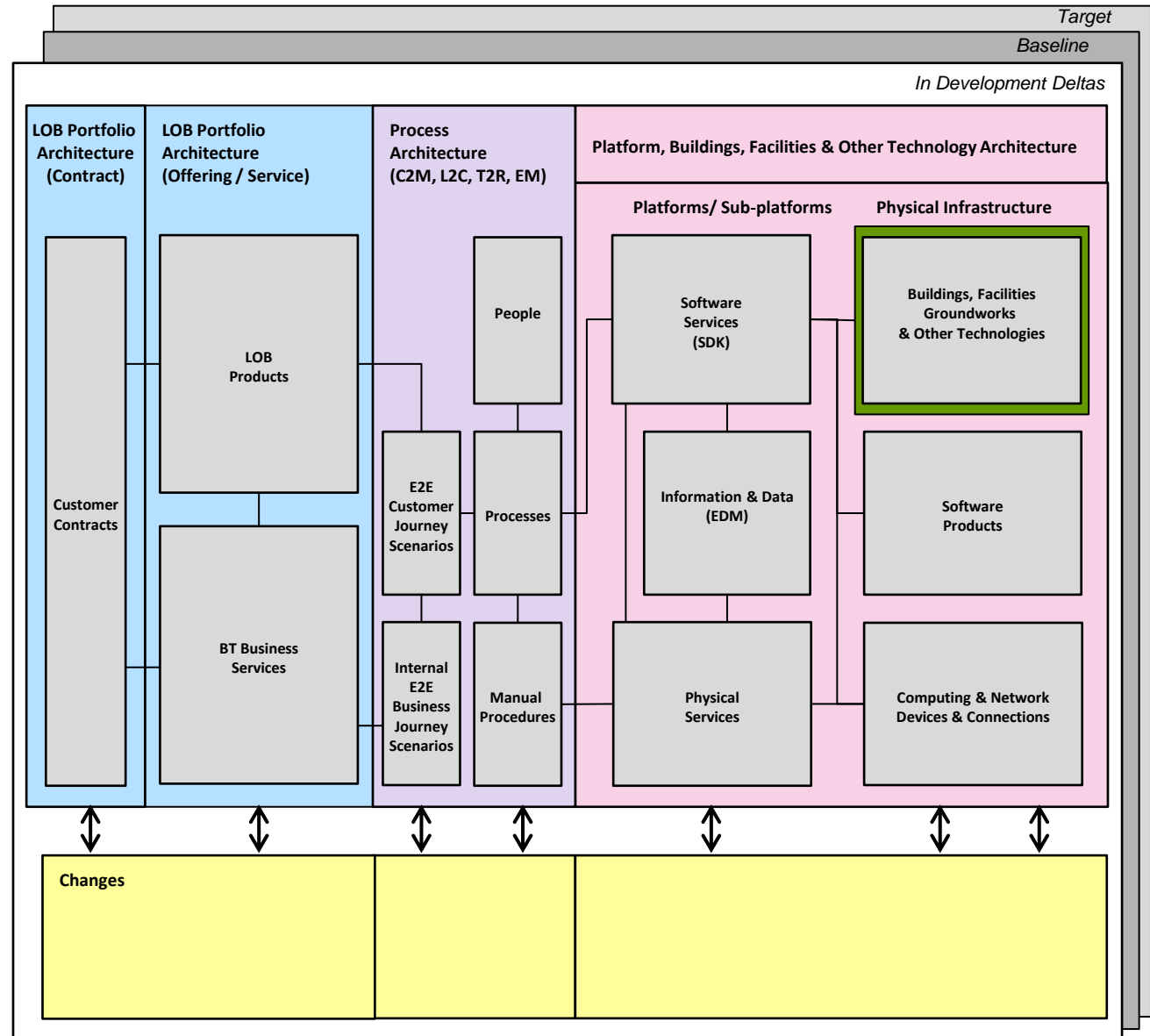
Placing the solution building blocks into the architecture views

We then place the top level solution building block onto the architecture views.

This creates an integrated view of the BT architecture enabling us to connect various components together and understand how they interact.

You can find out about the solution building blocks by accessing the underlying catalogues accessible from the DesignPoint reference library.

Please browse all of the catalogues in the reference library to find out what they cover and which solution building blocks are available.



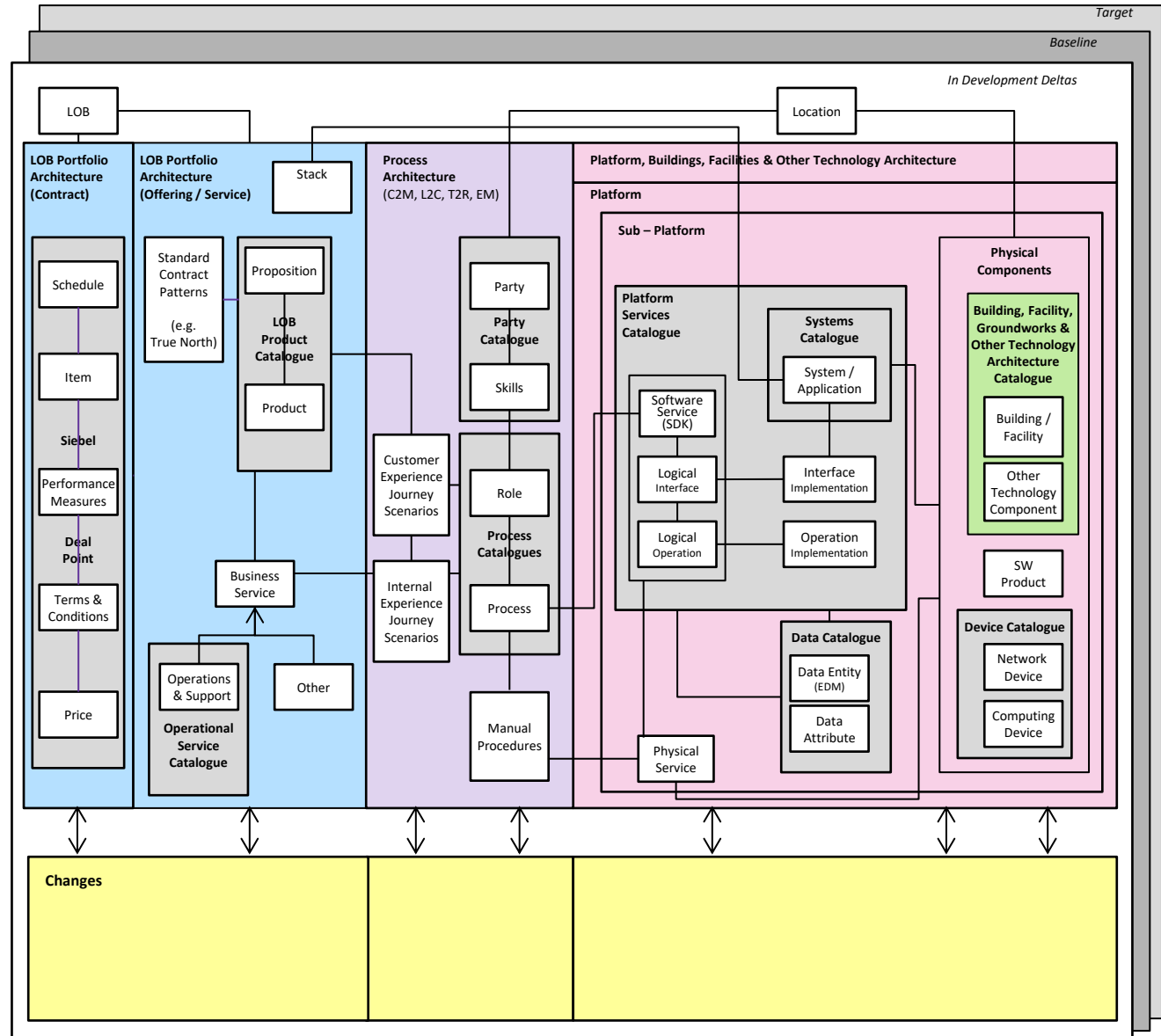
Expanding the solution building blocks

We add the next layer to the solution building blocks by exposing the level at which we tend to manipulate them on a day to day basis.

It is at this level that we create the architecture/high level design of a new solution.

It provides enough detail to understand:

- Which components are used within an e2e solution
- How they interact
- What the expected performance properties (non functionals) are.
- What outcomes they will deliver for our customers

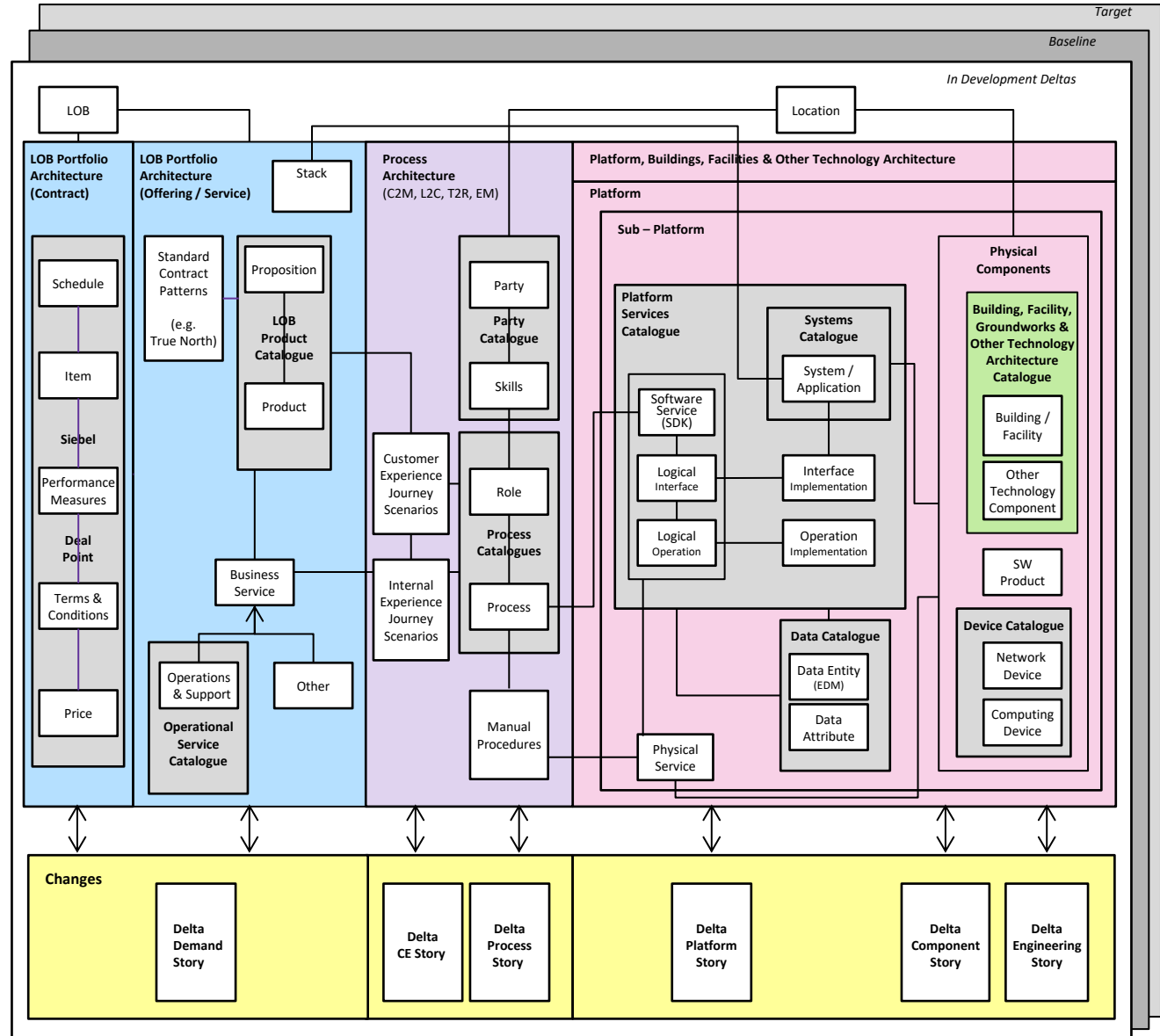


Aligning changes to the solution building blocks with the architecture

To manage change to components within the architecture we use user stories.

For each required change a delta change story of the relevant type should be produced.

The various user story types are developed as they flow through the change process describing the outcomes and acceptance criteria for the different types of solution building block / component that need to be created, changed or integrated.



DesignPoint content sections – A & D Reference Library example 1

The Reference Library contains links to many of the architecture and design repositories collectively described as the BT Business Elements catalogue. This is a key section of DesignPoint and navigation to the various elements in the catalogue is via either the clickable picture or

.....

The screenshot shows the BT DesignPoint Reference Library page. The browser address bar displays <https://office.bt.com/sites/CAO/DesignPoint/RefLibrary/SitePages/Home.aspx>. The page title is "Reference Library - Home". The main content area is titled "Reference Library" and includes a description: "The Reference Library provides guidelines, templates, patterns, and other forms of reference material that can be leveraged in order to architectures for the enterprise. (taken from TOGAF 9)." Below this is the "BT Business Elements Catalogue" section, which states: "The BT Business Elements Catalogue illustrates the different components a designer needs to consider in the design process and will where they fit in the overall design process. There are two views of the BT Business Elements Catalogue a high level view and a more detailed view." Two links are provided: "BT Business Elements - level 1 elements.pptx" and "BT Business Elements - level 2 elements.pptx". A text box on the right side of the page states: "The reference library page includes a description and then a graphic, following by a series of links. The graphic provides a visual representation of the key elements in the list. The repositories in the reference library are accessible by clicking on the graphic. Two examples are highlighted." The graphic at the bottom of the page is a complex diagram showing various architecture and design repositories. It is divided into several sections: "LOB Portfolio Architecture (Contract)", "LOB Portfolio Architecture (Offering / Service)", "Process Architecture (C2M, J2C, T2R, EM)", "Platform, Buildings, Facilities & Other Technology Architecture", "Matrix Architecture", and "Physical Infrastructure". Two specific elements are highlighted with red circles: "OB Products" in the "LOB Portfolio Architecture (Offering / Service)" section and "Software Services (SDK)" in the "Matrix Architecture" section. The diagram also includes other elements like "Customer Contracts", "BT Business Services", "People", "Processes", "Information and Data (EDM)", "Physical Services", "Buildings, Facilities Ground works & Other Technologies", "Software Products", and "Computing & Network Devices & Connections".

Simplified Elements Of An Enterprise Architecture

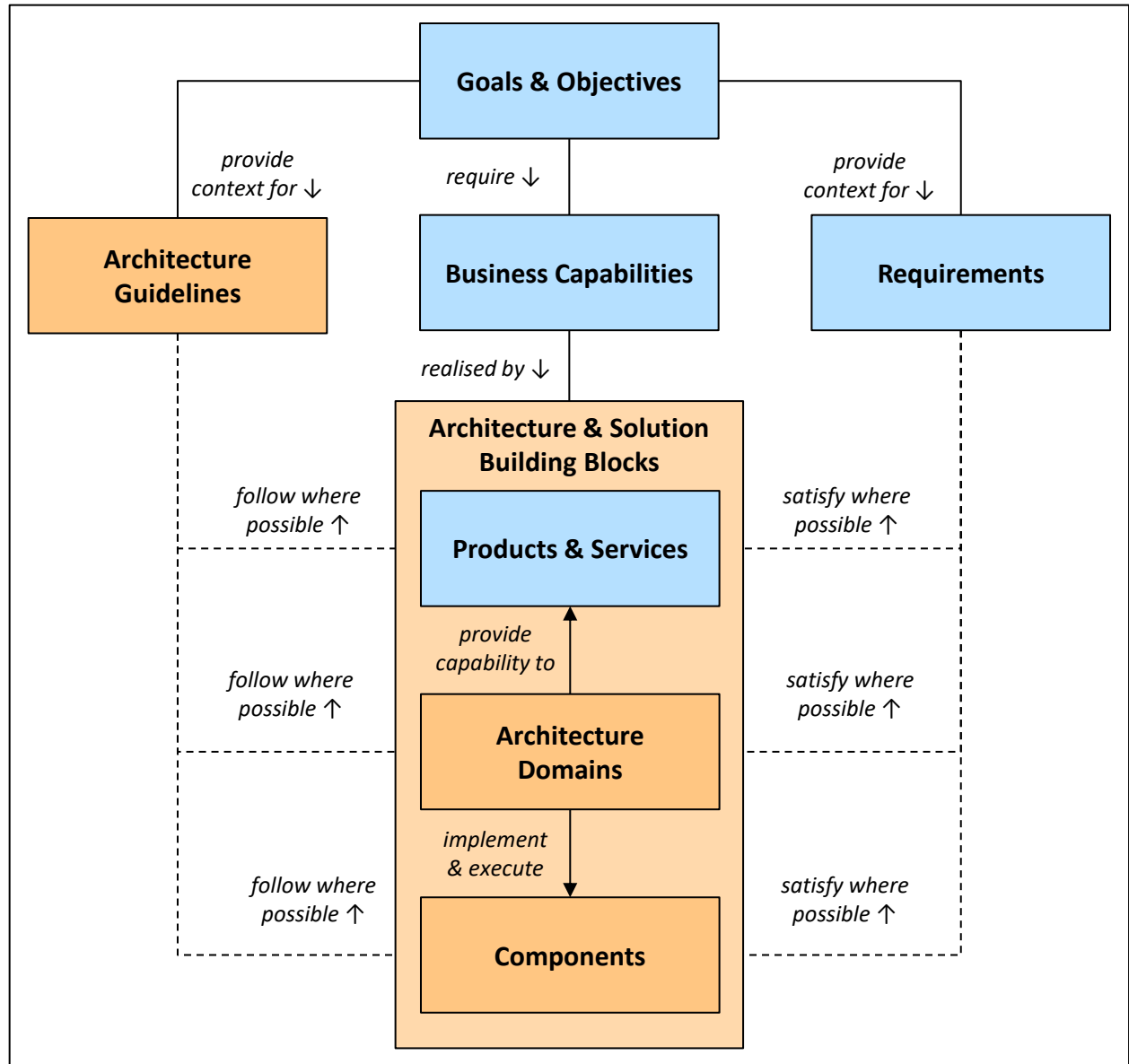
Our enterprise architecture connects our business goals, objectives and requirements to our information system implementations.

The architecture guidelines (the principles, policies and standards, roadmaps, reference models and patterns) provide the context and controls for how we develop and govern existing and new solution components and integrate e2e products and services for our markets.

The building blocks (representing the actual business and technology components) provide the “raw material” we use in our current solutions and that we can reuse or change to create new solutions.

The building blocks are grouped into

- Specifications of what is needed:
Architecture building blocks
- Implementations of what is needed:
Solution building blocks

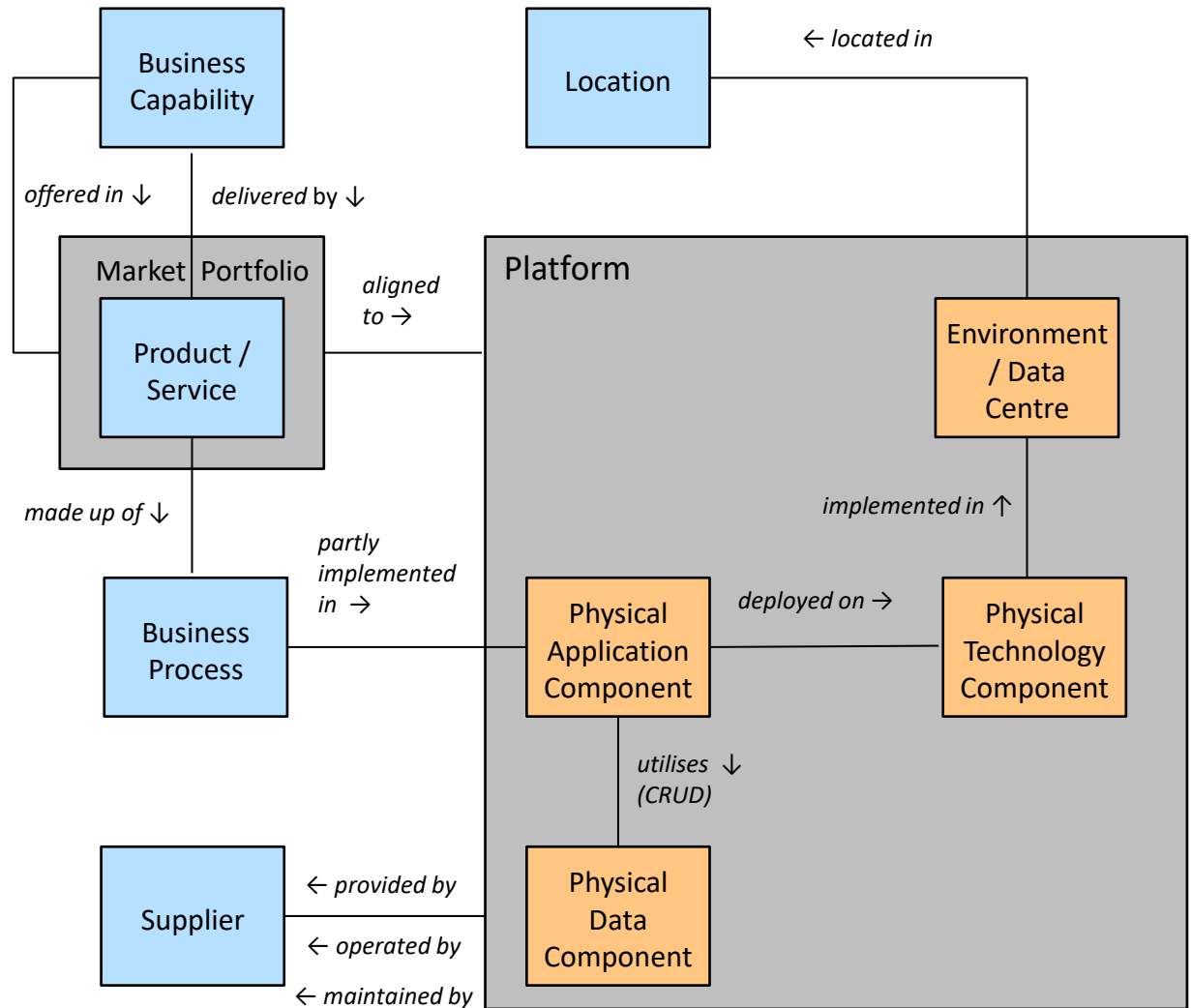


Simplified Enterprise Architecture Viewpoint

We have identified these elements as the most relevant to our business.

The generic enterprise architecture viewpoint identifies the main business and information systems elements for the markets and locations in which we operate.

This architecture viewpoint supports the development of separate and shared business and information systems components and their efficient and effective use, as determined by the relevant business goals, objectives and associated requirements for us and each market.



The Potential Catalogue Set (2)

For example: when creating a new application component a record of the component should be placed in the application component catalogue and references to that component created (if relevant) against the:

- APIs it uses,
- application services it is part of (if relevant),
- business capability it implements,
- data centre / environment it is deployed in
- market it addresses
- physical data components it uses,
- platforms it is placed in,
- processes it is part of,
- products and/or services it supports,
- suppliers who created it,
- suppliers who operate it,
- suppliers who maintain it,
- technology components it is deployed on, and the
- technology services it is part of.

		API	Application Component	Application Service	Business Capability	Data Centre / Environment	Location	Logical Data Component	Market	Physical Data Component	Platform	Process	Product	Service	Supplier	Technology Component	Technology Service	No Of Matrices For Row
Application	API		X															1
Application	Application Component	X		X	X	X			X	X	X	X	X	X	X	X	X	13
Application	Application Service		X		X	X			X	X	X	X	X	X	X	X	X	12
Business	Business Capability		X	X					X				X	X				5
Technology	Data Centre / Environment		X	X			X								X	X	X	6
Business	Location					X												1
Data	Logical Data Component									X								1
Business	Market		X	X	X						X		X	X				6
Data	Physical Data Component		X	X				X			X				X			5
Technology	Platform		X	X					X	X			X	X	X	X	X	9
Business	Process		X	X									X	X				4
Business	Product		X	X	X				X		X	X		X				7
Business	Service		X	X	X				X		X	X	X					7
Business	Supplier		X	X		X				X	X					X	X	7
Technology	Technology Component		X	X		X					X				X		X	6
Technology	Technology Service		X	X		X					X				X	X		6
	No Of Matrices For Column	1	13	12	5	6	1	1	6	5	9	4	7	7	7	6	6	96

(N.B. the total counts entries twice (horizontal and vertical) so the number of relations is 48)



How Much Work Is This And What Is The Value?

This seems at first sight to be a lot of work but when broken down as shown here, it can be seen to be both much simpler and of significant value.

Each domain architect is responsible for the relevant entity catalogues. The number of entities for each domain is relatively few and many of them (particularly in the business and technology domains do not change very often).

Part of effective solution design, quality assurance and repair requires that we understand, what we have, the impact across any other elements and what has actually been created.

Additionally, knowledge of what we already have and how it is deployed within our environments makes it much easier and faster to reuse components and their capabilities; avoiding time and money on new solutions that may not be needed and incur the greater risk of new deployments.

Business Domain Catalogues

Business Capability
Location
Market
Product
Business Service
Process
Supplier

Application Domain Catalogues

API
Application Component
Application Service

Cross Reference Entries

Data Domain Catalogues

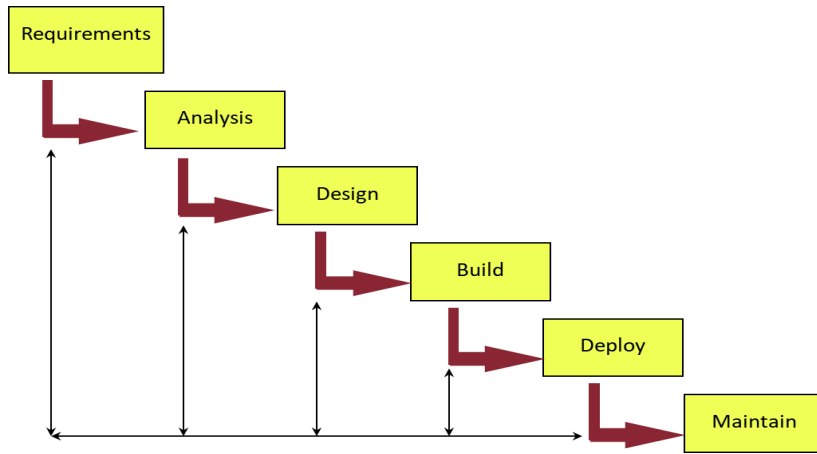
Logical Data Component
Physical Data Component

Technology Domain Catalogues

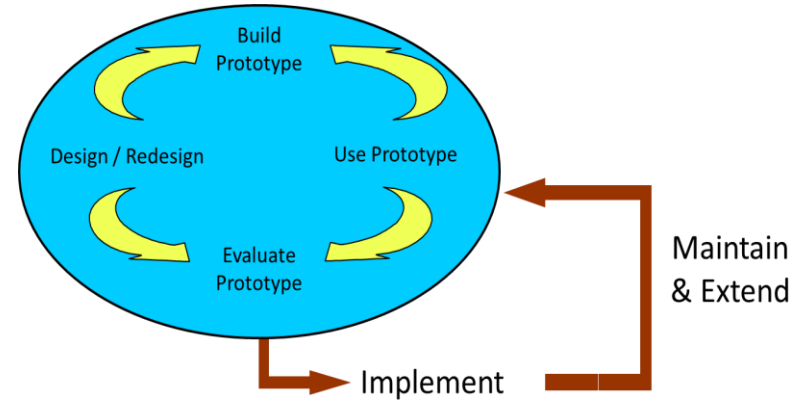
Data Centre / Environment
Platform
Technology Component
Technology Service

Development Lifecycle Shapes

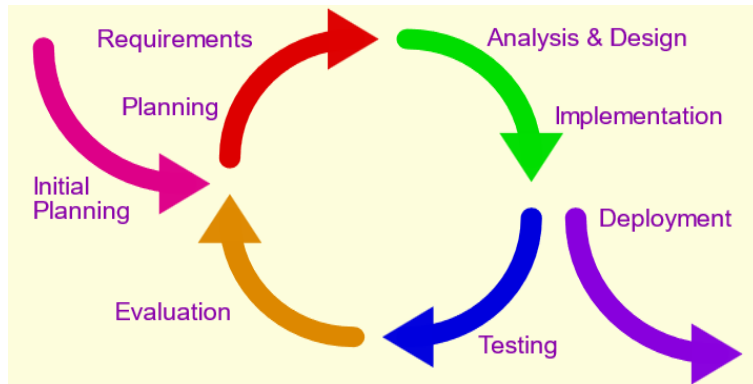
Waterfall



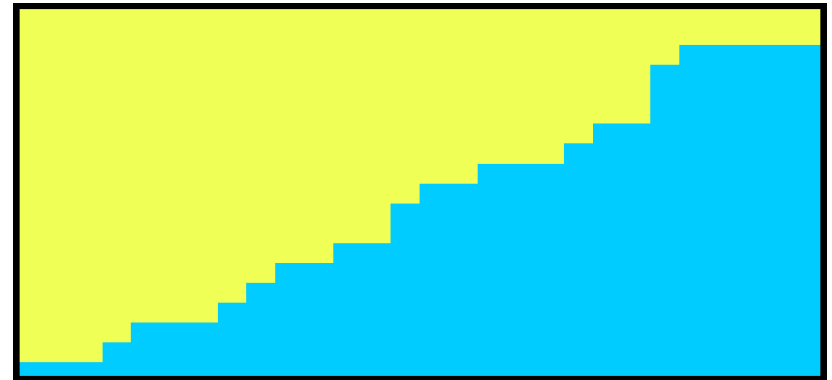
Prototyping



Iterative (Unified Process)



Evolutionary Incremental



The Agile Approach

Included In Original Inputs To The Agile Manifesto and Agile Alliance

Crystal Methods

Alistair Cockburn

Lean Development

Toyota Bob Charette

Rapid Application Development

Many

Dynamic Systems Development Method

UK DSDM Consortium

SCRUM

Schwaber & Sutherland

Adaptive Software Development

Jim Highsmith

Extreme Programming (XP)

Beck & Cunningham

Not Included In Original Inputs To The Agile Manifesto and Agile Alliance But Some Aspects Included Later

Business Rules Based Development

Rapid Evolutionary Development

Lowell Jay Arthur

Object Oriented Development

Many

Contract Based Development (Design By Contract)

Bertrand Meyer

Iterative Development

Tom Gilb

Quality Function Deployment (The Voice of The Customer)

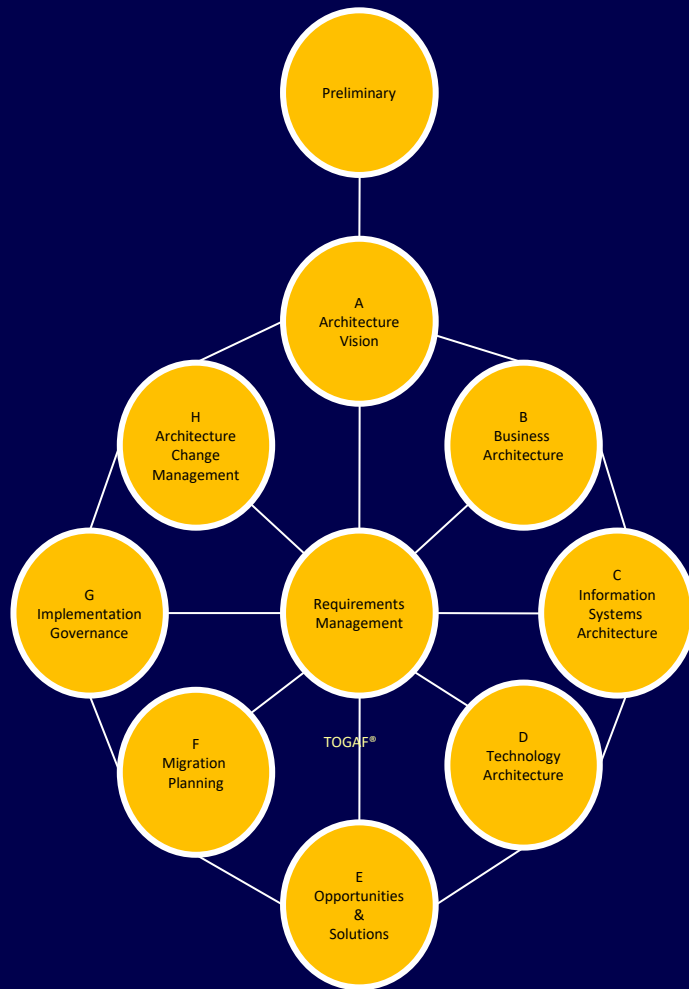
Many Japan - US

Component Based Development

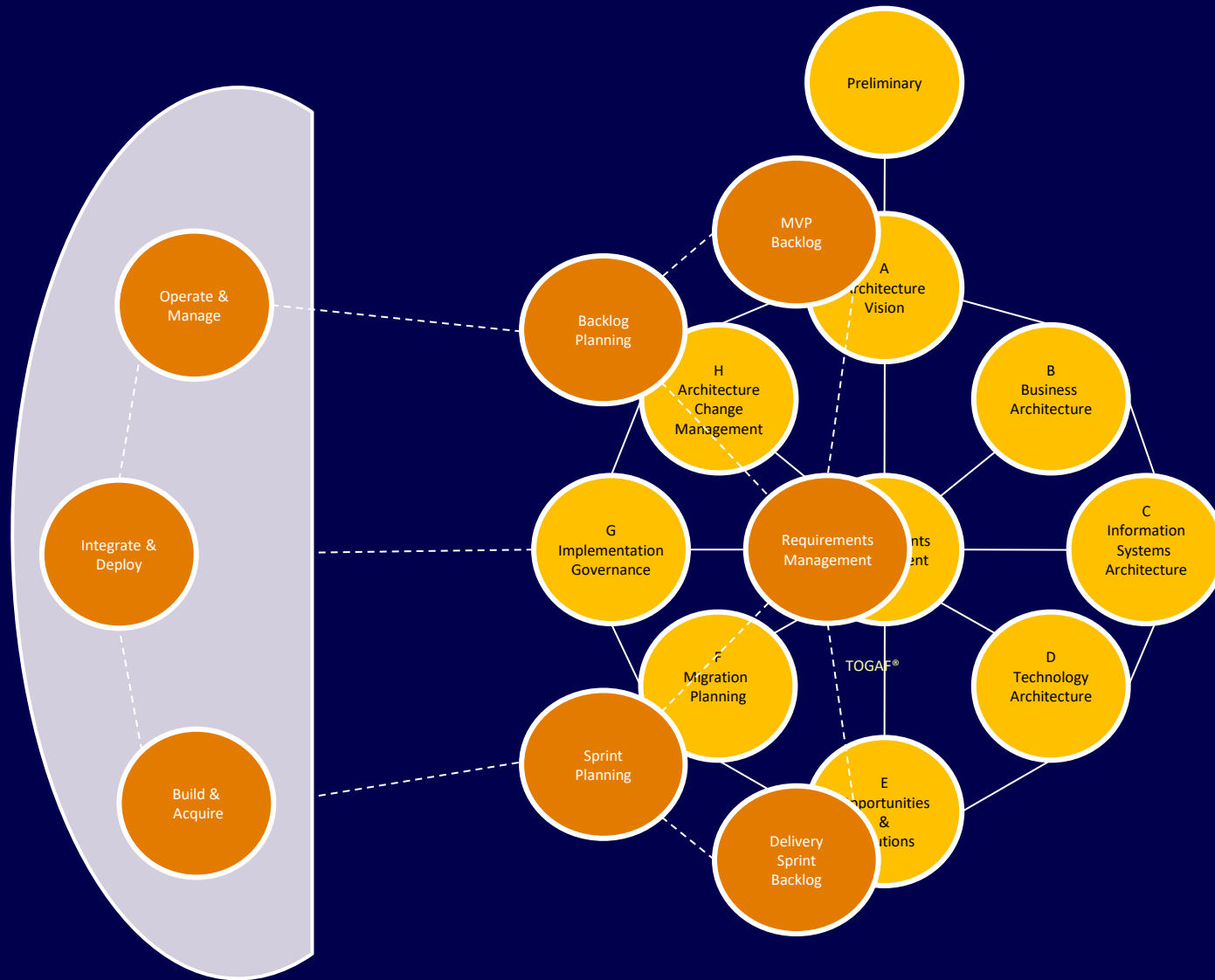
Many



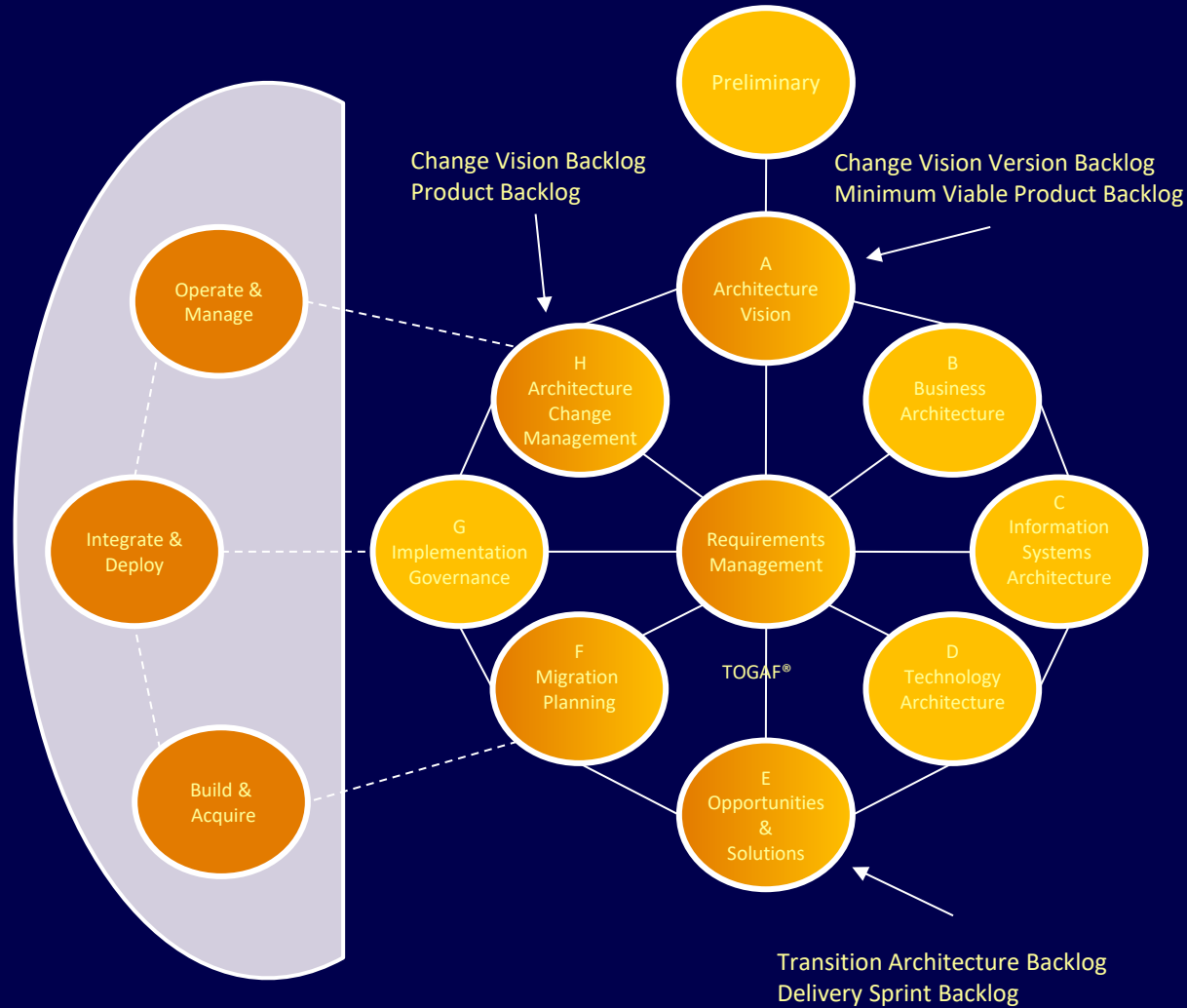
TOGAF & Agile Lifecycles



Overlaying TOGAF & Agile techniques



Blending TOGAF and Agile Techniques



Change Profiles

Much (but not all) of what we change
can be shaped by the concepts within the agile approaches.

True Enterprise Agility

Is Enabling The Optimum Change Profiles
Based On The Goals, Objectives & Requirements
Of Each Specific Change and Its Possible
Solution Options
(Not A One Size Fits All Approach)

Robust Change

e.g. Nuclear
Power Station,
Aircraft Control Systems
Core Banking Systems

Agile / Functional Change

e.g. Sales
Campaign
Async Messaging Systems
Information and Document Sharing

Rapid Change

e.g. Two Weeks
Left Before
Market Windows
Closes

Higher Integrity
Higher Risk

Lower Integrity
Lower Risk

Arbitrary Integrity
Arbitrary Risk

Fully Engineered/Iterative @60 process steps

Agile/Iterative @ 20 process steps

Fast to Market/Iterative @ 5 process steps

Fully Engineered/Iterative

@60 process steps

Aircraft Control Systems

ATM Systems

Bank Account Management Systems

Core Data Systems

Agile/Iterative

@ 20 process steps

Social Media Sites

General Information Sites

Information Aggregator Sites

Fast to Market/Iterative

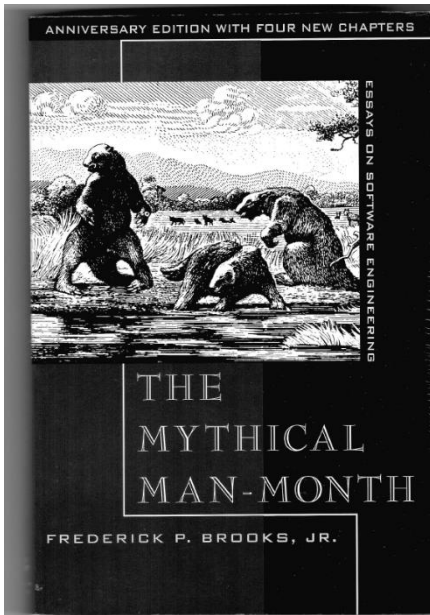
@ 5 process steps

Google Glass

Make The Time Window Non Critical Systems

Save the Business Systems

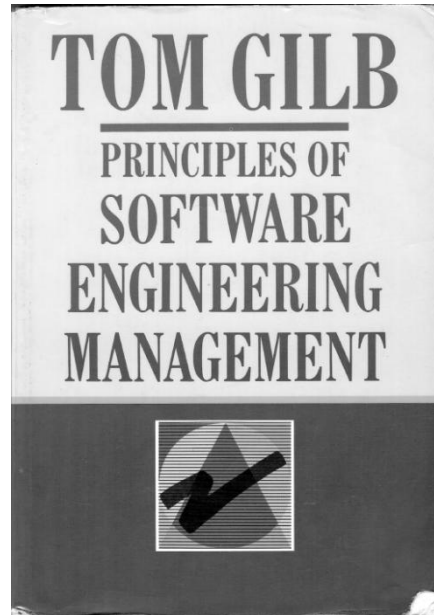
Recommended Books For Understanding The Developing Ideas



Fred Brooks 1975/95

Observation on creating large scale systems from architect of the IBM mainframe operating systems.

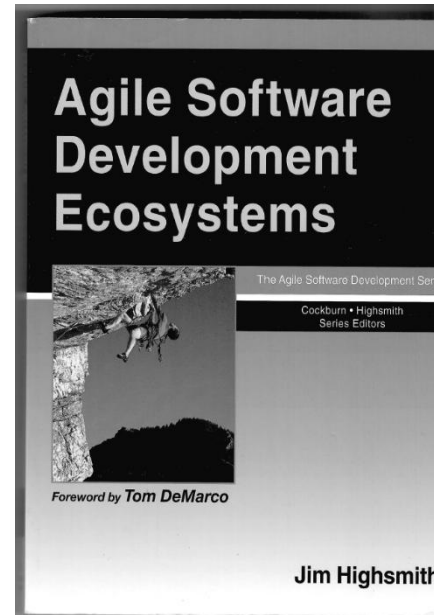
One of the first classics of software development.



Tom Gilb 1988

The classic text describing the flood of iterative and incremental customer focused requirements responsive approaches. Includes:

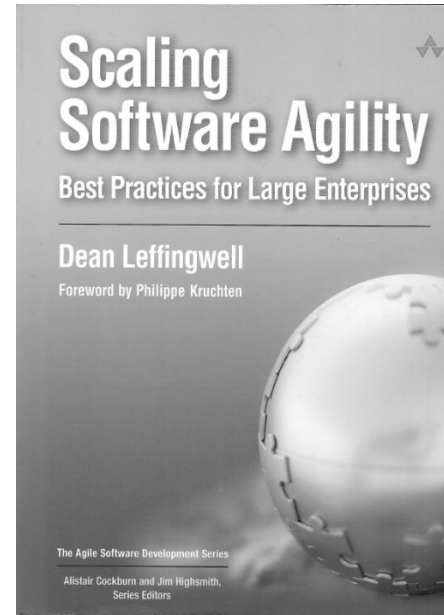
- The Principle of Fuzzy Numbers
- The Norwegian Mountain Survival Principle
- The Juicy Bits First Principle
- The Mountain Goat Principle
- The "How Little ?" Principle



Jim Highsmith 2002

A record of the Snowbird, Utah meeting that set up the Agile Manifesto.

Describes many of the 1980 and 1990s techniques that we aggregated into the Agile approaches.



Dean Leffingwell 2007

Described the issues associated with practicing agile at scale.

Is the basis for what subsequently emerged as SAFe®.

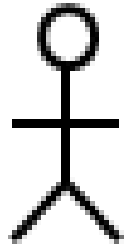
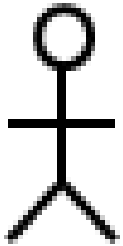


Collaborative Working



PM

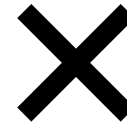
SA



?

25%

?



Time



?

25%

?



Function

Cost

Quality

Resource



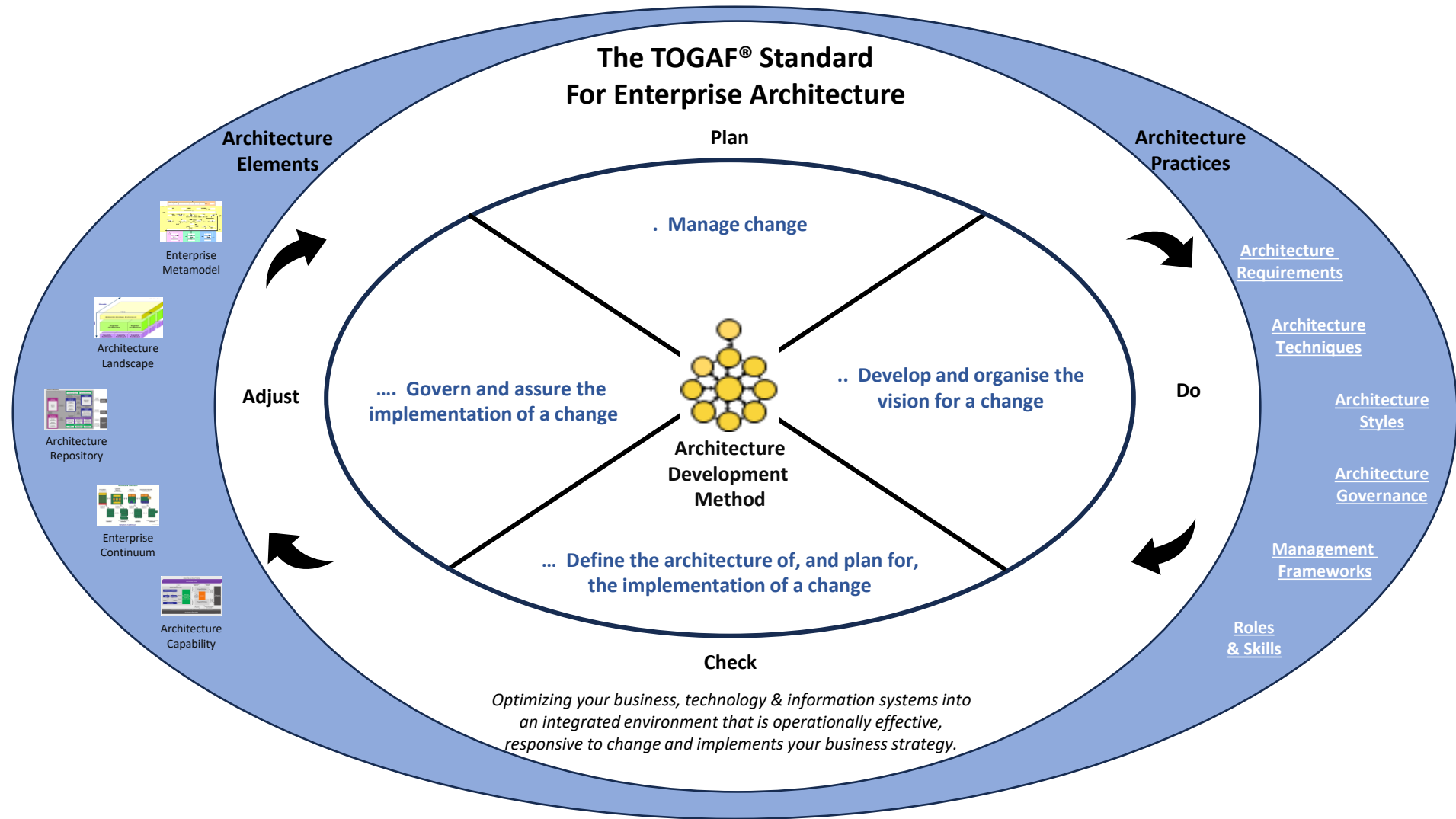
50%



Performance

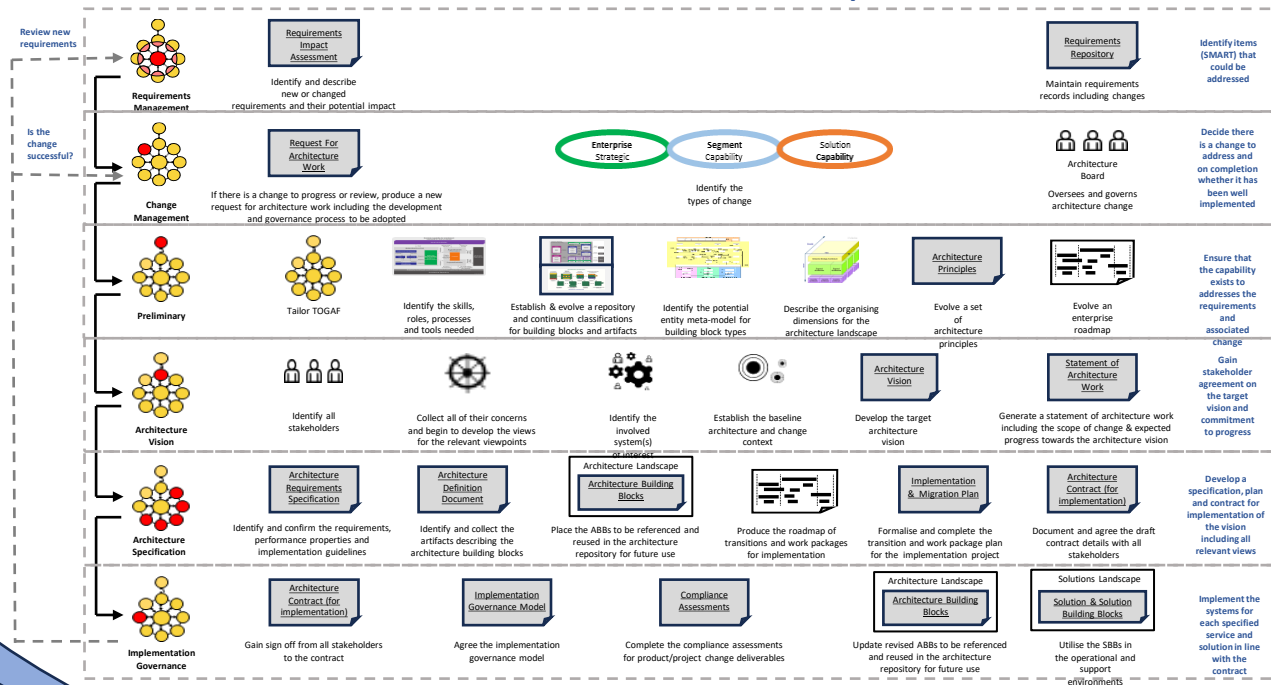


TOGAF Change Cycle

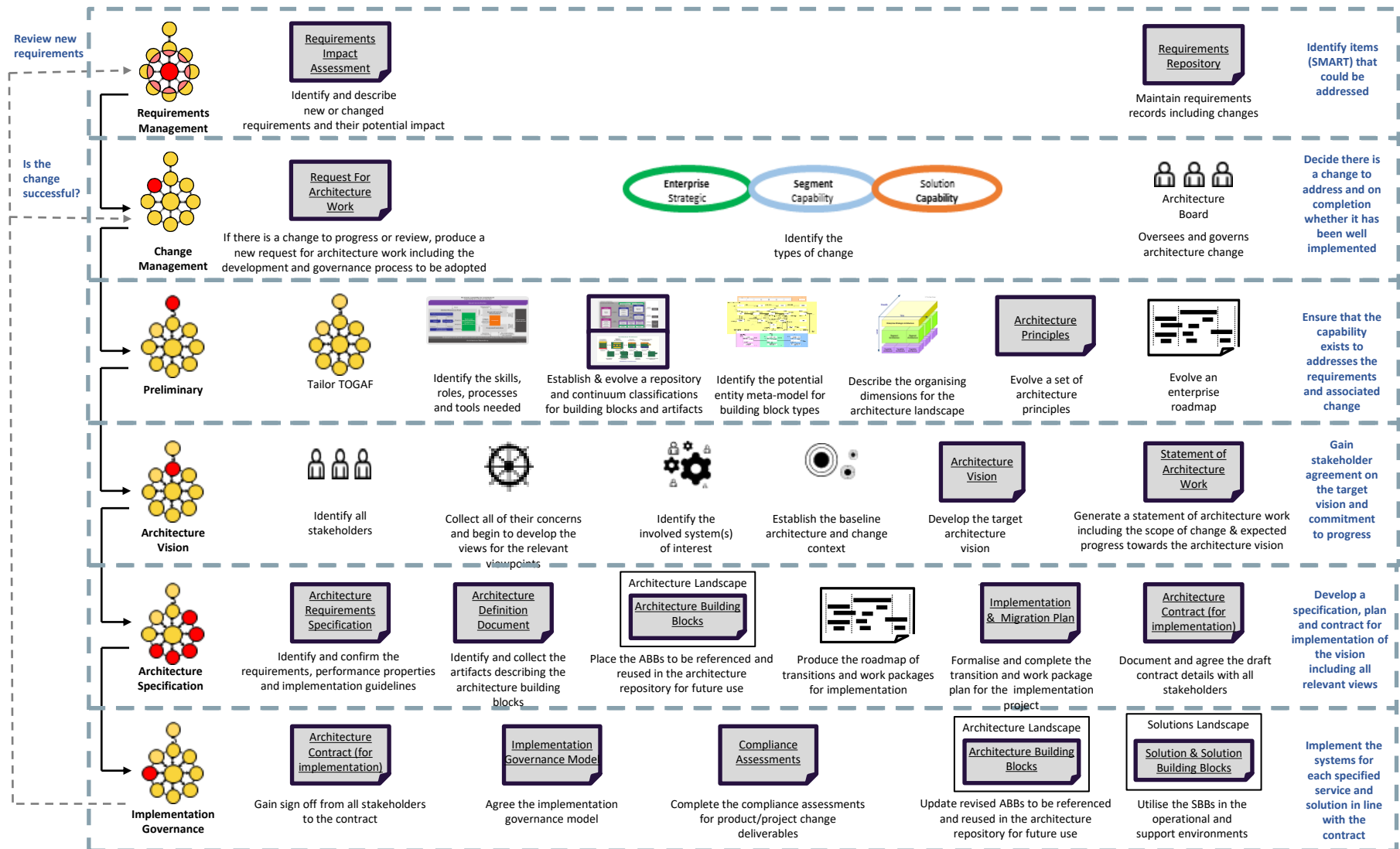


TOGAF Activities & Main Deliverables

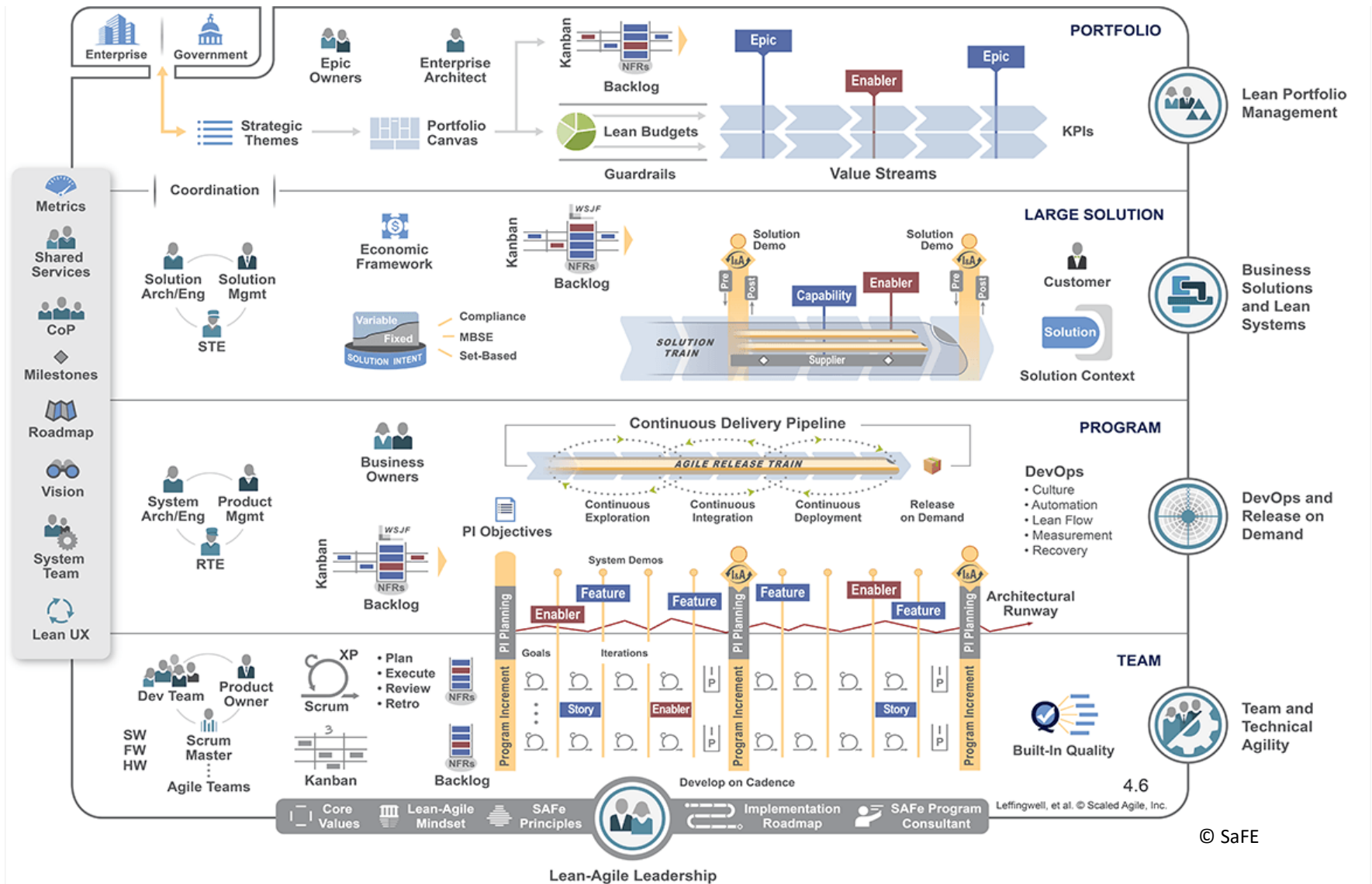
The TOGAF® Standard For Enterprise Architecture Delivering Effective Business, Technology & Information Systems Into Your Solutions Landscape



TOGAF Activities & Main Deliverables



The Four Viewpoints of Scalable Agile

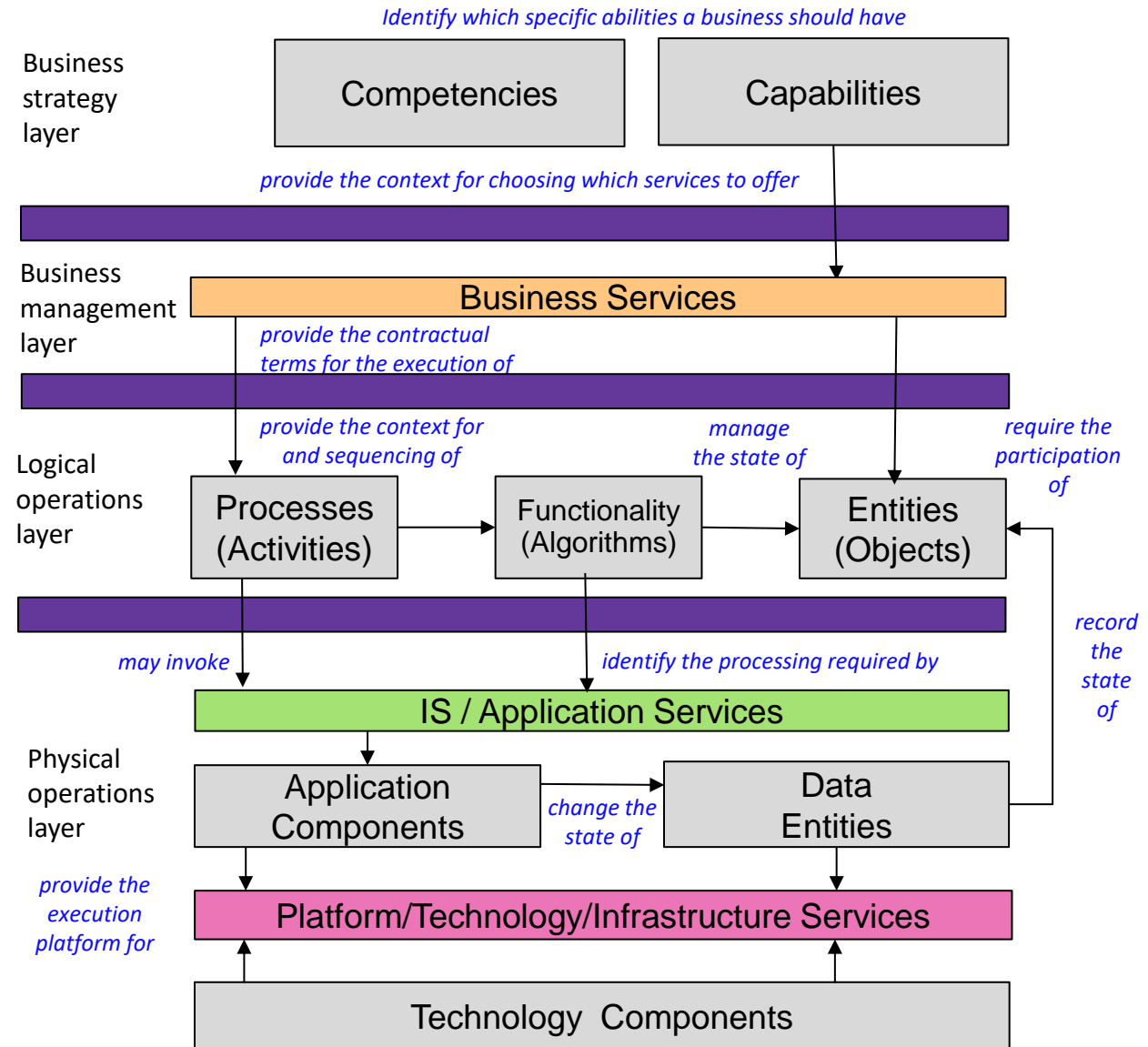


Making Sense Of The Capabilities, Services, Processes & Functions

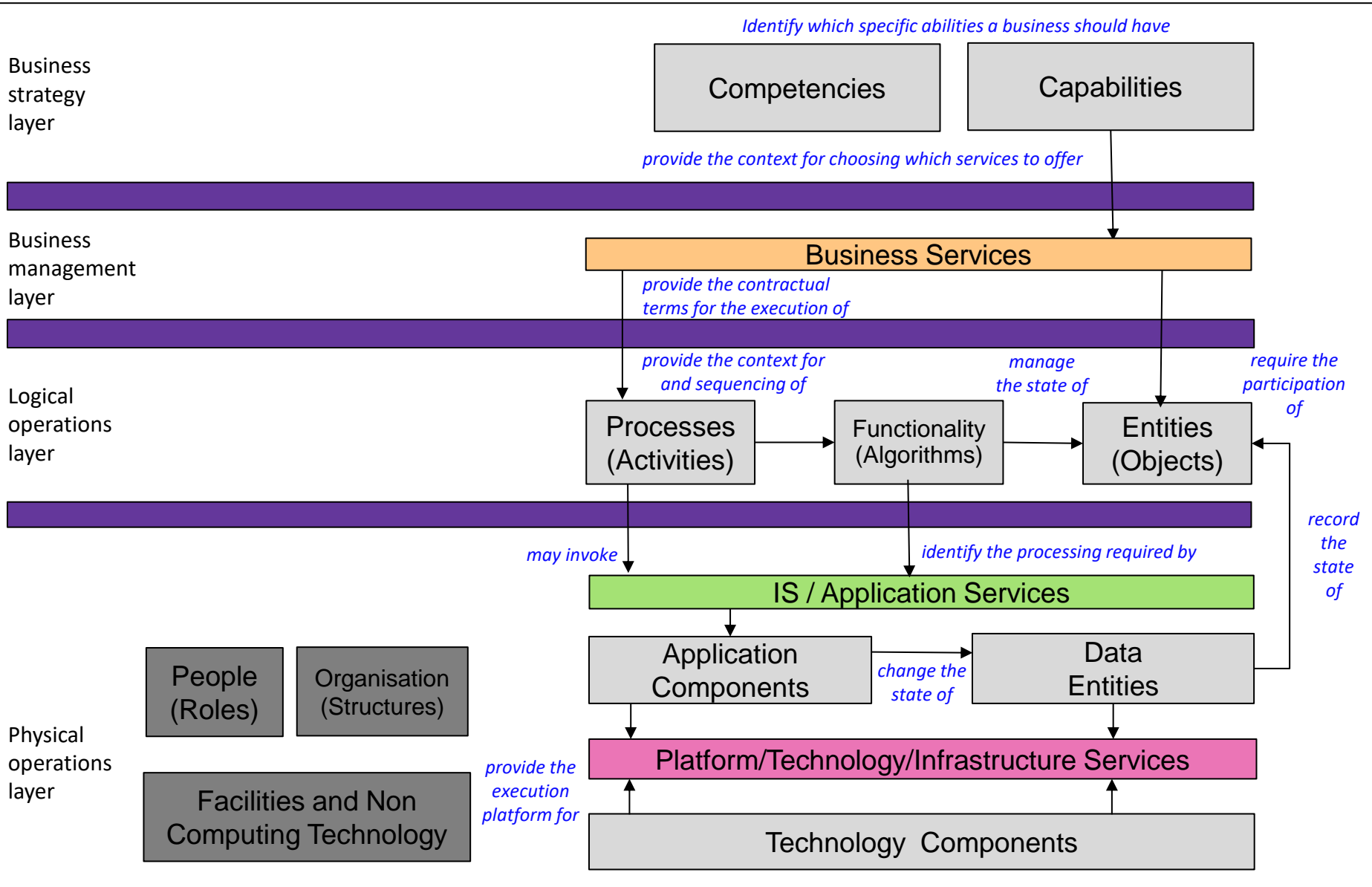
- The TOGAF content framework differentiates between the processes of a business and the services of a business.

- Business services are specific processes that have explicit, defined boundaries that are explicitly governed.
- Services are distinguished from processes through the explicit definition of a service contract that defines a post condition and its performance attributes.

- The granularity of business services should be determined according to the business drivers, goals, objectives, and measures for this area of the business. Finer-grained services permit closer management and measurement (and can be combined to create coarser-grained services), but require greater effort to govern.



Completing The Business Architecture View (POPIT)



Tools For Architecture Development - Archimate

There are ostensible advantages associated with selecting a single tool. Organizations following such a policy can hope to realise benefits such as reduced training, shared licenses, quantity discounts, maintenance, and easier data interchange.

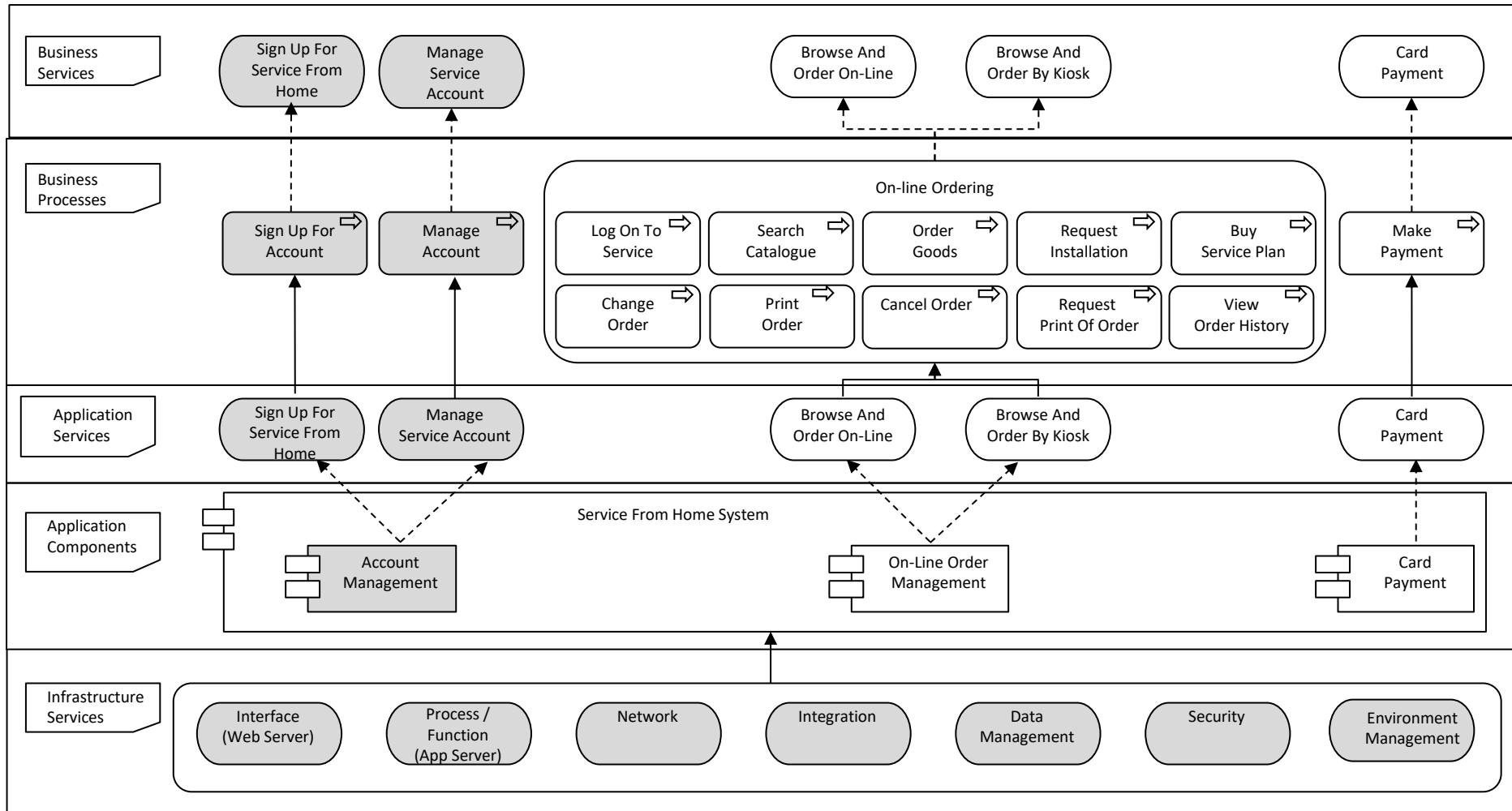
However, there are also reasons for refusing to identify a single mandated tool, and the fact that a single tool would not accommodate a variety of architecture development maturity levels and specific needs across an enterprise.

Successful enterprise architecture teams are often those that harmonise their architecture tools with their architecture maturity level, team/organizational capabilities, and objectives or focus, using a separate Portal to integrate information from various sources.

The Archimate Tool Components

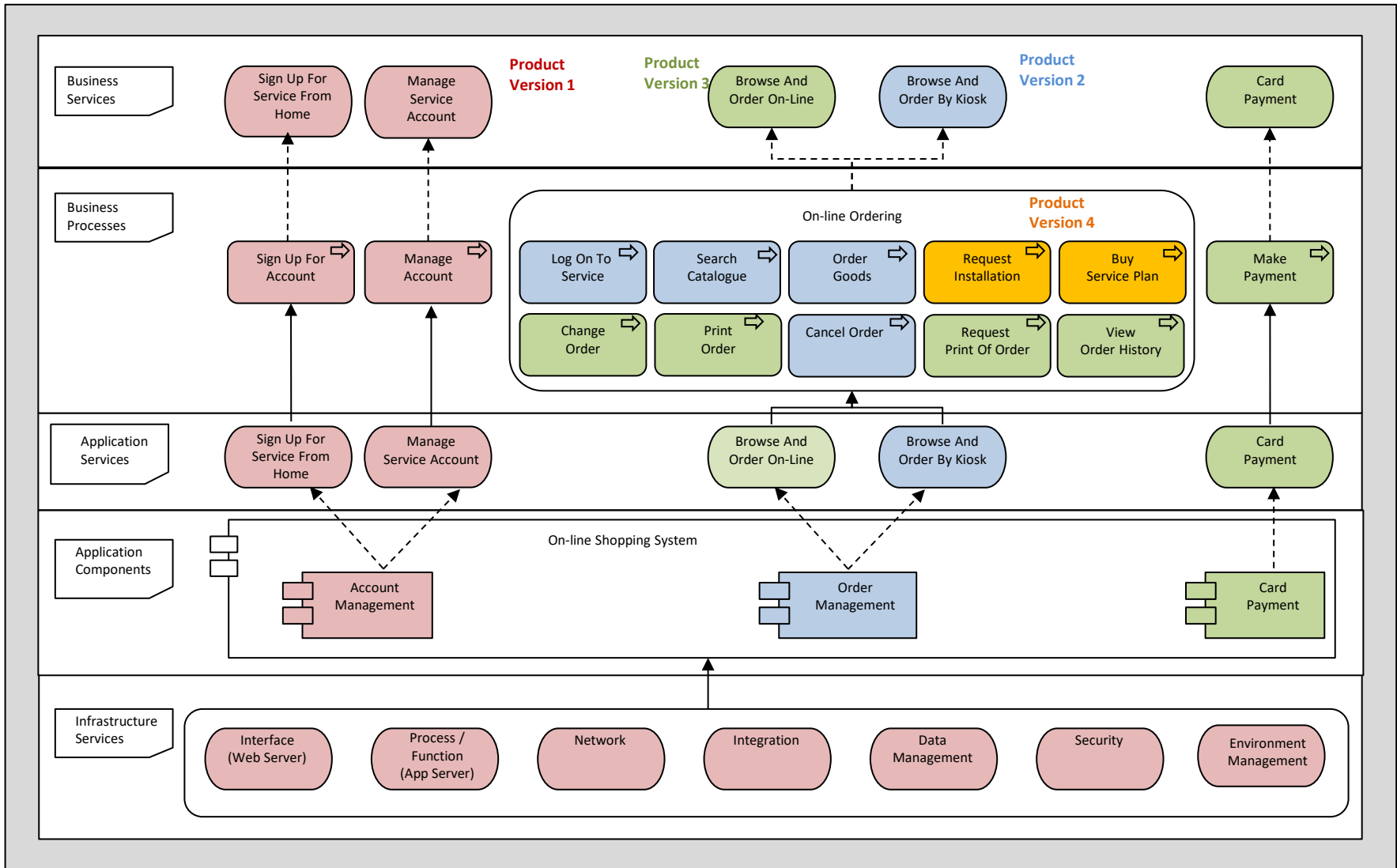
Extension (Steps P/A)	Core (Steps B/C/D)				Extension (Steps E/F/G)
Motivation		Active Structures	Behaviour	Passive Structures	Implementation & Migration
Stakeholder	Business View	Business actor	Business process	Business Entity / object	Work package
Goal (Objective)		Business collaboration	Business function	Product	Deliverable
Principle (Policy)		Location	Business interaction	Contract	Plateau
Driver	Information Systems / Application View	Application component	Application function	Data Entity / object	Gap
Requirement		Application interface	Application interaction		
Constraint	Technology View	Node	Infrastructure function	Artefact	(Capabilities)
Assessment		Network	Infrastructure service		(Architecture contracts)
(Measure)		Communication path			(Standards)
					(Guidelines)
					(Specifications)
					(Tools)

Project Context Diagram Archimate



A Project Context diagram shows the scope of a work package to be implemented as a part of a broader transformation roadmap. The Project Context diagram links a work package to the organizations, functions, services, processes, applications, data, and technology that will be added, removed, or impacted by the project. The Project Context diagram is also a valuable tool for project portfolio management and project mobilization.

Project Context Diagram Archimate (Versioned)



Why modelling?

The Unified Modelling Language (UML) is the preferred notation for modelling software rich business solutions. As most business processes are supported by ICT systems it has an application for most areas of our business.

Modelling for business and software systems was initially developed through the 1970s-1990s as businesses began to use more systematic methods of improving operation and increasing automation of processes.

Modelling enabled a better understanding of existing processes and systems and was increasingly aligned to standard taxonomies creating shared concepts and terms.

When we look to create new business solutions we are faced with the tasks of understanding:

- The current state of our business environment
- The specific requirements for change
- How we translate those requirements into changes in our business environment

To help us in this we develop models. Models are abstractions of something for the purpose of understanding and generally represent specific points of view and simplifications of the real world.

They enable us to focus on the essential elements and properties of a desired solution by:

- Highlighting different viewpoints and views
- Identifying important structures and patterns
- Improving communication with various stakeholders
- Enabling early testing before full scale creation
- Reducing complexity for analysis and overall design
- Providing pre-defined concepts for optimising complex relationships when needed

The basic properties of UML

UML is a standardised visual modelling language intended to be used for modelling business process and their underlying elements and the analysis, design, and implementation of software based systems.

UML can be applied to diverse business domains (e.g., banking, finance, internet, aerospace, healthcare, telecommunications, etc.) and can be used with all major object and component software development methods and for various implementation platforms (e.g., J2EE, .NET).

UML was developed in the early 1990s to provide a consolidated standard industry modelling language (from the pre-existing object, function, process and data oriented modelling languages) and has since become the worldwide standard managed by the Object Management Group (OMG).

- UML version 1.1 adopted by OMG and updated to 1.3, 1.4, 1.5 with minor revisions and additions.
- UML version 2.0 was introduced in Oct 2004 to support Model-Driven Architecture/Model-Driven Development 2.4 is the version released in 2011.
- Version UML 2.x refers to any of the version 2 minor versions.

It is intentionally development process independent and can be applied in the context of different processes. However, it has been developed in line with use case driven, iterative, agile and incremental development processes. An example of such a process is Unified Process (UP).

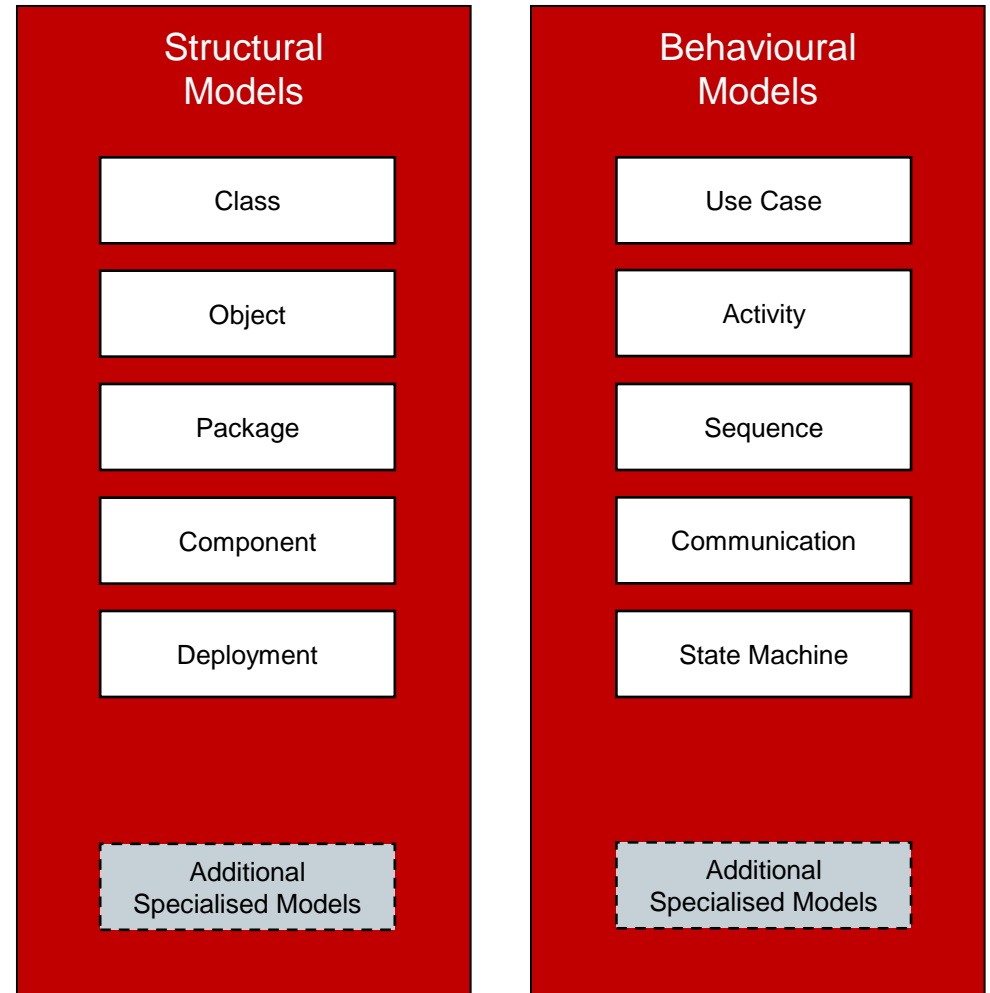
The standard UML models

UML comprises a set of standard structural and behavioural models (shown on this slide) supplemented by some more specialised ones such as:

- Composite structure
- Interaction overview
- Network architecture
- Timing
- Locality

The structural models represent the "things" in the problem space, how they are grouped, and their relationships.

The behavioural models represent actions/activities and flows in the problem space, how they are connected and how they are sequenced.



When you use the models

The models can be used to capture three different levels of information:

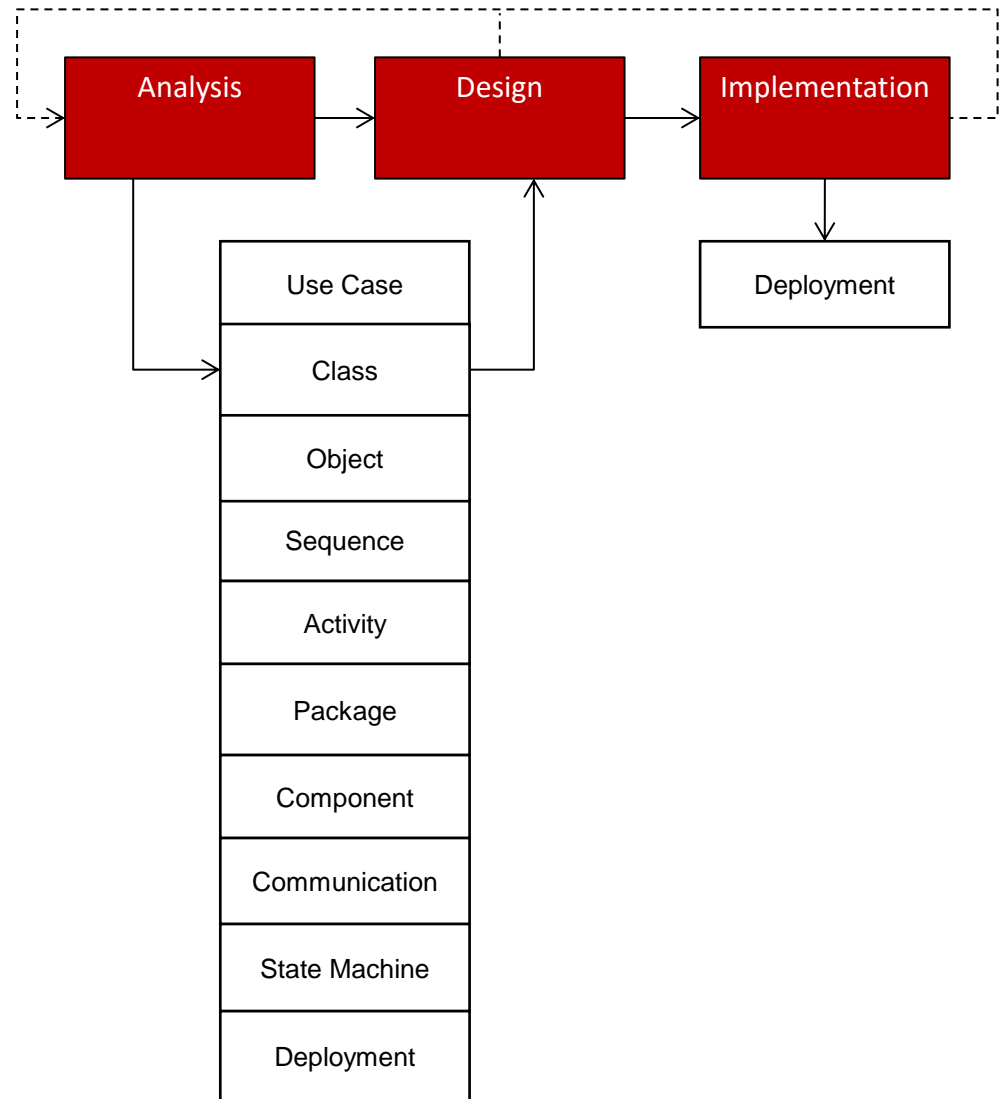
- Analysis (As Is and To Be)
- Design
- Implementation

All of the models can be used for analysing the current or proposed situation and capturing aspects of the design.

You should choose the models relevant to the specific solution being reviewed, changed or developed.

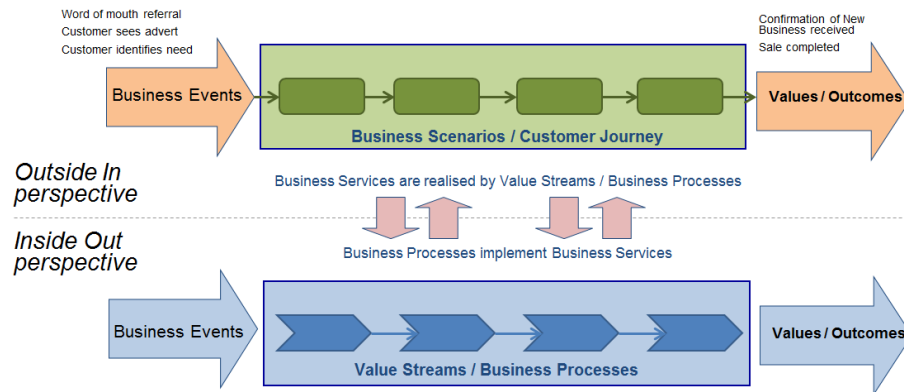
In a number of cases you will want to maintain both analysis and design models as on-going documentation.

You may also layer the analysis and design models to reflect different levels of detail such as an overall architecture view or a detailed design view.

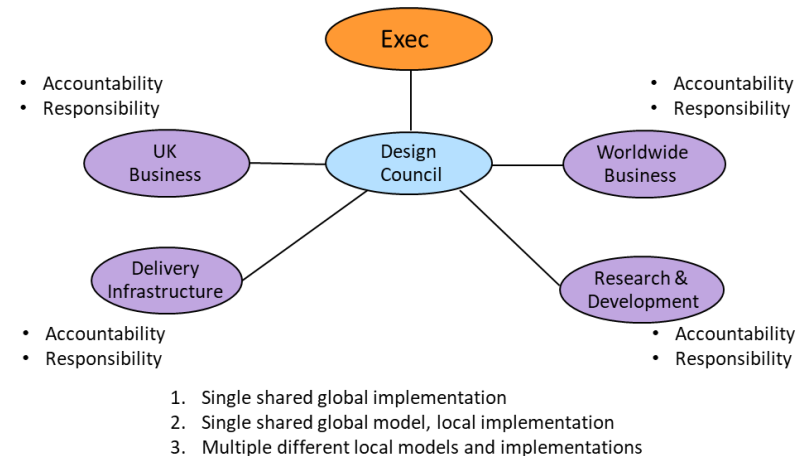


Example Business Models

Business Approach Model



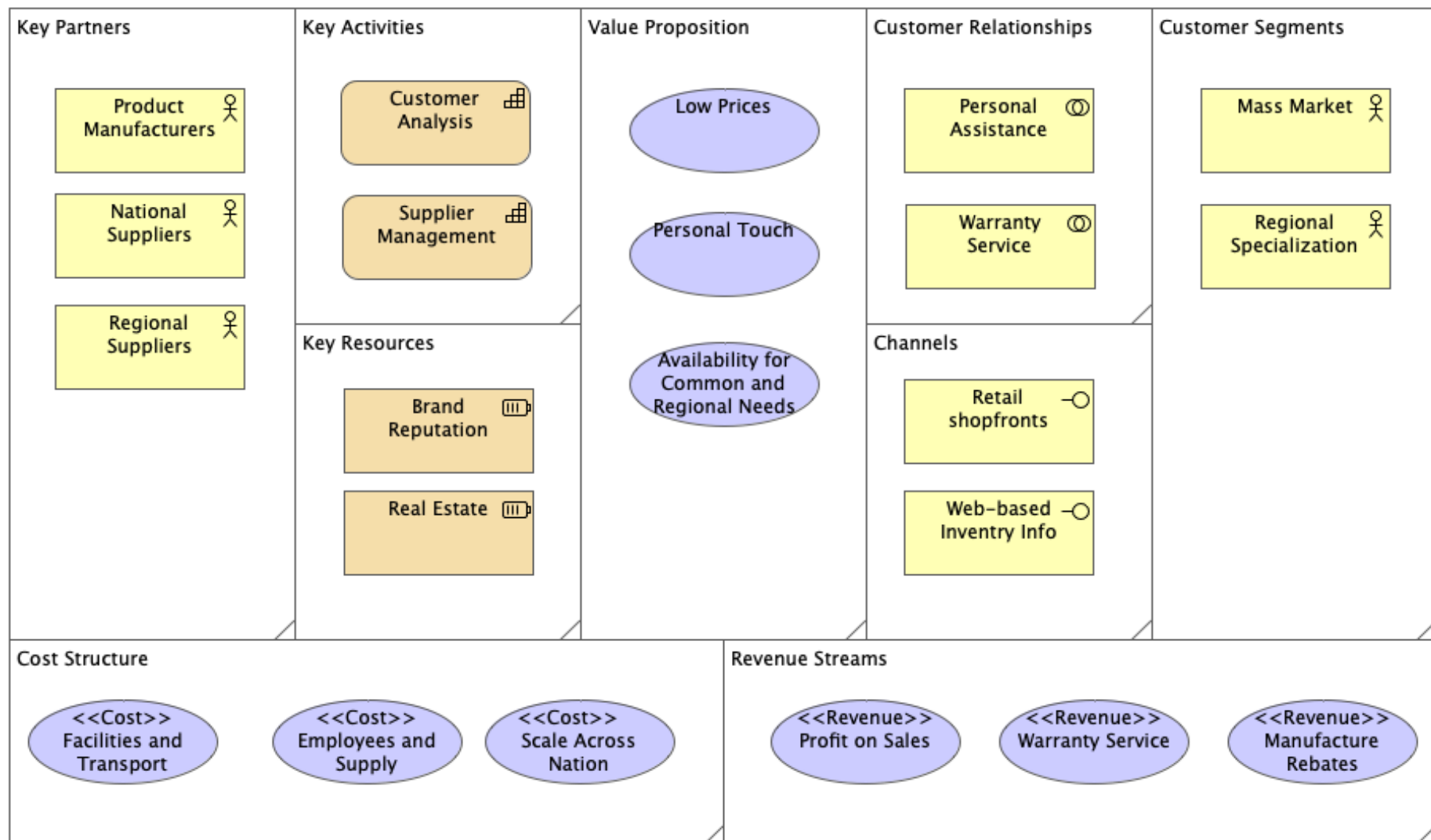
Business Operating Model



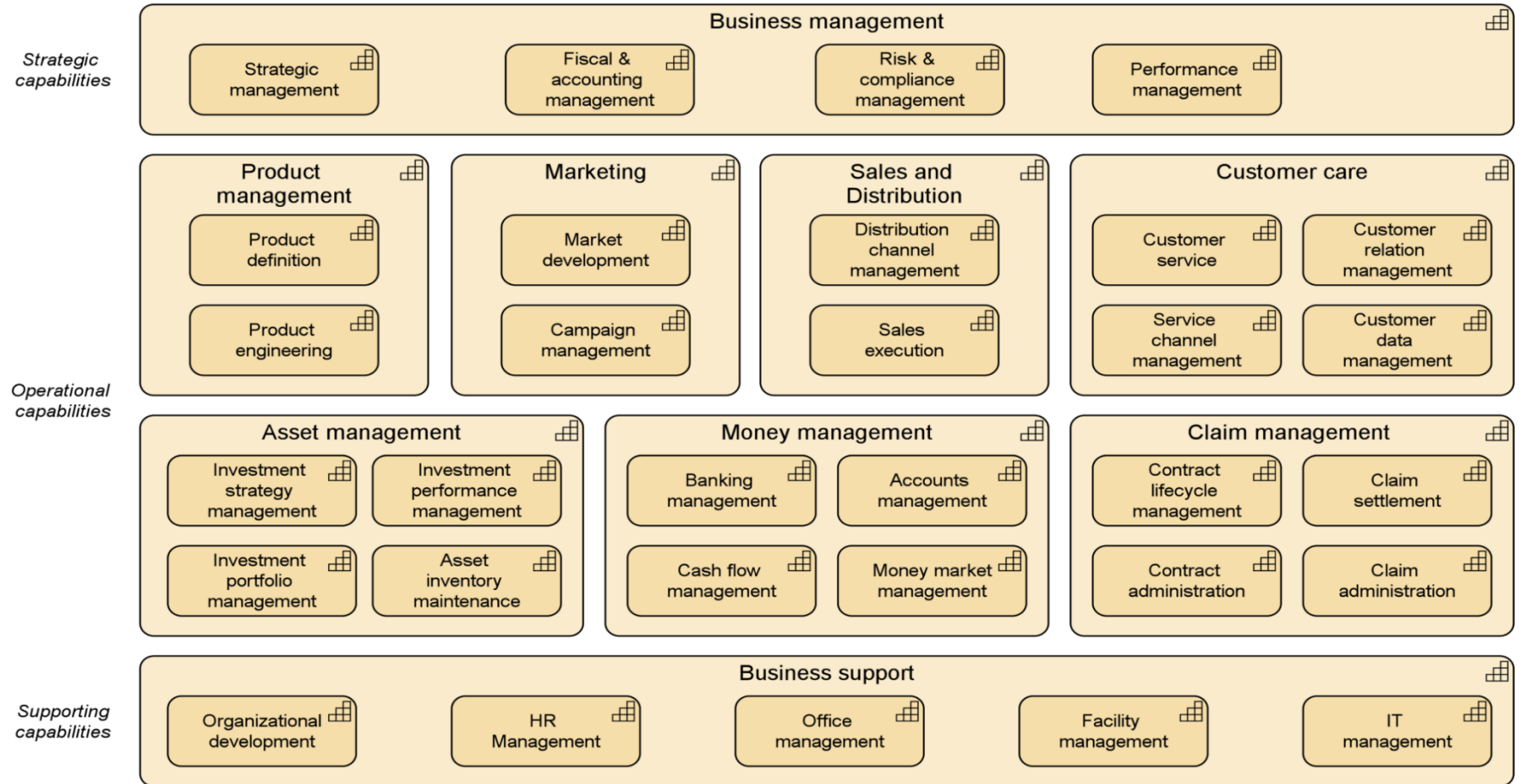
Example interoperability/integration patterns

A Business Model is an arbitrary abstract diagram that provides a coarse grained/high level view of how a business is structured and organised

Business Model Canvas



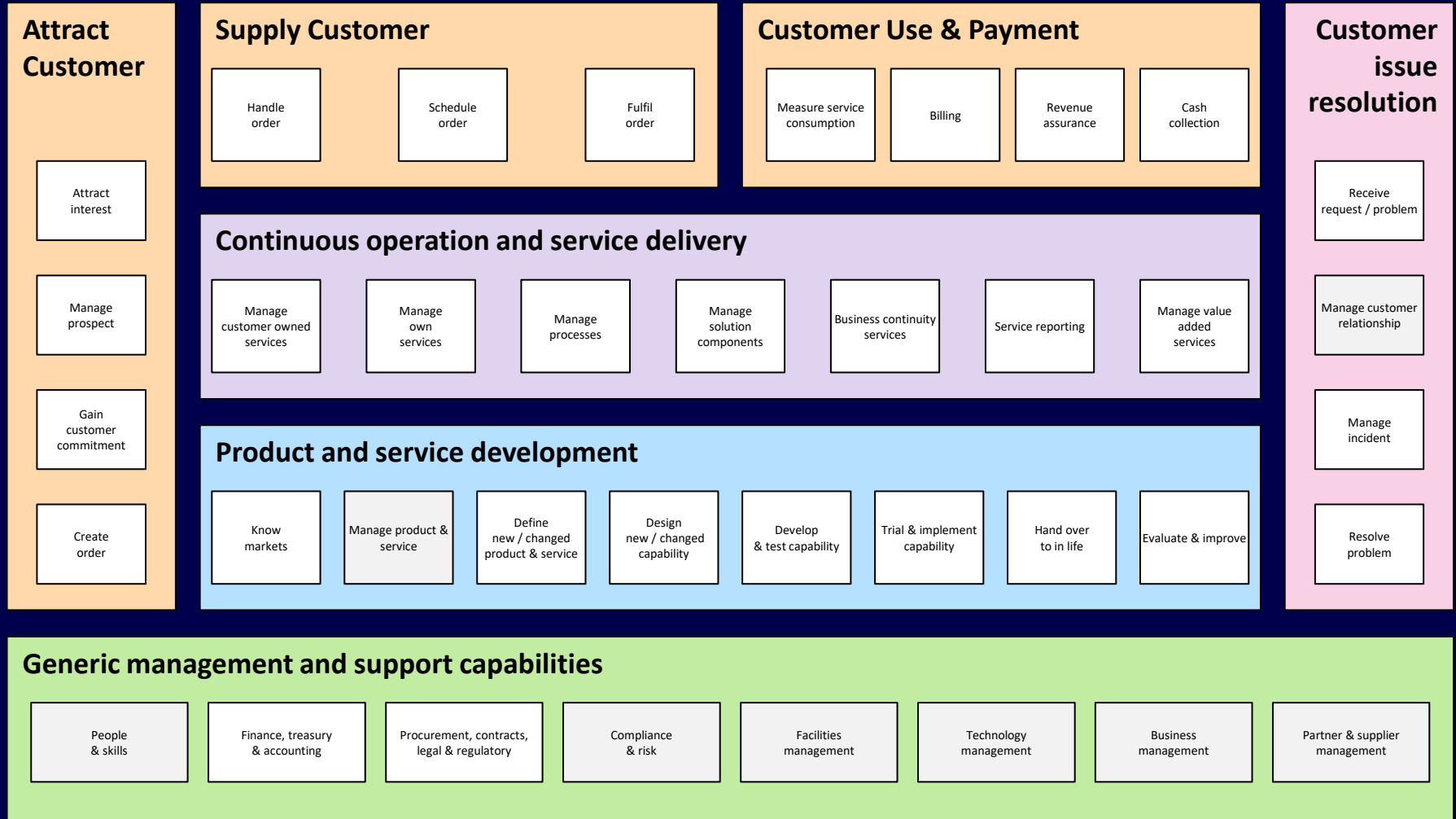
Example Business Capability Map



A Business Capability map is a coarse grained/high level view of business structure, functionality and activity to assist in planning, running and managing a business.

Value Streams & Business Capabilities

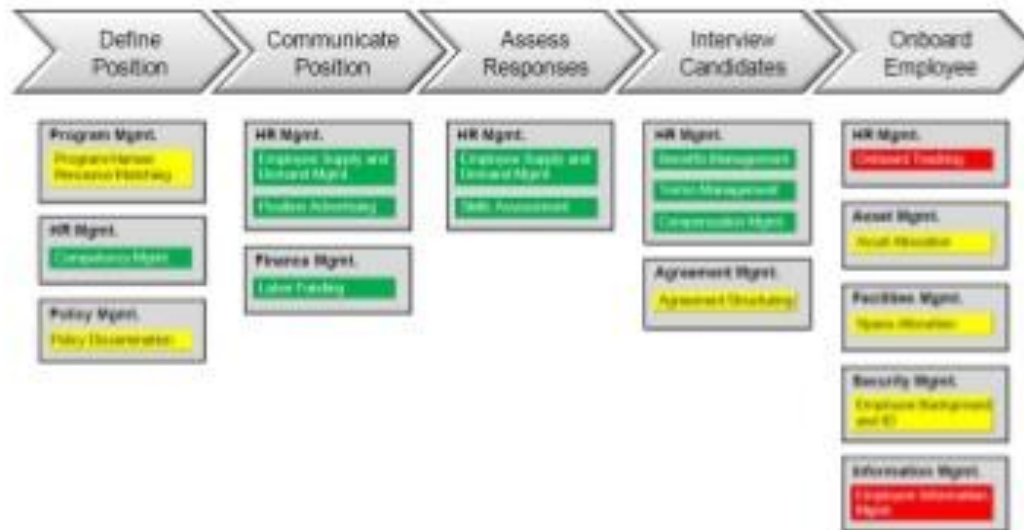
Example Business Capability (38) Map - Structured By Value Streams (7)



Series Guide: Value Streams



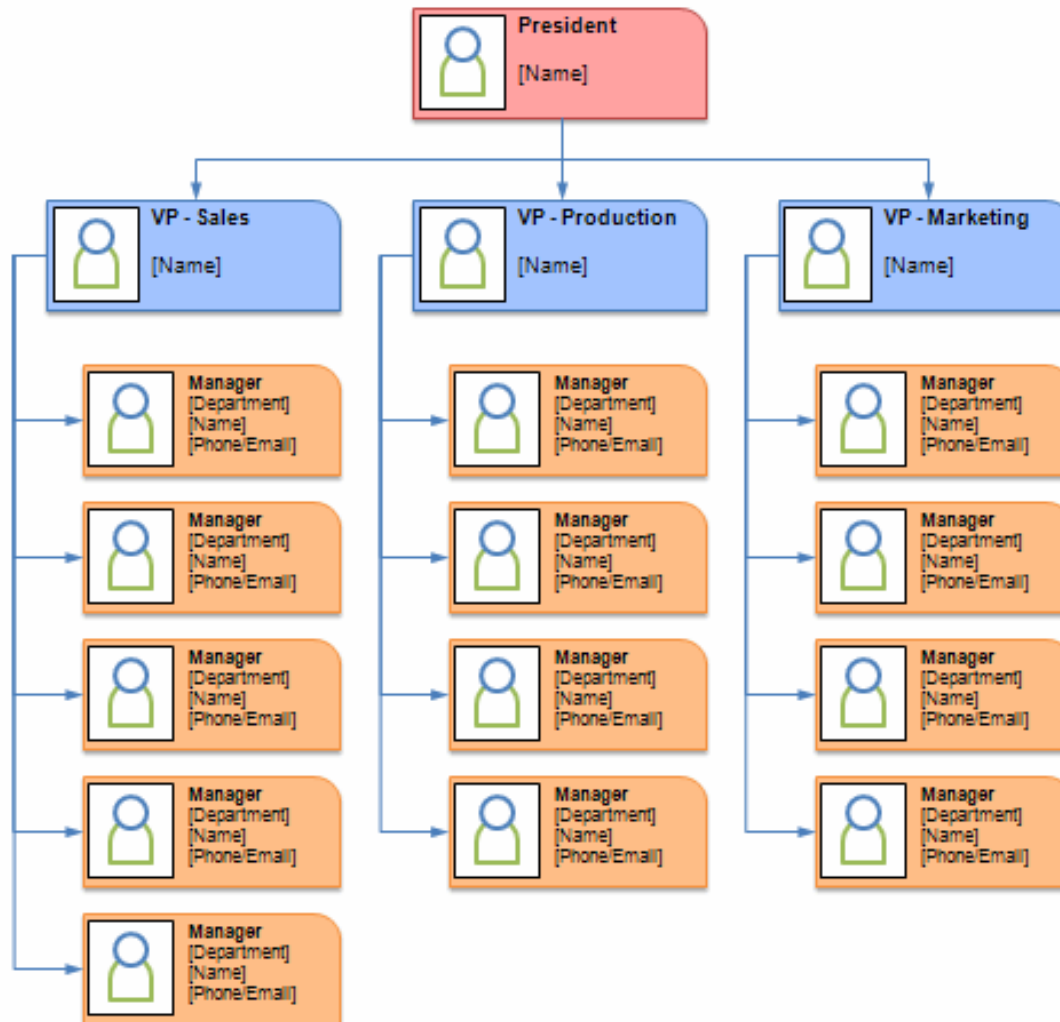
This series guide provides guidance on how to to identify, define, model, and map a value stream to...



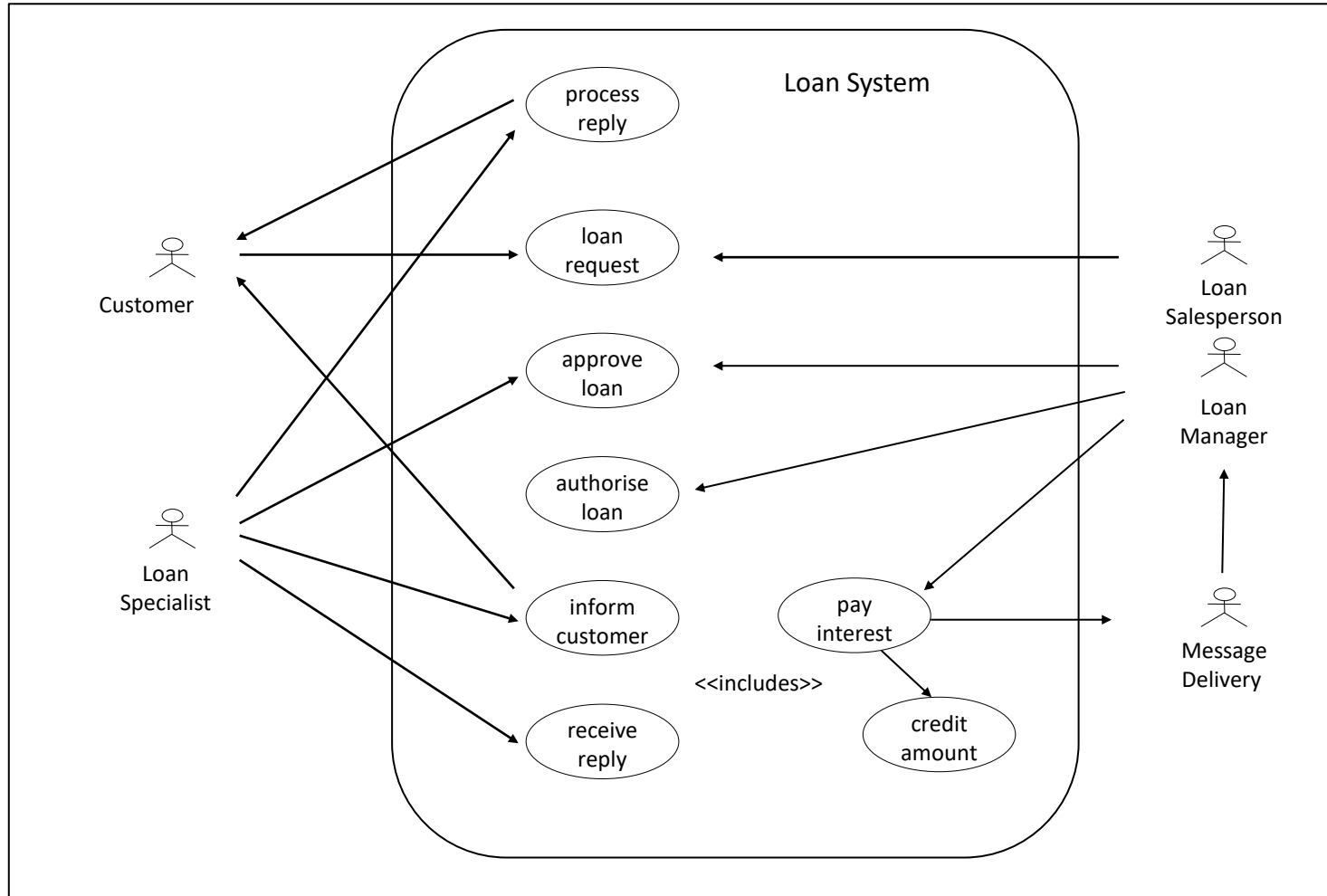
A Value Stream is a coarse grained/high level view of a full Process Flow focusing on key value elements not a full process flow model

Example Organisation Map

Company Organizational Chart



Example Use Case Model



A Use Case diagram shows Actors interacting with a System of concern identifying the "goal directed steps that deliver value to the user of the case"

Use case description

Each use case has a description in the form of a set of structured clauses.

This format can be varied, the one shown is about average for the format type and level of complexity .

The aim of a use case is to clearly describe all of the exchanges between the actors and the system; and the state that the system will be left in once the use case completes.

A use case should be written so that all stakeholders can understand it minimising “technical” jargon.

High level use cases are similar to the later developed concept of user stories but tend to have a more formal structure.

Use Case: Pay Interest

Description:

The due interest on all deposit accounts for the last month has been calculated and credited to the accounts and a report produced showing the interest paid.

Actors:

Manager

Printer Queue

Assumptions:

None

Steps:

- The Manager starts the pay interest Use Case (on the 28th of each month)
- The pay interest job is submitted to the system
- For each deposit account held within the deposit account file, the interest due is calculated and the credit amount Use Case is executed for each account that has interest due.
[Exception: - Deposit Account Value ≤ 0]
- When all of the accounts have been processed a report is created and stored in the system showing the interest paid on each account and the total amount of interest paid; an e-mail is sent to the Manager informing them to access and review the report; and the use case terminates

Alternate Course:

None.

Exceptions:

[Deposit Account Value ≤ 0]

If a deposit account is found with a balance of zero or less then no interest is calculated, the credit amount use case is not executed for that account, and the report entry for the account will clearly highlight the account value.

Use case description – quality and performance attributes

The use case description can be extended by including a statement about the performance and quality attributes.

The set of performance and quality attributes should reflect the latest set requested as part of our specify and plan process this can be as small as 5 and as many as 30-40.

A sample set is shown on the slide.

Use Case Performance Attributes:

- **The number of instances of each Actor:**
There is one manager for each branch.
- **The geographical distribution of the Actors / System Components:**
The managers will be located in the branch along with the branch system upon which the request for processing interest payments is input. The calculation and payments will be processed on a central server system.
- **The number of concurrent instances of the Use Case:**
A branch may have 1 concurrent instance of the Use Case. The central system may have up to 500 concurrent requests from branches for processing interest payments. These will be queued and processing sequentially
- **The response time required for an individual instance of this Use Case:**
The Use Case should be able to be executed overnight in the month end batch processing window of 17.00 to 08.00. All of the branch runs together should be processed in a maximum of 3 hours.
- **The security requirements for this Use Case:**
The Use Case should only be able to be executed by authorised Bank Employees performing the role of a Manager.
- **The reliability of this Use Case in its execution:**
The Use Case should always execute reliably unless there are system infrastructure problems from which the Use Case should be able to exit gracefully.

Supplementing the use case with an interface prototype

While not a formal part of UML there is often a need to produce a model of the user interface to improve the discussion about the requirements and to also identify and discuss all of the alternate and failure paths.

In some cases, initial discussions about requirements should ignore the physical interface issues and just focus on the business events. In others the user interface makes it much easier to have discussions about the business events (but be careful not to allow the nature of the interface to determine the business events).

Loan Specialist Activity Form

Activities Allocated / Not Started		Activities Started
Loan ID	Activity	Loan ID Status Selected Date
Loan LP1234	approval	Loan LP1214 approving 03/04/98
Loan LP1237	approval	Loan LP1238 informing 03/04/98
Loan LP1267	inform customer	
Loan LP1289	process reply	

Proposed Loan ID	<input type="text" value="LP1214"/>	Requested Amount	<input type="text" value="\$30,000"/>														
Current Status	<input type="text" value="approving"/>	Approved Amount	<input type="text"/>														
Date Activity Started	<input type="text" value="03/04/98"/>	Authorized Amount	<input type="text"/>														
Date Loan Requested	<input type="text" value="03/04/98"/>	<table border="1" style="width: 100%; border-collapse: collapse;"><thead><tr><th>Note</th><th>Date</th></tr></thead><tbody><tr><td>Other Loans</td><td>03/04/1998</td></tr><tr><td>Previous Requests</td><td>03/04/1998</td></tr><tr><td>Credit Rating</td><td>04/04/1998</td></tr><tr><td> </td><td> </td></tr><tr><td> </td><td> </td></tr><tr><td> </td><td> </td></tr></tbody></table>		Note	Date	Other Loans	03/04/1998	Previous Requests	03/04/1998	Credit Rating	04/04/1998						
Note	Date																
Other Loans	03/04/1998																
Previous Requests	03/04/1998																
Credit Rating	04/04/1998																
Date Approved	<input type="text"/>																
Date Refused	<input type="text"/>																
Date Authorized	<input type="text"/>																
Date Customer Informed	<input type="text"/>																
Date Accepted	<input type="text"/>																
Date Rejected	<input type="text"/>																
Date Account Credited	<input type="text"/>																

Supplementing the use case with scenarios using sequence diagrams

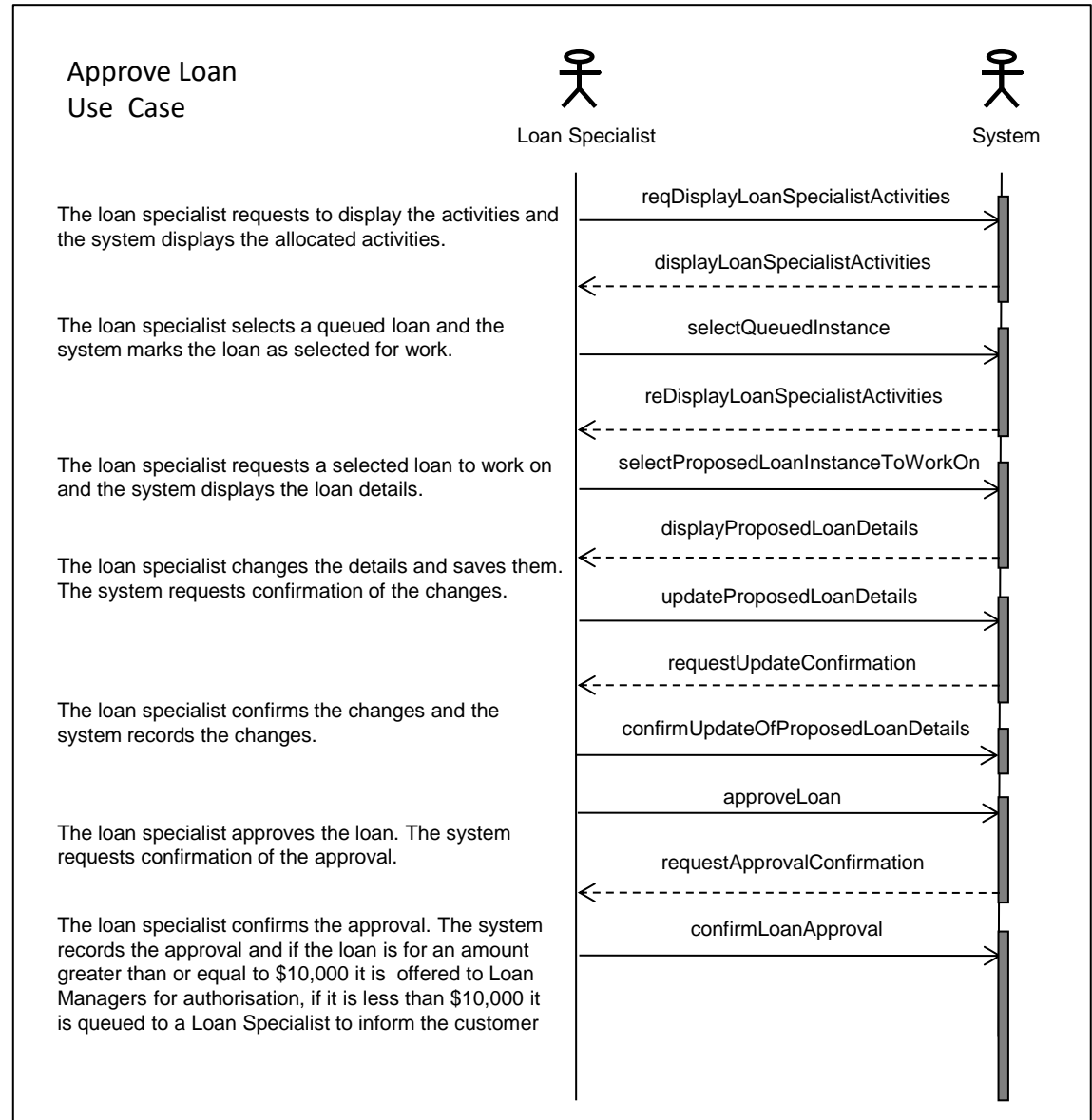
A scenario diagram is a specific type of sequence diagram that describes the interactions between an actor and a system when a use case is run.

Each specific event is identified and the exchange that occurs during the event.

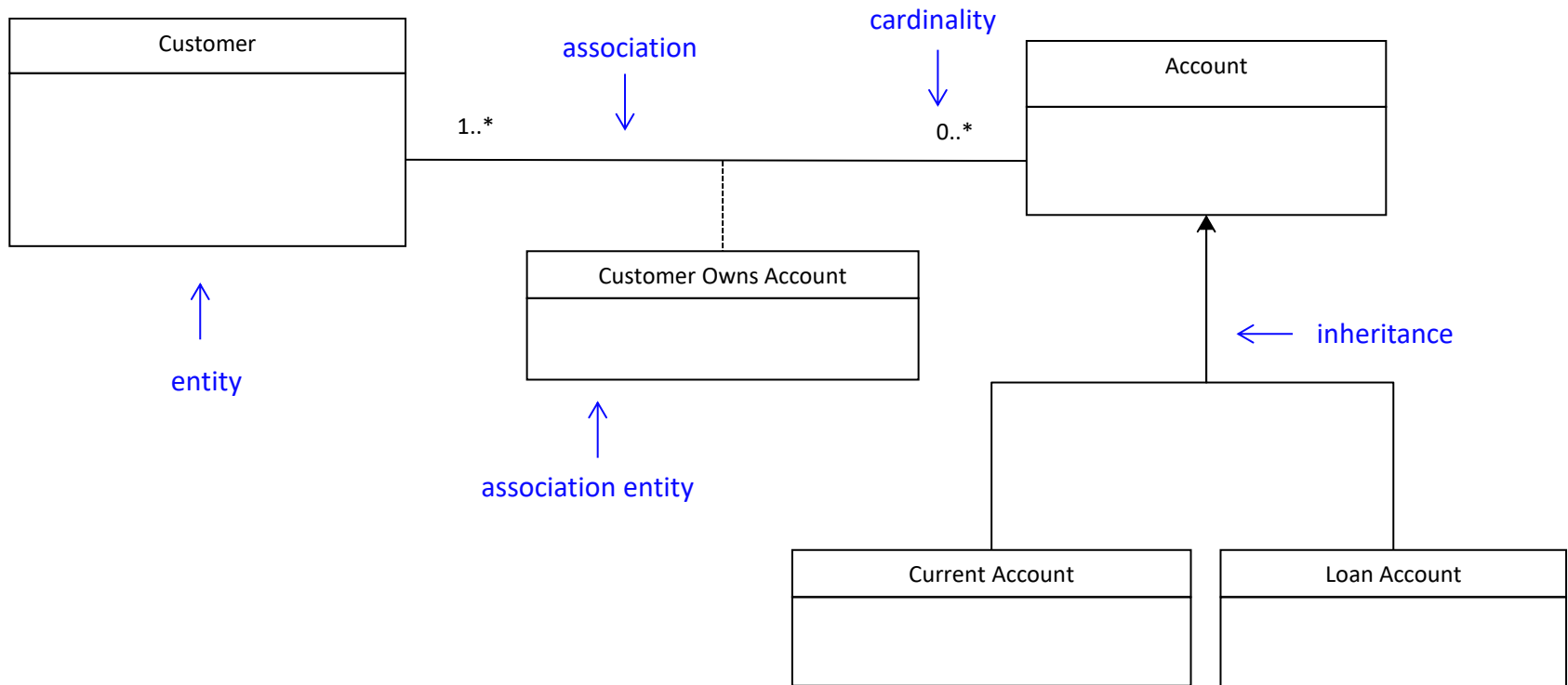
Multiple scenarios may be modelled for the “happy path”, “alternate paths” and “failure paths”.

The one shown is the “happy path” for the approve loan use case.

The scenario can be described by using the defining sentences and / or the specific operation names depending on the level of detail required, the audience, and the need to connect the logical operation to their specific implementations.



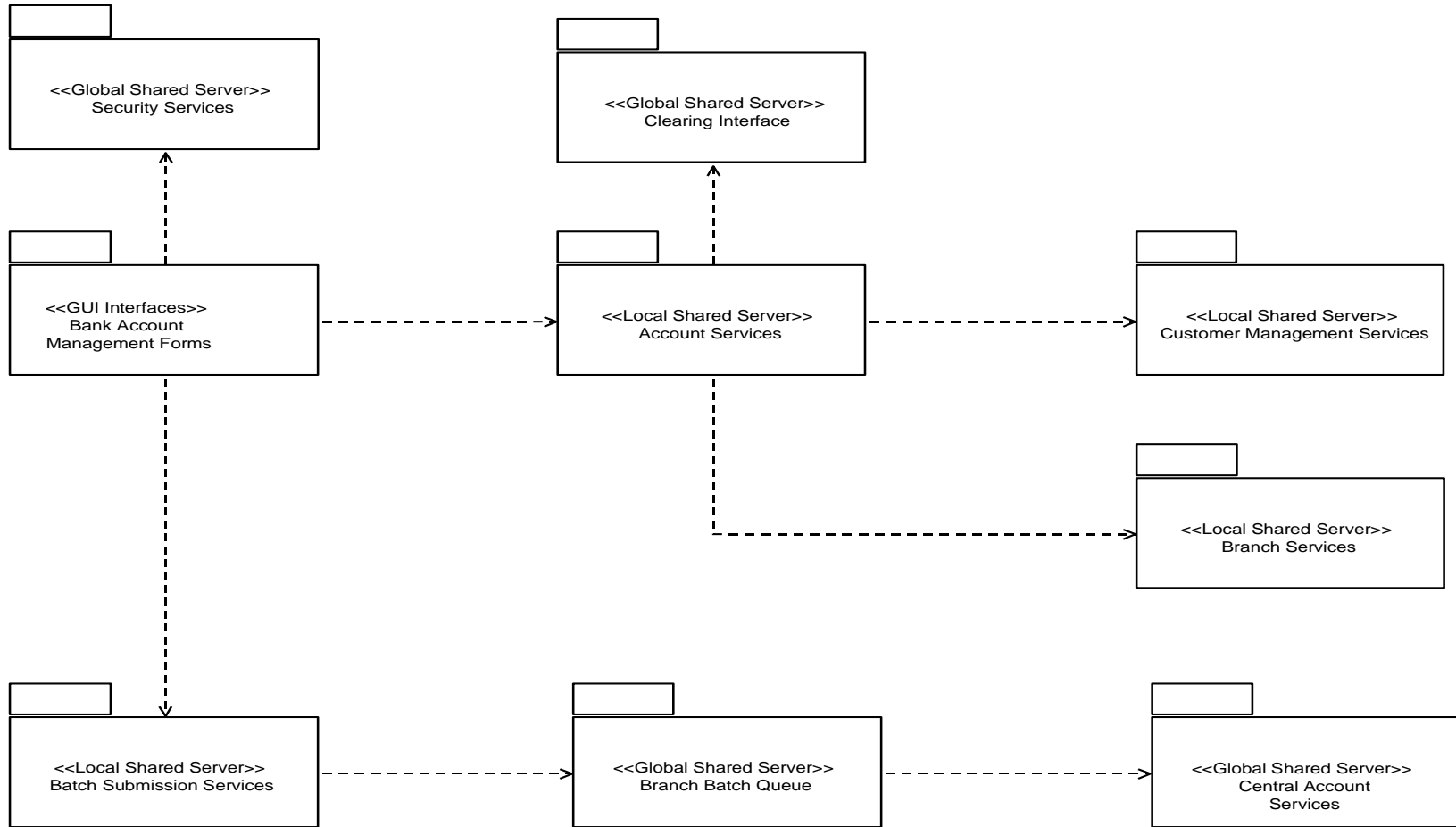
Information Model / Conceptual Data (Entity - Relationship) Diagram



The key purpose of the Conceptual Data diagram is to depict the relationships between critical data entities within the enterprise.

High Level / Vision Package Architecture (UML Package Diagrams)

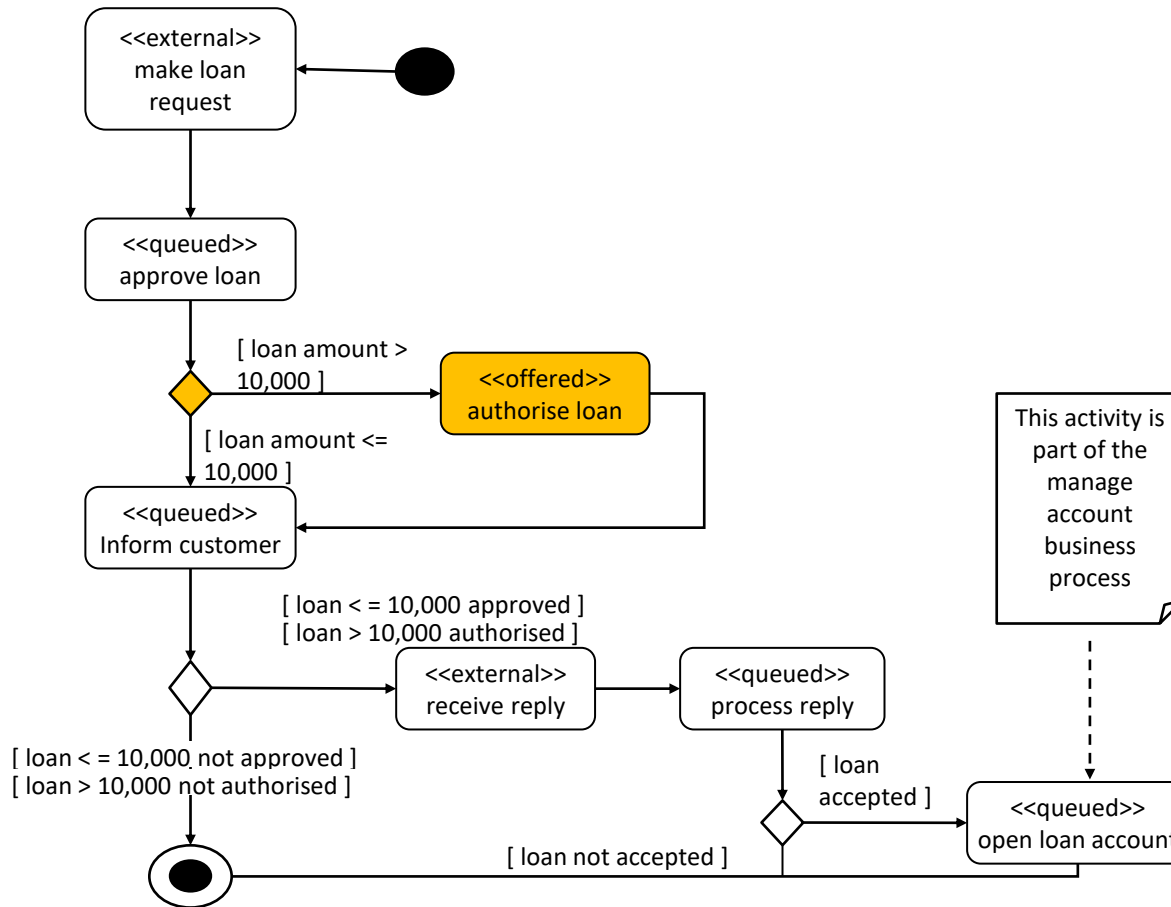
Banking System



Component Collaboration & Responsibilities

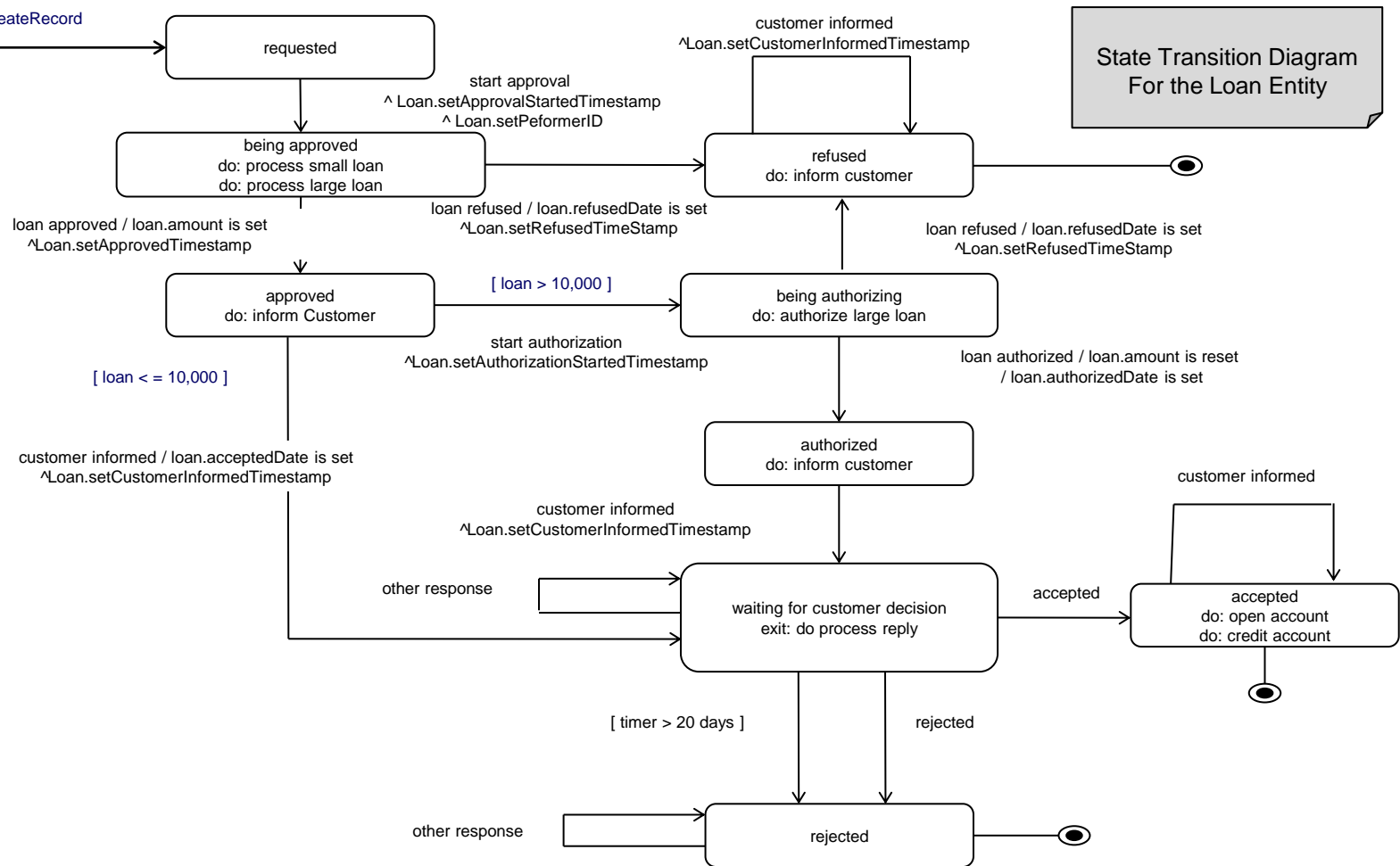
Component	Responsibilities	Collaborating Components
Account Services	Create Accounts	Branch Services
	Close Accounts	Customer Services
	Debit Accounts	Clearing Interface
	Credit Accounts	
	Transfer Funds	
	Get Balance	
	Validate Account Types	
	Validate Account	
Customer Services	Validate Customer	
Branch Services	Validate SortCode	
Clearing Interface	Update Target Account	
Central Account Services	Process Interest Payment	Batch Submission Services
Batch Submission Services	Submit Process Charges Request	Branch Batch Queue
	Process Charges Completed	
	Submit Pay Interest Request	
	Pay Interest Completed	
Branch Batch Queue	Process Process Charges Queue	Central Account Services
	Queue Process Charges	Batch Submission Services
	Process Pay Interest Queue	
	Queue Pay Interest	

Process Model



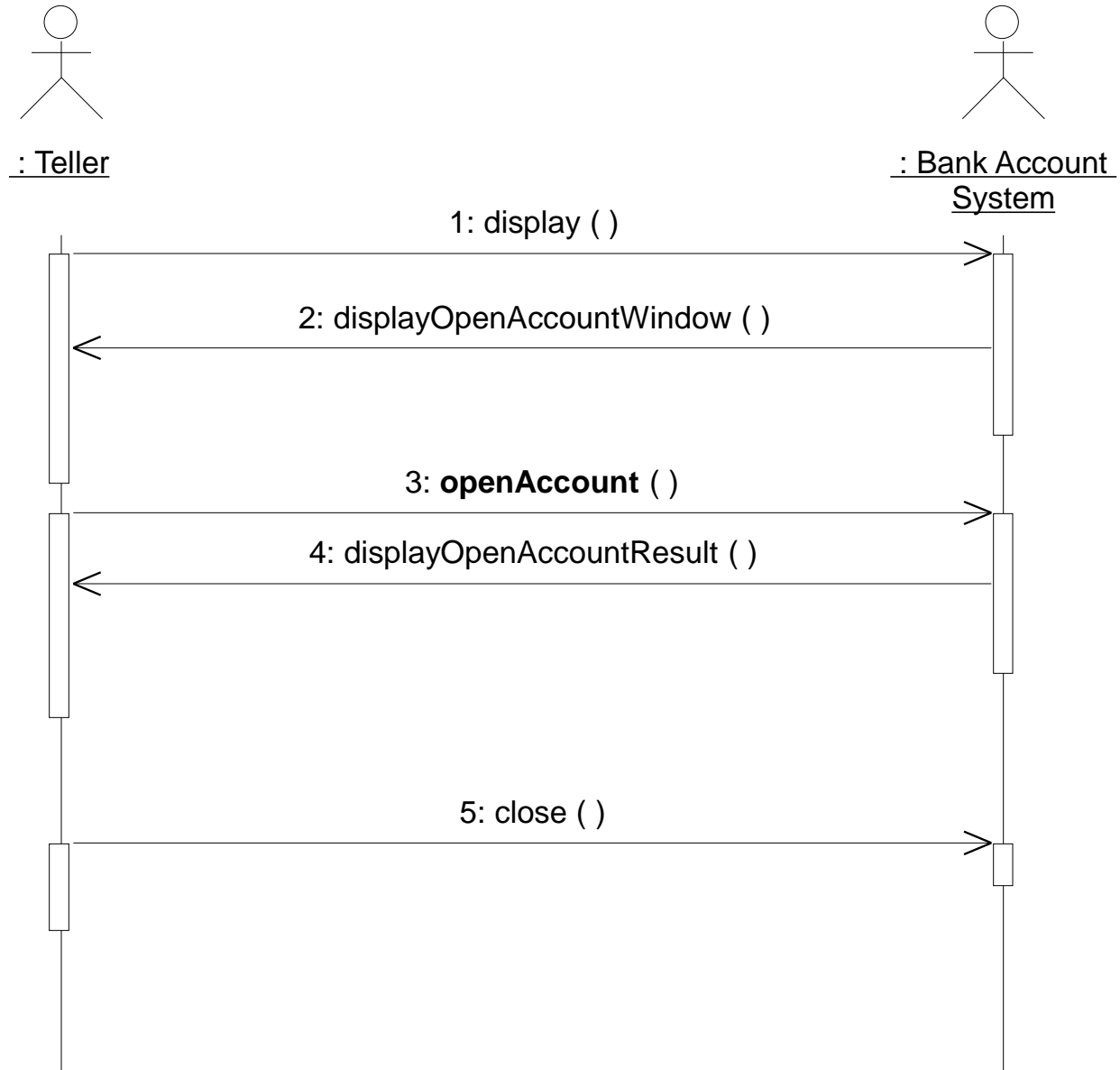
A Process Model describes the flow between a set of activities that generate some meaningful value to the process users and together deliver a coherent set of business function.

Example State Transition / Data Lifecycle Diagram

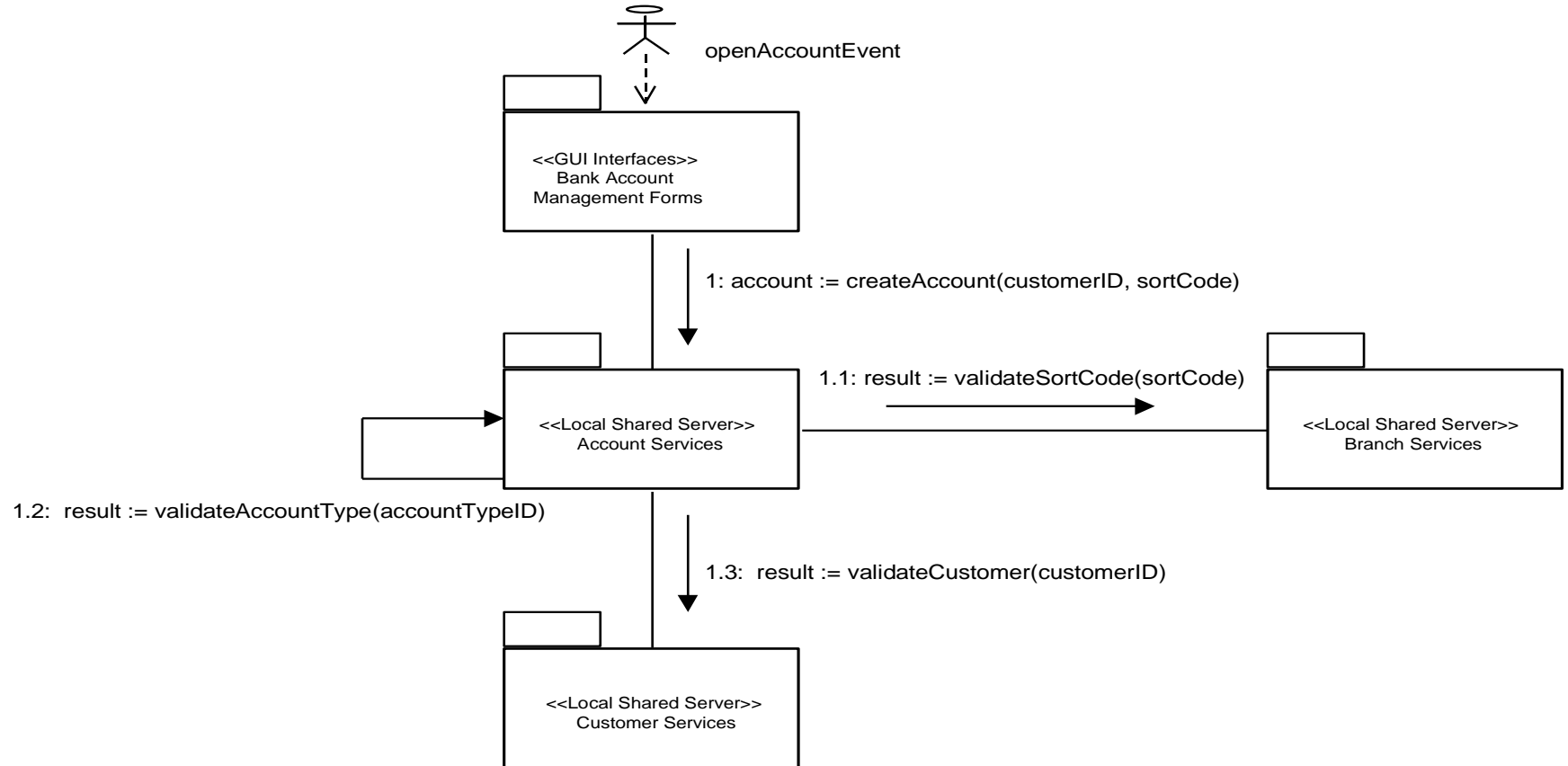


The State Transition / Data Lifecycle diagram is an essential part of managing business data throughout its lifecycle from conception until disposal within the constraints of the business process. The data is considered as an entity in its own right, decoupled from business process and activity. Each change in state is represented on the diagram which may include the event or rules that trigger that change in state. The separation of data from process allows common data requirements to be identified which enables resource sharing to be achieved more effectively.

Scenario Event Diagram – Open Account



Application Component Collaboration Diagram



Open Account System Operation Specification

Description: A new account record is created for a customer.

Reads: supplied customerId: Int
 supplied accountId: Int
 supplied sortCode: Int
 customer: CustomerInfo
 branch: Branch

Changes: new account: Account
 new returnMsg: String
 accountType: AccountType
 openAccountScreen: OpenAccountScreen

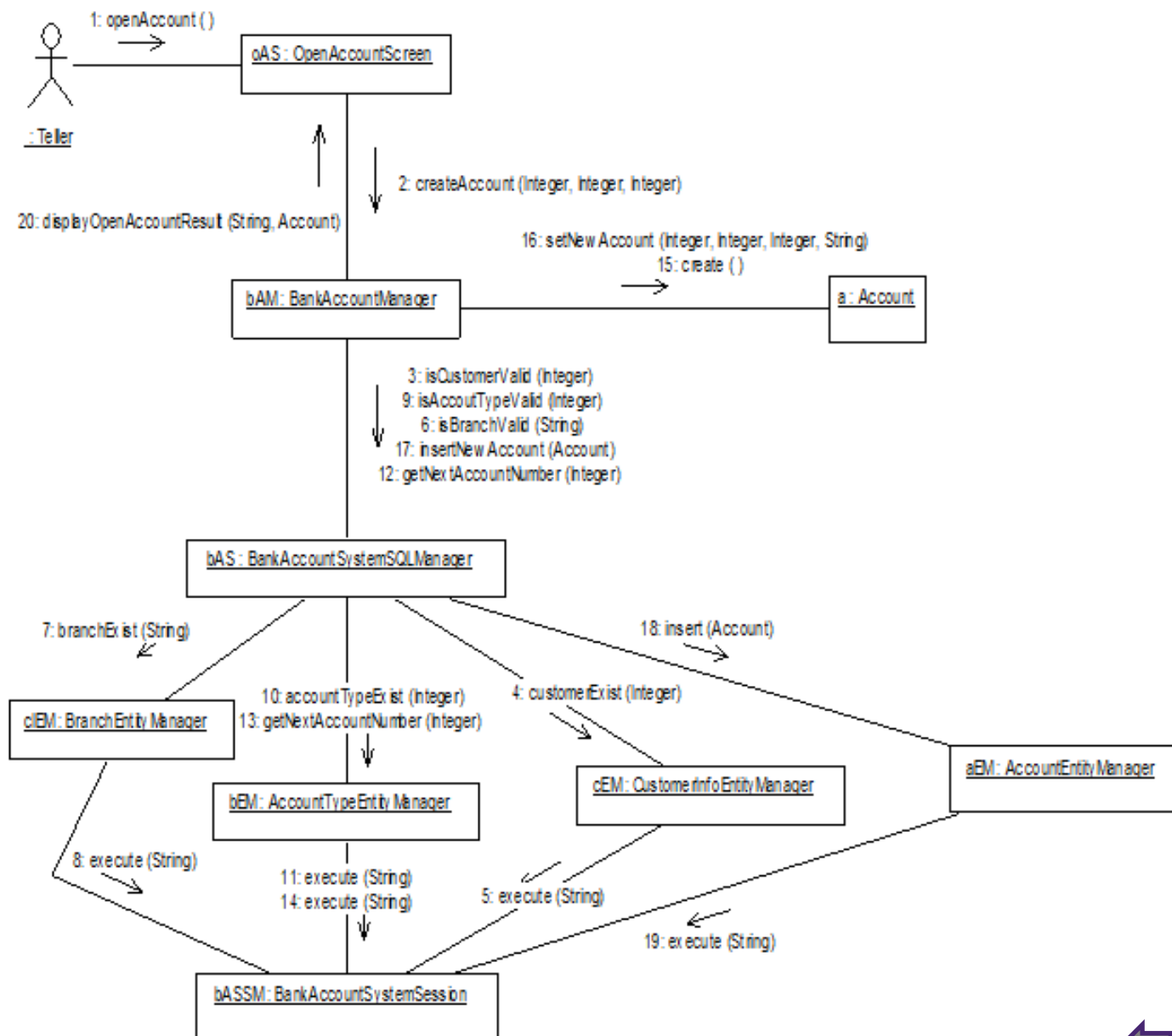
Sends:

Assumes:

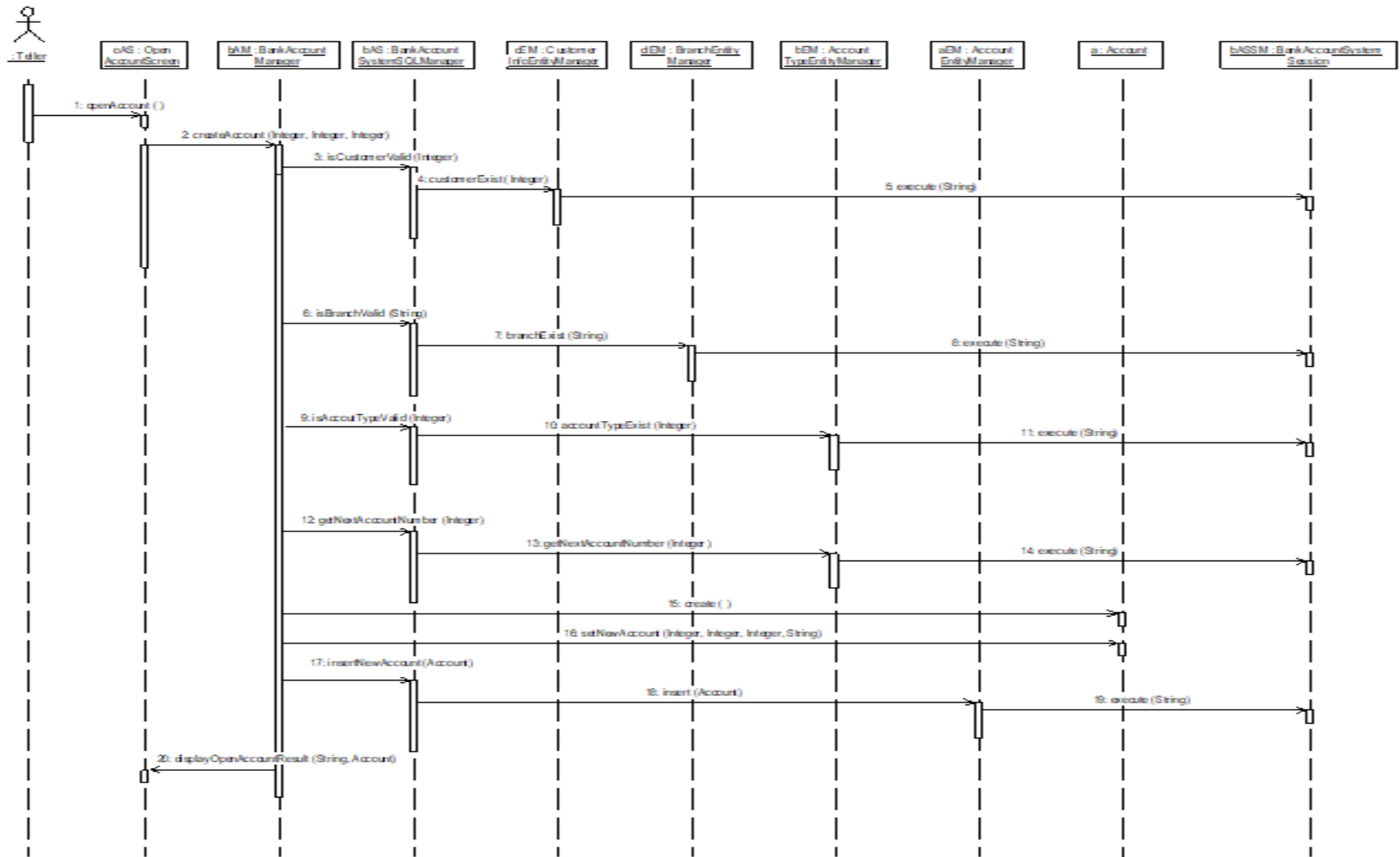
Results:

1. If there was no stored customer, for the supplied customerId then a returnMsg has been created, initialised with "Invalid Customer ID." and displayed on the openAccountScreen
1. If there was no stored branch, for the supplied sortCode then a returnMsg has been created, initialised with "Invalid Sort Code." and displayed on the openAccountScreen
1. If there was no stored accountType, for the supplied accountId then a returnMsg has been created, initialised with "Invalid Account Type." and displayed on the openAccountScreen
1. Otherwise, the stored accountType lastAccountNumber has been incremented by 1 and the new lastAccountNumber retrieved. A new account has been created for the supplied accountType, initialised with the retrieved accountNumber, today's date, and balance (set to 0) and stored. A returnMsg has been created and initialised with "Account <<accountNumber >> successfully Opened." and displayed on the openAccountScreen.

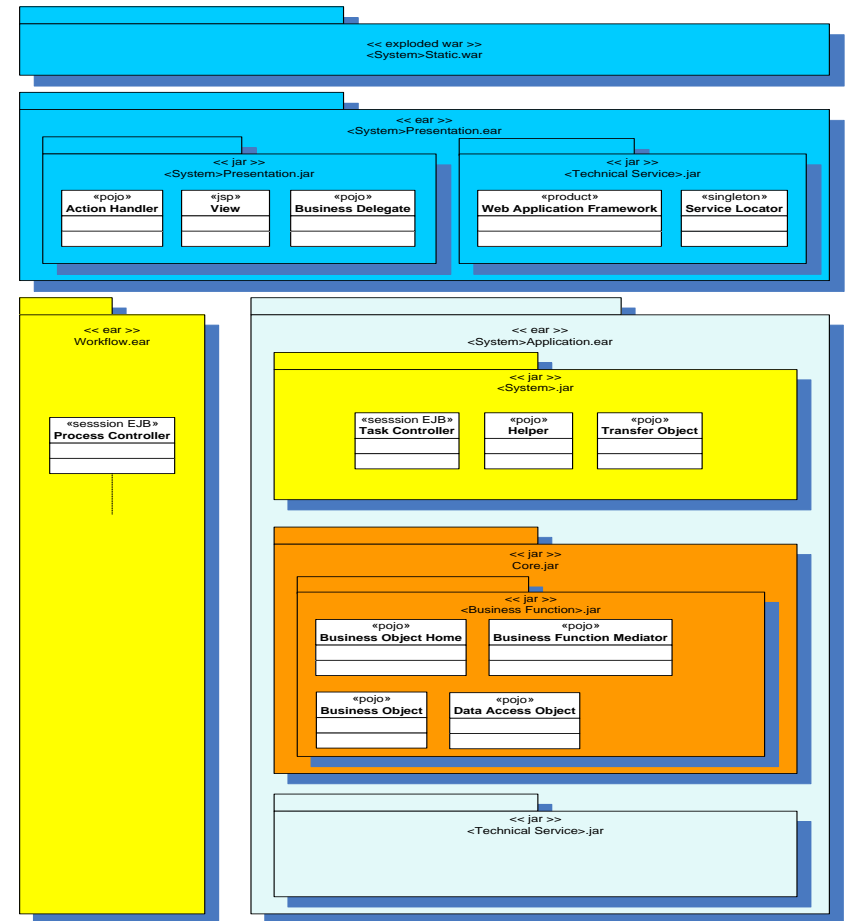
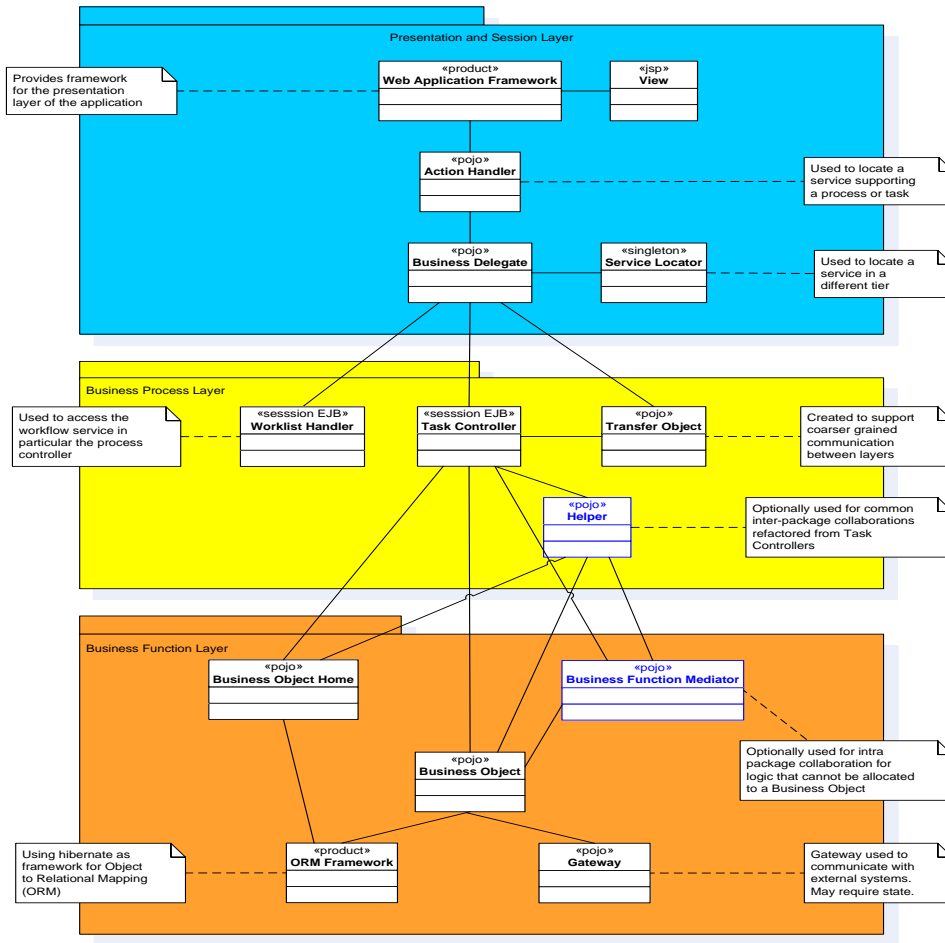
Open Account Communication Diagram



Open Account Interaction Diagram



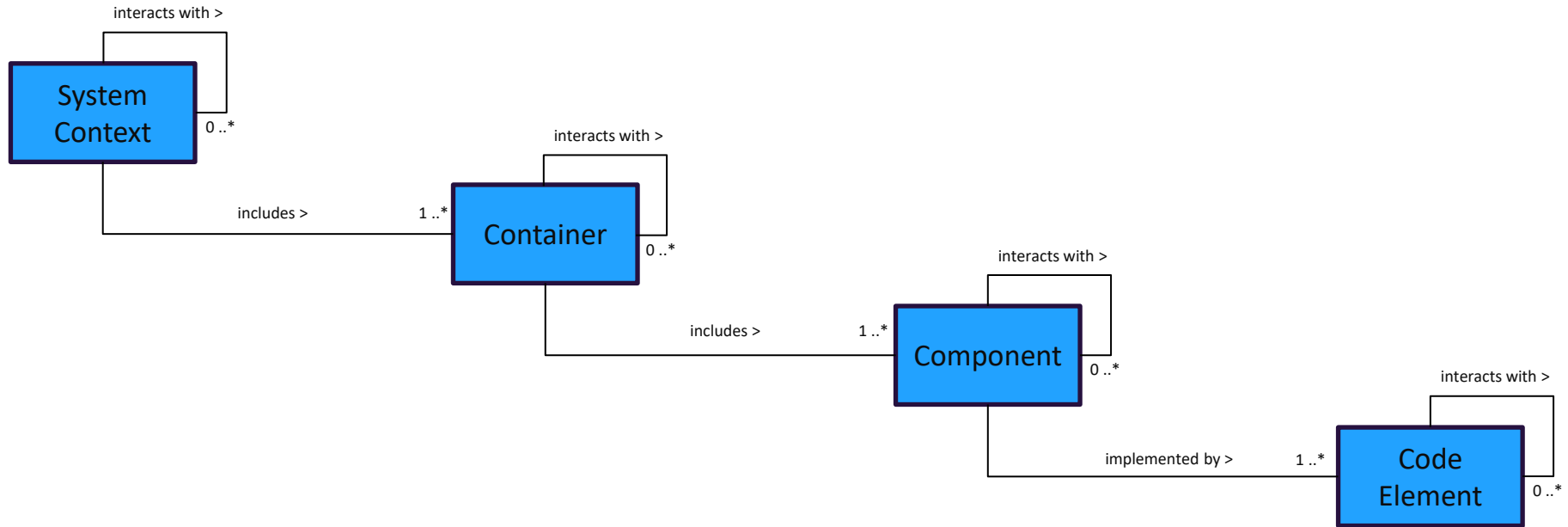
Application Communication Diagram



The purpose of the Application Communication diagram is to depict all models and mappings related to communication between applications in the metamodel entity. It shows application components and interfaces between components. Interfaces may be associated with data entities where appropriate. Applications may be associated with business services where appropriate. Communication should be logical and should only show intermediary technology where it is architecturally relevant.



C4 System - Software Architecture Viewpoint

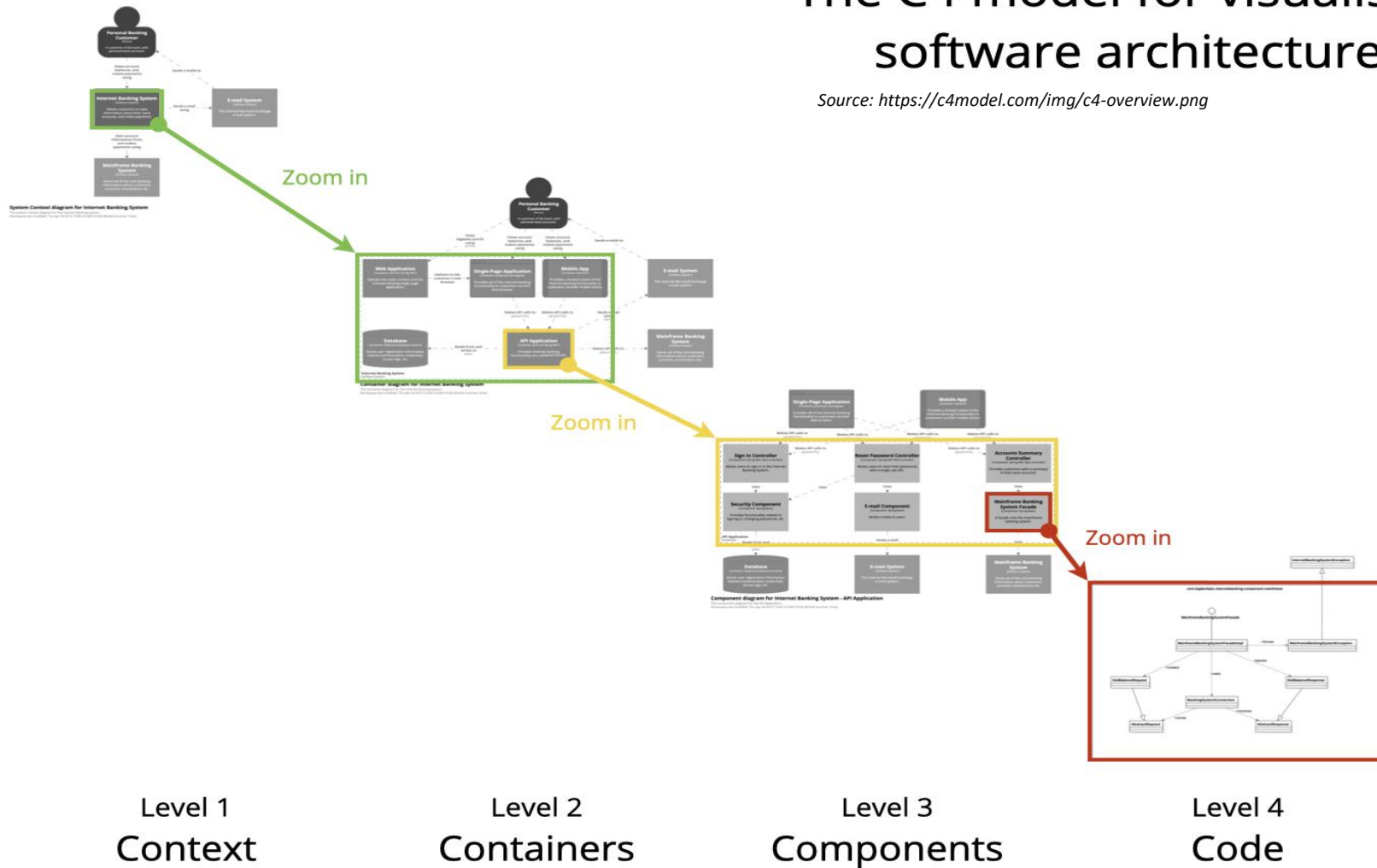


A **software system** is made up of one or more **containers** (applications and data stores), each of which contains one or more **components**, implemented by one or more **code elements** (classes, interfaces, objects, functions, etc.) & **people** may use the software systems that we build.

C4 System - Software Architecture Viewpoint

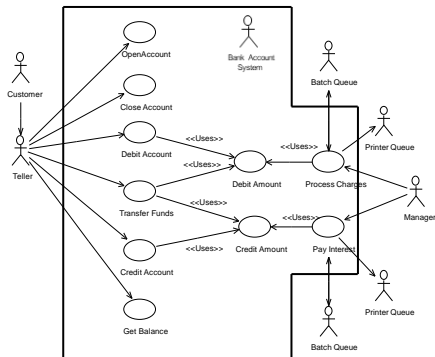
The C4 model for visualising software architecture

Source: <https://c4model.com/img/c4-overview.png>



C4 System – Software Approach – Example Similar Model Using UML

System Context



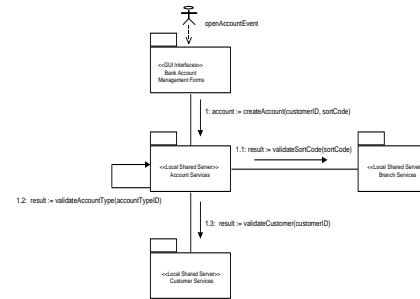
Use Case Context

Container



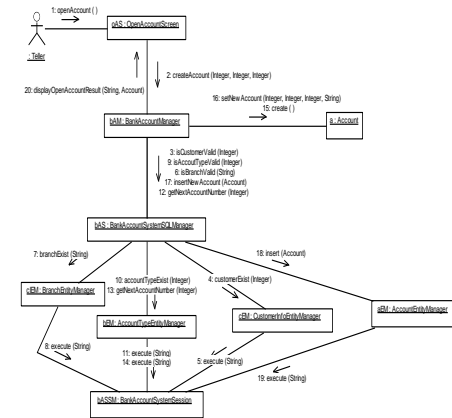
Generic Package Collaboration

Component



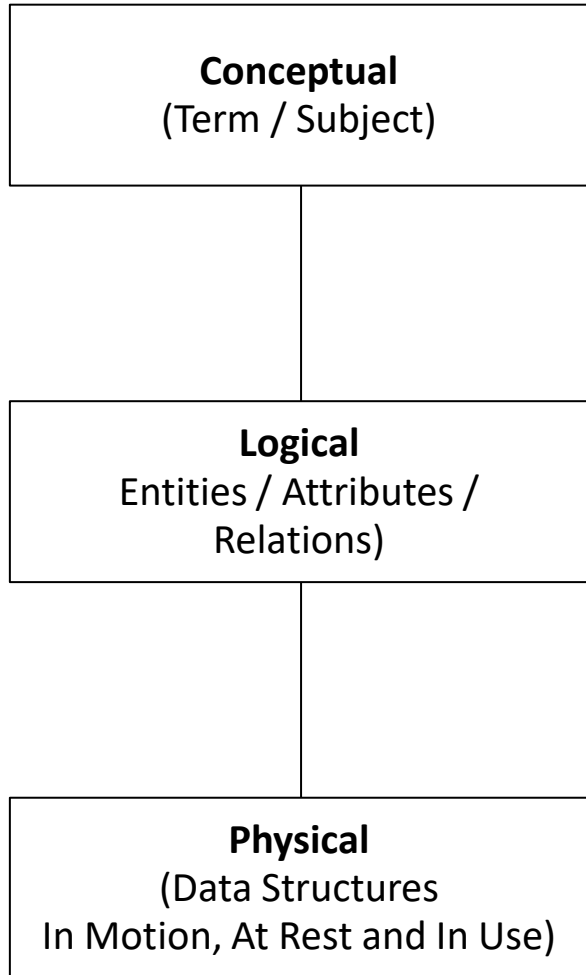
Package Event Collaboration

Code Element

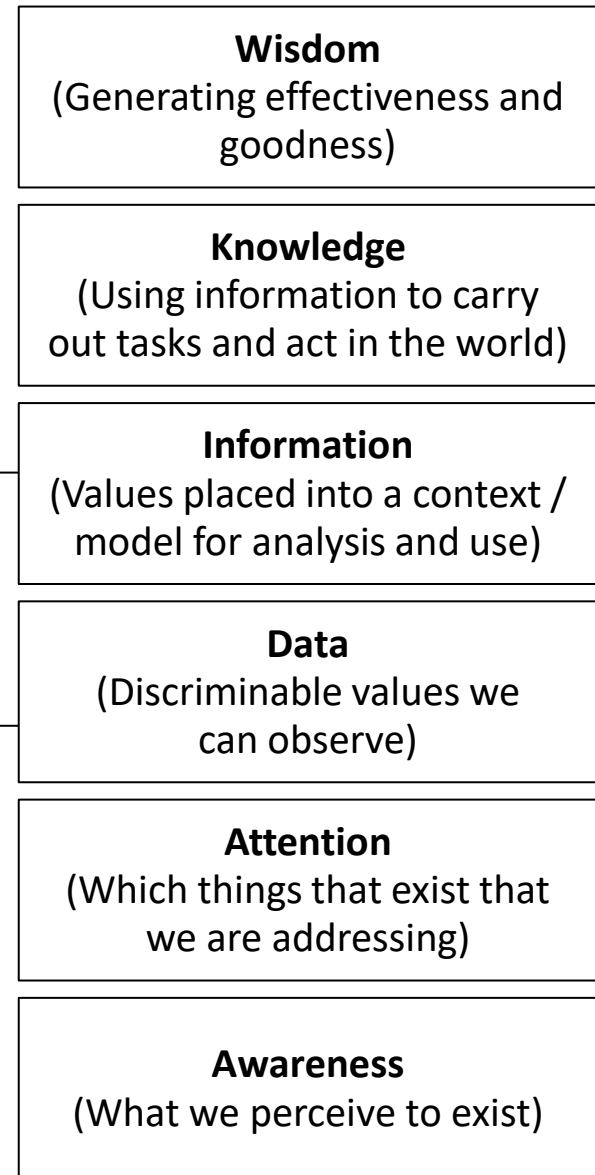


Event Code Element Interaction

Information / Data Organisation Hierarchies



BCS talks
about
these two



Pessimistic and Optimistic Patterns Of Interoperability

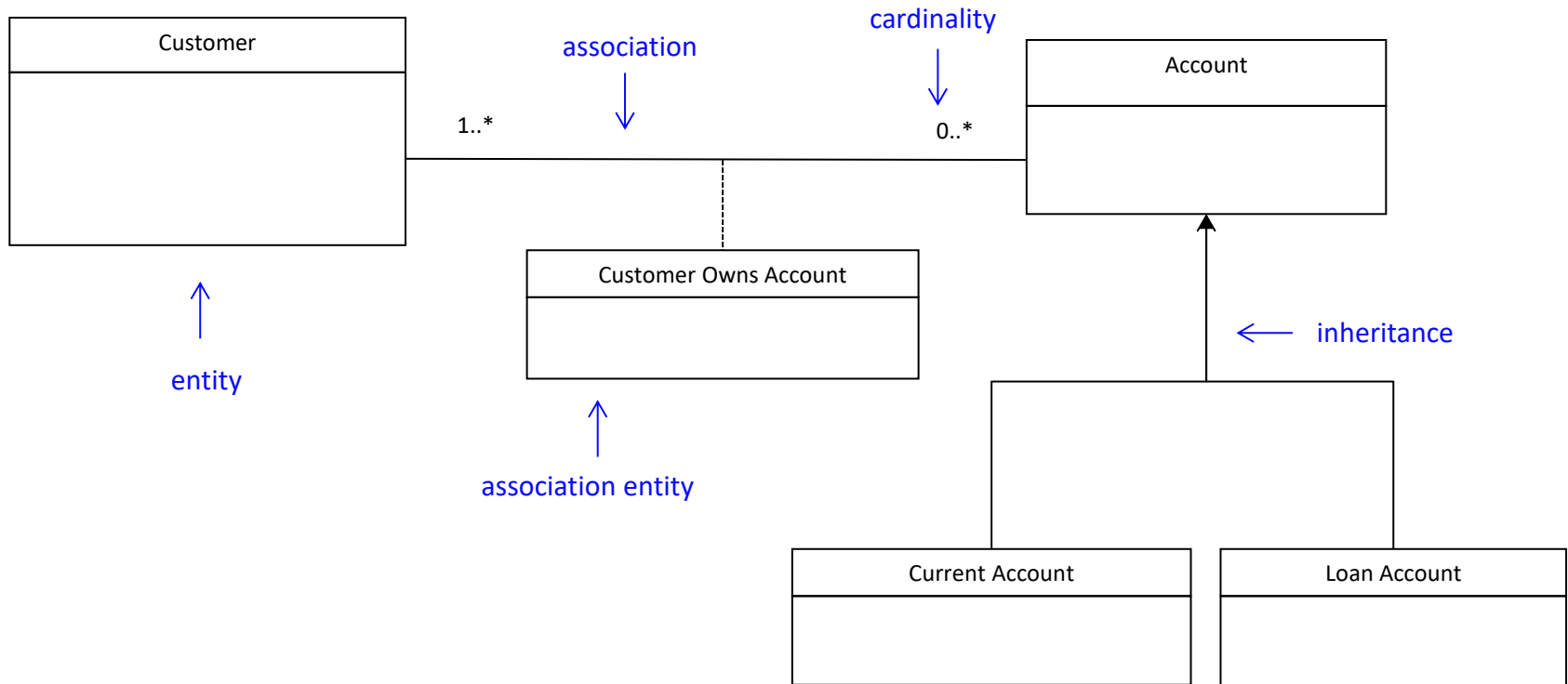
ACID (**A**tomic / **C**onsistent / **I**solated / **D**urable)

- Control / predictability / consistency
- Consistency is created beforehand ensuring the operations follow the rules.
- Planning across resource managers and explicit management of dependencies.

BASE (**B**asic **A**vailability / **S**oft State / **E**ventual Consistency)

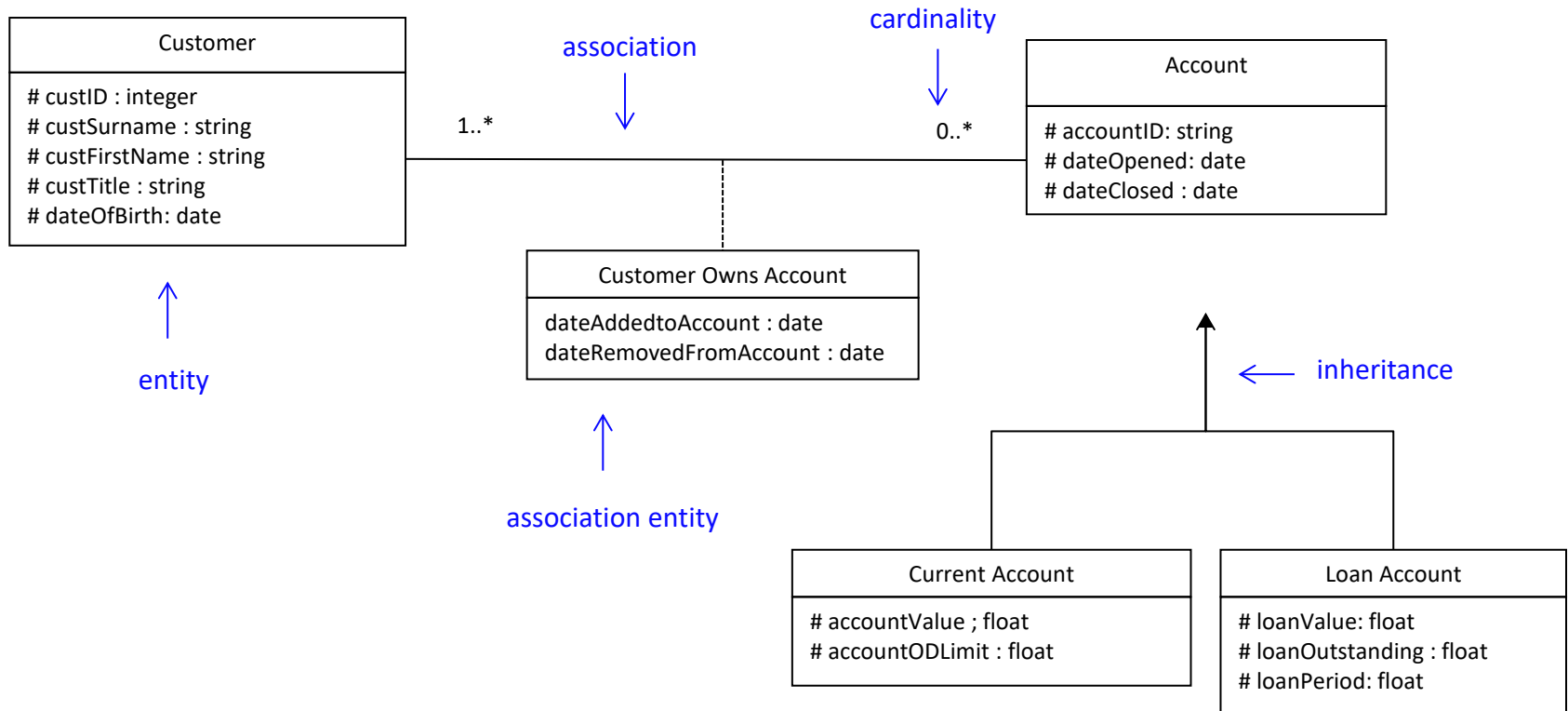
- Flexibility / simplicity / speed
- Consistency created afterwards as a result of the best efforts if any compensation is required (which may not fully address any problems).
- Real time discovery of dependencies and limited planning across resources managers.

Information Model / Conceptual Data (Entity - Relationship) Diagram



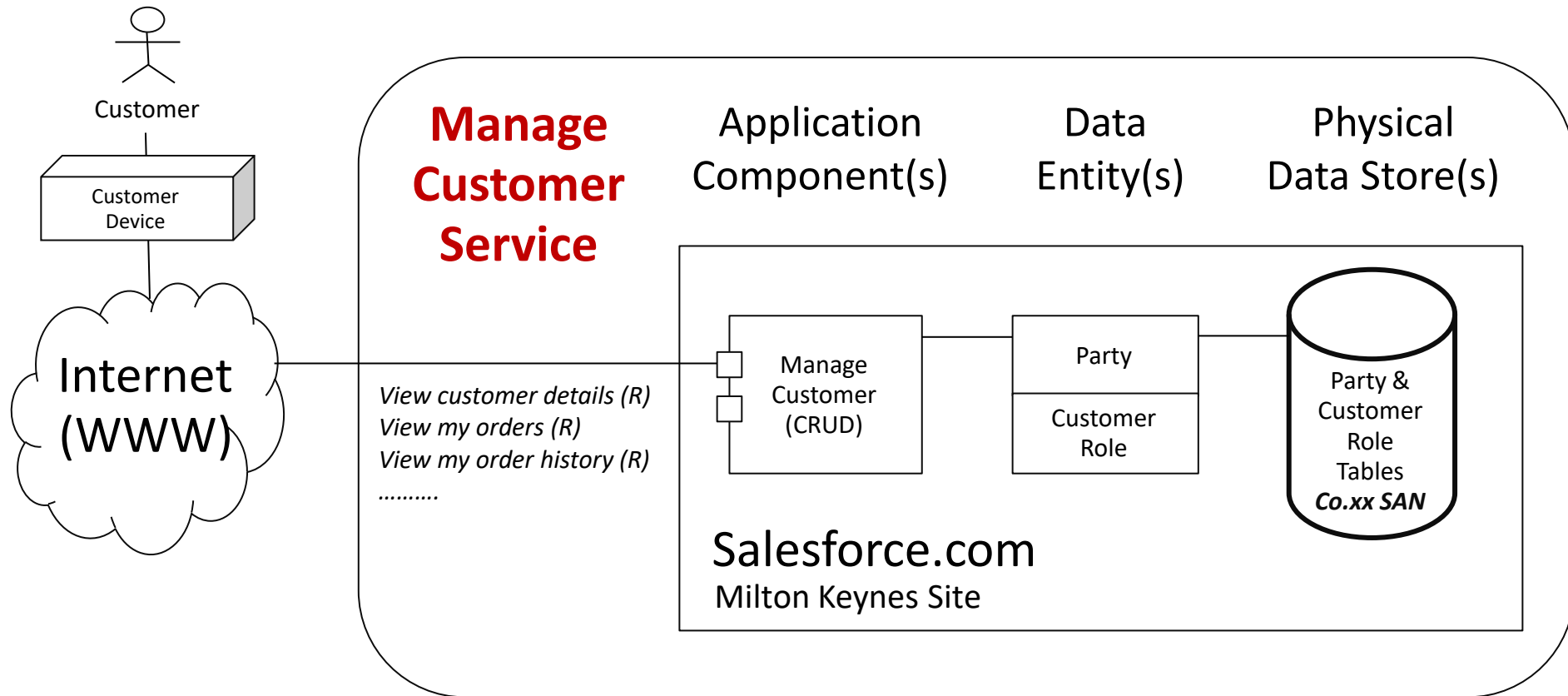
The key purpose of the Conceptual Data diagram is to depict the relationships between critical data entities within the enterprise.

Logical Data (Entity - Relationship) Diagram



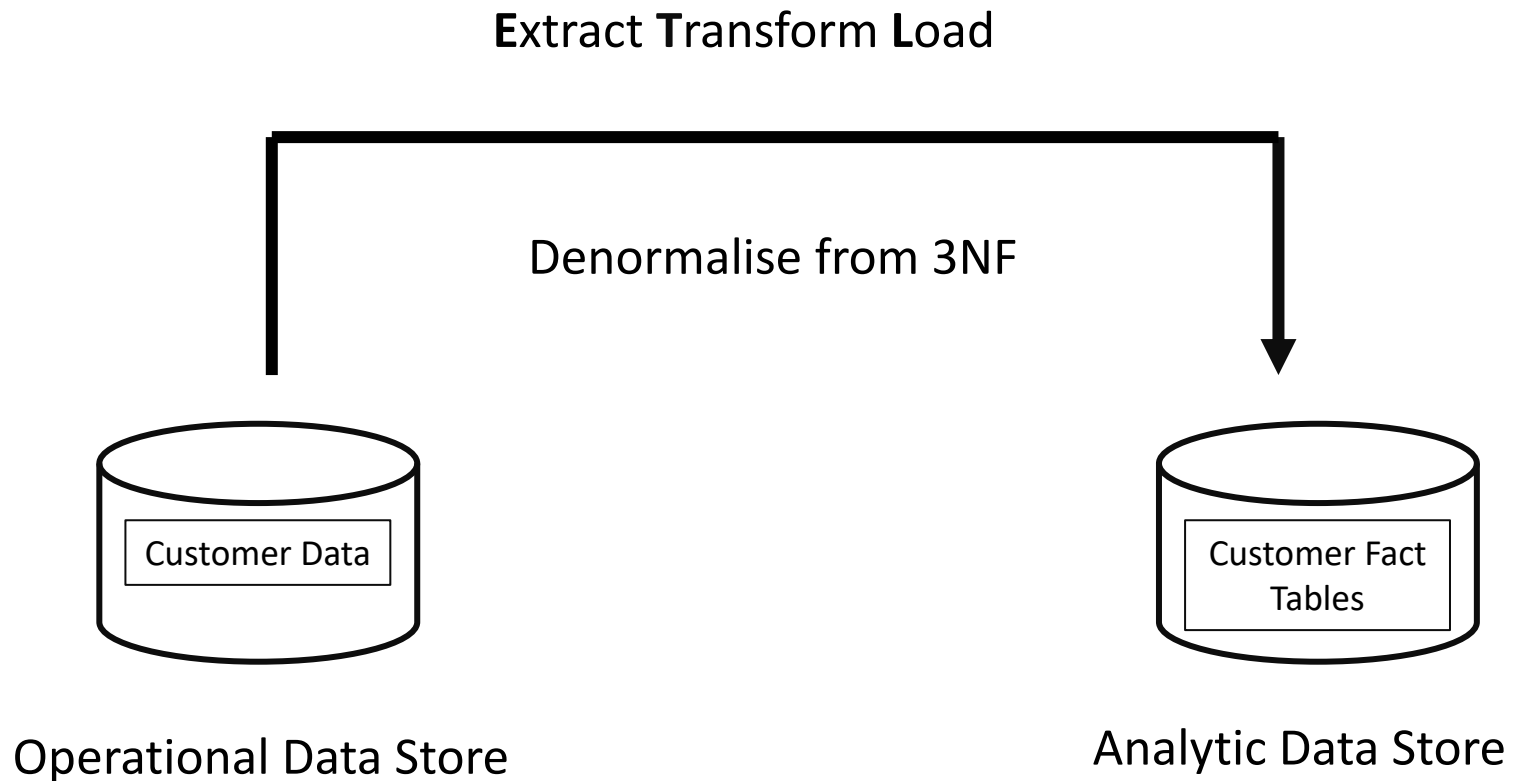
The key purpose of the Logical Data diagram is to show logical views of the relationships between critical data entities within the enterprise.

Physical Data Dissemination Diagram



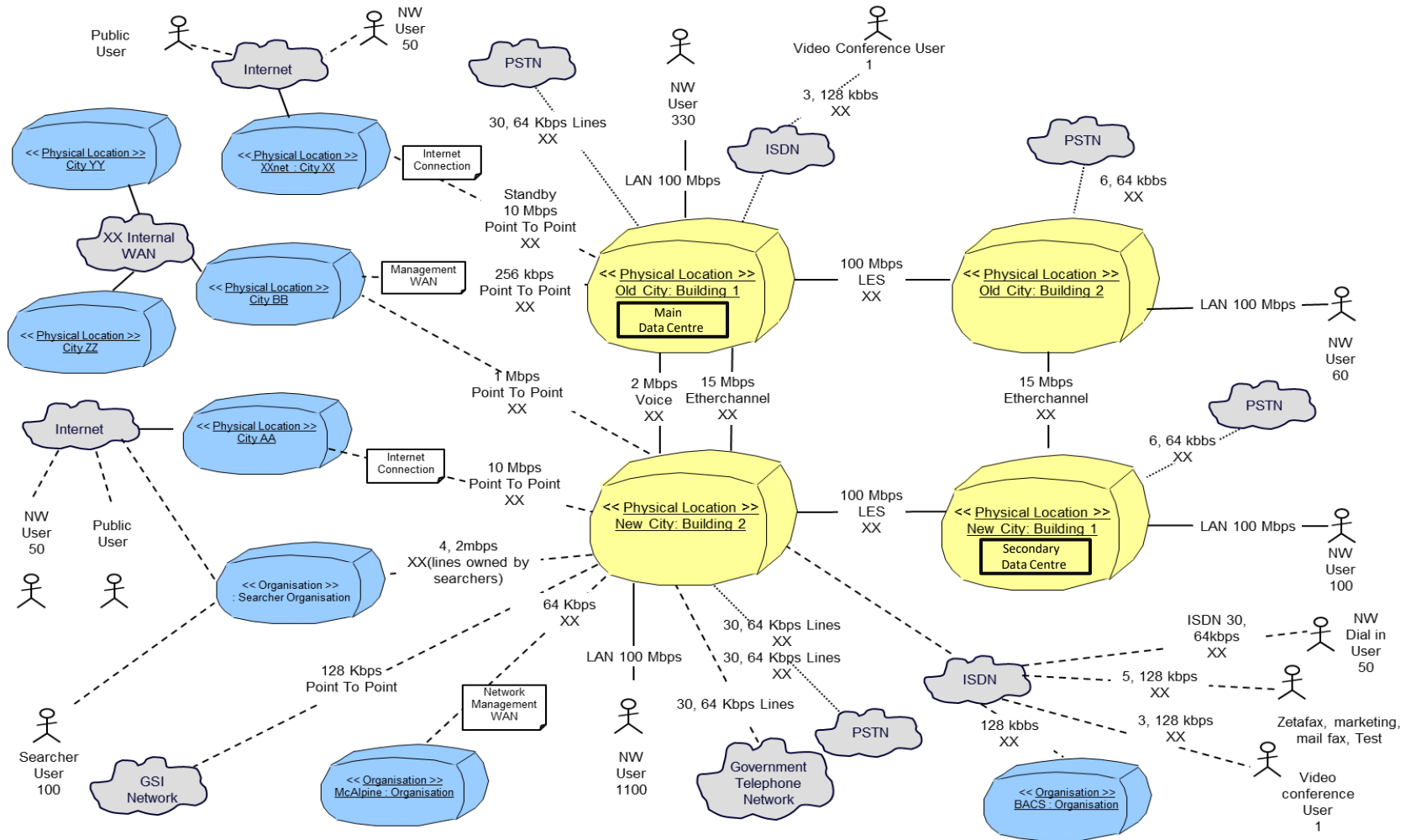
The purpose of the Data Dissemination diagram is to show the relationship between data entity, business service, and application components. The diagram shows how the logical entities are to be physically realized by application components. This allows effective sizing to be carried out and the IT footprint to be refined. Moreover, by assigning business value to data, an indication of the business criticality of application components can be gained. Additionally, the diagram may show data replication and application ownership of the master reference for data. In this instance, it can show two copies and the master-copy relationship between them. This diagram can include services; that is, services encapsulate data and they reside in an application, or services that reside on an application and access data encapsulated within the application.

Data Migration Diagram



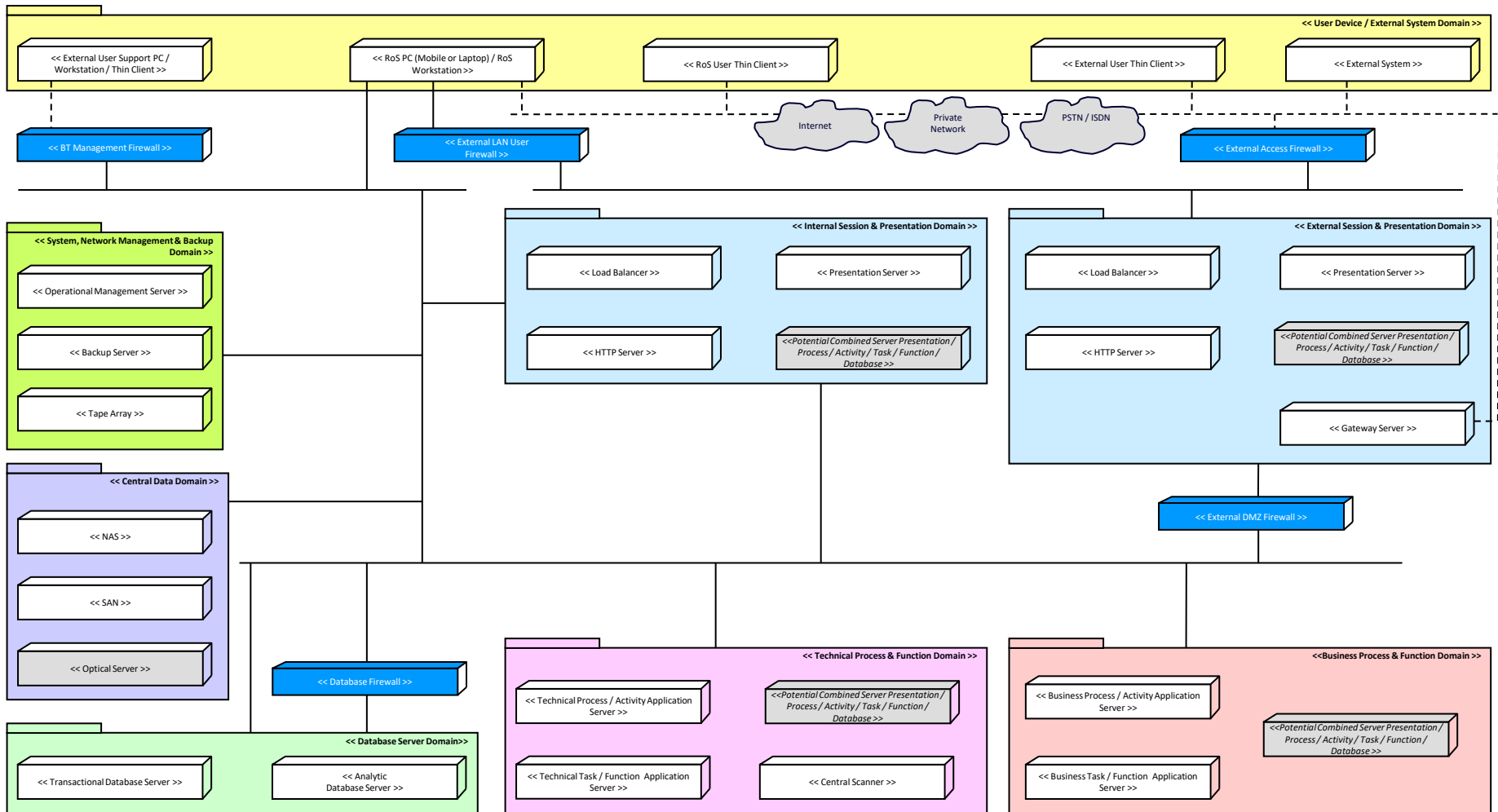
Data migration is critical when implementing a package or packaged service-based solution. This is particularly true when an existing legacy application is replaced with a package or an enterprise is to be migrated to a larger package/package services footprint. Packages tend to have their own data model and during data migration the legacy application data may need to be transformed prior to loading into the package.

Environments and Locations Diagram (Top Level)



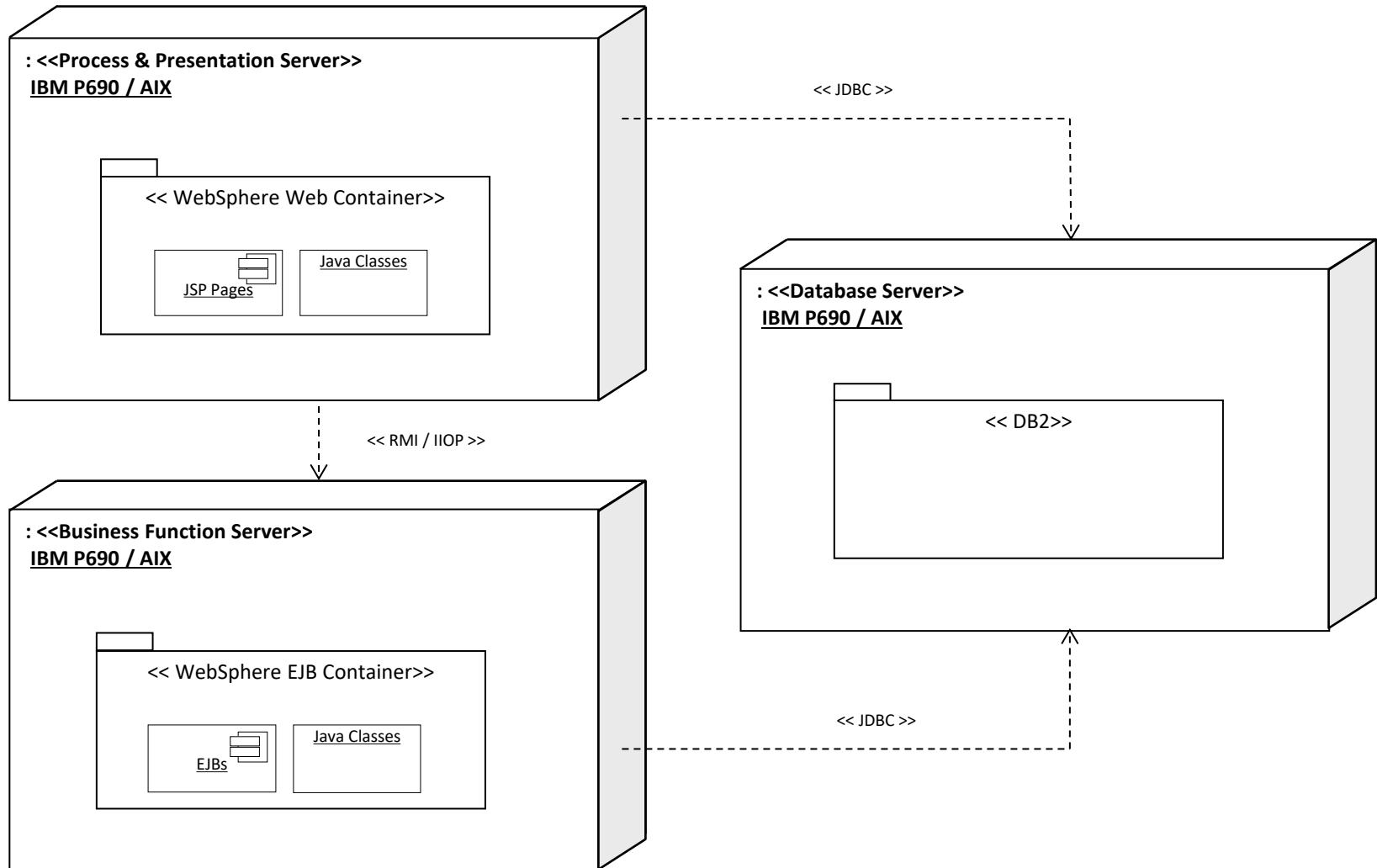
The Environments and Locations diagram depicts which locations host which applications, identifies what technologies and/or applications are used at which locations, and finally identifies the locations from which business users typically interact with the applications. This diagram should also show the existence and location of different deployment environments, including non-production environments, such as development and pre-production.

Logical Data Centre Diagram

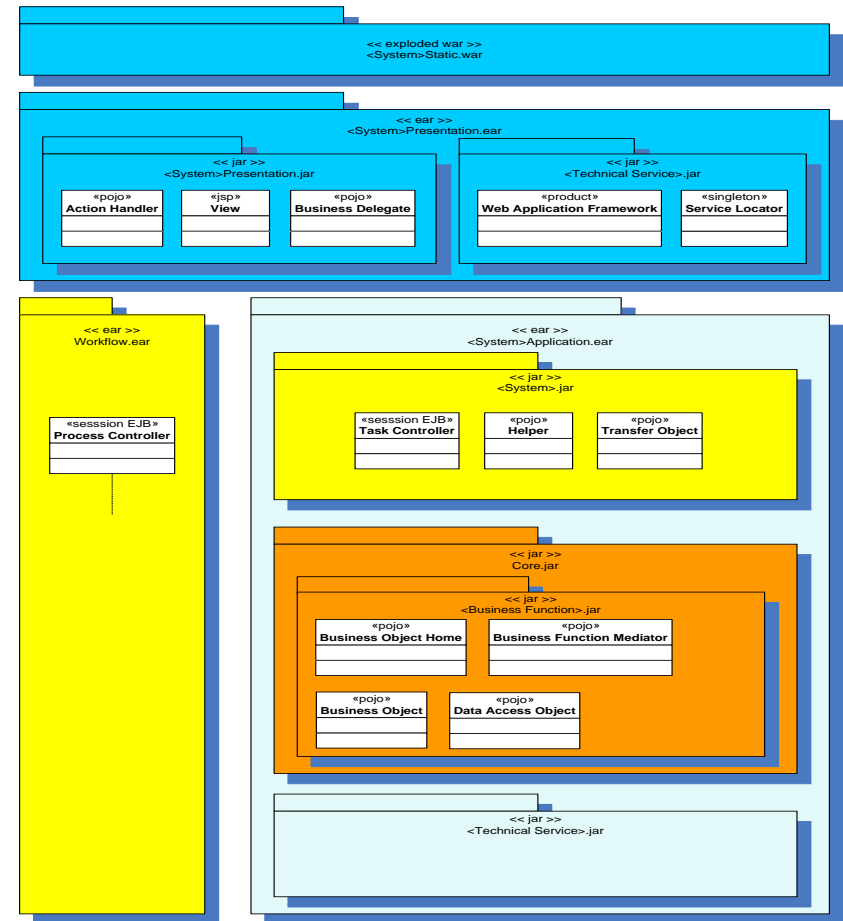
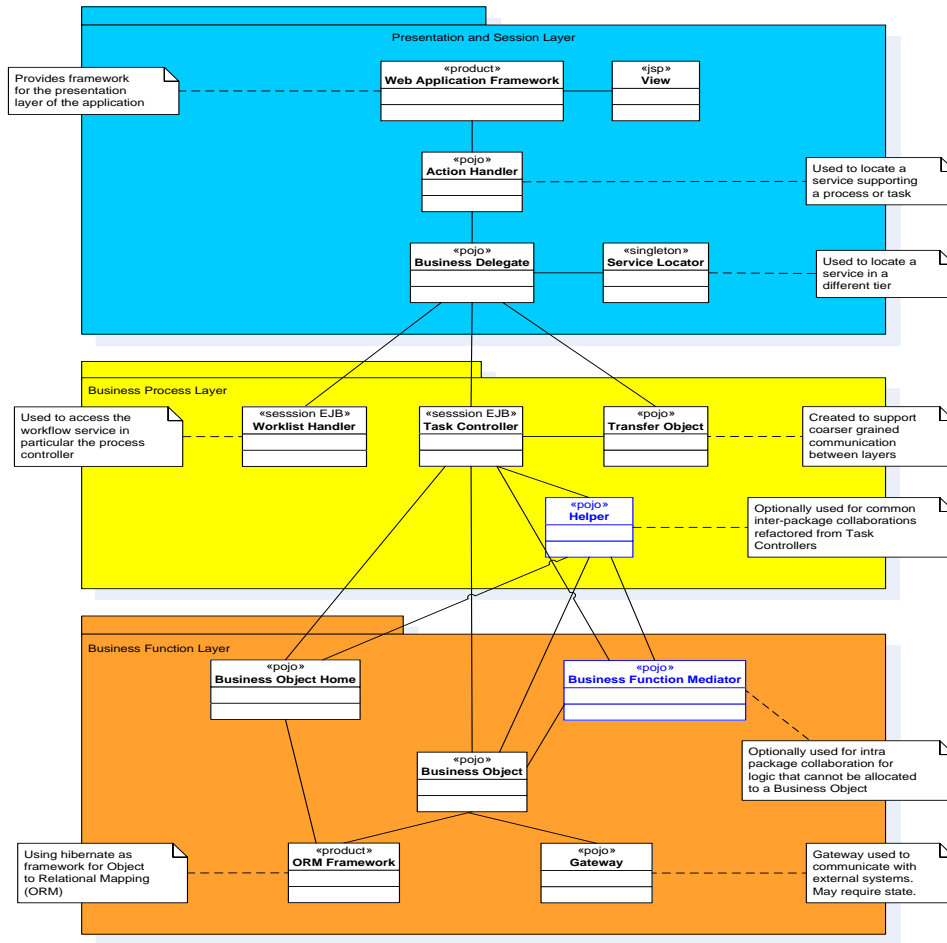


The Software Distribution diagram shows how application software is structured and distributed across the estate. It is useful in systems upgrade or application consolidation projects. This diagram shows how physical applications are distributed across physical technology and the location of that technology. This enables a clear view of how the software is hosted, but also enables managed operations staff to understand how that application software is maintained once installed.

Deployment diagrams

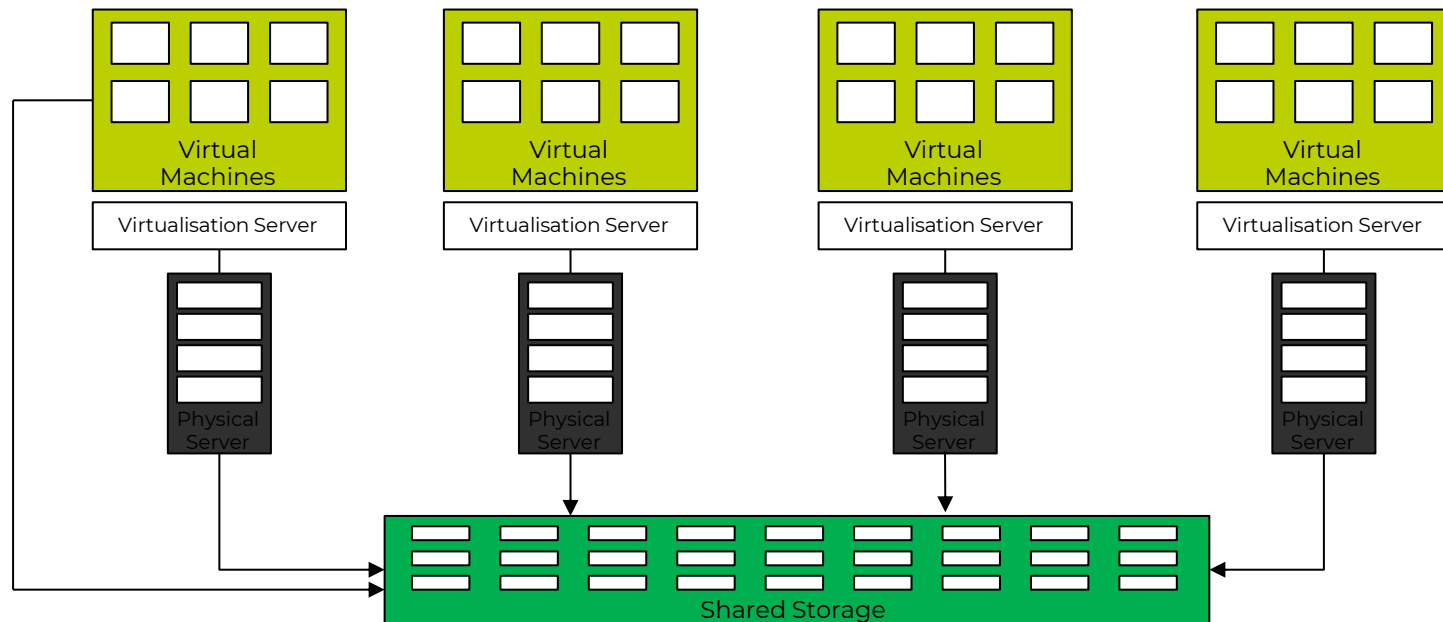


Application Packages To Be Deployed In Containers On Devices



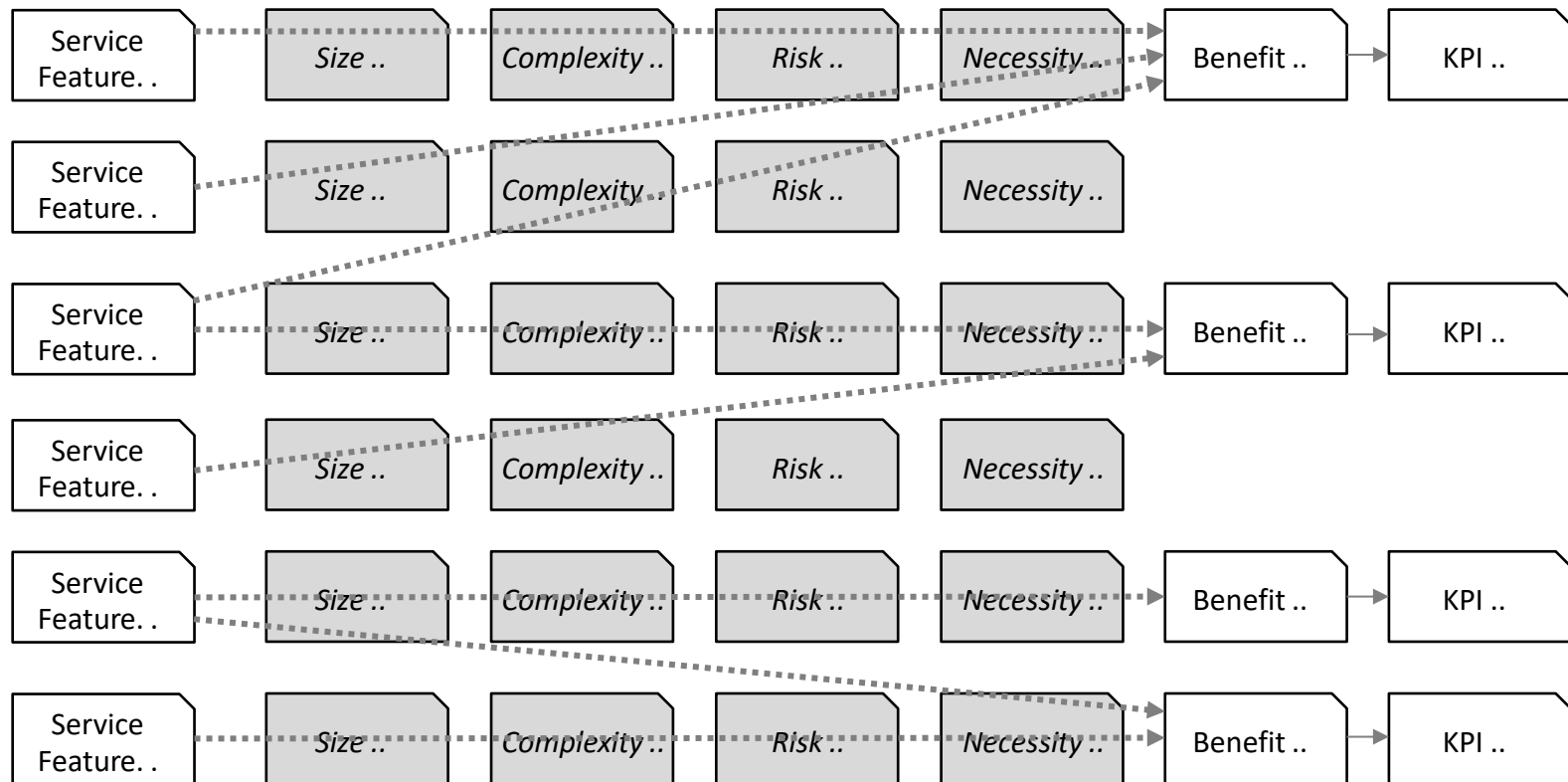
The purpose of the Application Communication diagram is to depict all models and mappings related to communication between applications in the metamodel entity. It shows application components and interfaces between components. Interfaces may be associated with data entities where appropriate. Applications may be associated with business services where appropriate. Communication should be logical and should only show intermediary technology where it is architecturally relevant.

Technology Solution Patterns show how infrastructure interacts to provide an effective execution and managed environment.
Example: Virtualisation Pattern



Roadmaps & Conformance Checkpoints

Example Benefits Diagram



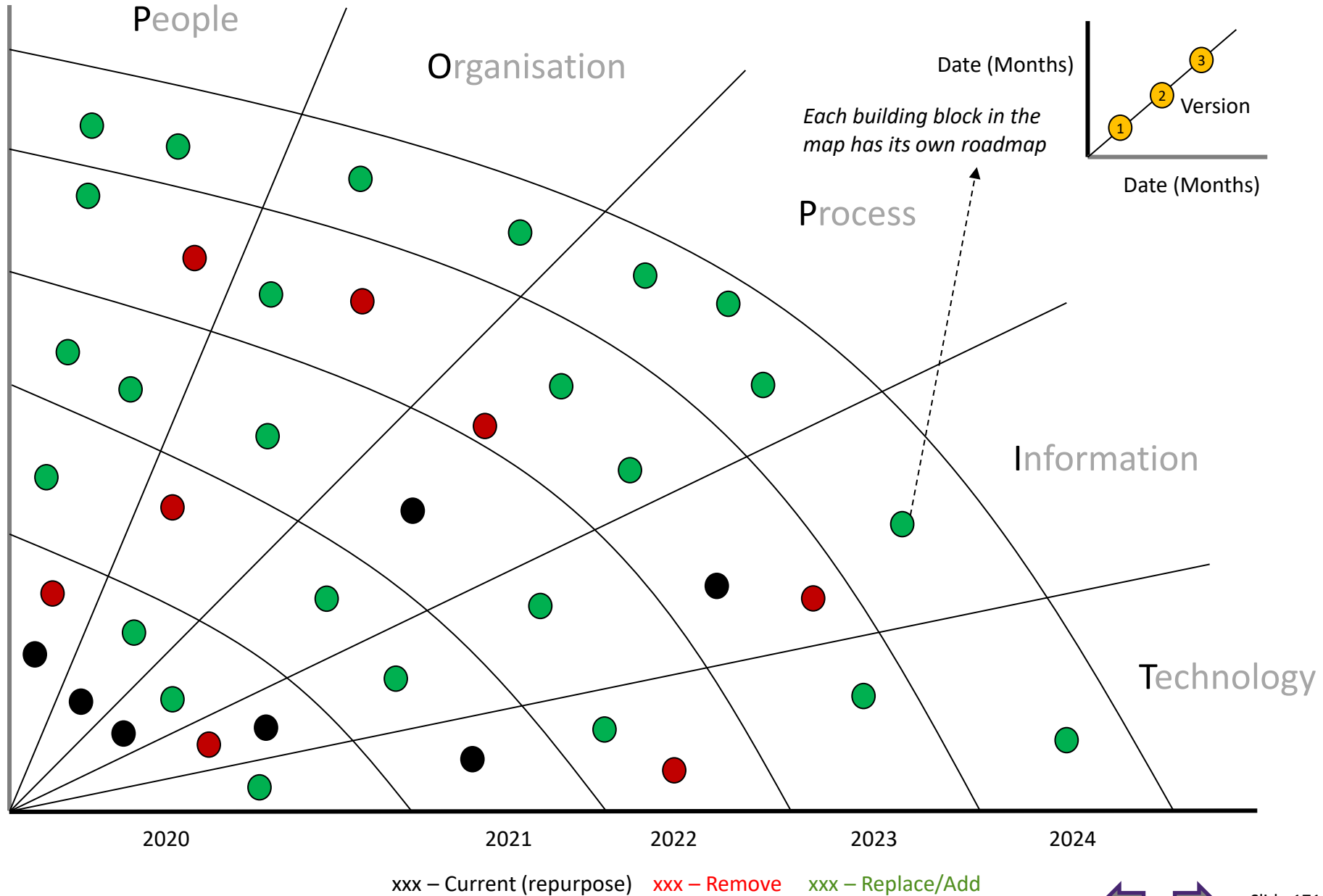
The Benefits diagram shows opportunities identified in an architecture definition, classified according to their relative size, benefit, and complexity. This diagram can be used by stakeholders to make selection, prioritization, and sequencing decisions on identified opportunities.

Example Roadmap - Timeline Format

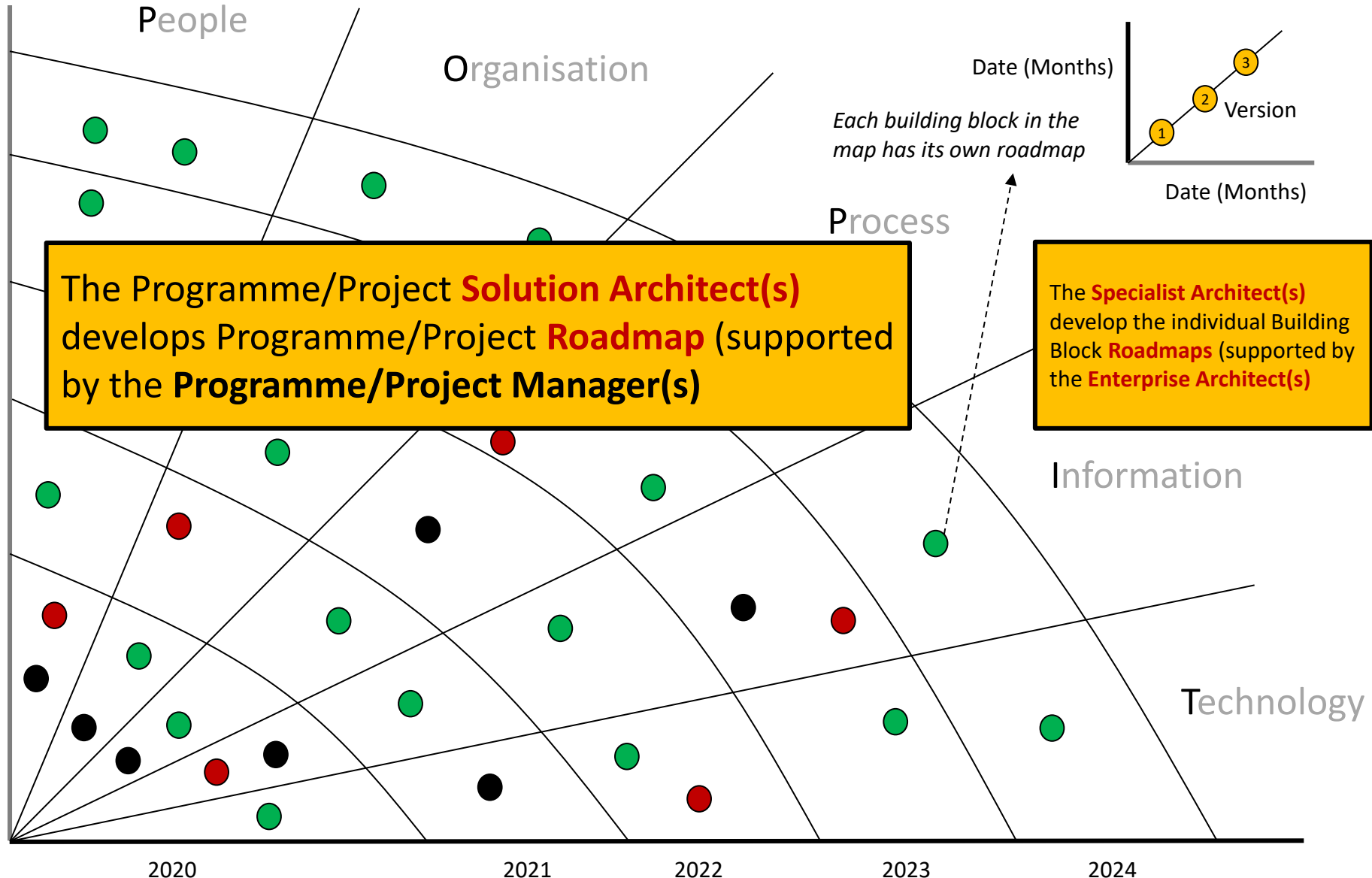
The roadmaps show the planned evolution of solutions over a period of time



Example Roadmap - Transformation Roadmap Format



Example Roadmap - Transformation Roadmap Owners

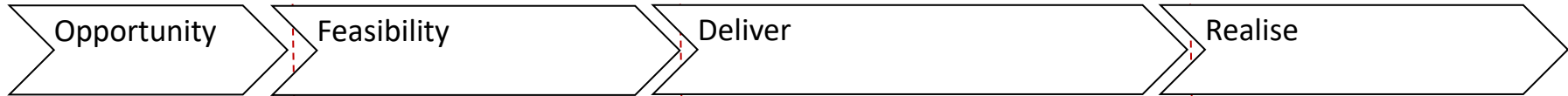


xxx – Current (repurpose) xxx – Remove xxx – Replace/Add

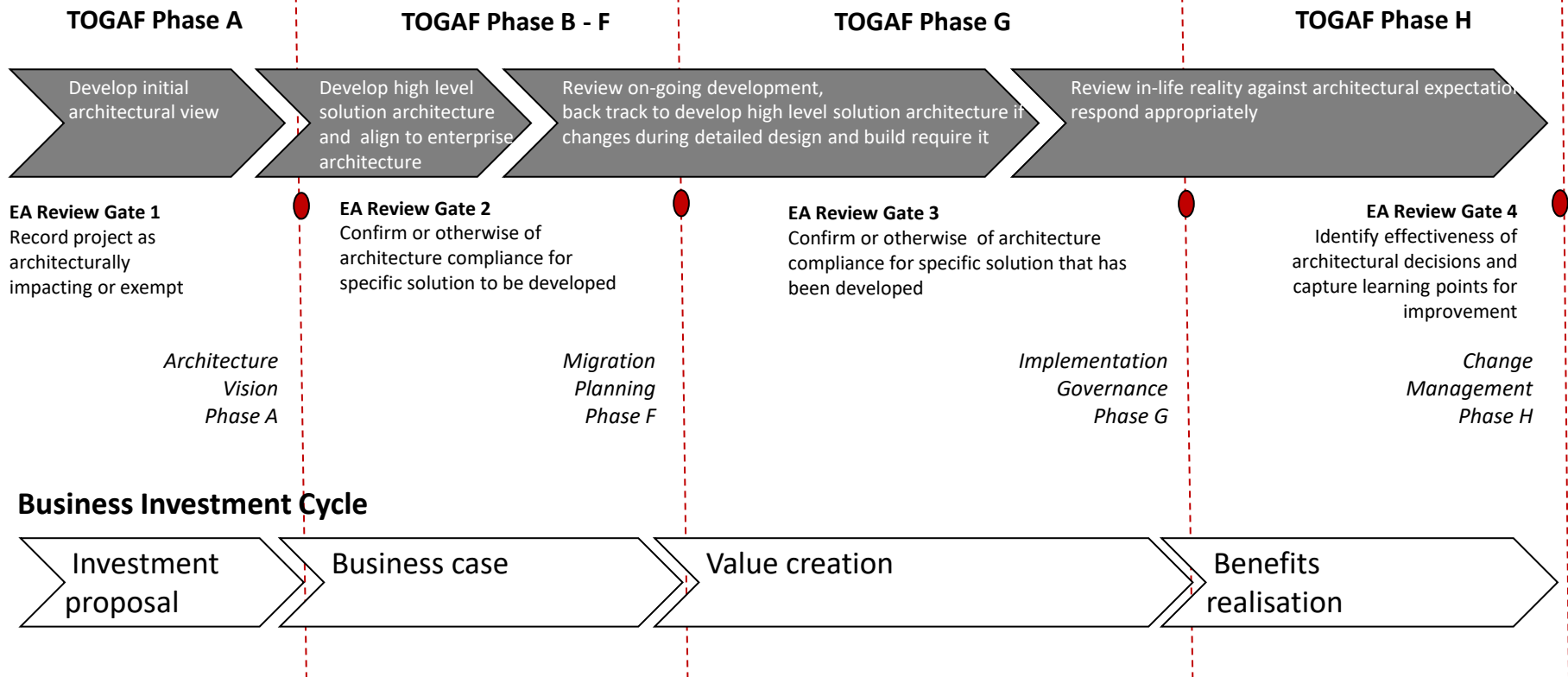


Real World Architecture Conformance Checkpoints

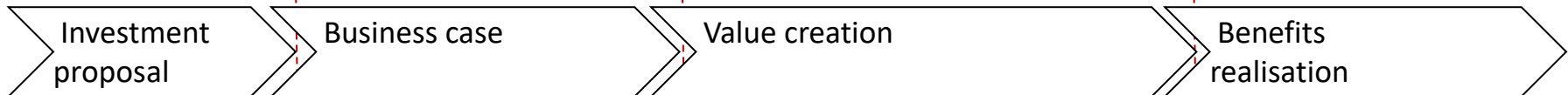
Project Management Cycle



Enterprise Architecture Conformance Cycle



Business Investment Cycle



The Balance Between Planning and Discovery

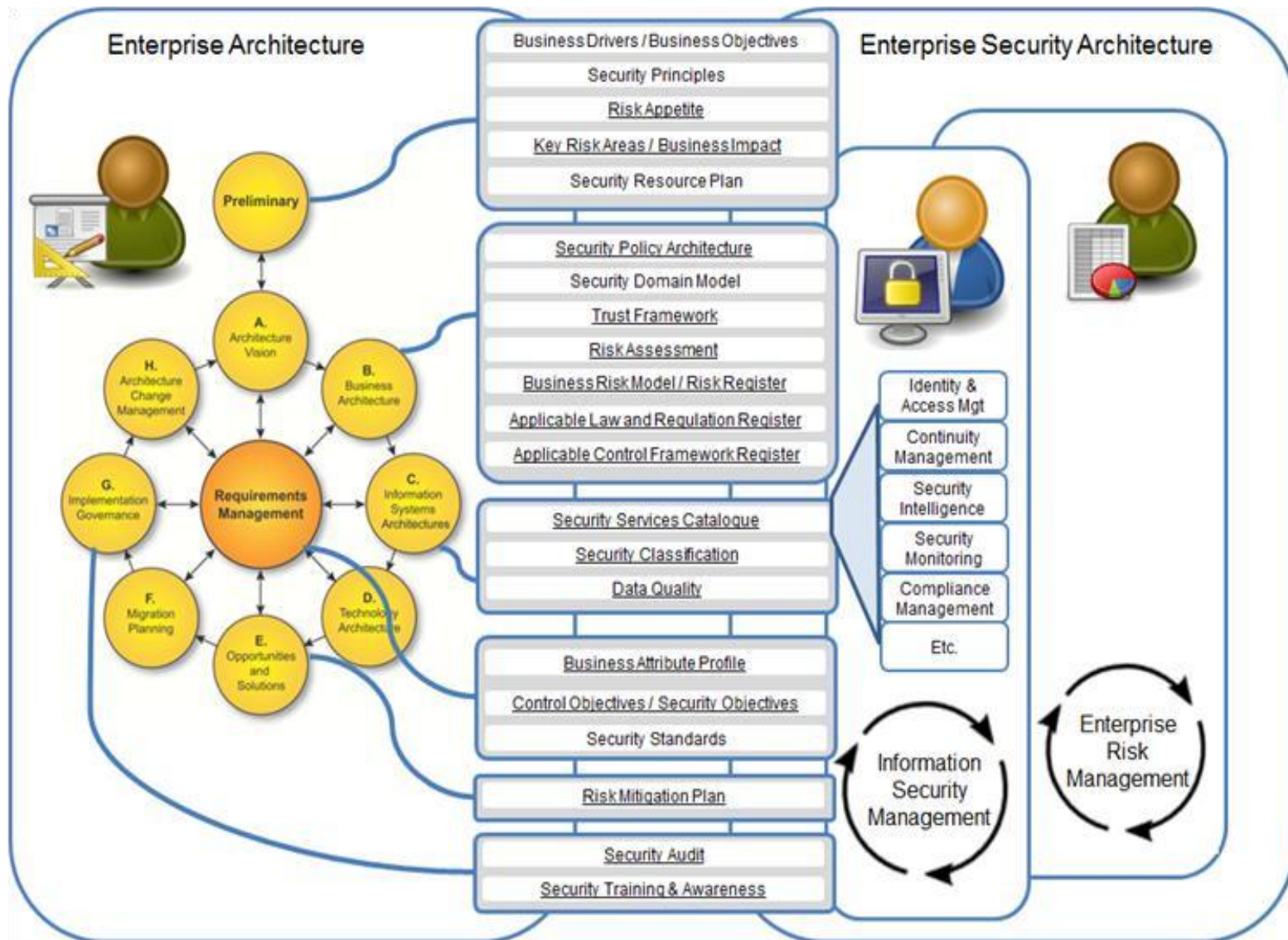
Why do we often need some level of planning and intentionality?

There may be many levels of constraint across building blocks and systems imposed by societal and organisation rules and regulations that must be planned and accounted for; as well as specific requirements for capability.

These may often be in contention (especially in multi-national, multi-product, multi-location enterprises).

Change Approach	
Intentional (Planning Practices)	Emergent (Discovery Practices)
<p><i>Stronger Coupling</i></p> <ul style="list-style-type: none"> • More planning, • Early global testing, • More early integration, • Larger and more varied systems and sub-systems, • More integrity & consistency. 	<p><i>Dimension Looser</i></p> <ul style="list-style-type: none"> • Less planning, • Early localised testing, • More late integration, • Smaller systems and sub-systems, • Less integrity & consistency.
<p><i>More Complex Cohesion</i></p>	<p><i>Dimension Simpler</i></p>

TOGAF Security Architecture View



Information Security Management

Information Security Management (ISM) is a process that defines the security objectives, assigns ownership of information security risks, and supports the implementation of security measures.

The security management process includes risk assessment, the definition and proper implementation of security measures, reporting about security status (measures defined, in place, and working), and the handling of security incidents.

According to ISO/IEC 27001:2013 [4], the ISM system preserves the security aspects of information by applying a risk management process, and it gives confidence to interested parties that risks are adequately managed.

The ISM system is part of and is integrated with the organization's processes and overall management structure. The standard specifies the requirements for the ISM system.

Enterprise Risk Management

The Information Technology security and information security industry has evolved over its lifetime a view of operational risk that is concerned only with threats, vulnerabilities, and loss events (negative impacts).

However, as noted earlier in Section 1.2, this Guide uses the ISO 31000:2009 [6] definition of “risk”, an “uncertainty of outcomes”, and risk management is presented as striking a balance between positive and negative outcomes resulting from the realization of either opportunities or threats.

SABSA Elements - 1

SABSA (Preliminary Elements)

Sherwood Applied Business Security Architecture

- Business Drivers/Business Objectives
- Security Principles
- Risk Appetite
- Key Risk Areas/Business Impact Analysis
- Security Resource Plan

SABSA (Architecture Vision Elements)

Sherwood Applied Business Security Architecture

- Satisfy the security stakeholders that the end-state does not represent any unknown or unacceptable risk and aligns with corporate policies, standards, and principles
- Satisfy business stakeholders - in particular those who control the budget - that the Security Architecture is instrumental in enabling and supporting the overall architecture required to deliver the business opportunities and benefits identified with the right balance between risk, compliance, and business benefits

SABSA (Requirements Elements)

Sherwood Applied Business Security Architecture

- Business Attribute Profile
- Control Objectives/Security Objectives
- Security Standards

SABSA (Business Architecture Elements)

Sherwood Applied Business Security Architecture

- Security Policy Architecture
- Security Domain Model
- Trust Framework
- Risk Assessment
- Business Risk Model/Risk Register
- Applicable Law and Regulation Register
- Applicable Control Framework Register

SABSA (IS Architecture Elements)

Sherwood Applied Business Security Architecture

- Security Services Catalogue
- Security Classification
- Data Quality

SABSA (Opportunities & Solutions Elements)

Sherwood Applied Business Security Architecture

- Risk Mitigation Plan

SABSA (Migration Planning Elements)

Sherwood Applied Business Security Architecture

- Migration is itself a business process that needs to be secured. The migration strategy should include a risk assessment and a Risk Mitigation Plan. In Phase F, the Risk Mitigation Plan is limited to the transition.
- In addition, migration planning should include a security impact analysis to understand any security impacts of the target state of the change.

SABSA (Technology Architecture Elements)

Sherwood Applied Business Security Architecture

- In most cases, the development of specific Technology Architecture security artifacts is not necessary, as long as it incorporates the relevant security controls and mechanisms defined in earlier phases.
- The Security Architect must ensure that the required controls are included in the Technology Architecture and verify whether the controls are used in an effective and efficient way. This includes the technology for the provision and regulation of system resources, such as electric power, processing capacity, network bandwidth, and memory.

SABSA (Implementation Governance Elements)

Sherwood Applied Business Security Architecture

- Security Audit
- Security Training and Awareness

SABSA (Change Management Elements)

Sherwood Applied Business Security Architecture

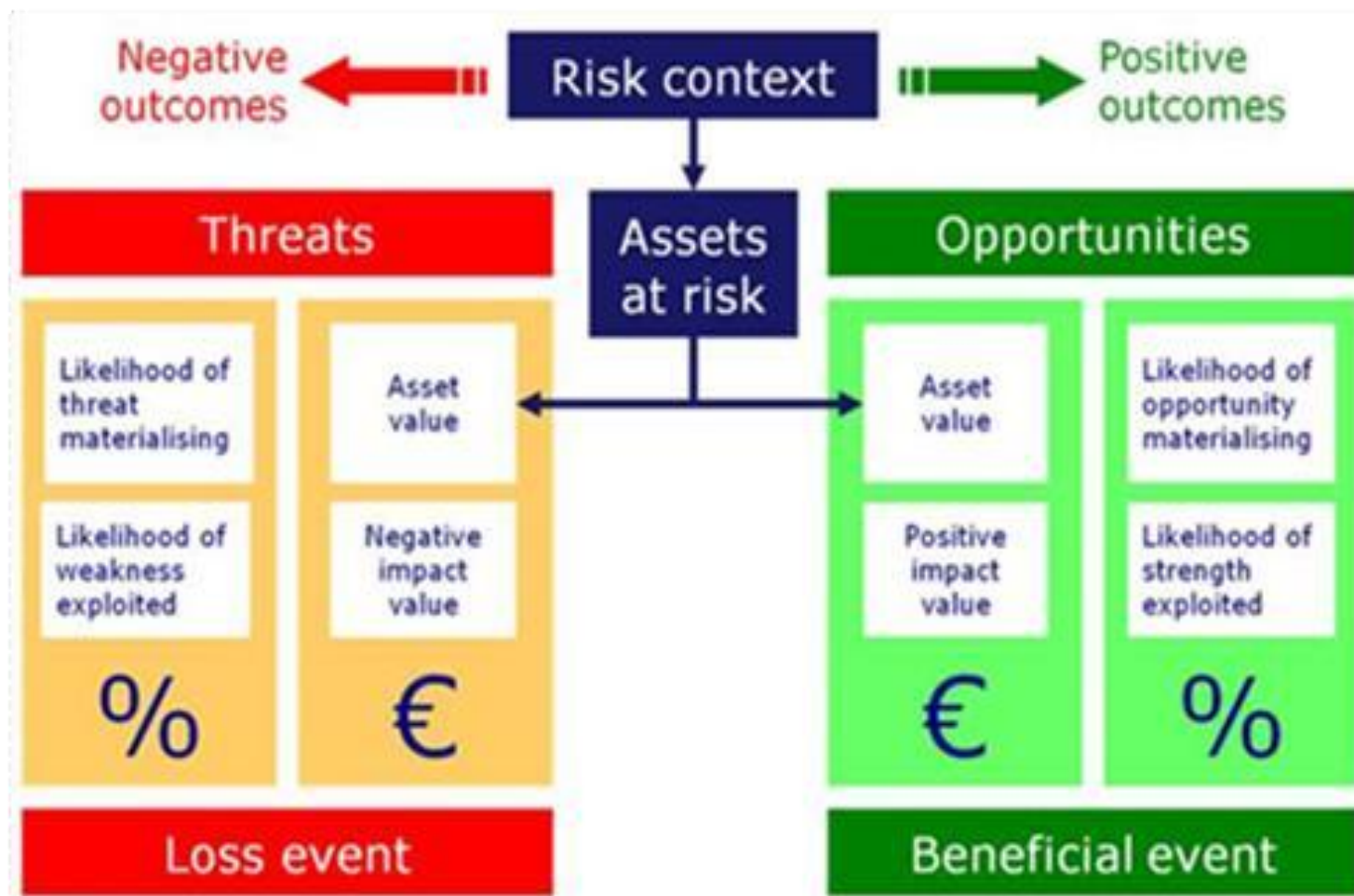
- Phase H does not produce tangible security outputs but defines two processes essential for security.
 - Enterprise Risk Management
 - Architecture Governance

Security Acronyms

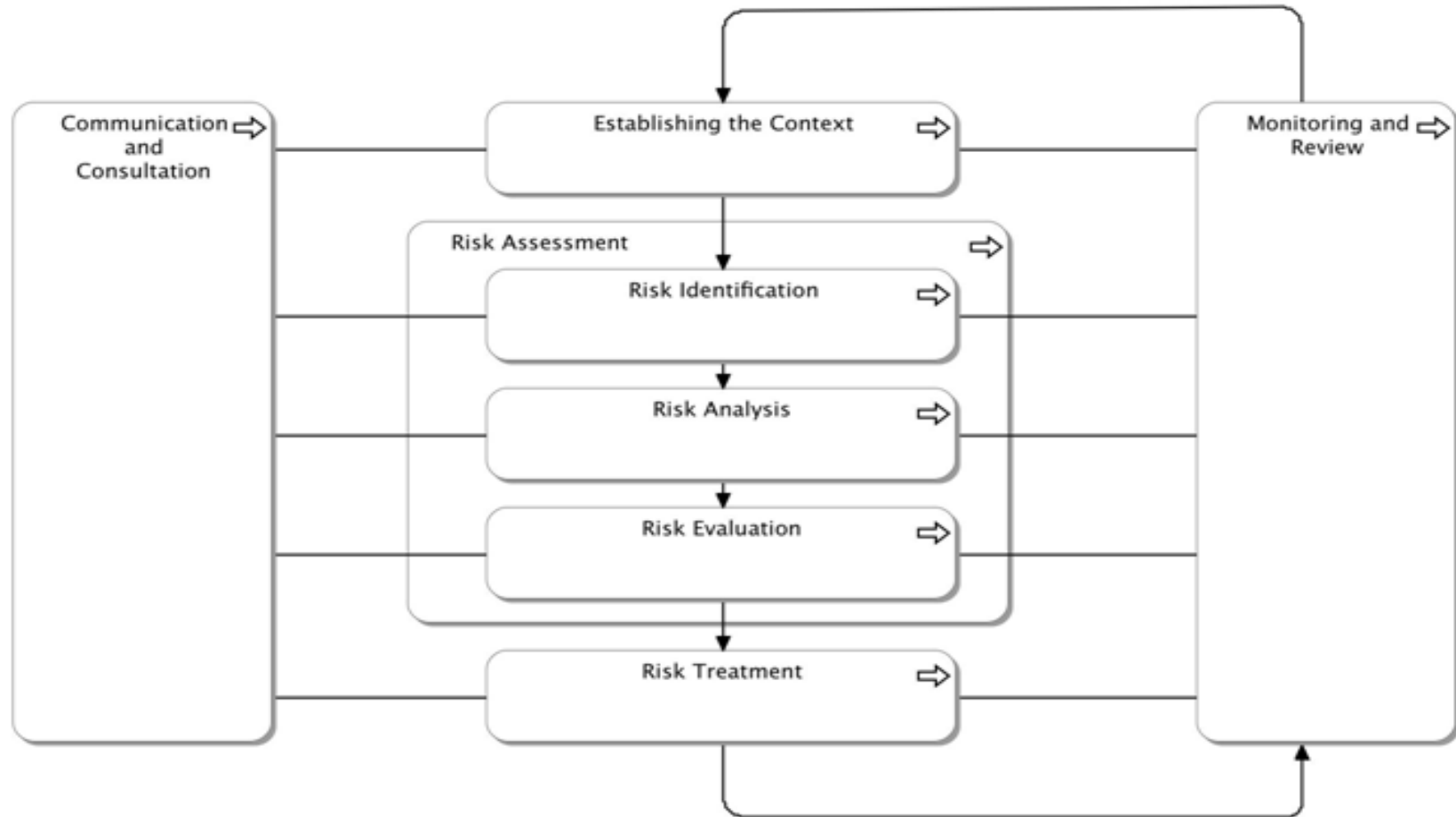
ABB	Architecture Building Block
ADM	Architecture Development Method
ERM	Enterprise Risk Management
ETSI	European Telecommunications Standards Institute
ISM	Information Security Management
ISMS	Information Security Management System
O-RA	Risk Analysis Standard (Open FAIR)
O-RT	Risk Taxonomy Standard (Open FAIR)
O-ESA	Open Enterprise Security Architecture
O-ISM3	Open Information Security Management Maturity Model
PCI-DSS	Payment Card Industry Data Security Standard
SBB	Solution Building Block

SABSA Sherwood Applied Business Security Architecture

SABSA Operational Risk Model

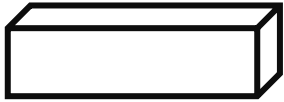


ISO 31000:2009 Model for Risk Management

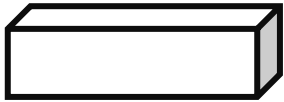


Topology Options

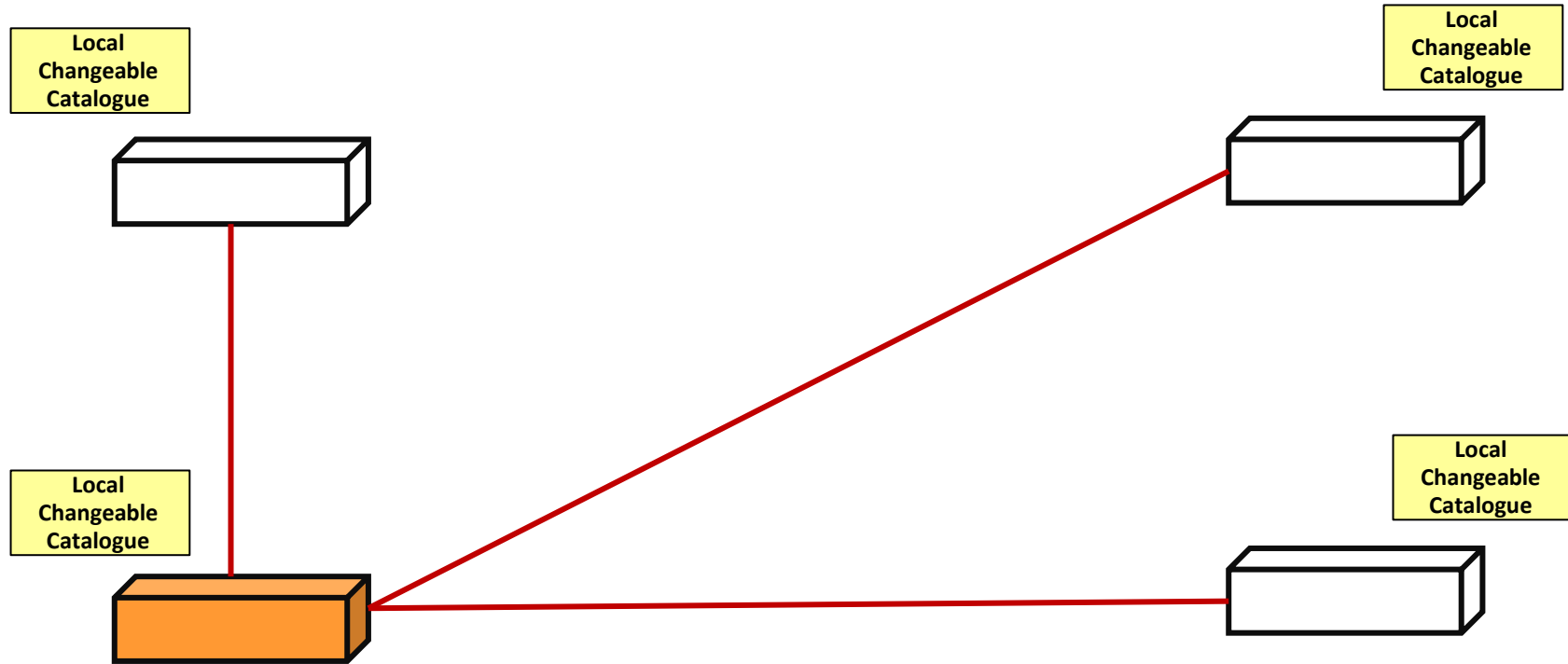
Network Topology



How to connect nodes in a network?



Network Topology (Point to Point)



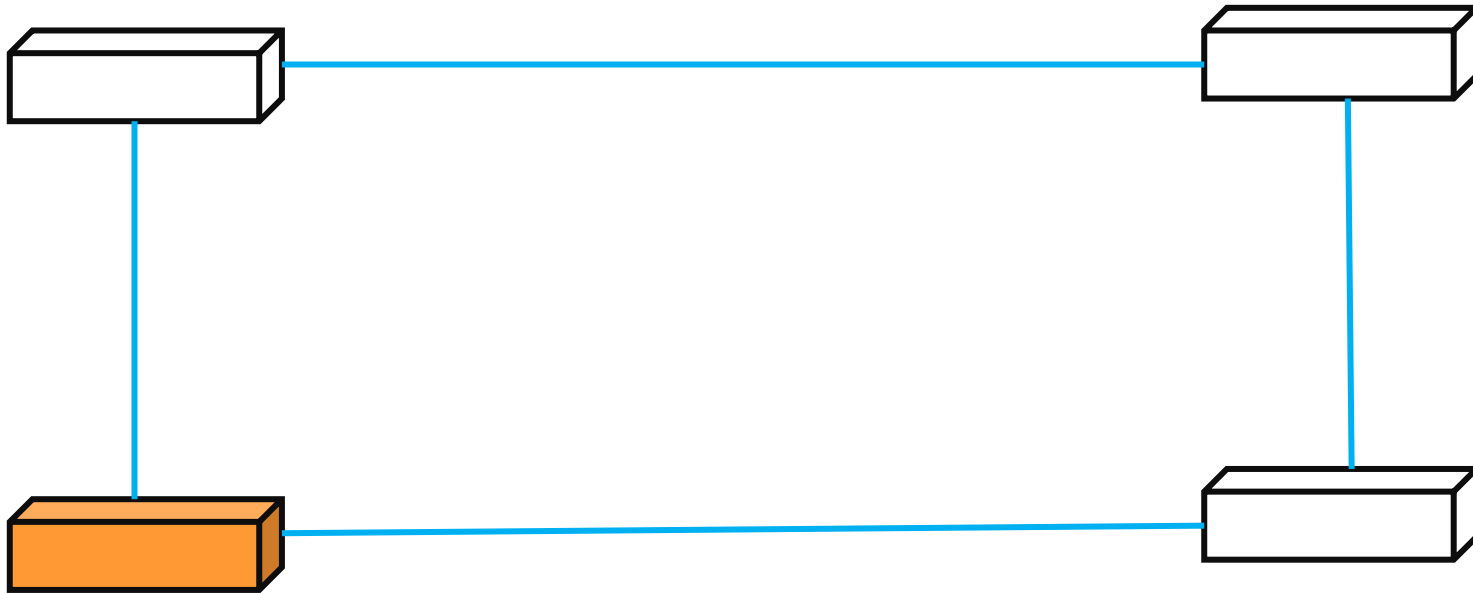
Point to Point ————





Hub/Bus ————

Ring ————

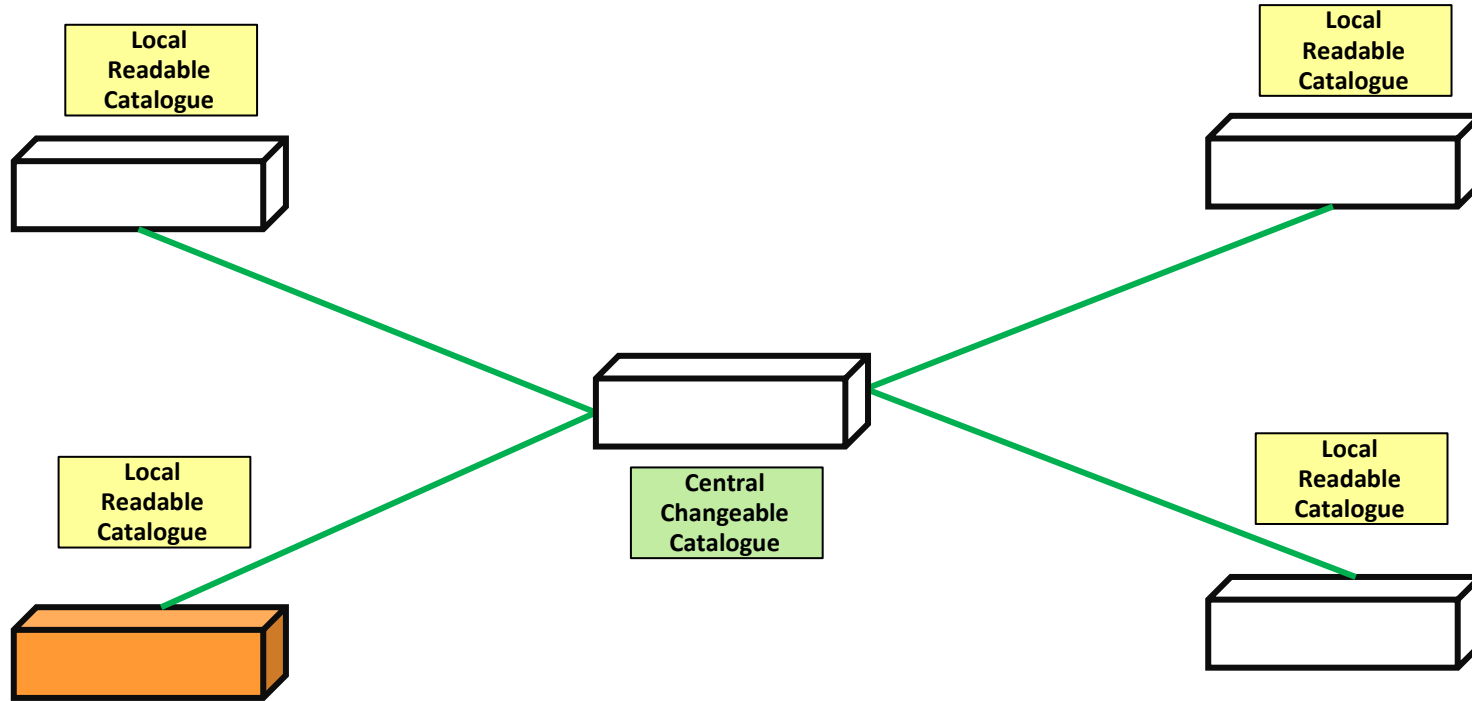
Broadcast ————

Network Topology (Ring)



- Point to Point 
- Hub/Bus 
- Ring 
- Broadcast 

Network Topology (Hub/Bus)



- Point to Point ————
- Hub/Bus ————
- Ring ————
- Broadcast ————

With this architecture consider
publish and subscribe mechanisms

Network Topology (Broadcast)



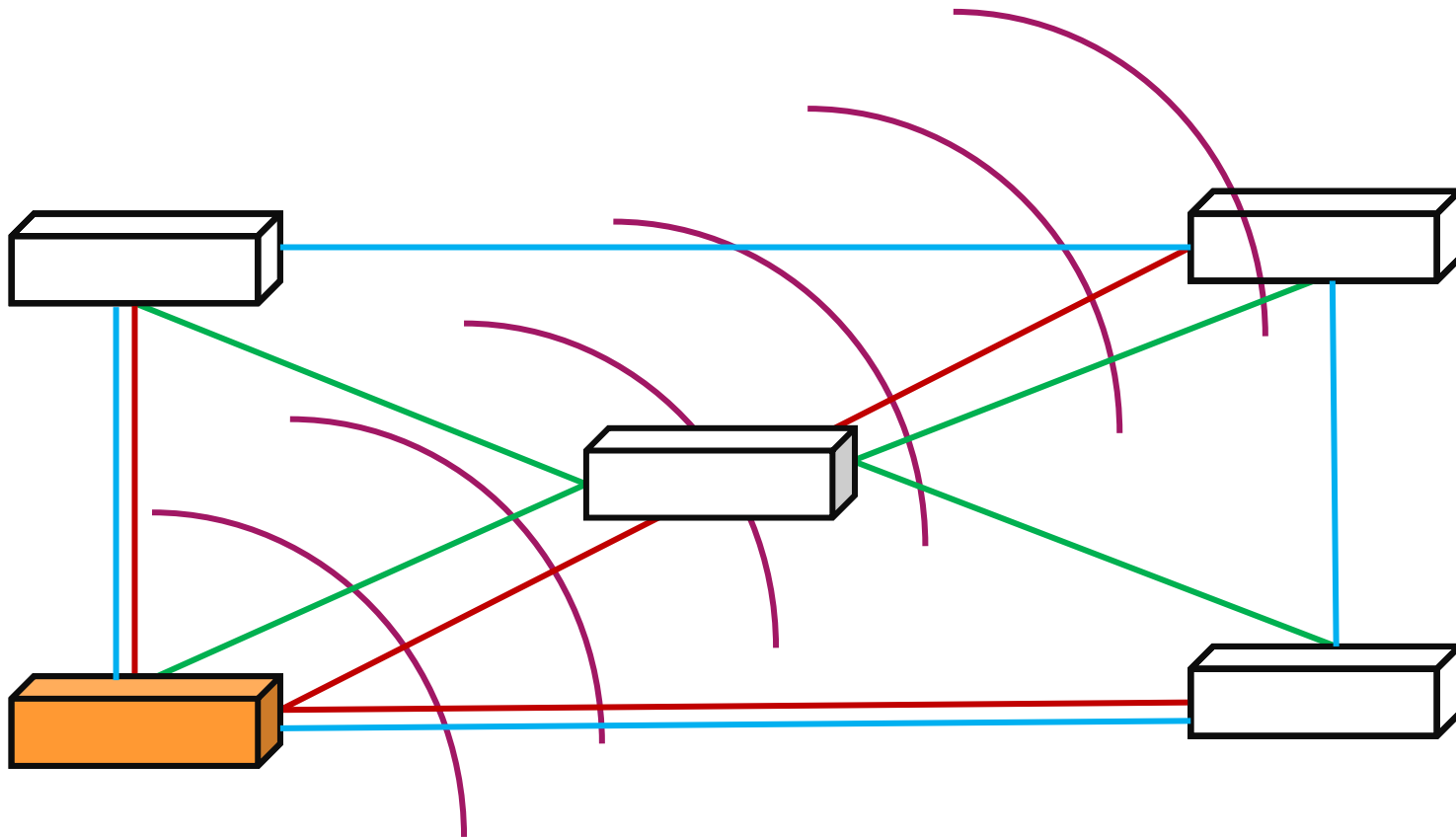
Point to Point 

Hub/Bus 

Ring 

Broadcast 

Network Topology (Set of Choices)



- Point to Point ————
- Hub/Bus ————
- Ring ————
- Broadcast ————

Example Maturity Model For EA

Example CMMI Style Model (For Enterprise Architecture Maturity)

Maturity Viewpoint	Viewpoint Element	Your capability score	Your capability evidence	Ad-hoc -Level 1	Repeatable - Level 2	Defined - Level 3	Managed - Level 4	Optimised - Level 5
Enterprise Architecture	Organisation	4		There is no repeated set of enterprise architecture roles or activities.	There is a set of repeated enterprise architecture roles and activities but they are not fully documented.	There is a defined and available organisation structure and definition of roles and responsibilities for requirements engineers.	Formal management with targets is in place for the enterprise architecture roles and teams.	The enterprise architecture team and support structure are frequently reviewed for its effectiveness and efficiency, with initiatives in place for performance improvement.
	Skills	3		There is no defined set of skills expected of requirements engineers.	There is a generally expected set of skills for requirements engineers but they are not documented or available.	There is a set of skills defined and available for most enterprise architecture roles.	The skills required by the organisation and the skills of each enterprise architect are recorded and reviewed.	The skills needed by the organisation and the skills of each enterprise architect are reviewed and investment is made to develop and improve those skills.
	People	2		There are few people in the organisation with the relevant enterprise architecture skills.	There are a number of enterprise architects in place but they have a variable knowledge and skillset.	There are enterprise architects identified and in place for most enterprise architecture roles.	The enterprise architects act as a collaborative and professional group in addition to their roles within specific management / reporting lines.	The enterprise architects in run an effective professional community continually identifying and addressing areas for improving enterprise architecture practice and outcomes.
	Training	3		There is no repeatable training in place for requirements engineers.	There is training in place for requirements engineers that most undertake but it is not formally documented and scheduled.	There is a defined and available set of training for requirements engineers.	Enterprise architect training is monitored, funding and time is made available and training outcomes are recorded.	Enterprise architecture needs and associated training are continually reviewed and the training programmes are revised as appropriate.
	Tools and Models	2		There are no standard and widely used tools and models to support enterprise architects.	There are commonly used tools in place but they are used on a case by case basis at the discretion of the enterprise architects.	There is a defined set of tools and models documented for enterprise architecture which are used in most cases.	The use of appropriate tools and models is funded and checked to ensure consistency in enterprise architecture deliverables.	The tools and models used to support enterprise architecture are continually reviewed and revised / replaced as appropriate.

Example CMMI Style Model (For Business Architecture Maturity)

Maturity Viewpoint	Viewpoint Element	Your capability score	Your capability evidence	Ad-hoc -Level 1	Repeatable - Level 2	Defined - Level 3	Managed - Level 4	Optimised - Level 5
Business Architecture	Organisation	2		There is no repeated set of business architecture roles or activities.	There is a set of repeated business architecture roles and activities but they are not fully documented.	There is a defined and available organisation structure and definition of roles and responsibilities for business architects.	Formal management with targets is in place for the business architecture roles and teams.	The business architecture team and support structure are frequently reviewed for its effectiveness and efficiency, with initiatives in place for performance improvement.
	Skills	3		There is no defined set of skills expected of business architects.	There is a generally excepted set of skills for business architects but they are not documented or available.	There is a set of skills defined and available for most business architecture roles.	The skills required by the organisation and the skills of each business architect are recorded and reviewed.	The skills needed by the organisation and the skills of each business architect are reviewed and investment is made to develop and improve those skills.
	People	4		There are few people in the organisation with the relevant business architecture skills.	There are a number of business architects in place but they have a variable knowledge and skillset.	There are business architects identified and in place for most business architecture roles.	The business architects act as a collaborative and professional group in addition to their roles within specific management/reporting lines.	The business architects run an effective professional community continually identifying and addressing areas for improving business architecture practice and outcomes.
	Training	5		There is no repeatable training in place for business architects.	There is training in place for business architects that most undertake but it is not formally documented and scheduled.	There is a defined and available set of training for business architects.	Business architect training is monitored, funding and time is made available and training outcomes are recorded.	Business architecture needs and associated training are continually reviewed and the training programmes are revised as appropriate.
	Tools and Models	3		There are no standard and widely used tools and models to support business architects.	There are commonly used tools in place but they are used on a case by case basis at the discretion of the business architects.	There is a defined set of tools and models documented for business architecture which are used in most cases.	The use of appropriate tools and models is funded and checked to ensure consistency in business architecture deliverables.	The tools and models used to support business architecture are continually reviewed and revised / replaced as appropriate.

Example Overall CMMI Style Assessment (For 10 Aspects)

Organisation Name	XXXXXXX XXXXXXX											
Assessed Architecture Maturity Level	3.42											
	Enterprise Architecture	Solution Architecture	Business Architecture	Information / Data Architecture	Application / Software Architecture	Technology Architecture	Requirements Capture / Management	Programme & Project Management	Operations Management	Change Management	Architecture Governance	Average
Organisation	4	4	2	4	5	4	3	4	3	3	4	3.64
Skills	3	5	3	5	4	3	2	2	4	3	5	3.55
People	2	3	4	2	3	4	3	3	3	3	3	3.00
Training	3	2	5	3	4	5	4	4	3	3	2	3.45
Tools	2	4	3	4	5	4	3	3	3	3	4	3.45
Total	2.80	3.60	3.40	3.60	4.20	4.00	3.00	3.20	3.20	3.00	3.60	3.42