Table of Contents

1 Introduction

ArchiMate (3.x) example views introduced here are organized into a layered framework according to ArchiMate standard (see ArchiMate version 3.0.1 here). These example views illustrate how ArchiMate concepts can be used. Some of the examples can be used as design patterns.

2 ArchiMate Example Views

2.1 Framework View

Figure 1: Framework View.

This view represents the framework that structures all the development aspects and related diagrams. The view can be modified according to what is appropriate in the case. As such, this view can be used for navigation between the diagrams. This version of the view is applied from ArchiMate (3) framework. *Motivation* is introduced as a "Layer" instead of an "Aspect" here.

3 Motivation Views

3.1 Motivation View

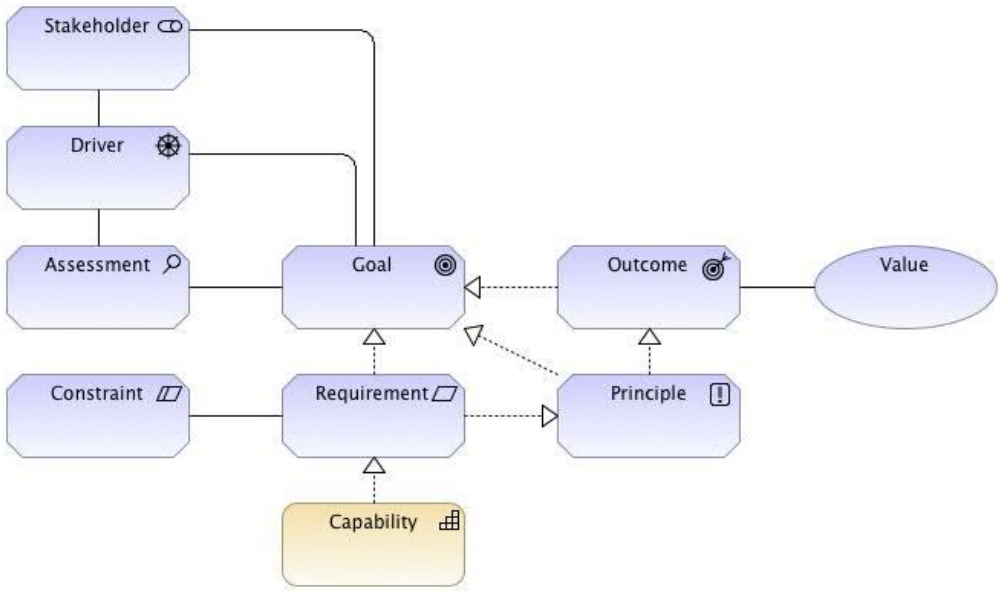


Figure 2: Motivation View.

This view can be used to analysis of the motivations, or reasons, that guide the design or change of an organization and its enterprise architecture. These motivational analysis are the starting points for all the change activities or business transformations in an organization. This view represents the vision of the development endeavour - whether the scale and scope comprises the whole organization or just part of it (e.g. line of business) or single program or project (solution level). Note: a *value* can be added e.g. to the outcome (or to any other ArchiMate element), to indicate what is the real value add!

Motivational elements are based on Business Motivation Model (BMM) [specification v.1.3, 2015, OMG].

3.2 Mission-Values-Vision View

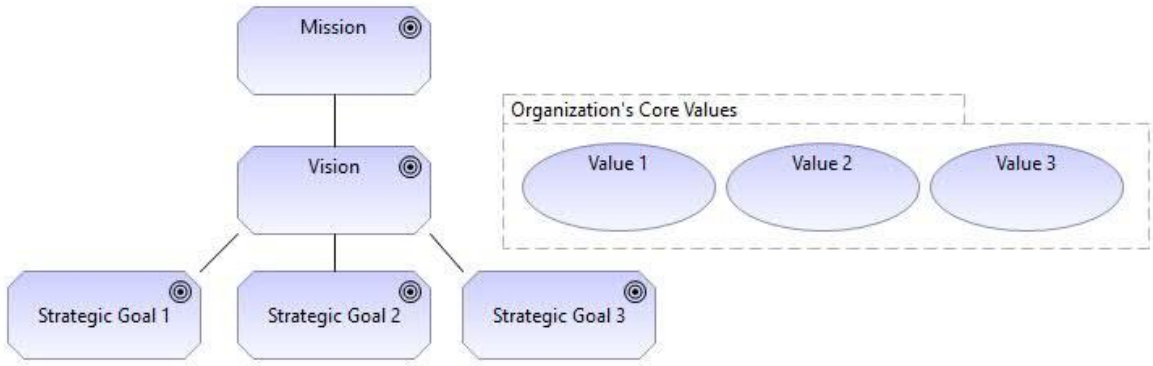


Figure 3: Mission-Vision-Values View.

This view can be used to represent the organization's mission, vision and core values. Mission expresses e.g. "What is the organization's purpose, what is it actually doing or intents to do, what is the primary reason for its existence?" Vision is the future state towards which the organization intents to evolve. Core values are what support the vision, shape the culture, and reflect what the organization values. Strategic goals need to be accomplished in order to achieve the vision of the organisation.

Reference: Aldea, A. – Iacob, M.-E. – Hillegersberg, J. - Quartel, D. – Franken, H. (2015) Modelling strategy with ArchiMate.

3.3 **Strategic Value Map View**

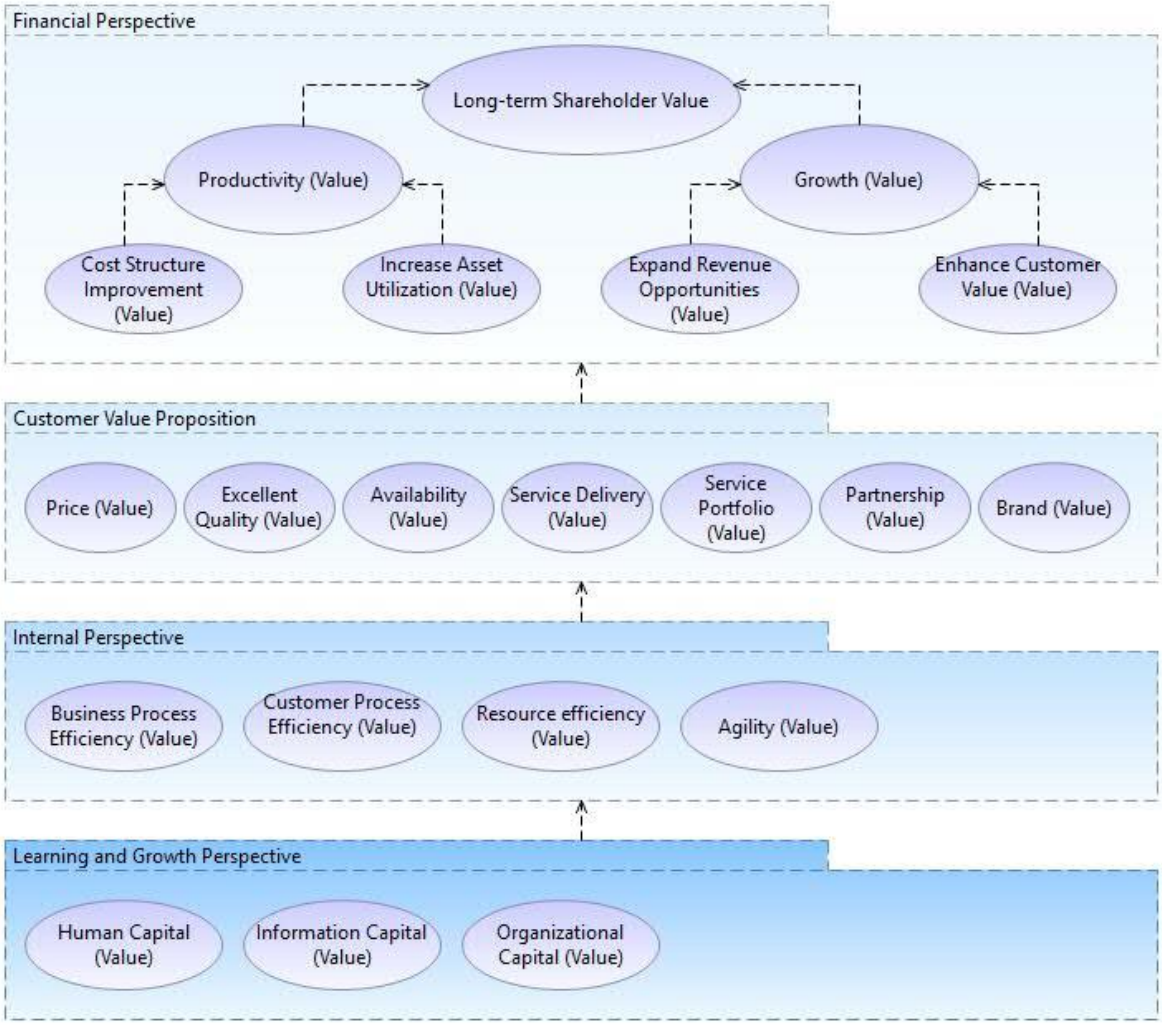


Figure 4: Value Map - Strategy Map View.

This view can be used for visualization of the strategies of the organization. This view contains strategic value-elements, from which all the development activities have to be derived - directly or indirectly. By visualizing the strategic values, it can be possible to trace all the other elements that are involved with the actual strategy execution. With this view, the strategy can be made available: visualized, communicated and linked to the reality.

3.4 Stakeholder Analysis View

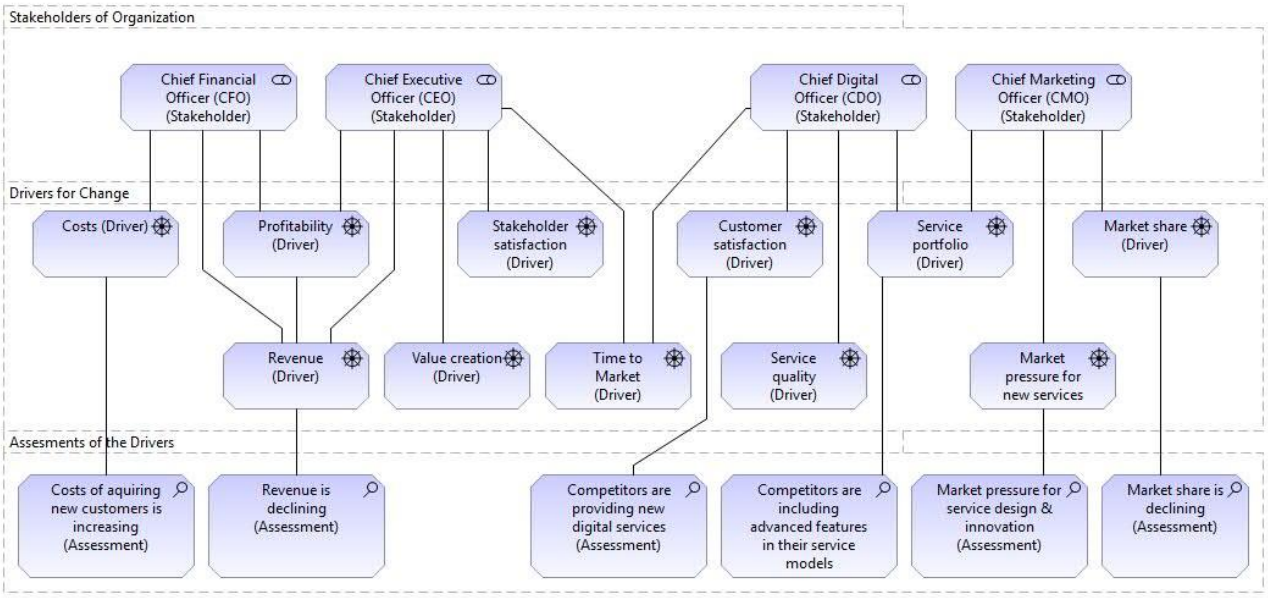


Figure 5: Stakeholder Analysis View.

This view can be used for stakeholder analysis for business development purposes: what are the drivers for change. First identify the relevant stakeholders and then the drivers for change that are in the interests of them. The "Assesment" concepts can be used for detailed analysis of drivers, e.g. according to SWOT (Strengths, Weaknesses, Opportunities, Threats) method. As typical, distinct stakeholder view diagrams can be created from different viewpoints. Another reason for splitting large diagrams into smaller, is to keep the diagrams compact and readable - for the sake of simplicity.

3.5 Stakeholder View

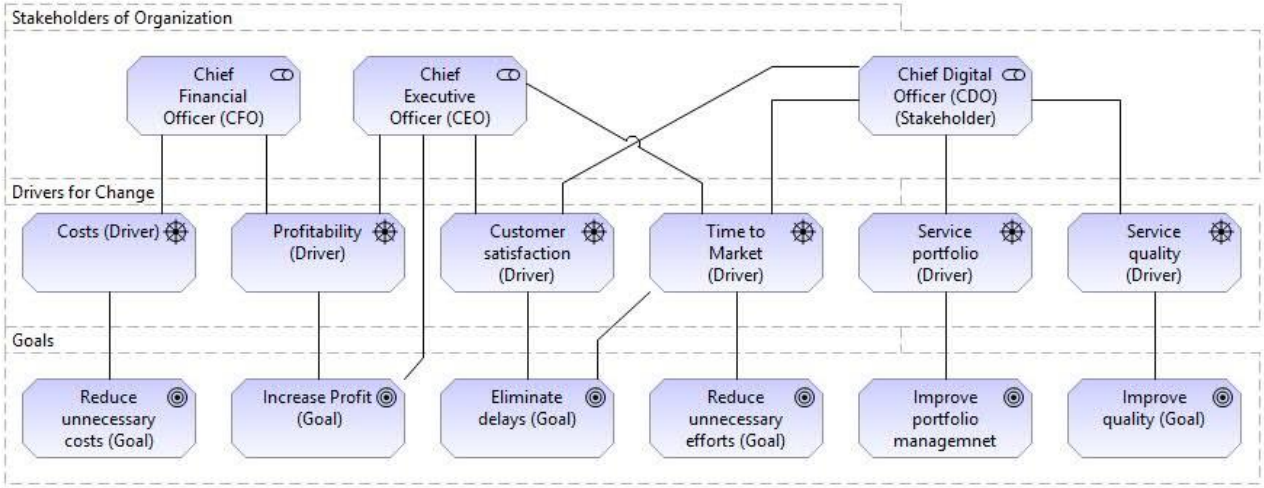


Figure 6: Stakeholder View.

This view can be used for linking stakeholder's drivers to business goals. Goals are the key elements of development in an organization. All the subsequent elements should be traced back to these primary reasons for all the change activities.

3.6 **Principles View**

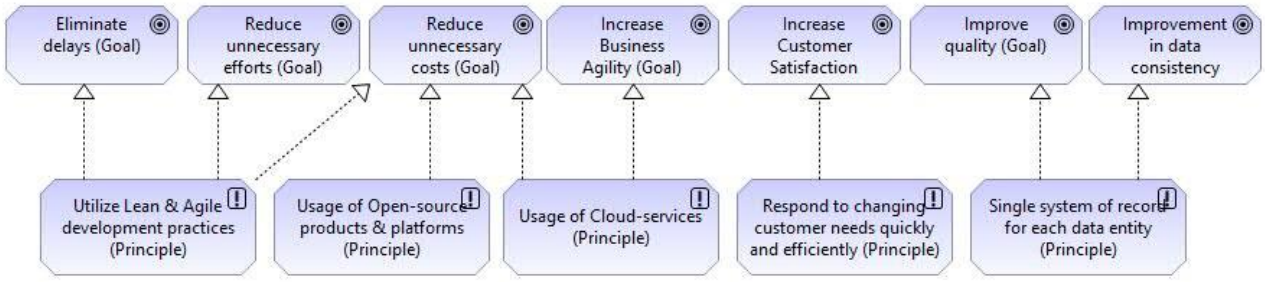


Figure 7: Principles View.

3.7 **Risk & Security View**

Mapping of Risk and Security Concepts to the ArchiMate. Security and data protection matters are part of the risk management. This modelling approach covers them both.

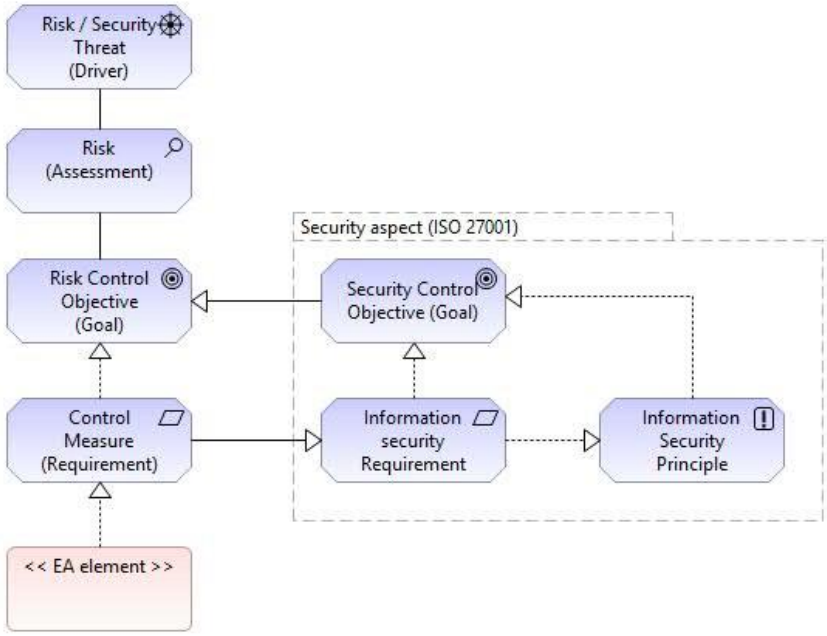


Figure 8: Risk and Security View.

References:

- *How to Model Enterprise Risk Management and Security with the ArchiMate® Language, Open Group, DocumentNo: W172, 2017.*
- *Modeling Enterprise Risk Management and Security with the ArchiMate® Language, Open Group, 2015*.

3.8 SWOT Analysis View

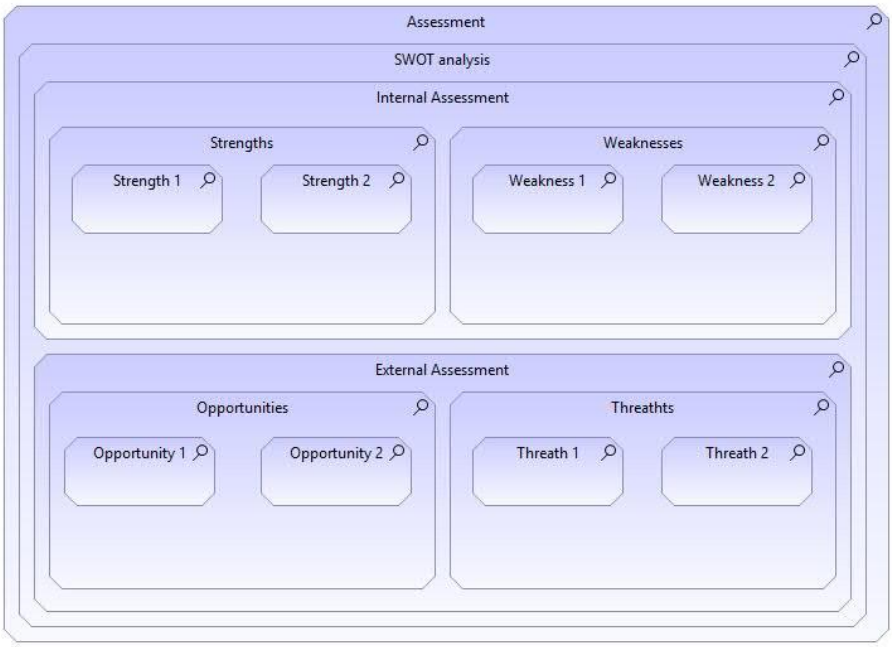


Figure 9: SWOT analysis View.

3.9 Goals View

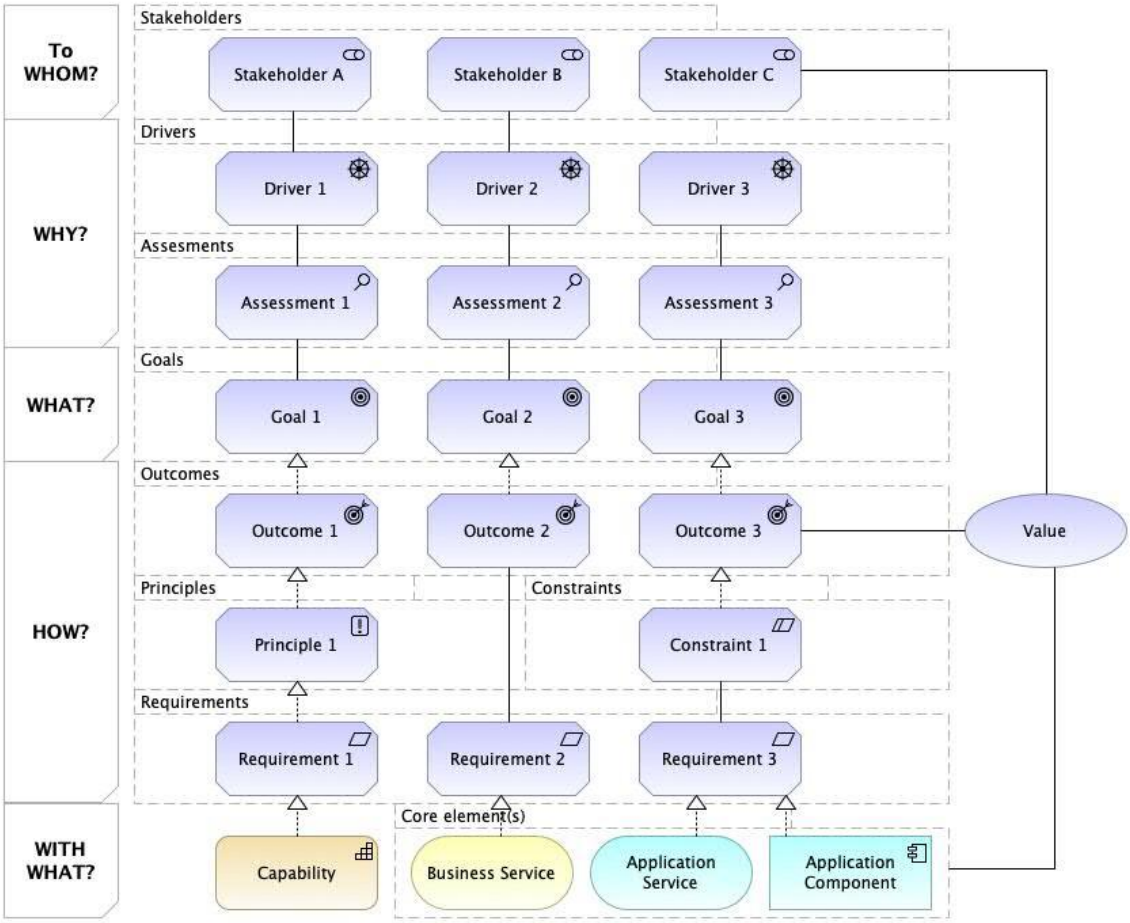


Figure 10: Goals View (with Value-element).

3.10 **Business Model View**

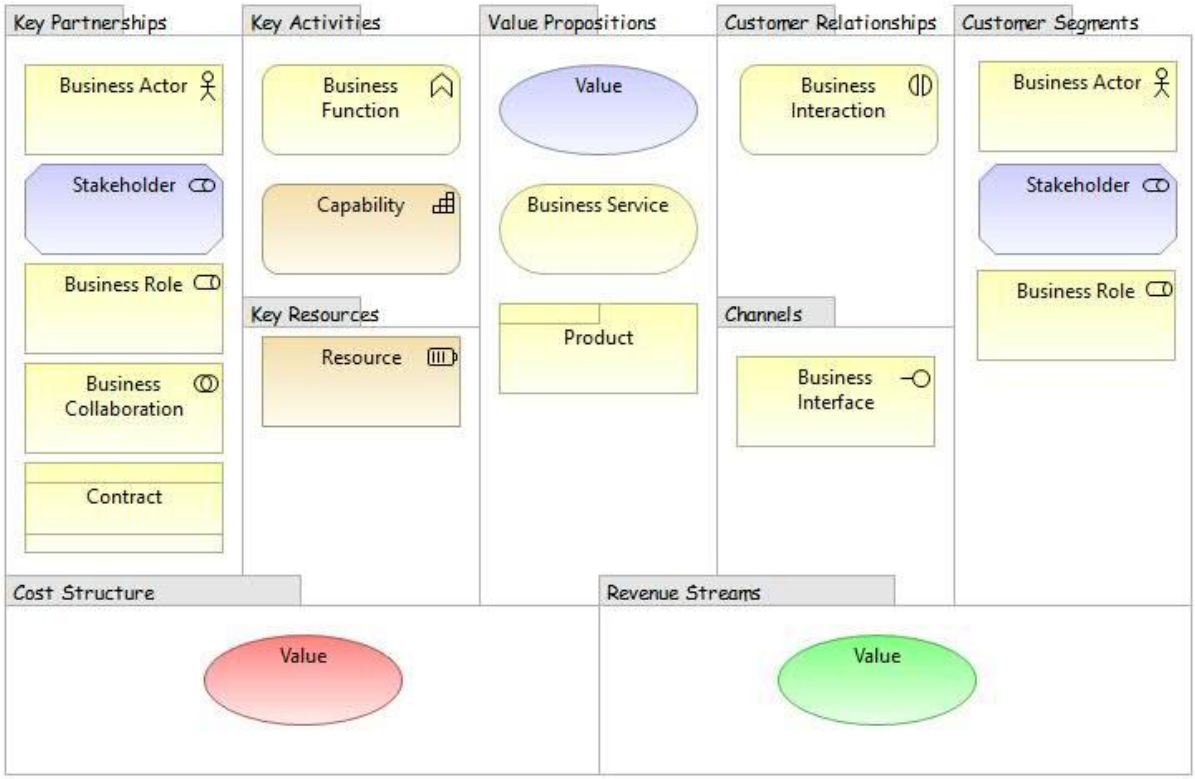


Figure 11: Business Model View.

This is the basic form of the Business Model Canvas (BMC) by A. Osterwalder, but it can be variated according to what is appropriate. There are also versioned approaches such as "Service Model Canvas" or "Lean Canvas". A BMC can be used e.g. for business model design and innovation.

Modeling a BMC with ArchiMate "facilitates tracing of requirements from business demands down to the design specifications. This helps discovering the effects of business model changes on architectural design." [L.O. Meertens et al.]

Holistic development includes built-in architecture-support for strategy and business model analysis. This enables business analysts and developers observe e.g. how well business model supports the strategy and how business model fit into the organization, and vice versa.

If the BMC is modeled within a modeling tool, an advantage of this approach is that all the elements of the BMC can be used in other views of the same model repository. And when *pivoting* the business model, all the changes are immediately visible. Business modelers can create new elements such as services, or utilize all the existing elements in the repository such as organization units or resources.

4 Strategy Views

4.1 Strategy View

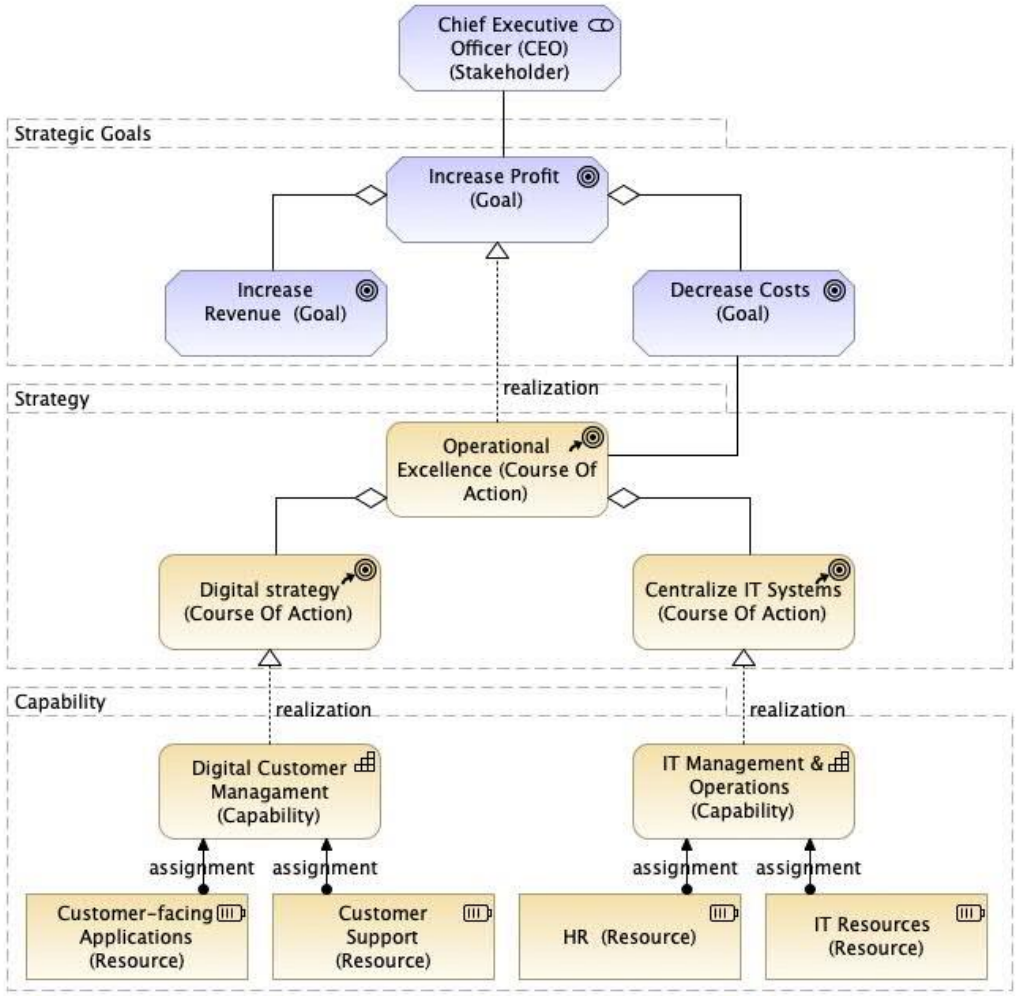


Figure 12: Strategy View.

ArchiMate version 3 now supports business strategy related concepts such as "Course of Action", "Capability" and "Resource", which can be used for modeling business strategies of the organization. The value and importance of this view is in the way that the goals of the organization can be linked to strategies, and how they can be linked into enterprise architecture via capabilities. This view can be used to apply the "Goal-Based Strategic Model" (Azevedo et al. 2015), in which goals constitute a hierarchy, so that higher-level goals can be decomposed into lower-level goals.

4.2 Business Motivation Model (BMM) View

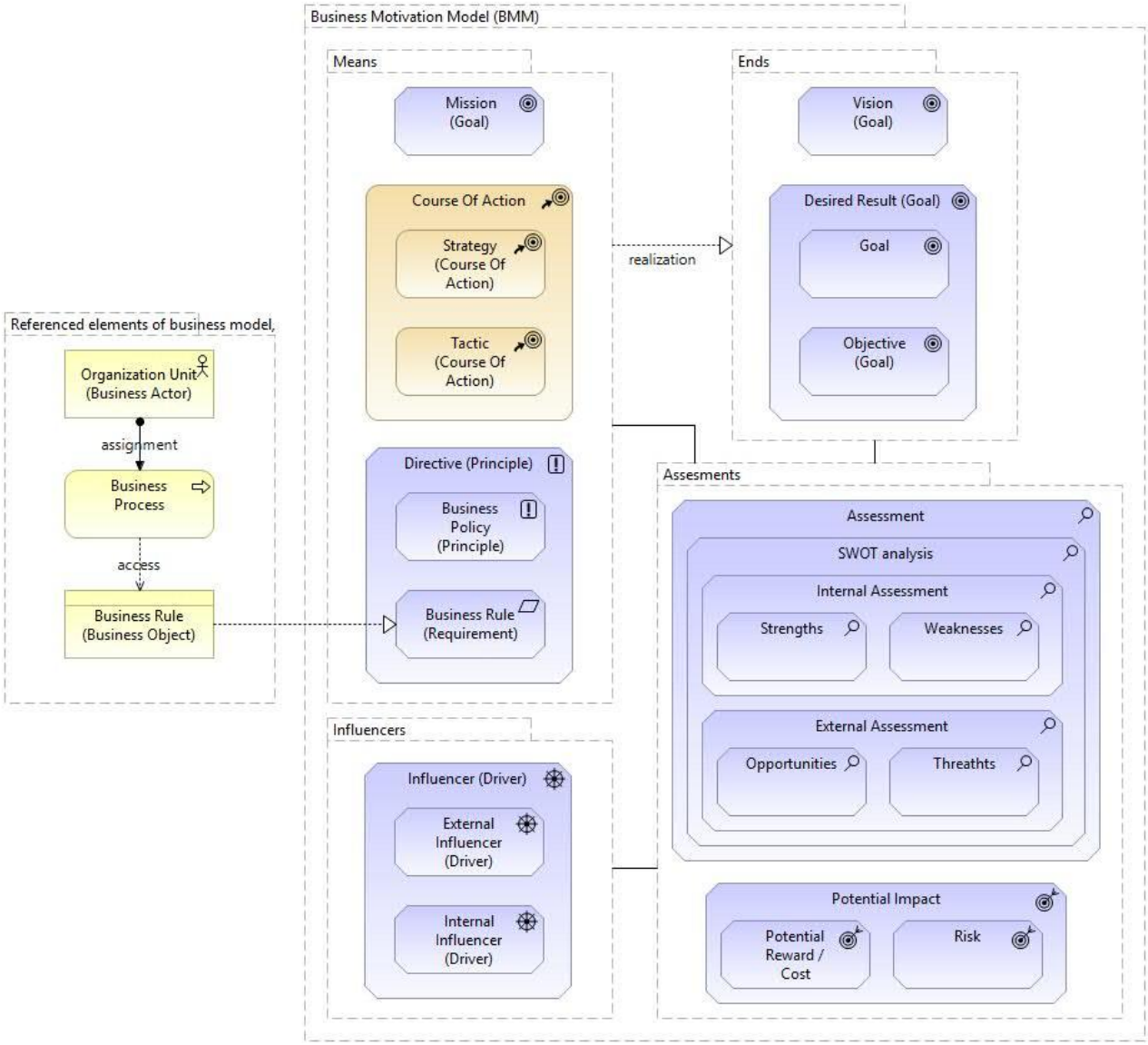


Figure 13: Business Motivation Model (BMM) View.

4.3 **Requirements View**

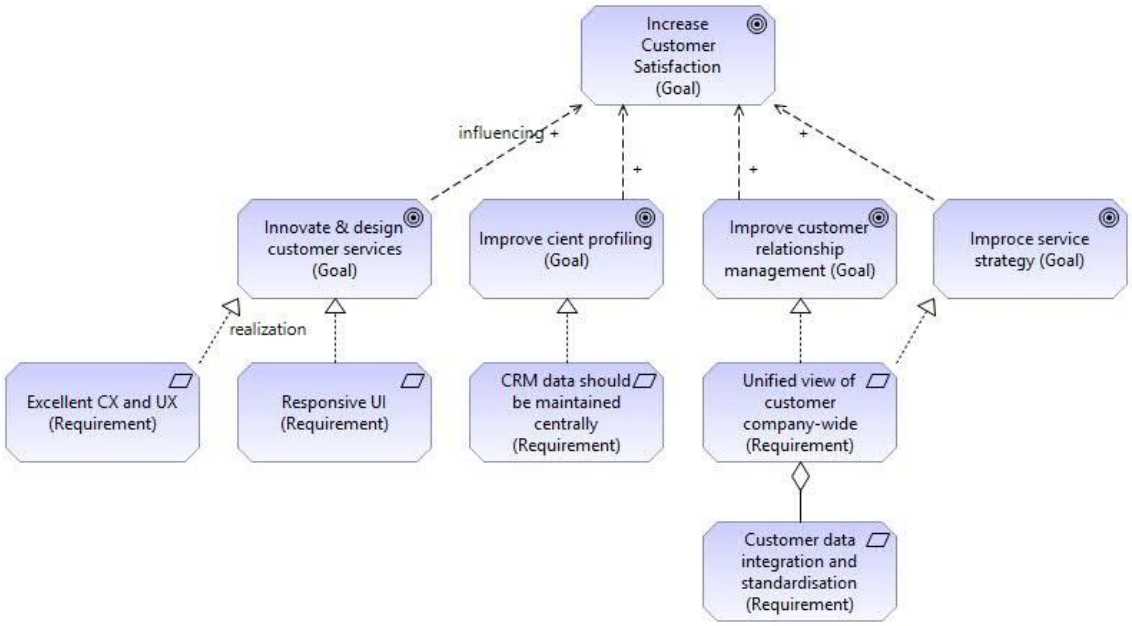


Figure 14:Requirements View .

This view can be used to gathering requirements based on the strategic goals. This is linking the strategies to implementations: it is possible to trace strategies to execution.

4.4 Strategy to Capability View

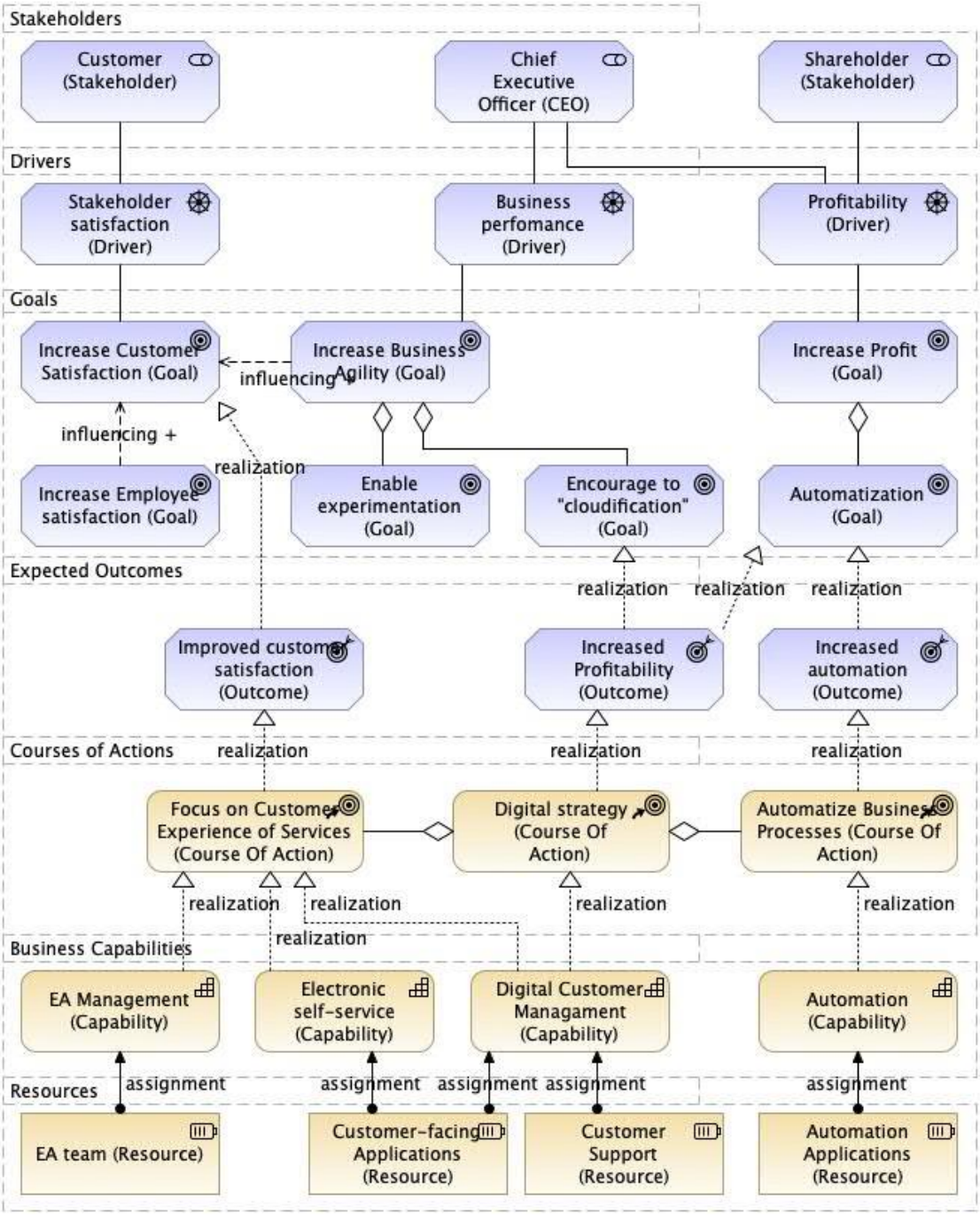


Figure 15: Strategy To Capability View.

This view can be used for *Capability-Based Planning (CBP)* purposes, together with other ArchiMate concepts such as "Driver" and "Goal" as shown in the diagram below. This view can be used to support *Strategy Planning* (and -Execution) purposes. As such, this kind of views can be used in Strategy-to-Capability phase, which can be included in the "Strategy-to-Portfolio" in IT4IT.

4.5 Capability Map View

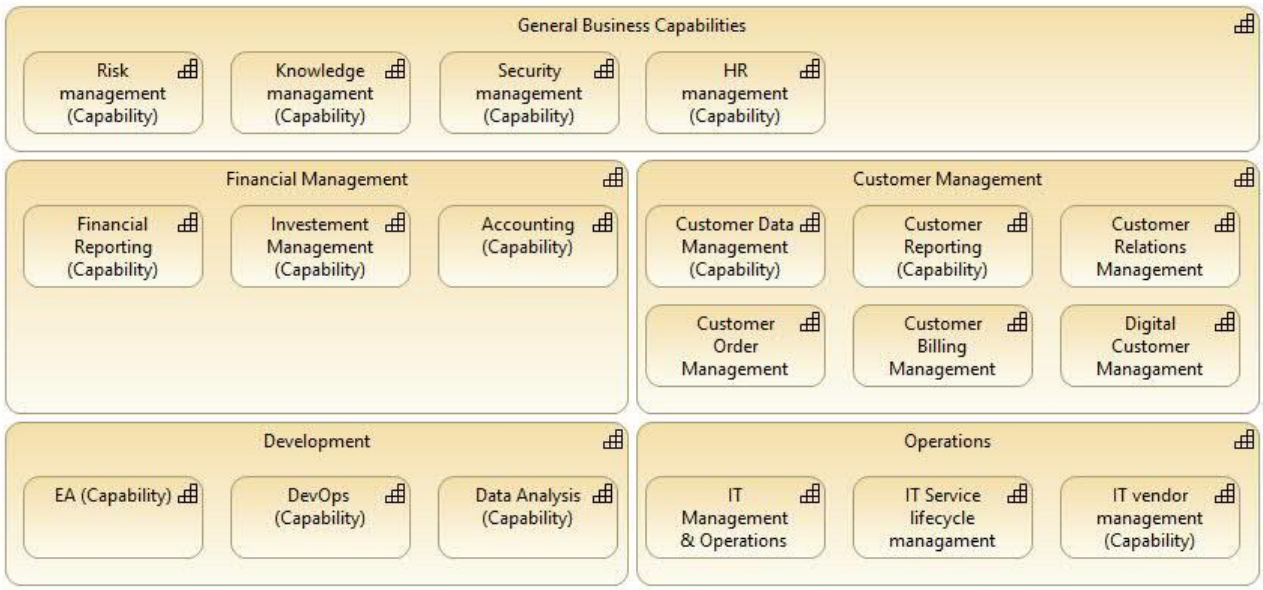


Figure 16: Capability Map View.

This view can be used for giving an overview of organizations capabilities: what the organization does or can do.

4.6 Capability Planning View

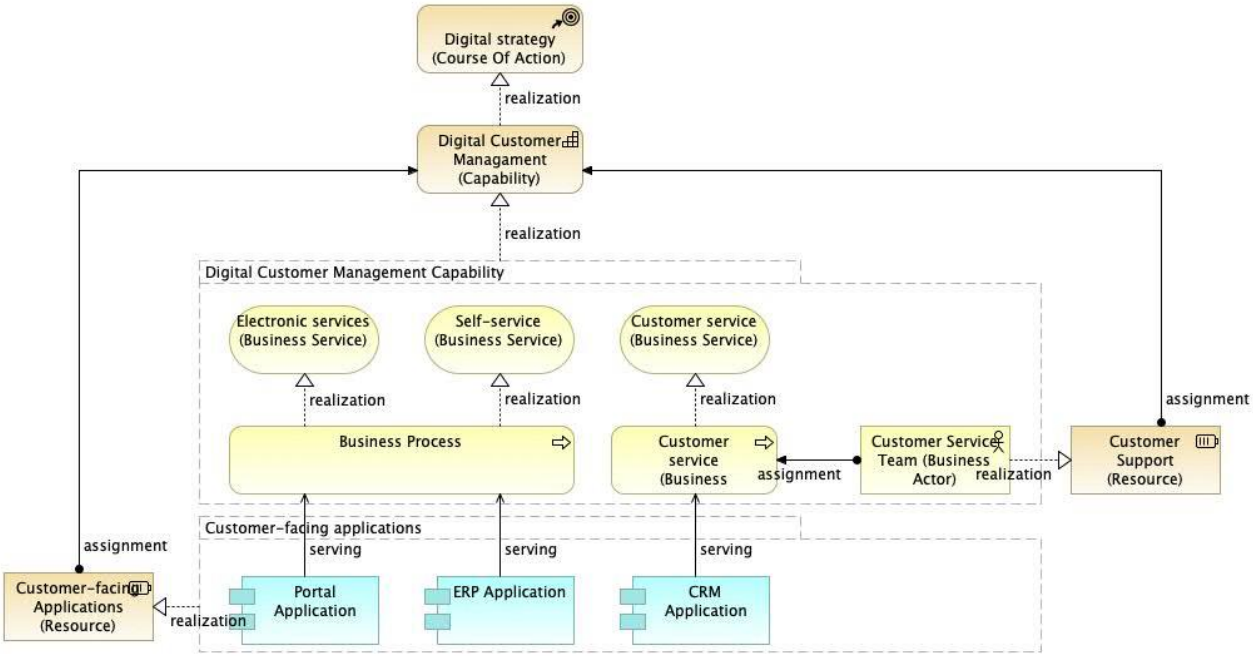


Figure 17: Capability Planning View.

This view can be used for e.g. *Capability-Based Planning (CBP)* purposes, which is "the Link between Strategy and Enterprise Architecture". This view can be used for e.g. mapping strategies to required capabilities, and mapping capabilities to resources and other building blocks.

4.7 Capability Realization View

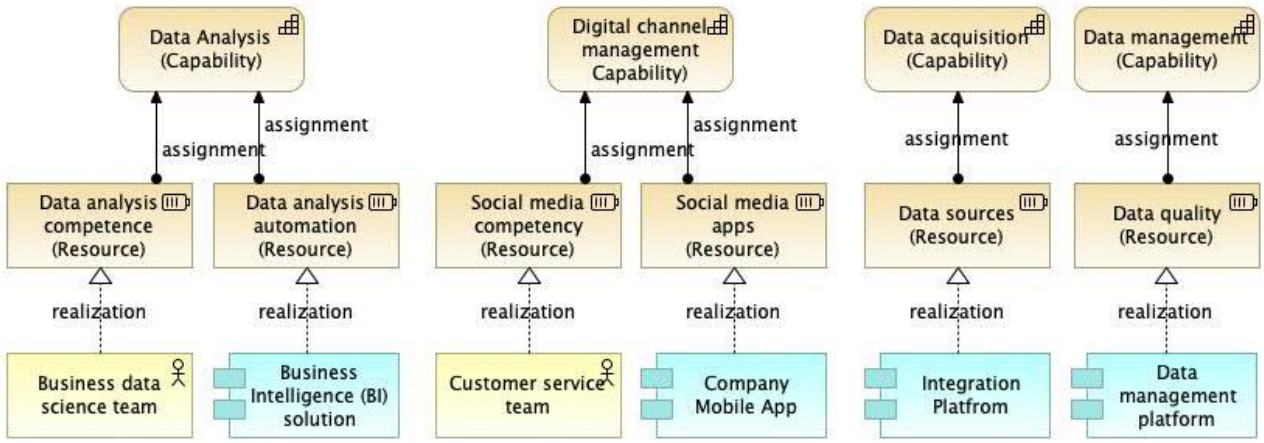


Figure 18: Capability Realization View.

4.8 Value Stream View

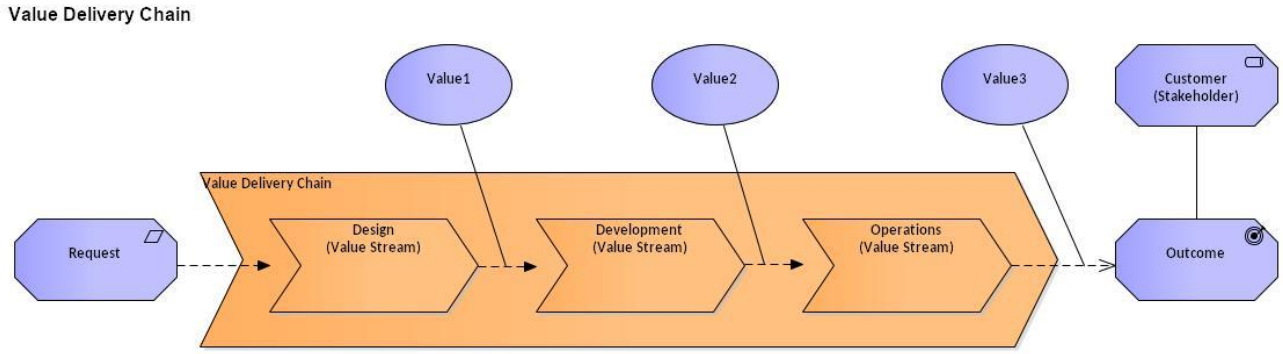


Figure 19:Value Delivery Chain-simple example.

Value Chains, Value Networks and Value Streams can be modelled with ArchiMate Value Stream -element, that is to be supported in the next version (3.1) of ArchiMate standard (2019). Here is an example view created with Sparx EA (other examples are created with Archi.).

4.9 Value Stream – Capability Cross mapping View

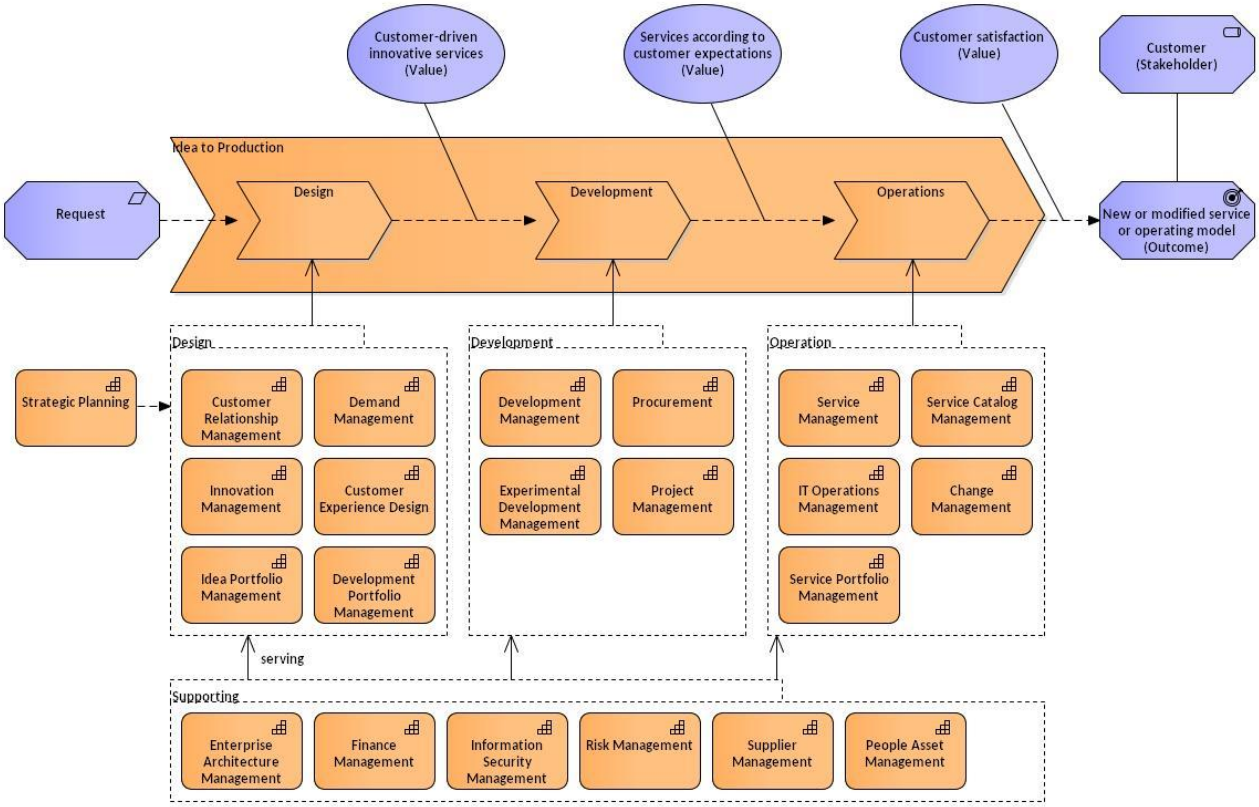


Figure 20: Value Delivery Chain.

Here is an extended example, which illustrates how capabilities support (serve) the value stream. This view can be used for defining WHY capabilities are needed, what is their linkage to the value creation.

5 Business Views

Business Architecture Layer Views.

In each layer there are several "maps" of elements that are managed within the EA-tool, such as Business Services Map, Process Map etc. After identified and introduced maps, those elements may be used in other diagrams (such as layered views). The purpose of the maps is to manage catalogs of "EA assets" as "portfolios" (analogous to portfolios of ideas, services and projects etc.). EA-tools typically provide other features for each element, e.g. properties or attributes. Those can be used to provide additional information per each element. This kind of extra information can also be used in different kinds of analysis purposes.

There can be several maps on each layer e.g. as follows:

- In Business Layer: Business Service, Business Actors, Business Processes
- In Application Layer: Application Services, Applications
- In Technology Layer: Technology Services, Platforms, Technologies etc.

Some example business layer maps are introduced here.

5.1 Business Services Map View

Figure 21: Business Services View.

This view gives an overview of the business services of the organization. This kind of view can be used as "Service Catalog" or "Service Portfolio" management purposes. It is important to identify what are the business services the organization is providing to its customers. In addition, a business service is a starting point for modelling all the underlying organizational processes and structures. As such, business services are the most important elements of the enterprise architecture.

5.2 Business Process Map View

Figure 22: Business Processes View.

This view can be used as "Process Map" which gives an overview of the business processes of the organization.

5.3 Business Process Co-Operation View

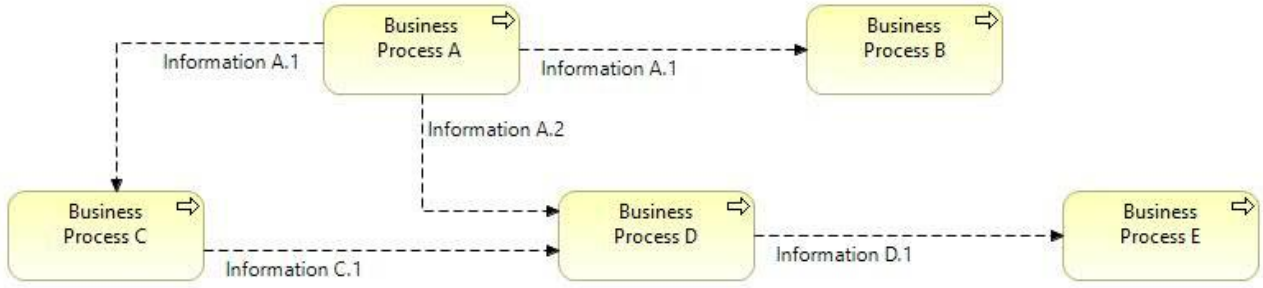


Figure 23: Business Process Co-operation View.

This view can be used e.g. for modelling the operating model.

5.4 Business Actors Map View

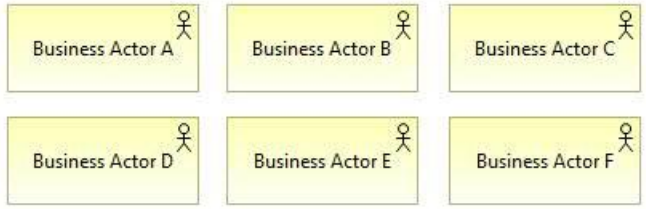


Figure 24: Business Actors View.

Business Actors can be a) internal or b) external. Internal business actors are e.g. organization units, and external business actors are e.g. customers, business partners, or other stakeholder groups that co-operate with the organization (such as public sector organizations or other governance authorities).

5.5 Business Actor Co-Operation View

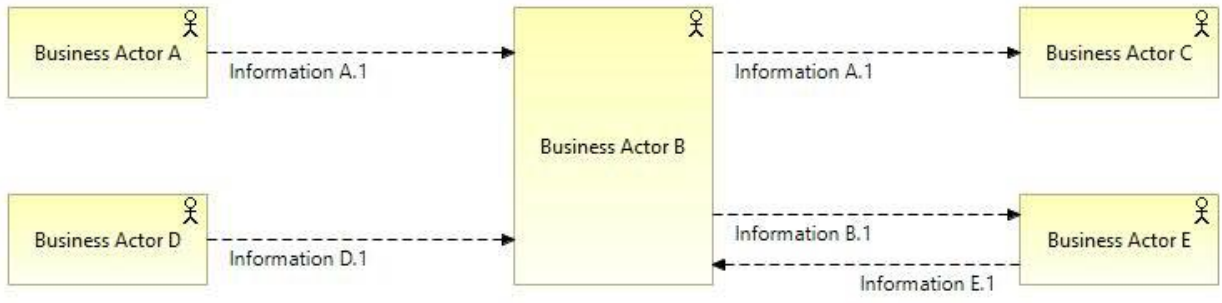


Figure 25: Business Actor Co-Operation View.

Two usage scenarios are as follows:

1) Intra-Enterprise View: Business actor co-operation viewpoint, which depicts how internal business actors co-operate, how they switch information.

2) Inter-Enterprise View: The *Ecosystem* viewpoint, which depicts the operational environment in which an organization operates. An ecosystem is a network of organizations and business partners, which are co-operating via interactions of collaborations. There are suppliers, sub-contractors and other b2b partners, customers etc.

5.6 Business Process View

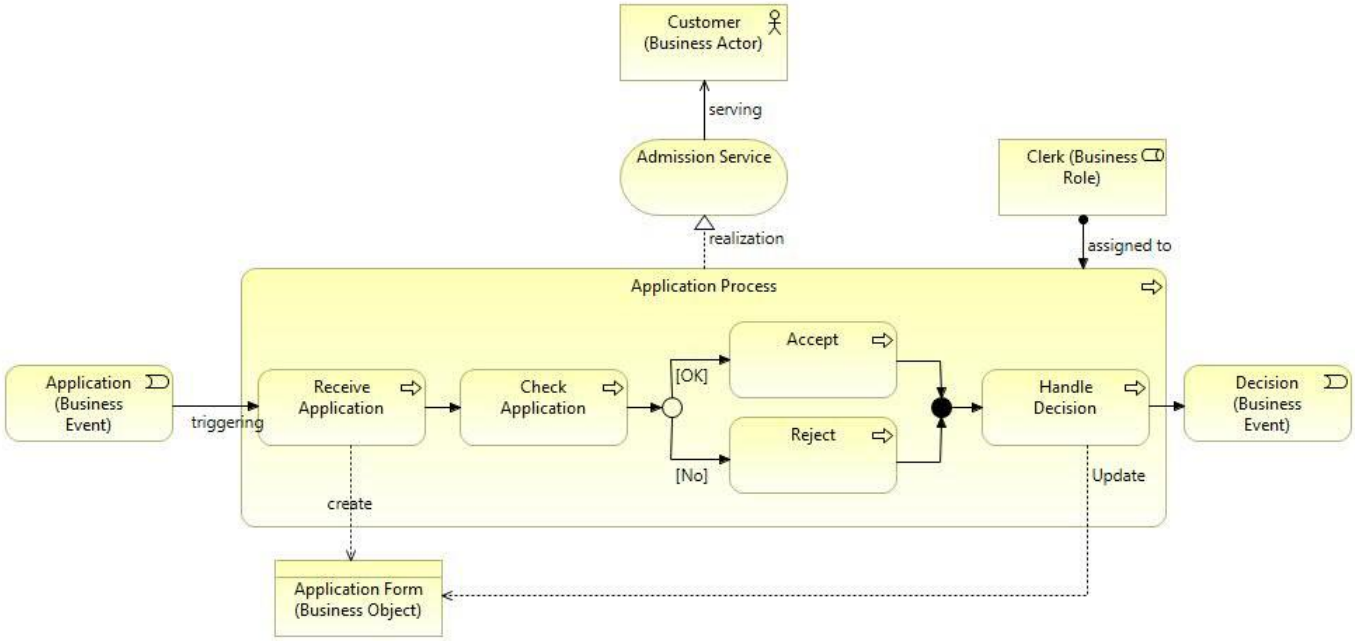


Figure 26: Business Process View .

This business process view provides a "high-level structure and composition of a business process (or several processes), the **services** that are offered, the assigned **roles** of **actors**, and the **information** used by the business

process" [ArchiMate 2.1 specification]. This process diagram contains "Junction" -elements to model "fork" and "join" in the process flow.

5.7 Business Process View With Business Roles As "Swimlanes" of a Process - A Layered Approach

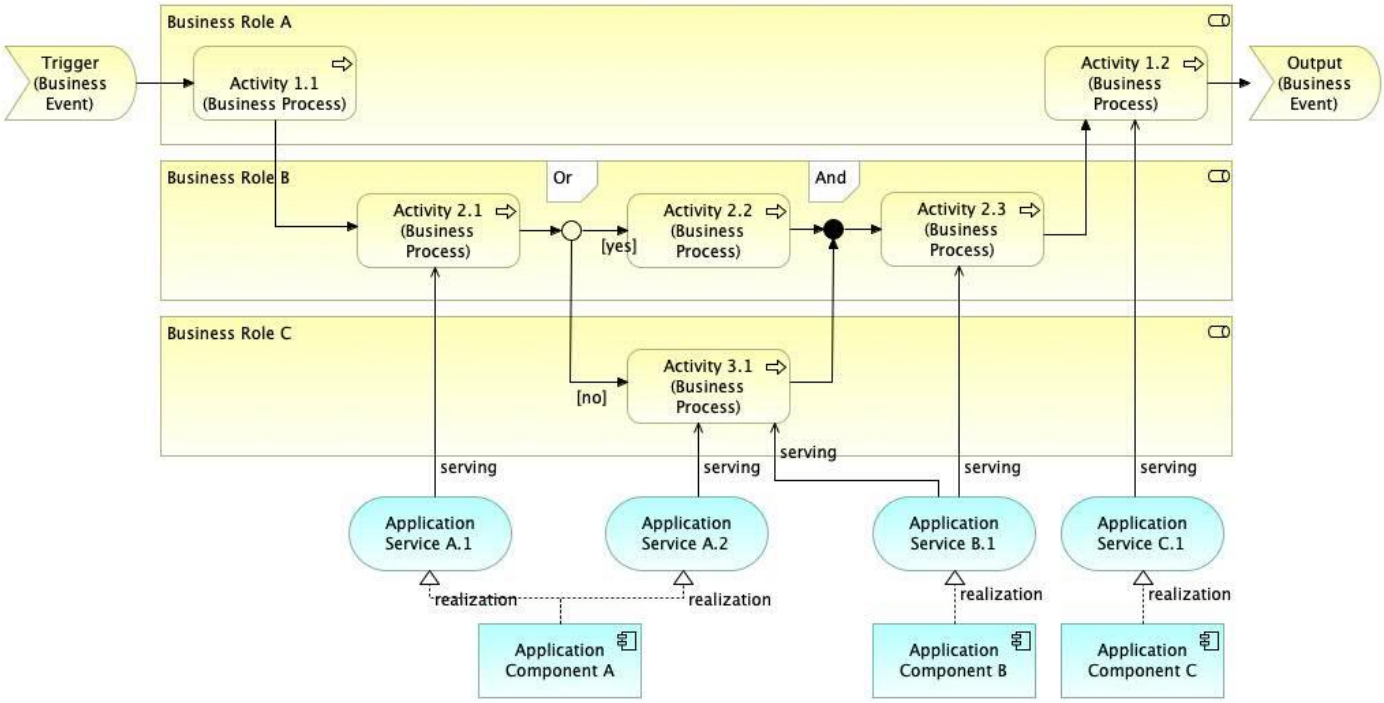


Figure 27: Business Process View With Swimlanes As Roles of a Process - A Layered Approach (2).

Note! Process steps (activities) are nested into business roles (visualized as "swimlines"), which means that: these Business Roles are assigned to these Business Processes / process steps. As such, this view is combination of business process view and layered view.

5.8 Customer Journey Map View

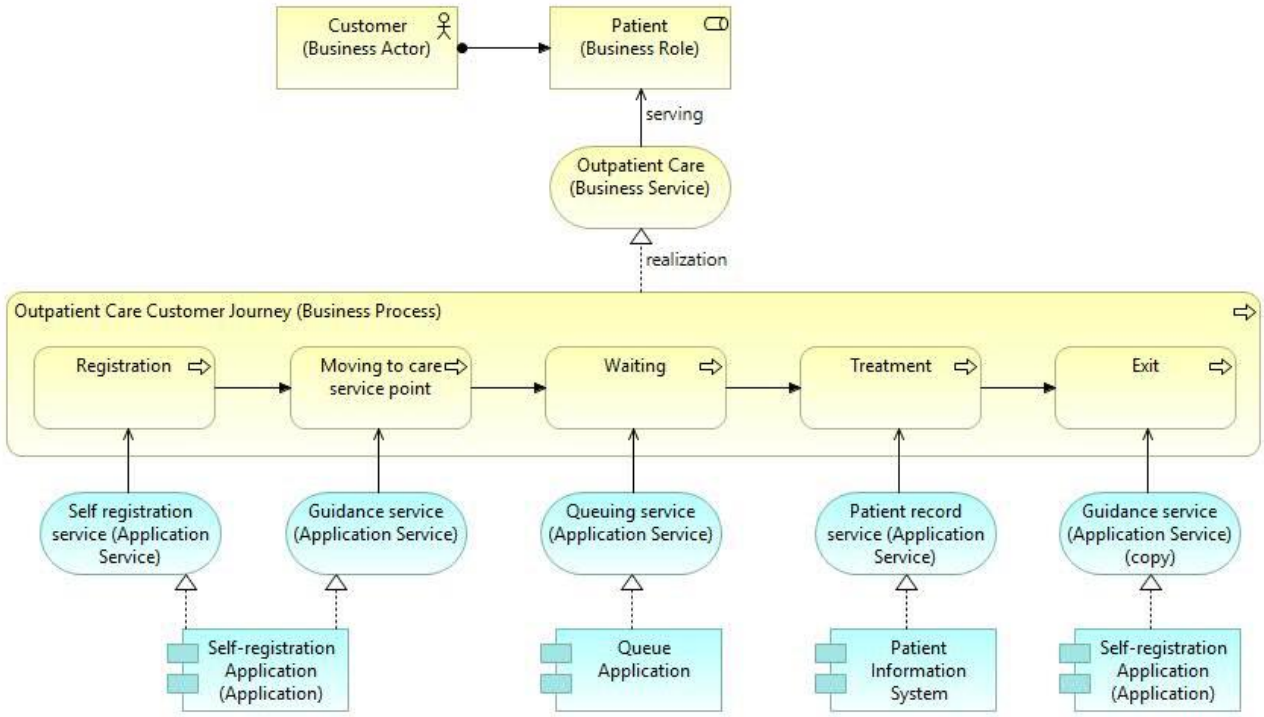


Figure 28: Customer Journey View.

This customer-centric viewpoint is focusing on customer experience. This "service design" related approach is concentrating the "outside-in" development of the service that is to be designed. This highlights the services and products as essential aspects that produces value to customers - and indirectly to the organization itself. A customer journey path can be used to visualization of a customer value stream, which spans over several application services and applications.

5.9 Service Blueprint View

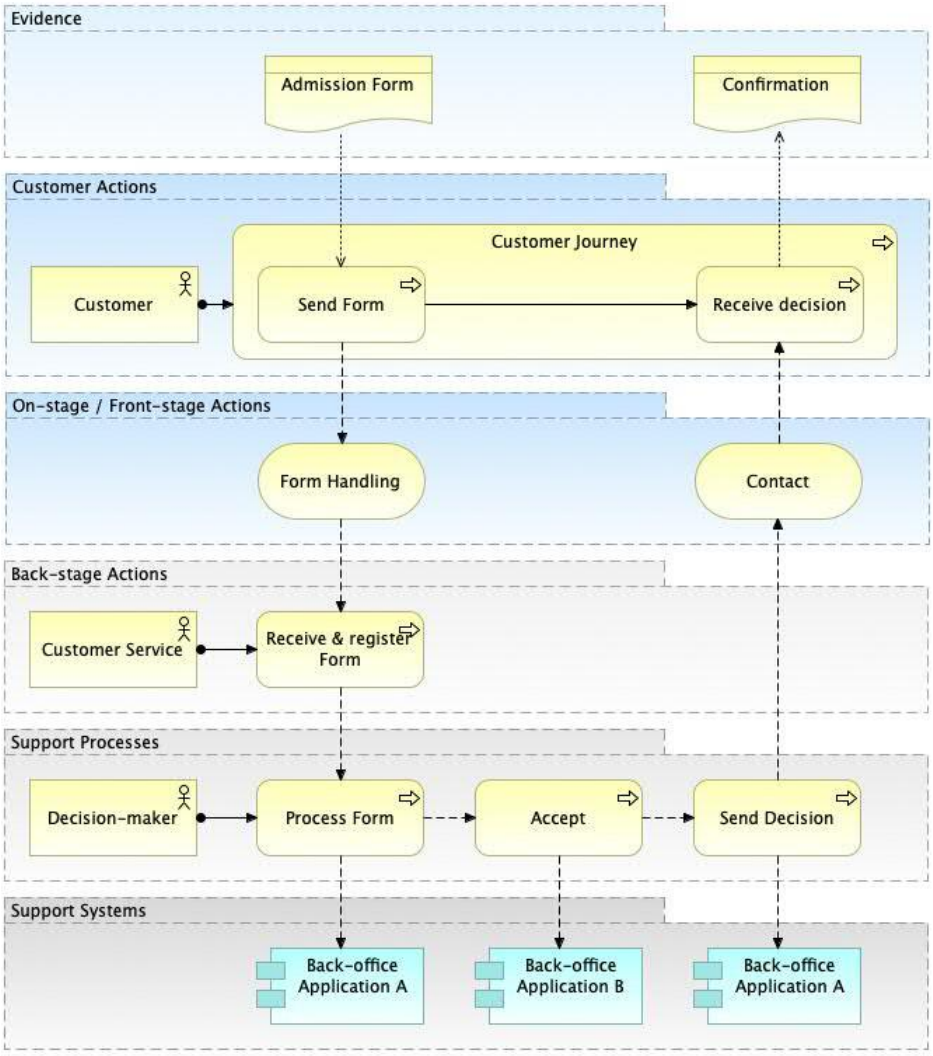


Figure 29: Service Blueprint View 1 (services & flows).

This viewpoint is customer- and service-centric, but it emphasizes also the "inside-out" part of the service. With the help of this approach, the service-driven development can identify the underlying behavioral and structural impacts of the service that is to be designed. As such, this viewpoint complements the customer-experience driven approach with process- and functional aspects.

There are several variations of this view. This example above focuses on information flows between the layers and elements.

5.10 User Story View

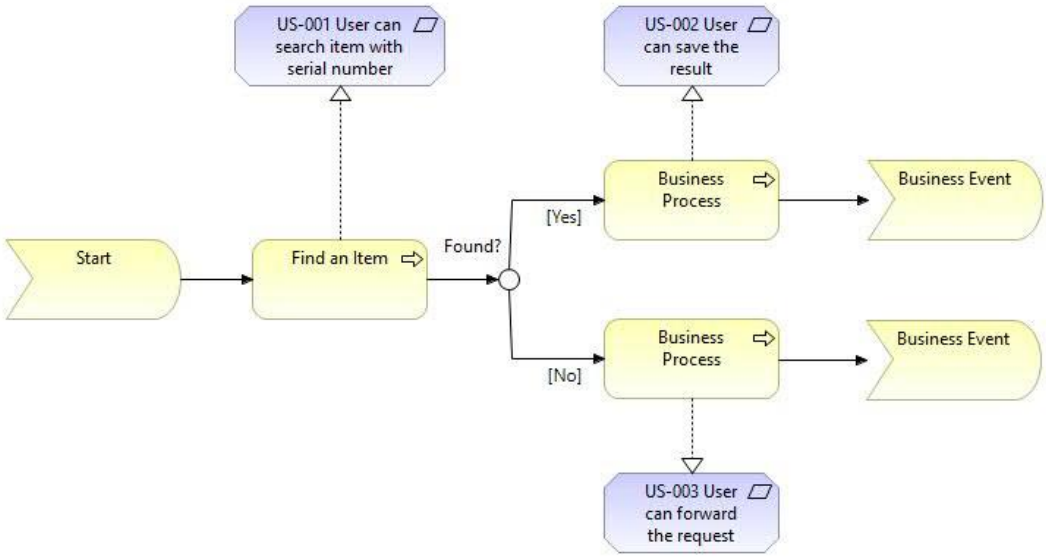


Figure 30: User Story View.

This view can be used to visualization of user stories.

5.11 Cloud-Service Models View

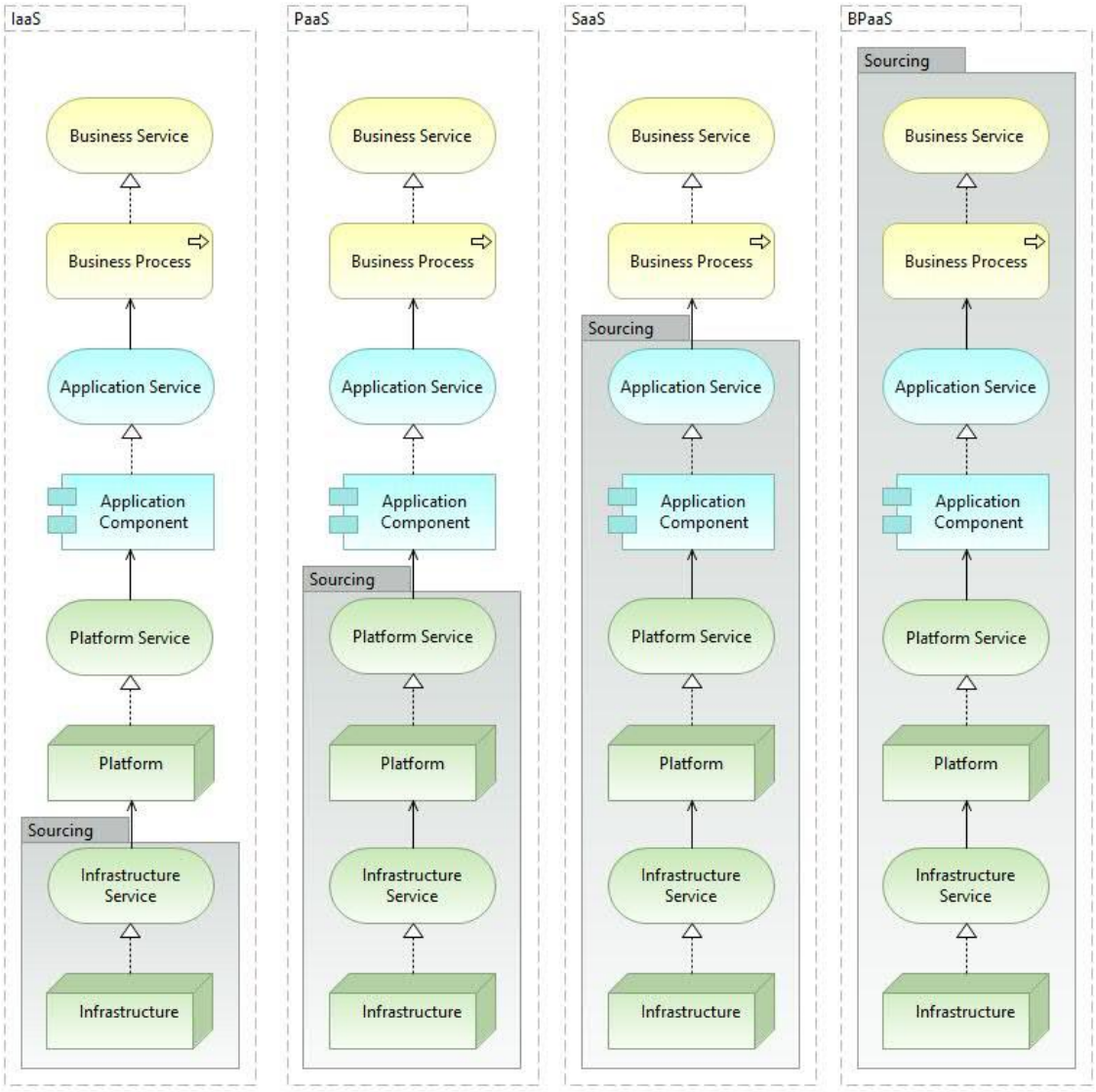


Figure 31: Cloud Service Models View. The "patterns".

5.12 Information View

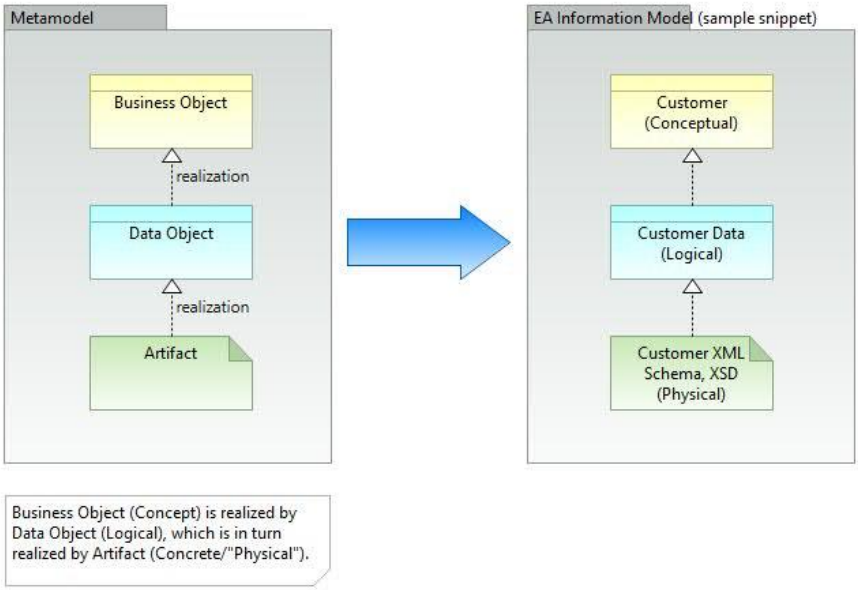


Figure 32:Information View.

Information can be modeled on different abstraction levels as follows: a) conceptual, b) logical and c) physical levels. The diagram above illustrates these abstraction levels.

5.13 Conceptual Data Model View

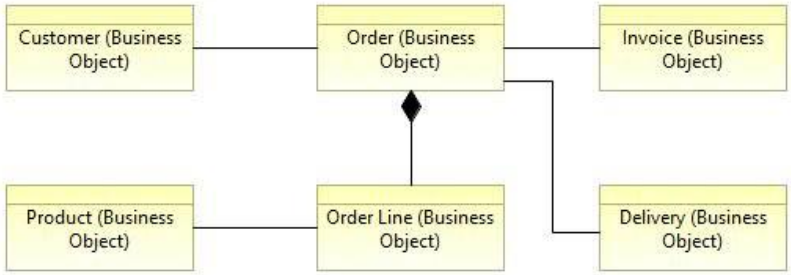


Figure 33: Conceptual Data Model View.

Information architecture of EA contains business objects a.k.a. concepts, that are used in business processes. These concepts and their relations can be represented in a conceptual data model.

5.14 "Service" Concept

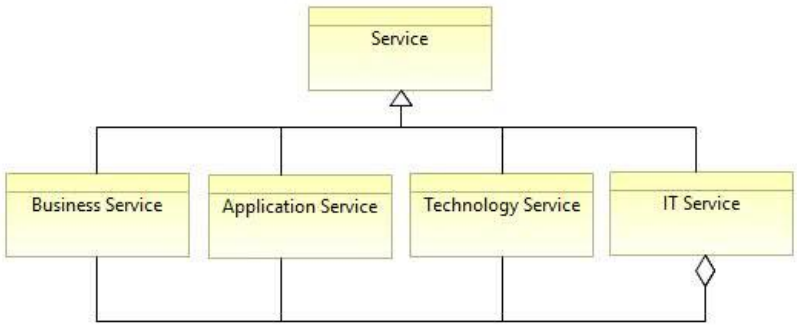


Figure 34: Service Concept.

The service-concept is quite often problematic, as can be understood in many different ways. To make clear distinction which service type is in question, good practice is to mention the prefix: business-, application- or technology service. The IT Service is related to production service according to ITIL. As such. IT Service maps to the Application Services the most.

5.15 Service and Product

Product -concept can be used as a composite element for grouping services. According to ArchiMate -specification:

"A product represents a coherent collection of services and/or passive structure elements, accompanied by a contract/set of agreements, which is offered as a whole to (internal or external) customers."

"A product may aggregate or compose business services, application services, and technology services, business objects, data objects, and technology objects, as well as a contract. Hence a product may aggregate or compose elements from other layers than the Business Layer. "

"A value may be associated with a product. The name of a product is usually the name which is used in the communication with customers, or possibly a more generic noun (e.g., “travel insurance”)."

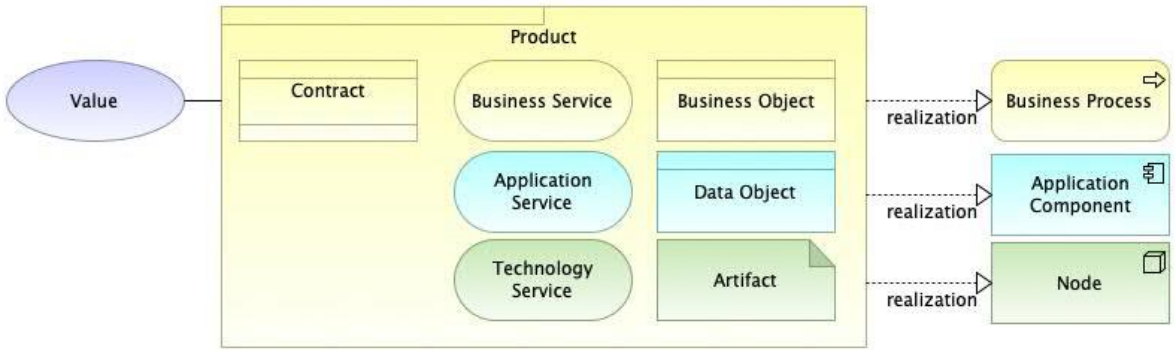


Figure 35: Product View.

6 Application Views

Application Architecture Layer Views.

6.1 Application Services Map View

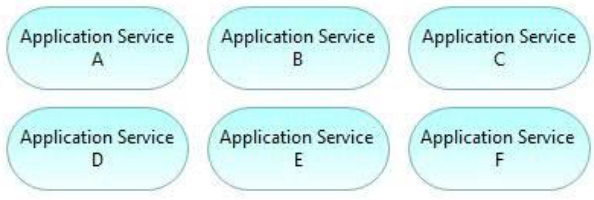


Figure 36: Application Services View.

6.2 Applications Map View

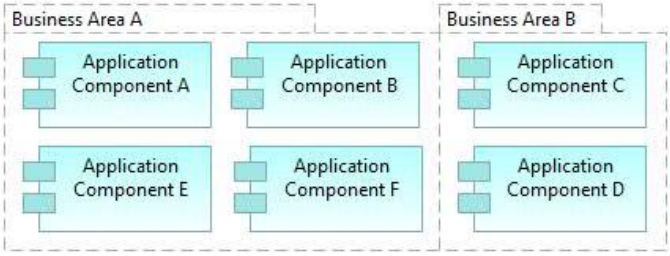


Figure 37: Applications Map View.

Application portfolio, in which applications can be divided into groups e.g. based on business units.

6.3 Application Co-Operation View (Data flows)

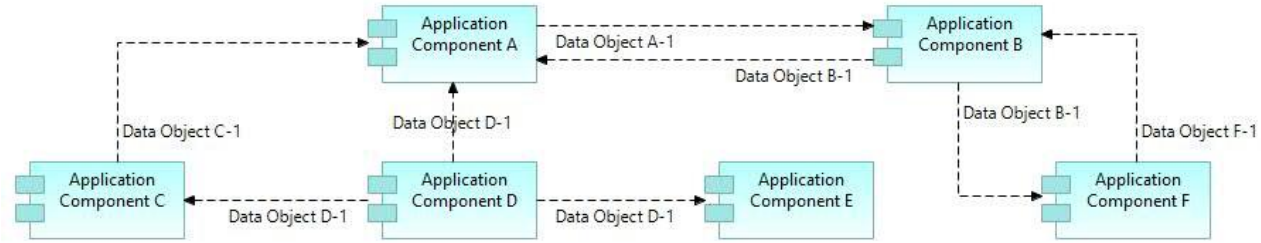


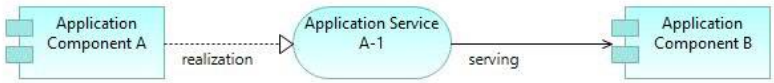
Figure 38: Application Co-operation View.

6.4 Application Integration View (Dynamic relationships)

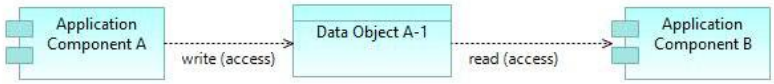
Several alternative approaches of modeling data switching between applications are shown in the examples (1 to 10) below.

- “Application A” owns a data object “A-1”, which is requested by the “Application B”.
- Data flows from “Application A” to “Application B”.
- “Application A” realizes a service “A-1” that is used by the “Application B”.
- Practically, “Application B” requests the Application Interface “A-1” and gets response…

1.
"Application A" provides an application service "A-1" that is used by "Application B".
This illustrates the dependency between the applications.
Structural relationships "Realization" and "Serving" are used.

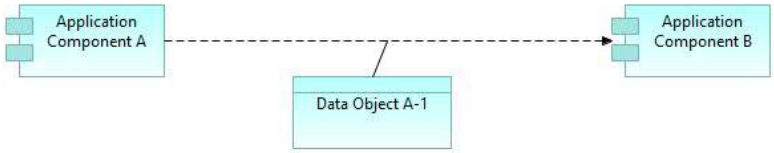


2.
A data object "A-1" flows from "Application A" to "Application B".
The data object is explicitly modeled, and dependency relationship "Access" is used.



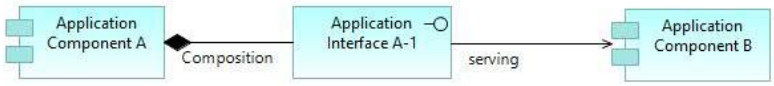
3.
The data object "A-1" flows from "Application A" to "Application B".
The name of the data object is marked as the name (label) of the flow connection between the application components.
Dynamic relationship "Flow" can be used in the cases when there are many data flows in the same diagram - for simplicity and better readability.

4.
A data object "A-1" flows from "Application A" to "Application B".
The data object is explicitly modeled, and related to "Flow" relation between the applications A and B.
This "relation to relation" is new feature of ArchiMate 3.0.

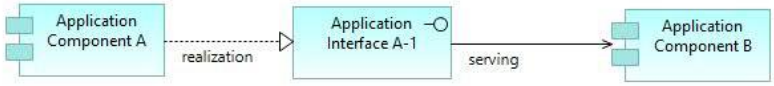


5.
"Application B" is the active part of the communication by initiating the interaction with "Triggering" dynamic relationship.
The data object "A-1" flows from "Application A" to "Application B".
Use of dynamic relationships: "Triggering" and "Flow".

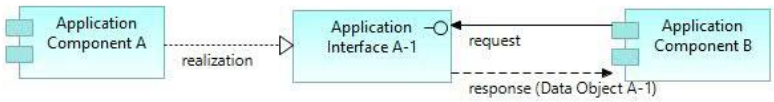
6.
"Application A" has an Interface "A-1" that is serving "Application B".
Use of structural relationship "Composition" and dependency relationship "Serving".
This is in line with ArchiMate specification version 2.x.



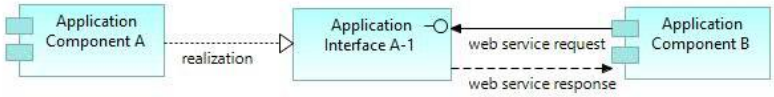
7.
"Application A" realizes an Interface "A-1" that is serving "Application B".
Application Interface "A-1" is structurally part (composed) of "Application A", but the relation can be modeled by using "Realization" relationship according to ArchiMate specification version 3.x.



8.
"Application A" realizes an Interface "A-1" that is requested by "Application B". Interface "A-1" responses to "Application B".
Use of dynamic relationships: "Triggering" and "Flow" together with structural relationship "Realization".
This is in line with ArchiMate specification version 3.x. This can be used when modeling more detailed synchronous integrations (such as request-reply mechanisms).



9.
This is simplification of the previous example, just showing the mechanism / protocol that is used when "Application B" is requesting the Interface "A-1" that is realized by the "Application A".
This is analogous to UML, as reading direction is from the caller application against the Interface / Application that is used.



10.
Application A" realizes an Interface "A-1" that is requested by "Application B". Interface "A-1" responses to "Application B".
This approach might be the most informative when using "Application Interface" instead of useing "Application Service". Using "Application Interface" is the most suitable to model more detailed dynamics between the applications.
Using "Application Service" can be used, instead, to model highl level behavioral dependencies between the applications.
Analogous to approach no 8.
Use of structural relationship "Composition" and dynamic relationships: "Triggering" and "Flow".

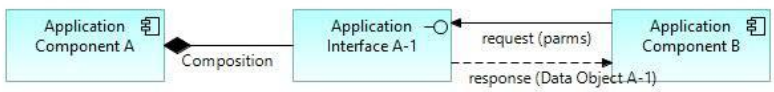


Figure 39: Application Integration View.

6.5 Application Structure View

This view is useful in designing or understanding the main structure of an application and its sub-components and the associated data. This diagram can be used e.g. to break down the structure of the application system under construction, to illustrate modularization / decomposition: what are the sub-systems / sub-components what are the application services (or application interfaces) they provide.

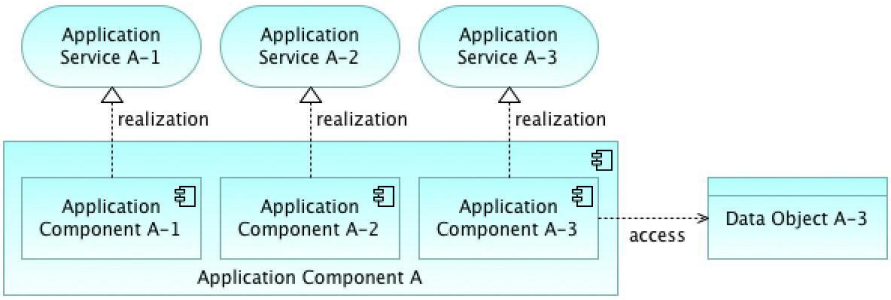


Figure 40: Application Structure View.

Note that application services (figure above) are the behavioral functionalities that are provided by the structural interfaces (GUIs and/or APIs in the figure below). Application Services and Application Interfaces are the "different sides of the same coin".

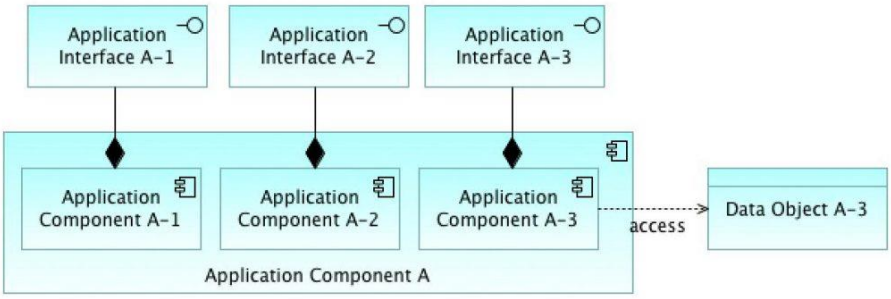


Figure 41: Application Structure View 2.

6.6 Application Architecture View

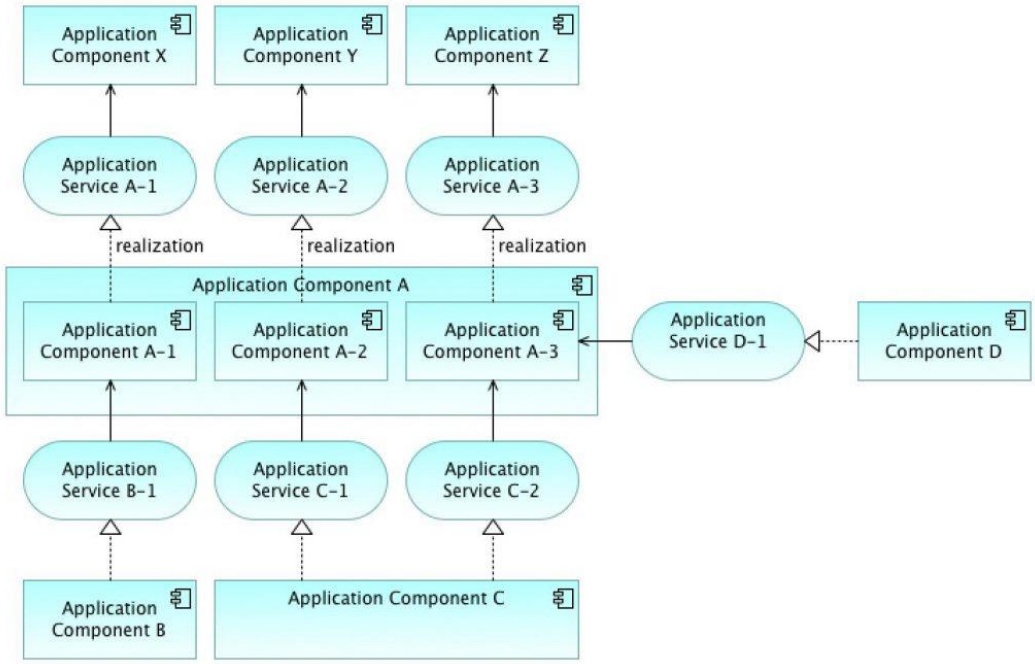


Figure 42: Application Architecture.

This view mixes EA level and solution level approaches, as there are are both applications and application modules in the same view.

6.7 Application Component Model (CM)

Application Component Model 0-n is an application architecture modelling approach, which consists of diagrams of different abstraction levels as follows:.

- At CM-0 -level the diagram describes how the application interacts with its environment, what are the interactions with adjacent applications and users. The target application is depicted as a *black-box*.
- At CM-1 -level the target application is decomposed into modules (main components), and what application services (or application interfaces) those modules provide and require. The target application is depicted as a *white-box*.
- At CM-2 -level the modules are decomposed into sub-components. (Number of necessary levels depends to what is appropriate on the case)

The Application Component Model (CM) diagrams below consist of application components and application services. Alternatively, application interfaces can be used instead of application services depending on the case. As always, it is important to utilize such a modelling style what is appropriate for the purpose, and model only those elements that are informative enough and provide certain added value. It is up to modeler, whether he or she likes to emphasize the functional aspects, or to be more concrete, and model e.g. the actual interfaces with exact naming.

Component Model diagrams below consist of application components and application services. Alternatively, application interfaces can be used instead of application services.

6.7.1 Application Component Model - 0 (CM-0)

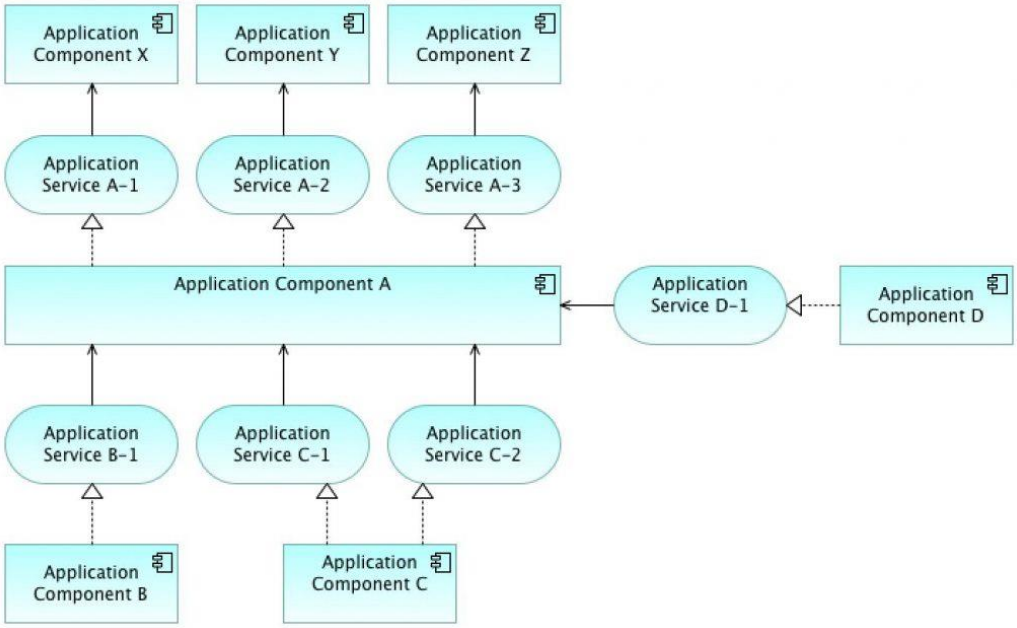


Figure 43: Application Component Model - 0.

Component Model - 0 (CM-0) level (above) illustrates interactions between target application and adjacent applications. All the relevant application services (or application interfaces) are introduced. The 0-level the diagram consists of enterprise architecture level components and their services, target application is in the middle.

6.7.2 Application Component Model - 1 (CM-1)

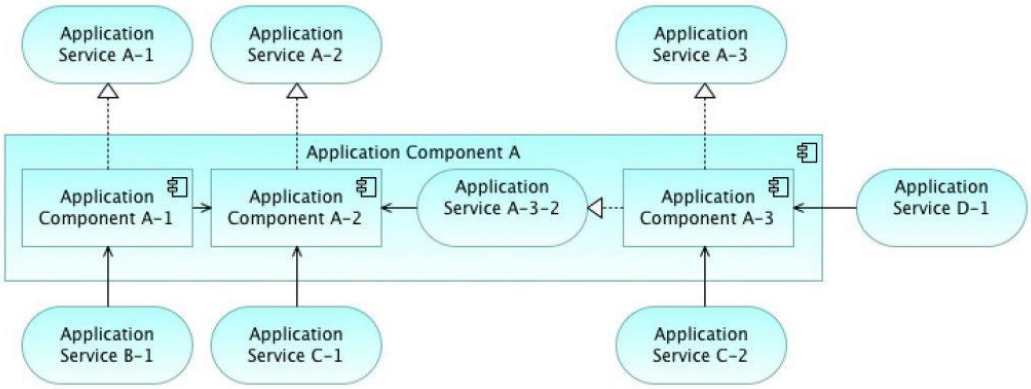


Figure 44: Application Component Model - 1.

Component Model - 1 (CM-1) level (above) illustrates how the target application is decomposed into modules (or main components), and which module realizes which application services (or application interfaces). Note! External applications can be left out from this level, but their services (or interfaces) are shown. When more low-level elements are shown, then more high-level elements can / have to be left out - for the sake of simplicity: to keep the diagram readable.

6.7.3 Application Component Model - 2 (CM-2)

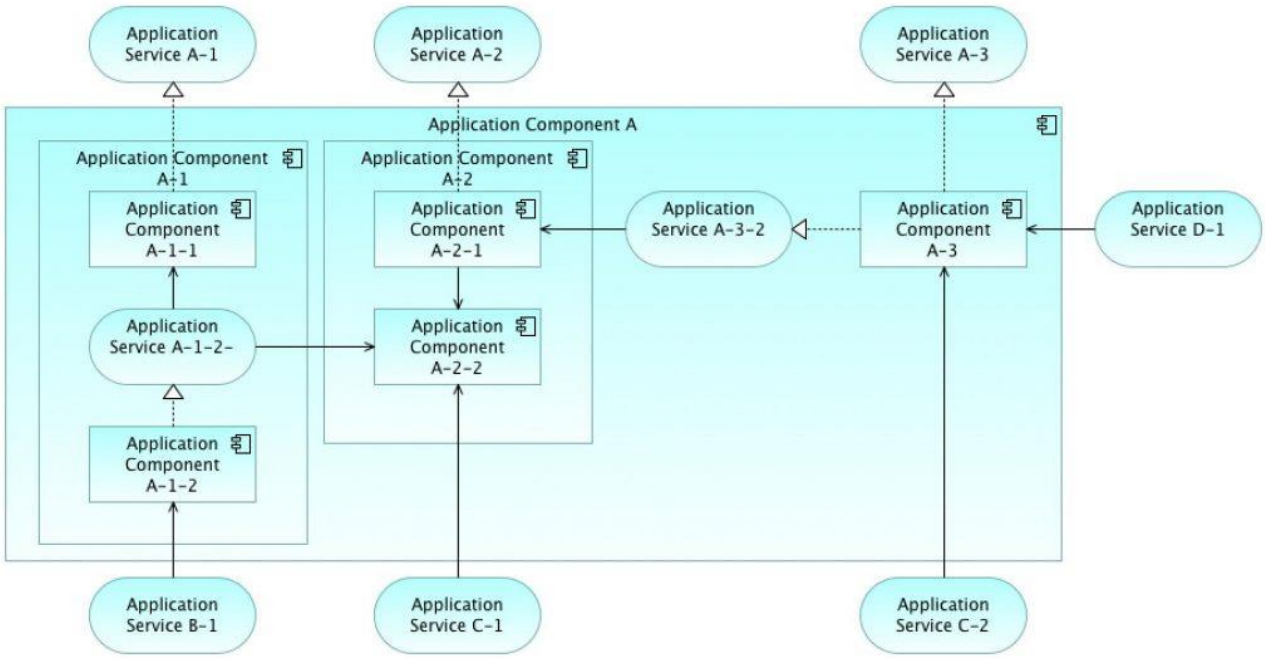


Figure 45: Application Component Model - 2.

Component Model - 2 (CM-2) level (above) illustrates how target application's modules are composed from sub-components, and how they interact.

6.8 Application Functions View

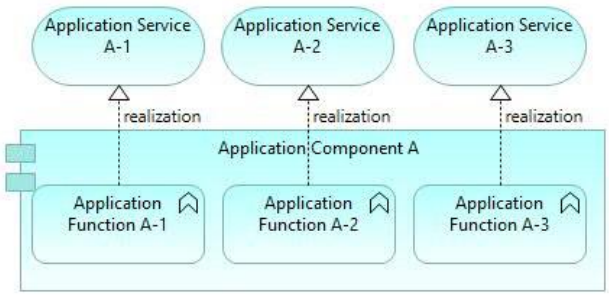


Figure 46: Application Functions View.

Application functional decomposition: what are the functions the system contains, and which application services they provide?

6.9 Application Process View

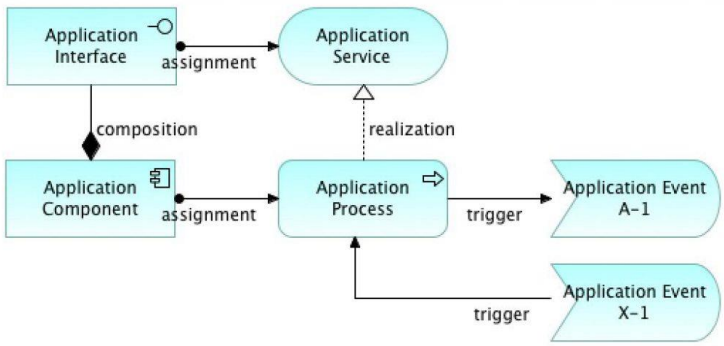


Figure 47: Application Process View.

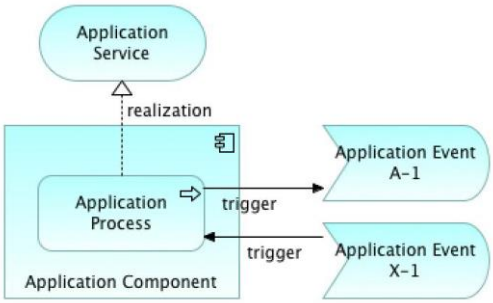


Figure 48: Application Process View - nesting.

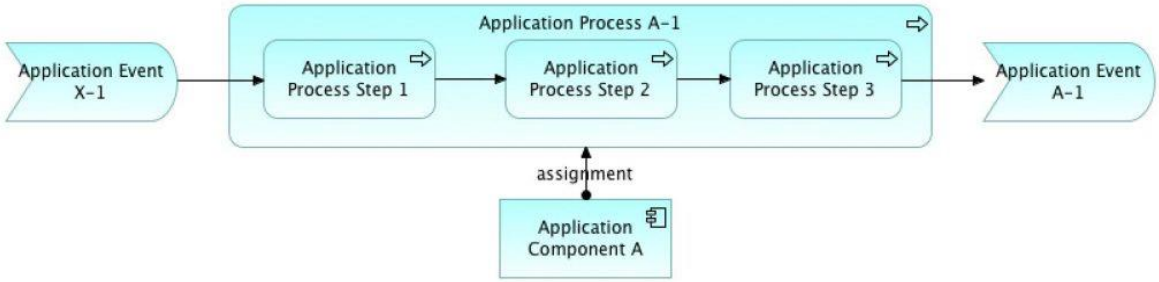


Figure 49: Application Process View - internals.

6.10 Application Component Sequence Diagram View

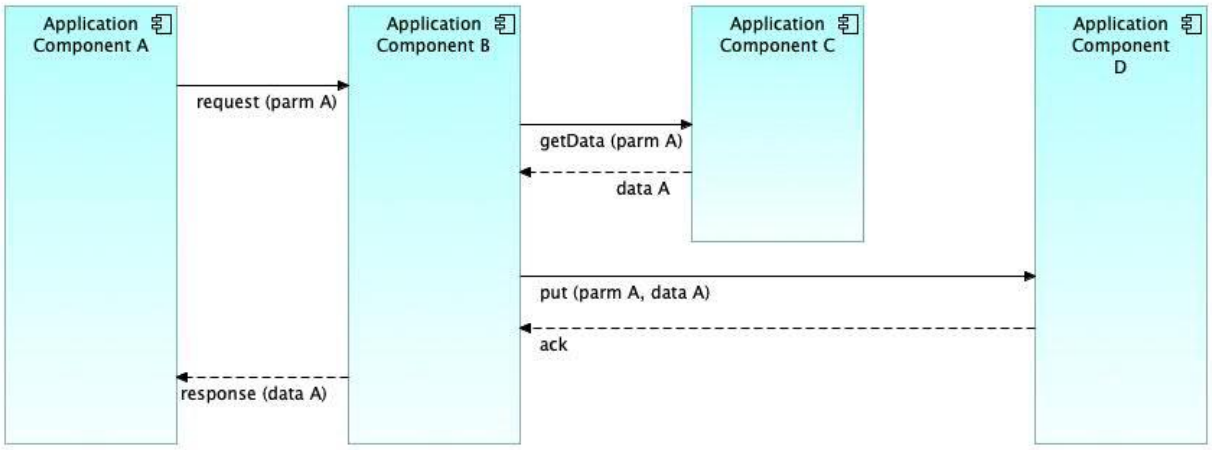


Figure 50: Application Sequence View.

Dynamic relations "Trigger" and "Flow" can be used for modelling dynamics between application components. The layout of this view can be positioned analogously to the UML sequence diagram.

6.11 ETL-process View

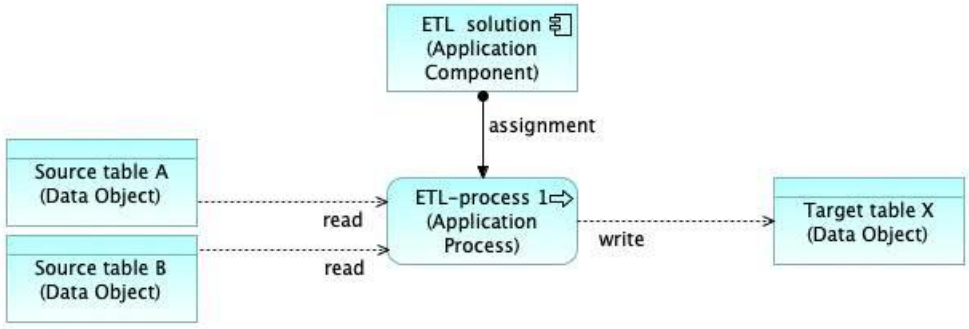


Figure 51: ETL-process View.

6.12 Layered View

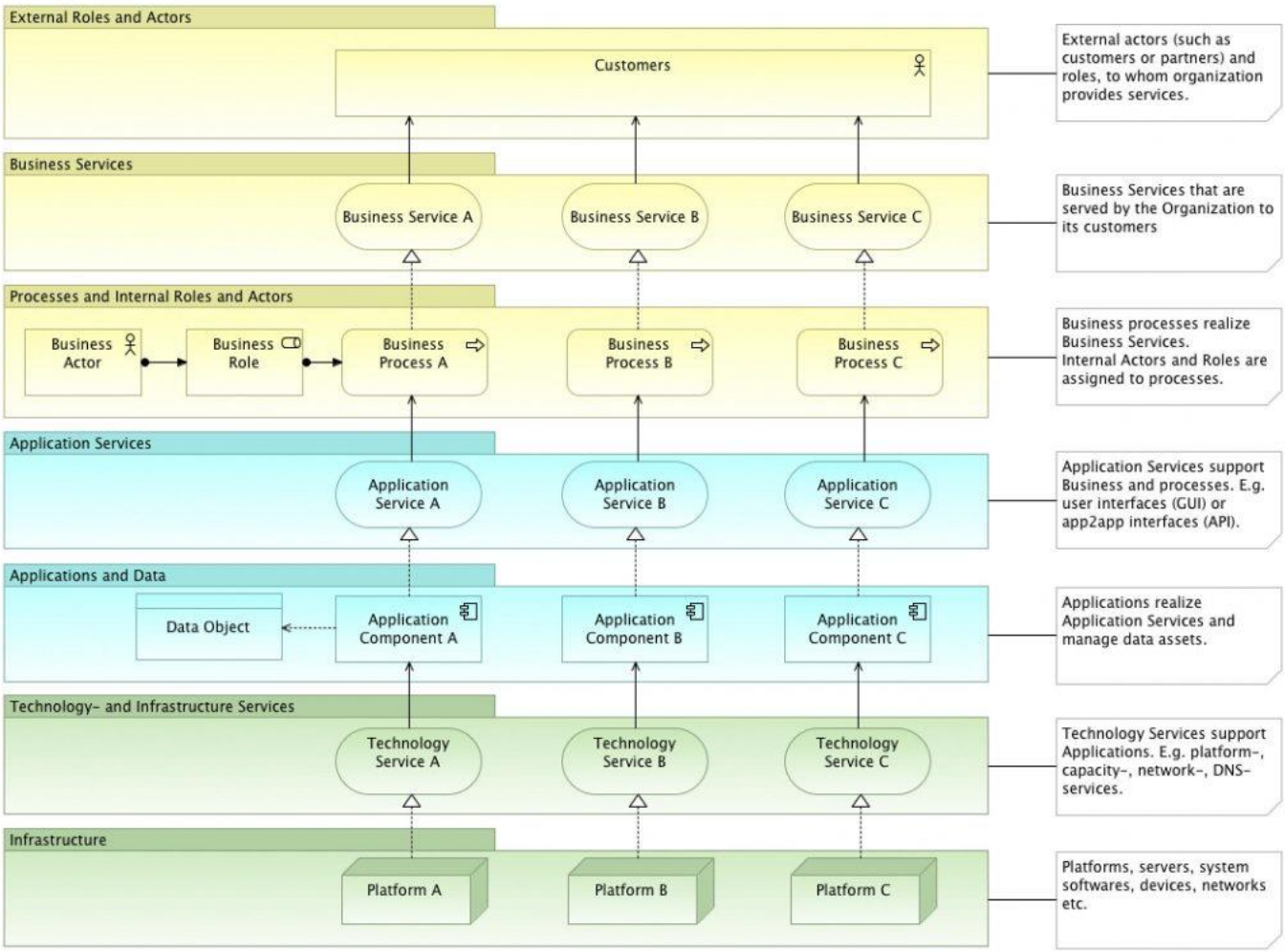


Figure 52: Layered View.

The Layered View can be used as an overview context diagram of a target area. The main advantage of this view is to illustrate the usage of applications in business processes and services they provide.

7 Technology Views

Technology Architecture Layer Views.

7.1 Infrastructure View

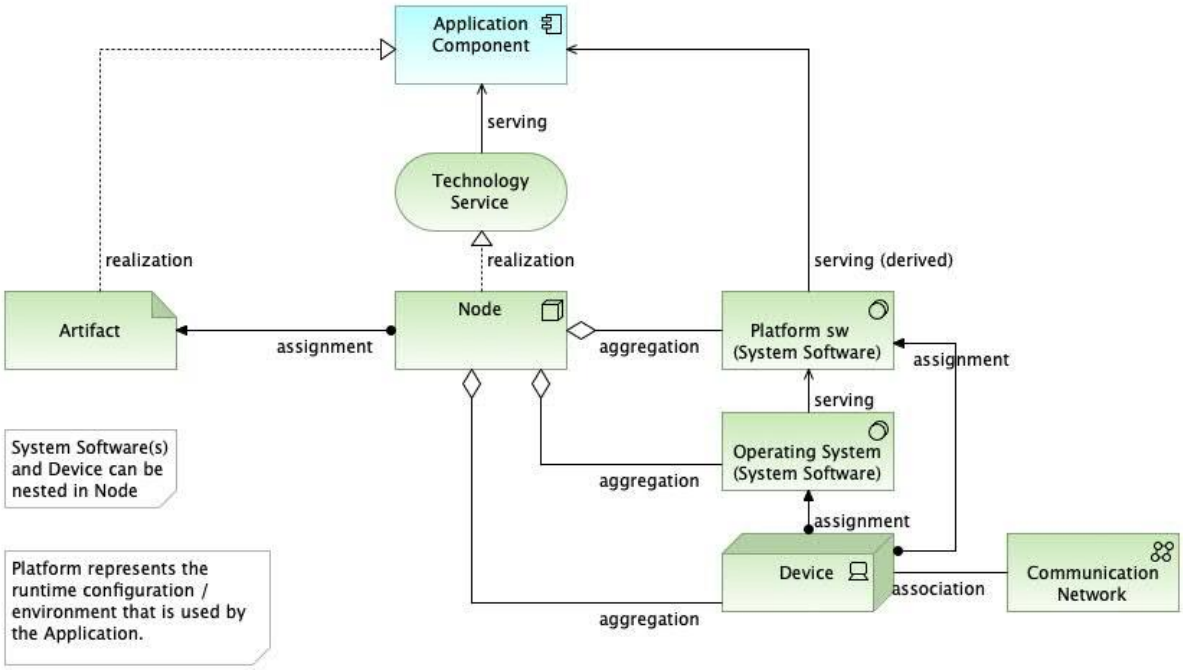


Figure 53: Infrastructure View.

This view represents a platform of an application. This pattern can be used to model a configuration of run-time environment, the deployment of the business application.

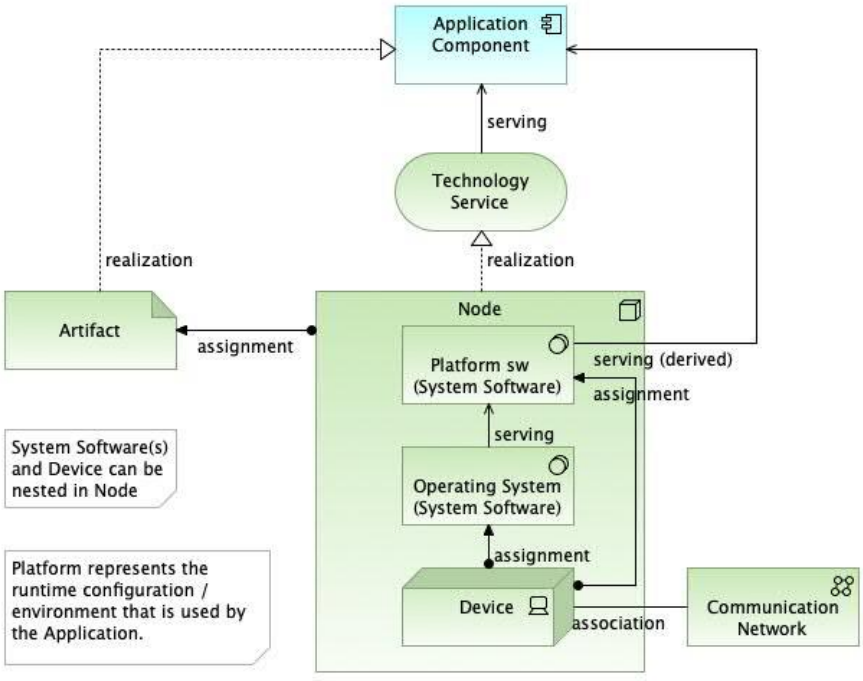


Figure 54: Infrastructure View (nesting).

8 Implementation and Migration Layer / Transformation Architecture Layer Views

8.1 Implementation Roadmap View

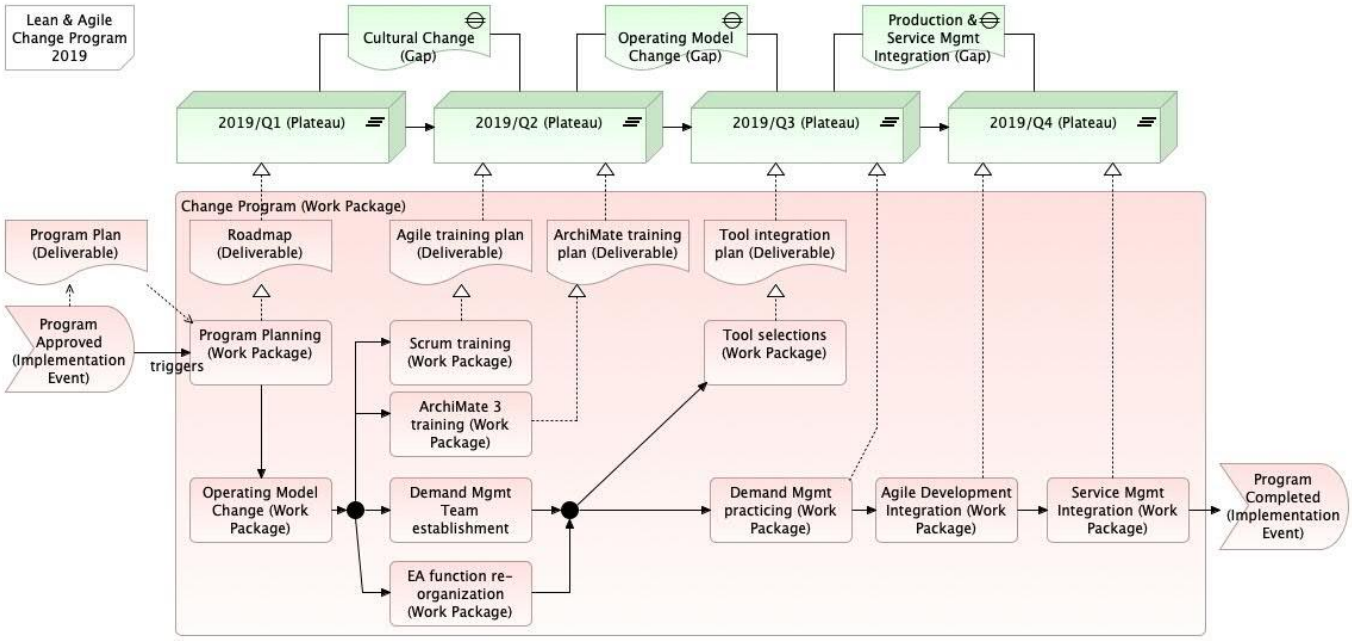


Figure 55: Implementation Roadmap View.

8.2 Kanban View

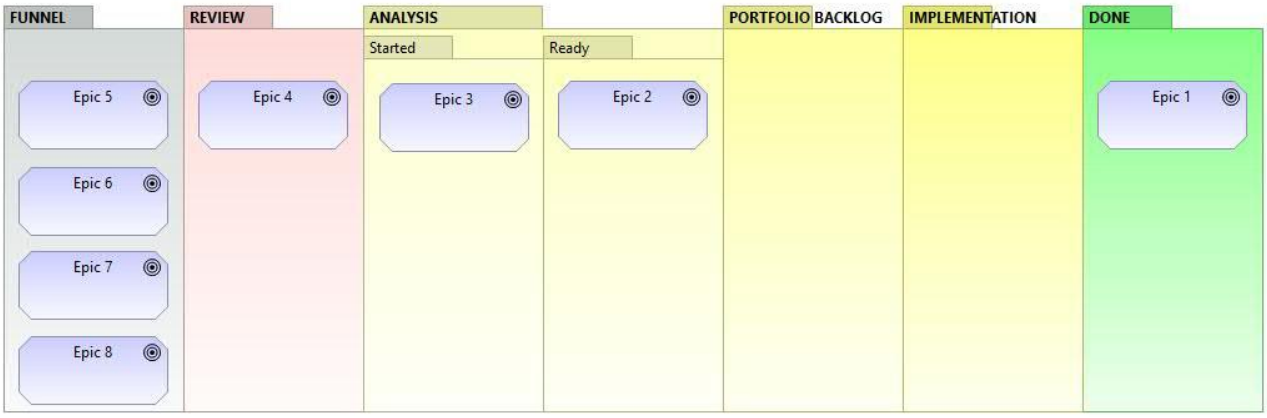


Figure 56: Kanban Board (EA)..

Kanban board can be used for visualization of work and the workflow. Kanban board shows how e.g. development requirements, epics, user stories etc. are flowing from backlog to ready state (Done). Kanban board can be applied for diverse purposes depending on the scale and scope of the development case. E.g in EA level by using "Epics" or in project level by using "User Stories" or "Requirements" as work items. The granularity of work items can vary depending on the case.

8.3 Generic View

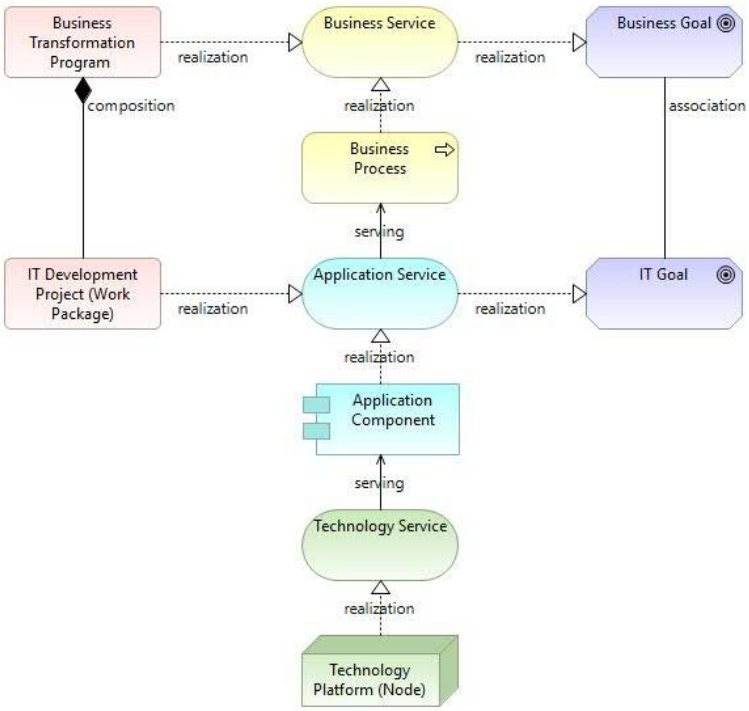


Figure 57: Generic View.

This kind of simplified view can be used e.g. as contextual diagram of specific service, program or project.

9 Extras

9.1 Co-operation View

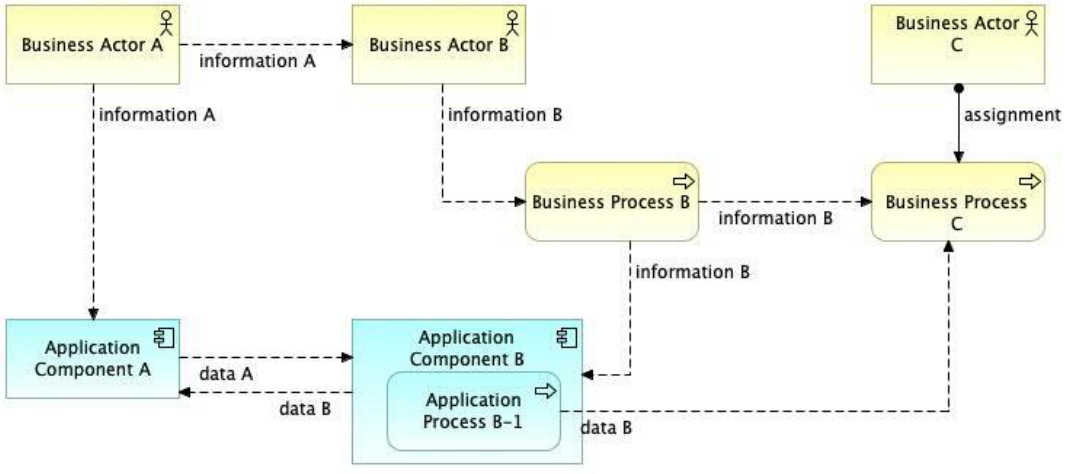


Figure 58: Application Co-operation View (Extended).

It is possible to mix the layers as shown in this data flow diagram example above.

9.2 Metamodel

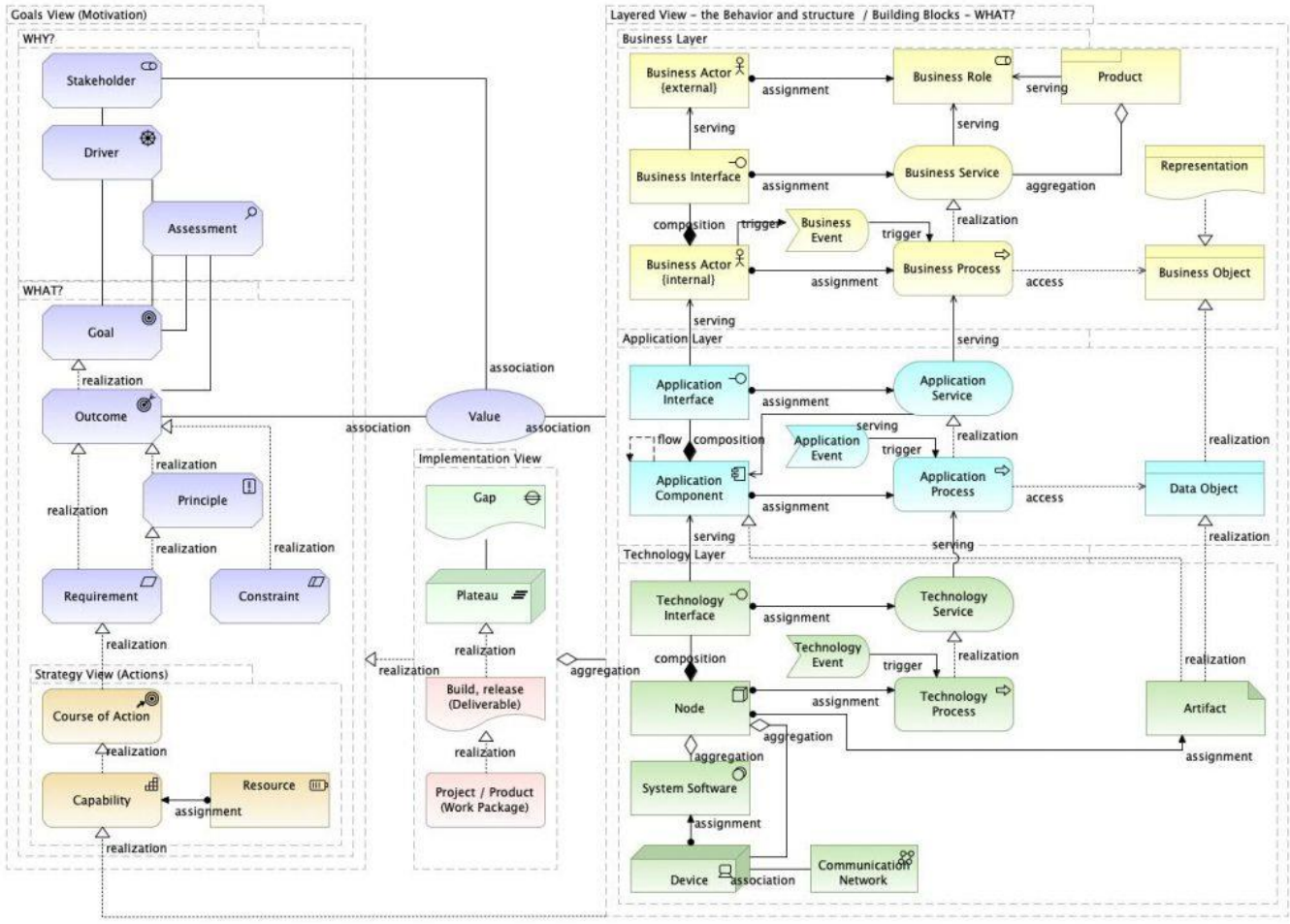


Figure 59: Metamodel.