ARCHIMATE®
FOUNDATION

ArchiMate Made Practical

Modeling according to ArchiMate guided by a collection of good practices

Colofon

Title :	ArchiMate Made Practical
Date :	17 november 2007
Version :	2.0
Change :	First translation of Dutch 2.0 version
Status :	Final
Editor :	Hans van Drunen, Egon Willemsz
Company :	ArchiMate Foundation
Authors :	Harmen van den Berg, Hans Bosma, Gertjan Dijk, Hans van Drunen, Jan van Gijsen, Frank Langeveld, Joost Luijpers, Thé Nguyen, Ger Oosting, Robert Slagter, Egon Willemsz
Translation:	Louis Dietvorst

Table of Contents

1 Introduction

1.1 Motive

How do I model an application landscape? How do I relate logical (implementation independent) and physical business processes (implementation dependent)? How do I match the application portfolio with the technical infrastructure? How do I visualize what data are required when delivering which products and services to customers? How do I model a service broker?

These are all examples of modeling questions that an enterprise architect will be confronted with. There are many variants of modeling solutions that are being practiced in order to answer these questions. These variants lead to inconsistencies and can limit good communication. Behind every model there is an original story, but a model can often converge into its own new “truth” where the original story will get lost over time, especially if the originator of the model is no longer involved in the maintenance of that model.

Companies that practice ‘working under architecture’ are using ArchiMate more and more as a descriptive language for capturing enterprise architectures consisting of domains like: products/services, processes, organization, data, applications and technical infrastructure. Using this language it is possible to model the structure *within* a domain as well as the relationships *between* domains. Another important aspect of ArchiMate is the possibility to visualize models using multiple viewpoints, specific to different stakeholders. Besides this, the language supports a formal foundation to enable analysis of models (for example on an aspect such as performance). ArchiMate is deliberately not intended to replace existing languages like BPMN, UML etc.; it is positioned to deliver value in addition to these existing languages, the intention is to adapt as much as possible to existing standards and practices.

In the Netherlands, a lot of experience with ArchiMate has resulted in recognizable models that are easier to communicate. The population of enterprise architects that use ArchiMate is growing. The downside to this is that the solution to identical modeling problems is created multiple times, where it is obvious that not all practices deliver identical value when being communicated to the stakeholders. Practically speaking, one has to be well skilled to capture and practice the language. The learning curve to start using it immediately or without being trained is relatively high.

1.2 Goal and Intended Public

The ArchiMate Foundation has received a considerable number of questions about how ArchiMate concepts should be adapted from people using ArchiMate to perform real work. The aim of this document is to start answering these questions by describing proven solutions, documented as good practices., This will lower the threshold for applying ArchiMate in practice and will increase the effective value of the developed models.

This document is written for enterprise architects who want to apply ArchiMate in practice as a language for capturing models of organizations, the ICT-support and the relationships between them.

This document should be treated in addition to existing ArchiMate documentation (see the appendix for references):

- Enterprise Architecture At Work.
- Inleiding in de ArchiMate-taal.
- Architecture Language Reference Manual.
- Viewpoints Functionality and Examples.
- ArchiMate Quick Reference

1.3 Reading Guide

The following chapters contain a collection of good practices. For each documented good practice, the following structure is used:

- **Name in the paragraph title**: a recognizable name that summarizes the contents of the good practice
- **Question:** a description of the question that is leading to the development of the good practice
- **Solution:** a description and visualization of the preferred solution that gives an answer to the question
- **Consequences:** a description of the consequences of the solution, for example in relation to communication with stakeholders, consequences for implementation, etc.
- **Alternatives:** a description of alternative solutions with the justification for each alternative
- **Relationships with other good practices:** a description of the relationships that exist with other good practices

In this document, two types of good practices have been collected:

- Good practices that give an answer to conceptual ArchiMate questions (how does one use a certain concept in practice?).
- Good practices that give an answer to modeling questions.

The good practices are organized as follows:

- Modeling within the business layer: chapter 2.
- Modeling within the application layer: chapter 3.
- Modeling within the infrastructure/technology layer: chapter 4.
- Modeling across boundaries: chapter 5.

1.4 Changes since version 1.0

This document is a renewed version of the booklet ‘ArchiMate in de praktijk’ which was first published in 2007 by the working group Usage/Tools within the ArchiMate Foundation. In this version 2.0, some existing best practices have been modified and some new best practices have been added.

The modified good practices are:

- Interface between applications.
- Functionality of an application.
- Service broker.

The new good practices are:
- Domain.
- Internal and external services.
- Necessity for using services.
- Simplifications with sustainable consistency.

1.5 Website

More information about the ArchiMate Foundation, the activities of the working group Usage/Tools and the collected good practices can be found on the website: www.archimate.org. On this website, new good practices will be published before these will be integrated in a next version of this booklet.

1.6 Call for new good practices

The contents of this document is by no means complete and leaves a lot of questions unanswered. This document is therefore designed to be a living document. It contains an initial overview of good practices collected from day-to-day practice.

We hope the reader will be able to get started with these good practices, and that new insights will be revealed which can of course be integrated into this document.

Do you have any modeling questions that you would like answered? Do you have good practices that you want to share with your peers? Or do you want to participate in the working group Usage/Tools within the ArchiMate Foundation, where we collect and discuss good practices and integrate them into this document? You can contact the chairs of the working group via the email address: usage@archimate.nl Or by telephone, contact Harmen van den Berg (+31 6 5119 8282) or Egon Willemsz (+31 6 5235 3089).

2 Modeling in the business layer

2.1 Overview of good practices

In this chapter, the following good practices are described:

1. Business services, business function and business process.
2. Constraining business service.
3. Constraining business function.
4. Constraining business process.
5. Business process and business interaction.
6. Viewpoints for process modeling.

2.2 Business service, business function and business process

Question: In the business layer, the concepts business service, business function and business process are positioned. The exact difference between these concepts is not clear in all cases, or to be more exact, in practice there sometimes exists confusion if something needs to be tagged as a business service, a business function or a business process. In the next section we give definitions and descriptions of these concepts.

Solution: A business service represents the added value that an organization delivers to its environment. One can make a distinction between internal and external services: Internal services represent the added value that is being delivered within the domain that the service belongs to. External services represent the added value that is being delivered to other domains or to the environment (for example customers).

A business function is an area that the organization wants to pay attention to (e.g. by putting energy into, structurally committing resources to etc) in order to support its business goals. A business function can therefore be positioned as a grouping of internal behavior based on a certain criteria (for example location (same department), communication, required skills, shared resources and shared knowledge). A business function represents a part of the added value of on organization.

A business process is a unit of internal behavior or a collection of causal (sequence or dependency) related units of internal behavior, with the goal of producing a predefined collection of products and services. A business process can be constructed from sub processes or activities. A business process is triggered (started) by one or more business events or other business processes.

Informally one could say that a business process consists of a number of activities or sub processes that are being executed in a certain sequence. Every activity is part of a business function. With other words, a process combines a chain of activities each of which are part of business functions, such as the figure below visualizes.

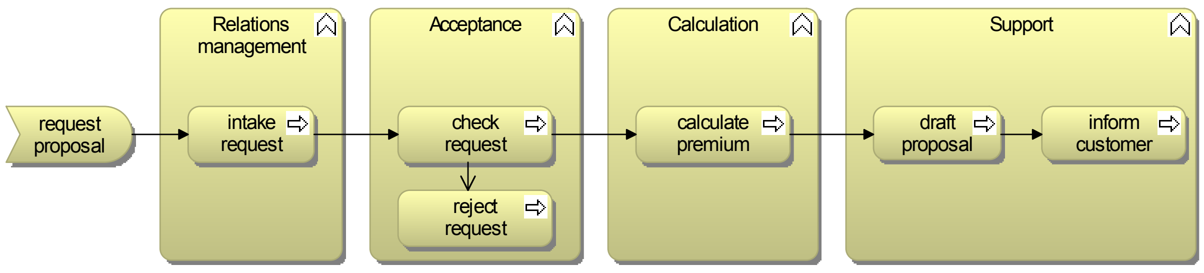

Figure 2-1: Example of business functions in relation to processes

A single process will not always belong to a single business function (as in this example): a business function will almost always consist of multiple activities and process steps and a process will often be realized by multiple business functions.

Business functions as well as business processes describe the inner way of working of an area or organization. A business service describes the parts of business processes and functions that are externally visible and usable. It describes the service that can be used, without making clear how that service is realized by means of processes or business functions.

In the figure below, the most important relationships between these concepts are visualized (being a part of the ArchiMate metamodel):

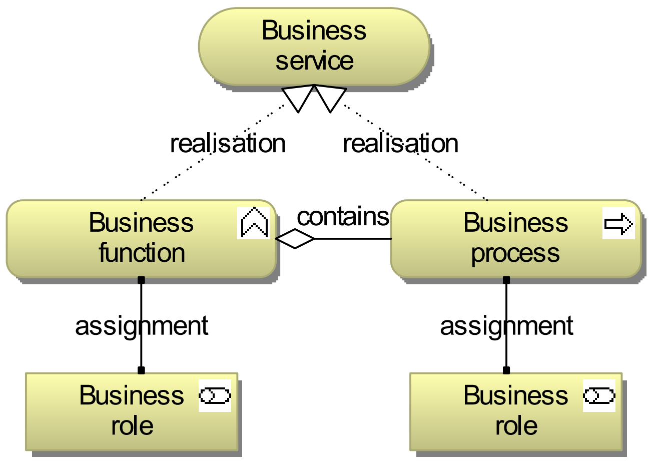

Figure 2-2: Most important relationships between concepts

Consequences: Not applicable.

Alternatives: Not applicable.

Relationships with other good practices: The distinction that exists between business service and business process / business function is also valid between application service and application function. Comparable criteria and heuristics can therefore also be used on the application layer.

2.3 Constraining business service

Question: In the business layer, one of the concepts is the business service. But how does one distinguish a business service and how is it being constrained?

Solution: If something is tagged as a business service, than that service should in principle be “consumable” by multiple consumers. Services can be grouped by means of aggregation relationships. This implies that it is only sensible to decompose a service (via aggregation) into partial services if these partial services can also be consumed independently of each other.

Apply the following heuristics to distinguish business services:

- Distinguish a service from the perspective of its consumer. The service should be recognizable and usable for the consumer. The naming convention should be done from the perspective of the service consumer.
- Distinguish services based on the activities that are executed in the business layer, and based on the products that are being delivered.
- Distinguish different services optimized to support different concerns.
- Prevent overlap in offered services: different services offer different behavior. Overlap in behavior is an indicator that you need to define the overlapping behavior as a separate service.
- A service is realized by one or more business functions or processes that represent concrete internal behavior of the organization. A business function can realize multiple business services.
- Always model which business functions or processes a business service realizes and which business processes (in case of internal services) consume the service.
- An internal business service is always used by at least one business function or business process
- Keep services consistent: make sure that comparable behavior is offered as services in a comparable manner
- Use services to hide implementation details. It is sufficient for a service consumer to know *that* a certain service is being offered and how the consumer must use the service. A service consumer does not need to know *how* a service is realized.

Consequences: Not applicable.

Alternatives: Not applicable.

Relationships with other good practices: Identical criteria and heuristics can also be used for application services in the application layer and for infrastructure services in the infrastructure layer.

2.4 Constraining business process

Question: In the business layer, one of the concepts is business process. But how does one distinguish a business process, and how is it being constrained?

Solution: Apply the following heuristics to distinguish business processes:

- A business process is triggered by a business event, and ultimately delivers a service or product to a customer, or partial products or partial services that are used as part of a service or product for a customer.
- Divide a process into phases that can be being treated sequentially. Examples of phases are request, handling and after sales. A phase often coincides with a process or function within the organization. A frequent pattern of phases is the value chain. One can recognize phases by the assignment of actions to different actors, combined with a timing sequence between the actions.
- Group actions based on the time that they happen e.g. online- or batch processing, daytime or nighttime).
- Divide a process into parts based on the knowledge and skills that are required to execute certain actions. This can be deduced from the functions that are tagged for each task.
- Divide a process into parts based on geographical boundary (physical location) where the activities take place, for example a region.
- Divide a process into sub processes that can be executed independently. This can occur where multiple (partial) products are being delivered.
- Distinguish partial processes that occur more than once. These common partial processes can be generalized and reused.
- Distinguish partial processes that are repeated as a whole.

Consequences: Not applicable.

Alternatives: Not applicable.

Relationships with other good practices: Not applicable.

2.5 Constraining business function

Question: In the business layer, one of the concepts is business function. But how does one distinguish a business function and how is it being constrained?

Solution: Apply the following heuristics to distinguish business functions:
- Make a separate function for the *interactions with each environment actor*. For example; create a function for suppliers, consumers and government. This heuristic has the following specializations:
  - *Separating relationship management from other functions*: Make a distinct function for activities that deal with customer contacts. The customer here is the external customer of the company, but for large companies it might also be applicable for an internal customer (for example another department).
  - *Separation to market*: Make a distinct function for each customer segment/target group of the company. Examples of these groups are business customers and private customers.
- Make distinct functions for processes with different kind of *triggers* (business events). In this way processes can be identified that have periodical triggers (monthly receipt of declarations) and that have incidental triggers (quotation request). A specialization is:

    - *Case distinction*: Make distinct functions for activities that are related to different ‘cases’. A company can make distinction between different damage claim cases: damages less than or greater than € 1000,-. In this example, a function for a small and a function for a large damage claim case could be distinguished.
- Make a distinct function for each *product group*, for example for activities related to damage insurance versus other insurance. A distinction can also be made centered around any business object:
    - *Distinction related to business object*: Make distinct functions for a group of activities, if they all work on a certain business object. This way, functions can be distinguished that are related to invoices, cash flows, offers, customer history etc.
- Make a distinct function for each phase or state that a product can be in. In case of a damage insurance claim this could be, for example, distinct functions for requesting, reward raising and handling of damage claims, closing and management.
- Make a distinct function for a group of activities, whenever the activities require special 1) skills, 2) expertise or 3) responsibilities. Examples are judicial knowledge, actuarial expertise, communication skills or authorizations for certain decisions.
- Make distinct functions for activities that *control* the primary process, especially planning and control.
- Make distinct functions for activities that *change* the primary process or its implementation. This leads for example to distinct functions for marketing, product development and system development.

Consequences: Not applicable.

Alternatives: Not applicable.

Relationships with other good practices: Comparable criteria and heuristics can also be used for application functions on the application layer.

2.6 Business process and business interaction

Question: Besides the business process concept, ArchiMate also understands the concept of the business interaction. A business interaction is a process that is executed by two or more actors (a collaboration of actors). Why is the business interaction concept needed on top of the business process concept? Isn’t it sufficient to relate a process to a collaboration of actors and thereby express that the process in itself is in fact an interaction?

Solution: A business interaction is defined as a specialization of a business process (or more precisely a specialization of the generic behavior concept). By using a business interaction, information is added to more explicitly describe the behavior that is executed by a collaboration of roles or actors. By relating a business process to a business collaboration the same information is expressed and it more explicitly describes the behavior that is executed by a collaboration of roles or actors. In the following figure, some variants are expressed:

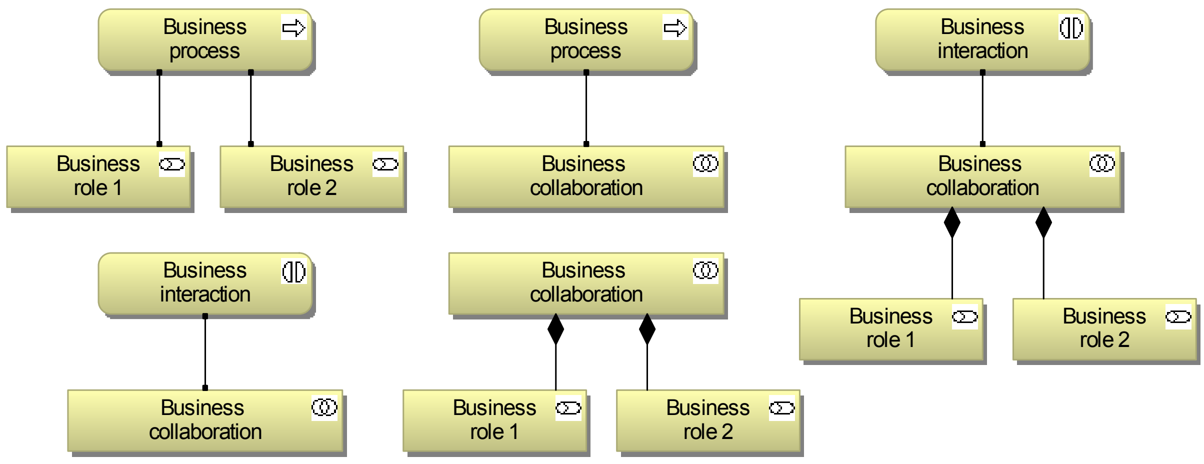

Figure 2-3: Variants of business process and business interaction

In the top left, a business process is modeled that is assigned to two roles, in the top middle a business process is assigned to a business collaboration, bottom left an interaction is assigned to a business collaboration, bottom middle shows a business collaboration consisting of two roles and to the right an interaction that is assigned to a business collaboration consisting of two roles.

This redundancy is not without a good cause. In some cases, a business collaboration is not explicitly modeled (for example if the focus is on the behavior and not on the actors), and at the same time you want to express that it is about an interaction with multiple roles. Another example is if one would want to express a collaboration, but not make explicit which parties or actors the make up the collaboration; this can be useful in, for example, a development phase where it is not yet clear who the final parties will be that ultimately will add value to a product of service.

Consequences: The goal the architect has with modeling or visualizing a business interaction or –collaboration will to a large extent determine which modeling solution is chosen and which aspect gets focus.

Alternatives: Not applicable.

Relationships with other good practices: A comparable distinction as described here is applicable to the application layer, especially related to application interaction and application collaboration. The examples and heuristics shown here are also applicable.

2.7 Viewpoints for process modeling

Question: How do you model a process in ArchiMate? The ArchiMate book shows a number of viewpoints but which combination of viewpoints is required to model a process containing sequential steps?

Solution: The ArchiMate book shows two viewpoints for process modeling: the Business process collaboration viewpoint and the Business process viewpoint. The difference between these viewpoints is that in the Business collaboration viewpoint the relation *between* processes and the relation with the environment is visualized, while in the

Business process viewpoint one business process is expressed. We concentrate here on the Business process viewpoint; the Business process collaboration viewpoint is analogous.

When modeling a process in ArchiMate, the following aspects can be visualized: the parts of the process, their cohesion, the assignment of process parts to roles, actors of business parts and the support by applications. This can be realized in the following sequence:

1. *Model the business process on high level:* Decompose the business process and make relationships by introducing business events and triggering relationships. One can add to that the way in which the process delivers business services to its environment.
2. *Information-exchange:* Model the information-exchange between processes by adding flow relationships between processes, or reading and writing of business objects.
3. *Assignment to organization:* Model the organizational part that executes the business process. This can be visualized by assignment relationships or by using swim lanes.
4. *Support by applications:* Model how applications support business processes. The application viewpoint can be used for this.

If required, a more generic distinction can be made between a ‘logical’ and ‘concrete’ level of modeling. A *logical* model expresses the functional structure and logical cohesion of business processes. No detail about time, place, people or machines is given. An *implementation model* will closely resemble what you can see or want to realize in reality. It describes *physical* aspects like humans and machines, with concrete assigned tasks. If an organization wants to distinguish a standard type of processes (template) in ArchiMate this should be defined before starting modeling. The relationship between ‘logical process’ and ‘physical process’ is of the composite type.

A process can be specialized into process types. For example, the process ‘Request insurance’ can be specialized into the sub processes ‘Request travel insurance’ and ‘Request damage insurance’.

Consequences: Not applicable.

Alternatives: Not applicable.

Relationships with other good practices: Refer to the good practices ‘Business service, business function and business process’ and ‘Constraining business process’.

3 Modeling in the application layer

3.1 Overview of good practices

In this chapter, the following good practices are described:

1. Application landscape.
2. Interfaces between applications.
3. Functionality of an application.

3.2 Application landscape

Question*:* My application landscape is very complex, how can I get grip on it, how can I model it?
An overview of applications is often visualized in an application landscape. This is done to provide an overview of the relationships between applications. This will often produce a large number of connecting lines and actually no overview at all. This kind of overview often only shows how complex the application landscape is. It does not give any assistance in application portfolio maintenance.

Solution: It is important to keep overviews readable. Apply a rule of maximum 10 to 15 applications per overview. If there are more, it is recommended to use a general model containing only a major grouping and a detailed model per group. Choose a viewpoint that is important for the stakeholder you are addressing. The format will always be an Application structure viewpoint. A classification can be done via process, organization, data, type of functionality or infrastructure. Group applications per area of interest and only connect the areas. With an increasing number of applications it will give more of an overview if relationships are visualized in a matrix.

Below an example is shown of an application landscape organized along application type (control, input processing, processing, output processing and complete process support).

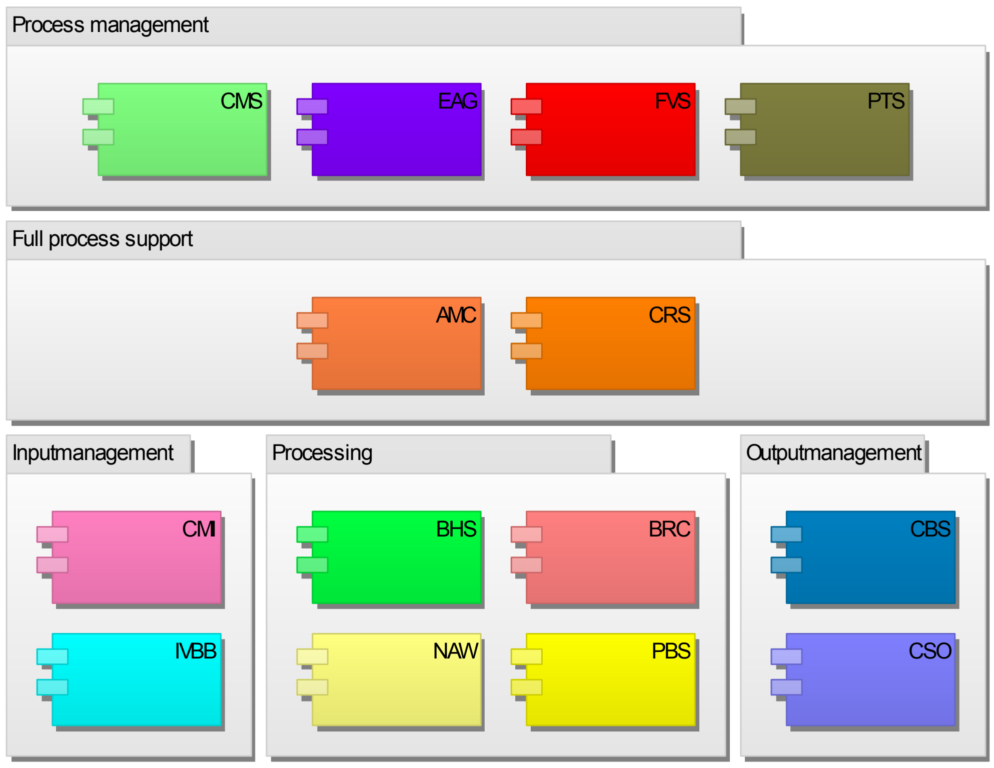

Figure 3-1: Example of an application landscape organized along type of functionality of applications

Below an example is shown of an application landscape according to usage pattern by front-office and back-office.

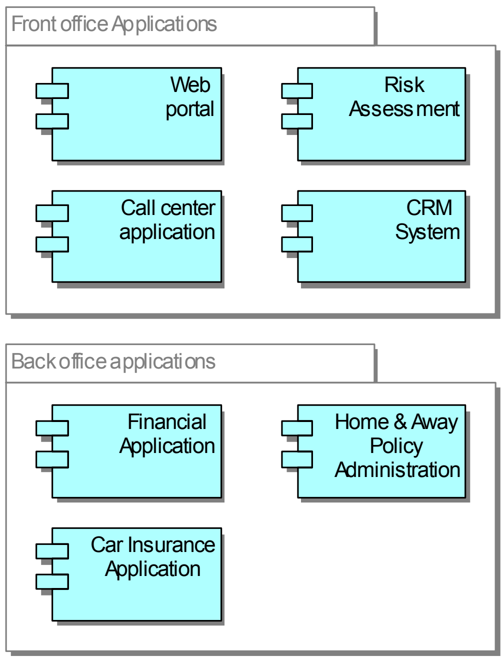

Figure 3-2: Example of an application landscape organized along usage of front-office and back-office

Consequences: For an overview of all relationships you need multiple views. Use a matrix for this purpose. A tool can be very valuable in this case to register information and produce a relationship matrix.

Alternatives: The alternative is the large 'view' with all applications and the many lines. This is only usable to express how complex the landscape is and does not give much additional added value.

Relationships with other good practices: There is an important relationship with 'Interfaces between applications'

3.3 Interfaces between applications

Question*:* It is desirable to visualize interfaces between applications. Which ArchiMate concepts can be used for this purpose?

Solution: An application is modeled as an application component with corresponding application functions. An interface between applications is modeled via an application service. The application service delivers data by means of an access relationship with a data object.
In the example model application function A realizes an application service that is used by B. The application service delivers the data that are in the data object.

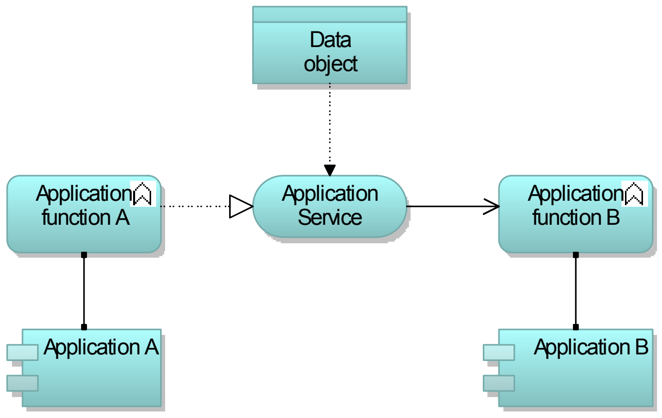

Figure 3-3: Application interface using application function, application service and connected data object

Consequences: If existing interfaces between applications in a company are visualized according to this pattern, the suggestion can be induced that a service oriented architecture is in place while in reality this is not so.

Alternatives: With compliance to rules of composition (refer to good practice 'Simplification with sustainable consistence'), application functions can be left out of the model. The figure below shows the result.

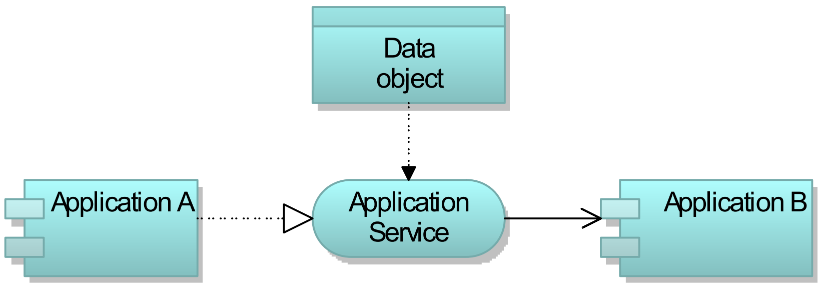

Figure 3-4: Application interface using application service and connected data object

The above interface can also be modeled without a data object:

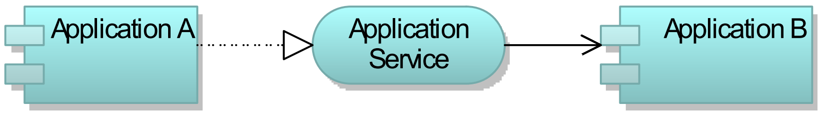

Figure 3-5: Application interface using application service

Not every organization will have a service oriented architecture. In that case it is also possible to model application interfaces using a flow relationship between the applications. By using an indirect relationship the model as shown below is achieved. Application A delivers customer data to application B.

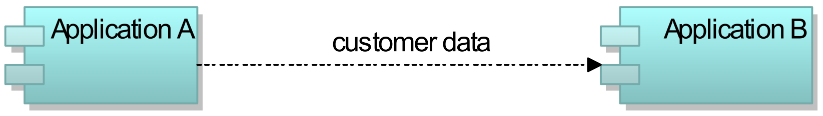

Figure 3-6: Application interface using a flow relationship

In some cases interfaces between applications are not realized directly, but via a database. One application writes into a table that is read by another application. Below figure shows this. Application A writes, Application B reads the same data object.

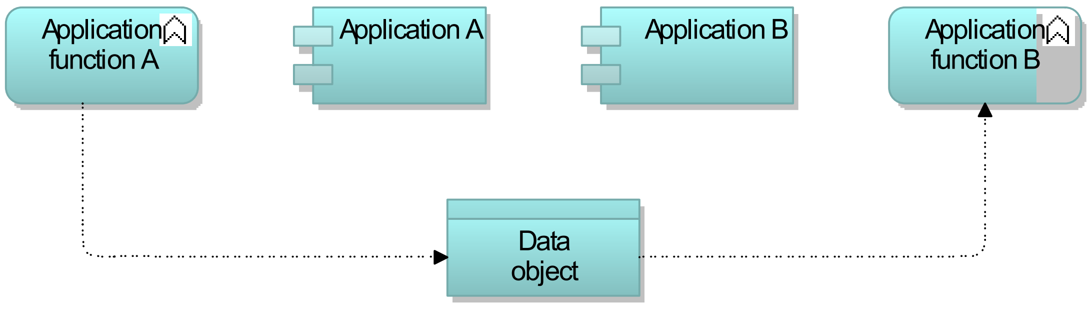

Figure 3-7: Application interface using data object

Relationships with other good practices: This can be used by constructing an application landscape using the good practice ‘Application landscape’. It is also applicable with the good practice ‘Service broker’.

3.4 Functionality of an application

Question: How can you model externally visible functionality of applications with ArchiMate?

Solution: The ArchiMate concept that was designed explicitly for modeling externally visible functionality is the application service. This is connected with a realization relationship to an application function belonging to an application. Corresponding interface(s) can be visualized if required. The figure below shows this modeling pattern:

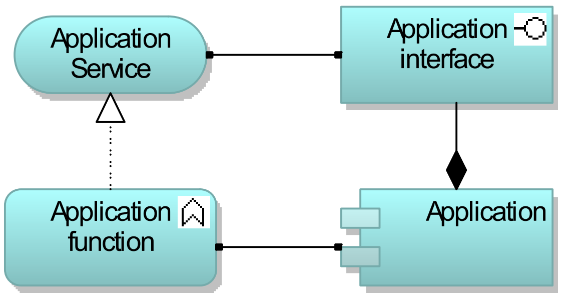

Figure 3-8: Externally visible functionality

This way of modeling can be used both for internal and external application services.

Consequences: If an application produces a lot of functionality, a model like this can become cluttered by the great number of realization relationships.
The suggestion can be easily induced that a service oriented architecture is in place while in reality this is not so.

Alternatives: This model pattern can be simplified by leaving out the application function and deduce the realization relationship to the application component (figure below to the left). If the interface has no relevance for the model, it can be left out (figure below to the right)

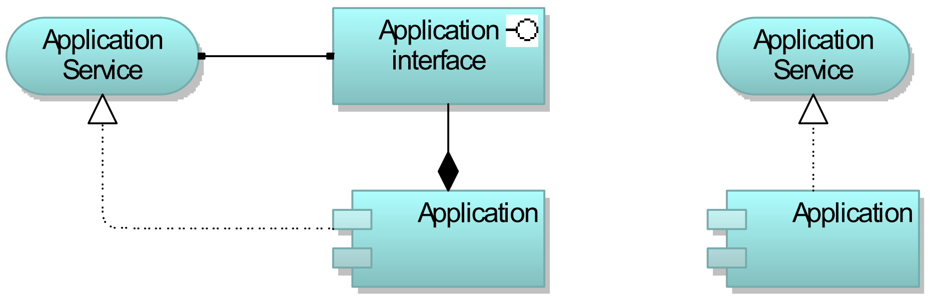

Figure 3-9: Externally visibly functionality - simplified

When no services can be distinguished, the externally visible functionality of an application can be visualized by delivering the application function via an interface to the 'consumer'. In the example below, a Business process is supported by an Application function, which is exposed by an Application interface.

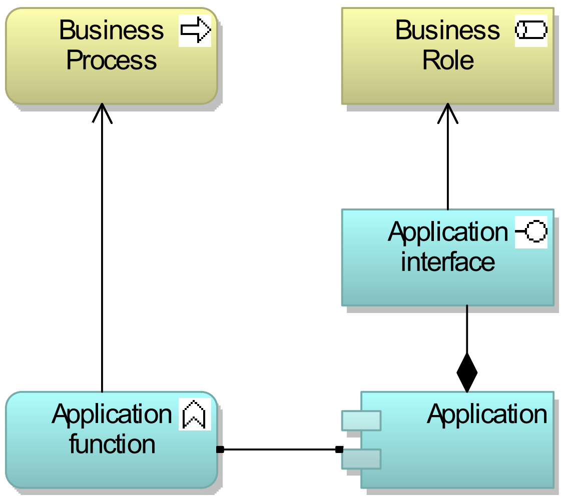

Figure 3-10: Externally visible functionality coupled to a process role

In the last view the application interface can also be left out. The result is a ‘used by’ relationship between the application and the business role.

Relationships with other good practices: For functionality that is not interactive there is a relationship with good practice ‘Interfaces between applications’. Visualizing the functionality of an application resembles visualizing applications within an application portfolio. The good practice ‘Application landscape’ is also applicable for the functionality of an application.

4 Modeling in the infrastructure- / technology layer

4.1 Overview of good practices

In this chapter the following good practices are described:
1. Infrastructure services.
2. Infrastructure landscape.

4.2 Infrastructure services

Question: Which services would one want to model on the infrastructure layer?
From maintenance point of view the infrastructure layer often requires a lot of detailed information: which systems exist and how are these connected to each other? The step in describing meaningful services for the environment is not commonplace: how does one start with modeling services on the infrastructure layer and which types of services must be distinguished?

Solution: Describe those services on the infrastructure layer that support applications. It is recommended to distinguish processing services, storage services and communication services. When modeling the infrastructure layer it is essential to use abstraction of internal details, for example implementation details. Domain specific languages are better suited for this.

Use the viewpoint ‘Infrastructure usage’ from the ArchiMate book (Lankhorst et al., 2005, p. 187). This viewpoint allows visualizing how applications are supported by software and hardware infrastructure.

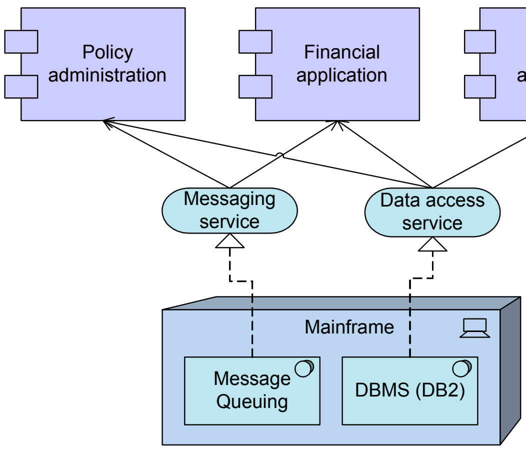

Figure 4-1: Example of infrastructure services

Apply the definition as described in ArchiMate (Lankhorst et al., 2005): an infrastructure service is a visible unit of functionality, delivered by one or more nodes, exposed via well defined interfaces and meaningful for the environment.

Consequences: Not applicable.

Alternatives: Not applicable.

Relationships with other good practices: Not applicable.

4.3 Infrastructure landscape

Question: How can an infrastructure landscape be modeled in ArchiMate? Enterprise architecture is used to register the essentials of architecture domains and visualize the relationships between those domains; on infrastructure level implementation specific details are often important. Which aspects should and which should not be modeled in an infrastructure landscape in ArchiMate?

Solution: When modeling an infrastructure landscape it is essential to maintain distance (or objectivity) from the stakeholder and their corresponding concerns. In most enterprise architecture initiatives the goal of modeling an infrastructure landscape is to visualize the most important elements (hardware and software) of the infrastructure, how these are connected to networks and what their geographical location is (such as main office and regional offices).
Limit an ArchiMate infrastructure landscape to the most important physical systems and networks and specific essential supporting software like operating systems, database management systems and middleware.

Use the Infrastructure viewpoint from the ArchiMate book (Lankhorst et al., 2005, p. 186). This viewpoint gives the possibility to show which essential hardware and software determine the infrastructure landscape and how this is connected via networks. It is recommended to group infrastructure elements in this viewpoint based on geographical location and make these groups explicit.

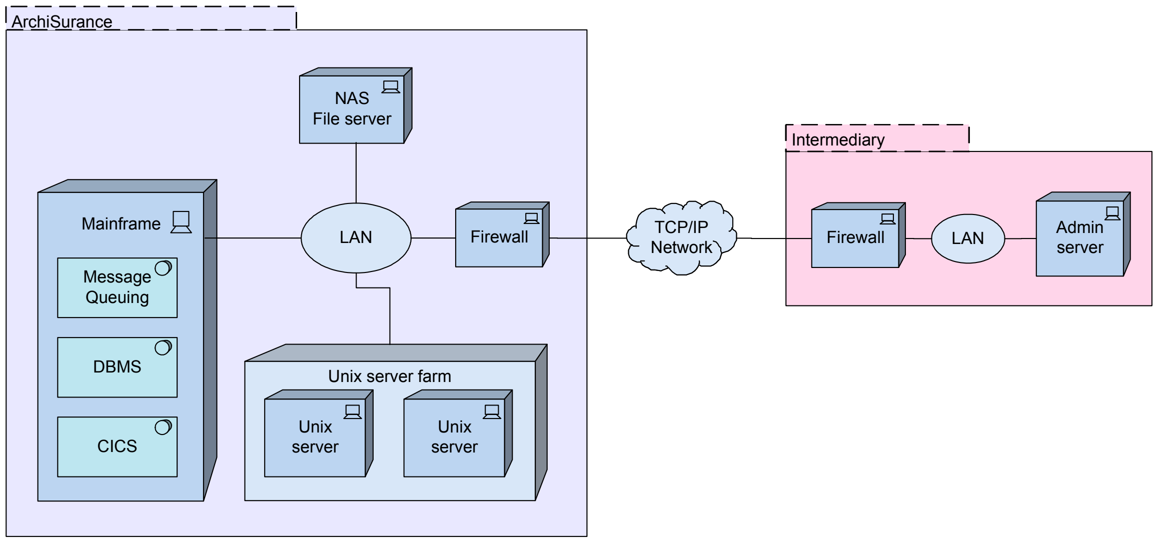

Figure 4-2: Example of an infrastructure landscape

Consequences: Not applicable.

Alternatives: Not applicable.

Relationships with other good practices: Not applicable.

5 Modeling over the boundaries of layers

5.1 Overview of good practices

In this chapter the following good practices are described:
1. Many used relationships.
2. Product and application services.
3. Support for batch and online processes.
4. Service broker.
5. Domain.
6. Internal and external services.
7. Necessity for using services.
8. Simplification with sustainable consistency.

5.2 Many used relationships

Question: ArchiMate consists of a considerable number of concepts and relationships. The question of which relationship is used to model between two concepts is exactly defined in ArchiMate, but in day-to-day practice it can be hard to pick the right choice from the allowed possibilities. The question must focus on which relationships between concepts should be used in the most practical cases. In the section below we describe which relationships should be modeled for these cases. To prevent misunderstandings: in the list summarized below not all ArchiMate relationships between concepts are described, only those relationships that we considered the most often used. A complete overview of all possible relationships between different concepts can be found in the ArchiMate reference manual.

Solution: Below an overview of most used relationships between concepts is shown (especially from the business layer and application layer).
- Business service:
  - a business service is realized by a business process or a business function;
  - a business service is used by a business role;
  - a business service has access to a business object (a business service creates, reads, modifies or destroys a business object);
  - a business service can consist of other business services and can use other business services.

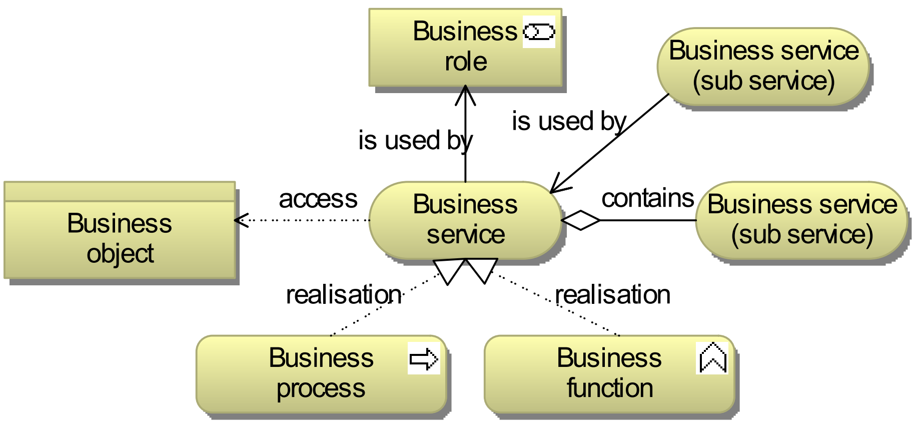

Figure 5-1: Most important relationships between concepts

- Business process:
  - a business process realizes a business service;
  - a business process exchanges data with other business processes (via the flow relationship);
  - a business process is triggered by or triggers a business event, a business function or other business processes;
  - a business process is assigned to a business role;
  - a business process is part of a business function;
  - a business process has access to a business object (a business process creates, reads, modifies or destroys a business object);
  - a business process uses application services.

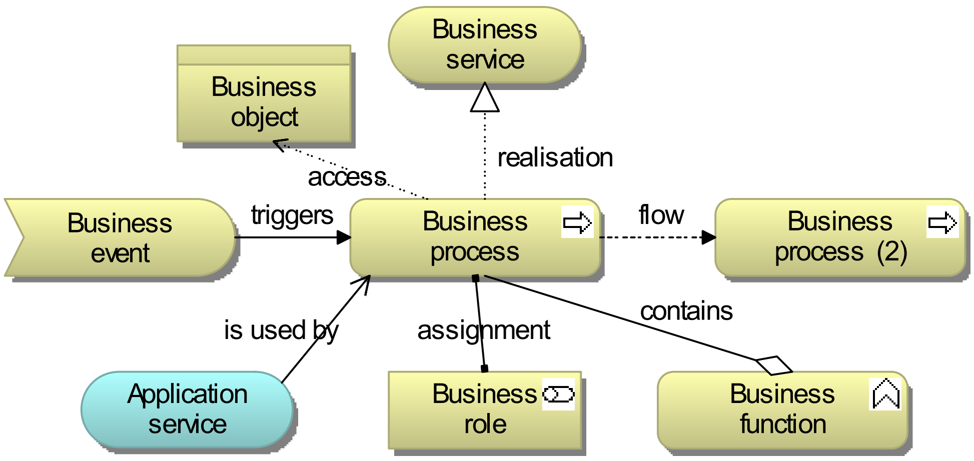

Figure 5-2: Most important relationships between concepts

- Business actor:
  - A business role can be assigned to a business actor.

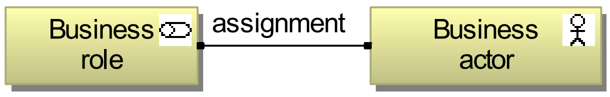

Figure 5-3: Relationship between business role and business actor

- Business role:
  - A business role can be assigned to a business actor;
  - A business role can be assigned to a business process, business function or business event;
  - A business role can use a business interface.

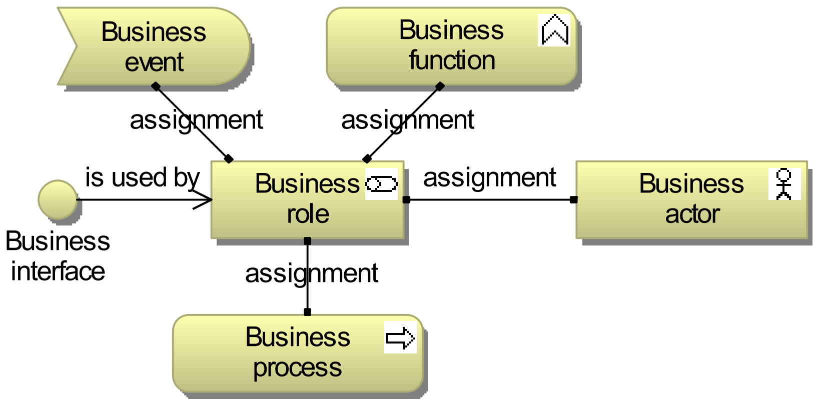

Figure 5-4: Most important relationships between concepts

- Business object:
  - A business object is created, read, modified or destroyed by a business process or business function (via access relationship);
  - A business object can have specializations;
  - A business representation realizes a business object;
  - A business object can point to other objects (aggregation relationship);
  - A business object can consist of other objects (composite relationship).

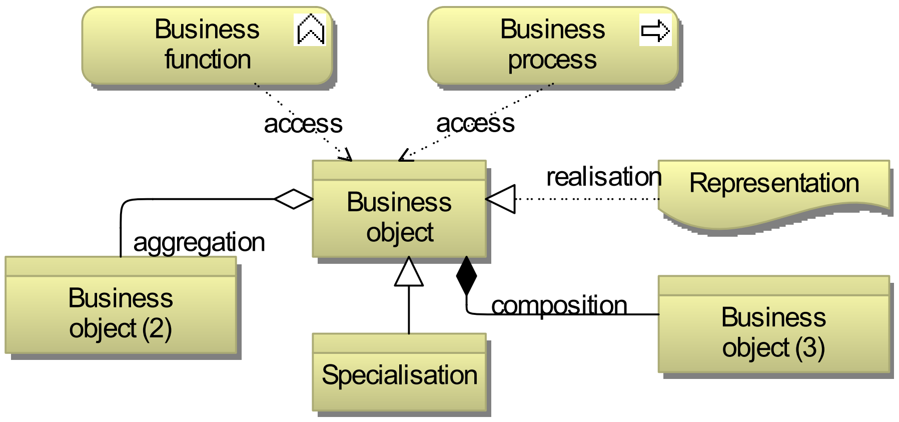

Figure 5-5: Most important relationships between concepts

- Business product:
  - A business product consists of services and contract(s);
  - A business product represents a value.

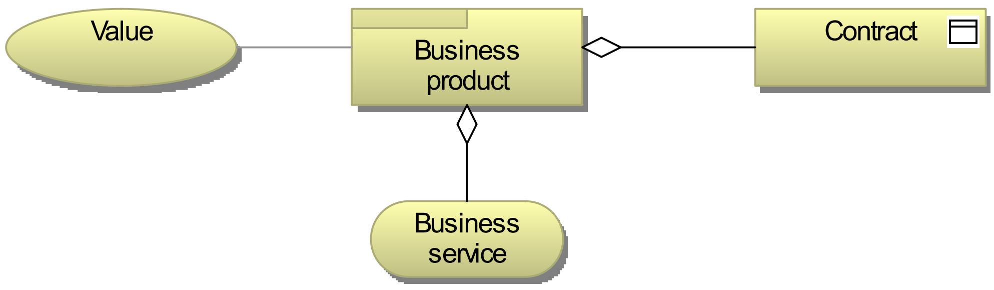

Figure 5-6: Most important relationships between concepts

- Application component:
  - An application component realizes an application service;
  - An application component has one or more application interfaces;
  - A data object is created, read, modified or destroyed by an application component or application function;
  - An application component can be part of an application collaboration (via aggregation relationship);
  - An application component can consist of multiple application components (via composite relationship);
  - An application component can be assigned to an application function;
  - An application component uses infrastructure services;
  - Data flows can exist between application components (flow relationship).

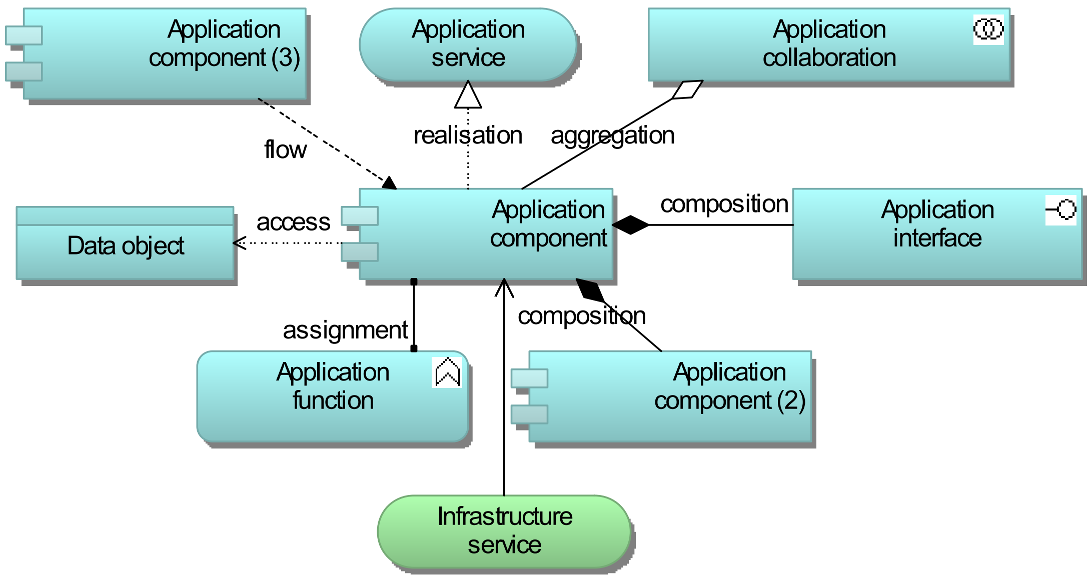

Figure 5-7: Most important relationships between concepts

- Application service:
  - an application service is realized by an application component or an application function;
  - an application service is used by an application component;
  - an application service has access to a data object (an application service creates, reads, modifies or destroys a data object);
  - an application service can consist of other application services and can use other application services.

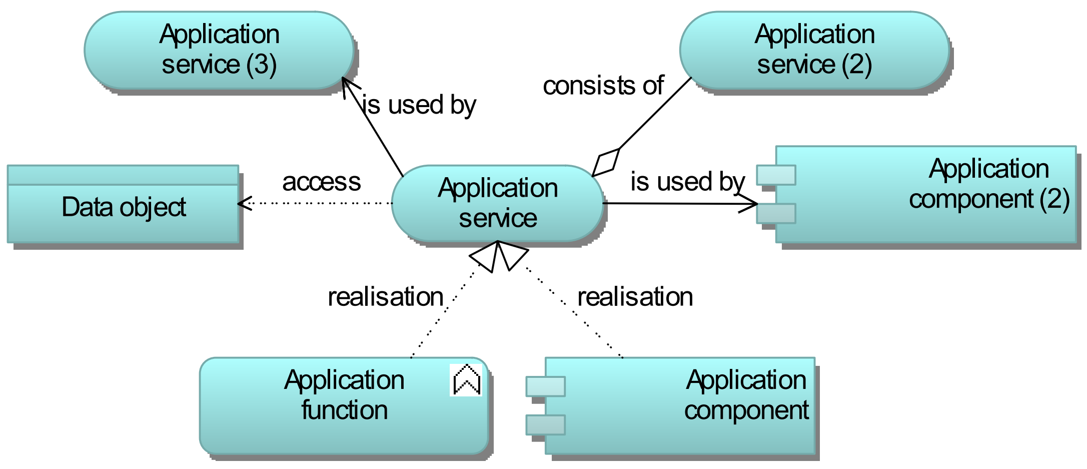

Figure 5-8: Most important relationships between concepts

- Data object:
  - A data object is created, read, modified or destroyed by an application component, application service or application function (via access relationship);
  - A data object can have specializations;
  - An artifact realizes a data object;
  - A data object can point to other data objects (aggregation relationship);
  - A data object can consist of other data objects (composite relationship).

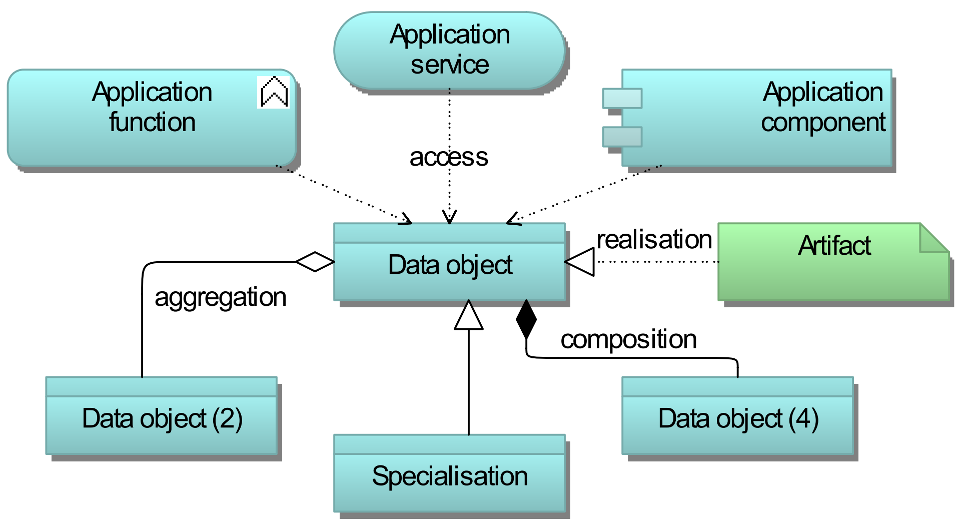

Figure 5-9: Most important relationships between concepts

Consequences: Beware; these are a number of frequently occurring situations where the use of relationships in ArchiMate is much broader than summarized above. A complete overview can be found in the Architecture Language Reference Manual.

Alternatives: Not applicable.

Relationships with other good practices: Not applicable.

5.3 Product and application services

Question: Isn't it strange that a product can not only consist of business services but also application- and infrastructure services? A customer cannot possibly 'consume' application services? So if an application service is used almost directly by a customer, wouldn't there be a business service in between? In some examples in the Architecture Language Reference Manual, a product is composed of application and business services.

Solution: The Architecture Language Reference Manual indeed contains examples where customers use application services directly. It is better to put a business service in between. In the product description of the Architecture Language Reference Manual (p. 26) the application services 'Account status' and 'Money transfer' will become business services.

The connection between a business service and an application service is routed via a business process that supports the business service, and is in its turn supported by an application service. This validates the indirect relationship that a business service uses an application service.

Consequences: You should consider deleting the aggregation relationship between product and application service.

Alternatives: Not applicable.

Relationships with other good practices: Not applicable.

5.4 Support of batch and online processes

Question: How can you model two variants of the same process in ArchiMate whereby one process is executed as a batch process and the other process is executed manually (online)?

Solution: Distinguish two variants of the business process: one for manual (online) processing and one for batch processing. Although the results of the processes can be identical, the manual (online) processing allows more room for exceptions and often more granular process steps can be distinguished. In both process descriptions it is sensible to consider actions as *units of time, place and behavior* when determining the 'size' of the actions. This will probably differentiate the two process variants.

Based on the two variants of the business process, describe the application services that support these processes. Use the actions that have been modeled in the business process as a starting point: the implication of this is that the application services for both variants can differ! Often it will be that application services that support a batch process are more granular than application services that support an online process, because in the latter case more granular actions can be distinguished. Beware, it is very well possible that *application functions* that realize the different functions of application services are common: this points to reuse of functionality in the batch and online processing.

Use the Application usage viewpoint from the ArchiMate book (Lankhorst et al., 2005, p. 184). This viewpoint allows one to visualize how business processes are supported by applications, via application services.

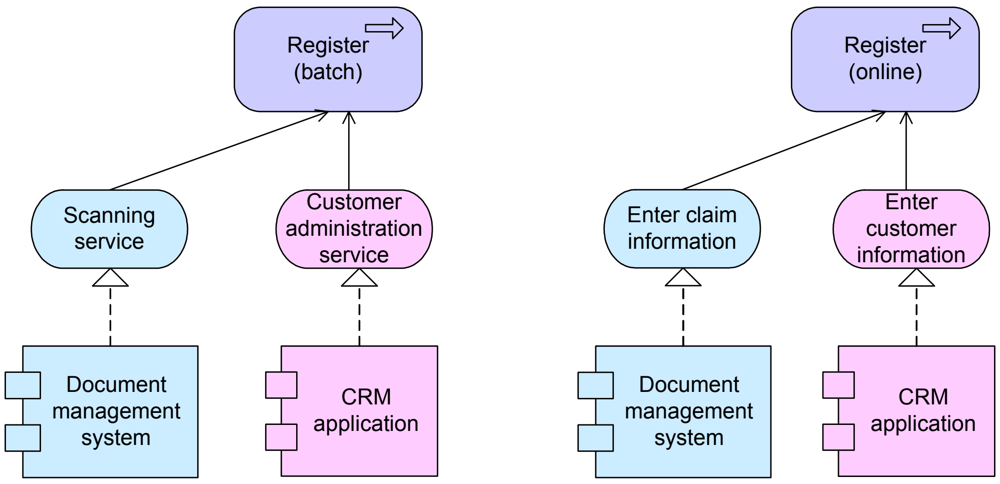
Register (batch)
Register (online)
Scanning service
Customer administration service
Enter claim information
Enter customer information
Document management system
CRM application
Document management system
CRM application

Figure 5-10: Example of a batch and online (manual) variant of the same process

Consequences: Not applicable.

Alternatives: Not applicable.

Relationships with other good practices: Refer to the good practices ‘Business service, business function and business process’, ‘Constraining business processes’ and ‘Viewpoints for process modeling’.

5.5 Service broker

Question*:* How do you model a service broker (or comparable component, for example a workflow handler or a middleware-type component)? On one hand it is clear that the broker is the center point of all services, but on the other hand it is not visible which application produces which service (via the broker). The goal of a broker is simplification of the many relationships between different applications. Sometimes the following modeling pattern is used:

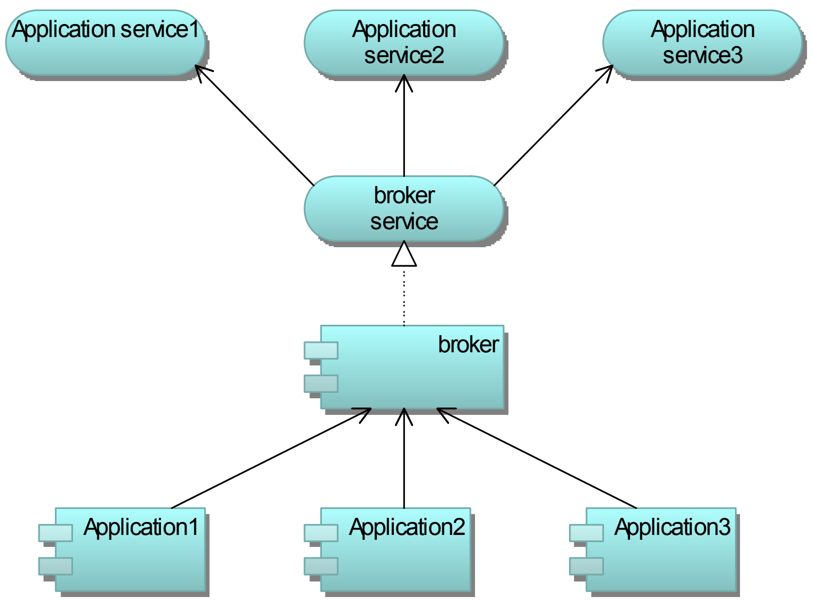

Figure 5-11: Service broker related to applications - undesired

Disadvantage of this way of modeling is that it is no longer evident which application delivers which service. It is also desirable to hide or unhide the broker (and its service) when required. This is not possible with the above modeling pattern.

Solution: The core of the solution is in relating the applications and application services on one hand (the application realizes the corresponding application service), and relate the application services and the broker service on the other hand (the broker service "contains" the application services). The situation described above will then be modeled as follows:

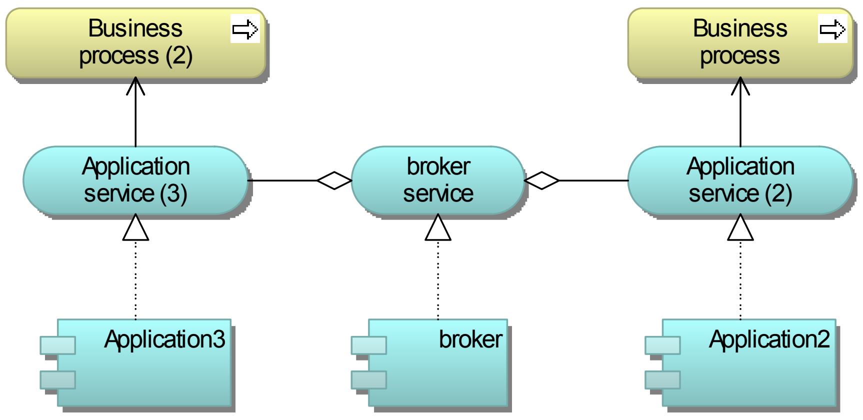

Figure 5-12: Service broker related to applications - desired

By generating different views (and leaving out certain relationships or showing these in a nested manner), different focal points can be achieved. In the following figures the relationship between applications with and without a broker and the role of the broker are visualized.

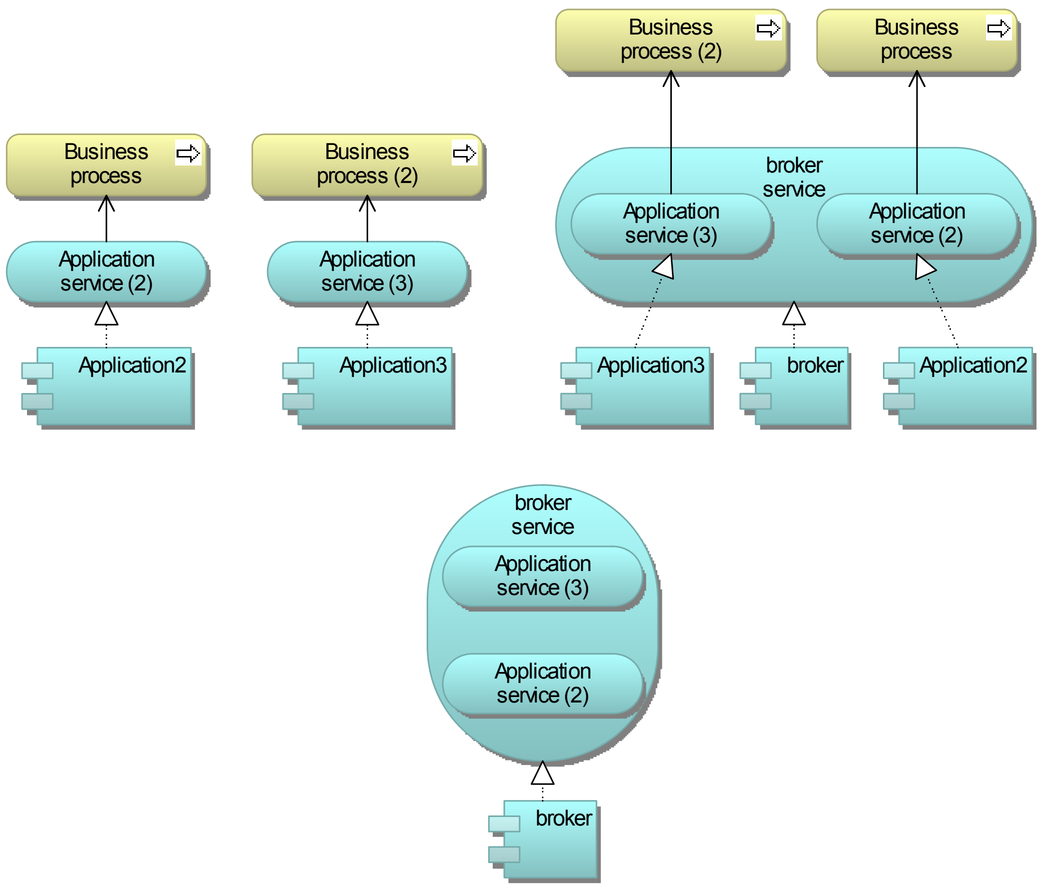

Figure 5-13: Service broker related to applications - views

Consequences: By modeling a broker and a broker service as shown above, it becomes possible to either show or hide the broker at will, which will improve the insight of the model for different stakeholder groups.

Alternatives: Not applicable

Relationships with other good practices: The service broker is part of the application landscape. Refer to good practice ‘Application landscape’.

5.6 Domain

Question*:* How can I model a ‘domain’ in ArchiMate? A ‘domain’ is defined as a certain demarcation meant to structure concepts. These could be business-, application-, information- or technical domains.

Solution: In ArchiMate the grouping relationship is suited to grouping domains. The relationship can be given a name. In the example below (the part of) the sales domain is shown.

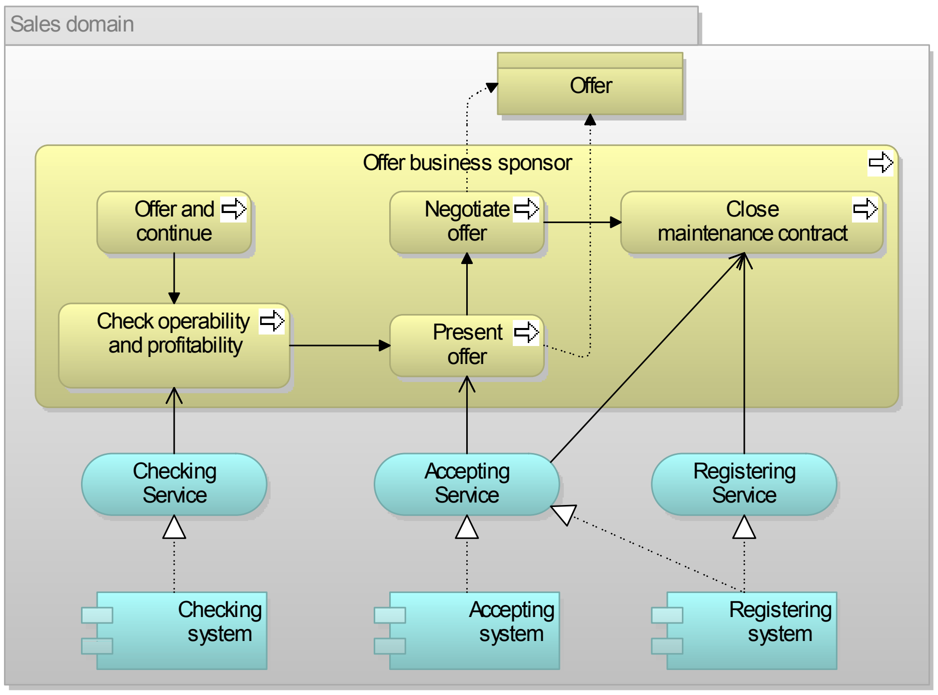

Figure 5-14: Example of a sales domain

Consequences: ArchiMate distinguishes domains and relationships. Grouping is a relationship. There is a desire to reuse the grouping relationship in other views (if possible without explicitly adding its elements, so only copying the 'grouping' itself). ArchiMate is not explicit when it comes to relationships or and how they are used when using the grouping concept. Existing tools do not support this.
The usage of the grouping relationship implies that domains can be flexibly handled. Any domain one could think of can be visualized in this way. To prevent cluttering it is recommended to beforehand develop the domain types that are distinguished within an organization and which concepts are part of that (aggregation relationship).

Alternatives: If a targeted domain consists of similar concepts (for example only application components) an occurrence of such a concept might also be used to construct the domain. The components in the domain will then be related to the domain with an aggregation relationship.

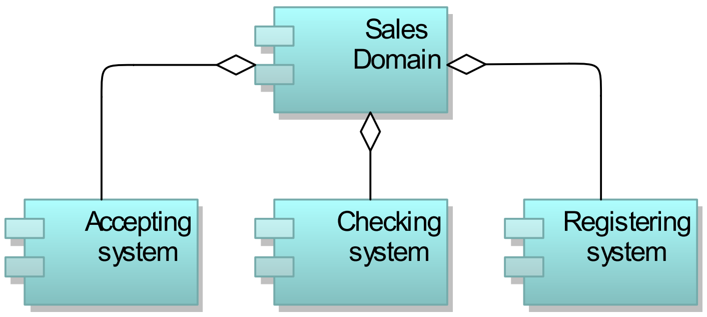

Figure 5-15: Example of a sales domain on the application layer

Process domains can be constructed with a business process or business function as container. For an application domains, an application component can be used. For an infrastructure domain, the node seems the most suitable container. There is currently a study in progress to extend the language with a separate 'Domain' concept.

Relationships with other good practices: Refer to good practice 'application landscape'.

5.7 Internal and external services

Question*:* ArchiMate distinguishes services that are used within the own layer and by the next layer above.
The 'own layer' is the business layer for business services, the application layer for application services and the infrastructure layer for infrastructure services. ArchiMate calls this "internal services".
The 'next upper layer' for the infrastructure layer is the application layer, for the application services it is the business layer and for business services it is the environment. ArchiMate calls this "external services". The concepts internal and external are therefore relative with respect to the layer where the service is being realized.
How do you visualize this difference? Also refer to EA at work, p. 86.

Solution: Defining services that are used within the own layer ('internal services') is what is often meant with Service Orientation. The modeled services usually coincide with actually implemented / to be implemented services (especially with respect to application services and infrastructure services).
Services that are used by the next upper layer ('external services') represent meaningful functionality that supports the next upper layer; this will not always refer to an implemented service (with respect to an actual externally 'callable' functionality).

We recommend as good practice to handle internal and external services as two dedicated groups and distinguish these in views, especially in those cases where distinction between internal and external services is important. External services will often have other demands than internal services, and other aspects will have to be registered. The service itself will often have a differing design, depending on the intended support within the own layer or the next upper layer.

Figure 5-16 is an example of how this distinction can be visualized: the internal and external services are distinguished by placing (grouping) them in separate layers[1]. In this example the Customer management service is an external service and is therefore positioned in the corresponding layer. The Customer data service is used only within the application layer and is therefore an internal service. Another possibility is to shape and/or color internal and external services accordingly.

[1] Notice that in Figure 5-16 a simplification has been applied by leaving out the application function – refer to good practice "Simplification with sustainable consistency".

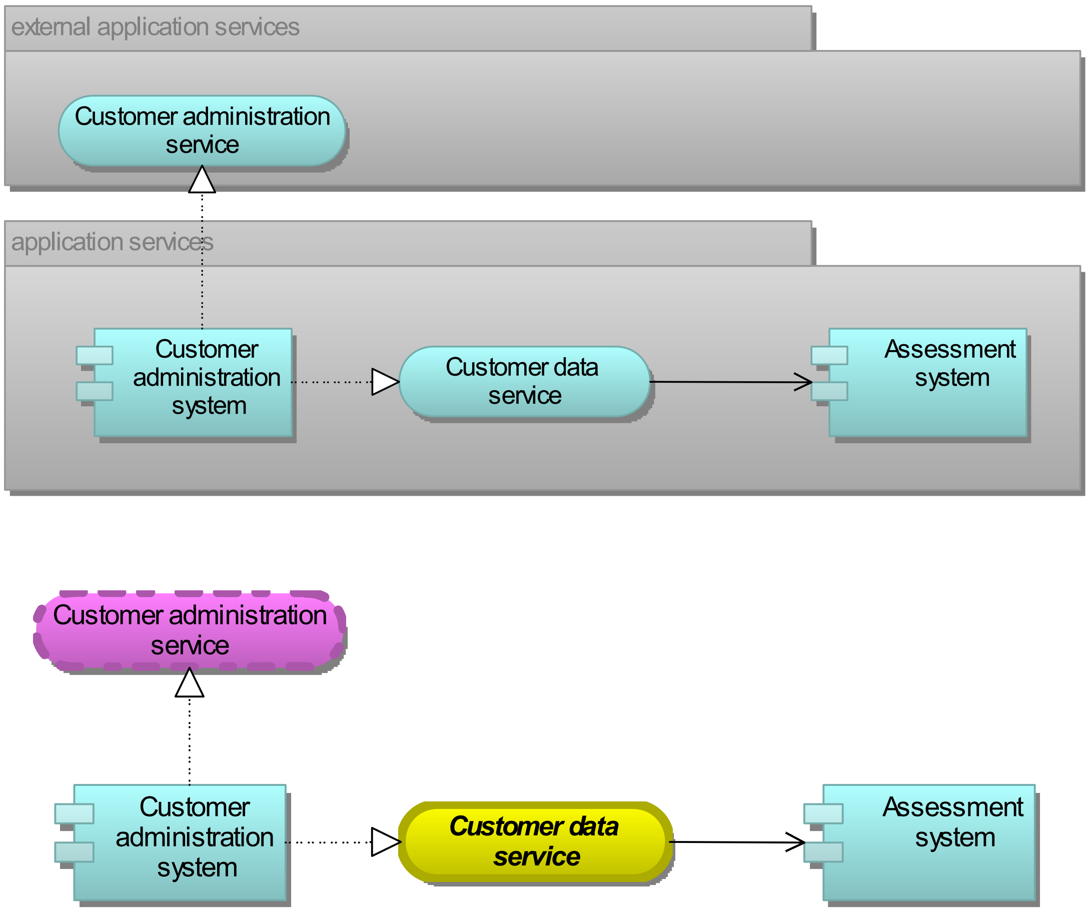
external application services
Customer administration service
application services
Customer administration system
Customer data service
Assessment system
Customer administration service
Customer administration system
Customer data service
Assessment system

Figure 5-16: In- and external services

Consequences: Make an explicit distinction between internal and external services.

Alternatives: In this good practice a distinction is made between internal and external services. Reasons may exist not to make this but another distinction, for example between business critical and non critical services, or between services with a gold, silver or bronze service level agreement. To make a distinction like this an identical approach as described above can be used, more specifically to use grouping, coloring or shaping to emphasize the distinction. The model and visualization goal determines the distinction that is so relevant that it has to be emphasized.

Relationships with other good practices: There is a relationship with good practice 'Interfaces between applications'.

5.8 Necessity for the usage of services

Question: Are services mandatory in ArchiMate? In my current situation we have no implemented services. We also do not have a service oriented vision.

Solution: Services in ArchiMate are not mandatory. If these are not distinguished and not foreseen for the future, the services can be left out at will.

Consequences: If not services are distinguished, then no services will be modeled. The rules applying to good practice 'Simplification with sustainable consistency' have to be used to use the ArchiMate concepts in a consistent manner.
Instead of a process that uses a service (1), this uses an application function (2) or, with increasing simplification, an application (3).

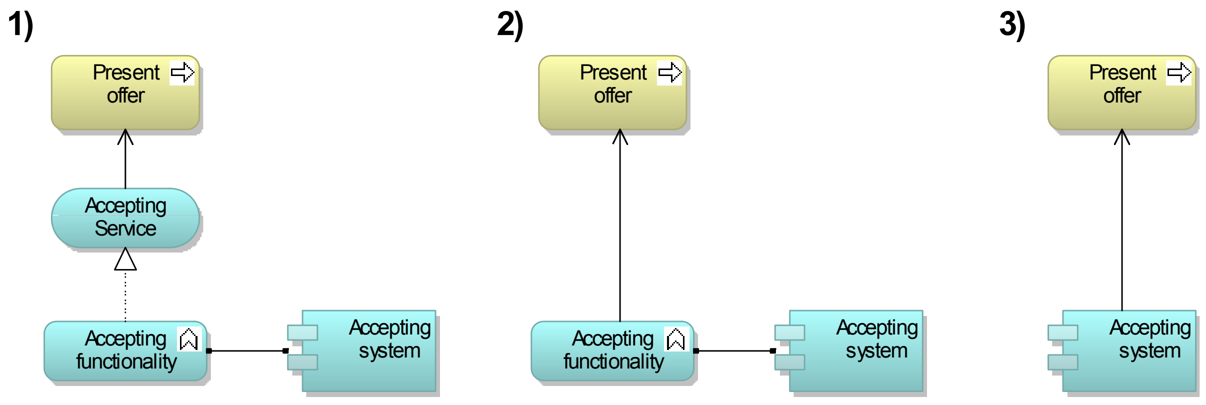

Figure 5-17: Use of an application function instead of an application service

Alternatives: Even if currently no services are distinguished, service oriented thinking can help to achieve another view on the world. This can lead to new insights. By considering the world from a service oriented viewpoint one often will arrive at other groupings and responsibilities.
If the service oriented approach is envisioned to be used in the future, that future situation can be modeled with services.

Relationships with other good practices: Refer to good practice 'Simplification with sustainable consistency'.

5.9 Simplification with sustainable consistency

Question*:* How can I simplify my views without compromising the consistency?

Solution: Within ArchiMate it is possible to deduce indirect relationships by compositions of relationships. This is described in paragraph 5.7 of the book EA at work. This is considered an indispensable instrument for visualizing architecture: leave out some details and at the same time maintain consistency between models.

For *structural* relationships, a powerful composition rule has been developed based on the *strength* of the relationship. In a chain of concepts the most 'weak' relationship in the chain determines the overall relationship between the endpoint concepts of the chain. The table below shows the strength of structural relationships. Figure 5-18 shows an example of this composite rule.

In this example, the indirect relationship between the process Registration and the application function Manage Customer data is a *used by* relationship because this has a lower weight than a *realization*.

Table Strength of structural relationships

Relationship	**Strength**
association	1
access	2
used by	3
realization	4
assignment	5
aggregation	6
composition	7

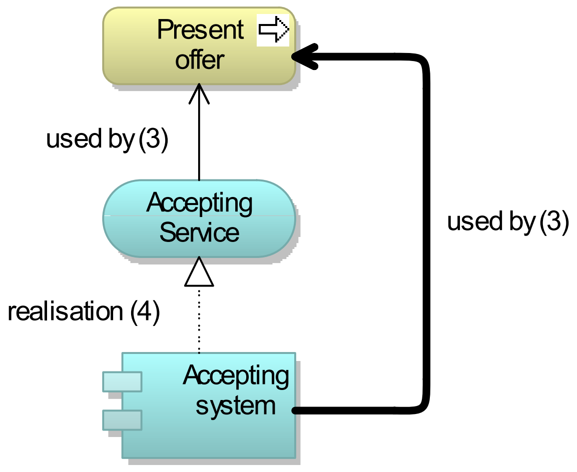

Figure 5-18: Example of an indirect relationship

The following rules apply for the two dynamic relationships:

- A triggering or flow relationship between behavioral elements (for example processes or functions) may be redirected to structural elements (for example business actors or application components) to which they are assigned.

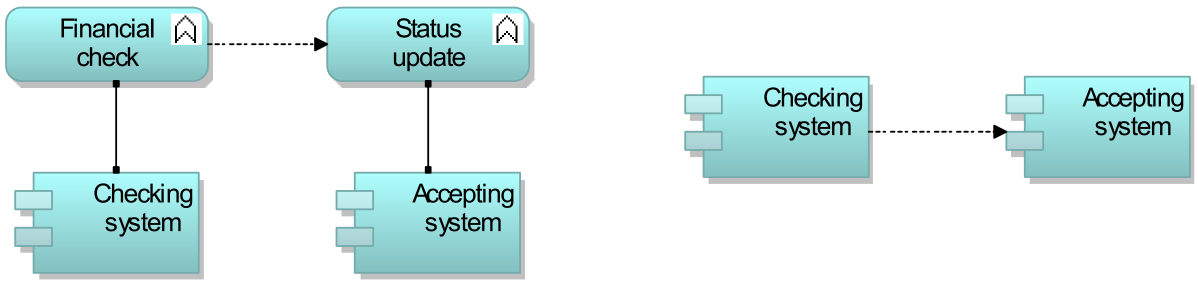

Figure 5-19: Example of an indirect relationship

- A triggering- or flow relationship between behavioral elements may be redirected to the services that they realize.

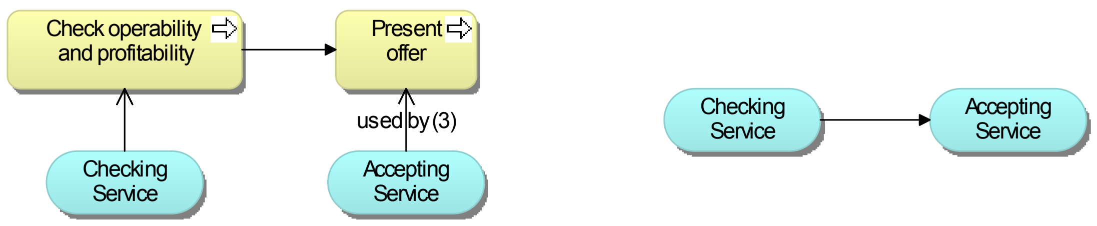

Figure 5-20: Example of an indirect relationship

Consequences: Views can be simplified in this manner to a level of detail that is relevant for the goal of the view, without losing consistency.
Not every tool supports this simplification. The consequences are that redundant relationships may exist. In the above example of the services this implies that the existing relationship between the processes is registered in the tool (left part). If the relationship between the services is visualized in another view (right part), the tool may introduce an extra triggering relationship between these services.

Alternatives: Not applicable

Relationships with other good practices: These simplifications can be applied to any good practice.

References

M. Lankhorst, *Enterprise Architecture At Work – Modeling, Communication, and Analysis*, Telematica Instituut/Springer, 2005, ISBN 3-540-24371-2.

H. Bosma, H. Jonkers and M. Lankhorst, *Inleiding in de ArchiMate-taal*, ArchiMate/D.1.1.6a, v1.1, Telematica Instituut/Ordina, 24 oktober 2005. https://doc.telin.nl/dscgi/ds.py/Get/File-49772.

R. van Buuren, S. Hoppenbrouwers, H. Jonkers, M. Lankhorst and G. Veldhuijzen van Zanten, *Architecture Language Reference Manual*, ArchiMate/D2.2.2b, v4.1, Telematica Instituut, 4 april 2006. https://doc.telin.nl/dscgi/ds.py/Get/File-31626.

H. ter Doest, M.-E. Iacob, M. Lankhorst, D. van Leeuwen and R. Slagter, *Viewpoints Functionality and Examples*, ArchiMate/D3.4.1a, v2.6, Telematica Instituut, 18 november 2004. https://doc.telin.nl/dscgi/ds.py/Get/File-35434.

ArchiMate Team, *ArchiMate Quick Reference*, Telematica Instituut, 2005. https://doc.telin.nl/dscgi/ds.py/Get/File-52048.