

ArchiMate

Application Layer

Specifica ArchiMate

<https://pubs.opengroup.org/architecture/archimate3-doc/ch-Application-Layer.html>

[< Previous](#)

[▲ Home](#)

[Next >](#)

ArchiMate® 3.1 Specification
Copyright © 2012-2019 The Open Group
Previous versions: [3.0.1 | 3.0.12.1]



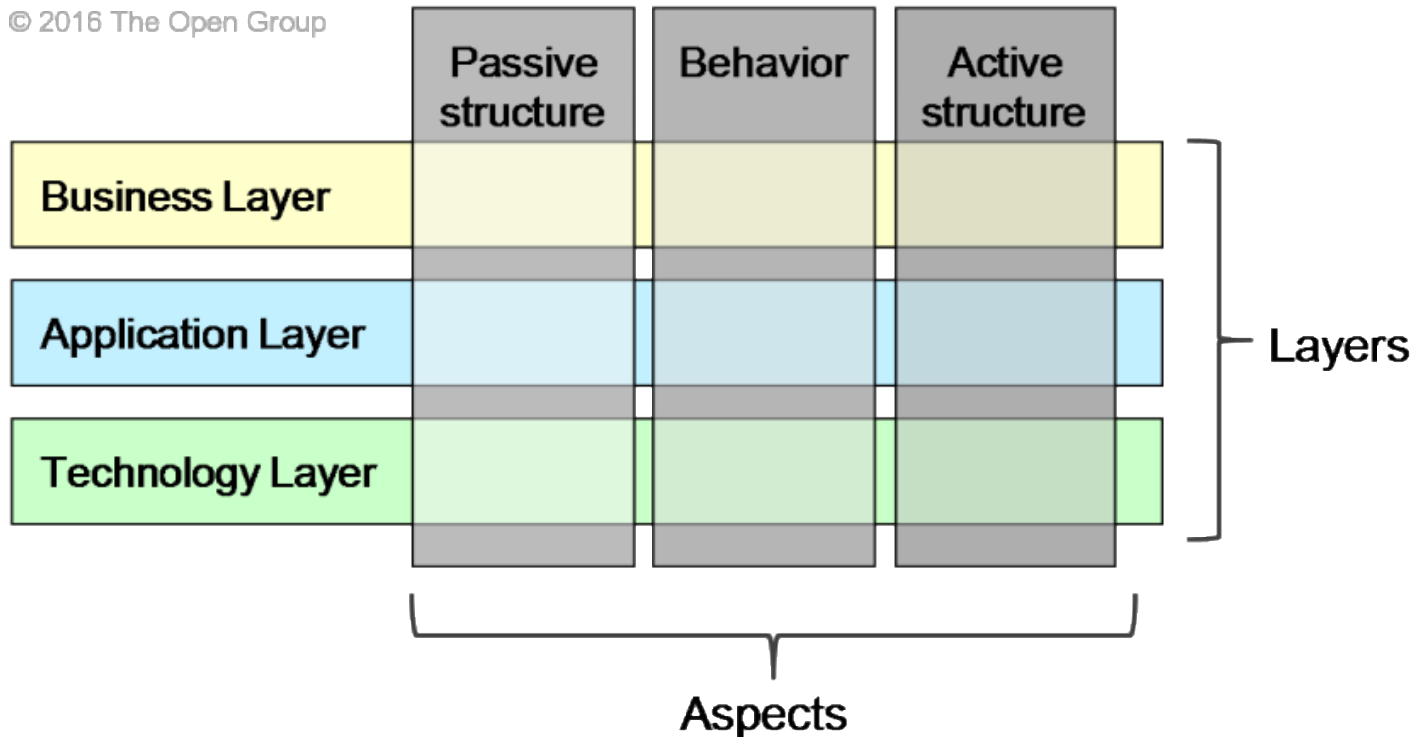
Welcome to the ArchiMate® 3.1 Specification, *a Standard of The Open Group*

[Frontmatter](#)

- 1 [Introduction](#)
 - 1.1 [Objective](#)
 - 1.2 [Overview](#)
 - 1.3 [Conformance](#)
 - 1.4 [Normative References](#)
 - 1.5 [Terminology](#)
 - 1.6 [Future Directions](#)
- 2 [Definitions](#)
 - 2.1 [ArchiMate Core Framework](#)
 - 2.2 [ArchiMate Core Language](#)
 - 2.3 [Architecture View](#)
 - 2.4 [Architecture Viewpoint](#)
 - 2.5 [Aspect](#)
 - 2.6 [Attribute](#)
 - 2.7 [Composite Element](#)
 - 2.8 [Concept](#)
 - 2.9 [Conformance](#)
 - 2.10 [Conforming Implementation](#)
 - 2.11 [Core Element](#)
 - 2.12 [Element](#)
 - 2.13 [Layer](#)
 - 2.14 [Model](#)
 - 2.15 [Relationship](#)
- 3 [Language Structure](#)
 - 3.1 [Language Design Considerations](#)
 - 3.2 [Top-Level Language Structure](#)
 - 3.3 [Layering of the ArchiMate Language](#)
 - 3.4 [The ArchiMate Core Framework](#)
 - 3.5 [The ArchiMate Full Framework](#)
 - 3.6 [Abstraction in the ArchiMate Language](#)
 - 3.7 [Concepts and their Notation](#)
 - 3.8 [Use of Nesting](#)
 - 3.9 [Use of Colors and Notational Cues](#)

ArchiMate CORE framework

© 2016 The Open Group



ArchiMate Core describe I differenti layer/domini dell'Enterprise Architecture:

- Layer Business: obiettivi e requisiti
- Layer Applicativo: funzionalità dei sistemi
- Layer Infrastrutturale: scelte tecnologiche (piattaforme di scambio dati, sicurezza...)

Application Layer

Obiettivo: descrivere l'infrastruttura applicativa che permette di sostenere i processi di business

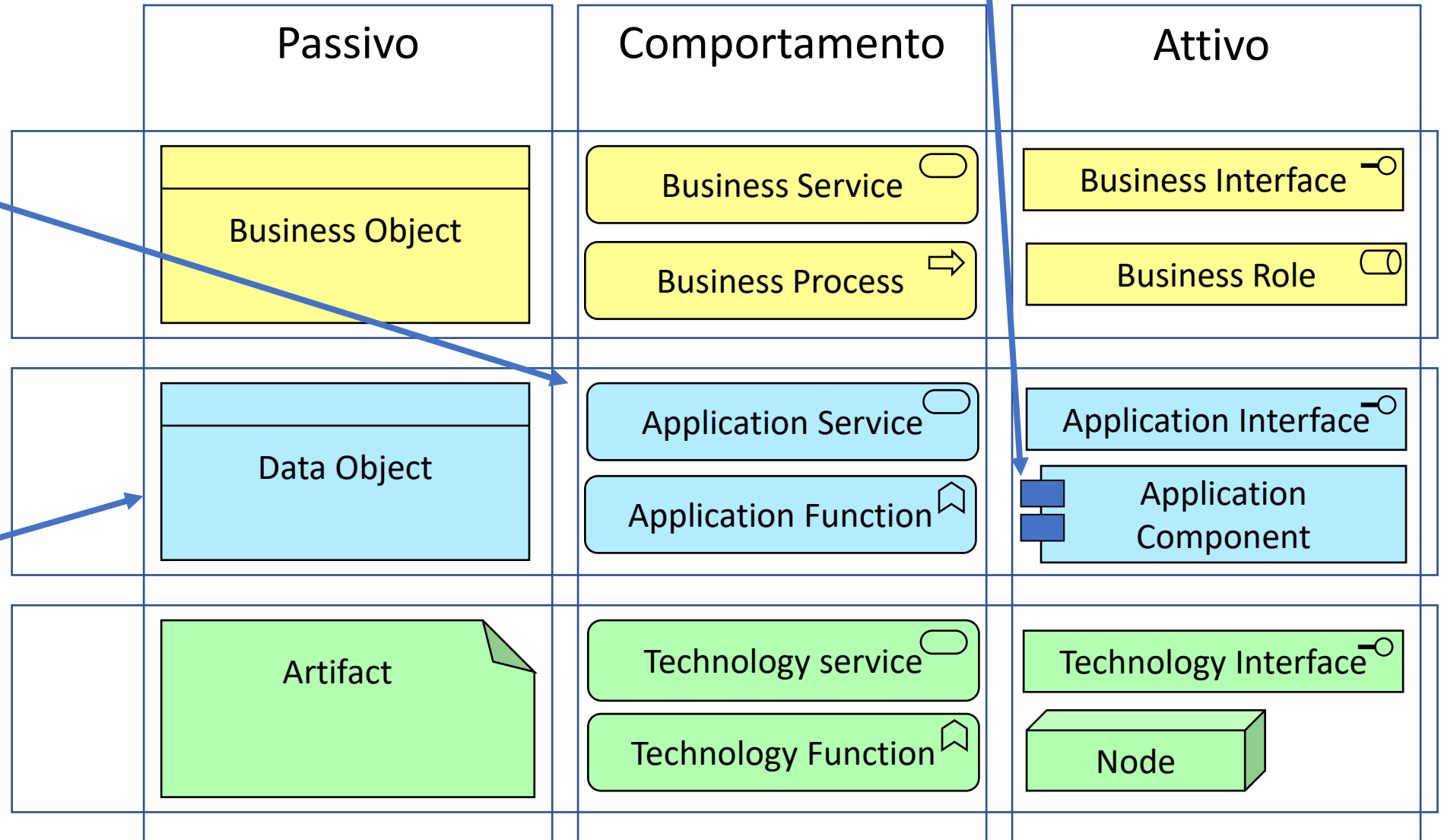
Architettura applicativa: descrive la struttura, il comportamento e le interazioni delle applicazioni dell'impresa

Application Layer

I componenti applicativi (attivi) svolgono un applicazione / compito

Cosa viene eseguito
(automaticamente)
ed esposto

Dati scambiati
(es. relazioni nel
DB, oggetti)



Elementi attivi

Application
component



Modulo che garantisce l'offerta / esecuzione di una determinata funzionalità (modulare e sostituibile)

Application
collaboration



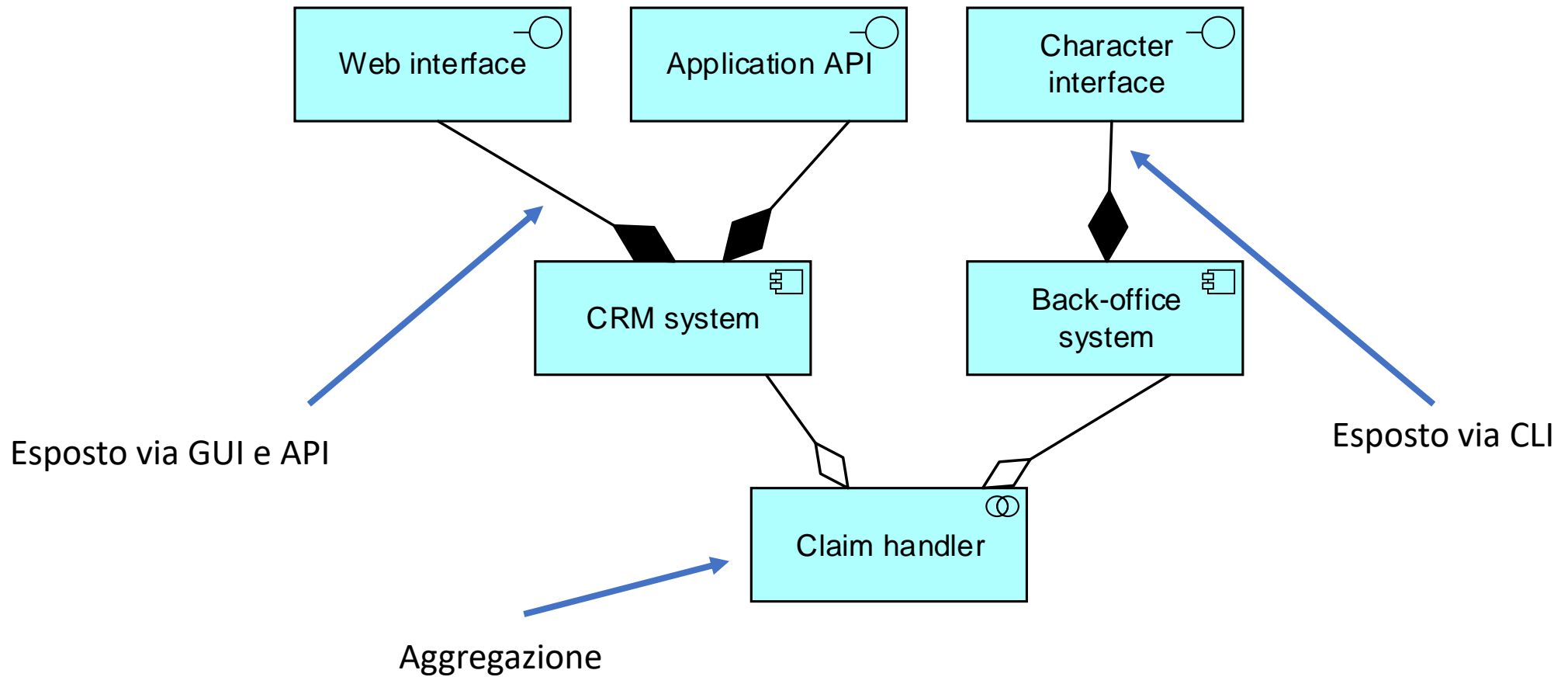
Aggregazione di uno o più moduli application component per comporre un comportamento collettivo

Application
interface



Modalità con cui un component può essere acceduto. A livello applicativo un modulo può essere offerto tramite GUI / CLI / API

Relazioni tra elementi attivi



Elementi comportamentali

Application
process



Una sequenza di comportamenti applicativi che ottiene uno specifico scopo

Application
function



Aggregazione di elementi comportamentali che possono essere eseguiti da un componente applicativo che espone una determinata funzionalità

Application
interaction



Un'applicazione collettiva eseguita da una o più componenti applicative (collaborazione)

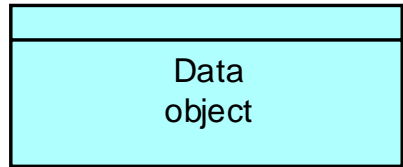
Application
service

Servizio applicativo: un comportamento applicativo esplicitamente definito, è il modo in cui le funzionalità sono esposte, tramite un'interfaccia adeguata

Application
event

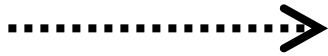
Evento applicativo: un elemento comportamentale che denota un cambiamento di stato

Elementi passivi



Elemento che permette di caratterizzare informazione / dati strutturati per il processamento delle informazioni

Relazioni



Accesso (*dipendenza*)

DA Comportamento

A Passivo

Il comportamento accede a un'entità passiva



Serve (*dipendenza*)

DA / A Entità dello stesso aspetto (escluso Passivo)

L'entità in testa usa l'entità in coda



Realizzazione (*struttura*)

DA / A Entità dello stesso livello

(DA Artefatto

A Componente)

L'entità in coda crea / fornisce / implementa l'entità in testa



Assegnamento (*struttura*)

DA Attivo

A Comportamento

DA Nodo

A Artefatto

L'entità attiva esegue il comportamento / l'artefatto



Composizione (*struttura*)

DA / A Entità dello stesso aspetto

L'entità in coda è una componente / parte indivisibile dell'entità in testa

Relazioni



Accesso (*dipendenza*)

DA Comportamento

A Passivo

Il comportamento accede a un'entità passiva



Serve (*dipendenza*)

DA / A Entità dello stesso aspetto (escluso Passivo)

L'entità in testa usa l'entità in coda



Realizzazione (*struttura*)

DA / A Entità dello stesso livello

(DA Artefatto

A Componente)

L'entità in coda crea / fornisce / implementa l'entità in testa



Assegnamento (*struttura*)

DA Attivo

A Comportamento

DA Nodo

A Artefatto

L'entità attiva esegue il comportamento / l'artefatto



Composizione (*struttura*)

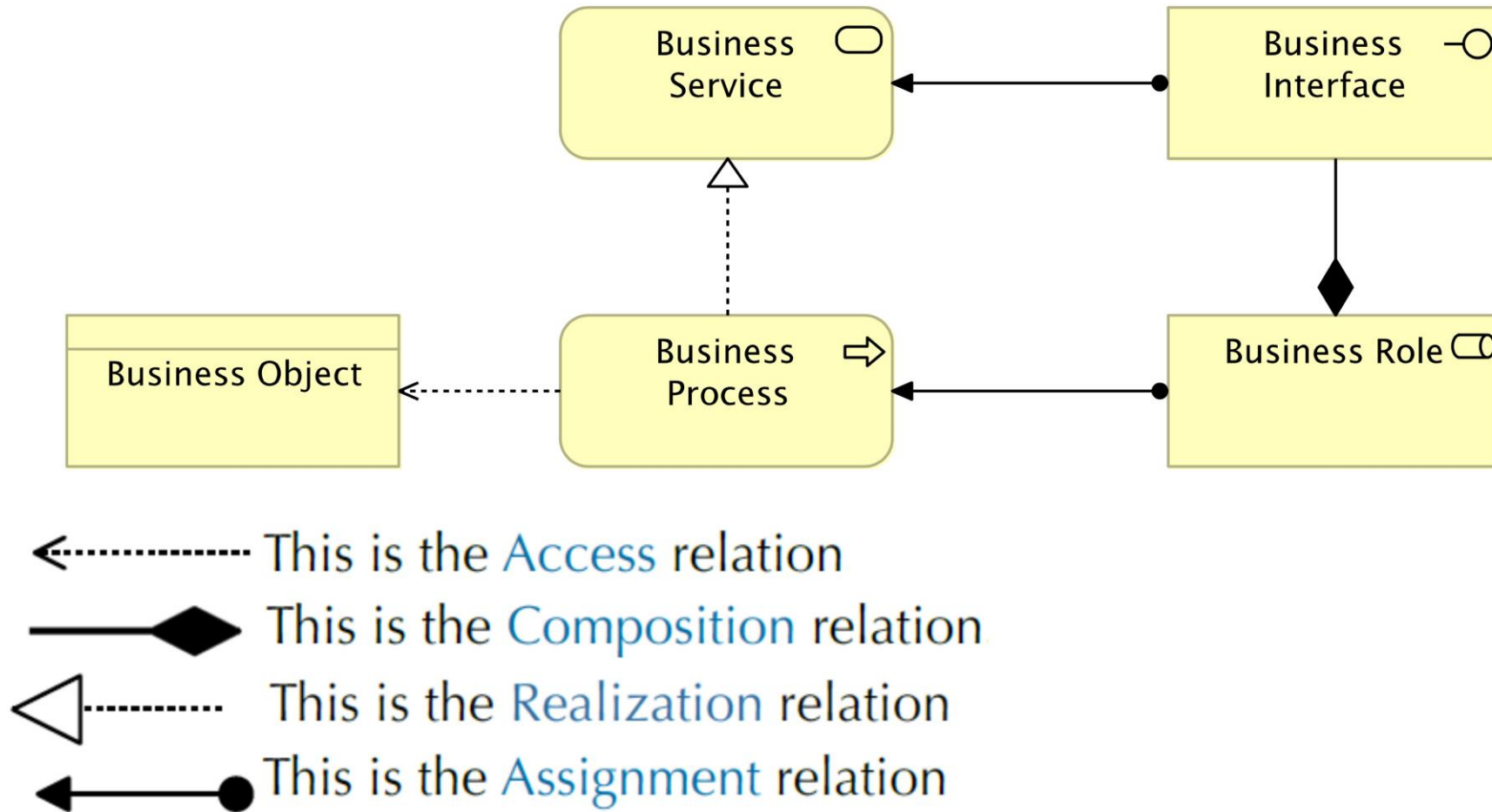
DA / A Entità dello stesso aspetto

L'entità in coda è una componente / parte indivisibile dell'entità in testa

Forza della relazione

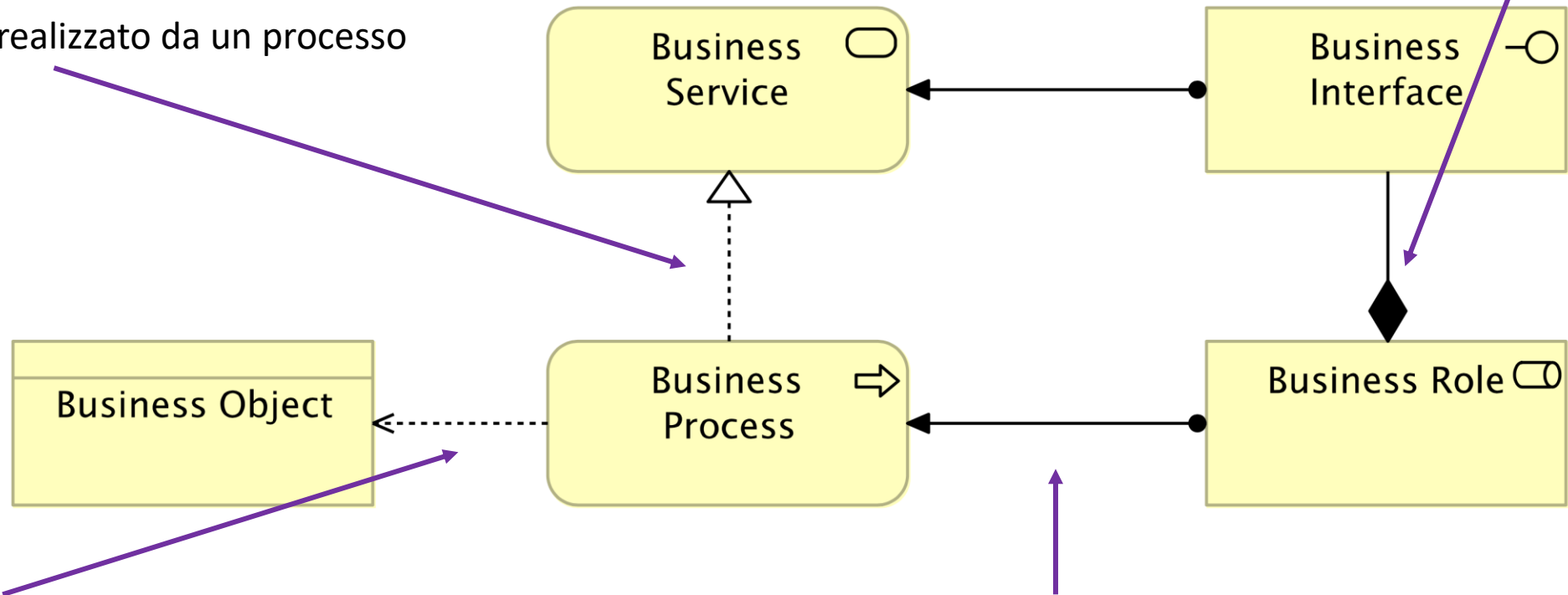


Pattern base (business layer)



Pattern base (business layer)

Il servizio viene realizzato da un processo



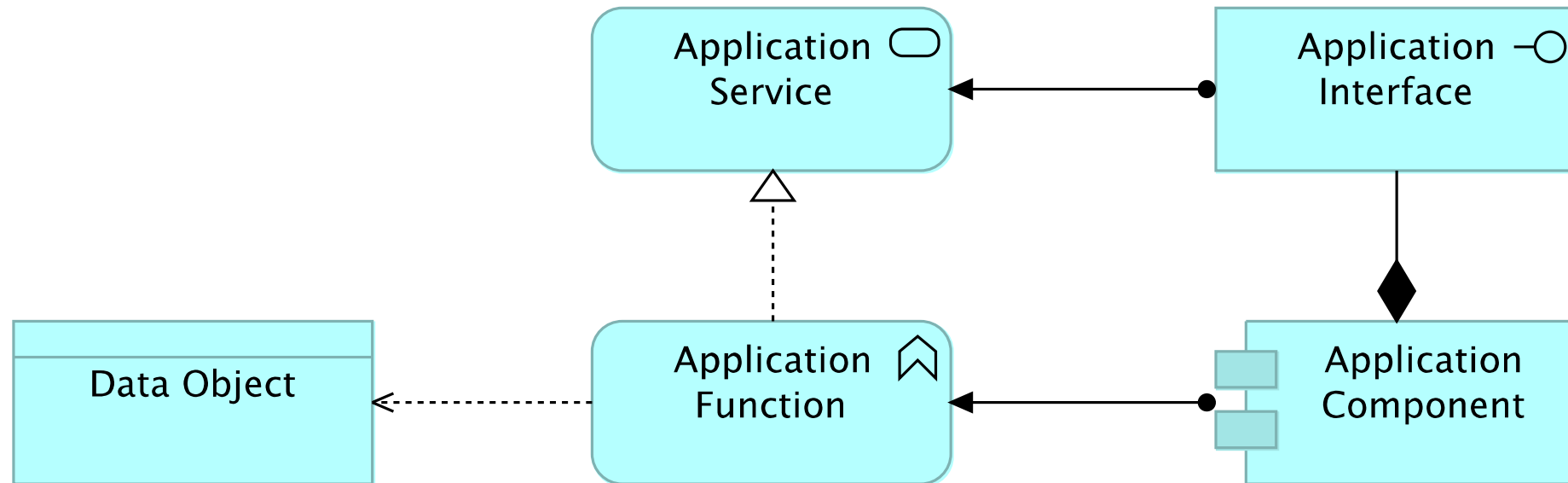
I ruoli compongono
un'interfaccia verso il
cliente

Il processo usa gli oggetti
come input e produce
degli output

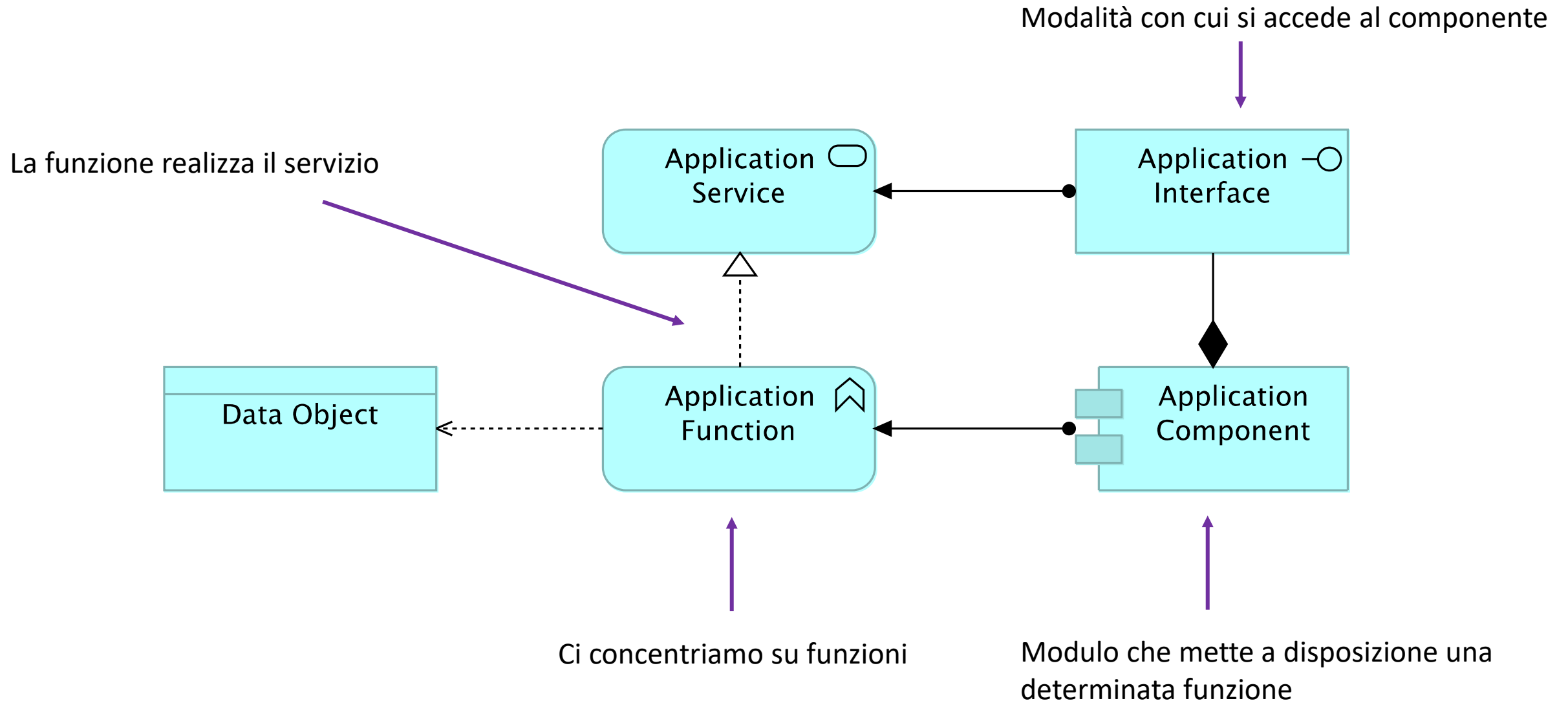
Il ruolo è assegnato ad un processo

Uso la composizione tra ruolo e interfaccia perché se elimino un ruolo plausibilmente elimino anche l'interfaccia. Invece servizio e processo hanno una relazione più debole: posso offrire lo stesso servizio cambiando il processo

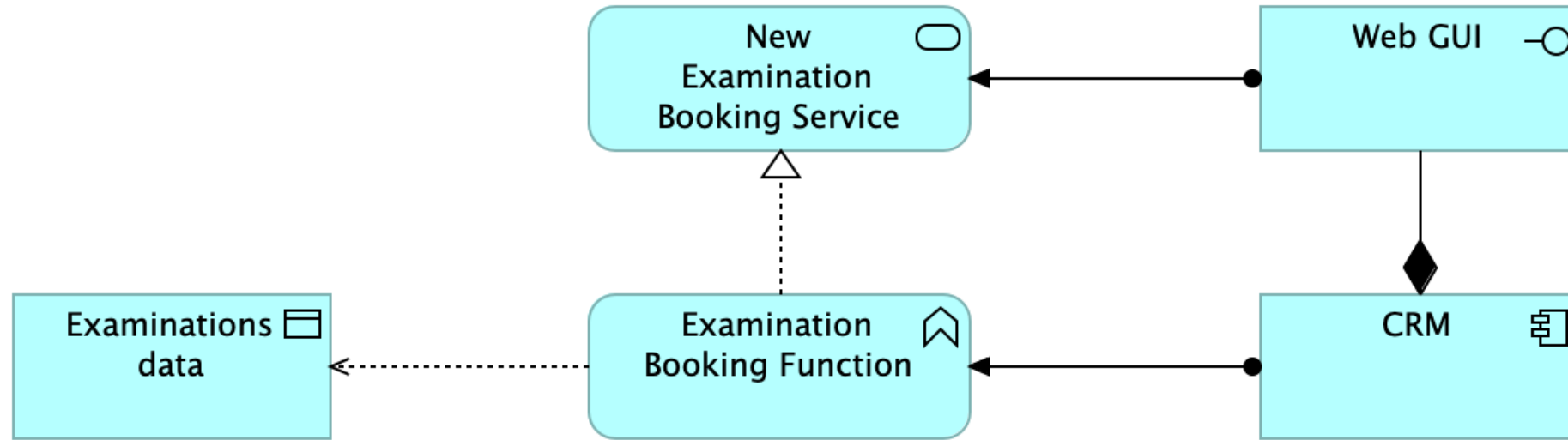
Pattern base (application layer)



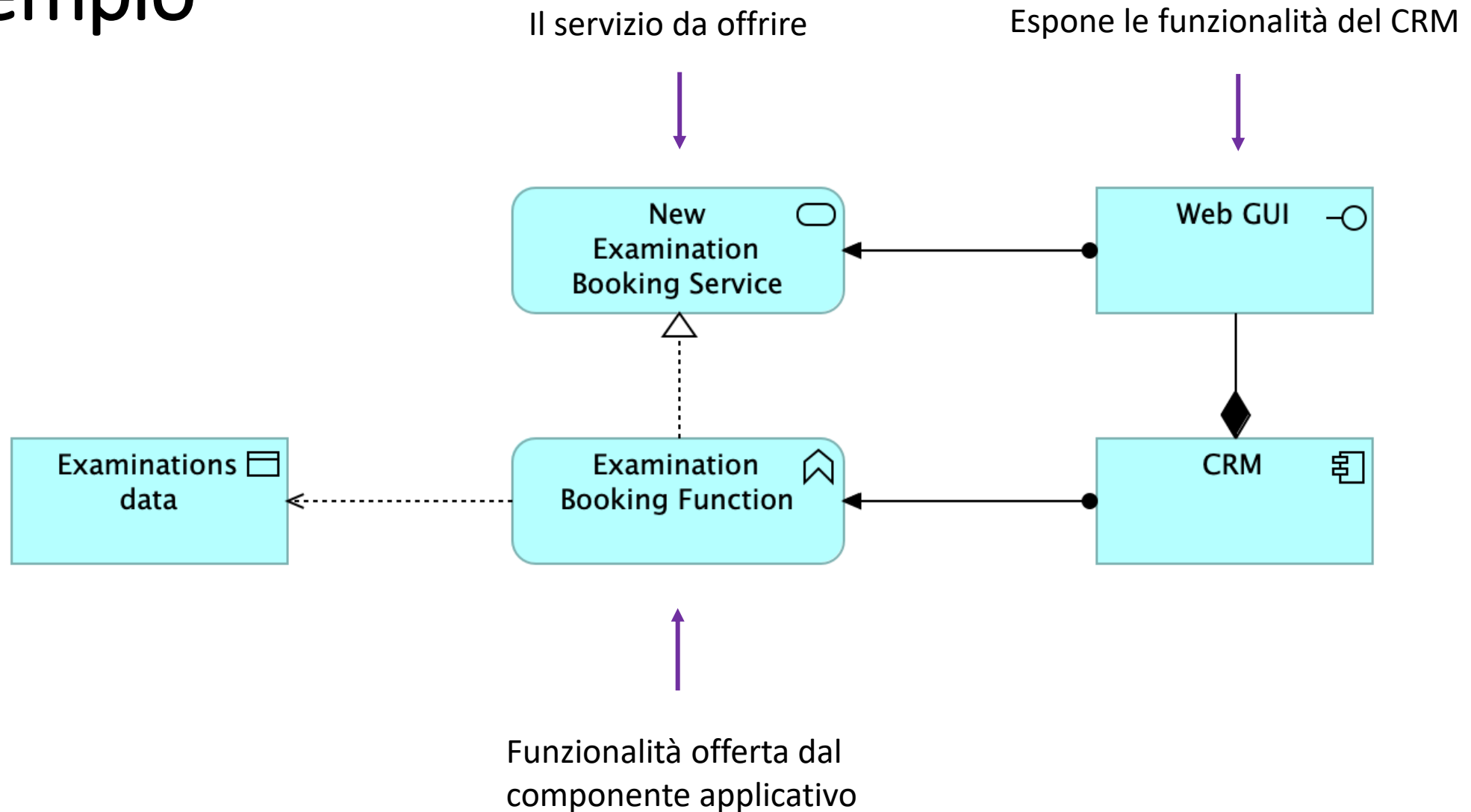
Pattern base (application layer)



Esempio

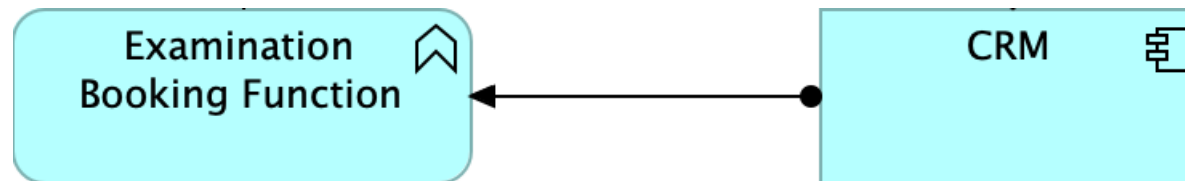


Esempio



Struttura e comportamento

- ArchiMate a livello application distingue tra
 - Elementi strutturali (structural elements) → *components*
 - *Mettono a disposizione le funzionalità*
 - Elementi comportamentali (behavioral elements) → *functions*
 - *Cosa caratterizza la funzionalità*
- Nell'application layer può essere difficile distinguere tra
 - Chi esegue (l'elemento attivo, i.e., il componente)
 - Cos'è eseguito (l'elemento comportamentale, i.e., la funzione o il processo)

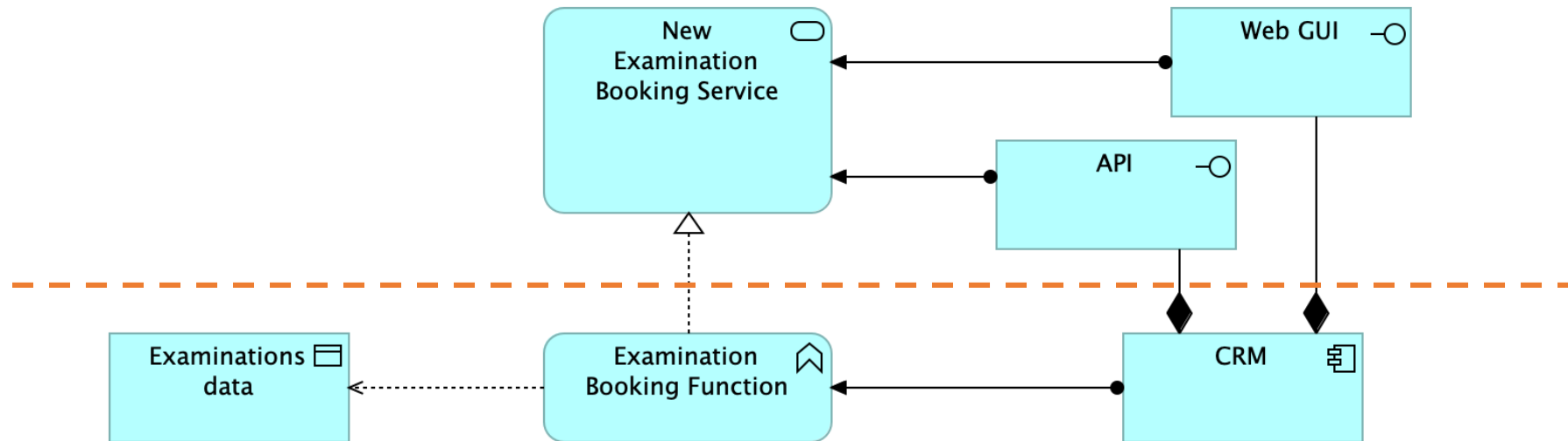


Funzioni e processi (application layer)

- Funzioni e processi sono eseguiti da un singolo componente
 - Se sono coinvolti più componenti, si usa una junction o un'interazione (collaboration) per renderlo esplicito
- Funzioni e processi sono aggregazioni → sono composti da altri elementi comportamentali
- Si usano le funzioni quando:
 - Gli elementi componenti hanno qualcosa in comune (es. il ruolo o le risorse)
- Si usano processi quando:
 - Gli elementi componenti ottengono collettivamente un determinato scopo (con un control flow)
- Per semplicità, nell'application layer useremo soltanto funzioni

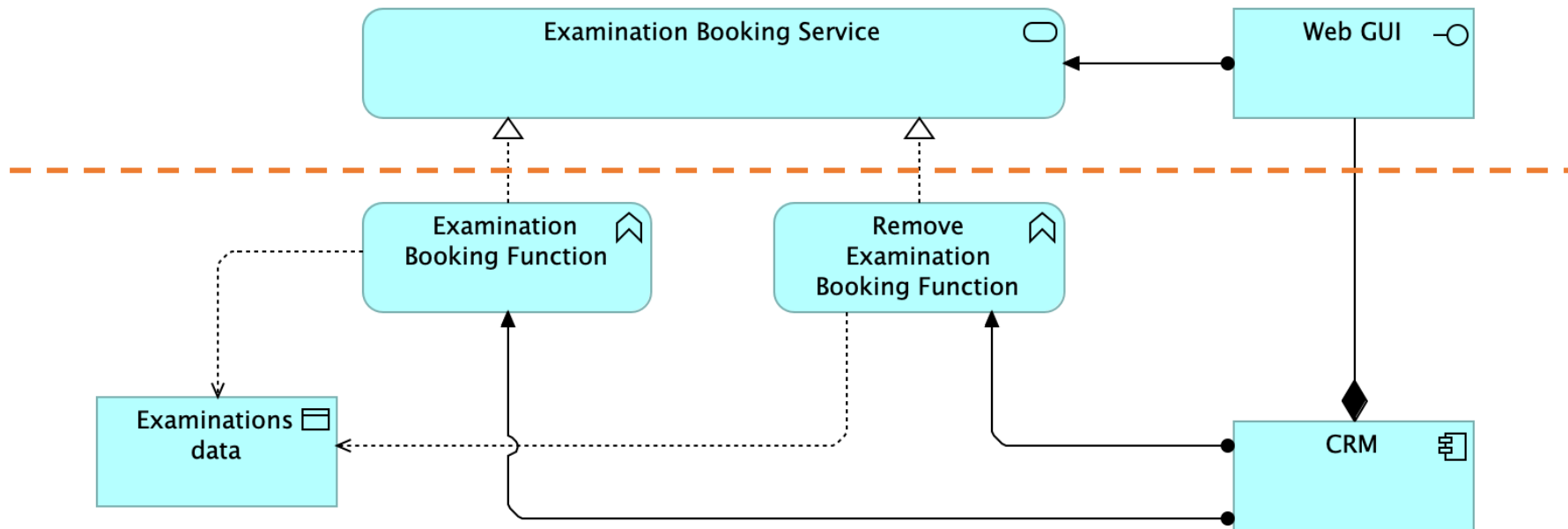
Sopra la linea di visibilità

- I processi o funzioni di business “vedono” un servizio accessibile attraverso una determinata interfaccia
- Le interfacce possono normalmente essere più di una (anche senza uso di junction)



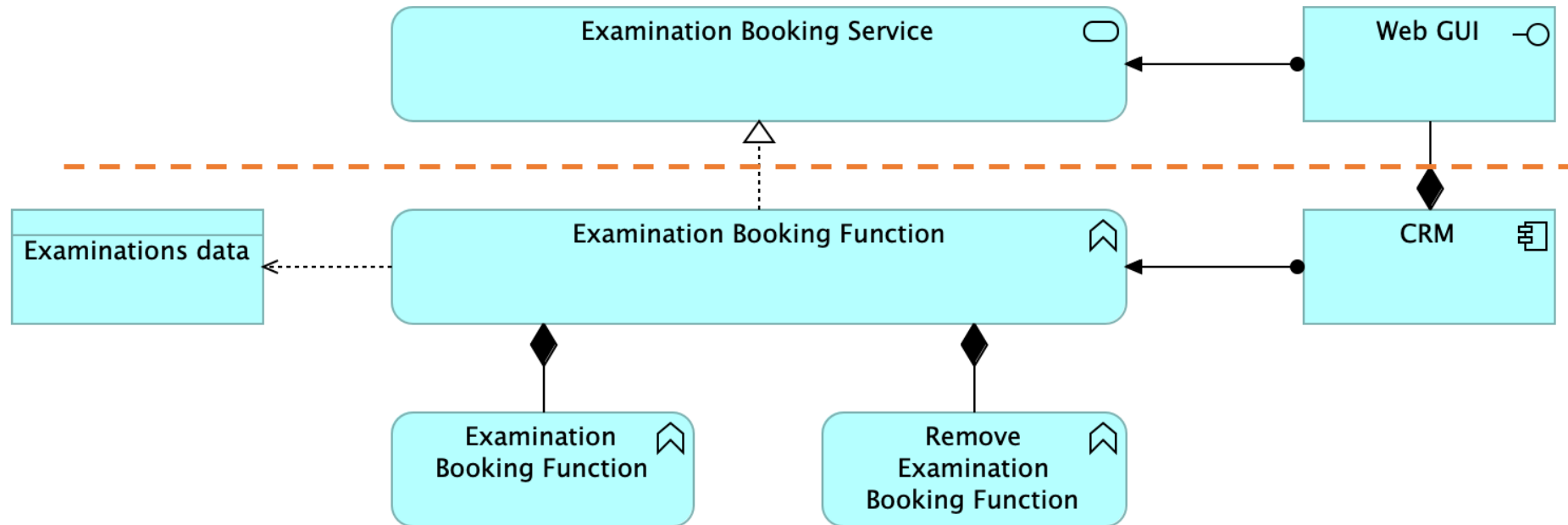
Sotto la linea di visibilità

- Come è articolato il servizio offerto, in termini di funzionalità e moduli in grado di offrirle
- Es. il modulo CRM offre due funzioni che concorrono a fornire il servizio (NB: le funzioni *non* sono visibili all'utilizzatore)

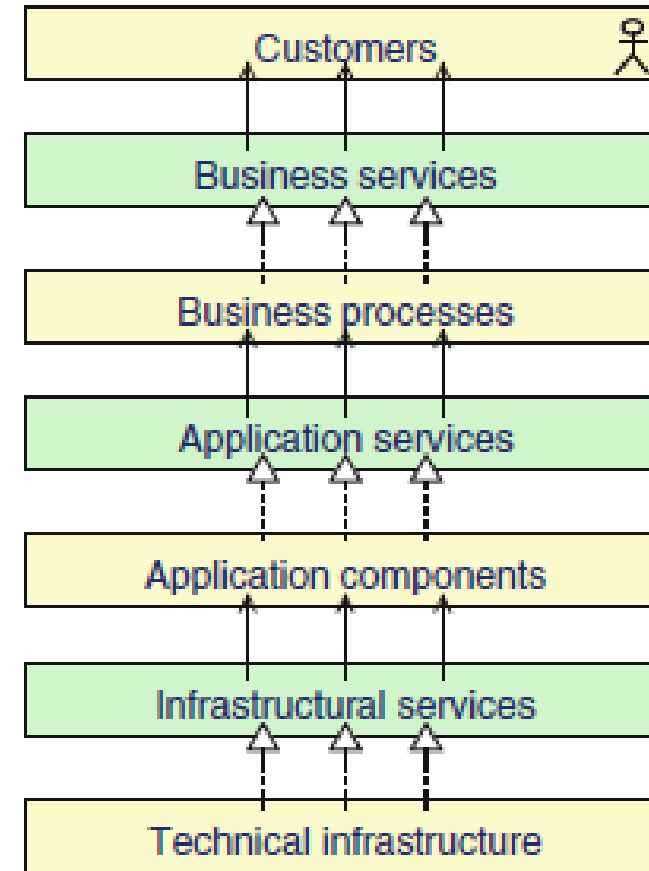
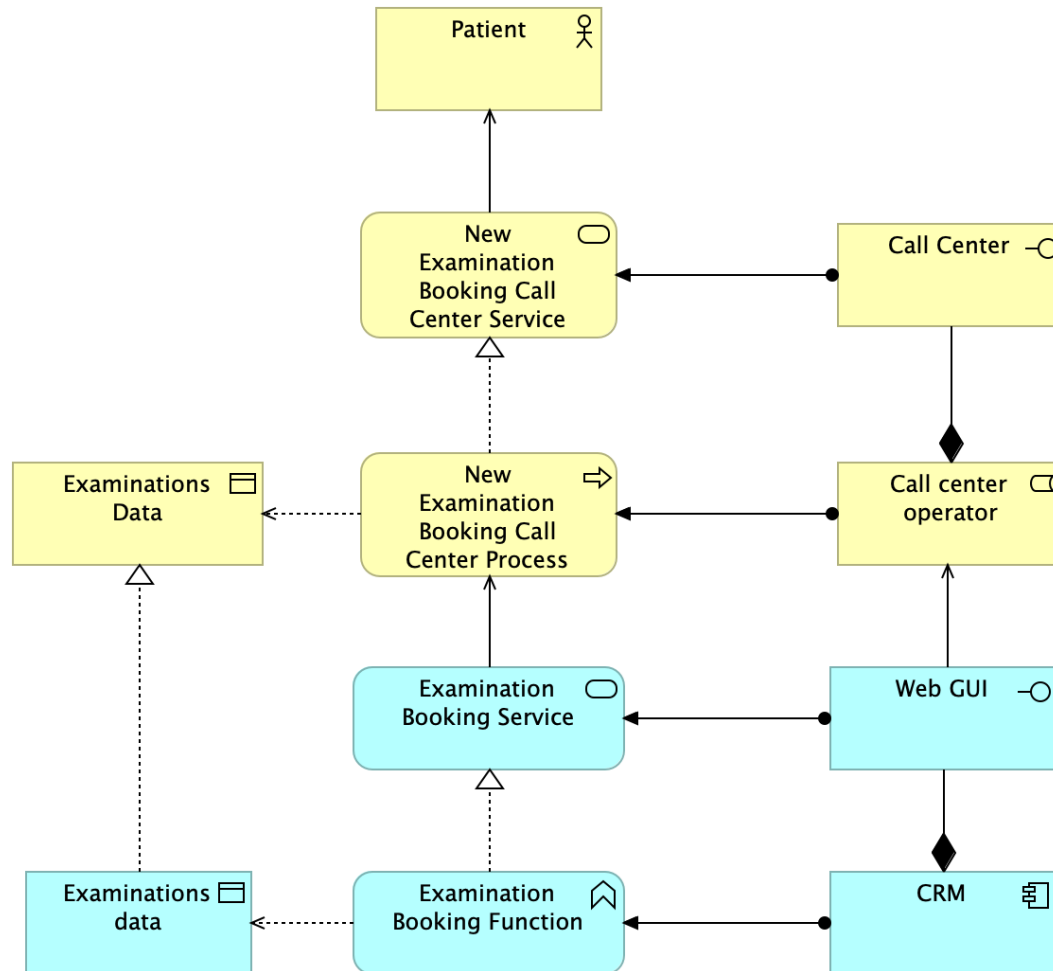


Sotto la linea di visibilità

- Si può utilizzare la composizione per aggregare le funzioni e rendere il modello più compatto

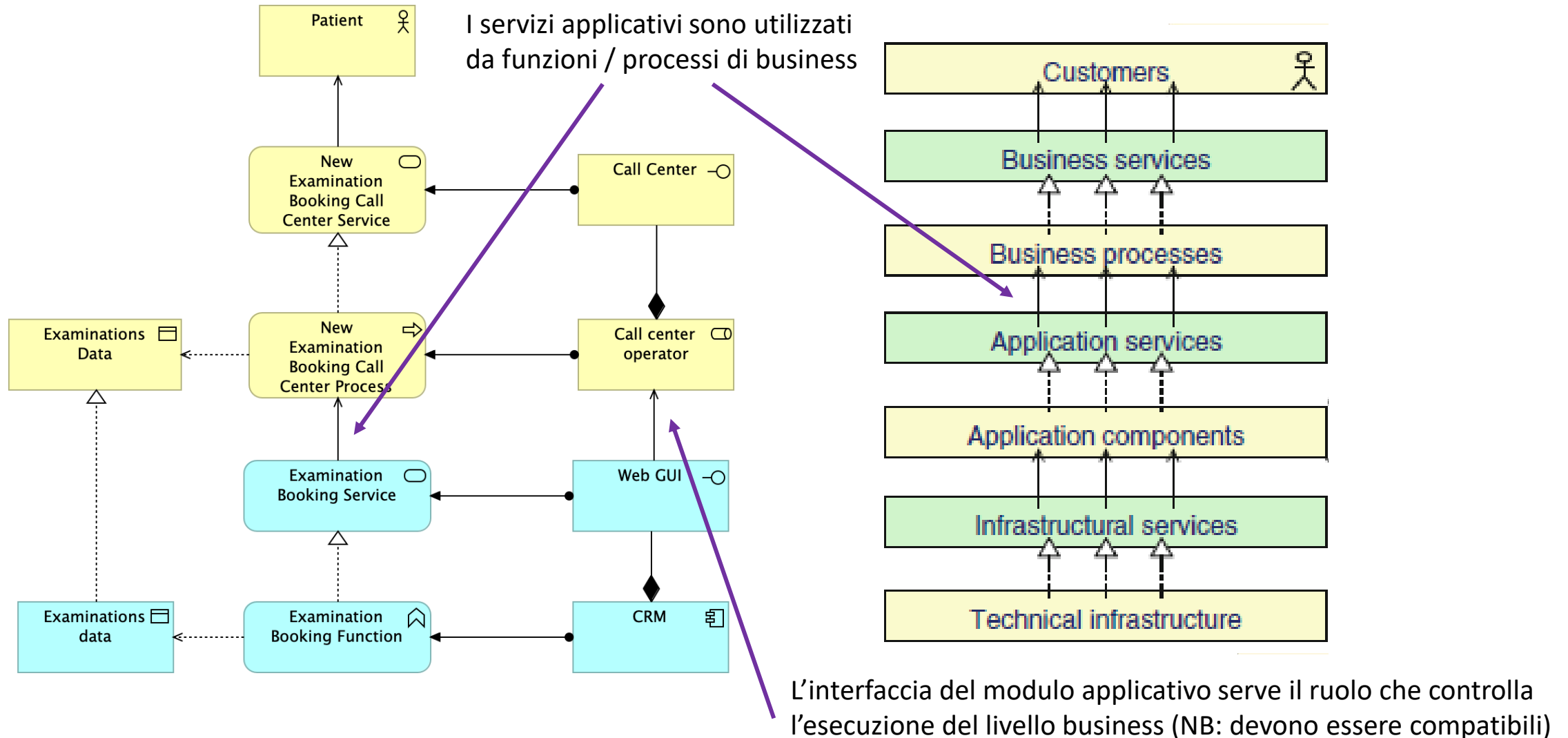


Connessione layer Application & Business

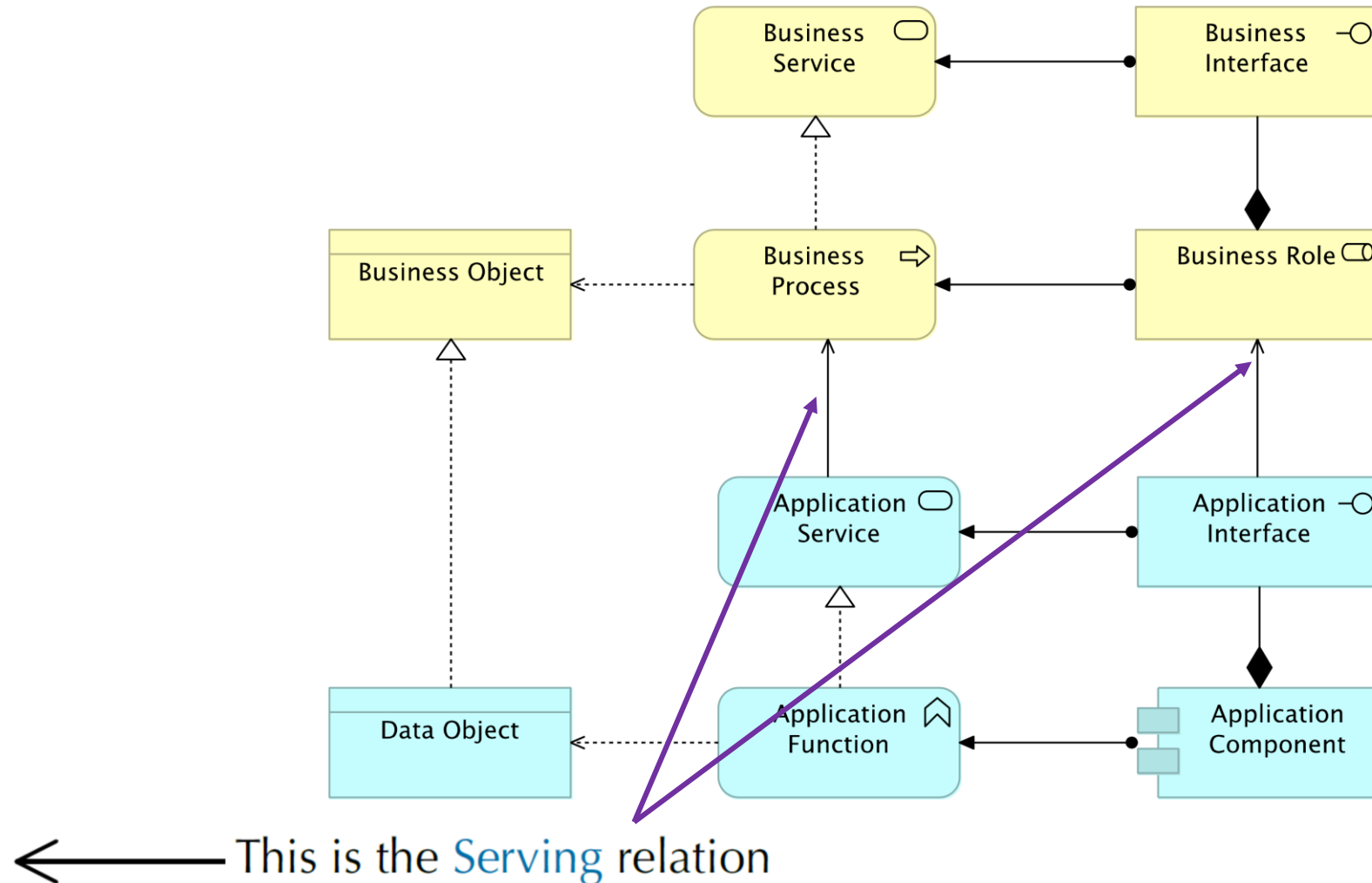


Connessione layer Application & Business

Un elemento visibile del pattern si collega con un elemento implementativo del layer superiore

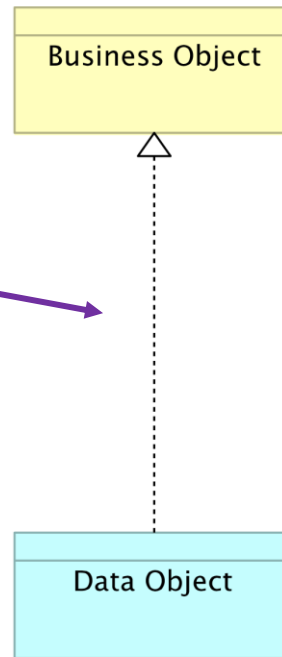


Pattern – Business + Application



Pattern – Business + Application

“Realizza” nel senso che il data object è la rappresentazione digitale di un business object



I *business object* rappresentano conoscenze o informazioni utilizzate e manipolate nel layer business, mentre i *data object* rappresentano dati che vengono memorizzati, elaborati e gestiti nel layer applicativo per realizzare dei *business object*

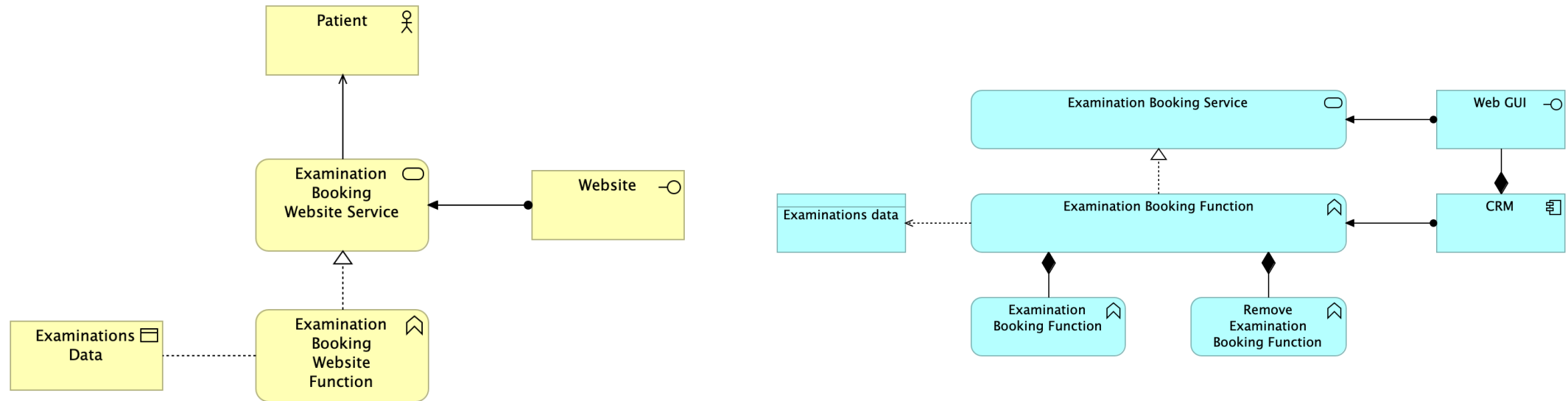
Esempi:

Profilo di un cliente → Record di un cliente

Fattura → Tabella dati fattura

Un servizio self-serve

A livello applicativo non cambia nulla, ma non c'è più un ruolo che accede all'interfaccia applicativa

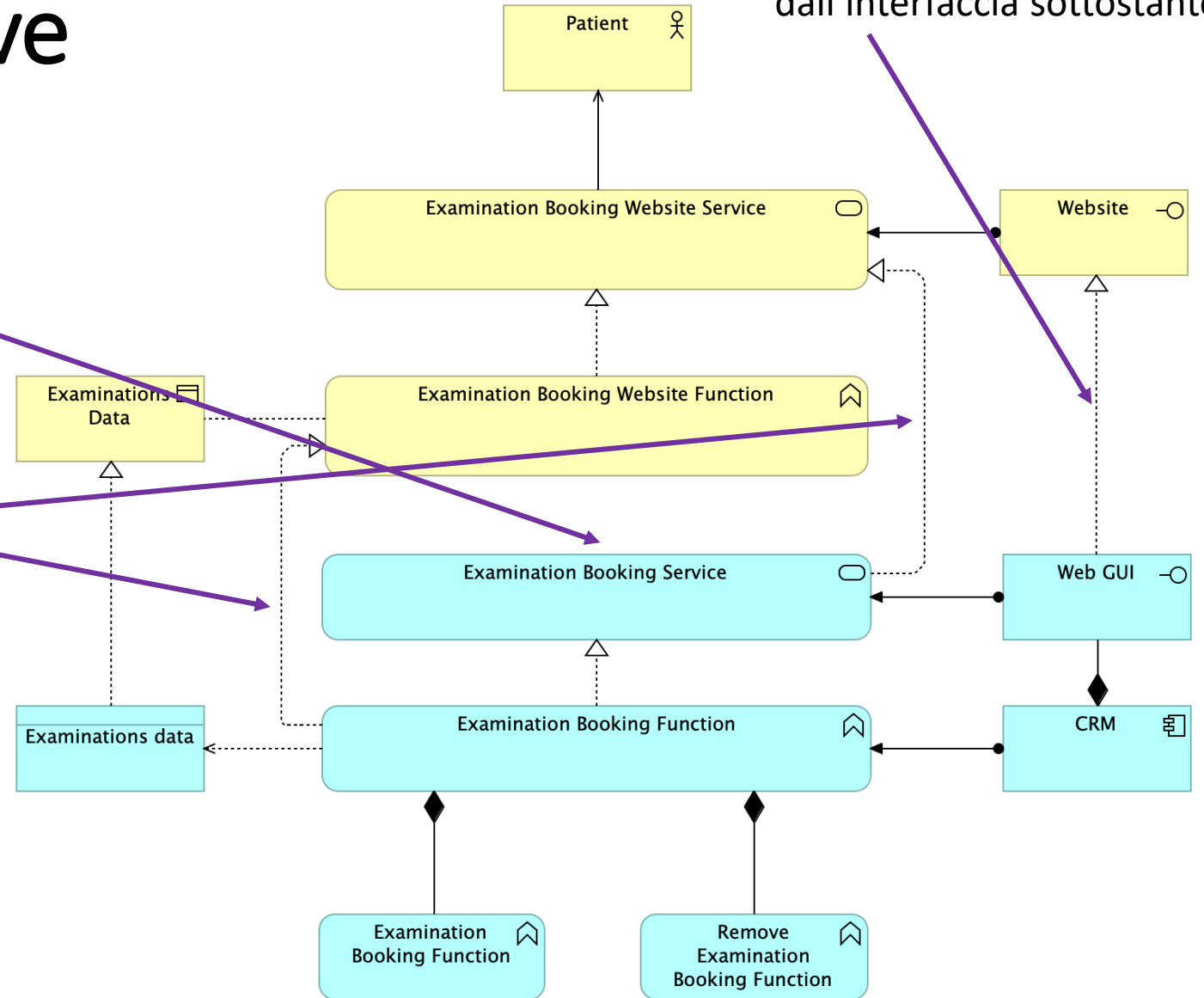


Un servizio self-serve

Manca il collegamento tra servizio applicativo
e implementazione del layer business

La funzione applicativa realizza la funzione di business (non ci sono “intermediari” delle funzionalità, il servizio applicativo viene “visto” direttamente dall’utente finale)

Il sito web è realizzato
dall'interfaccia sottostante

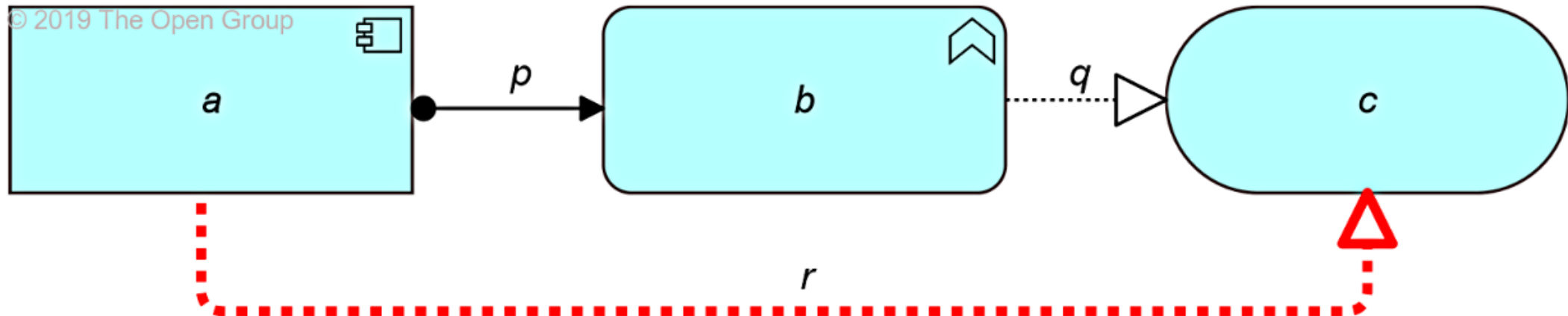


Relazioni Derivate

La regola per le relazioni strutturali e di dipendenza:

- *If two relationships $p(a,b):S$ and $q(b,c):T$ exist, with S and T being structural relationships, then a relationship $r(a,c):U$ can be derived, with U being the weakest of S and T .*
- *If two relationships $p(a,b):S$ and $q(b,c):T$ exist, with S being a structural relationship and T being a dependency relationship, then a relationship $r(a,c):T$ can be derived.*

Tutte le altre regole disponibili [qui](#)

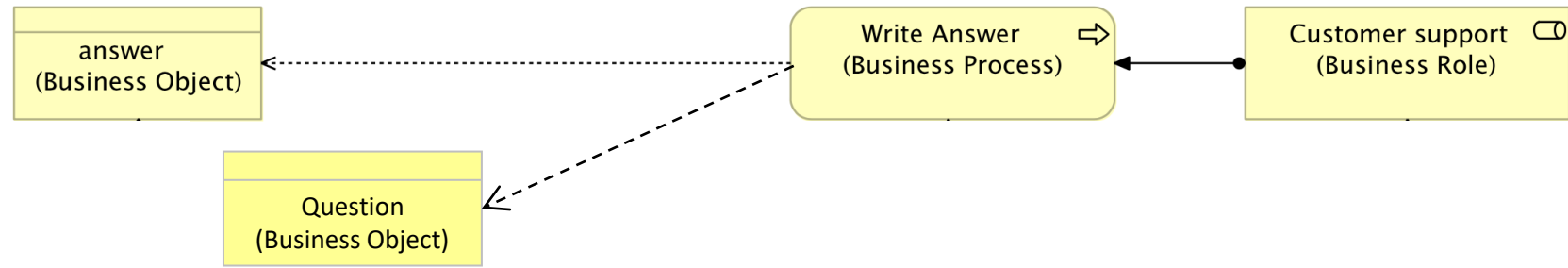


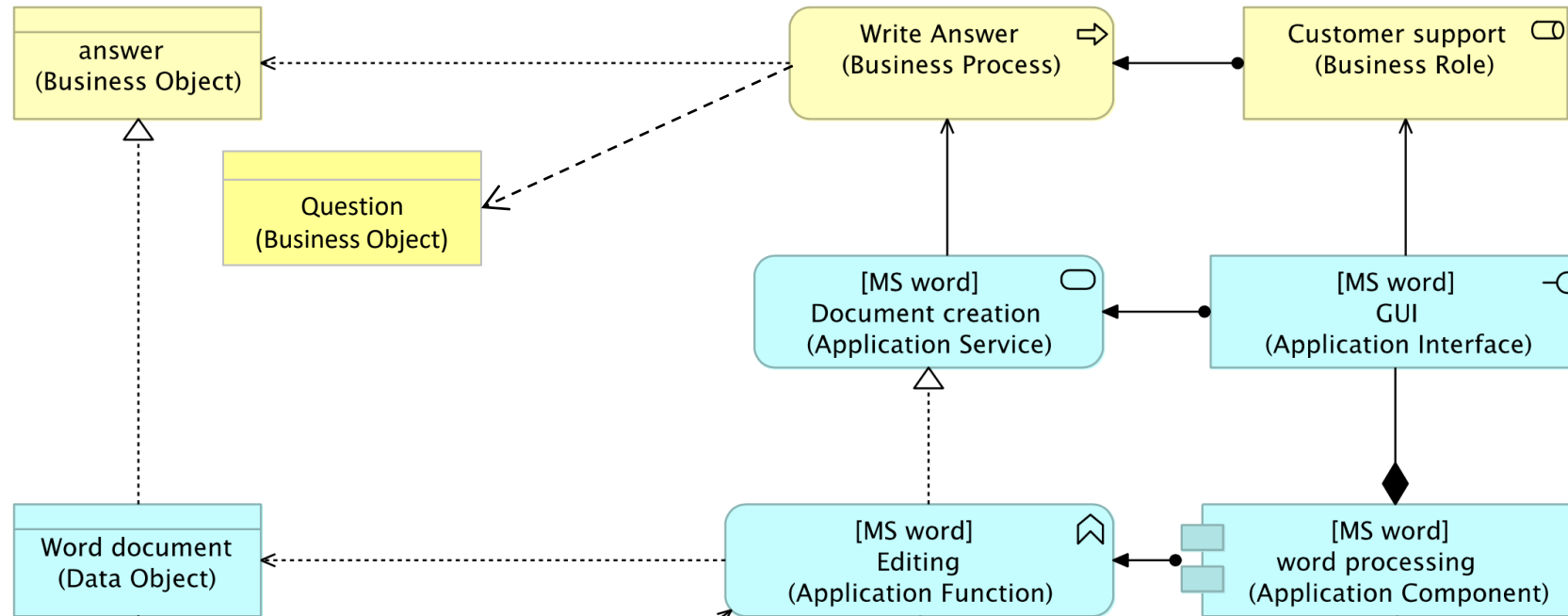
ArchiMate®

Esercizi con Archi

Esercizio – Risposta al cliente

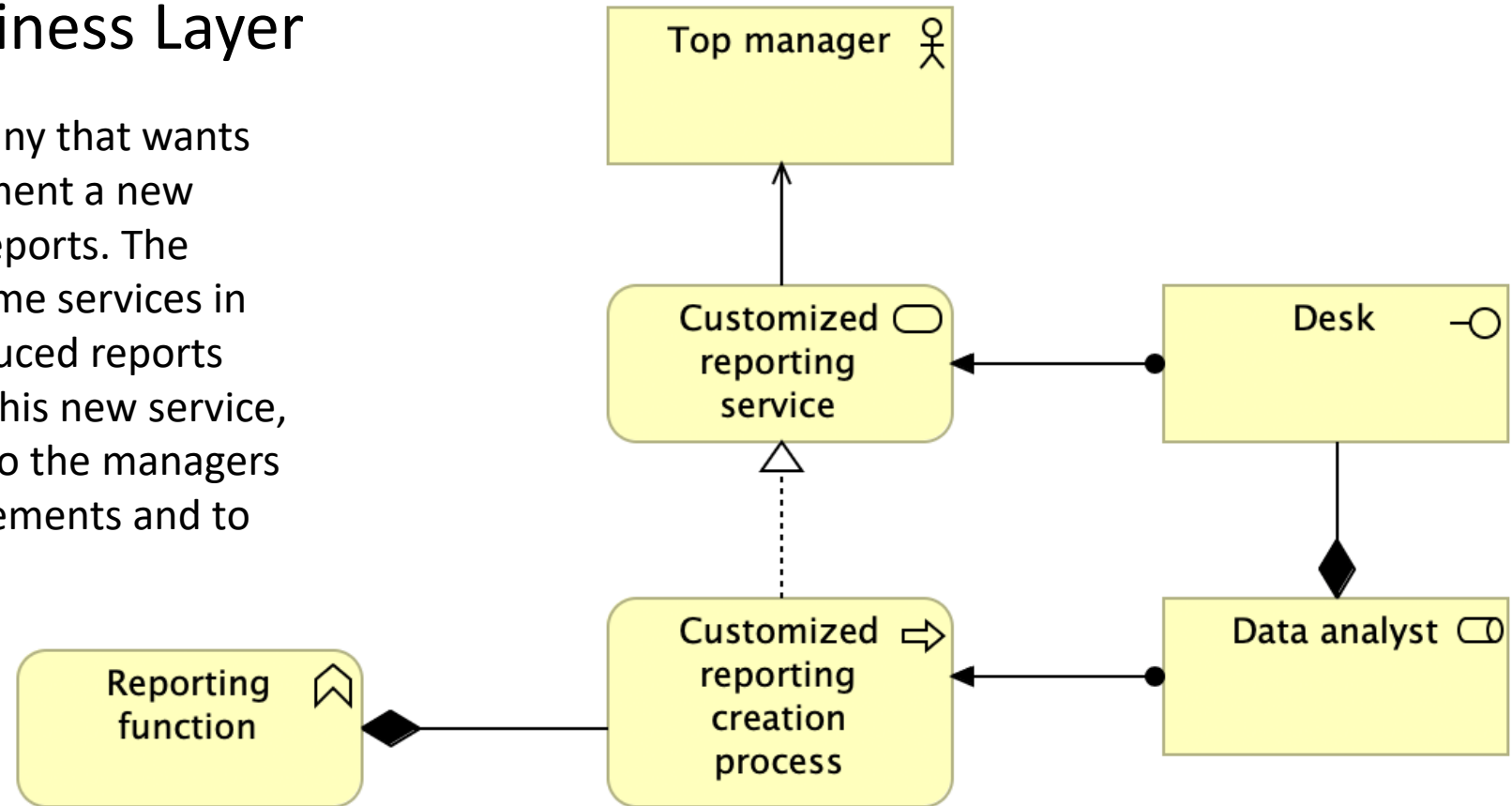
Si descriva tramite un modello ArchiMate l'Enterprise Architecture necessaria al servizio clienti di una piccola azienda, in merito alla necessità di redigere risposte alle domande provenienti dai clienti.





- Speedy 01 – Business Layer

Speedy is a delivery company that wants to offer to its top management a new service to create custom reports. The company already offers some services in this direction but the produced reports are not customizable. For this new service, a data analyst is available to the managers to understand their requirements and to develop the new report.



Archi

- Speedy 01 - Application layer

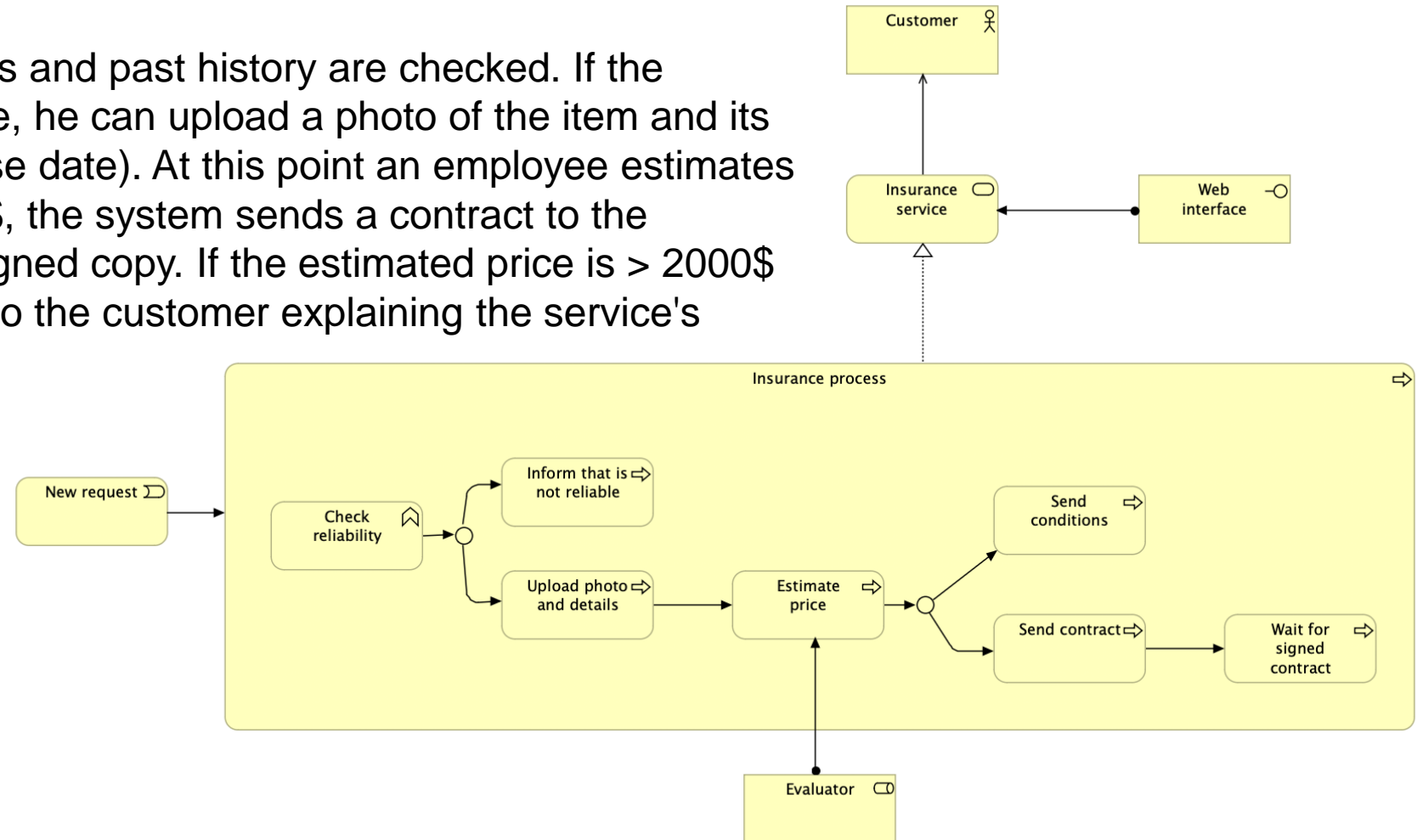
To offer this service, Speedy relies on a data warehouse.

Specifically, a reporting module, which is offered by the data warehouse, is in charge of interacting with the managed data marts to enable the activities which can be performed by the user. To this aim, the data warehouse periodically reads the data about the products from the organization's ERP system through an API, to create the data mart that is used by the reporting module.

• IC 01 – Business Layer

IC is an insurance company which wants to offer a new insurance service for small objects (< 2000\$) managed completely online for reliable customers. The new process starts when new insurance request arrives.

Then the customers credentials and past history are checked. If the customer is considered reliable, he can upload a photo of the item and its details (serial number, purchase date). At this point an employee estimates the item's price and, if < 2000\$, the system sends a contract to the customers and waits for the signed copy. If the estimated price is > 2000\$ the system sends a message to the customer explaining the service's conditions.



- IC 01 - Application layer

The IT system of the IC company is composed of a CRM system where information about the customers are managed to check the reliability.

Moreover, an ad-hoc program called "estimator" is made available through a desktop interface to the employee to estimate the price. Finally, a web application is available to support the finalization of the contract.