Lezione 8 modulo 4

In questo modulo riprenderemo la discussione su Archimate e, in particolare, andremo ad affrontare e a descrivere l'application layer, il secondo layer all'interno del framework core che caratterizza appunto Archimate. Le informazioni principali si trovano, come al solito, all'interno delle specifiche e in questo caso il capitolo a cui fa riferimento è il capitolo che è indicato appunto dall'url che trovate su questa slide. Come per il business layer, anche nell'application layer abbiamo tre elementi fondamentali, i famosi tre aspetti, che sono gli elementi attivi, gli elementi comportamentali e gli elementi passivi. In linea generale, la struttura poi di un diagramma che riguarda l'application layer non è molto diverso da quello che abbiamo visto per il business layer. Ovviamente, ci sono accezioni diverse, dovute al fatto che l'obiettivo di questo layer è descrivere qual è l'infrastruttura applicativa che permette di sostenere i processi di business, descritti appunto nel business layer. Entrando nel dettaglio, appunto, dei vari aspetti, gli elementi attivi, fra gli elementi attivi, perché ne vedremo solo un sottoinsieme, fra gli elementi attivi che considereremo in questo modulo, evidenziamo i tre mostrati sulla slide. In particolare, abbiamo l'application component, che è quell'elemento applicativo che rappresenta il modulo, l'elemento, diciamo così, più fisico, che garantisce l'esecuzione di una determinata funzionalità, l'offerta di una determinata funzionalità. Poi abbiamo l'application interface, che è il modo in cui quel module può essere acceduto; dato un application component, questo può essere esposto secondo diverse application interface. Anche nel business layer avevamo l'interface, la business interface e in quel caso , attraverso l'interface, noi indicavamo il mezzo con il quale il customer poteva andare a interagire con le funzionalità aziendali, i processi aziendali, che venivano forniti attraverso il concetto di business service. In questo caso, invece, parliamo di application interface. Per questo motivo, il range di valori che può assumere l'application interface, in questo caso diventa un po' più ristretto. A livello applicativo, un modulo può essere offerto principalmente secondo tre modalità: la modalità Grafical User Interface, la GUI, quindi un'interfaccia grafica che è accessibile quindi all'utente finale, un Application Program Interface, un API, quindi un'interfaccia programmativa richiamabile da altri moduli; ecco, qui poi, l'API può essere esposta attraverso diversi metodi, quali per esempio una REST API, oppure un API che è una signature messa a disposizione ad esempio via RMI con java; quindi anche l'API può avere dei sotto elementi e la terza classe viene rappresentata dal Command Line Interface, quindi un interfaccia a riga di comando, che è utilizzabile da un utente umano, ovviamente con un'esperienza maggiore, oppure può essere anche utilizzata per un'integrazione a livello applicativo, quindi può essere utilizzata da altre applicazioni per invocare le funzionalità offerte dal modulo. L'application collaboration è simile all'application collaboration che è stato visto anche a livello business e viene utilizzato ogni qualvolta io voglio evidenziare il fatto che uno più moduli concorrono nell'offerta di una funzionalità comune, attraverso una loro interazione. In questa slide vengono rappresentati alcuni esempi di relazioni che possono sussistere fra i vari elementi attivi; in questo caso, ad esempio, abbiamo che il nostro livello applicativo, portfolio applicativo, è composto da un CRM e da un sistema di back-office, il CRM viene esposto sia attraverso un'interfaccia web, intesa quindi come Graphical User Interface, per condurla a uno dei tre elementi che abbiamo visto prima e un API a livello applicativo. Mentre per quanto riguarda il back-office di interface è sempre una Graphical User Interface però legata a caratteri. In questo caso viene evidenziato come un collaboration, chiamato claim handler, rappresenta la collaborazione che esiste fra questi due moduli e viene rappresentata così come è stato visto anche nel business layer, quale aggregazione di due, in questo caso, o anche più, elementi attivi; quindi il claim handler è ottenuto grazie alla collaborazione di diversi moduli e questo permette di fornire ulteriori funzionalità. A livello comportamentale, anche in questo caso, ritroviamo alcuni elementi che abbiamo già visto a livello applicativo: c'è il concetto di processo, application process, il concetto di funzione, application function, il concetto di interazione, application interaction, che è un elemento che è legato abbastanza al concetto di collaborazione che abbiamo appena visto, il concetto di servizio applicativo e il concetto di evento applicativo. Come notate, non l'ho sottolineato prima, lo sottolineo adesso però è abbastanza evidente, perché è già stato citato quando abbiamo introdotto Archimate, utilizzeremo il

colore azzurro per indicare tutti gli elementi che fanno capo a livello applicativo. Questo in realtà è un di più, perché ripeto che la specifica è una specifica colorless, quindi dove il colore non ha nessun significato, però ci permette di addentrarci all'interno del fatto che qui si parla sempre di processi, ma a livello applicativo, che possono essere confusi con quelli di business perché la decorazione, quindi l'icona che viene utilizzata in alto a destra, nel nostro costrutto, è identico per quelli che sono sia il livello applicativo, che il livello di business. Il significato di application process, application function, application interaction, service e event, è abbastanza simile a quello che abbiamo visto a livello di business, con la differenza però che, ovviamente, si fa riferimento a un qualcosa più legato all'applicazione. Quindi, da un punto di vista meramente terminologico, l'application process è quello che il processo, visto anche a livello di business, quindi una sequenza di elementi comportamentali, in questo caso applicativi e non di business, che mi permettono tutti assieme di concorrere a uno specifico obiettivo. A livello invece di application function, quello che abbiamo è un'aggregazione di elementi comportamentali, in questo caso di tipo applicativo e non di business, che hanno in comune un certo tipo di aspetto, che possono essere le risorse utilizzate e tutte queste vengono possono essere in qualche modo raggruppate e andare a concorrere a una determinata funzionalità. E via via con tutti gli altri application service, nello specifico che è quello che utilizzeremo abbastanza spesso, è il modo con cui le funzionalità applicative sono esposte a chi poi le utilizzerà. Ovviamente esporre delle funzionalità applicative, richiede che l'elemento, o la persona, l'individuo, che va ad accedere a queste funzionalità, abbia l'interfaccia adeguata poi per accedervi. Per quanto riguarda gli elementi passivi ci concentreremo unicamente su un elemento, che è il Data object, che in qualche modo possiamo vedere come una rappresentazione dei dati strutturato, quindi più già vicino al concetto di database, che mi permette anche di andare a realizzare il concetto di business object, che invece è stato inserito all'interno appunto del business layer. Anche nell'application layer, e qui la semantica, così come la sintassi, è totalmente invariata, abbiamo il concetto di relazione, che è quell'elemento che mi permette, appunto, di legare diversi elementi, attivi, passivi, comportamentali, all'interno del mio diagramma. Come detto, le relazioni che abbiamo sono esattamente le stesse, hanno lo stesso significato, così come hanno lo stesso grado di forza, utile, questo, per definire quelle che sono le relazioni derivate. Per quanto riguarda quindi le relazioni, non è cambiato nulla rispetto a quello che abbiamo visto nel business layer. Sempre facendo riferimento a quello che è stato visto nel business layer, anche qui abbiamo un pattern base, un pattern di riferimento, che è molto utile quando si inizia a costruire il diagramma a livello applicativo, che è molto simile a quello che abbiamo visto a livello business. In questo caso però, non abbiamo ovviamente il concetto di business service, ma avremo l'application service, non abbiamo la business interface ma l' application interface, non abbiamo un ruolo, ma abbiamo in questo caso l'application component e qui l'elemento comportamentale vedremo che utilizzeremo non tanto il processo, anche se è possibile ma la funzione. Anzi, per semplicità eviteremo di utilizzare gli application process, ma rimarremo su qualcosa di più specifico che è l'application function. E infine avremo i data object, che sono gli elementi acceduti dall'application function, per andare a realizzare quello che è il servizio che voglio offrire a chi poi sta sopra e chi sta sopra il business layer. Poi vedremo che nel momento in cui andremo a realizzare un servizio automatizzato o meno, il modo con cui vado a accedere a questo servizio potrebbe cambiare. Quindi, in linea generale, quello che noi abbiamo è un pattern abbastanza simile a quello che abbiamo visto prima, se non identico, dal punto di vista strutturale, che mi racconta che cosa? Il fatto che un determinato servizio applicativo può essere acceduto dall'utilizzatore attraverso un'interfaccia, questa è un'interfaccia, il modo in cui un componente può essere acceduto per andare ad accedere, in realtà, a quella che è la funzione messa a disposizione appunto dal componente. Per fornire questa determinata funzione, l'elemento comportamentale, sempre per rispettare la solita regola, per la quale un elemento attivo non dovrebbe essere collegato direttamente a un elemento passivo, appunto l'elemento comportamentale governa l'accesso ai dati. Un esempio di applicazione di questo basic pattern, che si riferisce a quell'esempio che abbiamo già visto per quanto riguarda il business layer e che riguarda la prenotazione di esami all'interno di un ospedale, è rappresentato appunto nel diagramma nella slide. Qui io sto semplicemente dicendo che ho un CRM, che

espone le sue funzionalità attraverso una Graphical User Interface, nello specifico questa Graphical User Interface ha il compito di mettere a disposizione l'accesso al New Examination Booking Service quale servizio applicativo, messo a disposizione, poi lo vedremo come collegarlo a livello di business e, la cosa importante da capire, è che il mio CRM ha il compito di fornire questa funzionalità, oppure, vedendola al contrario, questa funzionalità è offerta dal CRM. Per poter offrire questa funzionalità, la funzionalità ha bisogno di alcuni dati, che sono i dati riguardanti appunto gli esami, gli appuntamenti e quant'altro. Qui è utile fare un approfondimento sul concetto di struttura e il concetto di comportamento. È utile, a livello applicativo, fare una distinzione fra gli elementi strutturali e gli elementi comportamentali. E in particolare, questo perché in Archimate è importante distinguere fra quello che è il componente che offre una funzionalità e la funzionalità. Molto spesso questi due elementi vengono confusi, se io dico il CRM, il CRM è quell'elemento che mi permette di gestire la mia clientela, il processo di interazione con la clientela, fidelizzazione, campagne di marketing e quant'altro. Ecco, è utile, all'interno della modellazione con Archimate, distinguere quello che è l'elemento attivo, cioè l'elemento, potremmo dire fisico, anche se un modulo applicativo non è nulla di fisico, rispetto a quella che è la funzionalità che offre questo elemento fisico. Quindi abbiamo il CRM, potremmo immaginarlo anche se in maniera impropria, perché poi vedremo che nel momento in cui introdurremo il livello tecnologico questo aspetto va rivisto un attimino, il CRM può essere visto come il programma che mette a disposizione un insieme di funzionalità. E questo deve essere in qualche modo diviso, quindi devo dividere, devo distinguere fra chi mette a disposizione una funzionalità, chi svolge quella funzionalità e cosa è svolto, cosa caratterizza effettivamente questa funzionalità. Ecco, all'inizio non è immediato fare questa distinzione, ma vedremo che attraverso degli esempi diventerà abbastanza intuitivo. È altrettanto utile, anche a livello applicativo, fare un ragionamento rispetto alla differenza che esiste fra funzioni e processi. È stato un qualcosa che abbiamo visto anche a livello di business, che ci ha permesso di capire e di approfondire il fatto che funzioni e processi non sono la stessa cosa, le funzioni sono un'aggregazione di elementi comportamentali che hanno qualcosa in comune, mentre i processi sono una aggregazione di elementi comportamentali raggruppati secondo un control flow e che mi permettono di concorrere tutti assieme, questi elementi comportamentali, a un obiettivo comune fra tutti. Ecco, utilizzare questo tipo di distinzione, che è ancora valido anche a livello applicativo, diventa un po' più complicato, perché diventa un po' più complicato capire il concetto di processo. Perché è anche ovvio che una determinata funzionalità può anche andare a essere descritta attraverso tutti quei passi che sono richiesti per l'esecuzione di alcuni algoritmi, piuttosto che alcune funzionalità, per andare a raggiungere quella che è la funzionalità generale. Ecco, molto spesso questo crea, anche per esperienza, abbastanza confusione; quindi, per essere un po' più semplici ma comunque immediati e completi nella descrizione, noi partiremo sempre dal presupposto che al livello applicativo, quindi a livello di business vale ancora quello che abbiamo detto, utilizzeremo unicamente delle funzioni e non i processi. I processi in realtà potrebbero venire utili nel momento in cui andremo a modellare situazioni in cui un processo di business è totalmente automatizzato e dietro c'è un orchestratore che mi permette di gestire tutte queste attività che vanno a comporre il mio processo. Però vanno un po' oltre quello che vogliamo fare in questo corso. Così come visto all'interno del business layer, anche qui è utile distinguere quello che accade sopra la visibility line e quello che accade sotto la visibility line, perché il pattern base che abbiamo utilizzato anche in questo caso, in qualche modo può avere due anime: un'anima, diciamo così, pubblica, quello che viene visto da chi poi utilizzerà il servizio, quindi vede il servizio e la sua interfaccia, e un'anima privata, che è l'implementazione di questo servizio. Ecco, in questo caso noi possiamo avere, a livello applicativo, così come è stato descritto anche all'interno della specifica di Archimate, che un servizio può anche essere messo a disposizione da più interfacce. Quindi in questo caso io sto dicendo che questo servizio può essere accessibile sia attraverso un'interfaccia grafica, sia attraverso un'interfaccia programmativa e questa è una situazione abbastanza normale per quanto riguarda le applicazioni oggi giorno. È naturale pensare che entrambe queste due interfacce vengano offerte dallo stesso modulo applicativo, dallo stesso componente. Quindi, al livello sopra la visibility line, io vado a dettagliare quello che è il mio servizio e quelle che sono le interfacce grazie alle quali io posso

accedere a questo servizio. Come detto, è ammissibile avere più di un'interfaccia, quindi il concetto di junction per mettere insieme l'assegnamento, così come l'abbiamo visto a livello di business non è in questo caso richiesto. Dietro la visibility line io avrò una situazione più o meno complessa a seconda del servizio che voglio mettere a disposizione, andando a rielaborare un minimo quello che abbiamo visto nell'esempio iniziale, potremmo per esempio dire che l'examination booking service mi permette non solo di aggiungere un nuovo esame quindi far riferimento a una funzionalità di examination booking function, ma anche di rimuovere un esame prenotato, attraverso un'altra funzionalità, remove examination booking function; in questo caso io sto dicendo che il CRM offre entrambe queste due funzioni e queste due funzioni concorrono nella fornitura di questo servizio. Questo servizio offre queste due funzionalità attraverso un'interfaccia, in questo caso l'interfaccia grafica. Entrambe queste due funzionalità fanno riferimento a questo data object per poter appunto offrire questa funzionalità. In questo modo io cosa sto dicendo però? Sto dicendo che esiste un servizio generico che fa riferimento però a due funzionalità che non sono visibili, perché ricordiamo che sono dietro la visibility line, non sono visibili a chi poi lo utilizzerà, qui c'è ancora un punto di domanda su chi lo utilizzerà e fra un po' risponderemo, scioglieremo il dubbio su questo punto di domanda. Non sono visibili, quindi examination booking service è in qualche modo un collettore che mi nasconde quello che è uno e quello che è l'altro; cioè, almeno a livello applicativo, io non vedo che ci sono, a livello dell'utilizzo di questo servizio applicativo, non vedo che ci sono queste due funzionalità distinte. Vedo che ci sono due funzionalità ma non so che dopo, dietro le quinte saranno effettivamente distinte. Posso rappresentare la stessa cosa anche in questo modo: utilizzo il concetto di funzione quale aggregazione di elementi comportamentali che hanno qualcosa in comune, in questo caso tutti concorrono alla realizzazione di un servizio comune e utilizzo la composizione per dire: guarda che questa funzione è composta da queste due sottofunzioni. Quello che ho scritto prima è identico dal punto di vista del contenuto a quello che ho scritto adesso, forse però una versione un po' po' più compatta e mi aiuta anche a dire: guarda che il CRM è in grado di, ha l'obiettivo di offrire tutte queste due funzionalità e questo non è altro che un raggruppatore che mi permette anche di dire guarda che l'examination booking service viene implementato da questa funzione e dietro le quinte questa funzione è l'aggregazione, o meglio la composizione di queste due sottofunzioni, di creazione e di rimozione del booking function. Poi vedremo attraverso qualche esercizio anche come si può lavorare in dettaglio su questo tipo di aspetto. Andiamo a questo punto a sciogliere il dubbio rispetto a chi è l'utilizzatore del servizio. E qui è utile andare a riprendere quel layering che abbiamo visto quando abbiamo introdotto Archimate, applicato, in questo caso a due layer, che sono quello applicativo e quello di business, attraverso quell'utilizzo dei pattern che abbiamo introdotto. Se vi ricordate, avevamo detto che ogni layer è diviso in due aspetti l'aspetto di offerta, visibile, quello dei servizi e quello dei processi che è l'implementazione di questi servizi, a livello applicativo abbiamo i servizi e abbiamo i componenti che realizzano questi servizi, i componenti attraverso le funzioni che questi componenti offrono. Quindi, cos'è che sto dicendo? Sto dicendo che i servizi applicativi sono utilizzati dai processi o dalle funzioni di business, quindi i servizi servono questi. E questo non fa altro che mappare esattamente questo elemento, application service serve i business process, o business function, dipende da quello che ho messo qui. Al tempo stesso io vado a dire guarda che la interfaccia che viene esposta dal modulo, serve il ruolo che è in capo, l'esecuzione delle attività presenti nel business process. Quindi, è ovvio pensare e dobbiamo sempre ricordarcelo, che se io metto una freccia di questo tipo, deve esserci compatibilità fra questa interfaccia e questo ruolo; se questo ruolo, come spesso avviene, se non sempre è un ruolo umano, questa interfaccia deve essere necessariamente un'interfaccia, o Command Line Interface o Graphical User Interface. Non posso avere un'Application Program Interface, perché se avessi un'Application Program Interface, l'utente non sarebbe in grado di utilizzarlo. Per questo motivo l'application program interface il più delle volte viene utilizzata per garantire l'integrazione, per mettere l'integrazione fra moduli applicativi diversi, quindi viene usato più nel back-end che non nel front-end e vedremo qualche esempio in uno dei prossimi moduli. quindi questo è il modo con cui io LEGO i due livelli; vediamo che solo l'elemento visibile del mio pattern va a collegarsi con l'elemento implementativo

del livello successivo. Quindi l'elemento implementativo del livello successivo non vede quello che succede dietro le quinte, che è un po' la stessa cosa che abbiamo detto quando abbiamo parlato di patient, che non vede quello che succede qui sotto. Quindi l'utente finale, che è il mio paziente, non vede il dettaglio e tantomeno non vede quello che succede qui. Il livello applicativo, in questo caso, è totalmente nascosto dal paziente, perché il paziente ha come unico punto di riferimento il call center operator, interagisce attraverso il call center. Questa è una delle possibilità, perché poi vedremo che è possibile anche modellare situazioni in cui l'utente finale interagisce direttamente con il sistema applicativo. Questo è abbastanza comune, perché nel momento in cui io voglio offrire un servizio via web a qualunque utente, quindi l'utente accede al mio sito web, in questo caso io non ho nessun ruolo che supervisiona quello che è è il mio processo, ma il mio processo è direttamente implementato da alcune funzionalità, visibili attraverso un determinato servizio applicativo, dal mio utente finale e fra poco lo vedremo. Quindi, questo è il primo caso in cui ho un insieme di elementi umani che fanno un po' da tramite rispetto a quelli che sono i sistemi informativi aziendali. Poi abbiamo anche, non lo approfondiremo troppo, perché è una dimensione, diciamo così, che richiede abbastanza tempo poi per essere analizzata in dettaglio, però in linea generale noi possiamo dire che un determinato data object realizza un determinato business object; poi da cardinalità che può esistere fra business object e data object è abbastanza varia e lo vedremo caso per caso. Passiamo quindi al caso, però del self server, cioè la possibilità che un paziente, in questo caso, utilizzando questo esempio, attraverso un sito web, possa eseguire questo processo. Questo processo che non è altro che il processo che viene esposto attraverso questo determinato servizio. A livello applicativo, rispetto a prima, non cambia assolutamente nulla, perché a livello applicativo quello che io ho è la stessa cosa: ho un un servizio offerto via web. Ecco, rispetto a prima la cosa che cambia è che io non ho nessun call center operator qui che fa da tramite e la mia domanda adesso è: come posso rappresentare la situazione in cui l'utente finale può andare a lavorare direttamente su questa web Graphical User Interface che va a realizzare quel website che non è altro che l'interfaccia poi di business? Però questa non è altro che questa qua, quindi web Gui e website sono praticamente la stessa cosa. Ecco, in questo caso il pattern da utilizzare è il seguente; cosa abbiamo? Abbiamo la mancanza di un collegamento diretto fra il servizio applicativo e quello che c'è dietro la visibility line, quindi le funzioni, in questo caso, dal livello di business. Non è che la funzione di business chiama questo, o viene servita dal servizio applicativo come nel caso di prima, ma c'è un gioco di realizes: una qui, una qui, una qui e una qui. Cosa mi rappresenta questo? Allora, innanzitutto abbiamo che una realizes abbastanza intuitiva è che il website, che a livello di business rappresenta l'interfaccia, non è altro che una realizzazione della web Graphical User Interface, quindi sto dicendo che il paziente per accedere a questo servizio, utilizza un sito web, che è il sito web applicativo vero e proprio, quindi accedo, praticamente, al mio sito web. Seconda cosa, io qui sto dicendo che la funzione applicativa implementa la funzione di business, quindi tutti i passi, che sono tutte le funzionalità, che sono offerte attraverso il servizio al mio utente finale, sono la realizzazione delle funzioni di business. E quello che viene visto dall'utente finale è esattamente quello che viene offerto dal livello applicativo. Quindi, in questo caso, non c'è una specie di intermediario fra la funzione di business e il servizio applicativo, ma il mio paziente vede direttamente quelle che sono le funzionalità messe a disposizione. È ovvio, non è che vede proprio le funzionalità, ma vede il servizio applicativo, che è la realizzazione del servizio di business. Quindi, giocando su queste realization, noi riusciamo a raccontare, in questo modo, il fatto che abbiamo un processo che è messo a disposizione attraverso un sito web e quindi può essere utilizzato come un servizio self server.