Lezione 4 modulo 2

In questo modulo, andiamo a vedere alcuni concetti avanzati di BPMN e vedremo che questa è una notazione piuttosto ricca e soprattutto vorrei sottolineare che non vedremo tutti i possibili simboli di BPMN, ma quelli che ci consentono di descrivere dei flussi, dei processi e, in questo modulo, ci concentreremo sull'approfondimento di alcuni concetti che abbiamo già visto e poi alla fine vedremo un esempio complessivo. Cominciamo con il concetto di evento. Noi abbiamo già visto gli eventi e abbiamo già visto che abbiamo tipologie diverse di eventi, degli eventi di inizio, degli eventi intermedi, col doppio bordo, e degli eventi di fine, col bordo più spesso. Abbiamo già visto gli eventi che abbiamo chiamato di tipo generico. Abbiamo già visto gli eventi con i messaggi e abbiamo già visto i timer. Ovviamente all'interno di un processo potrebbero avvenire varie condizioni di tipo eccezionale che è necessario rappresentare. Succede qualcosa per cui, ad esempio, voglio interrompere il processo, oppure voglio segnalare, e per questo BPMN offre una varietà di possibilità. Vediamo alcune possibilità che poi potremo utilizzare in seguito. Innanzitutto, in alcuni casi si vuole dichiarare che in effetti all'interno del processo c'è un errore. Il simbolo di errore è questo: una specie di fulmine per indicare che c'è un problema e questo ci consente di utilizzare questo tipo di eventi per iniziare un processo di gestione dell'errore, nel caso in cui andiamo a prendere un evento di inizio con il simbolo di errore, possiamo concludere un processo indicando che c'è stato un errore, oppure usarlo come evento intermedio, soprattutto questo lo faremo utilizzato, come abbiamo visto per i timer, insieme alle attività, quindi per uscire in caso di errore da un'attività che si sta svolgendo. Un altro tipo di evento abbastanza simile ma con un comportamento leggermente diverso è quello che viene chiamato Signal. Questo evento può essere utilizzato in due modalità: una di invio e una di ricezione. Come abbiamo visto per i messaggi, nel caso del messaggio che avevamo annerito, inviavamo un messaggio, eventi di tipo throw, oppure possiamo ricevere un messaggio ed è un evento di tipo catch. Anche signal può essere di tipo throw o di tipo catch, usando sempre l'evento annerito per distinguere fra i due casi. Il signal è un evento particolare perché viene ricevuto da tutto il processo, quindi è una segnalazione che viene mandata a tutti i rami del processo che sono in esecuzione e quindi viene sentita sostanzialmente da tutto il processo. Mentre invece, abbiamo visto ad esempio per il caso dei messaggi, che noi ci aspettiamo di inviare un messaggio che viene ricevuto, quindi avevamo sempre indicato che ci fosse un flusso, invece, nel caso dell'evento di tipo signal, sostanzialmente, noi lanceremo un messaggio con un evento di tipo throw e questo verrà ricevuto all'interno del processo in una data posizione e questa sarà dipendente dalla logica del processo. Vedremo questo in qualche esempio successivo. Un altro evento che si può verificare è un evento di tipo condizione. Usiamo questi simboli per indicare la condizione e lo possiamo trovare all'inizio, quindi la condizione che fa iniziare un processo, oppure la possiamo trovare all'interno di un evento intermedio che verrà utilizzata secondo la logica di utilizzo degli eventi intermedi che abbiamo già visto. L'ultimo evento che non abbiamo ancora incontrato è l'evento di terminazione, chiamato anche end. In questo caso, questi sono tutti eventi, che ovviamente abbiamo visto in questa colonna, sono tutti eventi che abbiamo chiamato di terminazione, ma questa è una terminazione, diciamo, forzata del processo. Termina tutto il processo qualunque attività si stia svolgendo all'interno del processo, perché vi ricordo che un processo può andare avanti in parallelo su più rami, ad esempio posso andare a svolgere l'attività A e l'attività B, e poi abbiamo detto che possiamo far terminare il processo. Qual è il significato associato a questa rappresentazione? Vuol dire che avvierò in parallelo i due rami, eseguirò in parallelo, sempre parallelo concettuale, entrambe le attività A e B verranno svolte e poi dopo A termino questo ramo e dopo B termino questo ramo. Supponiamo che termini prima A, questo non vuol dire che termino anche B, ma mi aspetterò di terminare questo processo quando anche B viene terminato, quindi raggiungo un termine alla fine di questo ramo. Quindi, in generale, in un processo noi potremmo anche avere tanti rami che sono in esecuzione in parallelo e quindi che termineranno poi indipendentemente uno dall'altro. Quando utilizzo il simbolo che abbiamo appena visto di terminazione forzata, se ad esempio lo inseriamo in un processo quasi uguale a quello precedente, e metto ad esempio la terminazione forzata qui, questo vorrà dire che, se A viene terminata, tutto il processo verrà terminato e quindi sostanzialmente

questo vorrà dire che non verrà eseguito tutto quello che ci si aspettava di eseguire in questo ramo. Ovviamente in un processo questo vorrà dire che noi non ci aspetteremo più messaggi, eventi, eccetera che arrivino da questo processo, quindi sostanzialmente il processo viene terminato. Ovviamente, invece, se in questo caso passo da B e quindi termina B e A è ancora in esecuzione, il processo rimane in esecuzione finché non arrivò alla fine di A come prima. Viste queste caratteristiche, diciamo, di gestione dei processi con diversi tipi di eventi, vediamo un po' come possono essere utilizzati questi eventi all'interno del processo. Vediamo adesso qual è l'interpretazione da dare per quanto riguarda gli eventi e la terminazione all'interno di sotto processi. Vediamo qua un esempio di sotto processo inserito all'interno di un processo. Questo sotto processo ci manda in esecuzione in parallelo A e B. Vediamo che dopo A mandiamo un messaggio e dopo B terminiamo il processo. Qual è l'interpretazione da dare a questa notazione? Innanzitutto rispetto alla terminazione. La terminazione avviene all'interno di quello che è il suo ambito di processo, quindi terminiamo non tutto il processo esterno ma solo il sotto processo in cui abbiamo la terminazione. Se arriviamo alla terminazione e stiamo eseguendo A, il messaggio non verrà inviato, perché verrà terminato anche questo ramo del sotto processo che è A, seguito da un messaggio, all'invio di un messaggio. Se invece abbiamo in esecuzione prima A, l'invio del messaggio e poi in parallelo abbiamo B e la terminazione, avremo anche l'invio del messaggio. Quindi l'ambito di esecuzione di una terminazione, abbiamo detto, termina tutti i rami del processo, ma tutti i rami di quel processo e non di processi, diciamo, all'interno del quale questo può essere convenuto come, ad esempio, un sotto processo. Lo stesso varrà anche per quanto riguarda diverse pool. Abbiamo detto che i processi all'interno delle pool sono processi separati e quindi le terminazioni riguarderanno l'ambito di quel processo, quindi di una singola pool, e non di tutte le pool che potrebbero essere collegate dall'invio di messaggi. Un altro uso degli eventi che abbiamo appena visto è quello di associarli alle attività. Ad esempio, una cosa che vediamo qua nella prima figura, è il fatto di dire: associo un evento di errore, un'attività, se si verifica un certo errore all'interno di A, noi andremo a eseguire B, altrimenti A terminerà regolarmente e proseguiremo con il messaggio in modo regolare. Cosa vuol dire avere un errore all'interno di A? Una delle possibilità è che noi non vediamo all'interno di A, nel senso che la consideriamo una attività di tipo elementare, ma sappiamo che può sollevare degli errori e quindi andremo a percorrere questo percorso. Un'altra possibilità è utilizzare, come sappiamo, i sotto processi. In questo caso, A viene descritto in un processo a parte, come in questo caso, in cui abbiamo un'attività C seguita normalmente da un'attività D, però vediamo anche che si può applicare un errore dopo l'esecuzione di C, quindi nel caso in cui ci sia un errore, noi andiamo, in questo caso, a generare, quindi, un evento di tipo throw, un errore che è un errore che sarà visibile all'esterno di questo sotto processo. Quindi, sostanzialmente, questo verrà catturato nel bordo di questa attività, quindi questo errore inviato all'interno del sotto processo di A, verrà segnalato al processo che comprende A e verrà utilizzato all'interno del processo che comprende A e B. Quindi vediamo, in questo caso, un caso in cui un messaggio generato viene utilizzato all'interno del processo ma ad un livello superiore di descrizione del processo. Vediamo adesso un altro aspetto che noi andiamo a voler considerare. Abbiamo detto che abbiamo degli eventi che succedono in un processo. Normalmente i nostri eventi, abbiamo visto, li abbiamo, o messi all'interno di un flusso, ad esempio il timer intermedio che mi dice che, ad esempio, devo aspettare un'ora tra A e B oppure abbiamo associato questi eventi, mettiamo sempre un timer, per dire ad esempio che c'è un timeout quindi, se è passata più di un'ora, prenderò questo percorso anziché avere il percorso regolare che seguirei dopo aver eseguito A. In questo caso sostanzialmente abbiamo utilizzato gli eventi all'interno dei flussi che avevamo definito per specificare meglio una durata oppure per specificare un tipo di eccezione. Ma un caso che si verifica spesso è che io, in un certo momento di un processo, possa essere in attesa dell'avvenimento, dell'accadimento, di uno o più eventi e quindi io possa prendere dei percorsi diversi a seconda dell'evento che si va a verificare. Facciamo un esempio. Un caso che si può verificare spesso nei processi è il seguente: quello in cui, ad esempio, mando una richiesta, farò tipicamente con un invio di messaggi e poi mi aspetto una risposta e un caso che è abbastanza frequente è che la risposta possa essere positiva, ad esempio, mando una richiesta con una proposta e la posta è accettata, quindi un messaggio di accettazione oppure una risposta negativa quindi un messaggio di rifiuto.

Ovviamente io voglio arrivare a questi due percorsi che saranno percorsi alternativi e ci vorrò arrivare con gateway. Il problema è che nei gateway che abbiamo visto finora abbiamo la possibilità di dare, ad esempio, delle condizioni per fare le alternative, ma qua non abbiamo condizioni da associare a questi due percorsi, perché la vera condizione è l'arrivo di un certo messaggio e quindi mettere uno XOR qui non sarebbe corretto perché, nel momento in cui arrivo allo XOR, qui devo poter testare la condizione, e questa condizione non è ancora disponibile. Per rendere dei flussi di questo tipo, BPMN propone un altro tipo di gateway che è l'Event-Based gateway. Questo ha una notazione con un doppio cerchio come elemento intermedio e all'interno un piccolo pentagono. Questo simbolo indica che io mi aspetto a valle del gateway l'arrivo di eventi, che in questo caso sono messaggi ma possono essere eventi anche di altro tipo, e prenderò il primo percorso per cui mi arriva l'evento, quindi, se mi arriva l'accettazione, io andrò avanti di qui, se mi arriva un rifiuto, passerò invece da questo secondo percorso. Vediamo un altro tipico esempio di questo tipo di gateway basato su eventi. Qua vediamo già disegnato il gateway e vediamo che, ad esempio, sto aspettando una conferma, quindi il percorso normale sarà un percorso in cui mi arriva questa conferma e vado avanti. Vediamo però che in questo caso possiamo anche inserire un altro tipo di alternativa, di passo alternativo, che è legato al tempo. Cominciamo a vedere questa situazione. Quando metto un timer a 30 minuti, vorrà dire che, se non ne arriva la conferma entro 30 minuti, farò quest'altro percorso. Quindi una situazione che troveremo spesso, cancelliamo per un momento questo ramo del percorso, sarà: mi aspetto una conferma, se mi arriva, bene, vado avanti col processo da questa parte, se non mi arriva scatta il timeout e proseguirò il processo da quest'altra parte. In questo esempio, vediamo anche che c'è un terzo tipo di evento. Questi eventi possono essere combinati secondo tipologie diverse nell'event gateway, ad esempio, mi può arrivare un segnale. Se come prima cosa, ricordo che noi prenderemo il primo percorso che si verifica, quindi il primo evento che arriva, mi arriva un segnale che è un report, ad esempio dal mercato, invece di aspettarmi la conferma, con il suo eventuale time out che avevamo qui, andrò a seguire questo terzo percorso. Ovviamente, se io avevo già preso questo percorso della conferma una volta che è arrivata, l'arrivo del report del mercato non verrà più considerato all'interno di questo processo. Quindi vedete che con questi tipi di gateway, noi possiamo cogliere l'arrivo di eventi di tipo diverso che possono essere di tipo temporale, ricezioni messaggi o varie tipologie di eventi eccezionali, quali possono essere condizioni, segnali, oppure errori. Per concludere, vediamo come questo può essere utilizzato all'interno di un processo che, ad esempio, vede coinvolti un cliente e la banca. Vediamo che questo processo inizia quando si verifica una particolare condizione. Ad esempio qua abbiamo un processo che riguarda una carta di credito di un cliente e vogliamo che la banca avvisi cliente se c'è stato, nell'ambito di tre giorni, un insieme di spese superiore a un limite prefissato, ad esempio qua di 1500 euro. Questa è la condizione che attiva questo processo, quindi, se abbiamo questa condizione, -1.500 euro in tre giorni, attiviamo il processo. In questo caso cosa fa questo processo? Invierà al cliente un SMS, segnalando che è stato rilevato un movimento anomalo e quindi un pericolo di una possibile frode. Il processo come va avanti? Se il cliente effettivamente non riconosce le sue spese, potrà rispondere inviando una conferma della frode, quindi il processo si aspetta da parte del cliente un eventuale conferma della frode. Se è problematico per il cliente aver fatto, registrato questa spesa, noi non ci aspetteremmo nessuna azione da parte del cliente. Come viene rappresentato questo? Se arriva una conferma, andremo avanti per questa strada e quindi bloccheremo la carta di credito; nel caso non dovesse arrivare la conferma, noi vogliamo concludere questo processo, diamo un timeout, supponiamo che siano tre giorni il timeout, se entro tre giorni non arriva questa conferma, concluderemo il processo invece per quest'altra strada. Ovviamente di nuovo abbiamo un'alternativa fra un evento, che è la ricezione un messaggio, e un evento, che è un time out, e questa alternativa è rappresentata con un gateway di tipo event-based. Questo ovviamente non è l'unico modo per rappresentare un processo di questo tipo. Ci sono spesso modi alternativi di rappresentare lo stesso significato. Un'altra cosa che vorrei indicare in questo diagramma è l'uso di un'altra notazione che non abbiamo visto finora e che noi, in genere, non utilizzeremo, ma può rendere più evidente quali sono le attività che inviano messaggi. Vediamo che possiamo mettere una piccola busta annerita e questo vuol dire che abbiamo caratterizzato l'attività come attività che invia messaggi. Spesso noi semplicemente ci

baseremo sul fatto che viene in effetti inviato un messaggio e quindi non utilizzeremo nei diagrammi questa notazione che possiamo considerare opzionale. Visto questo processo, abbiamo visto un modo per rappresentare appunto questa situazione, condizione di anomalia, invio di un messaggio per avvisare cliente e poi ricezione eventuale di un messaggio da parte del cliente e quindi blocco della carta di credito. Non è l'unico modo per rappresentare il processo, spesso vedremo che ci sono tanti modi alternativi per rappresentare il processo, vediamone uno. In questo caso, noi possiamo anche rendere il processo in un modo più sequenziale rispetto a quello di prima. Il nostro processo alla fine vuole dire che, se ho questa condizione di inizio, invio il messaggio, mi sto aspettando l'arrivo di un messaggio, vediamo di nuovo qua il nostro simbolino del messaggio come è per l'invio, lo possiamo indicare come simbolo associato all'attività che invia messaggi, anche per la ricezione avremo attività che possono ricevere messaggi e possiamo indicarlo come simbolo di annotazione dell'attività. Riprendendo il discorso, mandiamo il messaggio, ci aspettiamo la conferma, se arriva la conferma, noi bloccheremo la carta. In realtà, questo è il percorso normale che faremo in questo processo, se vogliamo appunto catturare le frodi. Il fatto che, se non arriva la conferma della frode da parte del cliente, invece non faremo nulla, lo possiamo anche rappresentare indicando un timeout per questa attività. Quindi anziché l'event-based gateway che abbiamo usato prima, che aspettava l'arrivo del messaggio oppure un timeout, noi ci mettiamo in attesa della conferma della frode e se non arriva questa conferma entro tre giorni, come prima termineremo il nostro processo. Queste due rappresentazioni di questo processo sono sostanzialmente equivalenti: in un caso ho messo in evidenza i due eventi che sono attesi, in questo caso ho messo in evidenza l'attività di attesa della conferma della frode che potrà ricevere il messaggio oppure scadere con un timeout e completare il processo senza eseguire il blocco della carta di credito.