Sistemi Informativi Aziendali

Appunti per il corso

Fulvio Corno Marco Torchiano

Politecnico di Torino – Dipartimento di Automatica e Informatica

Versione 2.0.0

20 ottobre 2020

INDICE

CAPITOLO 5

MODELLAZIONE DI PROCESSO

Making the simple complicated is commonplace;
making the complicated simple, awesomely simple, that's creativity
Charles Mingus

La modellazione di processo permette di comprendere e descrivere le operazioni e le azioni che vengono svolte sui dati da parte del sistema informativo, in seguito anche alle interazioni con gli utenti. Si tratta di un punto di vista non più statico ma dinamico. Dal diagramma di processo è possibile ricavare le relazioni logiche e temporali che il sistema informativo deve rispettare (talvolta dette anche *regole di business*), ad esempio che cosa deve essere fatto prima e cosa deve essere fatto dopo.

In questo corso si utilizzerà BPMN (Business Process Model and Notation): che rappresentano la sequenza di eventi che accadono nel sistema informativo dal punto di vista generale, considerando anche le azioni degli utenti che interagiscono con il sistema stesso.

5.1 Modelli di processo e workflow

In sintesi, la modellazione di processo ha come obiettivo quello di descrivere il più precisamente possibile un processo (o workflow). Attraverso un modello di processo sarà possibile comunicare, documentare, analizzare, convalidare il flusso delle operazioni.

5.1.1 Modello di processo

In sintesi un processo viene modellato attraverso i seguenti elementi:

attività che rappresentano le singole fasi attraverso cui il processo si sviluppa

regole che descrivono le interdipendente logico-temporali tra le varie azioni (esecuzione in serie, esecuzione in parallelo, ripetizione di azioni precedenti, scelta tra più percorsi alternativi, ...)

responsabilità con le quali si specifica, per ciascuna azione, chi sia la persona, l'ente, il reparto, il servizio, ... responsabile di portare a termine tale azione.

Un modello di processo racchiude in sé due diverse chiavi di lettura, complementari e coesistenti:

descrittivo Come strumento descrittivo, il diagramma ci dice come sono articolati i processi del sistema e l'organizzazione delle varie fasi di processo. In altre parole, *capire quali sono i processi in atto.*

prescrittivo Come strumento prescrittivo, il diagramma può essere utilizzato come allegato tecnico di un contratto che specifichi come deve funzionare il sistema. In altre parole, *descrivere quali processi si metteranno mettere in atto.*

Le notazioni adottate per la modellazione di processo possono essere classificate come:

- *formali*, hanno una semantica precisa e ben definita, questo permette l'eseguibilità del modello all'interno di un sistema di workflow; un esempio di notazione formale è BPMN [2];
- *semi-formali*, dove la semantica non è completamente definita o è parzialmente ambigua, in tale caso i modelli di processon non saranno immediatamente eseguibili, ma nondimeno potranno essere utili come punto di partenza per l'analisi ad alto livello; un esempio di notazione semi-formale sono gli UML *Activity Diagram*;
- *informali*, in cui i dettagli operativi sono limitati o non fanno riferimento ad una semantica ben definita, in questa categorie ricadono le descrizioni testuali, o notazioni di alto livello con limitati dettagli operativi, come IDEF-0 [1].

In generale, quando si utilizza una notazione formale è possibile omettere dettagli, quindi usarla in modo semi-formale. Questo permette di analizzare dei processi esistenti o definirne di nuovi.

5.1.2 Workflow

Quando si parla di esecuzione o automazione di un processo generalmente ci si riferisce ad un sistema di tipo *workflow*. Un sistema di gestionre di workflow (WfMS *Workflow Management System*) è in grado di utilizzare una descrizione formale di un processo e di eseguirla (*enactment*).

Data la descrizione formale di un processo, il WfMS (su richiesta di un utente) crea una istanza del processo: ovvero una specifica esecuzione di quel processo. Un'istanza di processo è distinta dalle altre istanze dello stesso processo (formale) in quanto opera su dati propri e possiede un proprio stato (ovvero a che punto del processo è giunta).

Per ogni attività prevista da un processo, il WfMS determina se e quando deve essere eseguita e la assegna ad un partecipante o la esegue se si tratta di un'attività automatizzata.

I partecipanti (utenti) di un workflow hanno quella che si chiama una *work list* ovvero un elenco di tutte le attività (per tutte le istanze di processo in cui è coinvolto) che il WfMS gli ha assegnato.

I processi modellati molto spesso coinvolgono le persone, e talvolta la progettazione di un processo ha lo scopo di semplificare e migliorare le attività degli utenti. Affinché l'ottimizzazione dei processi sia possibile, bisogna comprendere a fondo, tramite un processo di documentazione e analisi, come le persone lavorano e quali impatti la *reingegnerizzazione* di un processo possa avere sul piano organizzativo.

5.2 BPMN: elementi base

Alcune informazioni aggiuntive sui diagrammi di attività UML, oltre a quelle esposte nel seguito, si possono trovare sul sito ufficiale dell'OMG contenente tutta la documentazione su BPMN: `https://www.omg.org/spec/BPMN/2.0/`.

Figura 5.1: Costrutti principali di BPMN

Gli elementi fondamentali, illustrati in figura 5.1, utilizzati in un modello BPMN sono:

- gli *Eventi* (*Events*)
- le *attività* o *compiti* (*Activities/Tasks*)
- i *flussi* (*Flow*)
- i *Gateway* (in italiano traducibile approssimativamente con portali)

5.2.1 Attività

La notazione, per tutte le tipologie di azioni, è sempre la stessa (figura 5.2): ogni azione è rappresentata da un rettangolo con gli angoli arrotondati, contenente al proprio interno la descrizione sintetica dell'azione (ossia ciò che avviene all'esecuzione di tale azione).

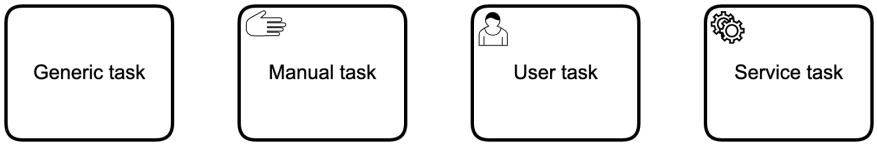

Figura 5.2: Rappresentazione di un'Attività in BMPN

Le attività, che rappresentano le singole fasi del processo, possono dividersi in:

1. azioni manuali, compiute esclusivamente da uno o più esseri umani utenti del sistema
2. azioni svolte e gestite dal sistema informativo, con l'ausilio degli utenti o attraverso l'interazione con gli utenti
3. azioni svolte dal sistema informativo in completa autonomia, senza l'intervento umano ed attraverso procedure automatiche.

Occorre prestare attenzione alle azioni del primo tipo (completamente manuali), ed in particolare occorre verificare se tale azioni *lascino traccia* nel sistema informativo. Nel caso in cui delle azioni manuali non lascino traccia nel sistema informativo, e l'utente non interagisca con il sistema informativo, allora tali azioni non devono essere modellate, in quanto il sistema informativo non avrà alcun modo per dedurre se tali azioni siano avvenute o meno, e di conseguenza passare ad eseguire le azioni successive. Vale quindi la seguente regola: una azione sarà rappresentabile in un modello di processo solamente se essa lascia traccia nel sistema informativo. Tutte le azioni che coinvolgono il sistema informativo (quelle di tipo 2. e 3.) evidentemente vi lasciano traccia, e quindi sono sempre modellabili.

Conviene fare chiarezza sul concetto di "lasciare traccia nel sistema informativo," visto che è un concetto che sarà utilizzato molto spesso nell'analisi dei processi. La conoscenza di un sistema informativo è rappresentata dalle informazioni che esso gestisce, ossia dalle informazioni rappresentate nel modello concettuale (diagramma delle classi). Pertanto, lasciare traccia nel sistema informativo significa che le informazioni presenti nel modello concettuale, dopo l'esecuzione dell'azione, saranno necessariamente diverse rispetto alle informazioni che vi erano presenti prima dell'esecuzione dell'azione. Ad esempio, l'azione avrà creato nuovi oggetti (istanze di classi), avrà aggiornato il valore di alcuni attributi, avrà aggiunto nuove occorrenze di associazioni, oppure avrà cancellato qualche oggetto od occorrenza. In assenza di una qualsiasi modifica, l'effetto dell'esecuzione dell'azione sul sistema informativo sarà nullo (potrà invece avere effetto sugli umani, o su altri sistemi informativi diversi da quello considerato, o sull'ambiente esterno, ecc.) e pertanto non deve essere presente nel modello BPMN. Un'ultima puntualizzazione: le informazioni verbali (le cose dette tra persone) e le conoscenze dei singoli utenti (i loro pensieri) **non** fanno parte del sistema informativo, in quanto tale informazione non è disponibile.

In sintesi: un modello BPMN rappresenta le azioni che lasciano traccia nel sistema informativo (inteso in senso informatico) e lo coinvolgono esplicitamente. In ogni caso, il modello concettuale è unico e condiviso tra tutti gli activity diagram.

La notazione, per tutte le tipologie di azioni, è sempre la stessa (figura 5.2): ogni azione è rappresentata da un rettangolo con gli angoli arrotondati, contenente al proprio interno la descrizione sintetica dell'azione (ossia ciò che avviene all'esecuzione di tale azione).

5.2.2 Eventi terminali

Oltre alle attività che sono "interne" al processo, la descrizione include anche eventi che rappresentano la connessione tra il processo stesso e l'esterno (tipicamente altri processi). BPMN prevede tre categorie di eventi:

- eventi di avvio: rappresentano l'evento che da avvio ad una nuova istanza di processo, nel caso più semplice corrisponde all'attore responsabile del processo che avvia una nuova istanza. Gli eventi di avvio sono rappresentati da un cerchio con il bordo sottile.
- eventi di fine: rappresentano la conclusione del processo, quando il processo raggiunge tale fase l'istanza di processo si considera conclusa. Gli eventi di fine sono rappresentati da un cerchio con il bordo spesso.
- eventi intermedi: rappresentano eventi esterni che possono alterare l'evoluzione di un processo (sia nel caso che il processo interessato sia quello in considerazione sia che sia un altro influenzato da quello in considerazione). Gli eventi intermedi sono rappresentati da un cerchio con un dobbio bordo sottile.

Figura 5.3: Categorie di eventi

5.2.3 Flusso di esecuzione

L'ordine con cui sono eseguiti gli elementi che descrivono i passi di un processo è indicato dal flusso di sequenza: delle frecce che collegano due elementi consecutivi. Graficamente il flusso di seguenza può essere un semplice segmento con una freccia che indica l'ordine oppure una spezzata con una freccia sull'ultimo tratto.

5.2.4 Gateway

I gateway rappresentano gli elementi che consentono di definire le regole con cui si succedono i passi di un processo (alternativa, concorrenza, etc.). I gateway sono rappresentati tramite dei rombi al cui interno compare un simbolo – ad esempio + X * – che indicano il tipo di regola da seguire per instradare l'esectuzione dei passi.

I due tipi più semplici sono il gateway esclusivo (X) che indica la scelta esclusiva di uno tra più percorsi alternativi, e il gateway parellelo (+) che indica la partenza di più flussi di attività indiepndenti tra di loro.

5.2.5 Semantica di esecuzione

Quando viene create un'istanza di un processo, nell'evento iniziale, possiamo immagianre che venga deposto un "token" (gettone) che si comporta come una sorta di un segnaposto (un po' come se fosse un gioco da tavolo). Il token si posterà sugli elementi che rappresentano i successivi passi del processo. Quando il token raggiunge un evento finale viene eliminato, questo corrisponde alla fine dell'istanza di processo. Il token è legato alle informazioni specifiche dell'istanza di processo, chi l'ha avviato, quali dati vengono processati, ecc.

Il token, una valta creato nell'evento di avvio segue il flusso e le regole di instradamento dei gateway, finchè giunge al nodo di fine in cui viene distrutto (figura 5.4).

Quando un token giunge in un'attività, l'attività è considerata abilitata, ovvero può essere svolta. Il sistema informativo (o workflow management system) informa l'utente che può compiere l'azione: ovvero viene aggiunto un elemento nella worklist dell'utente.

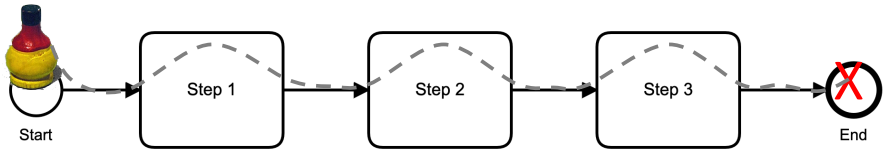

Figura 5.4: Esempio di processo e token.

L'abilitazione di un'attività non indica quando l'attività verrà iniziata o quando verrà completata. In pratica una volta che è abilitata l'attività verrà presa in considerazione dall'utente incaricato quando vorrà o potrà, ad esempio compatibilmente con i propri impegni. Anche la durata dell'attività non è nota: dipende dall'utente.

Quando l'utente dichiara completata l'azione, il token lascia l'azione e prosegue il suo percorso seguendo il flusso di sequenza in uscita dall'attività stessa.

Nel caso di attività automatiche (svolte autonomamente dal sistema informativo) l'attività verrà svolta dal SI non appena possibile. Ma anche in questo caso non è garantito ne un avvio ne un completamento immediato, anche se è probabile.

Ogni activity diagram associa un significato ai token che lo attraversano. Token di natura diversa devono viaggiare in diagrammi diversi. Per questo motivo, per ogni activity diagram, conviene apporre un'annotazione (indicativamente a fianco del nodo iniziale), che ricordi che cosa rappresenta il token di quel processo. Ovviamente processi diversi saranno attraversati da token con significato diverso.

5.3 BPMN: costrutti base

5.3.1 Sequenza

Il concetto di sequenza rappresenta il vincolo per cui un'attività può iniziare solo se un'altra precedente è già terminata. La sequenza costituisce lo strumento base per rappresentare l'ordinamento *logico* e *temporale* delle azioni. Temporale in quanto le azioni seguenti nella sequenza inizieranno certamente ad un istante di tempo successivo alle azioni che le precedono; logico, perché una sequenza implica che l'azione successiva non può iniziare, per qualche motivo legato a ciò che le azioni precedenti devono avere svolto (es. raccolto dei dati, effettuato delle elaborazioni).

La sequenza viene rappresentata, come illustrato in figura 5.5, attraverso un flusso di sequenza, ovvero freccia che collega le due attività da compiere in sequenza.

Facendo riferimento alla figura, il token viene creato nel nodo iniziale. Appena creato, immediatamente si sposta all'ingresso della prima azione. Il token potrà rimanere in attesa per tempo indefinito (perché l'utente non è pronto, o il sistema informativo è impegnato in altre cose), finché l'*Receive Order* non verrà iniziata; il fatto che il token abbia raggiunto l'attività significa che sono disponibili tutti i dati e che sono soddisfatte tutte le condizioni richieste, per poterla svolgere.

Al completamento dell'azione, il token viene "rilasciato" in uscita, e segue il percorso indicato dal flusso di sequenza. Ad esempio, appena terminata *Receive Order*, immediatamente il token si porta su *Send account information*, ossia abilita all'esecuzione questa attività.

Riassumendo, la semantica della connessione in sequenza, si può riassumere come: "L'attività *Send account information* viene abilitata all'esecuzione solamente dopo che l'attività *Receive Order* sarà completata."

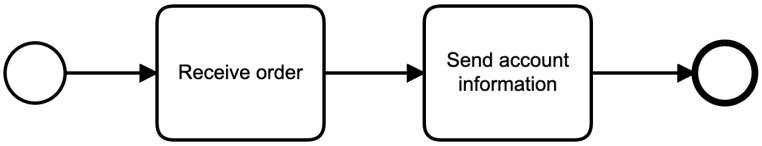

Figura 5.5: Sequenza

5.3.2 Parallelo

L'esecuzione parallela, detta anche *parallel split* oppure *fork* (figura 5.6), rappresenta una condizione in cui alcune azioni sono indipendenti tra loro. Nell'esempio non è rilevante se venga eseguita prima l'attività *Send account information* o *Fulfill order*: possono essere svolte insieme, una dopo l'altra o iniziare e finire in momenti diversi. L'aspetto chiave è che vengano entrambe eseguite dopo l'attività che precede il gateway, *Receive order*.

La struttura in parallelo viene rappresentata attraverso un gateway parallelo (rombo con +), che rappresenta il punto dove il flusso si moltiplica in diverse sequenze indipendenti. Il fork ha una sola freccia entrante, che riceve il token dalla parte precedente del processo ed ha due o più frecce uscenti in uscita.

La semantica di esecuzione del fork prevece che su ciascuno dei flussi uscenti sia inviato un "clone" del token in ingresso. Il passaggio del token attraverso il gateway è da intendersi come azione istantanea (non introduce ridardi e non "ferma" il token).

Il costrutto del parallel split specifica esplicitamente l'assenza di una qualunque ipotesi sull'ordinamento delle azioni, esso può essere usato quando non esiste alcun tipo di vincolo che richieda un ordinamento nell'esecuzione. I rami paralleli sono indipendenti, deve essere possibile portare a termine le azioni in ciascun ramo senza sapere se le altre azioni siano già iniziate o concluse. Ciascuna azione parallela ha a disposizione i dati su cui deve operare già dal momento in cui il token raggiunge il nodo fork, e di norma modificherà delle informazioni disgiunte da quelle modificate dalle altre azioni (altrimenti si potrebbero creare dei conflitti o delle situazioni di non determinismo, in cui il risultato finale potrebbe dipendere dall'ordine di esecuzione).

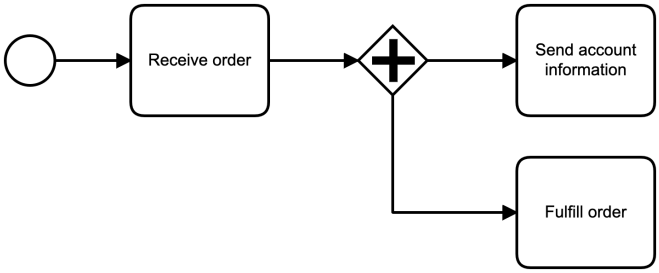

Figura 5.6: Esecuzione parallela (Fork)

Poiché il nodo fork crea più flussi di esecuzione indipendenti, emerge spesso la necessità di "riunire" i vari flussi, in modo da continuare l'esecuzione dell'attività solamente quando tutte le attività indipendenti sono definitivamente concluse. A questo scopo si può utilizzare il costrutto del *punto di sincronizzazione* (detto anche *join*), il quale (figura 5.7) riceve in ingresso più flussi e presenta in uscita una solo flusso.

Il *join* è rappresentato con un gateway graficamente identico al *fork*.

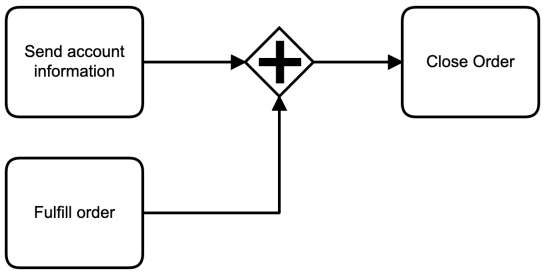

Figura 5.7: Punto di sincronizzazione (Join)

I diversi cloni del token che hanno attraversato le sequenze parallele si riuniscono nel punto di sincronizzazione, quando un token per ogni flusso entrante ha raggiunto il gateway, viene ricostruito

in uscita il token iniziale. Qualora uno dei rami termini prima di un'altro, il relativo token rimarrà "in attesa" all'ingresso del nodo join, finché anche tutti gli altri token non saranno giunti; a questo punto, i token in ingresso vengono fusi tra loro, ed immediatamente viene liberato il token in uscita.

I costrutti di join e fork molto spesso vengono utilizzati in combinazione. Normalmente ad ogni fork corrisponde un join che ne raccolga (tutti e soli) i flussi paralleli. Nella sezione **??** si vedranno alcune eccezioni a questa regola generale, utilizzate per modellare situazioni particolari.

Esempio La figura 5.8 riporta un esempio completo di applicazione dell'esecuzione parallela. La prima attività svolta dopo l'avvio del processo è quella di *Receive order*. Successivamente potranno essere svolte le attività *Send account information* e *Fulfill order*, in maniera indipendente, senza un ordine prestabilito tra loro. L'attività finale di *Close order* potrà essere svolta solo dopo che entrambe le attività precedenti sono state completate.

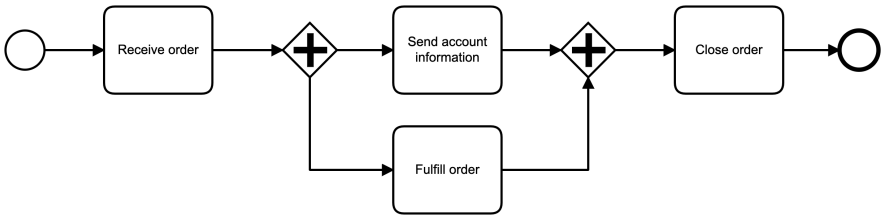

Figura 5.8: Esempio di esecuzione parallela

5.3.3 Scelta esclusiva

La scelta esclusiva (detta anche Choice, o If) permette di decidere fra due (o più) attività in una situazione dove è possibile eseguirne una in alternativa alle altre. In tal caso si modella l'esistenza di diversi cammini che il token può seguire instradando il processo in una direzione piuttosto che in un'altra.

La rappresentazione grafica della scelta esclusiva (figura 5.9) utilizza un gateway con una X all'interno (corrispondente al connettore logico XOR). Nel gateway di scelta c'è un solo flusso entrante e due o più flussi uscenti. Ogni flusso uscente è caratterizzato da un'etichetta che corrisponde alla condizione per cui la scelta ricadrà su quel percorso.

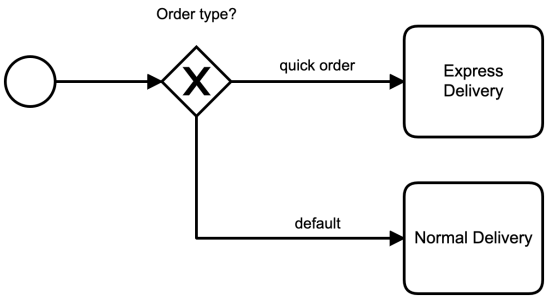

Figura 5.9: Punto di scelta (Choice)

Quando il token giunge in un gateway esclusivo, il sistema valuta le condizioni associate ai diversi flussi uscenti e seleziona quello per cui risulta soddisfatta. La valutazione delle condizioni può essere considerata istantanea, perciò il token, non appena giunge al gateway viene immediatamente smistato sul flusso corretto. Le condizioni, dette anche *guardie*, associate ai diversi flussi alternativi devono essere mutuamente esclusive ed esaustive (almeno una delle condizioni deve essere vera); questo garantisce che sia sempre possibile scegliere uno ed un solo flusso uscente. Per garantire che le condizioni siano esaustive è possibile usare la condizione *default* che viene convenzionalmente considerata vera se tutte le altre sono false. Inoltre deve essere possibile valutare le condizioni sulla

base delle informazioni presenti nel SI e possibilmente descritte nel modello informativo. La scelta esclusiva, da sola, può rappresentare la scelta di un utente per un'alternativa: quando la scelta dipende da una decisione di un utente, è necessario inserire nel processo un'attività, precedente la scelta, in cui il SI acquisisce la decisione, tale informazione può poi essere usata per la scelta.

Molto spesso, una volta terminata l'esecuzione su uno dei rami alternativi, il processo deve proseguire in modo comune, indipendentemente dalla scelta fatta: è necessario quindi un costrutto che permetta di "ricongiungere" i diversi flussi di esecuzione, e proseguire con un flusso unico. A questo scopo esiste un gateway di *merge* (o ricongiungimento), la cui rappresentazione è ancora graficamente identica al gateway di scelta esclusiva (figura 5.10) in cui però convergono due o più frecce in ingresso, mentre vi è una sola freccia in uscita. Il nodo di ricongiungimento non compie alcuna elaborazione né scelta, si limita a fare proseuire lungo l'unico flusso in uscita il token, non appena esso si arriva da uno qualsiasi dei flussi in ingresso. Il gateway di *merge* non comporta alcun ritardo di esecuzione né arresta il token. Non vi sono condizioni di guardia associate al nodo merge.

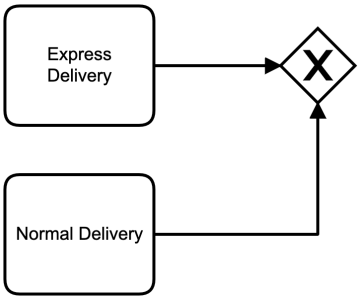

Figura 5.10: Punto di ricongiungimento (Merge)

È pratica comune quella di utilizzare la scelta ed il ricongiungimento in coppia. In generale, si tende a avere dei processi ben strutturati, in cui a ciascuna scelta corrisponde un ricongiungimento, in modo da ottenere processi in cui ciascuna porzione possegga un solo punto di ingresso ed un solo punto di uscita.

Esempio L'esempio riportato in figura 5.11 illustra un possibile utilizzo del costrutto di scelta e ricongiungimento. In tale esempio, dopo che l'ordine è stato preparato, vi sono due modalità di spedizione possibili: per corriere espresso oppure per posta ordinaria. Il nodo si scelta discrimia tra queste due alternative valutando le espressioni di guardia "Spedizione rapida" e "Spedizione normale": tali espressioni devono essere calcolabili facendo riferimento ad informazioni che siano già presenti nel modello concettuale (ad esempio, sono state acquisite durante l'azione di ricezione dell'ordine, e la classe che rappresenta l'ordine conterrà un attributo di tipo booleano che ricorda tale scelta).

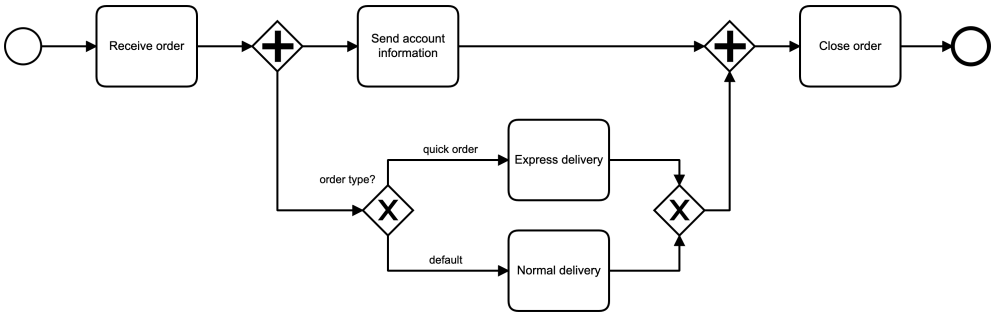

Figura 5.11: Esempio di scelta esclusiva e parallelismo

5.3.4 Pool e Swimlane

Quando si descrive il modello di un processo che verrà attuato tramite delle attività svolte da attori umani, è fondamentale che a ciascuna attività sia attribuita chiaramente la responsabilità della persona, gruppo o unità organizzativa che dovrà compierla (ad esempio, interagendo con in sistema per portarla a termine).

I partecipanti di alto livello (*business entities*) sono modellati da *Pool*, che racchiudono tutte le azioni che competono a loro. In genere un pool corrisponde ad una organizzazione (o talvolta parti distinte di una organizzazione). Graficamente un pool è rappresentato da un rettangolo con un comparto sulla sinistra che include il nome del partecipante. Normalmente la dimenzione orizzontale è più lunga di quella verticale.

All'interno di un partecipante o organizzazione è normale avere più ruoli coinvolte che possono corrispondere a individui, gruppi di lavoro, unità funzionali o unità organizzativa che avranno la responsabilità di compiere determinate attività. Questi ruoli sono rappresentati tramite delle corsie orizzontali (*swimlane*) all'interno di un pool e riportano all'estremità sinistra il nome del ruolo rappresentato.

Le Swimlane sono quindi un costrutto grafico per rappresentare le responsabilità delle varie azioni. Il disegno viene diviso in corsie e ogni corsia viene associata ad un attore.

Ad esempio, si consideri il semplice processo di figura 5.11. Lo stesso diagramma, se si attribuiscono le responsabilità, sarà composto da due partizioni, una per il reparto vendite ed una per quello spedizioni (figura 5.12).

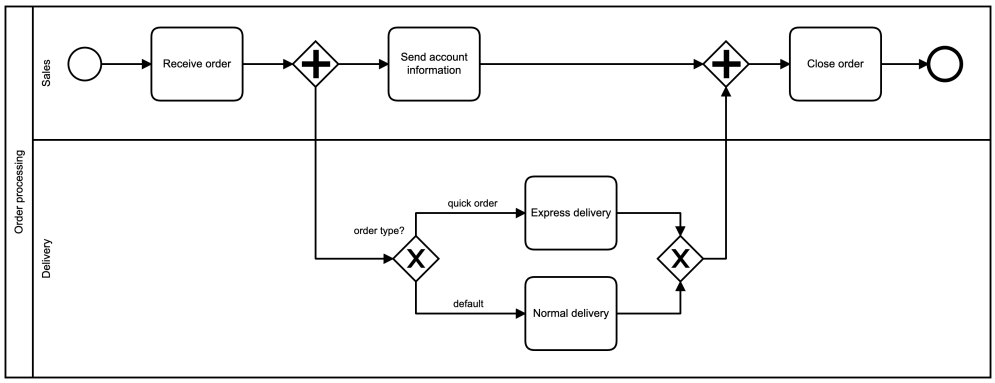

Figura 5.12: Esempio di pool e swimlane

Si può notare come la notazione delle swimlane permetta anche di identificare, in modo visualmente immediato i passaggi di responsabilità tra diversi attori, ed in in certo modo le comunicazioni tra tali attori, semplicemente identificando le frecce che attraversano il confine tra due corsie.

È bene notare che il concetto di partizionamento si applica *esclusivamente alle azioni*, in quanto sono l'unico costrutto a cui si attribuisce una responsabilità. Tutti i restanti nodi di controllo del flusso (eventi iniziale e finale e gateway) non appartengono ad alcuna partizione. Dal punto di vista grafico, potranno essere disegnati in una qualsiasi delle partizioni, facendosi guidare esclusivamente da criteri di tipo estetico e di chiarezza del disegno. Ad esempio, solitamente il nodo iniziale viene posto nella stessa corsia della prima azione del processo, anche se ciò non ha alcun significato particolare.

5.3.5 Commenti e connettori

Oltre agli elementi che hanno una semantica di esecuzione ben definita è utile poter aggiungere ad un processo altri elementi che ne facilitano la lettura e la comprensione. In particolare è pratica comune avere gli elementi mostrati in figura 5.13:

- **annotazioni**: che permettono di inserire dei commenti legati a particolari elementi del processo. Graficamente le annotazioni sono rappresentata con una specie di parentesi quadra aperta, seguita dal testo, eventualmente spezzato su più righe.

- **connettori**: permettono di spezzare il flusso del processo, ad esempio se lo sviluppo orizzontale è eccessivo è possibile "andare a capo". Graficamente un link si compone di un evento intermedio di invio (freccia piena) ed uno di ricezione (freccia vuota). Il token che attraversa il processo viene inviato dal primo e ricevuto dal secondo senza nessuna alterazione o ritardo: è come se ci vosse un flusso tra i due che però viene omesso per ragioni di disposizione grafica e di semplificazione del diagramma. I due link corrispondenti sono identificati dallo stesso nome.

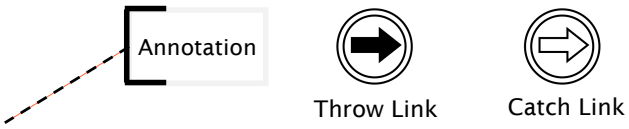

Figura 5.13: Annotazioni e link

5.4 BPMN: costrutti complessi

Oltre agli elementi fondamentali presentati nella sezione precedente, spesso è necessario poter rappresentare condizione più complesse che spesso non non sono intuitive e richiedono delle strutture più complesse. Molti di questi pattern sono tratti, con notevoli semplificazioni, da [3].

5.4.1 Processi strutturati

Una delle principali regole di modellazione è quella di mirare, il più possibile, a realizzare diagrammi di attività di tipo *strutturato*. Questo è un requisito comune a altri contesti (ad esempio si applica ai flow chart, ai linguaggi di programmazione, alle pagine web, ecc.): l'obiettivo è mantenere la complessità di un artefatto (modello, programma, diagramma, ...) ad un livello facilmente comprensibile dai lettori umani. Un aspetto che migliora la comprensibilità di un processo è la possibilità di identificare dei blocchi che possano essere analizzati e compresi separatamente. Questo è possibile se si hanno dei processi *strutturati*. In particolare, ciò permette di selezionare una determinata porzione di un processo, e sostituirla con una singola azione complessa (v. sezione 5.4.5), in modo da poter ignorare temporaneamente la sua implementazione.

Un processo si dice strutturato se:

- ogni nodo azione (azione semplice o azione complessa) ha esattamente una freccia entrante ed esattamente una freccia uscente;
- esiste un solo nodo di inizio, con una sola freccia uscente;
- esiste un solo nodo di terminazione, con una sola freccia entrante;
- ogni nodo fork ha un nodo join corrispondente, in cui i flussi di esecuzione che si sono separati vengono tutti riuniti;
- ogni nodo di scelta ha un nodo merge corrispondente, che ricongiunge i flussi di elaborazione alternativi;
- quando vi siano più costrutti di tipo fork-join o choice-merge, ciascun costrutto è completamente annidato all'interno di uno dei rami di uno degli altri costrutti, o viceversa lo comprende completamente. In altre parole, non è possibile "entrare a metà" o "uscire prima" da uno di questi costrutti: si può solamente entrare dal nodo iniziare ed uscire da quello finale.

Nella modellazione, ci si atterrà il più possibile all'utilizzo di processi strutturati. Qualora, per esigenze particolare (si vedranno alcuni esempi tra breve) fosse necessario fare delle eccezioni, avremo cura di inserire la parte di processo non strutturata all'interno di un'azione complessa, in modo da limitare la complessità ad una porzione ben documentabile del processo, isolata dal resto.

5.4.2 Ciclo strutturato

Esattamente come nei linguaggi di programmazione, anche nei modelli di processo è possibile che si debbano ripetere una serie di attività. Si tratta di un ciclo (*loop*) che, come accennato sopra, dovrebbe essere preferibilmente strutturato. Sono possibili due principali tipologie di cicli strutturati

- i cicli del tipo `do-while`, che verificano la condizione di permanenza nel ciclo alla fine, e quindi sono eseguiti almeno una volta. Ad esempio

```
do {
  read_item();
  pick_item();
} while( more_items );
```

- i cicli del tipo `while`, che verificano la condizione all'inizio del ciclo, e quindi potrebbero non essere mai eseguiti. Ad esempio:

```
while( more_items ){
  read_item();
  pick_item();
}
```

I cicli del primo tipo (`do-while`) possono essere creati mediante l'uso di un nodo di scelta e di un nodo di merge, nel quale il nodo di scelta venga incontrato *dopo* quello di merge, ed almeno una delle uscite riporti indietro al nodo di merge stesso.

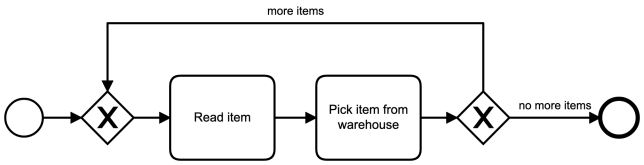

Figura 5.14: Esempio ciclo "Do-Repeat" strutturato

L'esempio di figura 5.14 rappresenta un ciclo di tipo "Do-Repeat" o "Do-While" (dal nome dei costrutti di alcuni linguaggi di programmazione), nel quale la condizione di ripetizione (il nodo choice) viene verificata al termine dell'iterazione (prima eseguo le azioni, poi mi chiedo se devo ripeterle). In tale esempio il token rappresenta un generico ordine che è composto da più elementi. Ciascun elemento viene selezionato e prelevato dal magazziono, e queste due operazioni vengono ripetute finché vi sono ancora elementi. In tal caso, i due blocchi centrali (seleziona e preleva) vengono ripetuti più volte, finché una condizione (nessun elemento rimanente) diventa vera. L'intero processo è strutturato, un quanto il ciclo, nel suo complesso, è dotato di un solo punto di ingresso ed un solo punto di uscita, ed i nodi choice e merge sono correttamente accoppiati. Ovviamente, come in tutti i nodi di tipo choice, anche qui le informazioni in base alle quali il processo decide iterare od uscire dall'iterazione sono definite dall'espressione di guardia, che dovrà essere calcolabile sulla base delle informazioni presenti nel modello concettuale.

I cicli del secondo tipo (`while`) richiedono che un solo gateway che svolge contemporaneamente il ruolo di scelta e di merge.

Un ciclo di tipo "While", come in figura 5.15, prevede che il test venga svolto prima dell'esecuzione delle azioni.

5.4.3 Terminazione implicita ed esplicita

Come visto nella sezione 5.2.5, l'evento di fine elimina il token e termina il processo. Tuttaviva è possibile che esistano in un processo più nodi di fine in rami diversi del processo. È importante poter determinare se la fine di un ramo implica la fine di tutto il processo o soltanto del ramo in questione.

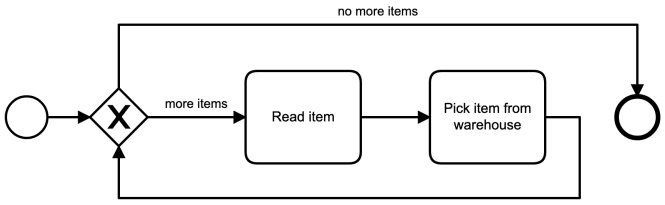

Figura 5.15: Esempio ciclo “While” strutturato

In BPMN, un processo può terminare per:

Terminazione implicita il processo termina quando tutte le azioni previste sono completate, per cui ciascun flusso parallelo di elaborazione deve giungere alla propria condizione di terminazione (figura 5.16). Questo si può rappresentare utilizzando un nodo Flow Final al termine di ciascun flusso parallelo, in modo che quando tutti i token sono arrivati a destinazione (e solo allora) il processo può considerarsi terminato con successo. Se invece alcuni token, su alcuni flussi, sono rimasti indietro, allora il processo non si considera ancora terminato e si attende che giungano a conclusione anche questi. In questo caso la terminazione è detta implicita in quanto non esiste un punto unico di controllo in cui si evidenzia la fine delle attività, ma questa avviene implicitamente quando tutti i flussi di elaborazione (cioè i token attivi) si sono esauriti.

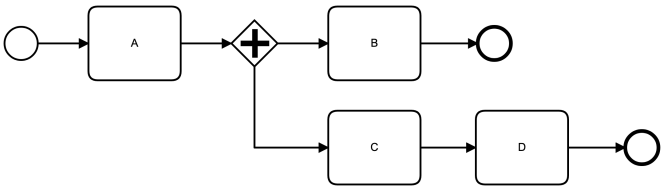

Figura 5.16: Esempio terminazione implicita.

Terminazione esplicita il processo termina quando viene raggiunto un determinato stato, solitamente identificato dall’evento di terminazione, che sarà unico per tutto il processo (figura 5.17). Quando il nodo finale viene raggiunto da un qualsiasi token, ogni eventuale attività in corso viene annullata, e l’attività nel suo complesso viene considerata come terminata con successo, anche se vi fossero ancora dei task in corso o in attesa di esecuzione. Questa è la modalità normale di terminazione, e prevede che tutti i flussi di esecuzione vengano fatti convergere (attraverso nodi join o nodi merge) sul nodo Activity Final. La terminazione è detta esplicita in quanto vi è un ben preciso nodo che identifica il termine dell’attività.

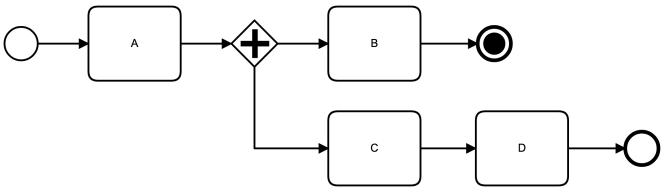

Figura 5.17: Esempio terminazione esplicita.

5.4.4 Scelta implicita

Può sorgere il problema di dover scegliere fra due attività che si possono svolgere in alternativa, ma di cui non sia noto a priori, né sia possibile acquisire informazioni per determinare quale si svolgerà. In queste situazioni non è possibile usare un gateway di scelta (che richiederebbe di conoscere

quale azione attivare prima di raggiungerla), ma occorre “abilitare” entrambe le alternative, per poi permettere di eseguire una sola delle due.

A tal fine, si può utilizzare un costrutto come quello riportato in figura 5.19: il gateway parallelo permette di fare arrivare il token ad entrambe le attività di spedizione (veloce e normale). A priori non possiamo sapere quale alternativa sarà eseguita, ma quando una delle due verrà eseguita, il token giunge al corrispondente nodo di terminazione esplicita, e di conseguenza il token viene rimosso dall’altra alternativa, che in pratica è annullata.

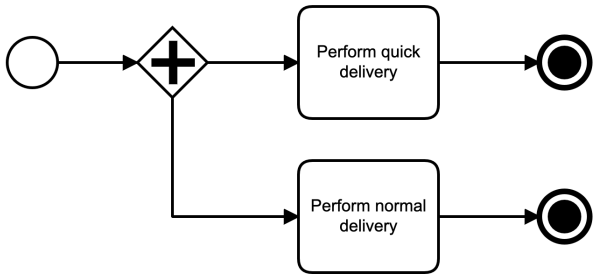

Figura 5.18: Esempio terminazione esplicita.

Figura 5.19: Scelta implicita

La scelta implicita (detta anche discriminatore struturato) richiede l’utilizzo di diagrammi non strutturati (un fork senza join, e due nodi di terminazione anziché uno).

Si noti che questo costrutto è diverso rispetto alla semplice scelta esclusiva (sezione 5.3.3, figura 5.20), in quanto la scelta viene effettuata, in tempo zero, dal sistema informativo sulla base delle informazioni disponibili. Nel discriminatore, invece, il sistema informativo non sa a priori quale alternativa verrà eseguita, e si dovrà predisporre per eseguirle entrambe. Una volta che una delle alternative verrà iniziata, allora l’altra sarà abbandonata. In un certo senso, la scelta viene fatta dall’utente che interagisce con il sistema, e non dal sistema informativo stesso.

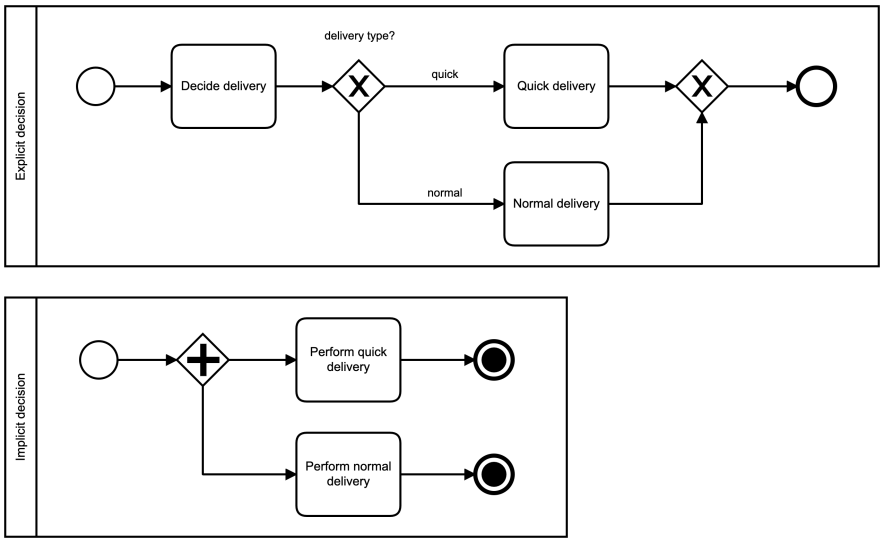

Figura 5.20: Confronto tra scelta esplicita ed implicita.

5.4.5 Azione complessa

Al crescere della complessità degli processi, risulta difficile descrivere l'intero comportamento in un unico activity diagram. A tale scopo, si possono rappresentare delle azioni complesse, in cui la descrizione del comportamento viene descritto in un altro activity diagram, subordinato al primo. Tecnicamente si parla di un an'attività di tipo *sotto-processo*, ossia la cui azione corrisponde a "richiamare" un altro (sotto) processo. Si parla anche di *azione complessa*, intendendo che l'azione rappresentata dal singolo nodo non è più semplice (o atomica), ma è più articolata.

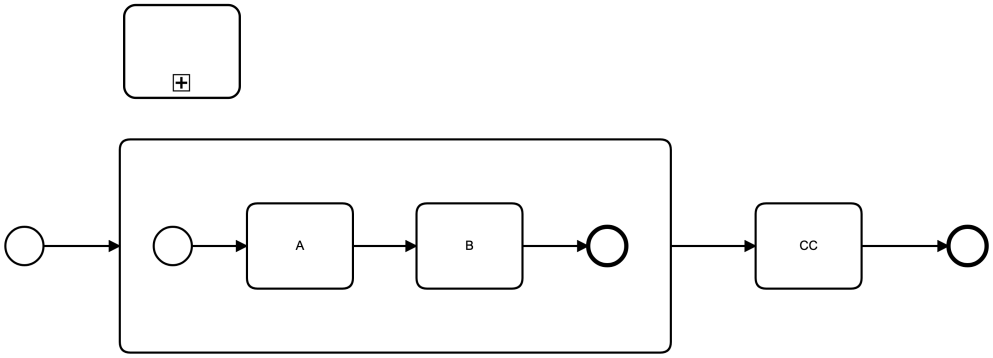

Figura 5.21: Esempio di sotto-processo (in alto) e sotto-processo espanso (in basso)

Il significato di questa notazione è analogo alla chiamata di una procedura, o funzione, in un linguaggio di programmazione. Nel momento in cui il token raggiunge un'azione complessa, viene avviato creata una nuova istanza di processo, con il relativo token. Il token del sotto-processo è del tutto indipendente dal token del processo esterno. Quando il sotto-processo termina, ovvero il token ha raggiunto il nodo terminale, l'istanza di sotto-processo viene chiusa, ed il token nel processo esterno viene sbloccato e prosegue con il flusso in uscita.

Una possibile motivazione per utilizzare una sotto processo è quella si semplificare la rappresentazione grafica, utilizzando un sotto-sottoprocesso come rappresentato nella parte alta di figura 5.21. La descrizione sarà poi fornita in un diagramma separato. Un altro uso dei sottoprocessi, ma in forma *espansa*, come mostrato nella parte inferiore di figura 5.21, è quello di eseguire un sottoinsieme delle azioni in un contesto diverso da quello del processo principale.

L'esempio mostrato in figura 5.22 mostra come è possibile inserire una scelta implicita in un processo espanso. Quando una delle due alternative viene eseguita, l'altra viene esclusa ed il sotto-processo viene terminato, questo sblocca il processo principale e consente di procedere

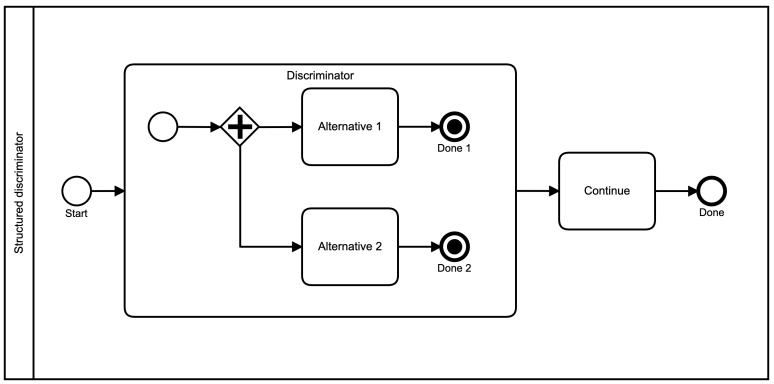

Figura 5.22: Esempio di sotto-processo (in alto) e sotto-processo espanso (in basso)

Un altro uso dei sottoprocessi espansi è per descrivere i cosiddetti processi *ad-hoc*, ovvero dei processi le cui attività non hanno un ordine predeterminato di esecuzione, potrebbero non essere eseguit per nulla oppure potrebbero esser eesguite più volte. I processi ad-hoc sono rappresentati tramite un sott-processo espanso e la notazione ~ (tilde) all'interno del sotto-processo.

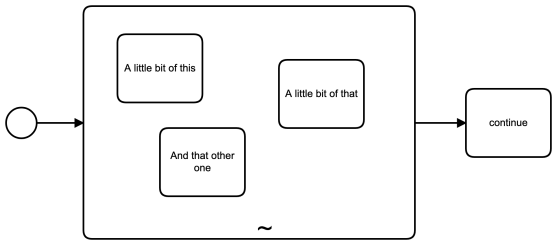

Figura 5.23: Esempio di sotto-processo (in alto) e sotto-processo espanso (in basso)

Infine, un sottoprocesso espanso permette le gestione di eventi asincroni (si veda 5.6) solo durante l'esecuzione di una parte delle attività.

5.5 BPMN: eventi

Gli eventi rappresentano qualche cosa che accade nel mondo esterno al processo e lo influenza, oppure un accadimento nel processo che influenza l'estero del progetto. In assenza di eventi un processo è un contesto a se stante che non interagisce con altri processi o con altri partecipanti o unità.

In BPMN sono definite tre categorie di eventi (figura 5.24):

- *Start*: eventi di inizio che scatenano l'avvio di un processo, ovvero la creazione di un'istanza di processo. Si possono avere diverse tipologie di evento di inizio: un inizio sempliche che rappresenta l'avvio esplicito del processo da parte di un utente autorizzato o un inzio temporizzato.
- *End*: eventi di fine che eliminano il token, ed eventualmente notificano qualche evento ad altri processi.
- *Intermediate*: eventi intermedi che permettono di bloccare il processo in attesa di un evento, oppure di generare un evento che influenzerà un altro processo.

Figura 5.24: Categorie di eventi

Oltre alle tre categorie di eventi, BPMN prevede diverse tipologie:

- timer: eventi legati al trascorrere del tempo,
- messaggi: eventi legati ad invio e ricezione di messaggi, indirizzati a specifici destinatari.
- segnali: eventi legati a segnali inviati a nessuno specifico destinatario,
- error: eventi legati all'occorrenza di un errore.

5.5.1 Eventi temporali

Tra gli eventi esterni, quello più ovvio è il trascorrere del tempo. Sono possibili eventi temporali (*timer*) sia di inizio che intermedi. I primi avviano il processo in corrispondenza di istanti predeterminati, i secondi mettono in pausa i processi fino al sopraggiungere di certi istanti.

Un evento temporale può fare riferimento a:

- un ora o giorno fissato,
- un intervallo di tempo (valido per gli eventi intermedi),
- un intervallo ricorrente (valido per gli eventi iniziali).

In figura 5.25 si osserva un esempio di evento temporale di inizio. Il processo viene attivato automaticamente dal SI ad intervalli regolari (ogni venerdì).

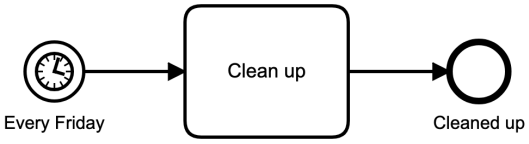

Figura 5.25: Esempio di evento temporale di inizio (*start timer*).

Per descrivere la necessitaà di processo di rimanere fermo in attesa per una certa durata o fino ad un certo istante si può usare un evento temporale intermedio, come nell'esempio di figura 5.26.

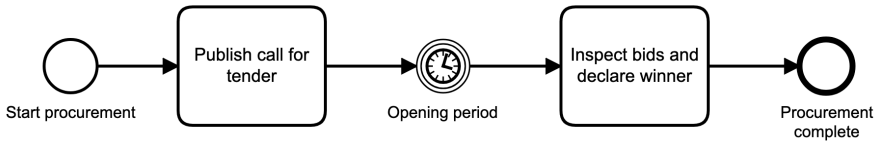

Figura 5.26: Esempio di evento temporale intermedio (*intermediate timer*).

Nel momento in cui il token raggiunge l'evento temporale, esso viene "fermato" e trattenuto fino allo scadere del tempo; raggiunto tale istante, il token viene rilasciato e prosegue il cammino. Quindi le azioni successive all'evento temporale potranno essere svolte solo dopo il tempo prefissato.

È importante sottolineare che il tempo di attesa deve essere noto al sistema informativo al momento in cui il token giunge nel nodo di evento. Tale tempo può quindi essere un valore costante oppure un valore calcolato in base alle informazioni disponibli nel SI in quell'istante. È perciò un errore utlizzare un evento temporale per descrivere l'attesa da parte del SI che un utente faccia qualche operazione: l'attessa dell'azione da parte dell'utente è parte della semantica delle attività: una volta abilitat, il SI resta in attesa che l'utente svolga l'attività.

Il nodo di attesa può specificare il tempo in due modi diversi:

1. in modo *assoluto*, specificando direttamente l'istante (giorno, ora, minuti, a seconda del livello di precisione richiesto) in cui il token dovrà essere liberato (es. "ore 20:00", "lunedì alle 9:00", "il 25 dicembre 2016");
2. in modo *relativo*, specificando la quantità di tempo per cui il token dovrà rimanere bloccato: tale quantità di tempo sarà misurata a partire dall'istante di ingresso del token nel nodo (es. "5 minuti", "2 giorni lavorativi", "4 ore e 30 minuti").

Solitamente, nel primo caso (istante assoluto di tempo di uscita) viene utilizzata una formulazione che indica un istante di tempo *ricorrente* (ossia che si ripete ogni giorno, ogni settimana, ogni anno, ...), poiché specificare un tempo *irripetibile* è poco utile. Ad esempio, il 31/12/2015 alle 9:00:00 si verifica un sola volta nella storia dell'universo, ed è improbabile che il sistema informativo debba

svolgere delle operazioni una sola volta nella storia. Ovviamente vi sono delle eccezioni: in un sistema informativo nato per gestire Expo 2015, le date di apertura e di chiusura dell'evento dovranno essere trattati come istanti unici ed irripetibili.

La specifica del tempo può avvenire anche in modo parametrico ("dopo X giorni lavorativi," "aspetta Y minuti"), purché il valore numero di tali parametri sia determinabile (partendo dalle informazioni in possesso al sistema informativo, quindi nel modello concettuale) nell'istante in cui il token raggiunge il nodo. In altre parole, nel momento stesso in cui il token raggiunge il nodo di attesa, è possibile determinare con precisione e senza eccezioni l'istante di tempo assoluto il cui il token verrà rilasciato (eventualmente facendo gli opportuni calcoli). Nessun altro avvenimento potrà liberare il token, se non il raggiungimento del tempo così determinato.

5.5.2 Messaggi

I messaggi hanno lo scopo di trasferire informazioni da un processo ad un altro. I processi sono rappresentati come pool diversi e possono rappresentare sia processi distinti all'interno della stessa organizzazione sia processi in organizzazioni diverse.

I messaggi sono sempre utilizzati in coppia (figura 5.27):

- invio (*throw*): rappresenta l'evento che invia le informazioni. In termini di semantica di esecuzione, quando il token raggiunge un evento di invio il processo istantaneamente invia un messaggio al destinatario ed il token procede (se si tratta di un evento intermedio) verso il flusso in uscita. Graficamente l'invio è rappresentato da un evento avente al centro un simbolo di una lettera con sfondo nero.
- ricezione (*catch*) rappresenta il punto in cui un processo si pone in attesa di un messaggio. Nell'esecuzione, quando il token giunge nell'evento di ricezione viene bloccato fino a quando arriva il messaggio (inviato dall'evento di invio corrispondente). Graficamente la ricezione è rappresentato da un evento avente al centro un simbolo di una lettera con sfondo bianco.

È prassi normale collegare, con una linea tratteggiata l'evento di invio a quello di ricezione corrispondente. I due eventi connessi devono trovarsi in due pool distinti: infatti all'interno di un pool/processo i dati sono condivisi e non c'è esigenza di inviare alcun messaggio.

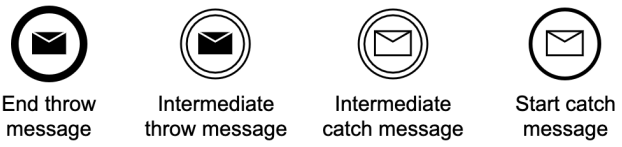

Figura 5.27: Eventi di invio e ricezione di messaggi.

Lo scambio di messaggi (e segnali) può svolgere due funzioni:

Sincronizzazione quando l'informazione importante da trasmettere è il fatto che si è raggiunto un determinato punto nell'elaborazione. Un segnale di sincronizzazione non porta con sé dati o informazioni aggiuntive, ma solamente l'informazione che il punto di invio segnale è stato raggiunto. Questa informazione può essere utilizzata per "sbloccare" altri processi che erano in attesa e che, per poter procedere, dovevano avere la certezza che alcune operazioni fossero completate. È analogo ad un messaggio "vai pure" (e, sottinteso, tu sai già dove e perché).

Scambio dati nel caso in cui il segnale, oltre a trasportare delle informazioni di sincronizzazione, trasporta anche delle informazioni specifiche (ad esempio, quelle corrispondenti all'istanza di una classe del modello concettuale). Il tal caso, al ricevimento del segnale, oltre all'informazione temporale (sul raggiungimento del punto di invio del segnale), il processo ricevente può incorporare anche una copia dei dati inviati. Questo tipo di segnali è essenziale nel caso di scambio di informazioni tra sistemi informativi diversi, in quanto essi non condividono l'accesso allo stesso sistema informativo.

L'esempio del modello mostrato in figura 5.28 mostra due pool che corrispondono ad un negozio online ed il gestore delle carte di credito che offre un servizio di POS virtuale. Nel pool principale, corrispondente ad un negozio online, dopo le fasi di riempimento del carrelo e di richiesta del conto, viene inviato un messaggio ad un altro pool, che rappresenta il servizio di POS viruale offerto da una banca o un gestore di carte di credito.

Il POS virtuale è un meccanismo abbastanza diffuso per i pagamenti su web. Menter i grossi siti di commercio elettronico (e.g. Amazon) gestiscono internamente le transazioni e le informazioni sulle carte di credito, gli altri siti si appoggiano a servizi di POS virutale per trattare i pagamenti. Il POS virtuale permette al gestore dei siti di non dover trattare le informazioni altamente riservate e critiche delegandolo a terze parti che possono garantire il livello di sicurezza necessario.

Il servizio di POS virtuale riceve dal negozio le informazioni sulla transazione: sostanzialmente l'importo da pagare ed il beneficiario. Il servizio, garantendo una comunicazione sicura, acquisisce i dati relativia alla carte di credito, convalida ed effettua la transazione di pagamento. Alla fine manda un messaggio al negozio online con l'esito della transazione.

Il negozio onlne in base all'esito della transazione potrà confermare l'ordine oppure terminare il processo.

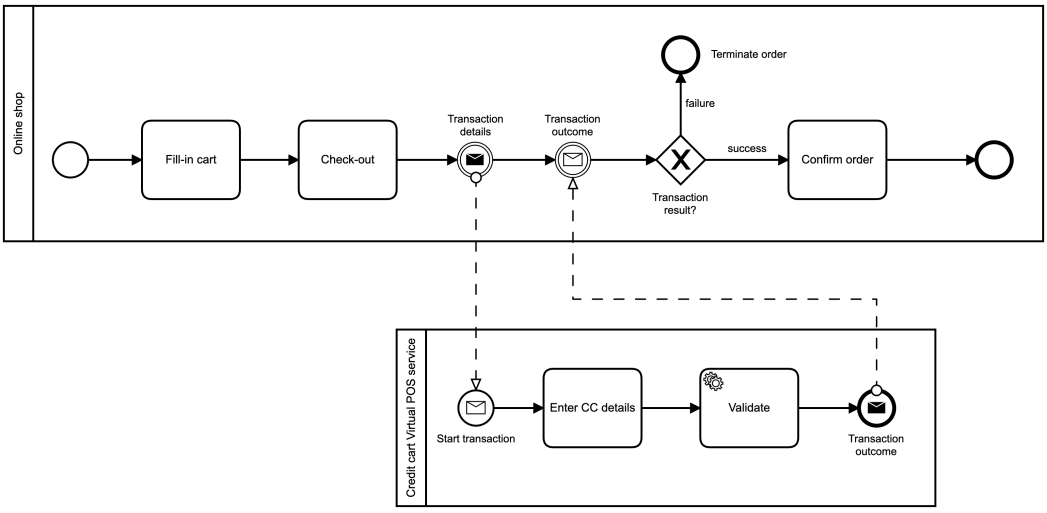

Figura 5.28: Esempio di uso di messaggi per comunicazioni tra enti diversi: POS virtuale

5.5.3 Segnali

I segnali sono analoghi ai messaggi: permettono l'interazione tra entità distinte o tra processi distinti. La differenze fondamentale è che mentre i messaggi hanno un preciso destinatario, i segnali sono inviati a tutti qualli che sono in ascolto: sta poi ai riceventi decidere se sono interessati al messaggio oppure se ignorarlo.

Come i messaggi anche i segnali sono solitamente in coppia (figura 5.29):

- invio (*throw*): rappresenta l'evento che invia le informazioni. Graficamente l'invio è rappresentato da un evento avente al centro un triangolo con sfondo nero.
- ricezione (*catch*) rappresenta il punto in cui un processo si pone in attesa di un segnale di un certo tipo. Graficamente la ricezione è rappresentato da un evento avente al centro un triangolo con sfondo bianco.

Il processo che riceve un segnale può prevedere, dopo l'evento di ricezione, una scelta per verificare se il segnale è rilevante per il processo oppure se può essere ignorato.

Figura 5.29: Eventi di invio e ricezione di segnali.

5.5.4 Scelta differita

Analogamente alla struttura della scelta implicita, è possibile effettuare una scelta in base a quale evento avviene per primo. Per fare questo è possibile utilizzare una gateway guidato dagli eventi. Come mostrato in figura 5.30, il gateway contiene un doppio cerchio con un pentagono a sfondo bianco. Il gateway può essere seguito esclusivamente da degli eventi intermedi di ricezione di tipo: timer, messaggio o segnale. È un errore mettere un'attività dopo un gateway guidato dagli eventi.

La semantica di esecuzione prevede che tutti gli eventi di ricezione, a valle del gateway siano attivati, il primo che avviene sottrae il token agli altri.

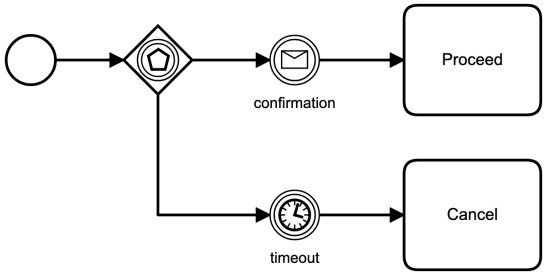

Figura 5.30: Scelta differita con gatway guidato dagli eventi (*Event-driven gateway*).

Figura 5.31 riporta un esempio di uso di una scelta differita (con il gateway guidato dagli eventi) per coordinare due processi. Il processo primario è quello che prevede la preparazione di un'asta: dopo l'apertura delle offerte c'è un gateway guidato dagli eventi che abilita la ricezione di messaggi (con le offerte) ed un timeout con la fine dell'asta.

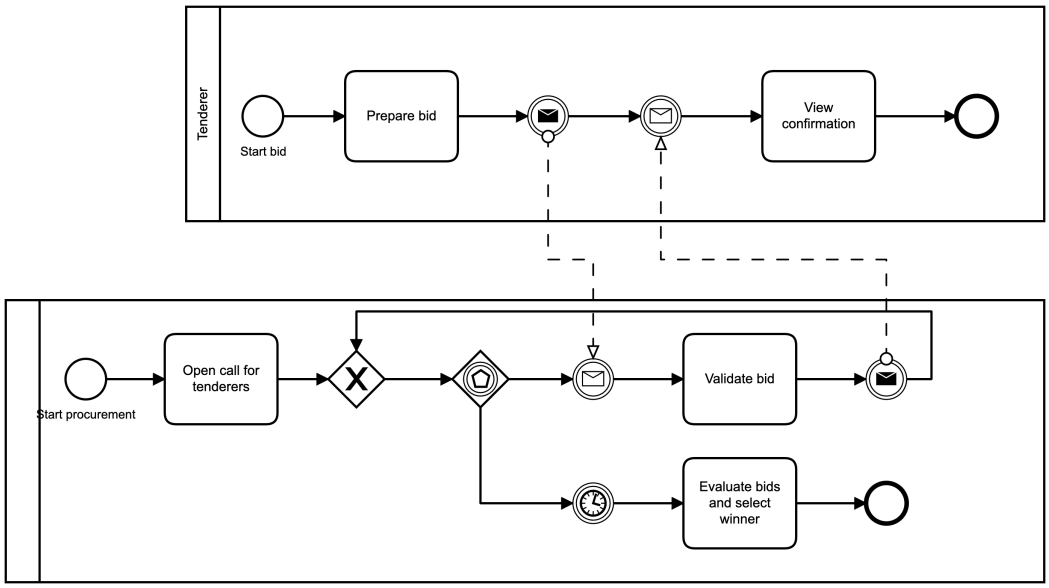

Figura 5.31: Esempio di scelta differita con timer e messaggi.

Il processo seguito da ciascun offerente è rappresentato dal pool superiore: dopo la preaprazione dell'offerta viene inviato il messaggio contenente l'offerta, dopo l'invio il processo si mette in attesa di una risposta con la conferma della ricezione dell'offerta, dopo la quale c'è una conferma mostrata all'utente.

Il processo principale prevede un ciclo che riceve le offerte le valida ed invia un messaggio di risposta con la conferma. L'altro ramo della scelta differita viene attivato alla scadenza del tempo per l'asta e prevede un'attivià di valutazione delle offerte e di scelta del vincitore.

Un'ulteriore evoluzione del processo è quella mostrata in figura 5.32 dove oltre all'uso di messaggi per lo scambio di offerte vengono utilizzati anche i segnali.

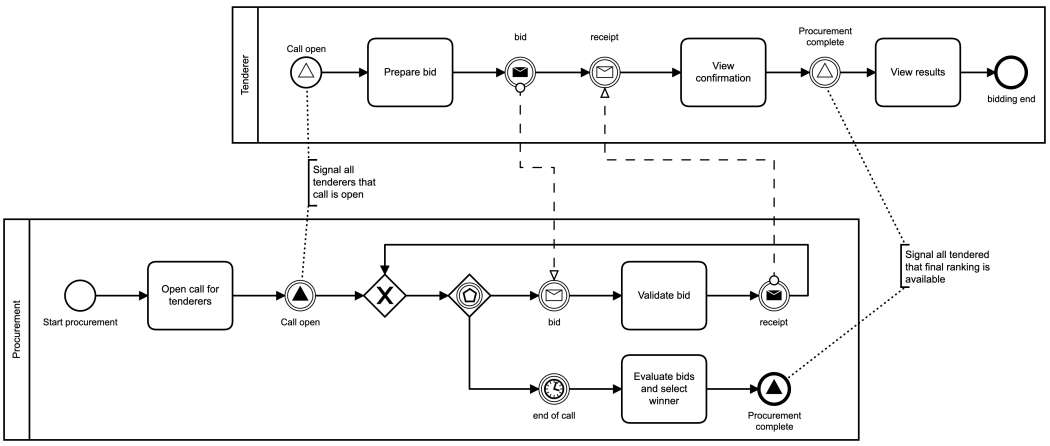

Figura 5.32: Esempio di scelta differita con timer, messaggi e segnali.

In questo processo, viene utilizzato un segnale per inviare una notifica a tutti i processi degli offerenti relativamente all'apertura dell'asta. Un analogo segnale viene poi inviato a valle della determinazione del vincitore dell'asta per informare tutti gli offerenti dell'esito finale.

5.5.5 Errori

DA SCRIVERE

5.6 BPMN: eventi asincroni

Le diverse tipologie di eventi descritti nella precedente sezione () sono ricevuti nel momento in cui un processo decide di attendere per essi. Ad esempio, per un messaggio, il mittente ed il destinatario sono contemporaneamente (sincronamente) pronti per lo scambio.

Esistono condizioni in cui un evento può accadere in un momento non predicibile, o inaspettatamente (es. per gli errori). In questi casi gli eventi devono essere gestiti in maniera asincronta.

Ci sono due tipologi e di costrutti BPMN che permettono di gestire eventi in maniera asincrona:

- **gli eventi a margine** di un'attività: permettono di specificare cosa fare nel caso di occorrenza di un evento durante l'esecuzione di un'attività,
- **i sottoprocessi per la gestionre di eventi**: permettono di descrivere come gestire un evento che può accedere in un istante qualsiasi durante l'esecuzione di un processo.

5.6.1 Eventi a margine

Gli eventi a margine sono specificati a livello di un'attività e indicano quali eventi possono accedere (o meglio essere presi in considerazione) durante l'esecuzione di un'attività (ovvero dal momento in cui il token la abilita fino a quando viene conclusa).

Il comportamente a fronte di un evento, rispetto all'esecuzione dell'attività, può essere di due tipi (figura 5.33):

- con interruzione (*interrupting*): in questo caso l'accadere dell'evento disabilita l'attività e d attiva il flusso in uscita dall'evento a margine. Gli eventi marginali con interruzione sono graficamente rappresentati come eventi intermedi con il doppio bordo continuo.
- senza interruzione (*non-interrupting*): l'occorrenza dell'evento non influenza l'esecuzione dell'attività ma attiva un nuovo percorso di esecuzione. Graficamente, gli eventi a margine senza interruzione sono rappresentati come eventi intermedi con il doppio bordo tratteggiato.

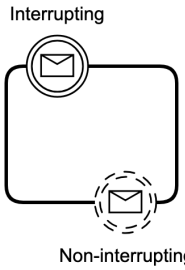

Figura 5.33: Eventi a margine (*Boundary events*).

A partire da un evento a margine viene definito un flusso in uscita che è quello che verrà attivato all'occorrenza dell'evento. Il flusso in uscita può combinarsi con quello principale in diversi modi come mostrato dagli esempi seguenti.

Figura 5.34 mostra un evento a margine con interruzione di tipo temporale. Quando il token attiva l'attività *Confirm* viene avviato il timer, se l'attività termina prima dello scadere del timeout il processo prosegue per il flusso principale. Se il timer scade prima del completamento dell'attività, l'attività viene disabilitata (interrotta) ed il token viene inviato sul flusso in uscita; in questo caso il processo viene terminato.

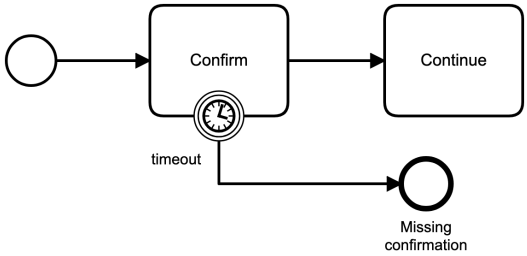

Figura 5.34: Esempio evento temporale a margine con interruzione.

Figura 5.35 mostra un evento a margine senza interruzione di tipo temporale. Quando il token attiva l'attività *Confirm* viene avviato il timer, se l'attività termina prima dello scadere del timeout il processo prosegue per il flusso principale. Se il timer scade prima del completamento dell'attività, viene generato un nuovo token che viene inviato lungo il flusso in uscita dall'evento marginale; in questo caso viene eseguita l'attività automatica di invio di un memento.

Figura 5.36 mostra come è possibile combinare il flusso di esecuzione legato ad un evento a margine con il flusso principale. Trattandosi di un evento a margine con interruzione di tipo messaggio, se e quando viene ricevuto il messaggio, il token viene sottratto all'attività (di fatto cancellandola) ed eseguita l'azione connessa all'evento. In ogni caso il flusso principale e quello dell'evento a margine di ricongiungono in un nodo di merge.

Figura 5.37 mostra un esempio di gestione di un errore: se si verifica un errore l'attività viene interrotta e vengono eseguite delle attività che possono risolvere il problema o mitigarlo.

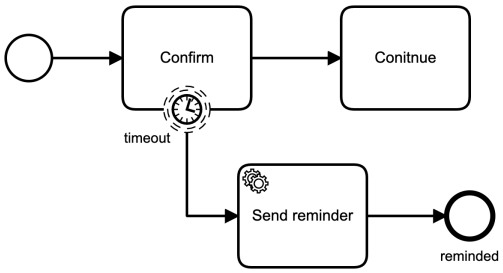

Figura 5.35: Esempio evento temporale a margine senza interruzione.

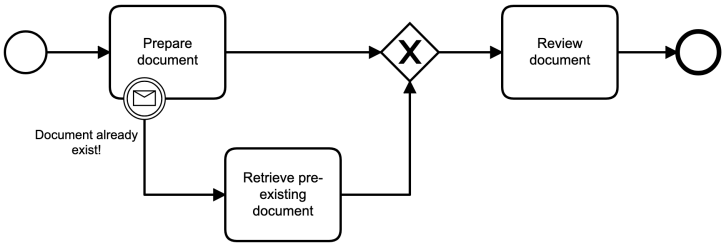

Figura 5.36: Esempio messaggio a margine con interruzione.

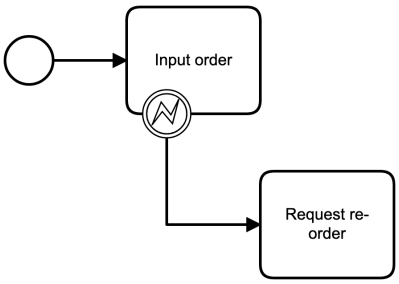

Figura 5.37: Esempio errore a margine con interruzione.

5.6.2 Sotto-preocessi per eventi

Gli eventi a margine, possono descrivere come trattare un evento asincrono che occorra durante un'attività. Nel caso in cui si voglia gestire un evento in qualunque attività si trovi il processo, sarebbe necessario utilizzare un evento a margine per ogni attività. In alternativa è possibile utilizzare un sotto-processo per la gestione degli eventi. Come illustrato in figura 5.38, graficamente si tratta di un sotto-processo espanso il cui bordo è puntinato. All'interno di tale sotto-processo l'evento di avvio permette di intercettare eventi che occorrano in un qualsiasi momento durante l'esecuzione del processo.

Anche per i sottoprocessi di gestione degli eventi è possibile avere una gestione:

- con interruzione (*interrupting*): in questo caso se accade l'evento viene cancellata (interrota) qualsiasi attività fosse in esecuzione ed attivato il processo. In questo caso l'evento di inizio del sottoprocesso viene rappresentato con il bordo continuo.

- senza interruzione (*non-interrupting*): l'occorrenza dell'evento non influenza l'esecuzione del processo, semplicemente viene attivato un nuovo token per l'esecuzione (contemporanea) del sottoprocesso. ma attiva un nuovo percorso di esecuzione. Graficamente, gli eventi a margine senza interruzione sono rappresentati come eventi intermedi con il doppio bordo tratteggiato.

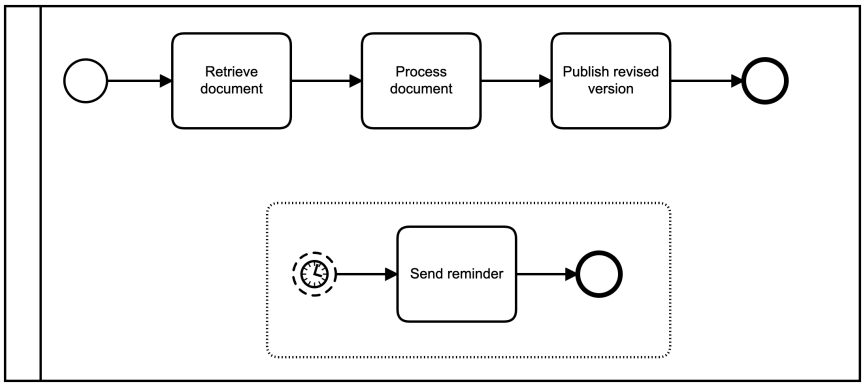

Figura 5.38: Esempio di sottorprocesso per la gestione di eventi.

5.7 Esempio di modellazione di processo

Si consideri il seguente scenario:

> *Un noto ristorante all'interno di un centro commerciale, utilizza un sistema informativo per la gestione della lista di attesa.*
>
> *Quando un cliente arriva, il cameriere, una volta noto il numero di persone nella comitiva, verifica sul sistema il tempo di attesa stimato per avere un tavolo. Quindi il cliente fornisce nome e numero di telefono cellulare, il sistema invia un SMS di conferma al telefono del cliente. A questo punto il cliente ha il tavolo prenotato.*
>
> *Il cliente può quindi andare via, ad esempio a visitare i negozi del centro commerciale. Ogni cinque minuti il sistema invierà un SMS con il tempo residuo di stimato.*
>
> *Quando il tavolo prenotato per il cliente si rende disponibile, un cameriere lo comunica al sistema che invia un SMS al cliente chiedendo di recarsi al ristorante non appena possibile.*
>
> *Quando il cliente si ripresenta al ristorante viene fatto accomodare al tavolo prenotato. Se il cliente non si presenta entro un tempo prefissato, il sistema richiama il prossimo cliente nella lista di attesa*

5.7.1 Modello concettuale

5.7.2 Processo

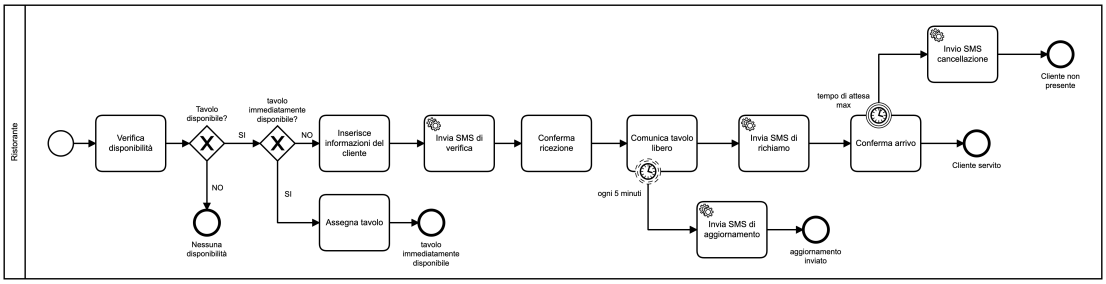

Figura 5.39: Processo di gestione della lista d'attesa.

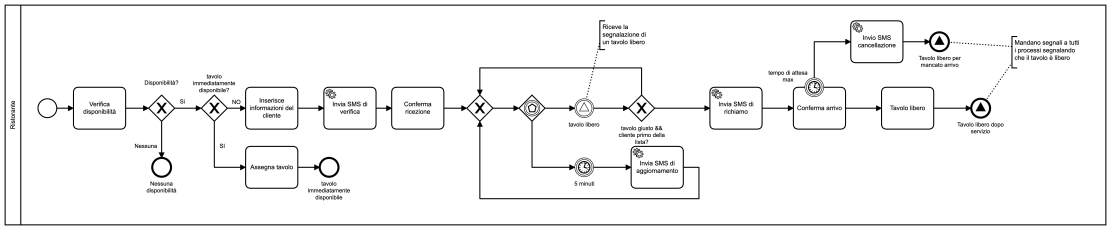

Figura 5.40: Variante del processo di gestione della lista d'attesa.

BIBLIOGRAFIA

[1] IDEF, "Idef0 function modeling method."

[2] OMG, *Business Process Modeling Notation (BPMN) Version 2.0*. Object Management Group, 2011.

[3] N. Russell, A. H. M. T. Hofstede, and N. Mulyar, "Workflow controlflow patterns: A revised view," tech. rep., BPM Center, 2006.

[4] B. Bruegge and A. H. Dutoit, *Object-Oriented Software Engineering Using UML, Patterns, and Java (3rd edition)*. Upper Saddle River, NJ, USA: Prentice Hall Press, 3rd ed., 2009.

[5] E. P. Council of the European Union, "Regolamento (ce) n. 1606/2002 del parlamento europeo e del consiglio, del 19 luglio 2002, relativo all'applicazione di principi contabili internazionali," Luglio 2002.

[6] *An Introductory Overview of ITIL® 2011*. TSO, 2012.

[7] M. E. Porter and V. E. Millar, "How information gives you competitive advantage," *Harvard Business Review*, July 1985.

[8] F. Roberts, *Measurement Theory with Applications to Decision Making, Utility, and the Social Sciences*. Addison-Wesley, 1979.

[9] S. W. Thomson, *Popular lectures and addresses*. Macmillan and Co., 1889.

[10] ISO/IEC/IEEE, *Systems and software engineering — Measurement process*, vol. ISO/IEC/IEEE 15939:2017(E). ISO/IEC/IEEE, 2017.

[11] D. T. Campbell, "Assessing the impact of planned social change," *Evaluation and Program Planning*, vol. 2, no. 1, pp. 67 – 90, 1979.

[12] R. N. Anthony, *Planning and control systems: a framework for analysis*. Division of Research, Harvard Business School, 1965.

[13] N. Bolloju and F. Leung, "Assisting novice analysts in developing quality conceptual models with uml," *Communications of the ACM*, vol. 49, p. 108, 112 2006.

[14] Bracchi, Francalanci, and Motta, *Sistemi informativi d'impresa*. McGraw Hill, 2010.

[15] P. Chen, "The entity-relationship model: toward a unified view of data," *ACM Transactions on Database Systems*, vol. 1, pp. 9–36, 1976.

[16] K. Fakhroutdinov, "Uml diagrams."

[17] M. Fowler, *UML Distilled: Guida rapida al lingugaggio di modellazione standard, 4a edizione*. Addison-Wesley, 2010.

[18] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd edition*. Addison-Wesley Professional, 2003.

[19] Laudon and Laudon, *Management dei Sistemi Informativi*. Prentice Hall, 2010.

[20] O. Lindland, G. Sindre, and A. Solvberg, "Understanding quality in conceptual modeling," *IEEE Software*, vol. 11, no. 2, pp. 42–49, 1994.

[21] OMG, *OMG Unified Modeling Language (OMG UML) Version 2.5*. Object Management Group, 2015.

[22] OMG, *Business Process Modeling Notation (BPMN) Version 1.0. OMG Final Adopted Specification*. Object Management Group, 2006.

[23] N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, and P. Wohed, "On the suitability of uml 2.0 activity diagrams for business process modelling," in *Proceedings of the 3rd Asia-Pacific Conference on Conceptual Modelling - Volume 53*, APCCM '06, (Darlinghurst, Australia, Australia), pp. 95–104, Australian Computer Society, Inc., 2006.

[24] S. S. Stevens, "On the theory of scales of measurement," *Science*, vol. 103, pp. 677–680, June 1946.

[25] I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard, *Object Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley Professional, 1992.

[26] A. Cockburn, *Writing effective use cases*. The crystal collection for software professionals, Addison-Wesley Professional Reading, 2000.

[27] S. Quintarelli, *Capitalismo Immateriale*. Bollati Boringhieri, 2019.

[28] W. B. Arthur, "Increasing returns and the new world of business," *Harvard Business Review*, July-August 1996.

LICENZA E COLOPHON

Questo volume è stato redatto con il sistema di composizione LaTeX[1] utilizzando il modello di stile `memoir`[2].

Il contenuto del testo è rilasciato con la licenza Creative Commons Attribuzione - Non commerciale - Condividi allo stesso modo 2.5 Italia (CC BY-NC-SA 2.5)[3].

[1] `http://www.latex-project.org/`
[2] `http://www.ctan.org/tex-archive/macros/latex/contrib/memoir/`
[3] `http://creativecommons.org/licenses/by-nc-sa/2.5/it/`