Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

PROGETTO E IMPLEMENTAZIONE DI UN SIMULATORE DI PROCESSI BASATO SU BPMN E BPSIM

Relatore:
Chiar.mo Prof.
DAVIDE ROSSI Relatore:
Chiar.mo Prof.
FRANCESCO POGGI

Presentata da:
RICCARDO CANDIDO

Sessione II
Anno Accademico 2017/2018

A Camilla e Cecilia

Introduzione

In questo elaborato si descrive lo stato dell'arte per quanto concerne le tematiche di simulazione dei processi aziendali definiti utilizzando la specifica *BPMN 2.0*. In seguito si descrive nel dettaglio lo sviluppo e il funzionamento di un applicazione in grado di simulare processi aziendali definiti tramite le specifiche *BPMN* e *BPSim*.

La notazione *BPMN* ha come obbiettivo fornire uno strumento di rappresentazione univoco al fine di modellare, progettare ed eventualmente informatizzare i processi aziendali. In questo elaborato si affianca a questa notazione un ulteriore standard chiamato *BPSim*. Questa notazione permette di decorare il modello con informazioni utili al fine di simulare il processo *BPMN* descritto.

Si è utilizzato *Camunda* come motore di simulazione. Questo applicativo è un workflow engine in grado di processare ed eseguire buona parte dei costrutti supportati dallo standard *BPMN 2.0*.

Si è deciso quindi di sviluppare di un applicazione che, utilizzando le funzionalità esposte dall'applicativo *Camunda*, eseguisse la simulazione integrando le informazioni fornite dalla specifica *BPSim* fornendo così risultati di notevole interesse per quanto concerne la valutazione dei processi aziendali.

Indice

Introduzione **i**

1 Business Process Simulation **1**

1.1 Business Process Simulation 1

1.2 Processo 3

1.3 Elementi per la modellazione e Misure di perfromance 4

1.3.1 Elementi base di modellazione 4

1.3.2 Misure della performance di un modello 5

1.4 Come modellare un processo 6

1.4.1 Project-based processes 7

1.4.2 Production-based processes 8

1.4.3 Distribution-based processes 8

1.4.4 Customer service based processes 9

2 BPMN e BPSim **11**

2.1 BPMN 11

2.1.1 Come utilizzare BPMN 13

2.1.2 Gli elementi BPMN 17

2.1.3 Strumenti per la modellazione *BPMN* 19

2.2 BPSim 21

2.2.1 Gli elementi BPSim 22

2.2.2 Scenario 22

2.2.3 Parameters 23

2.2.4 BPM Simulation Tools 25

3 Firefly Simulator **31**
3.1 Obbiettivi 31
3.2 Interazione con prodotti sviluppati da terze parti 32
3.2.1 Camunda 32
3.3 Requisiti tecnici 35
3.3.1 Parametri di input 36
3.3.2 Parametri di Output 40
3.3.3 Tipologia di utenti 40
3.3.4 Interfaccia 41

4 Architettura e Implementazione **43**
4.1 Architettura del progetto 43
4.2 Sequenza della simulazione 44
4.2.1 Pre-processamento del file BPMN 45
4.2.2 Conversione dei task in user task 45
4.2.3 Aggiunta dei listener negli user task 46
4.2.4 Conversione dei Boundary Events in Message Boundary Events . 46
4.2.5 Sostituzione delle espressioni di condizione 46
4.2.6 BPSim Parsing 47
4.2.7 Simulazione 48
4.3 Algoritmo di gestione della coda degli eventi 56
4.3.1 SimulationIntermediateEvent 56
4.3.2 SimulationStartEvent 56
4.3.3 SimulationTaskEvent 57
4.4 Risultati 65

5 Conclusioni **67**

Bibliografia **71**

Elenco delle figure

2.1 Private Business Process 14
2.2 Public Business Process 15
2.3 Collaboration 16
2.4 Esempio di una Choreography 16
2.5 Architettura BPSim 22
2.6 BIMP Home Screen 27
2.7 Risultati simulazione (BIMP) 28

3.1 Architettura di Camunda 32
3.2 Architettura Camunda Engine 33

4.1 Architettura dell'applicazione Firefly 43
4.2 Diagramma di sequenza Simulazione Firefly 44
4.3 Digramma UML del package Simulation 50
4.4 Coda di priorità 52
4.5 *SimulationClock* in processo con attività lineari 54
4.6 *SimulationClock* con attività parallele 55
4.7 *SimulationClock* con attività parallele e risorse 55
4.8 Algoritmo con task paralleli e risorse limitate (1) 58
4.9 Algoritmo con task paralleli e risorse limitate (2) 59
4.10 Algoritmo con task paralleli e risorse limitate (3) 60
4.11 Algoritmo con task paralleli e risorse limitate (4) 61
4.12 Algoritmo con task paralleli e risorse limitate (5) 62
4.13 Algoritmo con task paralleli e risorse limitate (6) 63

4.14 Algoritmo con task paralleli e risorse limitate (7) 64
4.15 Algoritmo con task paralleli e risorse limitate (8) 65

Elenco delle tabelle

2.1 Elenco degli elementi *BPMN* 17

3.1 Griglia incrociata parametri *BPSim* e costrutti *BPMN* 37

Capitolo 1

Business Process Simulation

Questo capitolo introduce la nozione di Business Process Simulation e per quale motivo tale strumento è considerato essenziale nei processi decisionali di un'azienda. Successivamente viene definito cosa è un processo aziendale e la sua caratterizzazione. Conseguentemente vengono descritti gli elementi utilizzati per la modellazione. Infine si discutono le best practises per modellare un processo descrivendo come queste si sono evolute.

1.1 Business Process Simulation

Nel settore industriale come in quello dei servizi nuove sfide emergono e viene richiesto un costante lavoro di raffinamento e miglioramento agli agenti che operano in questi settori al fine di minimizzare i tempi con i quali si forniscono nuovi servizi e prodotti ai clienti con il fine ultimo di soddisfare a pieno la domanda.

E' compito del business manager minimizzare i rischi e massimizzare i profitti valutando quali alternative sono percorribili nel minor tempo possibile.

Queste alternative possono avere diversa natura, possono richiedere un automatizzazione o l'esternalizzazione di un processo, l'espansione o il ridimensionamento della forza lavoro.

Valutare l'impatto di un cambiamento nei processi aziendali con accuratezza e velocità è diventato un fattore critico nei mercati.

La comprensione e la stima del tempo e del costo necessari al completamento di un prodotto è una sfida quotidiana per chi opera in questo campo.

Tipicamente queste tipologie di problemi vengono affrontate utilizzando strumenti manageriali di diversa natura. Questi strumenti vengono impiegati dal management per pianificare al meglio l'attività aziendale e pianificare i propri obbiettivi.

Ad ogni modo, questa tipologia di analisi rimane molto complessa per la natura intrinseca delle attività analizzate. Infatti l'analisi delle tempistiche diviene molto complicata nel momento in cui entrano in gioco le numerose interdipendenze che si vengono a creare fra le varie risorse disponibili. Rendendo quindi questo processo di analisi dei costi e delle risorse utilizzate molto complesso nonostante l'utilizzo dei suddetti strumenti manageriali.

A queste problematiche si aggiungono anche le richieste dei clienti; i quali con il passare del tempo tendono a pretendere costi sempre più bassi e qualità maggiore.

Ne consegue che parametri come il costo richiesto per completare una determinata attività o i tempi d'attesa diventano quindi fondamentali per fornire una determinata qualità del servizio a un prezzo sempre competitivo.

Tipicamente gli attori incaricati di prendere decisioni in merito alla gestione delle attività e al conseguente prezzo con il quale un prodotto si presenta sul mercato, hanno sempre utilizzato strumenti analitici. Tali strumenti però non tengono in considerazione l'aleatorietà e la dinamicità dei sistemi nei quali un'azienda è costretta ad operare.

Un problema tipico per una impresa è il dimensionamento delle scorte. La variazione nella domanda, nei tempi di processamento delle richieste, la complessità data dalle operazioni di approvvigionamento e le dinamiche di sistema portano a un incertezza che non può essere modellata o analizzata utilizzano gli strumenti più comuni quali fogli di lavoro elettronici e flowchart.

Al contrario una simulazione eseguita tramite l'utilizzo di sistemi informatici combinata con la semplicità e la comprensione tipica dei flowchart fornisce uno strumento che in termini di costi, accuratezze e rapidità nel processo decisionale non ha eguali oggigiorno.

Questo strumento permette di valutare alternative prima che avvengano ingenti investimenti in termini di risorse e tempo.

La capacità di visualizzare come un processo possa svolgersi e come questi si vada poi a posizionare all'interno della struttura aziendale misurandone le performance e valutandone le eventuali variazioni in un modello computazionale rende la simulazione lo strumento de facto, strettamente necessario nel processo decisionale aziendale.

Business Process Simulation (BPS) incarna il concetto che un business è una serie di processi in relazione fra di essi, e che questi processi consistono di attività che convertono input in output.

L-approccio BPS è in grado di catturare:

- I limiti delle risorse che si hanno a disposizione
- Le regole decisionali
- L'aleatorietà dell'ambiente nel quale l'azienda opera

Un process model, quando simulato mima le operazioni dell'attività modellata.

Questo viene realizzato eseguendo passo dopo passo gli eventi definiti dal modello tenendo conto delle tempistiche che questi eventi richiedono nel mondo reale. Infine la simulazione fornisce statistiche riguardanti gli elementi del modello. In questo modo è possibile valutare le performance di un processo analizzando i dati prodotti come output dalla simulazione.

1.2 Processo

Diverse sono le definizioni di processo che si possono trovare in letteratura. In questo caso specifico si definisce un processo come:

Una collezione di attività che prende uno o più tipologie di input e crea un output che è di valore per il consumatore [1].

E' possibile quindi astrarre il concetto di processo definendo un modello. La definizione di un modello di processo permette di stabilire uno schema di riferimento così da permetterne una riproduzione che non sia dipendente dal contesto:

Un modello di processo è una rappresentazione formale di una serie di attività in relazione fra loro che sono eseguite in un ordine prestabilito al fine di conseguire un determinato obbiettivo [1].

1.3 Elementi per la modellazione e Misure di performance

Sebbene i costrutti utilizzati per definire modelli di processo possano avere differenti nomenclature o caratteristiche per diversi prodotti, molti dei modelli di simulazione di business process contengono grosso modo gli stessi costrutti. Successivamente si descrivono le misure della performance di un modello.

1.3.1 Elementi base di modellazione

In questa sezione si descrivono gli elementi di modellazione utilizzati per descrivere un processo. Il primo a ipotizzare tale classificazione che è diventata lo standard de facto è stato Tumay in *Business Process Simulation* [3].

I quattro costrutti base sono:

1. **Token (entità)**: Altrimenti definiti come oggetti di flusso. Questi sono oggetti che possono essere processati dalle risorse. Di fatto rappresentano l'entità di una determinata attività che viene presa in carico dalla risorsa responsabile.

2. **Risorse**: Quegli elementi necessari a un gruppo di lavoro per comprendere un problema e implementare le relative soluzioni. Sono considerate risorse anche: il tempo che viene dedicato all'elaborazione di una soluzione, il capitale umano di un'azienda, etc.

3. **Attività**: I compiti svolti dalle entità come l'assemblaggio, la produzione di documenti o l'interazione con un cliente al fine di completare una richiesta. Inoltre un attività trova una definizione completa quando viene decorata: dal tempo richiesto per completarla e dalle risorse necessarie al suo soddisfacimento.

4. **Connettori**: I connettori sono utilizzati per collegare processi e attività Le entità seguono tali connettori in base al flusso definito dal modello. I connettori sono utili per definire flussi paralleli o per rieseguire determinati attività in base alle regole definite dal modello.

1.3.2 Misure della performance di un modello

Le misure più comuni utilizzate per valutare le perfomance di un modello di processo sono il cycle time, il conteggio delle entità, l'utilizzo delle risorse e i costi delle attività.

1. **Cycle time**: è il tempo totale che un entità impiega per attraversare l'intero process. Questo tempo include il tempo di processamento, il tempo di attesa, il tempo di spostamento da un attività all'altra, etc. Altri misure di notevole interesse sono corollario di questa; detto ciò è possibile fornire anche il cycle time medio nel caso in cui vengano eseguite simulazioni multiple.

2. **Conteggio delle entità**: Il conteggio delle entità si riferisce al numero totale delle entità che attraversano un processo o che sono ancora processate. Questa misura si ottiene calcolando la somma dei token processati e sommando i token ancora all'interno del processo. La grandezza di questa misura è in grado di fornire uno strumento utile per comprendere la confidenza da attribuire ai risultati ottenuti dalla simulazione.

3. **L'utilizzo delle risorse**: Durante una simulazione, le risorse cambiano stato da non disponibili a occupate. Questo tipo di misura fornisce una la percentuale di tempo che una risorsa impiega in un determinato stato. La disponibilità e l'assegnamento delle risorse è dettato dalla necessità delle attività nel modello di utilizzare o meno una risorsa. Ne consegue che l'utilizzo di una risorsa fornisce un utile statistica nel misurare e analizzare il sotto utilizzo o il sovra utilizzo delle risorse.

4. **Costi delle attività**: Una risorsa, quando è definita in un modello di process, è caratterizzata da: il numero di unità disponibili, il costo di utilizzo, il costo di installazione e i costi fissi a essa correlati. A questi parametri si aggiungono anche le risorse richieste per eseguirla e il tempo necessario per completarla. Durante una simulazione, gli strumenti di BPS devono essere in grado di tenere automaticamente traccia del tempo che ogni token impiega in ogni attività e il tempo che ogni risorsa impiega per completare l'attività in carico in un dato momento. I costi delle attività forniscono una misura realistica molto utile per analizzare i costi derivanti da un

processo. Gli strumenti BPS sono quindi in grado di fornire informazioni molto dettagliate per quanto concerne i costi di un'attività, ad ogni modo tali informazioni possono essere poi aggregate fornendo quindi misure di carattere più generale molto utili durante l'analisi delle performance di un processo.

Esistono inoltre, misure di performance avanzate che utilizzano elementi di modellazioni avanzati come attributi o espressioni. Queste misure sono in grado di dare informazioni utili riguardanti il livello di servizio fornito al cliente, la puntualità nel completamento di determinate attività, etc. [2].

1.4 Come modellare un processo

La fase iniziale nello sviluppo di un processo è la progettazione. Questa fase permette di descrivere un processo formalmente. Un modello di processo formale fa si che la definizione del processo non sia ambigua, ciò significa che durante la sua implementazione non vi sia spazio per interpretazioni personali nel suo funzionamento. Esistono diverse sintassi che si possono utilizzare per descrivere un processo. Fra i linguaggi più ampiamente diffusi per descrivere un processo troviamo:

1. **Flowchart**: i flowchart non sono altro che un insieme di rettangoli e line. Non hanno una sintassi ricca. Ne consegue che caratterizzare un processo utilizzando questa sintassi può diventare complesso.

2. **Event Process Chain Diagrams (EPC)**: Un grafo ordinato di eventi e funzioni; ogni attività all'interno del processo è attivata da un evento.

3. **Business Process Model and Notation (*BPMN*)**: Lo standard de facto per la rappresentazione dei diagrammi di business process [university michigan]. Questa sintassi verrà ampiamente discussa nel secondo capitolo.

4. **Business Process Execution Language (BPEL)**: Un linguaggio basato su xml per la rappresentazione dei business processes. BPEL è un linguaggio di orchestrazione, permettendo di interagire con altri servizi.

5. **Petri nets (PN)**: è un linguaggio matematico di modellazione per sistemi distribuiti. Ne esiste una sua versione grafica che può essere utilizzata per rappresentare i modelli di processo [2].

Per quanto concerne le tecniche di modellazione invece, esistono diverse metodologie che permettono di modellare un processo. E' necessario soffermarsi su queste tecniche poiché il rischio di modellare un processo in maniera errata comporta la produzione di risultati non affidabili.

Tali metodologie di Business process sono classificate in quattro macrocategorie :

1. Project-based processes
2. Production-based processes
3. Distribution-based processes
4. Customer service based processes

Questa classificazione, introdotta da Tumay in *Business Process Simulation* [3], non vuole imporre che tutti i processi esistenti debbano rientrare in una di queste categorie. Ad ogni modo le considerazioni presentate nelle sottosezioni che seguono dovrebbero fornire le principali linee guida da utilizzare nel momento in cui si vuole modellare un processo e analizzarlo utilizzando tecniche di simulazione.

1.4.1 Project-based processes

Questi processi sono solitamente dati in carico a una singola persona o a un gruppo ristretto di persone. Esempi tipici sono lo sviluppo di prodotto o i processi amministrativi all'interno di un'azienda. L'utilizzo della simulazione nell'analisi dei project-based processes fa si che si producano risultati molto più accurati perché i tempi per le attività tipiche per queste tipologie di processi tendono ad avere tempistiche molto variabili e l'utilizzo di risorse multiple fa si che si vengano a creare diverse interdipendenze che è difficile ipotizzare a priori. Questi tipi di simulazione devono tenere in considerazione diversi parametri quali la prioritizzazione di alcuni task, la curva di apprendimento per le

risorse investite nel processo, etc. Tutti questi parametri sono fondamentali per costruire un sistema di simulazione valido e che produca risultati apprezzabili.

In questa tipologia di processi è fondamentale essere in grado di eseguire un numero considerevole di simulazioni. Dal momento che il tempo per ogni singola attività può variare notevolmente data la natura del processo. L'esecuzione di prove multiple produce un numero considerevole di osservazioni le quali forniscono una più accurata stima delle misure di performance.

1.4.2 Production-based processes

In questo tipo di simulazioni gli output sono prodotti in grandi quantità, solitamente in lotti o come flusso continuo. Rientrano in questa categoria processi come: evasione degli ordini, reclami, etc. Per modellare accuratamente le attività che compongono tali processi, un modello deve consentire di tenere traccia delle singole entità e dei loro attributi.

Questa categoria di modelli di processo viene solitamente simulata con il fine di ottenere un processo a regime stazionario, cioè un modello nel quale viene sempre prodotta la medesima sequenza di prodotti.

Un concetto fondamentale nell'analisi delle prestazioni è l'eliminazione delle problematiche relative ai dati raccolti durante la fase di *warm-up* della simulazione. I dati prodotti in questa fase non vengono considerati validi. In questa prima parte della simulazione è quindi buona pratica non raccogliere i dati, attendendo che la fase di *warm-up* termini.

1.4.3 Distribution-based processes

Questa categoria di processi include i processi di trasporto e consegna. Qui i prodotti o le persone vengono trasportati tra le diverse sedi attraverso una rete di distribuzione. Una differenza fondamentale tra trasporto e consegna è che le Entità che vengono trasportate sono persone piuttosto che merci. I processi di trasporto si trovano nei sistemi di trasporto di massa come nelle compagnie aeree o ferroviarie. Al contrario i processi di consegna si trovano nella distribuzione della produzione, nei servizi postali o di logistica.

Quando si modellano i processi di distribuzione, è importante definire gli attributi delle singole entità in maniera puntuale, quali: destinazione, dimensione o costo. Quando si modella il trasporto, a volte può essere più appropriato rappresentare le risorse di trasporto come entità piuttosto che come risorse. La maggior parte dei processi di distribuzione è transitoria nel comportamento. Pertanto, il periodo di simulazione dovrebbe essere abbastanza lungo così da permettere di completare tutte le attività all'interno del processo, viene inoltre consigliato di eseguire il maggior numero di repliche per analizzare le misure di prestazione.

1.4.4 Customer service based processes

Questa tipologia di processi è quella che vede la maggior area di applicazione per la simulazione dei processi. Questo è dato dal fatto che il tempo totale di attesa è solitamente uguale al 95 % del tempo totale di esecuzione di un processo basato sui servizi.

Solitamente queste tipologie di servizi vedono come unici attori gli esseri umani. Alcuni esempi sono i servizi telefonici come i call centers o i servizi sanitari come ospedali e farmacie.

Gli esseri umani hanno un comportamento molto più complicato e non predicibile rispetto a prodotti, documenti etc.

A questo si aggiunge che le risorse, in questo caso gli umani, possono cambiare il proprio comportamento a seconda dello stato nel quale si trova l'entità che prendono in carico.

Senza contare che solitamente queste tipologie di servizi hanno tempistiche molto variabili e i clienti tendono ad arrivare in maniera casuale rendendo sempre più complessa la modellizzazione in queste tipologie di processo.

In questo caso è appropriato l'utilizzo di distribuzioni al fine di rappresentare la componente aleatoria tipica di questi fenomeni.

Ne consegue che è molto importante eseguire diverse simulazioni nel caso in cui si analizzino processi di questa tipologia, in modo da ottenere risultati statisticamente validi.

Capitolo 2

BPMN e BPSim

Questo capitolo tratta in primo luogo la specifica *BPMN* 2.0. Ne vengono descritti tutte le componenti e il loro utilizzo. In secondo luogo viene introdotta nel dettaglio la specifica *BPSim* e il ruolo che ricopre nel suo affiancamento a *BPMN*. Successivamente si descrive lo stato dell'arte per quanto concerne i programmi di simulazione che utilizza lo specifiche citate e le motivazioni che hanno portato a sviluppare l'applicazione oggetto di questo elaborato.

2.1 BPMN

BPMN è uno standard sviluppato dall'**Object Management Group Inc. (OMG)**. La versione della specifica utilizzata per questo caso di studio è la 2.0 rilasciata nel 2011 da OMG. OMG è un organizzazione con partecipazione aperta, non a scopo di lucro che opera nel campo dell'informatica. Il suo scopo è produrre e mantenere standard con il fine di rendere quest'ultimi interoperabili, portabili e riusabili in contesti enterprise. Fanno parte di questa organizzazione aziende operanti nel settore dell'informatica, utenti, agenzie governative ed università.

L'Object Management Group (OMG) ha sviluppato lo standard Business Process Model and Notation (*BPMN*). L'obbiettivo primario di *BPMN* è fornire una notazione che sia comprensibile a un ventaglio il più ampio possibile di ruoli. Questa specifica infatti è stata pensata perché possa essere utilizzata da coloro che operano nel campo del

business come gli analisti i quali sono incaricati di produrre la bozza iniziale dei processi aziendali, dagli sviluppatori che invece hanno il compito di implementare, utilizzando le tecnologie esistenti, il processo definito e infine dal management che deve gestire e monitorare tali processi. Quindi *BPMN* si va a posizionare in quella zona grigia che vi è fra la progettazione di un processo e la sua implementazione. Un altro obbiettivo, ma non meno importante, è assicurarsi che i linguaggi progettati per l'esecuzione del business process; possano essere visualizzati con una notazione business-oriented. Questa specifica rappresenta una fusione delle best practises definite dalla comunità di business modeling al fine di fornire la notazione e le semantiche dei diagrammi di collaborazione, dei diagrammi di processo e dei diagrammi di coreografia. L'intento di *BPMN* è di standardizzare la notazione utilizzata per definire i business process model al fine di ovviare tutte le problematiche relative alla presenza di diverse notazioni di modellazione. Nel fare ciò, *BPMN* fornisce un semplice meccanismo per comunicare le informazioni relative a un processo ad altri agenti operanti nello stesso business come sviluppatori, clienti e fornitori.

Nell'ultimo decennio si sono sviluppati altri linguaggi come WSBPEL (Web Service Business Process Execution Language). Al fine di fornire un meccanismo formale dei processi che è quello che si ripromette di fare *BPMN*. L'aspetto fondamentale che differenzia questi linguaggi da *BPMN*; è che tali linguaggi sono ottimizzati per interoperare con sistemi software tipici del business process management.

L'ottimizzazione di questi linguaggi per operazioni software, li rende meno adatti per l'utilizzo diretto da parte di coloro che operano nel management. Data la natura di WSBPEL, un Business Process complesso potrebbe essere organizzato potenzialmente in un formato complesso, disgiunto e non intuitivo che è gestito in maniera corretta dal sistema software ma che sarebbe di difficile comprensione per gli analisti business e i manager che hanno sviluppato tale processo. Quindi il grado di portabilità di questa rappresentazione a un livello di facile comprensione per l'uomo è discretamente basso. Al contrario le sezioni di Business Analysis hanno una notevole dimestichezza con la visualizzazione dei processi Business in formati come i flow-chart. Questo divario crea un gap tecnico notevole fra l'iniziale formato di progettazione di un Business Processes e il formato che linguaggi come WSBPEL forniscono.

BPMN fornisce diversi diagrammi, i quali sono pensati per essere utilizzati dalle persone che progettano e gestiscono i Business Processes, questi poi possono essere mappati in un linguaggio di esecuzione dei sistemi come WSBPEL.

Inoltre, *BPMN* fornisce un meccanismo standardizzato di visualizzazione per i Business Processes permettendo poi di eseguire lo stesso processo utilizzando un BPM engine.

Questa specifica fornisce una formato di interscambio per i business porcess. Il quale può essere utilizzato per scambiarsi definizione di processi in *BPMN* fra diversi strumenti di sviluppo.

L'obbiettivo di questa specifica è di promuovere la portabilità delle definizioni di processo, così che gli utenti possano usare tali definizioni rimanendo indipendenti dai tool con le quali sono state progettate.

2.1.1 Come utilizzare BPMN

Il Business Process modeling è utilizzato per comunicare un larga varietà di informazioni a una audience molto eterogenea. *BPMN* è progettato per coprire molti dei tipi di modellazione e permette la creazione di processi end-to-end. Gli elementi strutturali di *BPMN* permetto di differenziare facilmente le sezioni del diagramma. Vi sono 3 tipi base di sotto-modelli dentro un modello *BPMN* end-to-end.

1. Processi (Orchestrazione), i quali includono:
   (a) **Private non-executable** (internal) Business Processes
   (b) **Private executable** (internal) Business Processes
   (c) **Public Processes**
2. Collaborazioni, le quali possono includere processi e/o coreografie
3. Coreografie

Le tipologie di diagrammi mostrate successivamente vengono definite *token based*. Tale token attraversa i *Sequence Flows* e passa tramite gli elementi dei processi. Il token è un concetto teorico utilizzato per meglio spiegare il comportamento di un processo

che viene eseguito. Il comportamento degli elementi di un processo può essere definito descrivendo cosa avviene all'interno degli elementi attraversati dal token. Uno *Start Event* genera un token che deve essere eventualmente consumato da un evento di fine. Il percorso del token deve essere tracciabile tramite la rete di *Sequence Flows*, *Gateways* e *Activities* descritta nel processo.

Private (Internal) Business Processes

I Private Business Processes sono quelli interni a una organizzazione specifica (Figura 2.1). Questi processi sono stati generalmente chiamati workflow o BPM Processes. Un altro sinonimo spesso utilizzato nel linguaggio tipico dei web services è Orchestration di servizi. Esistono due tipi di private Processes: eseguibili e non eseguibili. Un processo eseguibile è un processo che è stato modellato con lo scopo di venire eseguito in funzione della semantica definita dalla specifica *BPMN*. E' facile immaginare che durante la fase di sviluppo del processo, ci saranno delle fasi dove il processo non ha abbastanza dettagli per essere eseguito. Un processo non eseguibile è un processo privato che è stato modellato con il fine ultimo di documentare il comportamento di un processo in maniera oggettiva.

Figura 2.1: Esempio di un Private Business Process

Public Processes

Un Public Process rappresenta l'interazione fra un processo privato e un altro processo o partecipante (Figura 2.2).

Solo quelle attività che sono utilizzate per comunicare gli altri partecipanti sono incluse nel Public Process. Tutti le altre attività "interne" al Private Business Process non sono mostrate nei Public Process. Così, i processi pubblici mostrano solo ciò che avviene nel mondo esterno come il flow dei messaggi e l'ordine di quei flussi di messaggio che

sono necessari per interagire con i processi. I Public Processes possono essere modellati separatamente o dentro una collaborazione per mostrare il flusso dei messaggi fra le attività del processo pubblico e gli altri partecipanti.

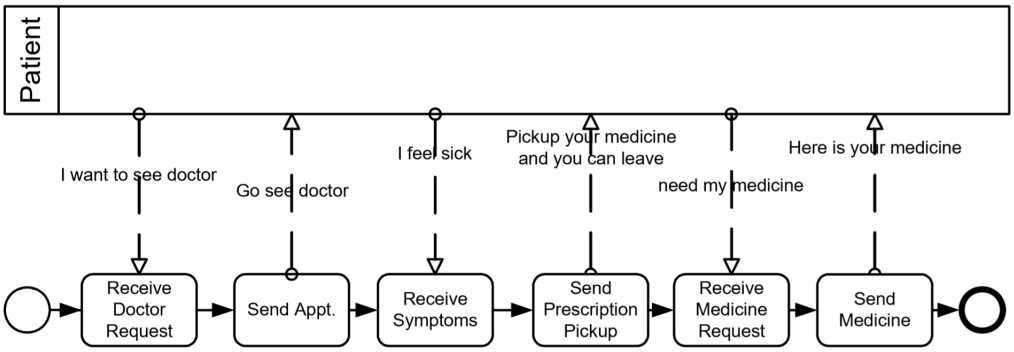


Figura 2.2: Esempio di un Public Business Process

Collaboration

Una Collaboration raffigura le interazioni fra due o più entità di business. Una Collaboration solitamente contiene due o più pools, i quali rappresentano i partecipanti. I messaggi scambiati fra i partecipanti sono rappresentati da Message flow, i quali permettono di connettere due o più pools. La collaborazione rappresenta due o più Public Processes i quali comunicano fra loro. Le Activities delle Collaboration possono essere considerati i punti di contatto che definiscono le interazioni fra i partecipanti al processo. I processi private è probabile abbiamo un maggior numero di attività e quindi un dettaglio maggiore di quanto non venga mostrato nei processi pubblici. Un pool potrebbe anche essere vuoto e trattato come una black-box nel caso non si sapesse nulla dei suoi meccanismi interni.

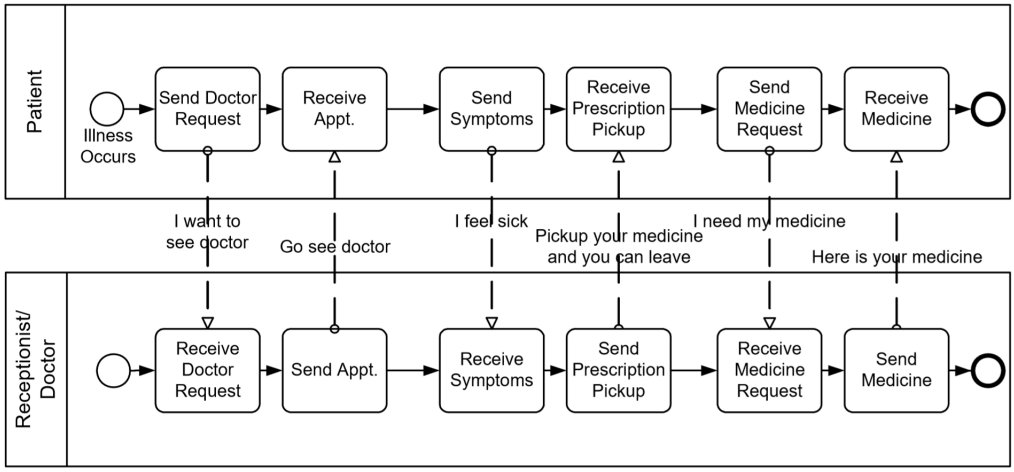


Figura 2.3: Esempio di una collaboration

Choreographies

Una Choreography è la definizione del comportamento atteso di un processo, una sorta di procedura contrattuale, fra i partecipanti. Mentre un processo normale esiste all'interno di un Pool, una coreografia esiste fra pool. La Choreography assomiglia a un Private Business Process dal momento che contiene una rete di Activities, Events, and Gateways.

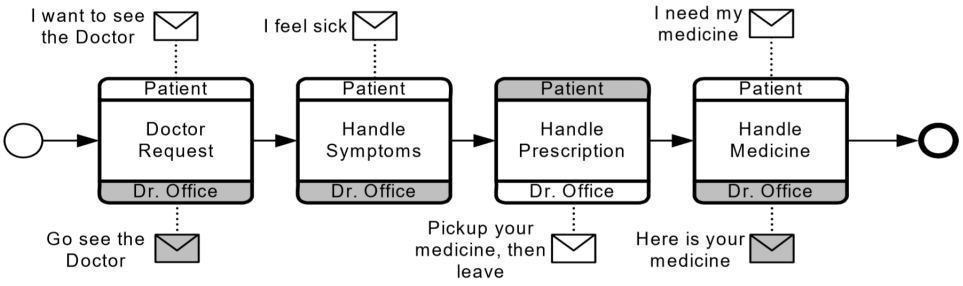


Figura 2.4: Choreography

Ad ogni modo, una coreografia è differente poiché le interazioni fra le attività rappresentano un insieme di scambi di messaggi, i quali involvono 2 o più partecipanti.

2.1.2 Gli elementi BPMN

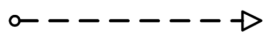
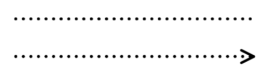
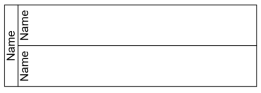
Nel caso di studio analizzato si utilizzeranno come esempi per lo più di processi privati. Si definiscono ora in dettaglio le componenti della notazione *BPMN*.

Tabella 2.1: Elenco degli elementi *BPMN*.

Nome	Descrizione	Notazione Grafica
Event	Un **Event** è qualcosa che accade durante il corso di un processo. Gli eventi hanno effetti sul flusso del processo, hanno una causa o impattano in maniera significativa i risultati. Gli eventi sono cerchi e variano il simbolo interno a seconda del tipo di evento che si vuole scatenare. Ci sono 3 tipi di eventi, a seconda di quando interagiscono nel flow: Start, Intermediate e End.	
Activity	Un'**Activity** è un termine generico per identificare il lavoro che un'azienda svolge in un processo. Un Activity può essere atomica o non-atomica. I Tipi di Activities che sono parte di un modello di processo sono: Sub-Process e Task. I quali sono rappresentati utilizzando rettangoli arrotondati.	
Gateway	Un **Gateway** è utilizzato per controllare la divergenza e la convergenza dei Sequence Flows in un Processo. Questo permette di determinare diverse tipologie di flusso. Dettagli interni al gateway permettono di controllarne il comportamento.	

Continua nella prossima pagina

Tabella 2.1 – Continuazione dalla pagina precedente

Nome	Descrizione	Notazione Grafica
Sequence Flow	Un **Sequence Flow** è utilizzato per mostrare l'ordine che le attività eseguono in un processo.	
Message Flow	Un **Message Flow** è utilizzato per mostrare il flow dei messaggi fra 2 partecipanti. Sono da considerarsi partecipanti in *BPMN*, le entità definite dai Pool in un Collaboration Diagram.	
Associtaion	Un **Association** è utilizzata per connettere informazioni e artefatti con gli elementi grafici di *BPMN*. Le frecce nelle associazioni indicano la direzione nella quale l'informazione si propaga.	
Pool	Un **Pool** è la rappresentazione grafica di un partecipante nella Collaboration. Inoltre agisce come "swimlane" è un contenitore grafico per il partizionamento di un set di Activities dagli altri Pools. Un Pool può avere dettagli interni relativi al processo che lo ospita. Oppure un pool può essere trattato come "black box" quindi senza dettagli interni.	
Lane	Una **Lane** è una sotto partizione all'interno di un Processo. Una lane si può estendere per l'intera lunghezza del processo, o verticalmente o orizzontalmente. Lanes sono utilizzate per organizzare e categorizzare le Activities.	

Continua nella prossima pagina

Tabella 2.1 – Continuazione dalla pagina precedente

Nome	Descrizione	Notazione Grafica
Data Object	I **Data Objects** forniscono informazioni riguardanti le Activities, definendo cosa queste necessitino per essere eseguite o cosa queste producano. I Data Objects possono rappresentare un oggetto singolo o una collezione di oggetti.	
Message	Un **message** è utilizzo per rappresentare i contenuti di una comunicazione fra 2 partecipanti.	
Group	Un **Group** raggruppa diversi elementi dentro la stessa Category. Questo tipo di raggruppamento non ha effetti sulla sequenza del flusso nel processo. Il nome della categoria appare nel diagramma come una label. Le categorie possono essere utilizzate per documentazioni o per fini d'analisi. I Gruppi sono uno dei modi disponibili per dividere in categorie gli oggetti presenti nel processo.	
Text Annotation	Le **Text Annotations** sono un meccanismo per modellare e fornire informazioni aggiuntive per il lettore del diagramma *BPMN*.	

2.1.3 Strumenti per la modellazione *BPMN*

Attualmente vi sono diversi strumenti che possono essere utilizzati per creare modelli di processo. Per questo caso di studio si citano:

- Signavo
- Camunda Modeler

- demo.BPMN.io (Camunda Modeler cloud based)
- lucidchart

Signavo

Signavio è un Process Manager. E' una soluzione cloud-based quindi permette a diversi utenti di poter collaborare a una stessa modellazione di processo in tempo reale. Il Process Manager di Signavio, è componente di una suite più complessa. Utilizzando questo prodotto si ha la possibilità di integrare la modellazione di un processo con altri prodotti Signavo permettendo un'analisi del processo aziendale più accurata e raffinata.

Camunda Modeler

Camunda Modeler è una applicazione desktop per modellare *BPMN* workflows e *DMN* decision models. Questa applicazione non è cloudbase. Anche questo prodotto è componente di una suite più ampia che permette un'analisi del processo aziendale più accurata. Tutti i modelli di test utilizzati per questo caso di studio sono stati prodotti con questo modeler.

BPMN.io (Camunda Modeler cloud based)

Questo Modeler è molto simile al Camunda Modeler utilizzato per la creazione dei *BPMN* in questo caso di studio, se non fosse che tale tool è cloud based ed è reperibile al seguente indirizzo web: *demo.BPMN.io*.

Lucidchart

Lucidchart è un servizio commerciale che permette agli utenti di collaborare insieme in tempo reale creando flowcharts e progetti UML. Fra i vari diagrammi che è possibile progettare utilizzando lucidchart vi è la anche *BPMN*. Una volta ultimato il diagramma è poi possibile esportarlo in diversi formati.

2.2 BPSim

La simulazione è una metodologia sistematica al fine di mimare il comportamento di un modello di processo simulando un business process utilizzando parametri reali, ci si può aspettare di predire con un certo grado di certezza come il processo si comporterà una volta implementato nell'organizzazione. L'esercizio di una simulazione basta su modelli ragionevolmente accurati può aiutare a determinare il numero di attori coinvolti nel processo, i tempi di attesa e di lavoro effettivo, i costi dell'operazione, etc. In questo modo un modello di simulazione può essere utilizzato per rispondere a una varietà di questioni circa un processo corrente una proposta di modifica a un processo esistente. Sebbene sia riconosciuto il valore delle pratiche di Business Process Management (*BPM*), la simulazione e l'analisi di business processes non è ancora sistematicamente utilizzata come strumento al miglioramento dei processi. Questo di fatto può avere diverse ragioni ma di certo la mancanza di uno standard generalmente accettato per la simulazione e l'analisi dei business process è da considerarsi uno dei motivi principali. La specifica *BPSim* dunque introduce il Business Process Simulation (BPSim) framework, uno standard che permette ai modelli di business process definiti in *BPMN* o *XPDL* di essere decorati di informazioni al fine di operare un analisi con metodi rigorosi. Questa specifica definisce la parametrizzazione e lo scambio delle analisi di processi fornendo una struttura e una capacità di analisi dei modelli di processo. Infatti è stata progettata come supporto alle fase di pre-esecuzione e post-esecuzione al fine di ottimizzare i suddetti modelli di processo. Descrive in maniera dettagliata il meta-modello che accompagna, così da facilitare il trasferimento dei dati fra i vari strumenti di simulazione. Il meta-modello introdotto da BPSim è descritto utilizzando *UML* (Unified Modeling Language)(Figura 2.5) mentre il formato di interscambio dati è definito utilizzando XSD (XML Schema Definition). Per supportare sia l'ottimizzazione nella fase di pre-esecuzione sia di post-esecuzione, il meta-modello e il formato di interscambio sono pensati per poter catturare sia gli input che gli output delle analisi di processo.

Uno degli obbiettivi che BPSim si prefigge è essere complementare agli standards già esistenti come Business Process Model and Notation 2.0 (*BPMN*).

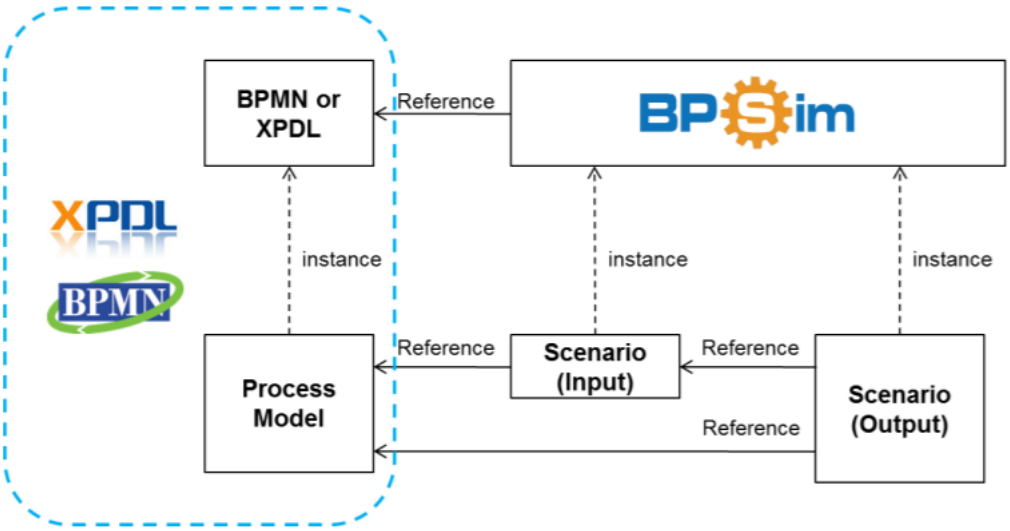


Figura 2.5: Architettura BPSim

2.2.1 Gli elementi BPSim

Gli elementi che compongono la specifica BPSim sono svariati. Si descrivono i principali componenti di BPSim: Scenario e Parameter.

2.2.2 Scenario

Nella BPSim process analysis, tutti i dati sono utilizzati per fornire informazioni complementari ai modelli di processo *BPMN* o *XPDL* al fine di ottimizzare il processo.

Nel caso dello Scenario si forniscono dati che sono messi in relazione con un business process model. Quindi uno stesso buisness process model può avere differenti scenari da testare.

Gli Scenari possono essere utilizzati per catturare:

- Parametri di input
- Risultati dell'analisi
- Dati storici presi da passato. Durante esecuzioni reali del modelli di processo.

Esistendo l'ereditarietà fra scenari uno scenario può ereditare e di conseguenza sovrascrivere un argomento. Quando si eredita da uno scenario, in tal caso bisogna specificare solo gli elementi che si desidera modificare nel corrente scenario. Uno scenario è composto da una collezioni di parametri relativi allo scenario stesso detti **Scenario Parameters** e da una seconda collezione di **Element Parameters**. Ogni Element Parameter di uno scenario si riferisce a uno specifico elemento di un processo dentro il business process model. Sia lo scenario che gli Element Parameters possono essere estesi utilizzando attributi proprietari.

2.2.3 Parameters

Ogni parametro di uno scenario referenzia un elemento specifico del processo dentro il quale è definito lo scenario. Per far si che il principio della separazione delle responsabilità fosse rispettato, element parameters sono divisi in diverse categorie. Ogni categoria raggruppa una collezione di parametri per un comune interesse (tempo, costi, etc.). I valori dei parametri possono essere decorati con la specifica di un calendario che definisce il periodo di applicabilità di tali parametri.

ElementParameters

L'*ElementParameter* class è la classe concreta che definisce tutte le tipologie di parametri.

L'istanza *ElementParameter* referenzia (attraverso l'attributo *elementRef*) un elemento del business process model. Nel caso in cui due Element Parameters referenzino lo stesso elemento, i parametri sono applicati nell'ordine nel quale appaiono nel modello, con conseguente sovrascritture nel caso uno stesso parametro compaia più di una volta.

TimeParameter

La classe *TimeParameter* raggruppa tutti i parametri relativi al tempo per un elemento del processo. Tutti i *TimeParameters* definiscono intervalli temporali e sono e sono definiti da un osservatore esterno al processo.

Il tempo che si impiega fra il completamento di un precedente elemento del processo al completamento dell'elemento corrente viene chiamato *Elapsed Time*. L'*Elapsed Time* può essere scisso in 2 intervalli di interesse che sono la *Duration* e il *Lag Time*. Il *Duration Time* definisce il tempo che va dall'inizio di una determinata attività di lavoro al suo completamento, mentre il *Lag Time* definisce il tempo che dal completamento di una attività all'inizio della successiva.

La *Duration* è la somma dei tempi di *Setup Time*, *Processing Time*, *Validation Time* e *Rework Time*. Formalmente abbiamo:

$$duration = setupTime + processingTime + validationTime + reworkTime \quad (2.1)$$

Nelle situazioni nelle quali questo tipo di granularità non è voluta si può specificare il *Processing Time* come parametro raggruppante queste tempistiche. Mentre il *Lag Time* è la somma di: *Transfer Time*, *Queue Time* e *Wait Time*.

$$lagTime = transferTime + queueTime + waitTime \quad (2.2)$$

In situazioni dove questa granularità non è desiderata si può specificare il *Wait Time* come parametro raggruppante queste tempistiche.

I TimeParameters devono essere obbligatoriamente risolti nei seguenti tipi di parametro: *NumericParameter*, *FloatingParameter* o *DurationParameter*.

Il valore di default di tutti i *TimeParameters* è considerato 0, eccezion fatta per l' elapsedTime che deve essere specificato altrimenti non viene considerato. I parametri temporali sono applicabili ai tipi di elemento che necessitano una collocazione temporale nel modello di processo.

ControlParameters

La classe *ControlParameter* raggruppa tutti i parametri che definiscono il control flow di un elemento del processo. Questa tipologia di parametri permette di definire ad esempio il numero di volte in cui un dato evento deve occorrere all'interno del processo o

quale tempo deve intercorrere fra un evento e il successivo. Questa tipologia di parametro é applicabile solo a certe categorie di elementi.

ResourceParameters

La classe *ResourceParameter* raggruppa tutti i parametri relativi alle risorse di un elemento del processo. Anche in questo caso questa tipologia di parametri sono applicabili solo a certe categorie di elementi.

PriorityParameters

La classe *PriorityParameters* raggruppa tutti i parametri relativi alla priorità di un elemento del processo. Anche in questo caso questa tipologia di parametri sono applicabili solo a certe categorie di elementi.

CostParameters

La classe *CostParameter* raggruppa tutti i parametri relativi ai costi di un elemento del processo. element Anche in questo caso questa tipologia di parametri sono applicabili solo a certe categorie di elementi.

PropertyParameters

La classe *PropertyParameter* raggruppa tutti gli attributi specifici per un dato elemento. Questi attributi sono definiti dagli agenti che modellizzano il processo. Anche in questo caso questa tipologia di parametri sono applicabili solo a certe categorie di elementi.

2.2.4 BPM Simulation Tools

Eseguendo una simulazione, si può aumentare il livello di confidenza e anche assicurare ad altre organizzazione che i cambiamenti proposti sono ragionevoli. L'idea base

è creare un modello software di un processo, ed eseguire le istanze di processo. Nella simulazione, l'utente deve specificare diversi parametri come il tasso degli arrivi, i tasso con il quale viene soddisfatta una determinata tipologia di task, le risorse disponibili, la capacità delle risorse, i costi orari per ogni risorsa, etc. Esistono applicazione che permettono di fare ciò. Nella sottosezione che segue si presenta uno di questi applicativi e se ne descrive l'utilizzo.

BIMP, Esempio di Simulation Tool

Ora si descrive un applicazione di BPM Simulation già esistente chiamata **BIMP**. La quale utilizza BPSim al fine di eseguire una simulazione sul Business Process Model definito. Tale applicazione è uno strumento web-based che può leggere un modello *BPMN* creato utilizzando un qualsiasi strumento di modellazione (Signavo, Camunda Modeler). Dopodiché si possono specificare certi parametri relativi ai tempi di servizio per ogni task la disponibilità delle risorse, etc. Infine si può eseguire il modello per un certo numero d'istanze di processo e ottenere i risultati della simulazione. Questo strumento permette di simulare business processes in scenari reali di grande scala. E' stato progettato e implementato da Marlon Dumas e i suoi colleghi dell'università di Tartu. Lo screen principale dell'applicazione di simulazione BIMP è organizzato in sezioni (Figura 2.6).

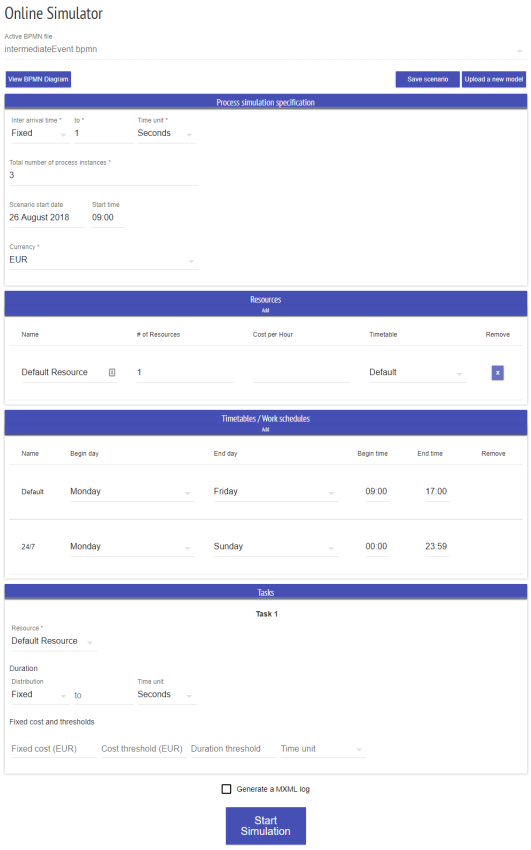


Figura 2.6: BIMP Home Screen

Nella sezione in alto, l'utente può specificare il tempo che intercorre fra l'arrivo delle istanze a seconda della distribuzione selezionata: Fixed, Normal, Exponential, Uniform, Triangular, Log-Normal, Gamma e Histogram .

Nella sezione successiva, si possono specificare i tipi di risorsa, il numero di risorse per ogni tipo disponibile e il loro costo orario.

E' anche possibile definire i tempi di lavoro e la loro programmazione. La sezione dei task permette di fornire alcune informazioni per ogni singolo task. Questi parametri consistono: nella durata, che può essere definita tramite una distribuzione, e in altri valori come costi, tipologia di risorse consumate, etc. L'ultima sezione è per i gateways. L'utente è chiamato a specificare le probabilità di ogni percorso in uscita per gli XOR gateway.

Una volta eseguita la simulazione è possibile apprezzare i risultati prodotti (Figura 2.7).

Figura 2.7: Risultati simulazione BIMP

I risultati prodotti mostrano:

- I risultati dei cicli di processi
- I tempi di attesa e i costi dei processi
- In aggiunta viene mostrato l'utilizzo delle risorse fornite con il dettaglio per istanza di processo e singola attività

Capitolo 3

Firefly Simulator

In questo capitolo si presentano le specifiche per lo sviluppo di un applicativo di simulazione dei processi aziendali. Si definiranno gli obbiettivi che tale software dovrà andare a soddisfare. Dopodiché si andranno a definire i prodotti (sviluppati da terzi) con i quali andrà a interagire. Successivamente si elencheranno i vincoli stabiliti per questa tipologia di applicativo definendo parametri di input e output.

Infine verranno descritte le categorie di utenti per le quale l'applicativo è stato pensato e come questi ci si interfacceranno.

3.1 Obbiettivi

Il software realizzato prende il nome di **Firefly Simulator**. Tale applicativo è un simulatore di processi aziendali. Dato un modello di processo definito utilizzando la specifica *BPMN* e decorato utilizzando la specifica *BPSim* è possibile eseguire una simulazione del processo apprezzando successivamente i risultati prodotti. L'applicativo deve dare la possibilità all'utente di poter eseguire una simulazione su un qualsiasi modello di processo; permettendo inoltre di specificare tutti i possibili parametri definiti dalla specifica *BPSim* per ogni elemento che compone il modello di processo. L'interazione fra software e utente avviene tramite l'utilizzo di una *command line interface*.

3.2 Interazione con prodotti sviluppati da terze parti

L'applicativo in questione dovrà utilizzare *Camunda* come engine di simulazione. *Camunda* è una soluzione open source che permette la simulazioni di modelli di processo.

3.2.1 Camunda

Camunda è un framework sviluppato in Java che supporta le specifiche *BPMN*, *CMMN* e *DMN*. A seconda delle necessità permette di automatizzare i processi ed eventualmente orchestrali.

L'architettura di *Camunda* prevede diverse componenti (Figura 3.1). In questo caso specifico l'applicativo **Firefly Simulator** andrà ad interagire solo con un componete della suite offerta da *Camunda*: il **Process Engine**

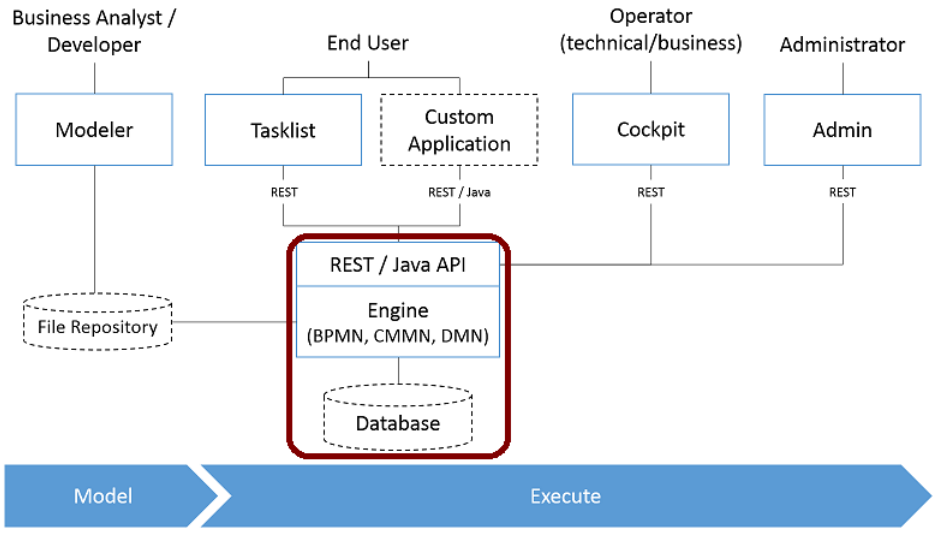


Figura 3.1: Architettura di Camunda

Camunda Engine

Il meccanismo più semplice per interagire con il *Camunda Engine* è tramite l'utilizzo della Java API messa a disposizione da *Camunda*. Il punto centrale di *Camunda* è il

ProcessEngine. Il *Process Engine* può essere configurato in diversi modi:

- Programmaticamente utilizzando la Java API
- Definendo la configurazione utilizzando file XML
- Utilizzando il framework *Spring*

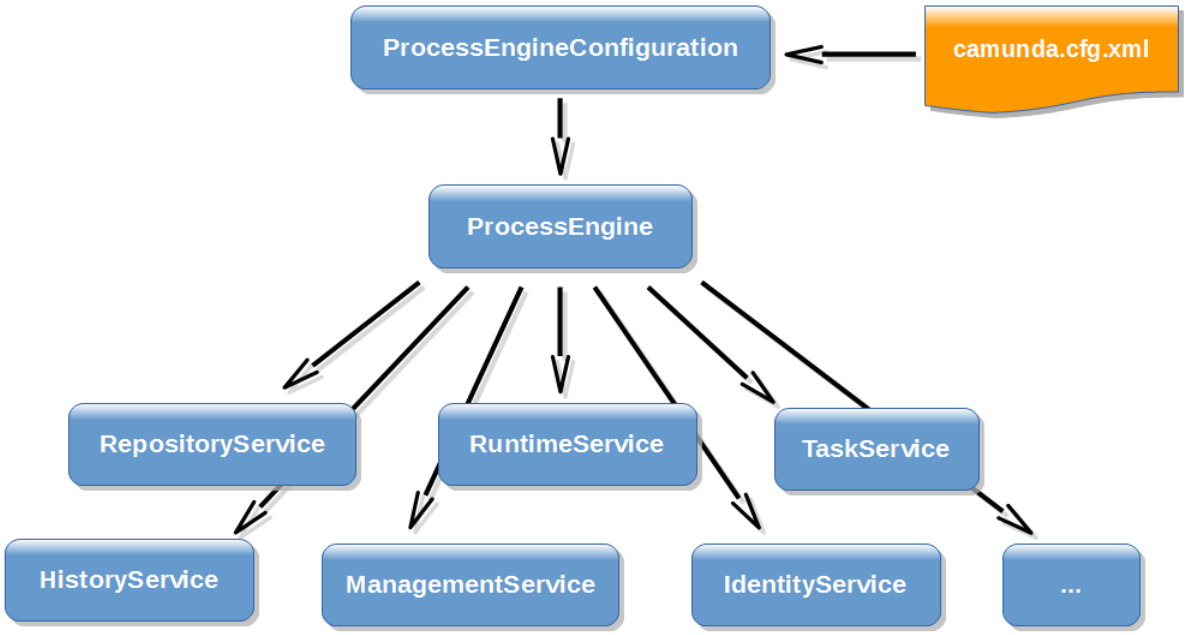


Figura 3.2: Architettura Camunda Engine

Per quanto concerne il caso di studio sottoposto la configurazione avviene utilizzando la Java API.

Una Volta creato il *ProcessEngine*, è possibile ottenere diversi servizi con i quali è possibile eseguire e gestire l'esecuzione del processo. E' bene ricordare che sia il *ProcessEngine* che i servizi da questo messi a disposizione sono *thread safe*, quindi anche se vengono inizializzate diverse istanze della stessa classe, le informazioni alle quali questi avranno accesso saranno le medesime.

Si definisce il concetto di *ExecutionListener* utilizzato per l'implementazione del ciclo di simulazione. L'*ExecutionListener* non è altro che una Callback Interface che notifica agli oggetti della classe che la implementa quando, ad esempio: una data istanza di processo viene presa in carico dal engine o quando una data attività all'interno del modello di processo svolge una determinata transazione di stato.

Di notevole importanza sono i due *service* principali resi disponibili dalla classe *ProcessEngine*: *RepositoryService* e *RuntimeService*

RepositoryService: il *RepositoryService* è il primo oggetto necessario per interagire con il *Camunda Engine*. Questo servizio permette di eseguire operazioni per gestire e manipolare le definizione dei processi che si stanno eseguendo. Sostanzialmente la definizione di un processo è da considerasi la controparte Java di un processo definito utilizzando la specifica *BPMN 2.0*. Infatti vuole rappresentare la struttura e il comportamento di ogni singolo passo del processo definito con la specifica *BPMN 2.0*.

Il *RepositoryService* permette di prendere tali artefatti e renderli processabili dall'engine. Con processabili si intende che tutti i processi caricati nell'engine vengono ispezionati e analizzati dopodiché vengono salvati nel database. Una volta fatto ciò tutti i processi possono essere eseguiti.

Utilizzando il *RepositoryService* è possibile eseguire le seguenti operazioni:

- Eseguire query per interrogare i processi in pancia all'engine.
- Sospendere e attivare i processi caricati nell'engine.
- Accedere a risorse o diagrammi di processi che sono stati automaticamente generati dall'engine.

RuntimeService: Il *RuntimeService* è incaricato di eseguire nuove *istanze di processo* derviate dalle *definizioni di processo*. La *definizone di un processo* è intesa come l'insieme della struttura e dei comportamenti per ogni singolo step nel processo. Mentre una *istanza di processo* è l'esecuzione di una *definizione di processo*.

Per una *definizione di processo* ci sono tipicamente molte istanze di processo che sono in esecuzione in un dato istante.

Il *RuntimeService* è responsabile per la gestione delle variabili di processo. Tali variabili sono associate con un istanza di processo e possono essere utilizzate nel processo come discriminante ad esempio negli exclusive gateway.

Inoltre il *RuntimeService* permette di interrogare le istanze di processo in esecuzione per comprenderne lo stato. Le esecuzioni di fatto rappresentano il concetto di token definito per la gestione del flusso nei diagrammi definiti dalla specifica *BPMN 2.0*.

Sostanzialmente una esecuzione non è altro che un puntatore che referenzia lo stato di una determinata istanza di processo.

Infine il *RuntimeService* è utilizzato ogni qual volta una istanza di processo sta attendendo una interazione dall'esterno e il processo necessita di continuare la sua esecuzione.

Un istanza di processo può avere diversi stati di attesa, di conseguenza questa servizio possiede diverse operazioni per segnalare che l'interazione dall'esterno è terminata è l'istanza di processo può essere fatta continuare.

3.3 Requisiti tecnici

L'applicazione può eseguire una simulazione *BPM* alla volta. I risultati devono essere salvati in un documento specificato dall'utente. Una volta terminata la simulazione e salvati i risultati l'applicativo può terminare. L'utente può specificare il numero di istanze di processo che vuole eseguire e a quali intervalli di tempo queste devono essere eseguite.

L'utente deve poter operare utilizzando una linea di comando di conseguenza non è richiesta alcuna interfaccia.

Creato il *BPMN* che definisce il processo e decorato con le informazioni *BPSim*, successivamente questo viene inviato al *Camunda Engine* per essere processato.

In un primo momento si devono estrarre le informazioni dal documento *BPSim*. Successivamente si esegue la simulazione. L'*engine BPMN Camunda* deve utilizzare le informazioni *BPSim* quando necessario al fine di catturare la complessità di certi fenomeni nel processo produttivo che si sta simulando. Infine una volta prodotti i risultati questi dovranno essere salvati per permetterne la consultazione.

3.3.1 Parametri di input

Si definiscono i parametri di input del programma. I parametri di input sono quattro:

- Modello di processo
- Numero di istanze di processo
- *InterTriggerTime* fra istanze di processo
- id del processo

Segue una descrizione nel dettaglio dei parametri di input.

Modello di processo

Dato un modello di processo definito tramite la specifica *BPMN 2.0* vi è un nodo che definisce i parametri *BPSim* della simulazione. I dati relativi alla simulazione dovranno essere processati e salvati al fine di renderli fruibili durante l'esecuzione della simulazione. Le informazioni relative al *BPMN* dovranno rispettare la specifica *formal/11-01-03 BPMN 2.0* rilasciata da *OMG*. Mentre per quanto riguarda *BPSim* le informazioni dovranno essere in linea con i dettagli forniti dalla specifica *WFMC-BPSWG-2016-1*.

Il modello di processo potrà utilizzare di tutte le opzioni previste dalla specifica *BPSim*.

Ogni elemento *BPMN* potrà avere determinati parametri *BPSim* ad esso associati. Nella fattispecie la tabella 3.1 definisce per ogni notazione *BPMN* i parametri *BPSim* a questo attribuibili. Poiché non sempre è possibile associare determinati parametri a certi elementi *BPMN*.

Tabella 3.1: Griglia incrociata parametri *BPSim* e costrutti *BPMN*.

		Time Parameters					
Category	Element Name	Queue Time	Wait Time	Setup Time	Processing Time	Validation Time	Rework Time
Events	Start Event Event End Event						
Activities	Task	x	x	x	x	x	x
	Sub Process	x	x	x	x	x	x
	Transaction	x	x	x	x	x	x
	Call Activity	x	x	x	x	x	x
	Event Sub Process	x	x	x	x	x	x
Gateways	Gateway						
Connecting Objects	Sequence Flow Message Flow Data Association Association						
Data	Data Object Data Store						
Seimlanes	Lane Pool						
Artifacts	Artifacts						
Arttributes	ResourceRole Resource						

		Control Parameters				Resource Parameters			
Category	Element Name	InterTriggerTimer	Trigger Count	Probability	Condition	Availability	Quantity	Role	Selection
Events	Start Event	x	x	x	x				
	Event	x		x	x				
	End Event								
Activities	Task	x	x						
	Sub Process	x	x						
	Transaction	x	x						
	Call Activity	x	x						
	Event Sub Process	x	x	x	x				
Gateways	Gateway	x	x						
Connecting Objects	Sequence Flow			x	x				
	Message Flow								
	Data Association								
	Association								
Data	Data Object								
	Data Store								
Seimlanes	Lane								
	Pool								
Artifacts	Artifacts								
Arttributes	ResourceRole								x
	Resource					x	x	x	

		Cost Parameters	
Category	Element Name	Fixed Cost	Unit Cost
Events	Start Event		
	Event		
	End Event		
Activities	Task	x	x
	Sub Process	x	x
	Transaction	x	x
	Call Activity	x	x
	Event Sub Process	x	
Gateways	Gateway		
Connecting Objects	Sequence Flow		
	Message Flow		
	Data Association		
	Association		
Data	Data Object		
	Data Store		
Swimlanes	Lane		
	Pool		
Artifacts	Artifacts		
Arttributes	ResourceRole		
	Resource	x	x

Numero di istanze di processo

Ogni qual volta si vuole eseguire la simulazione è possibile selezionare il numero di istanze di processo da eseguire. Questo permette di eseguire diverse istanze di processo data una singola definizione di processo.

Process Id

E' inoltre necessario specificare l'id del processo definito nel *BPMN Modeler*. Questo è un parametro di input necessario per poter utilizzare il *Camunda Engine*.

InterTriggerTime fra le istanze

E' inoltre possibile selezionare un intervallo che intercorre fra l'inizio delle varie istanze. Che prende il nome di *interTriggerTime*. Di default tale parametro è 0.

3.3.2 Parametri di Output

L'output atteso è un file xml contenente tutte le informazioni relative ai risultati. I record dei risultati devono contenere il dettaglio per ogni istanza di processo eseguita, per quanto concerne i dati prodotti si vuole tenere traccia dei costi e del tempo per ogni singola attività eseguita in ogni istanza di processo.

3.3.3 Tipologia di utenti

L'applicativo creato è stato pensato per utenti con una buona conoscenza delle specifiche *BPMN* e *BPSim*. L'idea è che questo software venga utilizzato come engine per la simulazione di processi definiti utilizzando *BPMN* e *BPSim*. Infatti l'esecuzione della simulazione e la produzione dei relativi risultati tramite opportuni standard permette a un qualsiasi altro applicativo di interfacciarsi e utilizzare le informazioni prodotte per ulteriori analisi e relative visualizzazioni grafiche.

3.3.4 Interfaccia

L'applicazione in questione vuole avere un interfaccia da linea di comando. Deve essere quindi possibile eseguire il programma specificando i parametri in input nel terminale. Non è quindi richiesto implementare alcuna user interface.

Capitolo 4

Architettura e Implementazione

Si è deciso di sviluppare tale progetto utilizzando il linguaggio di programmazione **Java**. Tutte le versioni del progetto sono compatibili con la *Java version 1.8.0.151*. Come gestore di pacchetti si è utilizzato *maven 3.5.2*, tutte le dipendenze al progetto possono essere trovate all'interno del *pom.xml* file, nella root del progetto.

4.1 Architettura del progetto

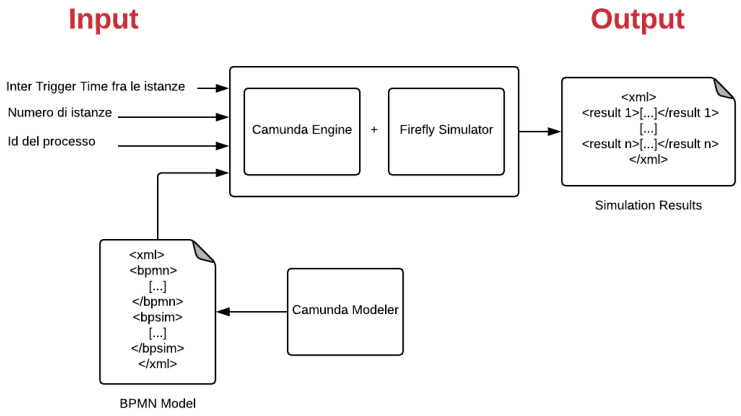


Figura 4.1: Architettura dell'applicazione Firefly

Il progetto in questione prevede una volta creato il *BPMN Model*, con il *Camunda Modeler* e decorato con i dettagli *BPSim*, che questo venga dato come input all'applicativo insieme all'id del processo al numero di istanze che si vogliono eseguire ed infine all'*interTriggerTime* fra le istanze. Una volta fatto ciò l'applicativo con l'aiuto del *Camunda Engine* esegue la simulazione e genera l'output che non è altro che un documento xml contenente i risultati prodotti (*Figura 4.1*).

4.2 Sequenza della simulazione

Una volta definito il processo e immessi i dati nell'applicazione avvengono una serie di operazioni al fine di preparare e completare la simulazione. Il diagramma di sequenza in *Figura 4.2* mostra l'ordine con il quale avvengono tali procedure.

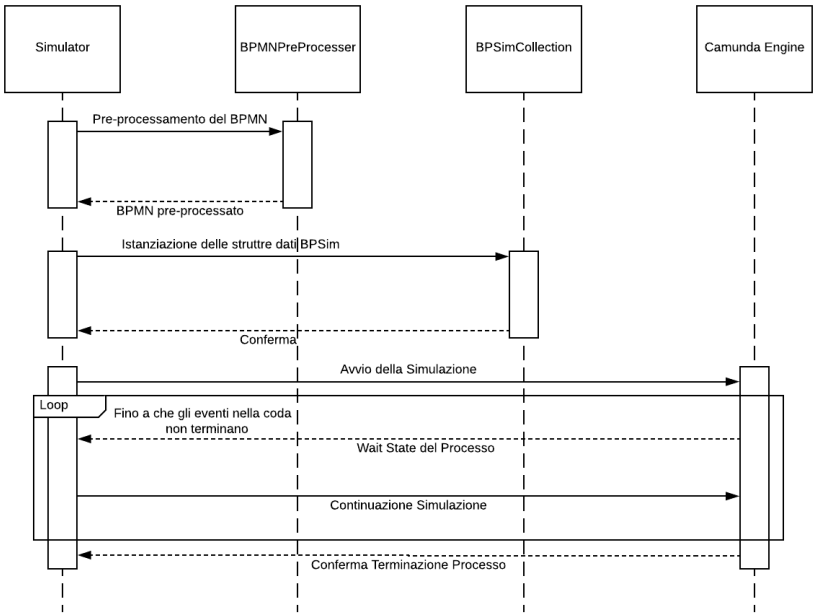


Figura 4.2: Diagramma di sequenza Simulazione Firefly

In una prima fase vi è un pre-processamento del *BPMN* il quale mira ad organizzare le informazioni all'interno del documento *BPMN* per meglio eseguire la simulazione.

Dopodichè vengono salvate in apposite strutture dati le informazioni definite utilizzando la specifica *BPSim* per poter poi essere riutilizzate durante la simulazione. Dopodichè viene avviato il ciclo di simulazione. Tale procedura non fa altro che utilizzare il *Camunda Engine* per eseguire la simulazione. Ad ogni step della simulazione, l'*Engine* ritorna il controllo al ciclo il quale effettua gli aggiornamenti necessari per poi produrre i risultati. Tutte le fasi che sono state descritte vengono ora viste nel dettaglio nelle sotto sezioni successive.

4.2.1 Pre-processamento del file BPMN

Come prima cosa l'applicativo legge il documento *BPMN*. Dopodiché è necessario pre-processare il file *BPMN*, apportando alcune modifiche. Tali modifiche rendono possibile l'esecuzione della simulazione. Nella specifica *BPMN 2.0* si è definito lo xml schema per gli elementi *BPMN*. Questo permette di modificare il *BPMN* trattandolo come un file xml. Le operazioni che avvengono nella fase di pre-processazione sono le seguenti:

- Conversione dei *task* in *user task*
- Aggiunta dei listener negli *user task*
- Conversione dei *Boundary Events* in *Message Boundary Events*
- Sostituzione delle espressioni di condizione

4.2.2 Conversione dei task in user task

Tutti i task presenti nel modello di processo vengono convertiti in User Tasks. Questo avviene perché gli User Task forzano il *Camunda Engine* a interrompere la simulazione e a rimandare il controllo al chiamante poiché si è in attesa di un input dell'utente. In questo caso la figura dello user è fittizia. Si utilizza questa peculiarità di *Camunda* per riprendere il comando del ciclo di simulazione e aggiornare il tempo di simulazione.

4.2.3 Aggiunta dei listener negli user task

Vengono aggiunti i listener ai task appena convertiti. I listeners permettono di eseguire del codice nel momento in cui un determinato componente del processo passa una transizione di stato durante l'esecuzione del processo. Sono stati creati diversi listeners per ogni singola componente *BPMN*, il loro funzionamento verrà precisato successivamente.

4.2.4 Conversione dei Boundary Events in Message Boundary Events

Si convertono tutti i Boundary Events in Boundary Message Events. Questo avviene per semplificare la loro attivazione durante la simulazione. Infatti ogni evento che si scatena nel modello di processo viene simulato come un messaggio che viene ricevuto durante l'esecuzione del processo. Questo dettaglio però non è percepibile all'utente che utilizza l'applicativo.

4.2.5 Sostituzione delle espressioni di condizione

Avviene la conversione delle espressioni di condizione cioè tutti quei nodi con nome *conditionExpression* vengono estratti e viene iniettato del codice **Groovy**.

```
// Groovy Code iniettato nel BPMN file
import util.FireflyUtil
// Con SequenceFlowId si intende l'id dell'elemento
// contenente l'espressione di condizione
FireflyUtil.booleanValueFlow("SequenceFlowId")
```

Quello che deve fare tale codice è restituire 1 nel caso in cui il branch nel quale è stata definita la condizione necessiti di essere eseguito, al contrario 0. Tale valore potrebbe essere specificato da una determinata funzione di probabilità definita nel nodo *BPSim*. Perciò il codice non fa altro che richiamare una funzione del di *Firefly Simulator*. Tale funzione ricerca se vi è una regola nel BPSim node che definisca il comportamento del *Gateway* al quale il *SequenceFlow* è connesso. E ritorna 1 o 0 a seconda del comportamento specificato nel nodo *BPSim*.

4.2.6 BPSim Parsing

Una volta pre-processato il *BPMN* è necessario collezionare tutte le informazioni relative a *BPSim* per fare ciò all'interno dell'applicazione è stato creato una classe chiamata *BPSimCollection*. Tale classe ha la responsabilità di riorganizzare le informazioni presenti nel nodo *BPSim* presente nel file *BPMN* al fine che siano utilizzabili durante la simulazione. In primo luogo è stato necessario convertire lo schema xml che definisce la specifica *BPSim* in classi che permettessero la creazione di oggetti accessibili a runtime.

Il *BPSim* schema definisce 43 oggetti ed eventuali attributi. Si è quindi creata la rispettiva classe per ogni singolo oggetto definito dalla specifica. Dato che gli elementi *BPSim* sono definiti tramite l'*xml schema defintion (xsd)* si è quindi utilizzato lo strumento *JAXB*. Tale tool da linea di comando permette di creare in maniera automatica le corrispettive classi Java partendo da da una loro definizione in *xsd*.

Sostanzialmente viene fatto un *unmarshaling* del nodo xml, utilizzando la JAXB library. L'*unmarshaling* è il termine con il quale si identifica la trasformazione di un oggetto con fini di salvataggio o trasmissione a una sua rappresentazione che sia eseguibile. Questo permette di avere a runtime un oggetto *BPSim* il quale rappresenta in tutto e per tutto il nodo *BPSim* contenuto dentro il file *BPMN*.

Questo ha permesso di creare un package chiamato *BPSim* nel quale sono contenute tutte le rappresentazioni degli elementi *BPSim*.

La struttura derivante dal *BPSim* schema fa si che la classe Java che rappresenta le definizioni *BPSim* sia molto complessa. Infatti per accedere a un parametro solitamente è necessario raggiungere 4 o 5 livelli di annidamento. Per questo motivo si è deciso di creare delle classi *wrapper* che rappresentassero in tutto e per tutto gli oggetti *BPSim* di maggior rilevanza per la simulazione; dopodiché si sono popolati tali oggetti per semplificare la lettura del codice.

Durante la fase di lettura del nodo *BPSim* a seconda degli elementi presenti vengono invocate le classi definite tramite il tool *JAXB* per permetterne la lettura e poi i dati in esse contenute vengono trasferite dentro gli oggetti *wrapper*.

Si hanno 4 strutture dati che si espongono all'esterno. Queste sono collezioni degli oggetti wrapper precedentemente creati:

- *indipendentIntermediateThrowEvents*

- *boundaryEvents*
- *taskObjects*
- *resourcesElementsAvaliability*

Queste strtture dati sono poi utilizzate all'interno della simulazione così da poter ottenere le informazioni relative agli oggetti attivati dal token.

In una prima fase si leggono tutti i parametri relativi allo scenario della simulazione. Dopodiché si leggono tutti gli oggetti relativi alle attività presenti creando un *hashMap*. Successivamente si leggono i *boundaryEvents* e gli *intermediateEvents*. Infine si crea un *hashMap* relativa alle risorse disponibili per la simulazione.

4.2.7 Simulazione

Una volta letti tutti i dati e create le strutture apposite. La simulazione può essere avviata. Per gestire la simulazione si è creato un package apposito chiamato *simulation*. Gli elementi al suo interno sono qui elencati:

- EventsHandler
- EventsQueue
- SimulationBoundaryEvent
- SimulationCatchEvent
- SimulationClock
- SimulationCosts
- SimulationResource
- SimulationEventsComparator
- SimulationEvent, il quale ha le seguente sottoclassi:
  - SimulationIntermediateEvent

- SimulationStartEvent
- SimulationTaskEvent

Segue un diagramma UML che descrive le interazioni fra questi oggetti.

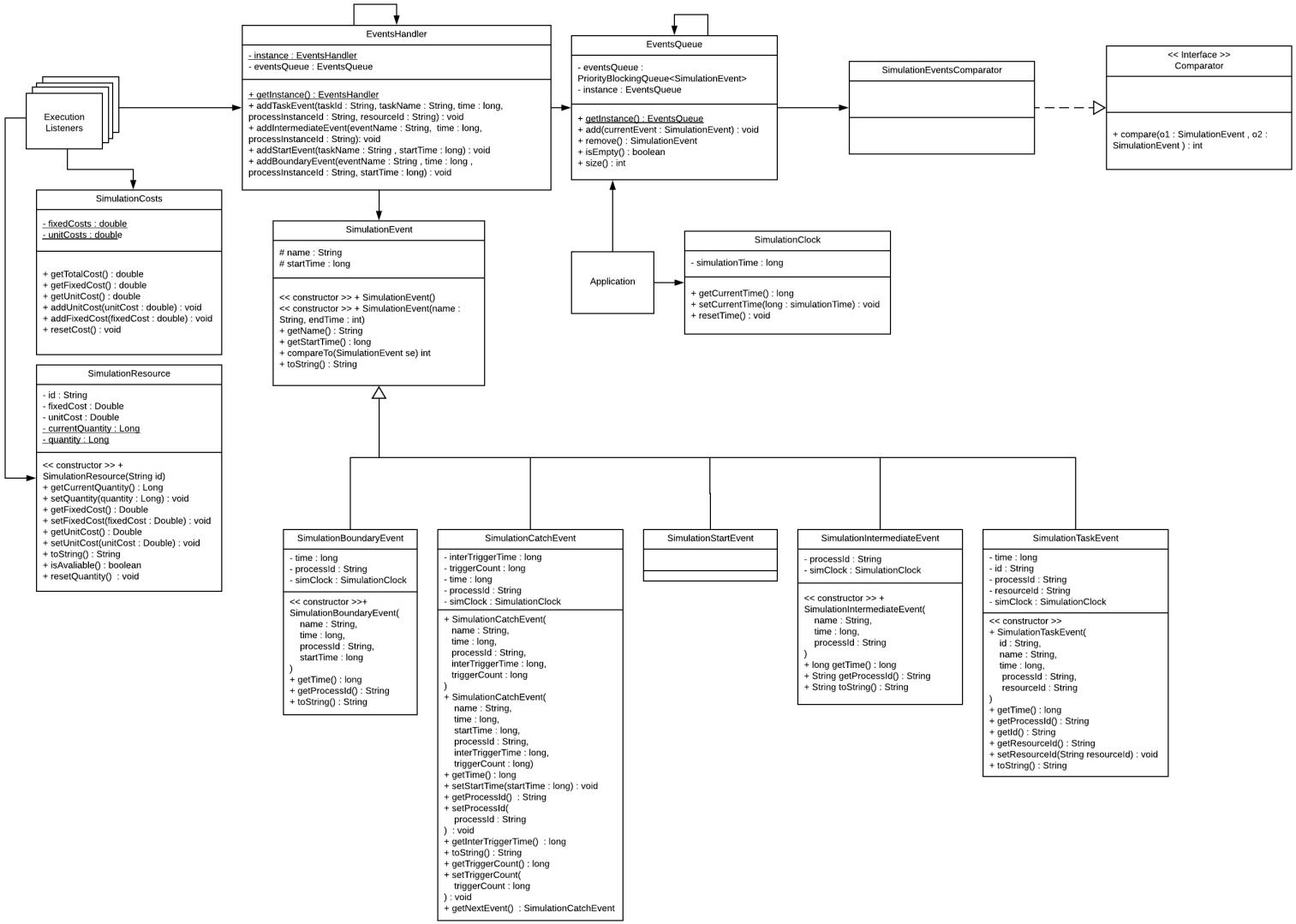


Figura 4.3: Digramma UML del package Simulation

All'interno della classe *Application* vi è un ciclo di simulazione. Il ciclo non fa altro che verificare la presenza di eventi all'interno della coda degli eventi e consumare tali eventi. In base alla tipologia di evento diverse operazioni vengono eseguite. Si descrivono in dettaglio le responsabilità per le singole classi.

SimulationResource

Per ogni tipologia di risorsa definita nel *BPSim* allegato al modello di progetto vi è una istanza della classe *SimulationResource*. Ogni istanza quindi contiene tutte le informazioni relative a una data risorsa e i tempi di rilascio per ogni risorsa disponibile. Il salvataggio dei tempi di rilascio è implementato utilizzando una *PriorityQueue*. La coda fa si che si possano salvare i tempi di rilascio delle risorse in maniera ordinata. Così che ogni qual volta sia necessario accedere al tempo di rilascio si possa accedere alla testa della coda ottenendo sempre il tempo di rilascio più datato. Poiché si vuole sempre utilizzare la risorsa che per prima si libera.

EventsQueue

La coda degli eventi, definita con il nome *EventsQueue*, è implementata utilizzando *PriorityBlockingQueue*. Essa contiene oggetti della classe *SimulationEvent*. Per questa tipologia di oggetti è stata scritta la classe *SimulationEventsComparator* la quale implementa l'interfaccia comparator. Questo ha permesso di definire il metodo di comparazione fra gli oggetti di tipo *SimulationEvent*. La discriminante che ci permette di asserire se un evento è maggiore o inferiore a un altro è il tempo di simulazione di creazione dell'evento (Figura 4.4).

Questo meccanismo di comprazione è stato necessario poiché si vuole che gli eventi una volta inseriti nella coda mantengano un ordine che sia in linea con il loro avvenimento all'interno della simulazione.

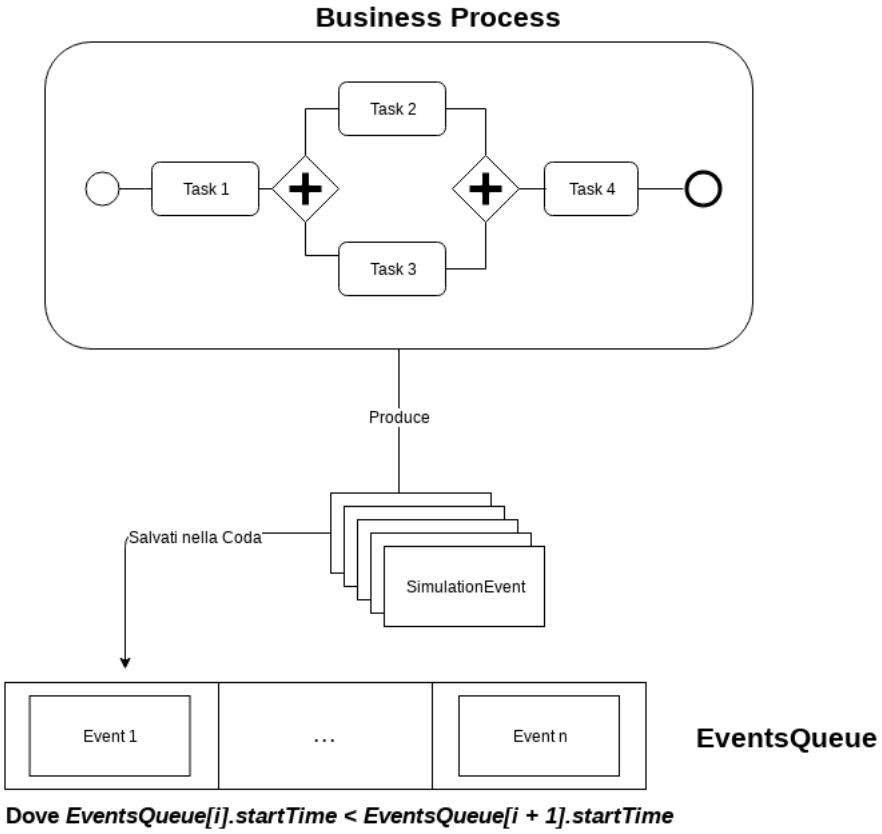


Figura 4.4: Coda di priorità

SimulationStartEvent

Ogni simulazione inizia con un evento di start. Gli eventi di start vengono creati durante la fase di inizializzazione. Una volta creato l'evento di start (*simulationStartEvent*) questo viene messa nella coda degli eventi. Quando il ciclo di simulazione trova un evento di start all'interno della coda esegue le seguenti operazioni:

- Viene avviato una nuova istanza di processo
- Vengono caricati tutti gli eventi intermedi definiti per tale processo all'interno della coda

SimulationTaskEvent

Nel caso degli eventi relativi ai task le operazioni eseguite sono:

- Caricamento della risorsa che deve svolgere il task corrente
- Nel caso in cui la risorsa sia disponibile o non sia definita, viene aggiornato il *simulationClock* e si termina l'attività.
- Al contrario se la risorsa non è disponibili perché impegnata nell'eseguire un'altra operazione in tal caso è necessario fare delle valutazioni che vanno a inficiare sull'aggiornamento delle risorse (tali valutazioni si discuteranno in seguito). Infine si aggiorna il *simulationClock* dopodiché si termina l'attività.

L'aggiornamento del *simulationClock* può variare a seconda delle operazioni richieste. Questa parte verrà descritta in maggior dettaglio successivamente.

SimulationCatchEvent

Nel caso in cui si estragga dalla coda degli eventi un *catchEvent*, invece vengono svolte le seguenti operazioni:

- Viene aggiornato il clock
- Viene avviato un nuovo token che ha come evento d'orgine il *catchEvent* corrente.

SimulationBoundaryEvent

Quando si ha un BoundaryEvent si eseguono le seguenti operazioni:

- Viene aggiornato il clock
- Viene avviato un nuovo token che ha come evento d'orgine il *boundaryEvent* corrente.

ExecutionListener

La creazione degli eventi avviene nelle classi che implementano l'interfaccia *executionListener*. Tali classi sono responsabili nel creare gli eventi che andranno poi inseriti all'interno della coda degli eventi. Queste classi quindi repriscono tutte le informazione relative all'elemento BPMN al quale sono associate per poi creare l'evento da mettere nella coda.

Queste classi sono contenuti all'interno del pacchetto *executionlistner*.

Nella fattispecie l'utente non deve essere a conoscenza di tali meccanismi poiché l'associazione degli execution listener viene fatta automaticamente durante la fase di pre-processamento del file BPMN.

SimulationClock

Il *simulationClock* è la classe che permette la gestione del tempo all'interno della simulazione. Tale classe non fa altro che esporre delle operazioni per aggiornare il tempo globale della simulazione. Come viene aggiornato il tempo è responsabilità del ciclo di simulazione contenuto dentro la classe *FireflySimulatorApp*.

Seguono degli esempi per meglio spiegare come viene aggiornato il tempo all'interno del ciclo di simulazione a seconda dei casi:

Dato un BPMN come in figura.

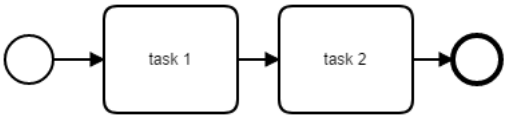


Figura 4.5: *SimulationClock* in processo con attività lineari

Il tempo non è altro che la somma dei tempi di esecuzione delle singole attività

$$tempototale = tempoTask1 + tempoTask2 \tag{4.1}$$

Dato un *BPMN* come in figura:

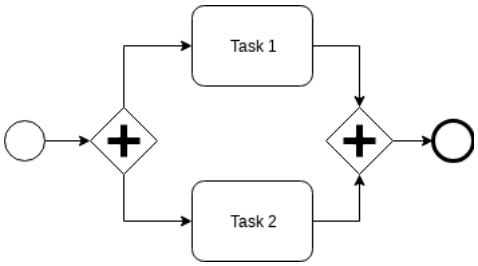


Figura 4.6: *SimulationClock* con attività parallele

In questo caso si hanno alcune attività che devono essere svolte in parallelo in tal caso se una delle due attività richiede un tempo maggiore dell'altra il tempo della simulazione una volta raggiunto l'inclusive *gateway* sarà quindi uguale al tempo maggiore fra i task svolti.

$$tempototale = max(tempoTask1, tempoTask2) \tag{4.2}$$

Dato un *BPMN* come in figura e **2 risorse disponibili** in grado di svolgere i task presenti.

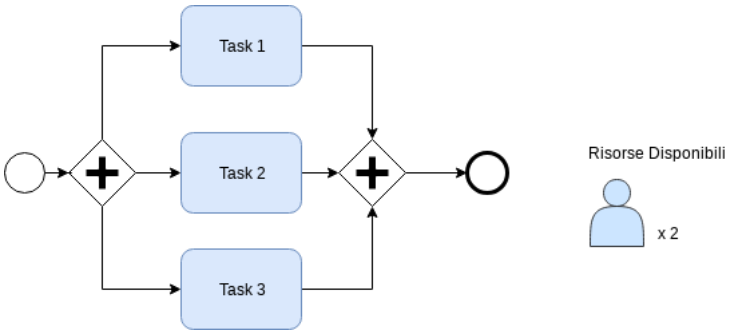


Figura 4.7: *SimulationClock* con attività parallele e risorse

Il tempo sarà uguale al massimo fra i tempi dei primi due task più il tempo richiesto dal terzo task

$$tempototale = max(tempoTask1, tempoTask2) + tempoTask3 \tag{4.3}$$

E' quindi stato necessario creare un meccanismo per gestire l'esecuzione concorrente di tali task ma che si risolve in un semplice computazione single thread dovuto al fatto che tutti i task sono stati convertiti in *userTask*, permettendo quindi che ad ogni esecuzione di un determinato task il controllo viene rimandato al ciclo di simulazione che può effettuare le opportune valutazioni.

4.3 Algoritmo di gestione della coda degli eventi

In questa sezione si descrive nel dettaglio l'algoritmo di gestione della coda degli eventi. Presentando opportuni esempi per meglio spiegare le variabili utilizzate e le operazioni effettuate. Le tipologie di eventi gestite all'interno del ciclo di simulazione sono le seguenti:

- SimulationIntermediateEvent
- SimulationStartEvent
- SimulationTaskEvent

4.3.1 SimulationIntermediateEvent

Gli eventi intermedi sono eventi definiti all'interno del *BPMN*, i quali vengono preprocessati e caricati all'interno della coda. Questi, una volta estratti dalla coda, generano nuovi token ma di fatto non inficiano sul tempo di simulazione. Ogni qual volta un token viene creato come conseguenza di un evento intermedio le attività a esso collegate vengono eseguite, variando quindi conseguentemente il tempo di simulazione.

4.3.2 SimulationStartEvent

Il *SimulationStartEvent* è un evento che permette la creazione delle istanze di processo ogni qual volta questo evento viene estratto dalla coda e una nuova istanza di processo viene avvitata. Questo implica che anche gli eventi intermedi raccolti durante la fase di preprocessamento vengono caricati.

4.3.3 SimulationTaskEvent

Nel caso in cui dalla coda degli eventi venga estratto un *SimulationTaskEvent* si hanno tre casi principali da gestire:

- **L'evento estratto non ha alcuna risorsa associata**: In questo caso l'evento viene consumato e il tempo di simulazione aggiornato. Quando il tempo di simulazione viene aggiornato viene prima fatta una comparazione fra il tempo di terminazione del task e il tempo attuale della simulazione. Nel caso in cui il tempo di terminazione del task sia maggiore dell'attuale tempo di simulazione allora il tempo di simulazione viene aggiornato in caso contrario no. Questo tipo di valutazione viene fatta nel caso in cui siano stati eseguiti altri task in parallelo che abbiano durate maggiori del task corrente.

- **L'evento estratto ha una risorsa associta necessaria per la sua esecuzione la quale è disponibile, e quindi può essere utilizzata per completare il task**: In questo caso vengono eseguite le medesime operazioni che si sono eseguite per il primo caso elencato con una unica eccezione; viene salvato temporaneamente nella risorsa utilizzata il tempo di terminazione del task corrente. Questa informazione servirà successivamente nel caso in cui tutte le risorse di una determinata tipologia non siano disponibili, la gestione dei tempi di rilascio delle risorse verrà discussa successivamente.

- **L'evento estratto ha una risorsa associata necessaria per la sua esecuzione la quale non è disponibile**: Nel terzo caso viene fatta una prima valutazione fra il tempo di rilascio più datato della risorsa necessaria e il tempo di presa in carico del task corrente. Se il tempo di presa in carico dell'ultimo task è inferiore al tempo di presa in carico del task corrente allora l'aggiornamento del tempo di simulazione non è altro che il tempo di simulazione attuale maggiorato del tempo necessario per completare il task corrente. Se il tempo di rilascio della risorsa è maggiore o uguale del tempo di presa in carico del task corrente allora significa che il task corrente ha dovuto attendere che una risorsa si liberasse per permettere il suo completamento. Prima di aggiornare il tempo di simulazione viene verificato che il tempo corrente di simulazione non sia già maggiore del tempo nel quale la

risorsa si sia liberata maggiorato del tempo di attività del task, se questa verifica risulta negativa allora si procede con l'aggiornamento del tempo di simulazione che quindi diverrà uguale al tempo di rilascio della risorsa più il tempo richiesto per completare il task corrente. Infine viene aggiornato il tempo di rilascio per la risorsa.

Si presenta un esempio per meglio spiegare il funzionamento dell'algoritmo appena descritto:

Il modello di processo presentato in figura ha un totale di 6 attività. Terminata la prima attività vi è un *parallel gateway*. Da qui il processo si dirama in 4 attività parallele e che quindi possono essere eseguite simultaneamente. Il colore delle singole attività identifica anche la tipologia di risorsa responsabile per il suo completamento. Le risorse disponibili per questo tipo di processo sono di due tipologie: blu e verde. Le quali hanno rispettivamente 1 e 2 unità disponibili. Le attività senza colore sono da considerarsi senza alcuna risorsa associata quindi vengono eseguite in maniera lineare influendo solo sul tempo di simulazione.

Il token, identificato in rosso, viene in una prima fase assegnato all'evento di start, il tempo di simulazione pari a 0 e con una coda degli eventi contenente il solo evento di inizio processo.

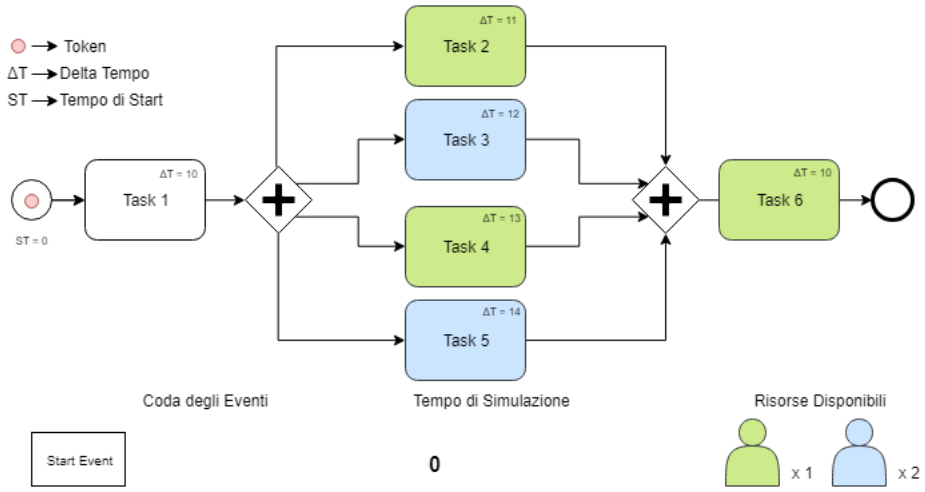


Figura 4.8: Algoritmo con task paralleli e risorse limitate (1)

Successivamente il token raggiungerà l'attività *task 1*. Il tempo di arrivo del token è 0 e il tempo richiesto dall'attività per essere completata è 10 secondi; nessuna risorsa è richiesta per il completamento dell'attività ciò ricade nel primo scenario ipotizzato: l'evento estratto non ha alcuna risorsa associata. Quindi l'aggiornamento del tempo avviene in maniera lineare. Al termine del completamento dell'attività abbiamo un tempo di simulazione pari a 10 secondi e ancora nessuna risorsa utilizzata.

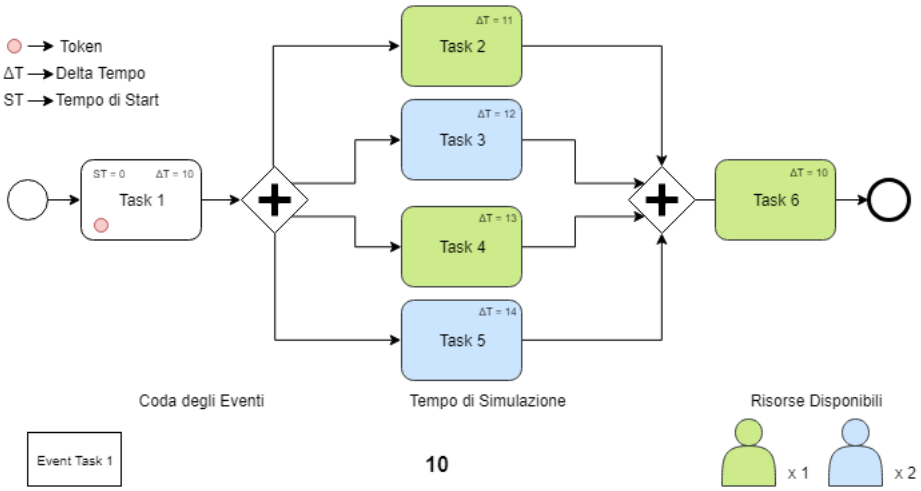


Figura 4.9: Algoritmo con task paralleli e risorse limitate (2)

Una volta passato il *parallel gateway* si hanno 4 task. Questi task possono essere eseguiti simultaneamente e possono arrivare in un qualsiasi ordine all'interno della coda degli eventi. In questo caso specifico si prende in carico l'attività *task 2*. Tale attività deve essere completate da una risorsa di tipo verde e il tempo di completamento richiesto è di 11 secondi, infine il tempo di presa in carico dell'attività da parte della risorsa è uguale a 10 secondi. Date queste condizioni si ricade nello scenario per il quale: l'evento estratto ha una risorsa associta necessaria per la sua esecuzione la quale è disponibile, e quindi può essere utilizzata per completare il task. Di conseguenza il tempo totale di simulazione viene aggiornato a 21 secondi e la risorse in questione viene marcata come non disponibile.

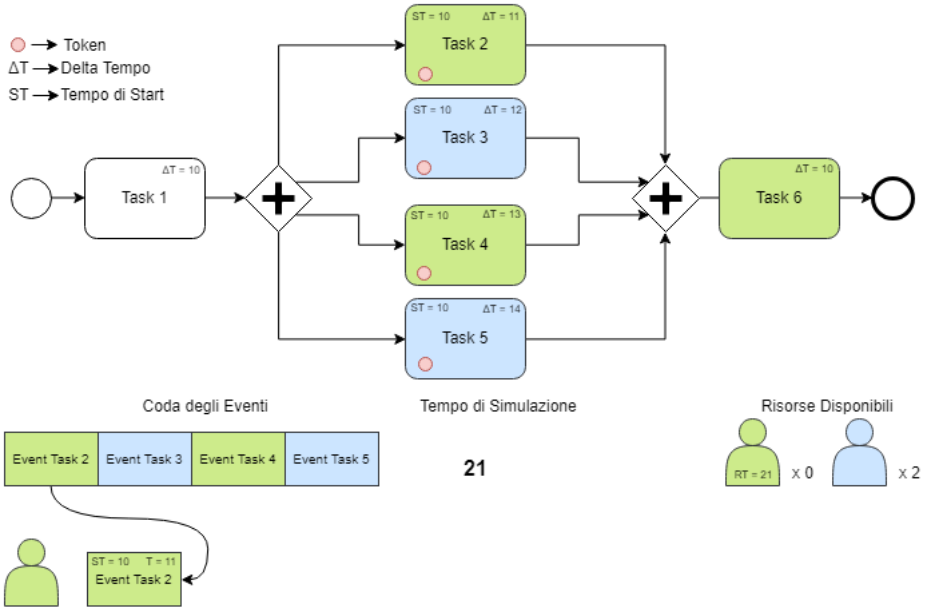


Figura 4.10: Algoritmo con task paralleli e risorse limitate (3)

Procedendo con la consumazione degli eventi si completa anche l'attività *task 3* la quale ricade nelle condizioni precedentemente descritte per l'attività *task 2* se non fosse che in questa caso dato che il tempo di esecuzione del task è maggiore del precedente quindi il tempo totale di simulazione viene aggiornato a 22. Questo avviene perché considerando il tempo di presa in carico dell'attività e il suo delta la loro somma è maggiore rispetto al tempo corrente di simulazione e di conseguenza può avvenire l'aggiornamento

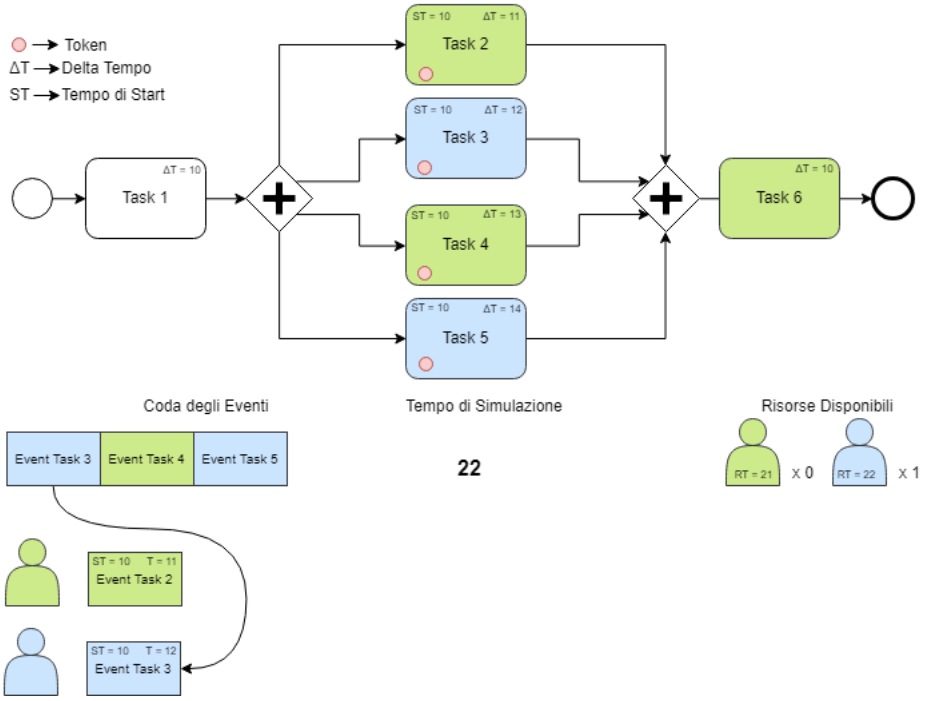


Figura 4.11: Algoritmo con task paralleli e risorse limitate (4)

Completata anche l'attività *task 3* nella coda degli eventi viene estratto l'evento *task 4*. Questo evento ha un tempo di presa in carico uguale a 10 secondi e un delta tempo uguale a 13 secondi. Al contrario di tutte le attività precedentemente descritte questa attività non ha alcuna risorsa disponibile per poter essere eseguita. In tal caso ci troviamo nello scenario dove: **l'evento estratto ha una risorsa associata necessaria per la sua esecuzione la quale non è disponibile**. Viene fatta quindi una prima valutazione fra il tempo di completamento dell'ultimo task da parte della risorsa responsabile per il completamento dell'attività corrente. In questo caso il tempo di rilascio della risorsa è maggiore del tempo di presa in carico del task corrente poiché la risorsa si libera a tempo 22 ma il task corrente è arrivato nella coda quando il tempo di simulazione era 10. Di conseguenza il task corrente ha dovuto attendere che una risorsa si liberasse per permettere il suo completamento. Prima di aggiornare il tempo di simulazione viene verificato che il tempo corrente di simulazione non sia già maggiore del tempo nel quale la risorsa si sia liberata maggiorato del tempo di attività del task, dato che questa verifica

risulta negativa allora si procede con l'aggiornamento del tempo di simulazione che quindi diverrà uguale al tempo di rilascio della risorsa più il tempo richiesto per completare il task corrente per un totale di 34 secondi.

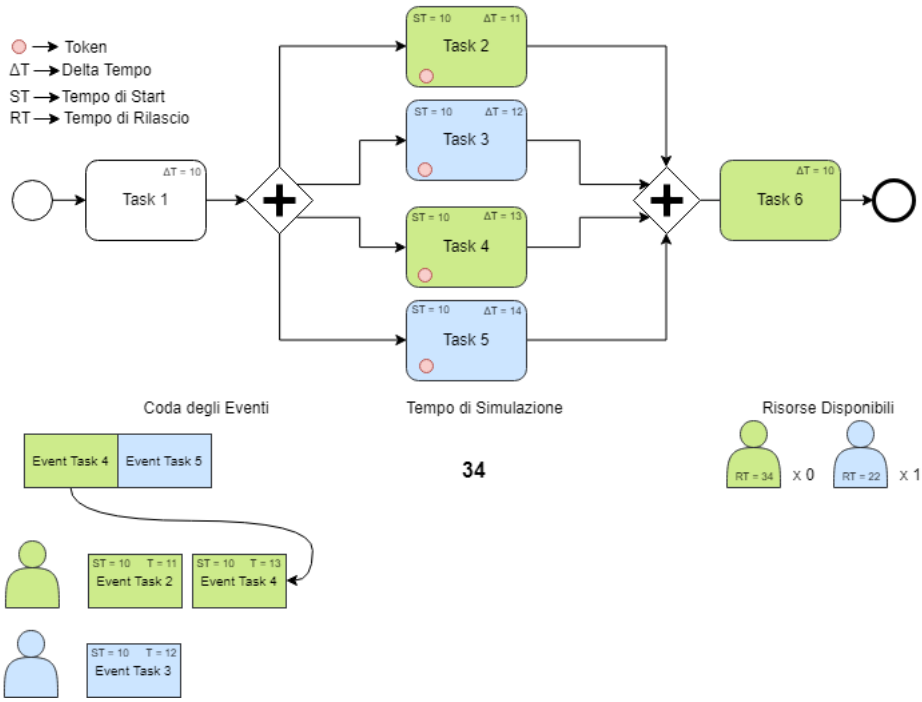


Figura 4.12: Algoritmo con task paralleli e risorse limitate (5)

Successivamente si prende in carico l'attività *task 5* la quale non influenza il tempo di simulazione visto che vi è una risorsa in grado di completare tale attività con un tempo di completamento dell'attività inferiore al tempo di simulazione attuale. Si noti che la coda dei tempi di rilascio della risorsa blu vede un nuovo elemento uguale al tempo di rilascio registrato durante il completamento dell'operazione descritta, ma essendo maggiore dell'elemento già presente non si trova nella testa della coda.

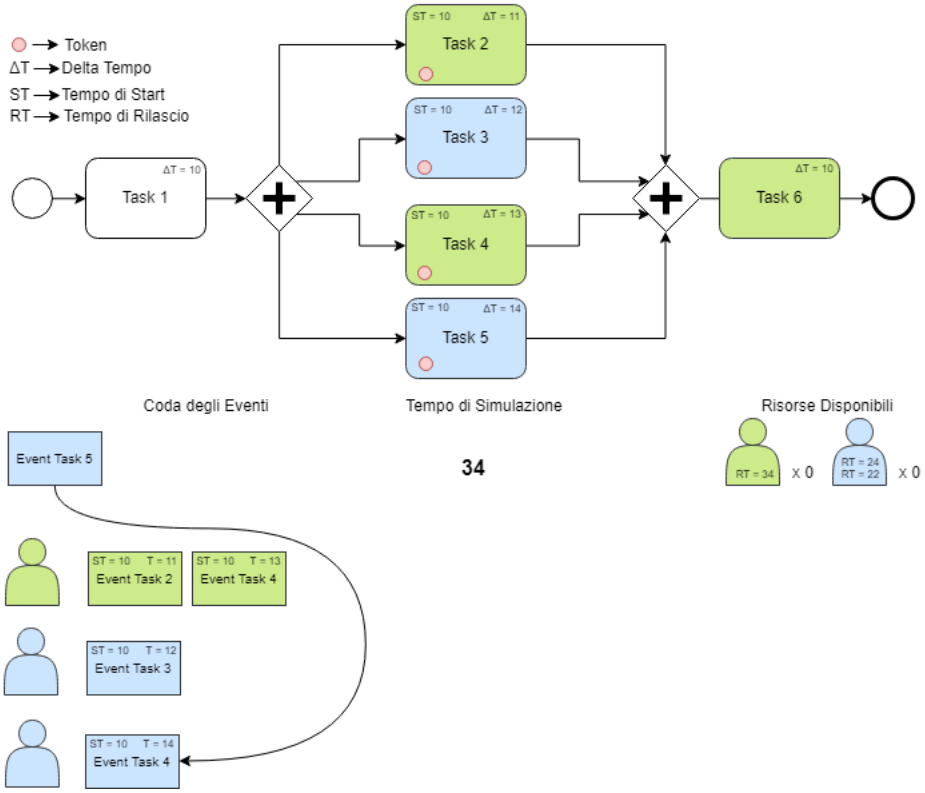


Figura 4.13: Algoritmo con task paralleli e risorse limitate (6)

Infine una volta completato il *parallel gateway* viene completata anche l'ultimo task. Tale attività non ha risorse disponibili per poter essere completata, ma dal momento che il tempo di rilascio della risorsa risale a 34 che è minore o uguale al tempo di presa in carico del task corrente che di fatto è anche esso 34, la risorsa viene automaticamente resa disponibile e viene completato anche l'ultimo task.

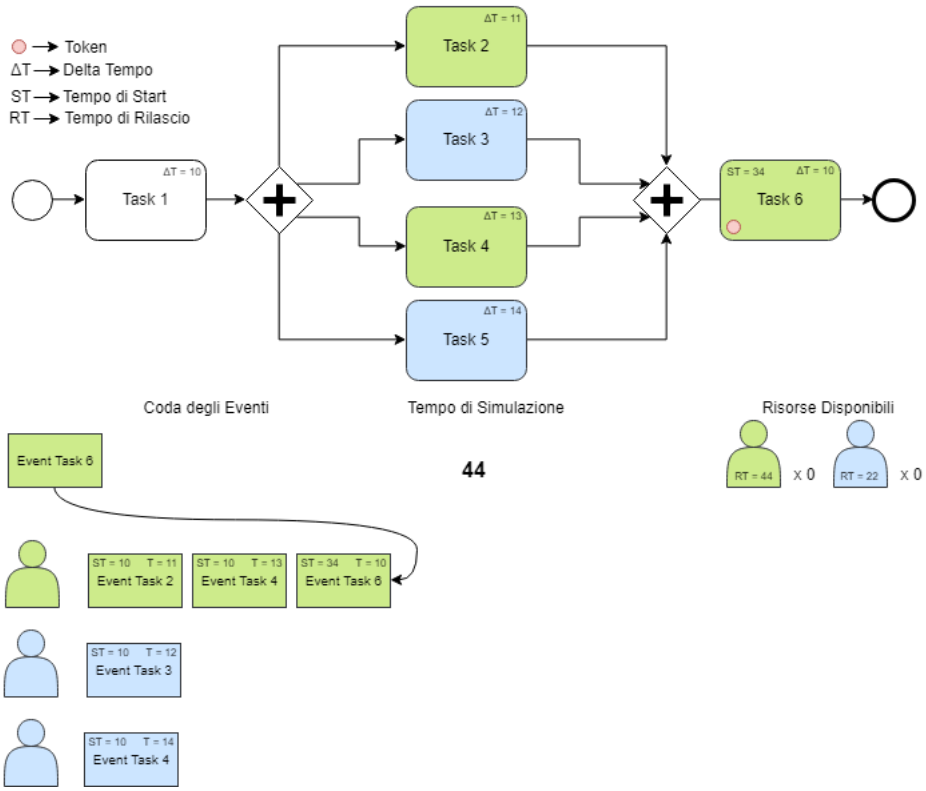


Figura 4.14: Algoritmo con task paralleli e risorse limitate (7)

Infine una volta che il token raggiunge l'evento di fine la simulazione termina con un tempo di esecuzione totale pari a 44 secondi.

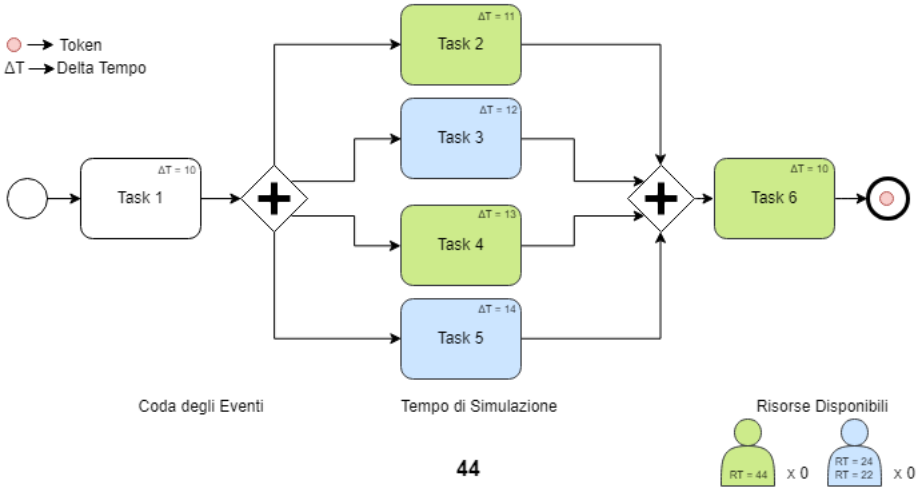


Figura 4.15: Algoritmo con task paralleli e risorse limitate (8)

4.4 Risultati

Una volta completata la Simulazione è tempo di scrivere i risultati. I risultati vengono collezionati dalle classi:

- *SimulationResult*
- *ResultsCatalog*, che non è altro che una collezione di *SimulationResult*

Le informazioni contenute in queste classi vengono poi salvate in un file xml specificato fra i parametri in input.

I risultati contengono i seguenti dettagli:

- Il tempo totale di esecuzione della simulazione
- Il costo totale della simulazione

A questi risultati globali si aggiungono anche le seguenti informazioni :

- I costi per ogni singola attività presente nel processo
- Il tempo per ogni singola attività presente nel processo

Infine vengono forniti i dettagli relativi all'utilizzo delle risorse. Quindi per ogni risorsa si danno i seguenti dettagli:

- Numero totale di attività completate
- Tempo totale per completare le attività assegnate
- Il tempo per ogni singola attività assegnata

Capitolo 5

Conclusioni

In conclusione, in questo elaborato si è analizzato lo stato dell'arte per le tematiche di simulazione dei processi aziendali definiti dalla specifica *BPMN 2.0*. Successivamente si è analizzata la specifica *BPSim*, e si sono discussi alcuni esempi di applicativi in grado di simulare processi aziendali definiti utilizzando le specifiche *BPMN* e *BPSim*.

Si è quindi descritto il processo di progettazione e implementazione di un applicativo che tramite l'utilizzo di queste specifiche fosse in grado di simulare i processi aziendali.

Per la realizzazione di questo progetto si è utilizzato un motore di simulazione per i processi aziendali open source chiamato: *Camunda*. Tutta la parte di gestione del modello *BPMN* è della relativa esecuzione è da attribuire a questo strumento. Per quanto concerne invece l'integrazione delle informazioni fornite dalla specifica *BPSim* si è realizzata l'applicazione Firefly la quale, interagendo con il l'engine *Camunda*, è in grado di fornire risultati di notevole rilevanza per quanto concerne la valutazione dei processi aziendali.

Il progetto Firefly è stato analizzato sia da un punto di vista architetturale definendo la struttura generale del progetto e come questo interagisce con i vari componenti utilizzati. Sia da un punto di vista algoritmico, formalizzando i problemi riscontrati durante la realizzazione del progetto e conseguentemente presentando opportune euristiche per la loro risoluzione.

Il codice di questo progetto è oggi consultabile su *Github* (http://github.com/rikirenz/camunda-simulation-bpmn). Inoltre è stata utilizzata la piattaforma di Continues integrtion *Tra-*

vis. Questa ha permesso di eseguire sistematicamente i test sviluppati al fine di validare i casi d'uso supportati, all'interno di un ambiente completamente agnostico così da validare ulteriormente la fattibilità dello studio svolto.

Fra gli sviluppi futuri più plausibili , vi è la creazione di un'applicazione web che utilizzando l'applicato *Firefly* permetta la simulazione di processi aziendali tramite l'utilizzo di una interfaccia web.

69

Bibliografia

[1] Weske, M. (2012). "Chapter 1: Introduction". Business Process Management: Concepts, Languages, Architectures. Springer Science Business Media. pp. 1–24.

[2] Kumar, Akhil (2018). Business process management. New York: Routledge. Print.

[3] Tumay, Kerim. "Business Process Simulation." Winter Simulation Conference (1995).

[4] Hammer, M., Champy, J. (1993). Reengineering the corporation: A manifesto for business revolution. New York, NY: HarperBusiness. Capitolo: The experience of process redesing

[5] OMG Business Process Model and Notation (BPMN), Version 2.0. , Object Management Group , Object Management Group (2011).

[6] BPSim 2013: Workflow Management Coalition: BPSim – Business Process Simulation Specification, Document Number WFMC -BPSWG-2012-1, 2013

[7] BPSim 2014: Workflow Management Coalition: BPSim Implementer's Guide, 2014

[8] BUSINESS PROCESS GLOSSARY, BPM Glossary [online] Disponibile: `https://www.businessprocessglossary.com/` [Acceduta il 9 Dicembre 2018]

[9] Camunda, BPMN Modeling Reference di Camunda [online] Disponibile: `https://Camunda.org/BPMN/reference/` [Acceduta il 9 Dicembre 2018]

[10] Github, GitHub Guides [online] Disponibile: `https://guides.github.com/` [Acceduta il 9 Dicembre 2018]

[11] Travis CI, Travis CI User Documentation [online] Disponibile: `https://docs.travis-ci.com/` [Acceduta il 9 Dicembre 2018]

[12] Maven, Maven Documentation [online] Disponibile: `https://maven.apache.org/guides/` [Acceduta il 9 Dicembre 2018]

[13] JAXB tool, Java Architecture for XML Binding. [online] Disponibile: `https://docs.oracle.com/javase/8/docs/technotes/guides/xml/jaxb/index.html` [Acceduta il 9 Dicembre 2018]

[14] JUnit tool, JUnit 5 [online] Disponibile: `https://junit.org/junit5/docs/current/api/overview-summary.html` [Acceduta il 9 Dicembre 2018]

Ringraziamenti

Ringrazio Davide Rossi e Francesco Poggi per avermi aiutato a completare questo elaborato. Ringrazio la mia Famiglia per avermi supportato in questo percorso.

Ringrazio Roberta Giacomo e Bonni per essermi stati tanto vicino. Ringrazio Shippo per esserci sempre nonostante passino gli anni e la grande distanza che ci divide.

Ringrazio Candre per aver reso il primo anno al CERN un esperienza indimenticabile.

Ringrazio Sara per aver reso quest'ultimo anno meno difficile e per regalarmi grandi momenti di serenità.

Ringrazio Gabriele e Barbara per il lavoro che abbiamo fatto insieme in questi anni.