Alma Mater Studiorum · Università di Bologna

**SCUOLA DI SCIENZE**
**Corso di Laurea Magistrale in Informatica**

# Modellazione dinamica di simulazioni BPMN: progettazione e realizzazione

**Relatore:**
**Chiar.mo Prof.**
**Davide Rossi**

**Presentata da:**
**Pierluigi Lazazzera**

**Correlatore:**
**Chiar.mo Prof.**
**Francesco Poggi**

**Sessione II**
**Anno Accademico 2018/2019**

*Alla mia famiglia,*
*Ad Arjola*

# Introduzione

Questo progetto di tesi consiste nella progettazione e realizzazione di una modellazione dinamica che è in grado, dato un diagramma di processo che segue le specifiche della notazione BPMN, di integrare ad esso la notazione di simulazione BPSim. La suddetta parte, permetterà di implementare, successivamente, una Desktop Progressive Web Application (Desktop PWA) per fare editing interattivo di simulazioni di processi aziendali definiti tramite le specifiche BPMN e BPSim.
Viene descritto, in seguito, in questo elaborato, lo stato dell'arte delle tematiche di simulazione dei processi aziendali definiti utilizzando le specifiche BPMN 2.0 e BPSim.
La notazione BPMN ha come obiettivo quello di 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 che permette di decorare il modello con informazioni utili al fine di simulare il processo BPMN descritto.
Successivamente si descrive nel dettaglio la progettazione e realizzazione della modellazione dinamica.
Si è utilizzato TypeScript per realizzare una struttura dati relativa ai parametri di simulazione e HTML, CSS, Javascript e JQuery per la realizzazione della web application che permettesse la manipolazione di questa struttura.
Integrare le informazioni fornite dalla specifica BPSim in diagrammi BPMN e quindi permettere la simulazione di processi è un'attività particolarmente interessante per le aziende, poiché permette di minimizzare i tempi con i quali si forniscono nuovi prodotti e servizi, massimizzare i profitti e minimizzare i rischi.

# Abstract

Lo scopo di questo lavoro di tesi è quello di implementare uno strumento che consenta di modificare i parametri di simulazione BPSim per i diagrammi BPMN. In particolare, ci si concentra sulla progettazione e implementazione di una modellazione dinamica che è la struttura dati di una progressive web application in grado di integrare la simulazione BPSim in un diagramma di processo.
L'integrazione delle informazioni fornite dalla specifica BPSim nei diagrammi BPMN e quindi consentire la simulazione dei processi è un'attività particolarmente interessante per le aziende, poiché consente di ridurre al minimo i tempi di fornitura di nuovi prodotti e servizi, massimizzare i profitti e minimizzare i rischi.

The purpose of this work is to implement a tool which allows editing BPSim simulation parameters for BPMN diagrams. Specifically, this project consists in the design and implementation of a dynamic modeling which is the "database" of a progressive web application that is able to integrate the BPSim simulation to a process diagram.
Integrating the information provided by the BPSim specification into BPMN diagrams and therefore allowing the simulation of processes is a particularly interesting activity for companies, since it allows to minimize the times with which new products and services are provided, maximize profits and minimize risks.

# Indice

# Elenco delle figure

# Elenco delle tabelle

# Capitolo 1

# Contesto e stato dell'arte

Questo capitolo introduce la nozione di Business Process Simulation e per quale motivo questo strumento è considerato essenziale nei processi decisionali di un'azienda.
Successivamente viene definita la specifica BPMN 2.0. Vengono descritte 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. Inoltre si descrive lo stato dell'arte per quanto concerne i simulatori che utilizzano le specifiche appena citate e vengono fatte emergere le motivazioni che hanno portato a sviluppare l'applicazione oggetto di questo elaborato.

## 1.1 Business Process Simulation

In vari settori, come ad esempio quello industriale o quello dei servizi, emergono nuove sfide e viene richiesto un costante lavoro di raffinamento e miglioramento a coloro i quali 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.
È 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 l'esternalizzazione di un processo o un'automatizzazione dello stesso, oppure il ridimensionamento o l'espansione della forza lavoro.
Valutare l'impatto che un cambiamento nei processi aziendali può apportare, con accuratezza e velocità, è diventato un fattore critico nei mercati.
La comprensione e la stima dei tempi e dei costi 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, che

sono impiegati dal management per pianificare al meglio l'attività aziendale e pianificare i propri obiettivi.
Tuttavia, questo tipo di analisi resta molto complessa data la natura delle attività che è possibile analizzare, 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 questi problemi appena descritti si aggiungono anche le richieste dei clienti, i quali con il passare del tempo tendono a pretendere costi sempre più bassi ed ad un aumento della qualità.
È evidente, quindi, che parametri come i tempi d'attesa per completare una certa attività o il costo richiesto per ultimare la stessa diventano quindi fondamentali per fornire, ad un prezzo competitivo, una determinata qualità del servizio.
Solitamente gli attori che hanno l'onere di prendere decisioni in merito alla gestione delle attività e, di conseguenza, decidono anche il prezzo con il quale un prodotto si presenta sul mercato, hanno sempre utilizzato strumenti analitici. Tali strumenti però non tengono in considerazione la dinamicità dei sistemi nei quali un'azienda è costretta ad operare.
Un esempio pratico di questo tipo di situazione può essere il problema tipico del dimensionamento delle scorte. La complessità data dalle operazioni di approvvigionamento, la variazione nella domanda e nei tempi di processing delle richieste e le dinamiche di sistema portano a un incertezza che non può essere modellata o analizzata utilizzando soltanto gli strumenti più comuni come ad esempio flowchart e fogli di lavoro elettronici.
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, rapidità e accuratezza non ha eguali oggigiorno. Questo strumento, quindi, permette di valutare alternative prima che avvengano ingenti investimenti in termini di tempo e risorse.
La capacità di visualizzare come un processo possa svolgersi e come questi vada poi a interagire con la 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, nel momento della sua simulazione, 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.

### 1.1.1 Processo

Esistono diverse definizioni di processo 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].
Si può 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 obiettivo [1].

### 1.1.2 Elementi per la modellazione

I costrutti utilizzati per definire modelli di processo possono avere differenti nomenclature o caratteristiche per diversi prodotti, ma in molti dei modelli di simulazione di business process sono semanticamente equivalenti.
Il primo a ipotizzare una classificazione per gli elementi di modellazione per descrivere un processo è stato Tumay in *Business Process Simulation* [2]. Tale classificazione è, successivamente, diventata lo standard de facto.
I quattro costrutti base sono i seguenti:

- **Token (Entità)**
  I token sono anche definiti come oggetti di flusso. Questi sono oggetti che possono essere processati dalle risorse. Rappresentano, in sostanza, l'entità di una determinata attività che viene presa in carico dalla risorsa responsabile.

- **Attività**
  Le attività non sono altro che i compiti svolti dalle entità, come ad esempio l'assemblaggio, l'interazione con un cliente al fine di completare una richiesta o la produzione di documenti. Inoltre un attività trova una definizione completa quando viene decorata, ovvero quando vengono aggiunte ad essa informazioni rilevanti, quali ad esempio il tempo richiesto per completarla oppure le risorse necessarie al suo soddisfacimento.

- **Risorse**
  Le risorse sono quegli elementi necessari a un gruppo di lavoro per comprendere un problema e implementare le relative soluzioni. Sono considerate risorse, ad esempio, il capitale umano di un'azienda, il tempo che viene dedicato all'elaborazione di una soluzione, etc.

- **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 determinate attività in base alle regole definite dal modello.

### 1.1.3 Misure di performance

In letteratura esistono diverse misure atte a valutare le performance di un modello di processo.
Le misure più comuni utilizzate per valutare le performance di un modello di processo sono le seguenti:

- **Cycle Time**
  Il cycle time è il tempo totale che un entità impiega per attraversare l'intero process. Questo tempo include il tempo di spostamento da un attività all'altra, il tempo di processamento, il tempo di attesa, etc. È possibile fornire anche il cycle time medio nel caso in cui vengano eseguite simulazioni multiple.
  Questa misura viene considerata molto importante e, talvolta, le altre misure sono considerate misure di "contorno" in confronto al cycle time [4].

- **Costi delle attività**
  Un'attività, quando è definita in un modello di process, è caratterizzata da:

  - il numero di unità disponibili;
  - il costo di installazione;
  - il costo di utilizzo;
  - i costi fissi;
  - il numero di risorse richieste per eseguirla;
  - 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.

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

- **Conteggio delle entità**
  Il conteggio delle entità si riferisce al numero totale delle entità che attraversano un processo o che sono già state 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.

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 la puntualità nel completamento di determinate attività, il livello di servizio fornito al cliente, etc. [3].

### 1.1.4 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 è tale se la definizione del processo non è ambigua, ciò significa che durante la sua implementazione non deve esserci spazio per interpretazioni personali nel suo funzionamento. Esistono diverse sintassi che si possono utilizzare per descrivere un processo. Fra le modalità più diffuse per descrivere un processo troviamo:

- **Business Process Model and Notation (BPMN)**
  Lo standard de facto per la rappresentazione dei diagrammi di business process [5]. Questa sintassi sarà ampiamente discussa nella sezione 1.2 BPMN.

- **Business Process Execution Language (BPEL)**
  BPEL è un linguaggio basato su XML per la rappresentazione dei business pro-

cesses. È possibile definire BPEL come un linguaggio di orchestrazione, ovvero permette di interagire con altri servizi.

- **Event Process Chain Diagrams (EPC)**
  EPC non è altro che un grafo ordinato di eventi e funzioni. Ogni attività all'interno del processo è attivata da un evento.

- **Flowchart**
  I flowchart non sono altro che un insieme di rettangoli e linee. Non hanno una sintassi ricca, quindi è evidente che caratterizzare un processo utilizzando questa sintassi può diventare complesso.

- **Petri Nets**
  Petri Nets 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 [3].

Per quanto concerne le tecniche di modellazione invece, esistono diverse metodologie che permettono di modellare un processo. È 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:

- Project-based processes;
- Production-based processes;
- Distribution-based processes;
- Customer service based processes.

Questa classificazione, introdotta da Tumay in Business Process Simulation [2], 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.

### 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 in modo che si producano risultati molto più accurati, dato che i tempi per le attività tipiche per queste tipologie di processi tendono ad avere tempistiche molto

variabili e l'utilizzo di risorse multiple permette il fatto 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 curva di apprendimento per le risorse investite nel processo, la prioritizzazione di alcuni task, 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, poiché 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.

### Production-based processes

In questo tipo di simulazioni gli output sono prodotti in grande quantità, solitamente in lotti o come flusso continuo. Degli esempi di processi che rientrano in questa categoria processi possono essere l'evasione degli ordini, la gestione dei 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.
Il fine ultimo della simulazione di questa tipologia di processi è quello di ottenere un processo che a regime risulti stazionario, ovvero un modello dove viene prodotta, infine, la stessa sequenza di prodotti.
È importante specificare che, nell'analisi delle prestazioni, sono sempre da eliminare le problematiche relative ai dati raccolti durante la fase di warm-up della simulazione. I dati prodotti in questa fase non vengono considerati validi ed è quindi buona pratica non raccoglierne i dati ed iniziare a farlo non appena la fase di warm-up termina.

### Distribution-based processes

Questa categoria di processi include processi come ad esempio quelli 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 sulle entità che vengono trasportate: i processi di trasporto si trovano nei sistemi di trasporto di massa come nelle compagnie aeree o ferroviarie (le entità trasportate sono persone), al contrario i processi di consegna si trovano nella distribuzione della produzione, nei servizi postali o di logistica (le entità trasportate sono merci o oggetti vari).
Quando si modella il trasporto, a volte può essere più appropriato rappresentare le risorse di trasporto come entità piuttosto che come risorse. Quando, invece, si modellano

i processi di distribuzione è importante definire gli attributi destinazione, dimensione e costo, delle singole entità, in maniera precisa. La maggior parte dei processi di distribuzione è transitoria nel comportamento, pertanto il periodo di simulazione dovrebbe essere abbastanza lungo 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.

### Customer service based processes

Questa tipologia di processi è quella che utilizzata maggiormente nella simulazione dei processi.
Solitamente queste tipologie di servizi vedono come unici attori gli esseri umani. Alcuni esempi di customer service based processes sono i servizi telefonici come i call center 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. ed a questo si aggiunge il fatto 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 che i clienti tendono ad arrivare in maniera casuale rendendo sempre più complessa la modellazione 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.

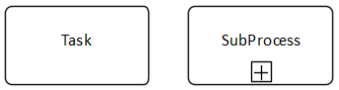
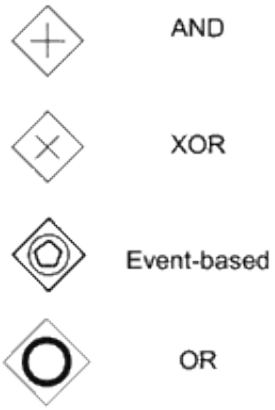
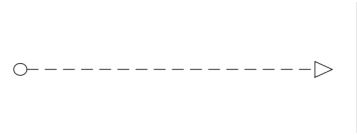
## 1.2 BPMN

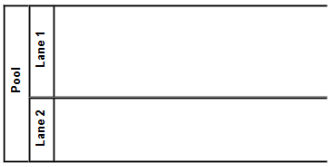
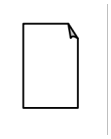
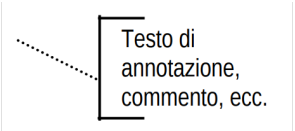
La notazione BPMN è sviluppata dalla "Business Process Management Initiative" e dallo "Object Management Group", associazioni no-profit che raccolgono operatori nel campo dell'informatica e dell'analisi organizzativa.
Questa notazione ha come obiettivo fornire uno standard di rappresentazione che sia facile da utilizzare e da comprendere da parte degli utenti business interessati al problema della modellazione, progettazione ed eventuale informatizzazione dei processi aziendali, come ad esempio gli analisti di processo che costruiscono le bozze iniziali dei processi organizzativi in esame o da progettare, oppure programmatori e sviluppatori delle applicazioni informatiche per la gestione di tali processi e infine manager e dirigenti responsabili della gestione e del monitoraggio dei processi stessi.
La notazione BPMN è sostanzialmente una derivazione del formalismo dei flow chart ma con alcune aggiunte e modificazioni che permettono di superarne alcuni limiti nella modellazione dei processi aziendali. Permette di costruire dei diagrammi di processo (BPD – Business Process Diagram) che rappresentano in pratica dei grafi o reti costituiti da "oggetti" rappresentati dalle attività di processo, collegati da flussi di controllo che definiscono la relazione logica, le dipendenze e l'ordine di esecuzione delle attività stesse. La versione 1.0 dello standard è stata rilasciata nel 2004, mentre in questo progetto viene utilizzata la versione 2.0, rilasciata nel 2011. La BPMN consente spesso una stretta integrazione con i sistemi di sviluppo software. L'obiettivo di questa specifica è, infatti, quello di promuovere la portabilità delle definizioni di processo, affinché gli utenti possano usare tali definizioni rimanendo indipendenti dagli applicativi software con i quali si possono progettare i Business Process Diagram [6].

### 1.2.1 Elementi BPMN

Si definiscono ora in dettaglio le componenti della notazione 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* indica genericamente un compito o operazione svolti all'interno di un processo. Un'attività può rappresentare un compito elementare e atomico (task), ossia non ulteriormente scomposto nell'analisi del processo in questione ed in tal caso sarà rappresentata con un rettangolo smussato, oppure un sotto-processo che potrà venire ulteriormente scomposto nei suoi task elementari. In questo secondo caso si usa apporre il simbolo + nella parte bassa centrale del rettangolo. |  |
| Gateway | Un *Gateway* definisce i punti del processo in cui i flussi delle attività divergono oppure convergono ed è definito con la forma di un rombo. È utilizzato per rappresentare i tradizionali punti di decisione come nei classici flow chart, ma anche semplici biforcazioni del flusso delle attività in attività parallele o viceversa il ricongiungimento di attività parallele in un flusso unico. Con determinati simboli è specificare il tipo di meccanismo di controllo dei flussi di attività che si biforcano o convergono. Ad esempio, una biforcazione o ricongiungimento in due attività parallele si può indicare con un + all'interno del rombo. |  |
| Sequence Flow | Una *Sequence Flow* è utilizzata per mostrare l'ordine che le attività che si eseguono in un processo e viene rappresentata graficamente con una freccia piena. |  |
| Message Flow | Una *Message Flow* simboleggia il fatto che un messaggio viene scambiato tra due diverse attività o entità partecipanti al processo, una che trasmette e l'altra che riceve il messaggio e viene rappresentata graficamente da linea tratteggiata con una freccia vuota. |  |

| | | |
|---|---|---|
| Association | Una *Association* è usata per indicare un semplice legame tra dati, testi e altri oggetti. Sono usate anche per indicare gli input e gli output delle attività e viene rappresentata graficamente con una linea a puntini e una freccia a punta aperta. |  |
| Pool | Una *Pool* rappresenta un'entità organizzativa ben definita che svolge un proprio processo eventualmente interagendo con altre unità organizzative. La notazione grafica è un rettangolo e serve a distinguere ciò che viene svolto da quella unità organizzativa rispetto a ciò che è responsabilità di altri e sarà contenuto in altre pool. Una pool può avere dettagli interni relativi al processo che lo ospita, oppure può essere trattata come una black-box e quindi senza dettagli interni. |  |
| Lane | Una *Lane* è una sotto partizione all'interno di una pool. Una lane si può estendere per l'intera lunghezza della pool, sia verticalmente che orizzontalmente. Le Lanes sono utilizzate per organizzare e categorizzare le attività. |  |
| Message | Un *Message* è utilizzato per rappresentare i contenuti di una comunicazione fra 2 partecipanti. |  |
| Data Object | I *Data Object* servono a descrivere i tipi di dati che sono necessari o prodotti da un'attività. Sono collegati alle attività attraverso il connettore *association*. |  |
| Text Annotation | La *Text Annotation* permette al modellista di aggiungere un testo di chiarimento o commento al fine di fornire ulteriori informazioni a chi deve interpretare il diagramma. |  |

| | | |
|---|---|---|
| Group | Un *Group* è rappresentato da un rettangolo con l'angolo arrotondato ed è disegnato con una linea tratteggiata. Il raggruppamento può essere utilizzato a scopo di documentazione o analisi, ma non influisce sul flusso del diagramma. |  |

Tabella 1.2: Tabella degli elementi BPMN

### 1.2.2 Software per la modellazione BPMN

Esistono diversi tools che possono essere utilizzati per creare modelli di processo e che utilizzano la notazione BPMN. Di seguito sono elencati i software più conosciuti.

- **Camunda Modeler**
  È un'applicazione desktop che permette di creare BPMN workflows e tabelle decisionali DMN in un editor utilizzabile sia dagli utenti finali che dagli sviluppatori. È possibile, inoltre, eseguire i workflows e ciò permette di costruire applicazioni essenziali per i progetti di automazione dei processi [7].

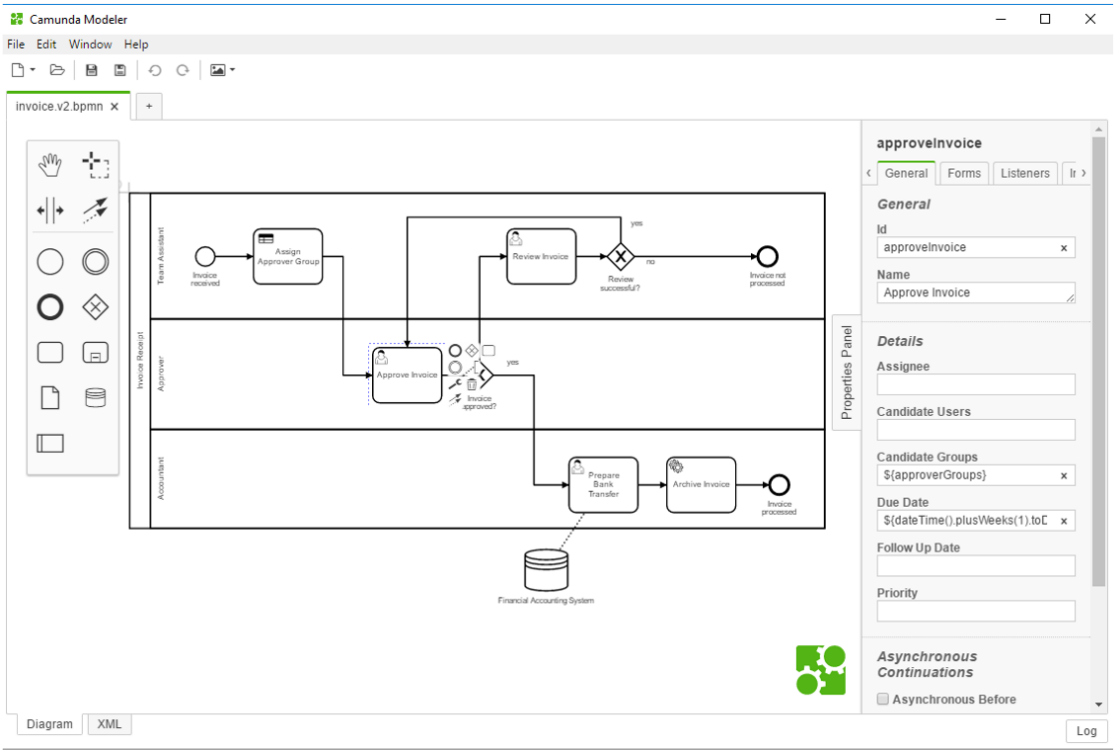



Figura 1.1: Esempio di diagramma BPMN in Camunda Modeler

- **BPMN.io**
  Questo business process modeler non è altro che la versione web based del Camunda Modeler. È possibile, inoltre, aggiungere ulteriori funzionalità, in una pagina web, come ad esempio colorare degli elementi del diagramma o aggiungere dei commenti per facilitare l'interpretazione dell'intero processo [8].

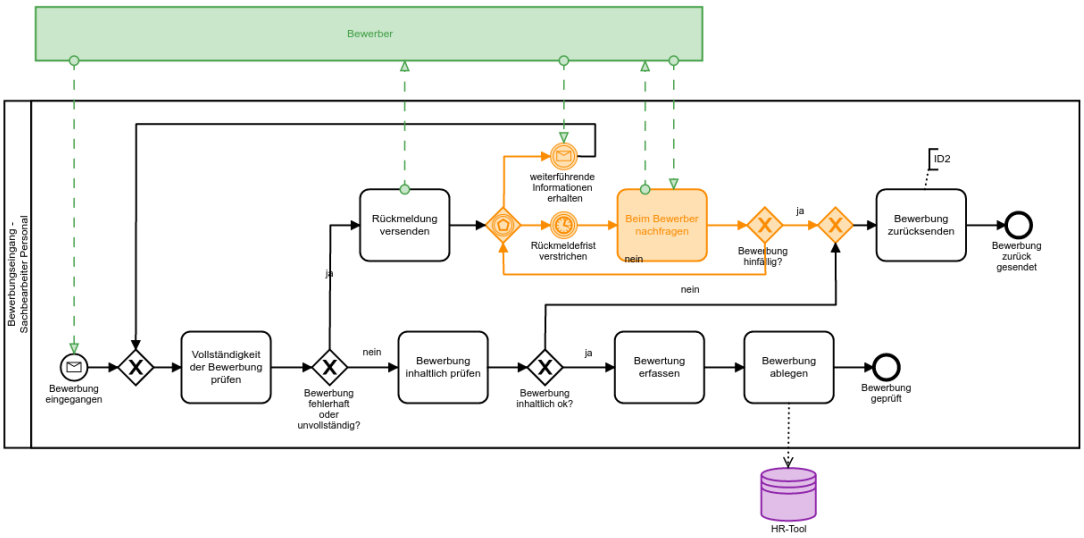



Figura 1.2: Esempio di diagramma BPMN in BPMN.io esteso con la funzionalità dei colori

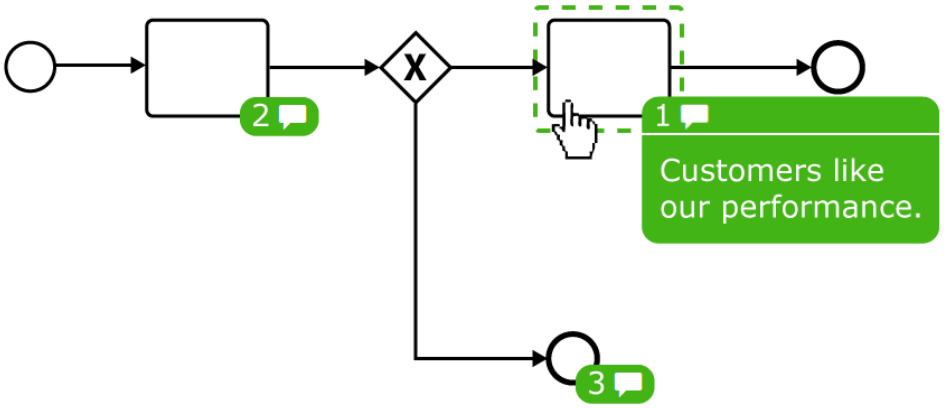



Figura 1.3: Esempio di diagramma BPMN in BPMN.io esteso con la funzionalità dei commenti

- **Signavio**
  Signavio è un web based business process modeler che permette a diversi utenti di poter collaborare a una stessa modellazione di processo in tempo reale. Il modeler di Signavio, è una componente di una suite più complessa. Utilizzando questo prodotto si ha la possibilità di integrare la modellazione di un processo con altri prodotti Signavio permettendo un'analisi del processo aziendale più accurata [9].

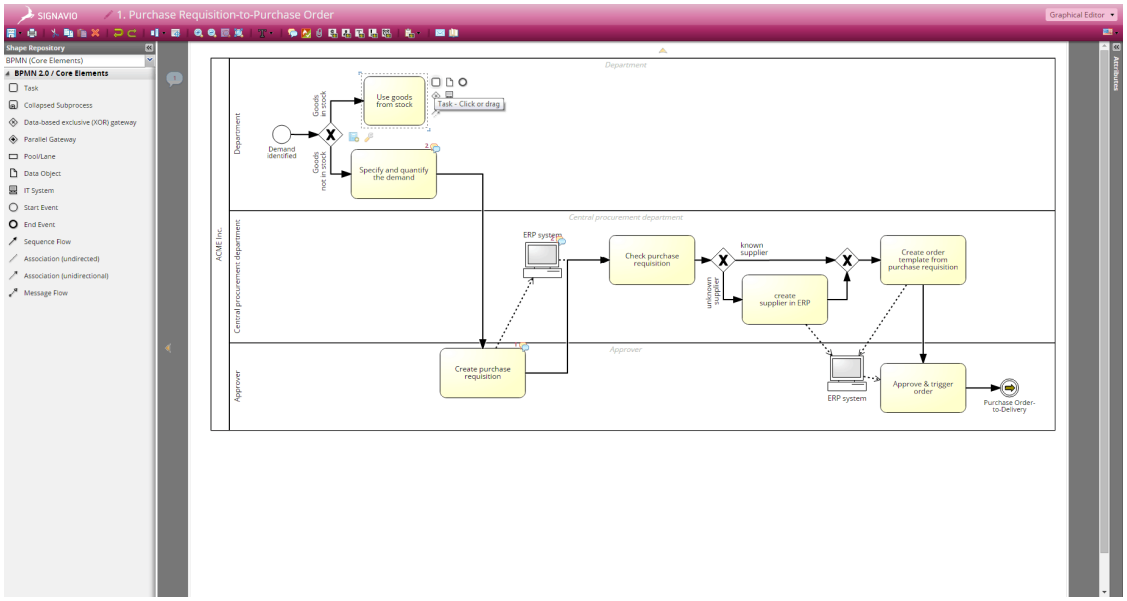



Figura 1.4: Esempio di diagramma BPMN in Signavio

- **Bizagi**
  Bizagi è un'applicazione desktop che permette la realizzazione di diagrammi di processo. È gratuito per utenti singoli e piccole organizzazioni. Nella versione a pagamento aggiunge features interessanti, infatti permette a più utenti di collaborare sullo stesso diagramma e di ricevere notifiche in real time.
  È presente un forum dove è possibile postare per risolvere eventuali problematiche incontrate nello sviluppo del diagramma di processo ed anche una serie di video tutorial per aiutare gli utenti nel prendere dimestichezza con l'applicazione [10].

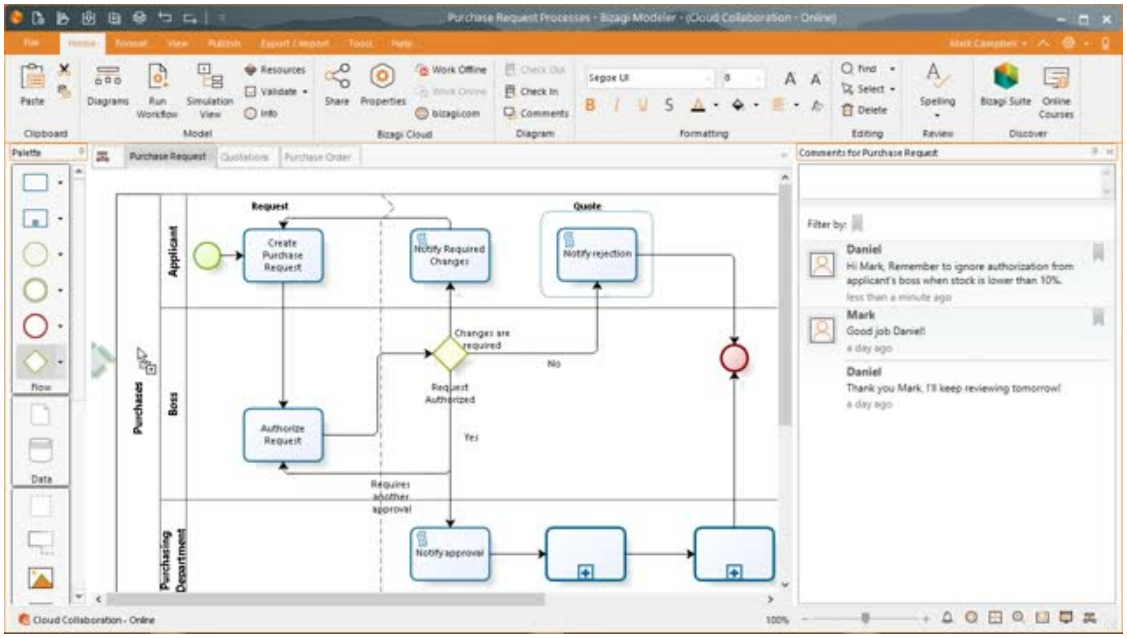



Figura 1.5: Esempio di diagramma BPMN in Bizagi

- **Visual Paradigm**
  A differenza dei precedenti software open-source, il business process modeler di Visual Paradigm fa parte di un'applicazione più grande di tipo enterprise che permette non solo la modellazione di diagrammi con notazione BPMN, ma anche, ad esempio, diagrammi UML [11].

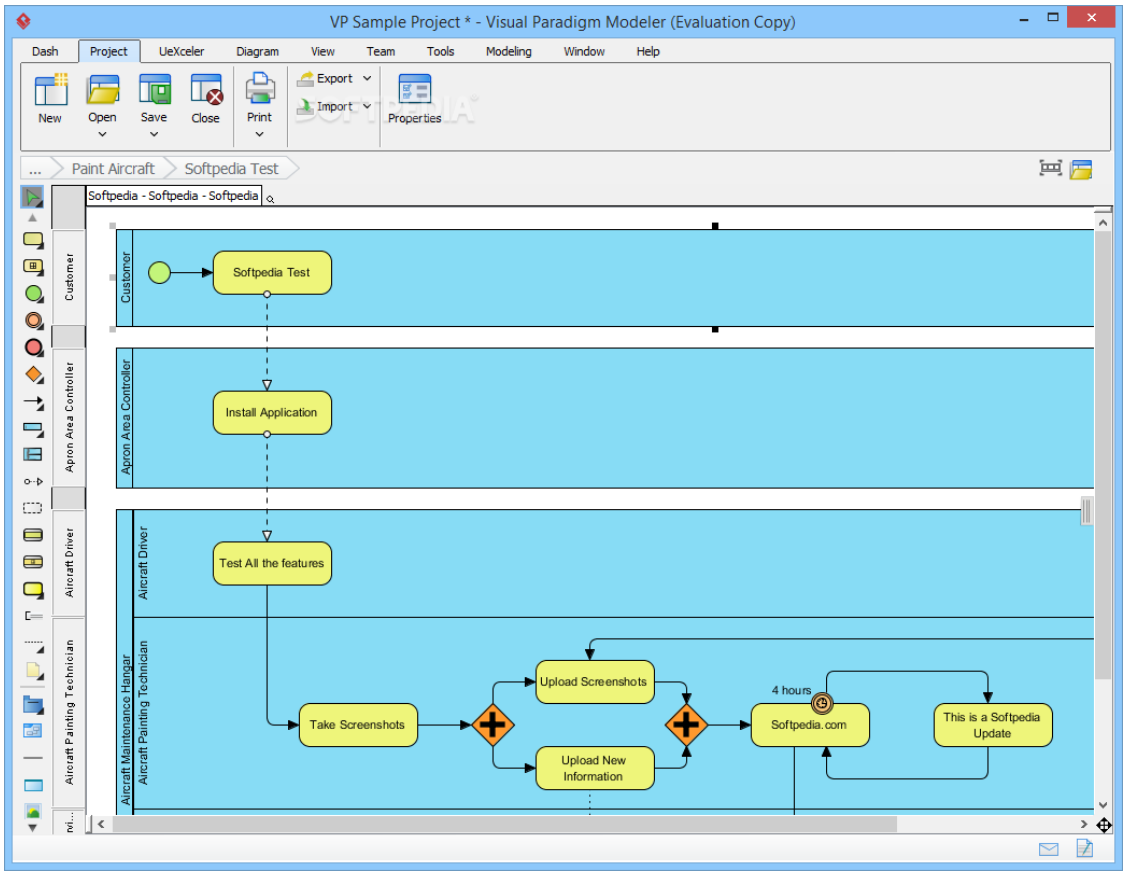



Figura 1.6: Esempio di diagramma BPMN in Visual Paradigm

## 1.3 BPSim

La simulazione e l'analisi dei processi aziendali vengono riconosciute come parte integrante dell'ottimizzazione delle prestazioni dei processi aziendali, che se inadeguati, comportano il mancato soddisfacimento delle aspettative dei clienti e comportamenti organizzativi indesiderati che possono a loro volta comportare la perdita di entrate o peggio. Questo è il motivo per cui è importante analizzare a fondo i processi aziendali in un ambiente isolato e sicuro prima che vengano implementati.
Esistono evidenti vantaggi nella simulazione e dell'analisi dei processi di business rispetto ai test di nuovi processi di business nel mondo reale, ad esempio non vi è alcun disturbo alle operazioni correnti ed ha un costo decisamente inferiore lo sperimentare sulla trasformazione del business.
Sebbene la simulazione sia ritenuta parte integrante del Business Process Management (BPM), la simulazione e l'analisi dei processi aziendali non è ancora sistematicamente utilizzata nella maggior parte dei progetti di miglioramento dei processi aziendali. Le motivazioni possono essere molte, ma sicuramente la mancanza di esistenza di standard può essere considerata la ragione principale. Sebbene esistano standard maturi per la definizione di modelli di processi aziendali (ad es. BPMN e XPDL), non esisteva, prima dell'avvendo di BPSim uno standard generalmente accettato per la simulazione e l'analisi dei processi aziendali.

Il Business Process Simulation (BPSim) framework è una specifica standardizzata che consente di ampliare i modelli di processi aziendali catturati in BPMN o XPDL con informazioni a supporto di rigorosi metodi di analisi. Questa specifica definisce la parametrizzazione e l'interscambio di dati di analisi di processo consentendo l'analisi strutturale e di capacità 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. Il metamodello BPSim viene acquisito utilizzando Unified Modeling Language (UML) e il formato di interscambio viene definito utilizzando una XML Schema Definition (XSD). La figura 1.7 sintetizza l'architettura della specifica BPSim. Si noti che il meta-modello BPSim e il formato di interscambio rappresentano il nucleo del materiale normativo di questa specifica. Al fine di supportare sia l'ottimizzazione pre-esecuzione che post-esecuzione, il metamodello e il formato di interscambio consentono l'acquisizione di input e output dell'analisi di processo, infatti sia i valori stimati che i valori di esecuzione storici sono supportati come parametrizzazione del modello di processo aziendale.
Uno degli obiettivi di questa specifica è quello di essere complementare agli standard già esistenti relativi alla modellazione dei processi aziendali. La versione della specifica BPSim utilizzata in questo progetto è basata sulla Business Model Model and Notation (BPMN) versione 2.0 di Object Management Group (OMG) e XML Process Definition Language (XPDL) versione 2.2 della Workflow Management Coalition (WfMC). Il modello concettuale BPSim è presentato di seguito [12].

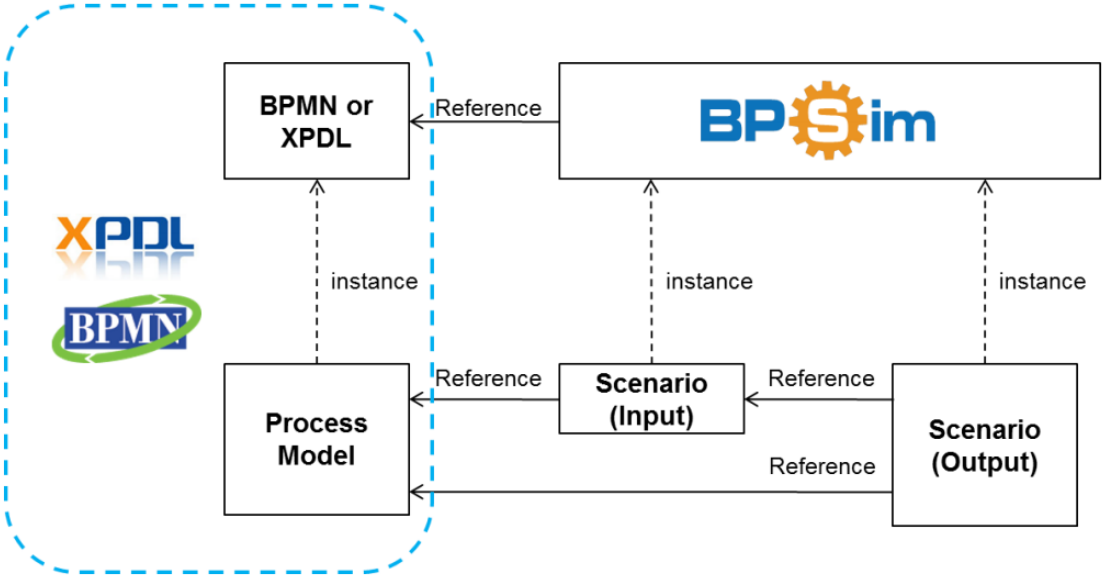



Figura 1.7: Architettura BPSim

### 1.3.1 Elementi BPSim

Esistono vari elementi nello standard BPSim. Di seguito viene riportata una descrizione dei principali elementi.

#### 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 business process model può vere differenti scenari da testare. Gli scenari possono essere utilizzati per acquisire:

- input delle specifiche dei parametri per l'analisi, la simulazione e l'ottimizzazione;
- risultati di analisi, simulazione e ottimizzazione;
- dati storici relativi all'esecuzione nel mondo reale passata del modello di processo aziendale.

Nella specifica BPSim è possibile avere scenari come risultato di altri scenari ed è anche possibile definire che uno scenario erediti da un altro scenario e aggiunga o modifichi alcuni parametri. In tal caso, oltre all'aggiunta di parametri per un determinato elemento, devono essere specificate nello scenario ereditato solo le modifiche ai valori di parametri esistenti per un elemento BPMN. Riassumento, si può affermare che uno scenario è composto da una raccolta di parametri degli elementi e dove ogni parametro di un elemento di uno scenario fa riferimento a un elemento specifico di un processo all'interno del business process model.

### Parameters

Ogni parametro di uno scenario referenzia un elemento specifico del processo dentro il quale è definito lo scenario. Per favorire quella che in gergo viene chiamata "separation of concerns", i parametri degli elementi sono divisi in categorie, dove ogni categoria raggruppa una serie di parametri accomunati da delle caratteristiche (costo, tempo, etc.). Inoltre è possibile specificare un periodo di validità per i valori dei parametri con un apposito elemento chiamato Calendar, infatti è possibile, ad esempio, che un determinato parametro abbia un valore X nei giorni feriali ed un valore Y nei fine settimana. Di seguito sono elencati i vari tipi di parametri esistenti:

- **Element Parameters**
  La classe ElementParameter 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.
- **Time Parameters**
  La classe TimeParameter raggruppa tutti i parametri relativi al tempo per un determinato elemento del processo. Tutti i TimeParameters definiscono intervalli temporali in varie maniere e sono definiti dal punto di vista di un osservatore esterno al processo.
- **Control Parameters**
  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 parametri è applicabile solo a certe categorie di elementi.
- **Resource Parameters**
  La classe ResourceParameter raggruppa tutti i parametri relativi alle risorse di un elemento del processo. Nuovamente, questo tipo di parametri è applicabile solo a certe categorie di elementi.

- **Cost Parameters**
  La classe CostParameter raggruppa tutti i parametri relativi ai costi di un elemento del processo. È possibile applicare questi parametri solo a certe categorie di elementi.

- **Property Parameters**
  La classe PropertyParameter raggruppa tutti gli attributi specifici per un dato elemento. Questi attributi sono definiti dagli agenti che modellizzano il processo. Questa tipologia di parametri è applicabile solo a certe categorie di elementi.

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

### Parameter Types

Un parametro può avere diversi valori associati ad esso. Ogni valore aggiuntivo può specificare una validità temporale con un parametro di tipo Calendar e deve essere di un tipo specifico. Di seguito sono elencati i vari tipi di parameter value esistenti:

- **Constant Parameter**
  I Constant Parameters sono parameter values che si risolveranno sempre allo stesso valore nel tempo.

- **Distribution Parameter**
  I Distribution Parameters sono parameter values che hanno valori diversi nel tempo ma statisticamente distribuiti secondo una data distribuzione.

- **Enum Parameter**
  Gli Enum Parameter sono parameter values che permettono alla specifica BPSim di supportare l'uso di dati storici in due modi, fornendo i numeri reali come parametri, ovvero una sequenza di tempi di elaborazione per un'attività, oppure, in un modo più comune, utilizzando i dati storici per un periodo di tempo appropriato da utilizzare per generare una distribuzione.

- **Expression Parameter**
  Gli Expression Parameters sono parameter values che sono una combinazione di valori espliciti, operatori e funzioni. I valori vengono calcolati in fase di esecuzione fornendo un risultato determinato dall'espressione (si utilizza una XPATH Expression).

### 1.3.2 Software per la simulazione di diagrammi BPMN

Esistono diversi software che permettono la simulazione di diagrammi con notazione BPMN. Di seguito vengono presentati i più conosciuti, ma soltanto i primi due seguono lo standard BPSim, utilizzato anche in questo progetto di tesi.

#### BIMP

BIMP è un simulatore web-based gratuito, veloce e semplice di modelli di processi aziendali BPMN ed è supportato dall'Università di Tartu e dal Consiglio di ricerca estone. Di seguito vengono elencati i tre passaggi necessari per la simulazione di processi BPMN in BIMP:

1. **Caricamento del diagramma BPMN**
   È possibile caricare modelli BPMN 2.0 creati utilizzando uno strumento BPMN conforme agli standard come BPMN.io, Camunda o Signavio;

2. **Editing dei parametri di simulazione**
   Dopo aver caricato un modello BPMN, viene richiesto di creare uno scenario di simulazione che includa una serie di dati, come ad esempio il numero di istanze di processo o la durata e i costi di task ed eventi;

3. **Simulazione ed analisi dei risultati**
   Dopo aver eseguito lo scenario di simulazione, BIMP fornisce un dashboard dove è possibile vedere, ad esempio, il costo dell'esecuzione del processo o dove sono locati i colli di bottiglia.

Se da un lato BIMP è uno strumento molto potente, in grado di produrre una serie di analisi interessanti per le aziende, dall'altro lato non rispetta lo standard BPSim e quindi non permette l'utilizzo di diagrammi BPMN che presentano già la notazione BPSim [13].

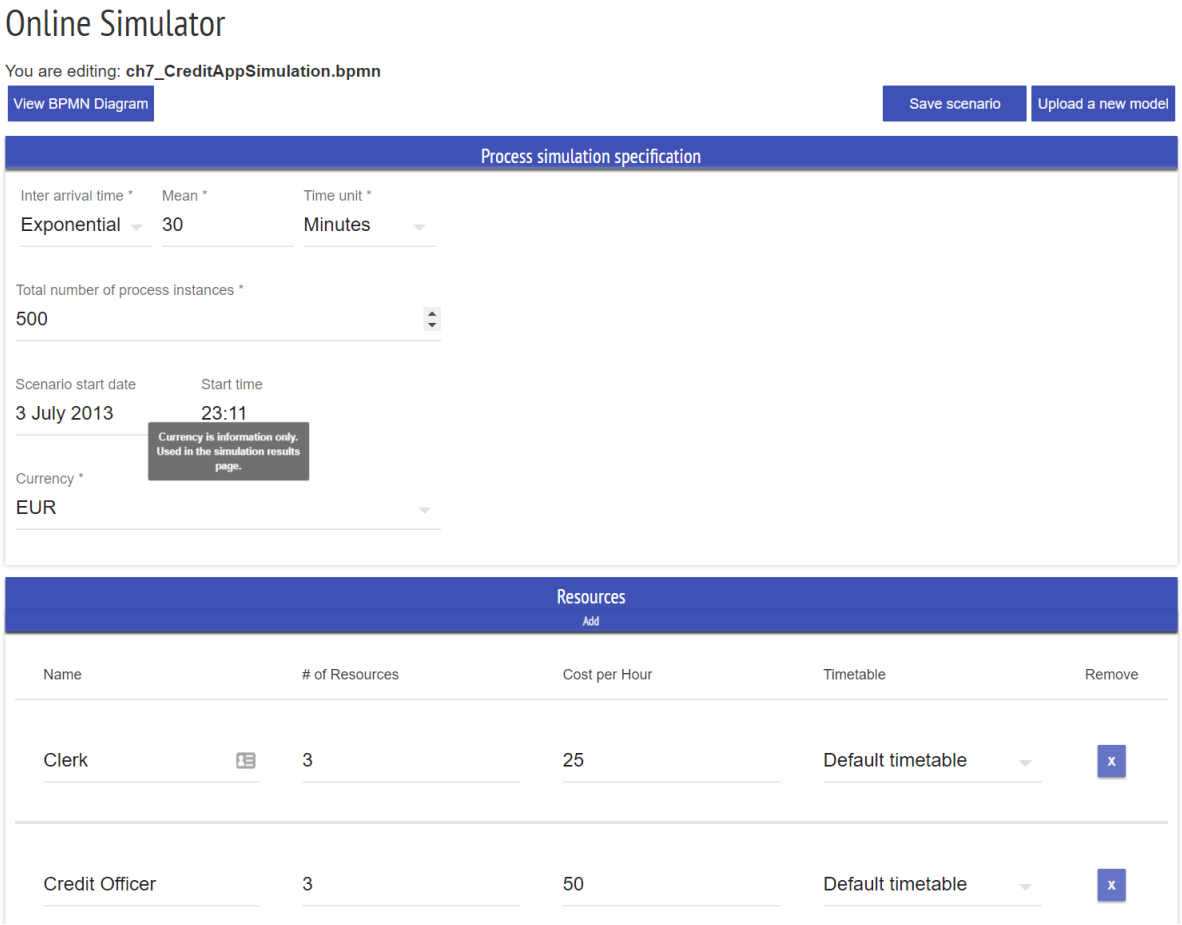



Figura 1.8: Esempio di manipolazione di parametri di simulazione in BIMP

### Sparx System Enterprise Architect

Sparx System Enterprise Architect è un'applicazione desktop enterprise che permette sia di costruire diagrammi BPMN, che simularli e utilizza la notazione BPSim. Si possono aggiungere quasi tutti i parametri presenti nello standard BPSim e, una volta avviata la simulazione, sono disponibili una serie di grafici utili per capire le criticità dei processi aziendali e i costi degli stessi. Non è purtroppo presente una versione web-based di questo software. È un ottimo prodotto software per chi vuole simulare processi aziendali, però non è gratuito: il costo per una singola licenza è di $499 [14].

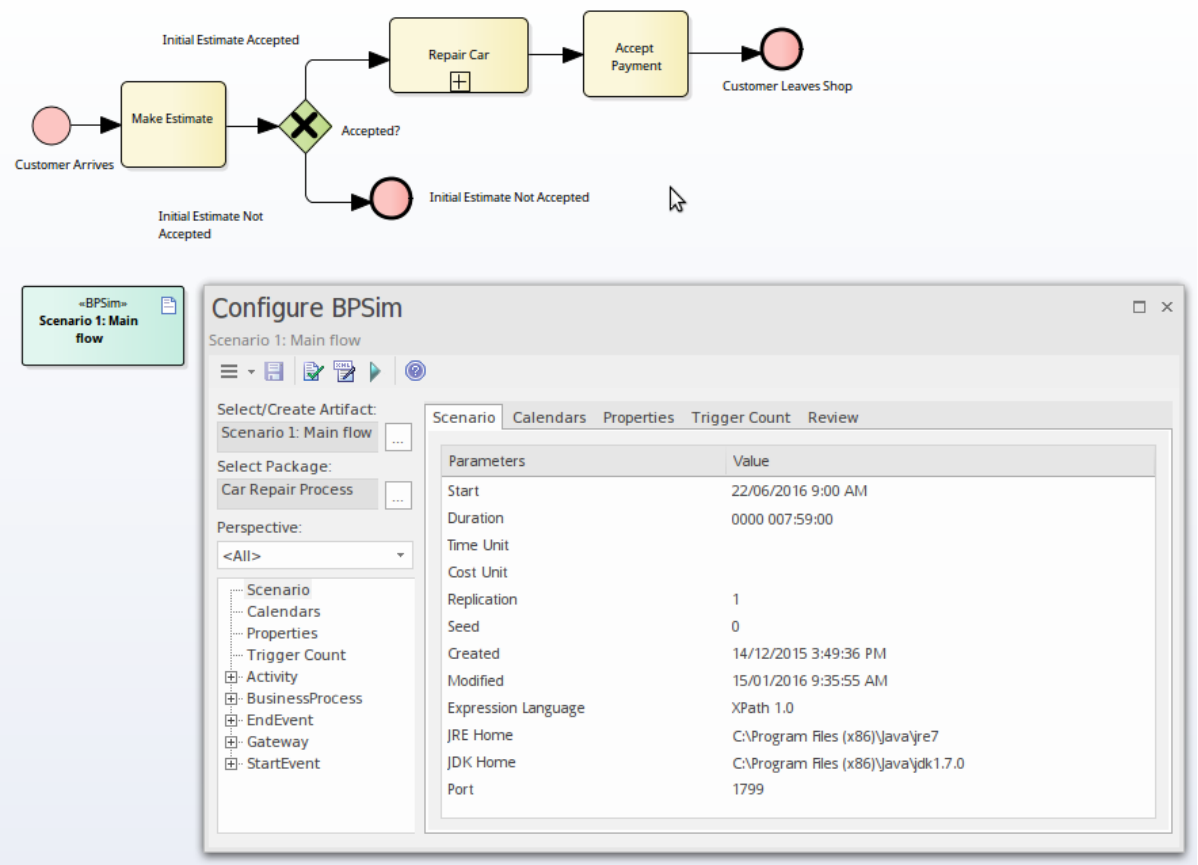



Figura 1.9: Esempio di manipolazione di parametri di simulazione in Sparx System Enterprise Architect

### Simul8

Simul8 è un'applicazione desktop enterprise che permette di costruire e simulare diagrammi BPMN e che segue la notazione BPSim. In Simul8 è possibile aggiungere parametri di simulazione, far partire la simulazione ed analizzarne i risultati. Il prodotto software ha tra i punti di forza sicuramente il fatto di essere uno strumento completo per chi vuole simulare processi BPMN, ma presenta anche alcune caratterische non eccellenti: solamente da poco è nata una versione web-based che non ha ancora tutte le features presenti nella versione desktop ed inoltre il costo per utilizzare Simul8 non è alla portata di tutti, basti pensare che, per utilizzare la versione "Basic" del software, occorre spendere $1995, mentre per la versione con una licenza per tre utenti distinti serve spendere $19995 [15].

### Visual Paradigm

Visual Paradigm è stato descritto in precedenza come un'applicazione desktop che permette la realizzazione di diagrammi BPMN, ma che ha anche altre features interessanti: una di queste è la possibilità di simulare diagrammi di processo BPMN.
In Visual Paradigm è possibile aggiungere una serie di parametri di simulazione per ogni elemento del diagramma, ma viene adottato uno standard deciso internamente anziché quello BPSim [16].
Sebbene questo strumento abbia un'interfaccia utente intuitiva ed è facile da usare, rispetto alla simulazione presenta ancora notevoli carenze, infatti non include gran parte delle proprietà di simulazione presenti, invece, nello standard BPSim. D'altro canto, i report forniti potrebbero essere considerati soddisfacenti, in quanto includono dati relativi al tempo di attesa di ciascuna attività, alla percentuale di utilizzo delle risorse e al costo del processo [17].

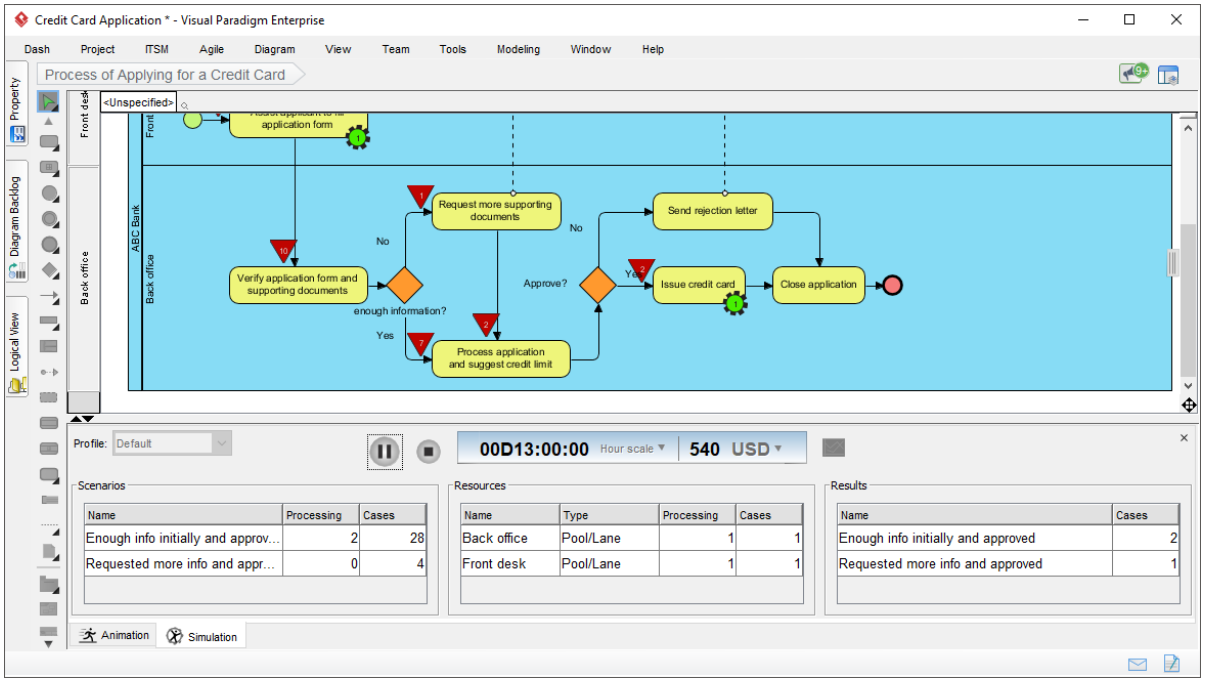



Figura 1.10: Esempio di manipolazione di parametri di simulazione in Visual Paradigm

### BP Simulator

BPSimulator è un simulatore web-based gratuito di modelli aziendali BPMN. Permette il caricamento di un diagramma BPMN e mostra, con una grafica particolarmente colorata, gli elementi di simulazione disponibili relativi al diagramma caricato. Come per BIMP, anche qui manca la possibilità di caricare diagrammi BPMN che presentano già parametri di simulazione e di salvare in un file con estensione *.bpmn* i parametri di simulazione aggiunti. Tra i lati positivi troviamo il fatto che sia di immediata comprensione e la possibilità di consultare interessanti analisi alla fine della simulazione [18].


Figura 1.11: Esempio di simulazione in BP Simulator

### Signavio

Signavio è stato descritto precedentemente come un tool che permette la realizzazione di diagrammi BPMN, ma questo strumento ne permette anche la simulazione, non rispettando lo standard BPSim.
Nella versione gratuita sono presenti due tipi di simulazione:

- **simulazione step-by-step**: permette di seguire la simulazione elemento per elemento e di focalizzarsi completamente sul flusso del processo;
- **simulazione one-case**: permette di simulare un caso specifico e di analizzarne i costi e i tempi.

La versione a pagamento presenta ulteriori tipi di simulazione che hanno come features aggiuntiva interessante la possibilità di esportare i risultati della simulazione [19].

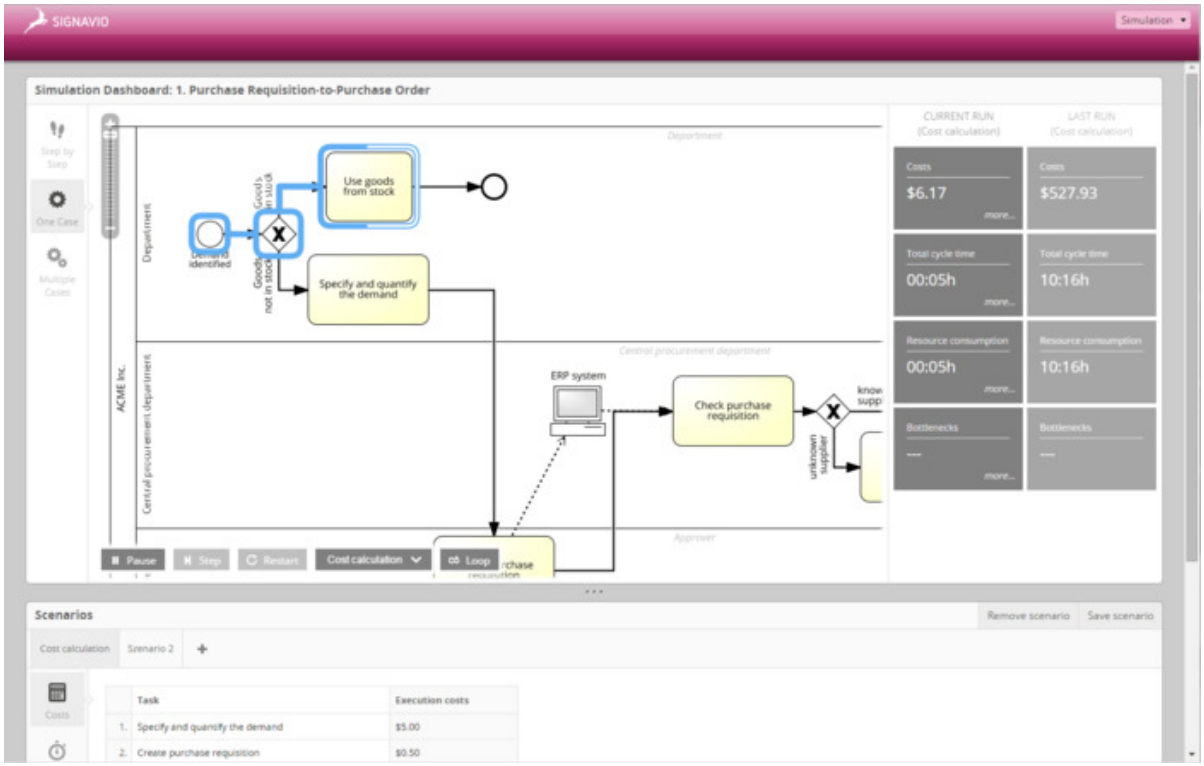



Figura 1.12: Esempio di manipolazione di parametri di simulazione in Signavio

### Adonis

Adonis è un tool che permette, tra le varie features, la simulazione di diagrammi BPMN, non rispettando lo standard BPSim. È possibile importare ed esportare il diagramma con i parametri di simulazione, ovviamente solo se prodotto con Adonis.
La simulazione, in Adonis, è di tipo event-based e tiene conto delle probabilità e delle frequenze inserite. Per specifiche query di simulazione è possibile definire i cosiddetti agenti di simulazione, ad esempio per la registrazione di cifre specifiche del cliente. I risultati della simulazione possono essere modificati in forma tabellare o in una vista grafica.
È presente una community di supporto per risolvere ogni tipo di problema relativo all'utilizzo del software [20].

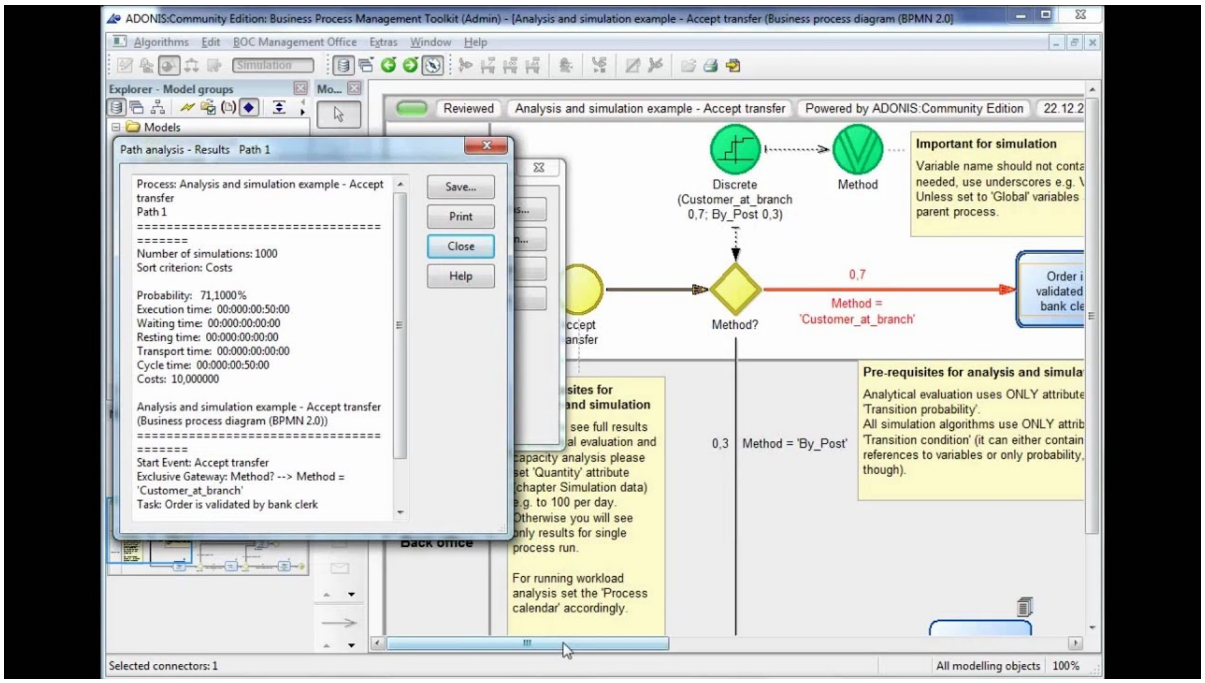



Figura 1.13: Esempio di simulazione in Adonis

### Bizagi

Bizagi è un tool che permette, tra le varie features, la creazione di diagrammi BPMN, ma anche la loro simulazione e rispetta lo standard BPSim. È possibile settare quasi tutti i parametri presenti nello standard ed è presente la possibilità di esportare i risultati in forma tabellare su Excel. Un'ulteriore nota positiva è data dal fatto che è possibile utilizzare il simulatore di Bizagi nella versione Bizagi Studio che non è a pagamento [21]. Uno dei vantaggi evidenziati dagli sviluppatori di Bizagi è la funzione "What-if Analysis". Questa funzione consente agli utenti di "clonare" qualsiasi scenario sviluppato e modificare i parametri in cui vengono identificati i colli di bottiglia nel processo. Il report prodotto successivamente confronta gli scenari simulati e identifica quali sono gli elementi che sono cambiati, permettendo così di valutare l'impatto delle modifiche apportate nel modello di processo [17].

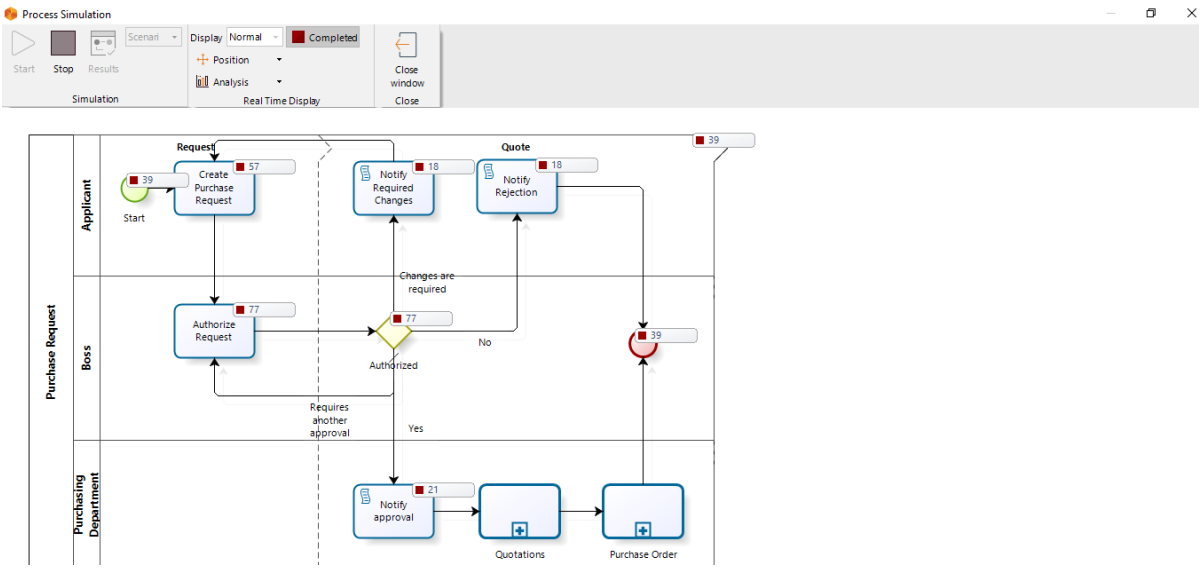



Figura 1.14: Esempio di simulazione in Bizagi

### Trisotech

Trisotech è un prodotto software enterprise che permette sia la creazione di diagrammi BPMN, ma anche la loro simulazione e rispetta lo standard BPSim. Con la funzione di simulazione, è possibile selezionare scenari predefiniti e vedere come funzionerà il diagramma con i parametri indicati, oppure si possono anche creare i parametri personalizzati se si ha bisogno di un processo di simulazione più avanzato [22].

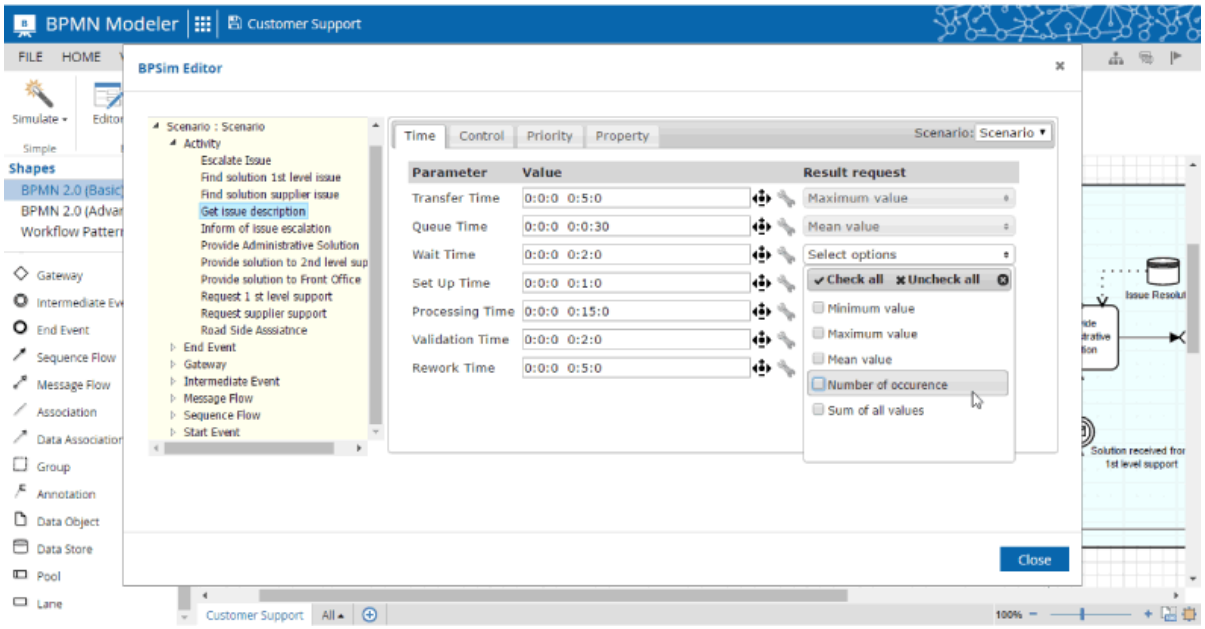



Figura 1.15: Esempio di simulazione in Trisotech

### BonitaSoft

Lo strumento BonitaSoft consente la rappresentazione grafica dei processi aziendali in BPMN, ma per quanto riguarda la simulazione dei modelli di processo, BonitaSoft è ancora in una fase di sviluppo molto precoce, poiché include solo la definizione delle risorse e il caricamento dei profili, sebbene usi la notazione BPSim. Un'altra limitazione di questo strumento riguarda l'assenza di informazioni sulle code. Dal momento che non ci sono capacità di animazione grafica, come attualmente accade con la maggior parte degli strumenti di simulazione, non si può, anche visivamente, identificare le attività che hanno code più grandi. Anche i report ottenuti non ne producono informazioni su questa situazione, rendendo molto difficile identificare i colli di bottiglia che possono esistere in

un processo di business. Sebbene questo strumento fornisca un modulo di simulazione, l'impossibilità di ottenere i dati necessari per una corretta analisi del comportamento dei processi il suo reale utilizzo è molto limitato [17].

# Capitolo 2

# Progettazione

In questo capitolo si presentano le specifiche per lo sviluppo di una Desktop Progressive Web Application che permette la modellazione dinamica dei parametri di simulazione dei processi aziendali. Vengono poi definiti gli obiettivi che tale software deve soddisfare e i prodotti con i quali necessariamente occorre interagire. Successivamente si elencano i vincoli stabiliti sui diagrammi BPMN in input (con eventuale notazione già presente BPSIM) per questa tipologia di applicativo e il tipo di output prodotto dalla Desktop PWA. Infine sono descritte le tipologie di utenti per le quali l'applicativo è stato pensato e come questi vi si possono interfacciare.

## 2.1 Obiettivi

L'applicazione realizzata in questo progetto di tesi ha come obiettivo quello di poter permettere la manipolazione di parametri di simulazione su un qualunque diagramma con notazione BPMN. In particolare deve essere possibile, dato un diagramma BPMN, con o senza notazione BPSim:

- leggere diagrammi BPMN con notazione BPSim che rispettano le specifiche BPMN e BPSim;
- aggiungere nuovi parametri di simulazione;
- modificare parametri di simulazione;
- eliminare parametri di simulazione;
- generare un file *.bpmn* con i parametri di simulazione aggiornati.

Per realizzare questi obiettivi, occorre creare una struttura dati di supporto, che sia in grado di salvare i continui cambiamenti di stato dell'interfaccia grafica della PWA e che sia consistente con ciò che viene mostrato graficamente.

## 2.2 Prodotti utilizzati

### 2.2.1 TypeScript

L'intera struttura dati è stata realizzata utilizzando TypeScript.
TypeScript estende la sintassi di JavaScript, in questo modo qualunque programma scritto in JavaScript è anche in grado di funzionare con TypeScript senza nessuna modifica. È stato progettato per lo sviluppo di grandi applicazioni e viene successivamente ricompilato in JavaScript per poter essere interpretato da web browser o app [23].

```typescript
class Greeter {
    greeting: string;
    constructor(message: string) {
        this.greeting = message;
    }
    greet() {
        if (this.greeting === "carl") {
            return "Hello, son";
        } else {
            return "Hello, " + this.greeting;
        }
    }
}

var greeter = new Greeter("carl");
```

```javascript
var Greeter = (function () {
    function Greeter(message) {
        this.greeting = message;
    }
    Greeter.prototype.greet = function () {
        if (this.greeting === "carl") {
            return "Hello, son";
        }
        else {
            return "Hello, " + this.greeting;
        }
    };
    return Greeter;
})();
var greeter = new Greeter("carl");
```

Figura 2.1: Esempio di "traduzione" di codice TypeScript in codice JavaScript

TypeScript è un linguaggio che estende la base esistente di JavaScript aggiungendo alcune features, tra le quali le più importanti sono:

- tipi di dato;
- tipo enum;
- firma dei metodi;
- classi;
- interfacce;
- moduli.

Nello specifico, per la realizzazione di questo progetto di tesi, sono state sfruttate, tra le features aggiuntive di TypeScript, le classi, il tipo enum e il poter tipare in maniera esplicita i dati e ciò ha reso la struttura dati più consistente e più facile da debuggare.

### 2.2.2 Npm

Npm è un gestore di pacchetti per il linguaggio di programmazione JavaScript. È il gestore di pacchetti predefinito per l'ambiente di runtime JavaScript Node.js. Consiste in un client da linea di comando, chiamato anch'esso npm, e un database online di pacchetti pubblici e privati, chiamato npm registry. Il registry è accessibile via client e i pacchetti disponibili sono consultabili sul sito web di npm [24].
In questo elaborato npm è stato utilizzato, perché ha permesso l'installazione di vari pacchetti che hanno permesso la realizzazione sia dell'interfaccia web che la modellazione della struttura dati dinamica. Per quanto riguarda, nello specifico, la realizzazione della struttura dati, ecco una lista dei pacchetti installati utilizzati:

- **typescript**: compilatore per file TypeScript;
- **ts-loader**: loader per i file TypeScript;
- **xml-loader**: plugin che permette di caricare i file XML;
- **mocha**: test framework per JavaScript;
- **chai**: assertion library;
- **nyc**: tool che permette di verificare la test coverage;
- **vkbeautify**: plugin JavaScript che abbellisce e/o minifica il testo di file XML, JSON, CSS e SQL.

### 2.2.3 Mocha

Mocha è un framework di test JavaScript, che permette test asincroni e rapporti sulla coverage dei test. Scrivere test, però, spesso richiede l'uso di una assertion library (librerie che permettono di verificare che che i test svolti siano corretti) e Mocha permette l'utilizzo di una qualsiasi di queste library. Se si utilizza Mocha in un ambiente Node.js, è possibile utilizzare il modulo di assert incorporato come libreria di asserzioni, ma anche altre librerie di asserzioni più estese come Chai, Expect.js e Should.js [25]. Nello specifico in questo progetto si è deciso di utilizzare il framework Mocha per il test e l'assertion library Chai, che aggiunge ulteriori funzioni rispetto all'assertion library di default [26].

Per questo elaborato, Mocha e Chai sono stati utilizzati per testare la robustezza della struttura dati dinamica. Sono stati svolti, infatti, vari test che hanno permesso la realizzazione di una struttura dati solida in grado di dare supporto e consistenza alla parte grafica della Desktop PWA.

## 2.3 Requisiti

### 2.3.1 Vincoli sul diagramma in input

Nel progetto di tesi sono state prese una serie di decisioni progettuali sui vincoli da imporre al diagramma con notazione BPMN e BPSim che viene dato in pasto alla Desktop PWA. Naturalmente, questi vincoli sono stati assunti rispettando la documentazione presente e l'XML schema della notazione di simulazione BPSim. Il funzionamento dell'intero sistema non è garantito, quindi, in caso di mancato rispetto dei vincoli che seguono:

- il diagramma deve avere una sintassi che rispetta la notazione BPMN;
- il diagramma deve eventualmente avere una sintassi che rispetta la notazione BPSim;
- l'ID è univoco, quindi non possono esistere elementi distinti che condividono lo stesso ID;
- non è possibile avere uno *Scenario* senza ID;
- non è possibile avere un *Calendar* senza ID;
- non è possibile, nel campo *inherits*, avere l'ID dello scenario corrente;
- non è possibile, nel campo *inherits*, avere un ID inesistente;
- non è possibile avere come valore del campo *validFor* di un qualunque *Parameter* un ID di un *Calendar* che non esiste nello scenario in cui si trova il *Parameter*;
- non è possibile trovare valori non numerici nei campi di tipo numerico e valori non booleani nei campi di tipo booleano;
- non è possibile avere valori nei campi di tipo enumerato diversi da quelli permessi nella specifica;
- non è possibile avere un campo di un Parameter Type ripetuto più volte;
- per determinati elementi è possibile inserire solo alcuni parametri come nelle figure: fig 2.2, fig 2.3, fig 2.4, fig 2.5, fig 2.6 e fig 2.7.

| | | Time Parameters | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | transferTime | queueTime | waitTime | setupTime | processingTime | validationTime | reworkTime |
| Events | Start Event | No - BPMN Events map to time point and thus cannot have Time Parameters which are time intervals | | | | | | |
| | Intermediate Event | | | | | | | |
| | End Event | | | | | | | |
| Activities | Task | **Yes** | | | | | | |
| | Sub Process | **Yes - but only for activities without decomposition** | | | | | | |
| | Transaction | | | | | | | |
| | Call Activity | | | | | | | |
| | Event Sub Process | | | | | | | |
| Gateways | Gateway | No - BPMN Gateways do not map to time intervals as they are only visualizations of branching logic | | | | | | |
| Connecting Objects | Sequence Flow | No - BPMN Connecting Objects do not have time interval associated to them | | | | | | |
| | Message Flow | | | | | | | |
| | Data Association | | | | | | | |
| | Association | | | | | | | |
| Data | Data Object | No - BPMN Data Elements have no impact on the Simulation or the Analysis of the Process | | | | | | |
| | Data Store | | | | | | | |
| Swimlanes | Lane | No - BPMN Swimlanes have no impact on the Simulation or the Analysis of the Process | | | | | | |
| | Pool | | | | | | | |
| Artifacts | Artifact | No - BPMN Artifacts have no impact on the Simulation or the Analysis of the Process | | | | | | |
| Attributes | ResourceRole | No | | | | | | |
| | Resource | | | | | | | |

Figura 2.2: Applicabilità di TimeParameters agli elementi BPMN

| | | Control Parameters | | | |
|---|---|---|---|---|---|
| | | interTriggerTimer | triggerCount | probability | condition |
| Events | Start Event | **Yes** | **Yes** | **Yes - but inside Event Sub Process only** | |
| | Intermediate Event | **Yes - but for Catch Event only** | No | **Yes - but for Boundary Event only** | |
| | End Event | No | | | |
| Activities | Task | **Yes - but only for activities without decomposition without incoming sequence flow** | | No | |
| | Sub Process | | | | |
| | Transaction | | | | |
| | Call Activity | | | | |
| | Event Sub Process | **Yes - but only for Event Sub Process without decomposition** | | | |
| Gateways | Gateway | **Yes - but only for Event Based Gateway starting a process** | | No | |
| Connecting Objects | Sequence Flow | No | | **Yes** | |
| | Message Flow | | | No | |
| | Data Association | | | | |
| | Association | | | | |
| Data | Data Object | No - BPMN Data Elements have no impact on the Simulation or the Analysis of the Process | | | |
| | Data Store | | | | |
| Swimlanes | Lane | No - BPMN Swimlanes have no impact on the Simulation or the Analysis of the Process | | | |
| | Pool | | | | |
| Artifacts | Artifact | No - BPMN Artifacts have no impact on the Simulation or the Analysis of the Process | | | |
| Attributes | ResourceRole | No | | | |
| | Resource | | | | |

Figura 2.3: Applicabilità di ControlParameters agli elementi BPMN

| | | Resource Parameters | | | |
|---|---|---|---|---|---|
| | | availability | quantity | role | selection |
| Events | Start Event | No | | | |
| | Intermediate Event | | | | |
| | End Event | | | | |
| Activities | Task | No | | | |
| | Sub Process | | | | |
| | Transaction | | | | |
| | Call Activity | | | | |
| | Event Sub Process | | | | |
| Gateways | Gateway | No | | | |
| Connecting Objects | Sequence Flow | No | | | |
| | Message Flow | | | | |
| | Data Association | | | | |
| | Association | | | | |
| Data | Data Object | No - BPMN Data Elements have no impact on the Simulation or the Analysis of the Process | | | |
| | Data Store | | | | |
| Swimlanes | Lane | No - BPMN Swimlanes have no impact on the Simulation or the Analysis of the Process | | | |
| | Pool | | | | |
| Artifacts | Artifact | No - BPMN Artifacts have no impact on the Simulation or the Analysis of the Process | | | |
| Attributes | ResourceRole | No | | | **Yes** |
| | Resource | **Yes** | | | No |

Figura 2.4: Applicabilità di ResourceParameters agli elementi BPMN

| | | Cost Parameters | |
|---|---|---|---|
| | | fixedCost | unitCost |
| Events | Start Event | No | |
| | Intermediate Event | | |
| | End Event | | |
| Activities | Task | **Yes** | |
| | Sub Process | | |
| | Transaction | | |
| | Call Activity | | |
| | Event Sub Process | | |
| Gateways | Gateway | No | |
| Connecting Objects | Sequence Flow | No | |
| | Message Flow | | |
| | Data Association | | |
| | Association | | |
| Data | Data Object | No - BPMN Data Elements have no impact on the Simulation or the Analysis of the Process | |
| | Data Store | | |
| Swimlanes | Lane | No - BPMN Swimlanes have no impact on the Simulation or the Analysis of the Process | |
| | Pool | | |
| Artifacts | Artifact | No - BPMN Artifacts have no impact on the Simulation or the Analysis of the Process | |
| Attributes | ResourceRole | No | |
| | Resource | **Yes** | |

Figura 2.5: Applicabilità di CostParameters agli elementi BPMN

| | | Property Parameters | |
|---|---|---|---|
| | | property | queueLength |
| Events | Start Event | **Yes** | No |
| | Intermediate Event | | |
| | End Event | | |
| Activities | Task | **Yes** | **Yes** |
| | Sub Process | | **Yes** – But only for activities without decompositions |
| | Transaction | | |
| | Call Activity | | |
| | Event Sub Process | | |
| Gateways | Gateway | No | |
| Connecting Objects | Sequence Flow | **Yes** | No |
| | Message Flow | | |
| | Data Association | No | |
| | Association | | |
| Data | Data Object | No - BPMN Data Elements have no impact on the Simulation or the Analysis of the Process | |
| | Data Store | | |
| Swimlanes | Lane | No - BPMN Swimlanes have no impact on the Simulation or the Analysis of the Process | |
| | Pool | | |
| Artifacts | Artifact | No - BPMN Artifacts have no impact on the Simulation or the Analysis of the Process | |
| Attributes | ResourceRole | No | |
| | Resource | | |

Figura 2.6: Applicabilità di PropertyParameters agli elementi BPMN

| | | Priority Parameters | |
|---|---|---|---|
| | | interruptible | priority |
| Events | Start Event | No | |
| | Intermediate Event | | |
| | End Event | | |
| Activities | Task | **Yes** | |
| | Sub Process | **Yes - but only for activities without decomposition** | |
| | Transaction | | |
| Gateways | Gateway | No | |
| Connecting Objects | Sequence Flow | No | |
| | Message Flow | | |
| | Data Association | | |
| | Association | | |
| Data | Data Object | No - BPMN Data Elements have no impact on the Simulation or the Analysis of the Process | |
| | Data Store | | |
| Swimlanes | Lane | No - BPMN Swimlanes have no impact on the Simulation or the Analysis of the Process | |
| | Pool | | |
| Artifacts | Artifact | No - BPMN Artifacts have no impact on the Simulation or the Analysis of the Process | |
| Attributes | ResourceRole | No | |
| | Resource | | |

Figura 2.7: Applicabilità di PriorityParameters agli elementi BPMN

### 2.3.2 Vincoli sul diagramma in output

Una volta terminata la manipolazione dei parametri di simulazione, è possibile generare un file in output con estensione *.bpmn* che contiene la specifica dei parametri di simulazione. I vincoli che avrà tale file sono i seguenti:

- il campo *vendor* di *Scenario* avrà un valore predefinito, ovvero i cognomi dei due sviluppatori del progetto di tesi, poiché viene imposto che un qualunque utilizzo della Desktop PWA porti ad avvalorare in questa maniera il suddetto campo;
- non è possibile che il campo *VendorExtension* di *Scenario* sia avvalorato, poiché non sono previste estensioni dello stesso da parte degli sviluppatori, ma la sua eventuale presenza nel file di input viene conservata in quello di output;
- non è possibile che il campo *result* di *Scenario* sia avvalorato (se non già avvalorato nel file di partenza), poiché questo campo ha senso che abbia un valore solo in caso di simulazioni effettuate in precedenza, con il salvataggio delle stesse già completato;
- sono presenti alcuni valori di default (se non sovrascritti da valori custom dell'utente) in determinati campi, come scritto nella documentazione ufficiale di BPSim;
- non sono presenti quei *Parameter* che hanno solo graficamente avvalorato il campo *ResultRequest*, ma che non presentano alcun *Value*;
- non saranno prodotti nel file finale i commenti eventualmente presenti nel file dato in input. Questo avviene poiché i commenti impedivano il parsing corretto degli elementi di simulazione e non risultato fondamentali ai fini dello sviluppo di questo progetto di tesi.

### 2.3.3 Front-end della Desktop PWA

Come già detto in precedenza, la parte oggetto di questa tesi, ovvero la manipolazione di una struttura dati dinamica, viene utilizzata da una Desktop PWA, che permette l'effettiva manipolazione degli elementi di simulazione. Grazie, infatti, alla parte grafica, è possibile settare tutti i parametri di simulazione.
La pagina mostrata all'utente finale è divisa in due parti principali:

1. la parte a sinistra che permette la visualizzazione del diagramma BPMN con i relativi elementi del processo;

2. la parte a destra relativa alla simulazione.

Tra le funzionalità principali della pagina troviamo:

- è possibile cliccare sugli elementi del processo per giungere direttamente alla sezione della simulazione dedicata ad essi;

- è possibile muoversi sul diagramma BPMN e zoommare sulla zona del diagramma;

- si può navigare la sezione dedicata alla simulazione direttamente senza passare dagli elementi del diagramma;

- è possibile selezionare un solo scenario per volta da visualizzare nella sezione dedicata ai parametri di simulazione;

- è possibile generare il file finale grazie ad un apposito bottone.

La relazione che intercorre tra la parte grafica appena descritta e l'elaborato oggetto di questo prodotto di tesi consiste nel salvataggio della modifica dei parametri mostrati all'utente nella struttura dati dinamica. Per i *Parameter*, questo salvataggio avviene nel momento in cui un utente cambia lo *Scenario* che sta visualizzando, o decide di generare il file di output finale, mentre per il salvataggio degli *ID* di alcuni elementi, il salvataggio avviene istantaneamente.
Questo salvataggio di elementi nella struttura dati è molto utile nel momento in cui viene cambiato lo *Scenario* che si vuole visualizzare: in quel momento i dati immessi vengono salvati, affinché nel caso in cui l'utente voglia ritornare sullo *Scenario* appena cambiato, egli potrà ritrovare tutti i parametri settati. La pagina, infatti, viene ripopolata grazie ai dati salvati nella struttura dati dinamica. È evidente il vantaggio che porta avere uno strumento che fa da "ponte" tra il file dato in input al sistema e ciò che viene visualizzato graficamente.
Di seguito, nelle figure 2.8 e 2.9, vi è un esempio della pagina della Desktop PWA.

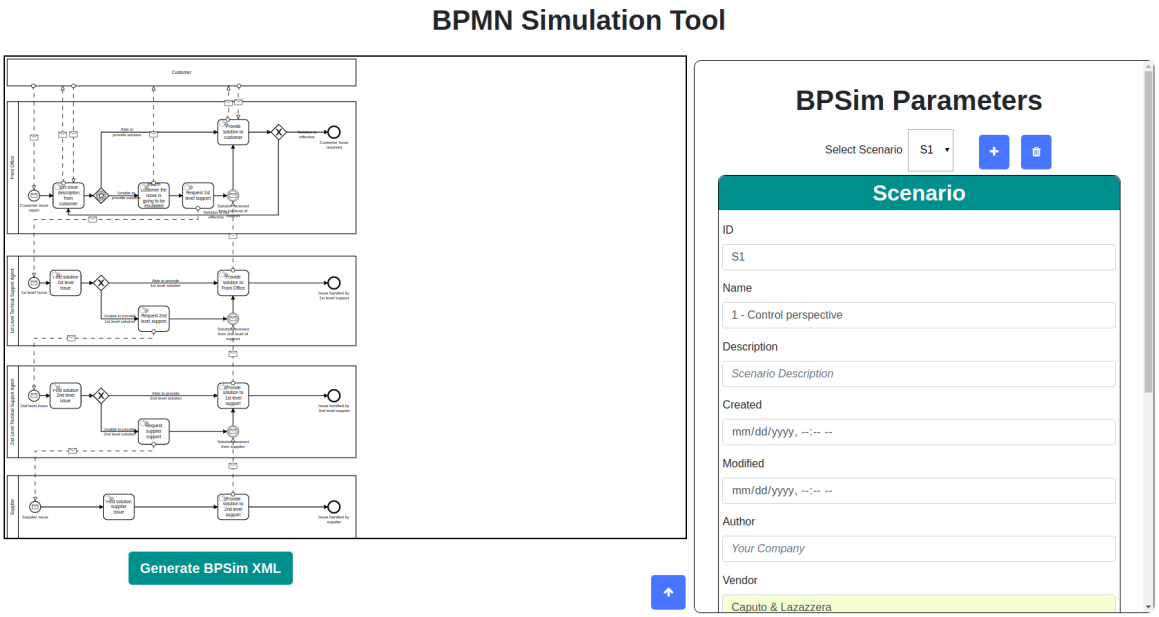



Figura 2.8: Pagina della Desktop PWA - parte 1

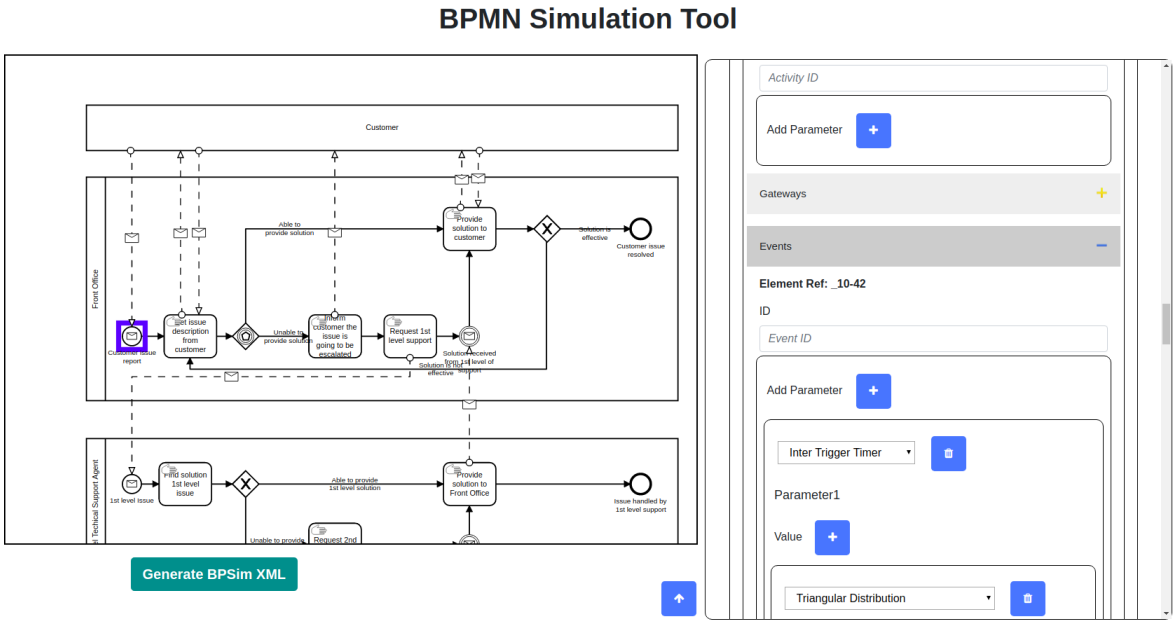



Figura 2.9: Pagina della Desktop PWA - parte 2

## 2.4 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 modellatore dei parametri di simulazione di processi definiti utilizzando BPMN e BPSim. L'essere conformi a uno standard internazionale riconosciuto consente a questo prodotto di essere in grado di interfacciarsi, senza particolari problemi, con qualsiasi altro software che rispetti gli standard sopracitati. Ad esempio, l'output prodotto da questa Desktop PWA, potrebbe essere l'input da dare in pasto ad un simulatore, purché ovviamente esso rispetto gli standard BPMN e BPSim.

# Capitolo 3

# Realizzazione

Si è deciso di sviluppare la Desktop PWA utilizzando HTML, CSS, JavaScript, JQuery e TypeScript.
Nello specifico, per la parte relativa a questo progetto di tesi, ovvero la creazione di una modellazione dinamica di parametri di simulazione, sono stati utilizzati principalmente TypeScript, per la creazione vera e propria della struttura dati, e JQuery, per l'interfacciamento con la parte grafica. Come gestore di pacchetti si è utilizzato npm 6.10.1 e tutte le dipendenze al progetto possono essere trovate all'interno del file package.json, nella root del progetto.

## 3.1 Architettura

Per utilizzare la Desktop PWA occorre in un primo momento caricare un diagramma BPMN, che eventualmente può contenere anche già una parte di notazione di simulazione BPSim. Una volta caricato il file, l'interfaccia grafica mostrerà il diagramma BPMN ed i parametri di simulazione nelle loro rispettive sezioni. Tutto ciò è reso possibile dalla struttura dati dinamica che salva tutte le informazioni lette dal diagramma in input.
La struttura dati e il front-end sono in continua comunicazione durante l'esecuzione dell'applicazione, poiché ogni modifica che l'utente va a fare sul lato grafico deve essere prontamente salvata nella struttura dati. Questo passaggio è estremamente importante dato che, in un qualunque momento, l'utente può decidere di generare il file finale contenente i parametri di simulazione aggiunti: questo file viene prodotto, infatti, andando ad analizzare tutti i campi avvalorati della struttura dati.
La figura 3.1 rappresenta l'intera architettura della Desktop PWA appena descritta.

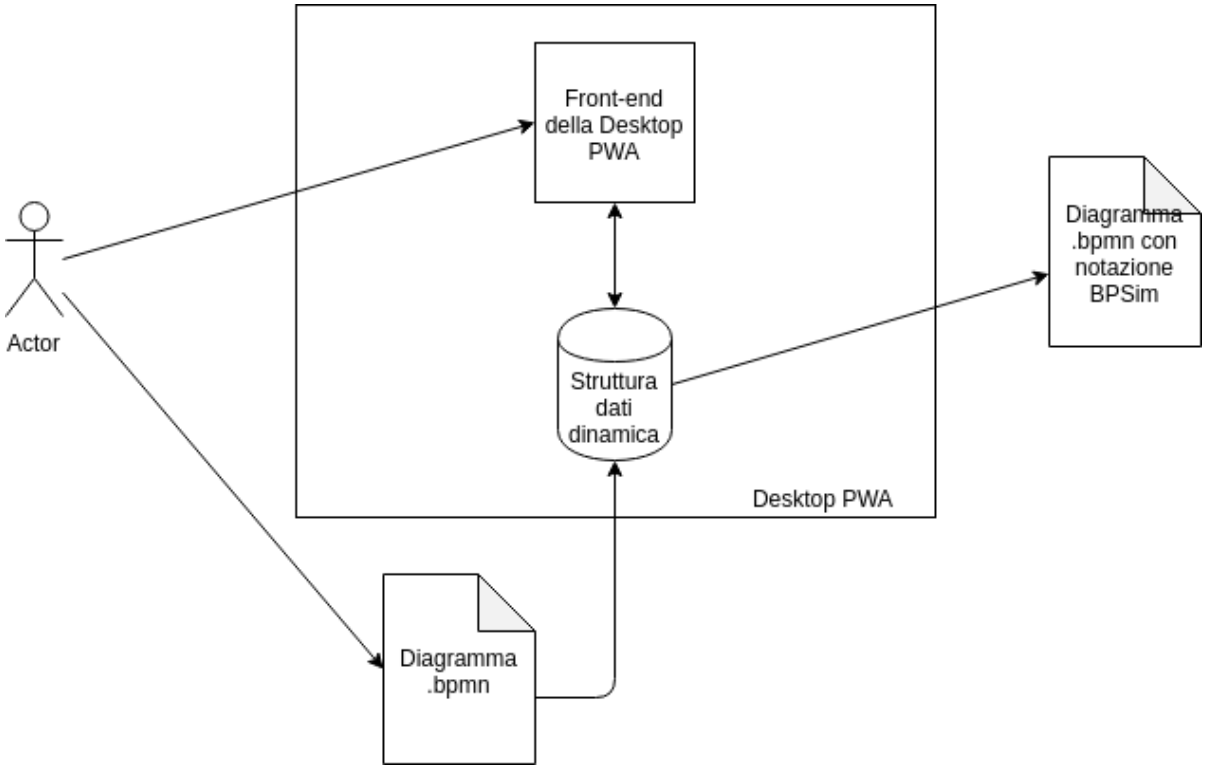



Figura 3.1: Architettura della Desktop PWA

## 3.2 Descrizione della struttura dati

L'intera struttura dati presenta una forma ad albero ed è stata realizzata in TypeScript, poiché si è ritenuto di fondamentale importanza sfruttare delle feature che TypeScript aggiunge rispetto a JavaScript, ovvero la possibilità di esplicitare i tipi e il definire classi esplicitamente.
Di seguito viene illustrato il diagramma delle classi, che rispecchia perfettamente il modo con il quale è stata realizzata la struttura dati.
La figura 3.2, riportata di seguito, rappresenta la parte superiore dell'albero: si parte dalla classe *BPSimData* che rappresenta il nodo radice che fa da collettore di tutti gli elementi della classe *Scenario*. In ogni *Scenario*, oltre ad una serie di attributi di tipo semplice, sono presenti *n* elementi di tipo *ElementParameters*, nello specifico tanti quanti sono gli elementi grafici (task, eventi, gateway, ecc.) presenti nel diagramma BPMN ai quali si vuole aggiungere parametri di simulazione e un elemento di tipo *ScenarioParameters*.

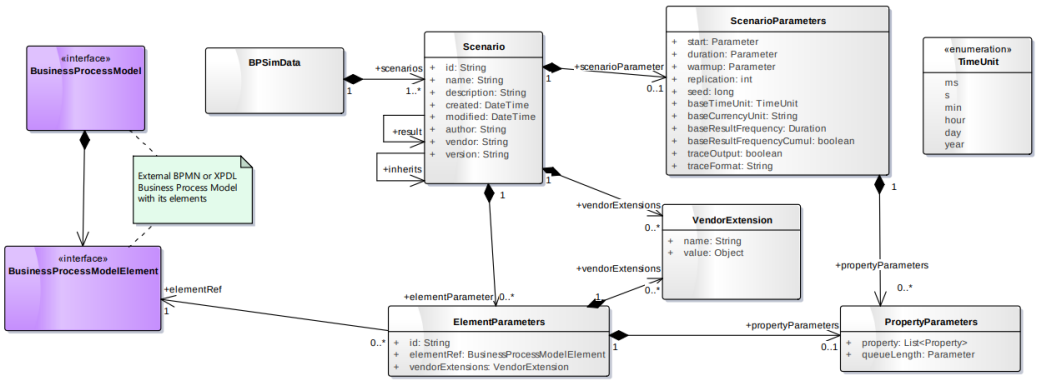



Figura 3.2: Diagramma delle classi in relazione alla classe Scenario

La figura 3.3, riportata di seguito, mostra la struttura della classe *ElementParameters*, ovvero la classe che raccoglie tutte le informazioni relative agli elementi del diagramma BPMN. Oltre all'*id* e all'attributo *elementRef*, che si riferisce al nome dell'elemento in BPMN, è possibile trovare (non obbligatoriamente) una serie di elementi di tipo complesso, che raggruppano parametri accomunati da determinate caratteristiche, come *TimeParameters, ControlParameters, ResourceParameters, CostParameters, PropertyParameters* e *PriorityParameters*.

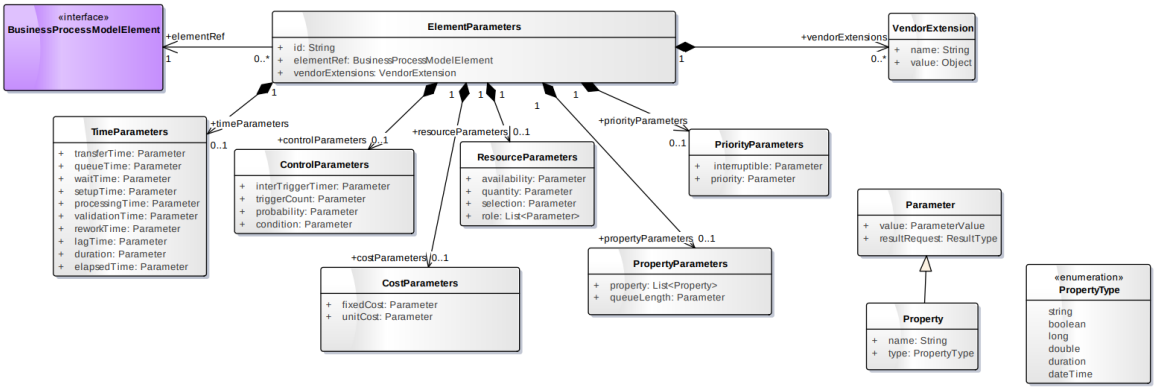



Figura 3.3: Diagramma delle classi in relazione alla classe ElementParameters

A loro volta, queste ultime classi elencate, hanno dei campi di tipo *Parameter*, la cui struttura è mostrata nella figura 3.4. Un elemento di tipo *Parameter* ha un attributo di tipo *ResultType* e più attributi di tipo *ParameterValue*. Per essere più precisi, non viene mai utilizzata propriamente la classe *ParameterValue*, poiché ci si serve di classi figlie della stessa, ovvero le classi *ConstantParameter, DistributionParameter, ExpressionParameter* ed *EnumParameter*.

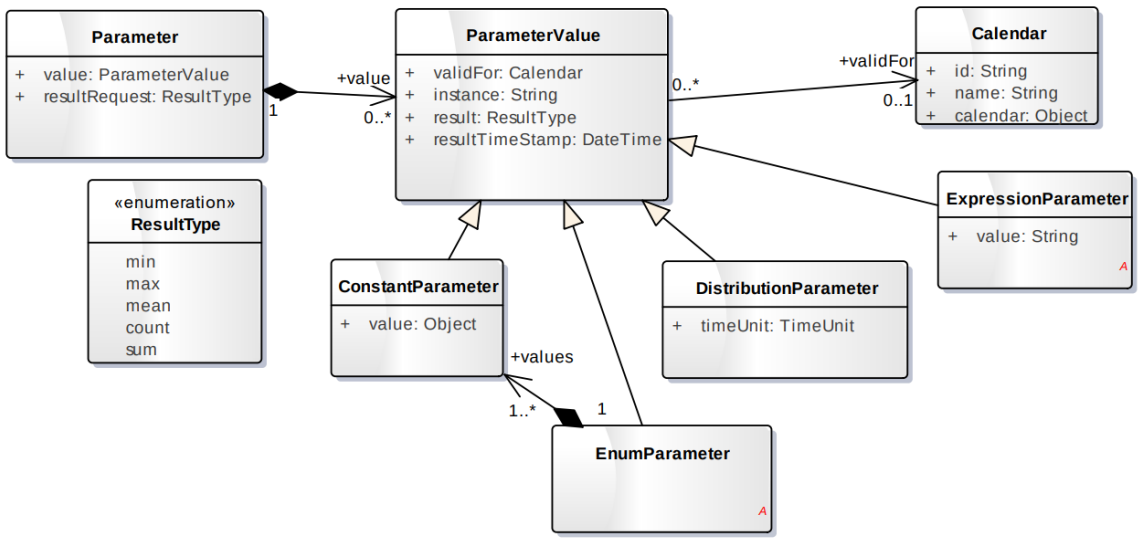



Figura 3.4: Diagramma delle classi in relazione alla classe Parameter

Due delle quattro classi appena citate si risolvono in una delle corrispettive classi figlie, come mostrato nella figura 3.5 per la classe *ConstantParameter* e nella figura 3.6 per la classe *DistributionParameter*.
La classe *EnumParameter*, invece, si risolve come una collection di elementi di tipo *ConstantParamater*, mentre la classe *ExpressionParameter* è già "completa".

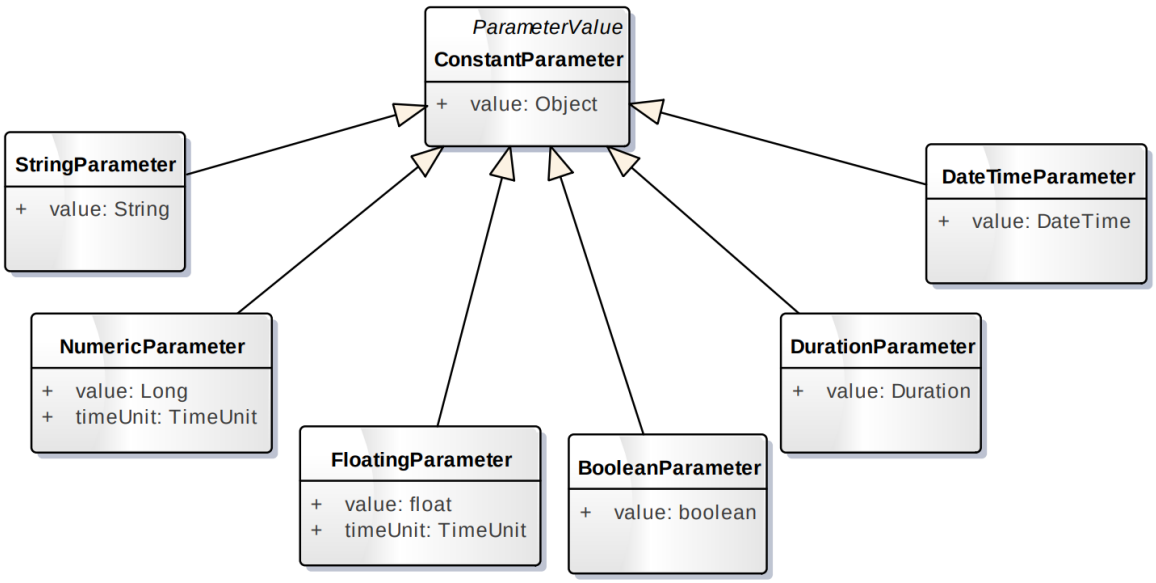



Figura 3.5: Diagramma delle classi in relazione alla classe ConstantParameter

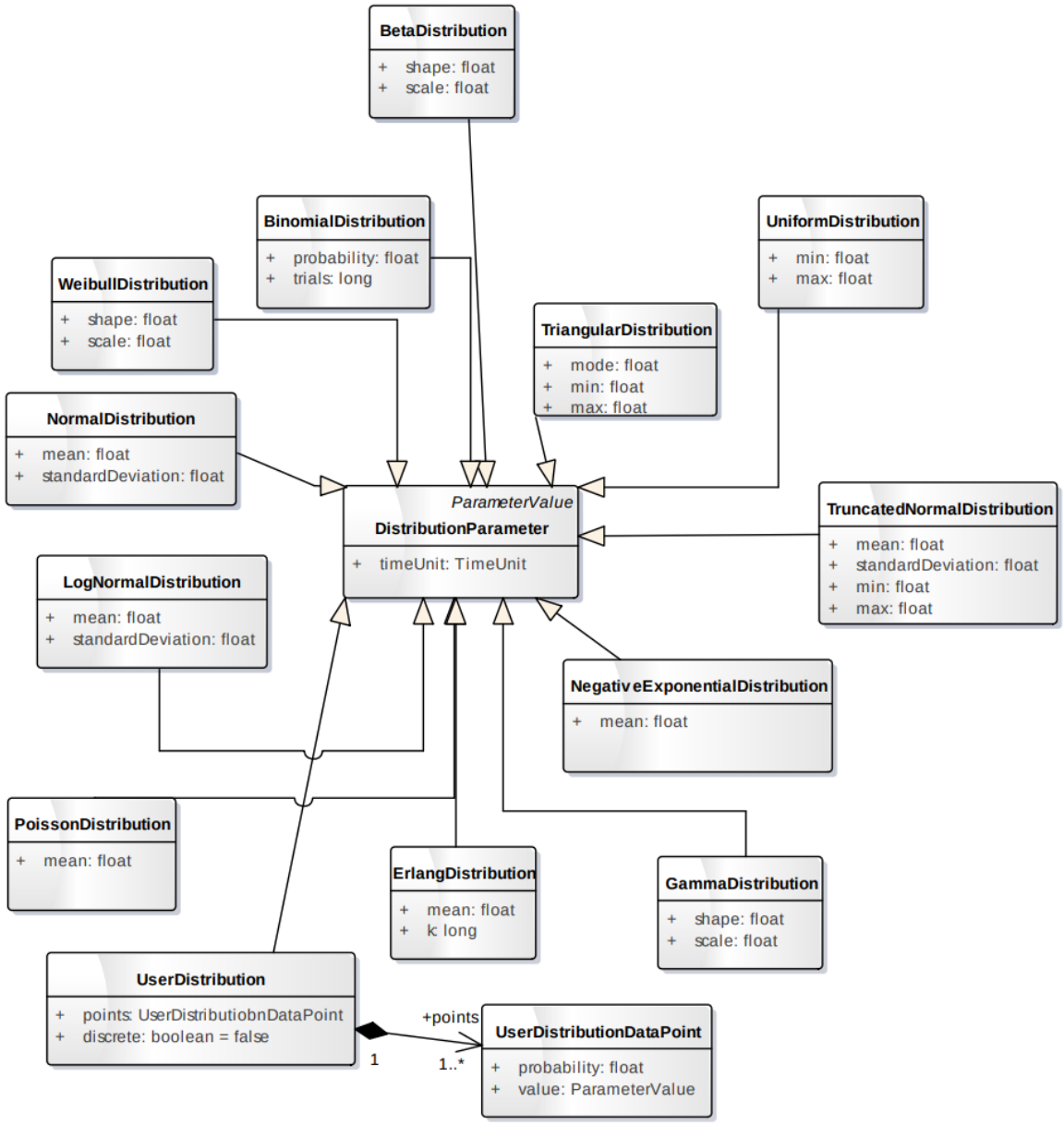



Figura 3.6: Diagramma delle classi in relazione alla classe DistributionParameter

## 3.3 Salvataggio degli elementi del diagramma in input in una struttura dati

La struttura dati appena descritta deve essere opportunamente avvalorata non appena l'utente carica il diagramma BPMN. Ovviamente ciò avviene solo nel caso in cui siano presenti dei parametri di simulazione nel file caricato, altrimenti verrà creata una struttura dati vuota, pronta per essere avvalorata.
Se, quindi, sono presenti dei parametri di simulazione, occorre individuare nel file *.bpmn* la sezione relativa ad essi, ovvero tutto ciò che è compreso nei tag, come nell'esempio che segue:

```
<semantic:relationship type="BPSimData">
    <semantic:extensionElements>
        ...     // Parametri di simulazione
    </semantic:extensionElements>
</semantic:relationship>
```

Fatto ciò, viene parsata ogni riga compresa tra questi tag e vengono eliminati i commenti, dato che non forniscono informazioni utili per la creazione degli elementi di simulazione. In seguito viene popolata ricorsivamente la struttura dati: si parte dagli elementi "foglia", ovvero quegli elementi che si trovano agli estremi della struttura ad albero e che quindi non hanno "figli", fino a risalire all'elemento radice ovvero l'oggetto di tipo *BPSimData*.
Le figure 3.7, 3.8 e 3.9 mostrano l'implementazione della funzione *buildDataTree()* che permette la creazione della struttura dati dinamica, avvalorata con i parametri di simulazione presenti nel diagramma.

```
// * Funzione ricorsiva che popola una struttura dati ad albero in base ai valori degli elementi della simulazione
// * ritorna un array di 2 elementi, uno è il nodo, l'altro è il nodo sotto forma di oggetto
function buildDataTree(nodo, nodoObject) {

    let numFigli = nodo.childElementCount;
    let nodoFiglio;

    let childNodes = nodo.childNodes;
    let temp = [];

    for (let i = 0; i < childNodes.length; i++) {
        // * togliamo dai figli quelli che hanno campo "#text" poiche sarebbero gli invii dell'XML
        if (childNodes[i].nodeName != '#text') {
            temp.push(childNodes[i]);
        }
    }
    childNodes = temp;
    let haveMoreValue = false;
    let moreValuesTempArray = [];
    while (numFigli > 0) {

        let childToPass = childNodes.shift(); // * shift = pop ma fatta in testa
        nodoFiglio = buildDataTree(childToPass, createObj(childToPass));
        let nameAttr = nodoFiglio[0].localName.charAt(0).toLowerCase() + nodoFiglio[0].localName.slice(1);

        if (isParameter(nodoFiglio[0].localName)) {

            let parameterFieldsToDelete = [];
            for (let i = 0; i < Object.keys(nodoFiglio[1]).length; i++) {
                // salvo tutti quei parametri che si sono creati in più ovvero quelli che non iniziano per '_'
                if (Object.keys(nodoFiglio[1])[i].charAt(0) != "_" /*&& Object.keys(nodoFiglio[1])[i] != "name"*/) {
                    let temp = nodoFiglio[1][Object.keys(nodoFiglio[1])[i]];
                    parameterFieldsToDelete.push(temp);
                }
            }

            if (parameterFieldsToDelete.length > 0) {
                if (parameterFieldsToDelete.length > 1 || parameterFieldsToDelete[0].length > 1) {
                    let newParameterFieldsToDelete = []
                    for (let i in parameterFieldsToDelete) {
                        for (let j in parameterFieldsToDelete[i]) {
                            newParameterFieldsToDelete.push(parameterFieldsToDelete[i][j]);
                        }
                    }
                    parameterFieldsToDelete = newParameterFieldsToDelete;
                }
            }
```

Figura 3.7: Implementazione della funzione che crea la struttura dati - parte 1

```
        let tempResultRequest = nodoFiglio[1].resultRequest;

        let tempName;
        let tempType;

        if (nodoFiglio[0].localName == "Property") {
            tempName = nodoFiglio[1].name;
            tempType = nodoFiglio[1].type;
        }

        nodoFiglio[1] = new factory[nodoFiglio[0].localName]();
        nodoFiglio[1].resultRequest = tempResultRequest;
        nodoFiglio[1].value = parameterFieldsToDelete;
        if (nodoFiglio[0].localName == "Property") {
            nodoFiglio[1].name = tempName;
            nodoFiglio[1].type = tempType;
        }
        if (isArrayAttribute(nodoFiglio[0].localName)) {
            let tempArray = [];
            tempArray.push(nodoFiglio[1]);
            nodoObject[nameAttr] = tempArray;
        } else {
            nodoObject[nameAttr] = nodoFiglio[1];
        }
    } else {
        if (isArrayAttribute(nodoFiglio[0].localName)) {
            let tempArray = [];
            tempArray.push(nodoFiglio[1]);

            if (nodoFiglio[0].localName == "UserDistributionDataPoint") {
                let newTempArray = [];

                for (let i = 0; i < tempArray.length; i++) {
                    let singlePoint = tempArray[i];
                    let nameToEliminate = [];
                    for (let j = 0; j < Object.keys(singlePoint).length; j++) {
                        if (Object.keys(singlePoint)[j].charAt(0) != "_") {

                            if (singlePoint[Object.keys(singlePoint)[j]][0] == undefined) {
                                singlePoint["value"].push(singlePoint[Object.keys(singlePoint)[j]]);
                            } else {
                                singlePoint["value"].push(singlePoint[Object.keys(singlePoint)[j]][0]);
                            }
                            nameToEliminate.push(Object.keys(singlePoint)[j])
                        }
                    }
                    for (let j in nameToEliminate) {
                        delete singlePoint[nameToEliminate[j]];
```

Figura 3.8: Implementazione della funzione che crea la struttura dati - parte 2

```
                    }
                    newTempArray.push(singlePoint);
                }
                nodoObject["points"] = newTempArray;
            } else {
                nodoObject[nameAttr] = tempArray;
            }
        } else {
            if (nameAttr.includes("Distribution") ||
                isConstantParameter(nameAttr) ||
                nameAttr.includes("Expression") ||
                nameAttr.includes("Enum")) {

                haveMoreValue = true;
                moreValuesTempArray.push([nameAttr, nodoFiglio[1]]);
            } else {

                nodoObject[nameAttr] = nodoFiglio[1];
            }
        }
    }
    numFigli--;
}
if (haveMoreValue) {
    if (moreValuesTempArray.length != 1) {
        nodoObject[moreValuesTempArray[0][0]] = []
        nodoObject[moreValuesTempArray[0][0]].push(moreValuesTempArray[0][1])
        for (let i in moreValuesTempArray) {
            if (i != 0) {
                if (moreValuesTempArray[i][0] == moreValuesTempArray[i - 1][0]) {
                    nodoObject[moreValuesTempArray[i][0]].push(moreValuesTempArray[i][1])
                } else {
                    nodoObject[moreValuesTempArray[i][0]] = []
                    nodoObject[moreValuesTempArray[i][0]].push(moreValuesTempArray[i][1])
                }
            }
        }
    } else {
        nodoObject[moreValuesTempArray[0][0]] = moreValuesTempArray[0][1]
    }

}

let nodo_nodoObj = [];
nodo_nodoObj.push(nodo);
nodo_nodoObj.push(nodoObject);
return nodo_nodoObj;
}
```

Figura 3.9: Implementazione della funzione che crea la struttura dati - parte 3

## 3.4 Manipolazione della struttura dati

Una volta caricati, ove esistenti, i parametri di simulazione, essi vengono mostrati nella parte grafica dell'applicativo. È possibile, quindi, modificare gli stessi, crearne di nuovi o eliminare quelli appena creati o caricati.
È evidente come queste modifiche che l'utente può apportare sui parametri debbano essere riportate nella struttura dati e ciò avviene in momenti differenti a seconda della modifica attuata.
Ad esempio, se l'utente decide di modificare l'ID di un qualunque elemento di simulazione, questa modifica viene subito salvata nella struttura dati, poiché è necessario fare un controllo sul nuovo ID inserito: non è possibile, infatti, permettere l'esistenza di due elementi che presentano un ID uguale.
Le altre modifiche, invece, non sono salvate dopo ogni modifica dell'utente, infatti, con l'obiettivo di non rallentare l'esecuzione della PWA, ma è possibile salvarle in due momenti:

1. al cambio di scenario;
2. al click del tasto che permettere di generare il diagramma di processo finale con i parametri di simulazione.

La figura 3.10 è un esempio di ciò che è stato appena descritto: la funzione *saveScenarioParameterAttribute()* permette di salvare tutte le modifiche che l'utente ha fatto nella sezione relativa agli *ScenarioParameters* e verrà chiamata solo nei 2 momenti sopracitati.

```
function saveScenarioParameterAttribute(field) {
    let value = field.value;
    let fieldName = field.id.split("-")[1];
    if (field.type == "checkbox") {
        //salvo il cambimento della checkbox
        if (dataTreeObjGlobal.scenario[currentScenarioGlobal - 1].scenarioParameters[fieldName] == "true") {
            dataTreeObjGlobal.scenario[currentScenarioGlobal - 1].scenarioParameters[fieldName] = "false";
        } else {
            dataTreeObjGlobal.scenario[currentScenarioGlobal - 1].scenarioParameters[fieldName] = "true";
        }
    } else {
        let notNumber = false;
        if (fieldName == "replication" || fieldName == "seed") {
            if (!value.match(/^\d+$/)) {
                setTimeout(function () {
                    vex.dialog.alert("ERROR: You must insert an int value in replications");
                }, 10);
                if (fieldName == "replication") {
                    $('#scenarioParametersAttribute-replication-input')
                        .val(dataTreeObjGlobal.scenario[currentScenarioGlobal - 1].scenarioParameters.replication);
                }
                if (fieldName == "seed") {
                    $('#scenarioParametersAttribute-seed-input')
                        .val(dataTreeObjGlobal.scenario[currentScenarioGlobal - 1].scenarioParameters.seed);
                }
                notNumber = true;
            } else {
                value = parseInt(value, 10);
            }
        }
        if (value == "") {
            value = undefined;
        }
        if (!notNumber) {
            dataTreeObjGlobal.scenario[currentScenarioGlobal - 1].scenarioParameters[fieldName] = value;
        }
    }
}
```

Figura 3.10: Implementazione della funzione che salva le modifiche della sezione ScenarioParameters nella struttura dati

## 3.5 Serializzazione XML degli elementi della struttura dati

La serializzazione degli elementi della struttura dati avviene nel momento in cui l'utente ritiene di aver ultimato l'interazione con la Desktop PWA e desidera scaricare un file contenente il diagramma BPMN con i relativi parametri di simulazione.
Nel momento in cui l'utente preme il tasto che permette il download del suddetto file, viene chiamata la funzione che scrive il file, il cui sviluppo è presente nella figura 3.11.
In questa funzione sono salvate le modifiche fatte dall'utente, ove presenti, sullo scenario che è visualizzato (le modifiche su altri scenari sono state salvate al cambio dello scenario, come descritto in precedenza) ed in seguito viene chiamata la funzione *toXMLelement()* sull'oggetto di tipo *BPSimData*.
Questa funzione, implementata in ognuna delle classi che rappresenta un parametro di simulazione, è in grado di tradurre qualunque elemento di simulazione nel corrispettivo codice XML: come detto, viene chiamata sull'oggetto *BPSimData* e, per ogni suo attributo complesso, viene richiamata fino a raggiungere quegli oggetti "foglia" che non presentano tipi complessi. Un esempio della funzione *toXMLelement()* è presente nella figura 3.12, dove è mostrata nello specifico l'implementazione della stessa per la classe *Scenario*.
In questo esempio, gli attributi semplici vengono tradotti in XML con la funzione *eventuallyAddAttribute()*, mentre quelli complessi verranno risolti con la chiamata delle rispettive funzioni *toXMLElement()*.
Una volta ottenuto il codice XML relativo ai parametri di simulazione, viene utilizzata una funzione della libreria *vkBeautify*, che ristruttura il codice XML (in sintesi va a capo ad ogni chiusura di tag).

```
$('#generate-bpsim').on("click", function () {

    saveCurrentScenarioComplexElement(dataTreeObjGlobal.scenario[currentScenarioGlobal - 1]);

    let scenarioSelected = $('#scenario-picker').val();

    //salvo i calendari
    saveLocalCalendars();
    // salvataggio nuovi calendar per far si che i calendar nuovi diventino vecchi
    populateCalendarForm(dataTreeObjGlobal.scenario[currentScenarioGlobal - 1].calendar);

    while (extensionElementXML[0].firstChild) {
        extensionElementXML[0].removeChild(extensionElementXML[0].firstChild);
    }
    extensionElementXML[0].appendChild(dataTreeObjGlobal.toXMLelement(bpsimPrefixGlobal, bpsimNamespaceURI));

    download("bpmn-simulation.bpmn", vkbeautify.xml(new XMLSerializer().serializeToString(xmlDoc)));
});
```

Figura 3.11: Implementazione della funzione che salva i parametri di simulazione in un file

```
toXMLelement(bpsimPrefix: string, bpsimNamespaceUri: string): any{

    let parser = new DOMParser();
    let xmlDoc = parser.parseFromString(undefined, "text/xml");

    let scenarioXMLelement = xmlDoc.createElementNS(bpsimNamespaceUri, bpsimPrefix+":Scenario");

    this.eventuallyAddAttribute(scenarioXMLelement, "id", this._id);
    this.eventuallyAddAttribute(scenarioXMLelement, "name", this._name);
    this.eventuallyAddAttribute(scenarioXMLelement, "description", this._description);
    this.eventuallyAddAttribute(scenarioXMLelement, "created", this._created);
    this.eventuallyAddAttribute(scenarioXMLelement, "modified", this._modified);
    this.eventuallyAddAttribute(scenarioXMLelement, "author", this._author);
    this.eventuallyAddAttribute(scenarioXMLelement, "vendor", this._vendor);
    this.eventuallyAddAttribute(scenarioXMLelement, "version", this._version);
    this.eventuallyAddAttribute(scenarioXMLelement, "result", this._result);
    this.eventuallyAddAttribute(scenarioXMLelement, "inherits", this._inherits);

    if(this._scenarioParameters != undefined){
        scenarioXMLelement.appendChild(this._scenarioParameters.toXMLelement(bpsimPrefix, bpsimNamespaceUri));
    }

    for(let i=0; i< this._elementParameters.length; i++) {
        scenarioXMLelement.appendChild(this._elementParameters[i].toXMLelement(bpsimPrefix, bpsimNamespaceUri));
    }

    for(let i=0; i< this._vendorExtensions.length; i++) {
        scenarioXMLelement.appendChild(this._vendorExtensions[i].toXMLelement(bpsimPrefix, bpsimNamespaceUri));
    }

    for(let i=0; i< this._calendar.length; i++) {
        scenarioXMLelement.appendChild(this._calendar[i].toXMLelement(bpsimPrefix, bpsimNamespaceUri));
    }

    return scenarioXMLelement;
}
```

Figura 3.12: Implementazione della funzione che traduce i parametri di simulazione in codice XML

## 3.6 Testing

Sono stati realizzati una serie di test utilizzando il framework Mocha e l'assertion library Chai.
È di fondamentale importanza, nel momento in cui si sta realizzando un qualunque progetto software, creare una serie di casi di test per verificare la robustezza del proprio prodotto, nello specifico si deve cercare di coprire tutti i casi possibili che potrebbero avvenire durante l'utilizzo di un qualunque utente.
Nello specifico, per testare la robustezza della struttura dati dinamica, sono stati creati una serie di test che coprono circa il 75% degli statements e il 72% delle linee di codice scritte.
La figura 3.13 rappresenta quanto appena descritto: sono stati, infatti, realizzati dei test per coprire la quasi totalità dei possibili "stati" della struttura dati.
È stato, inoltre, creato un test che verifica la corretta produzione del codice XML da parte della funzione che scrive lo stesso nel file di output.
È possibile lanciare l'esecuzione automatica dei casi di test realizzati con il seguente comando:

```
npm run test
```

.

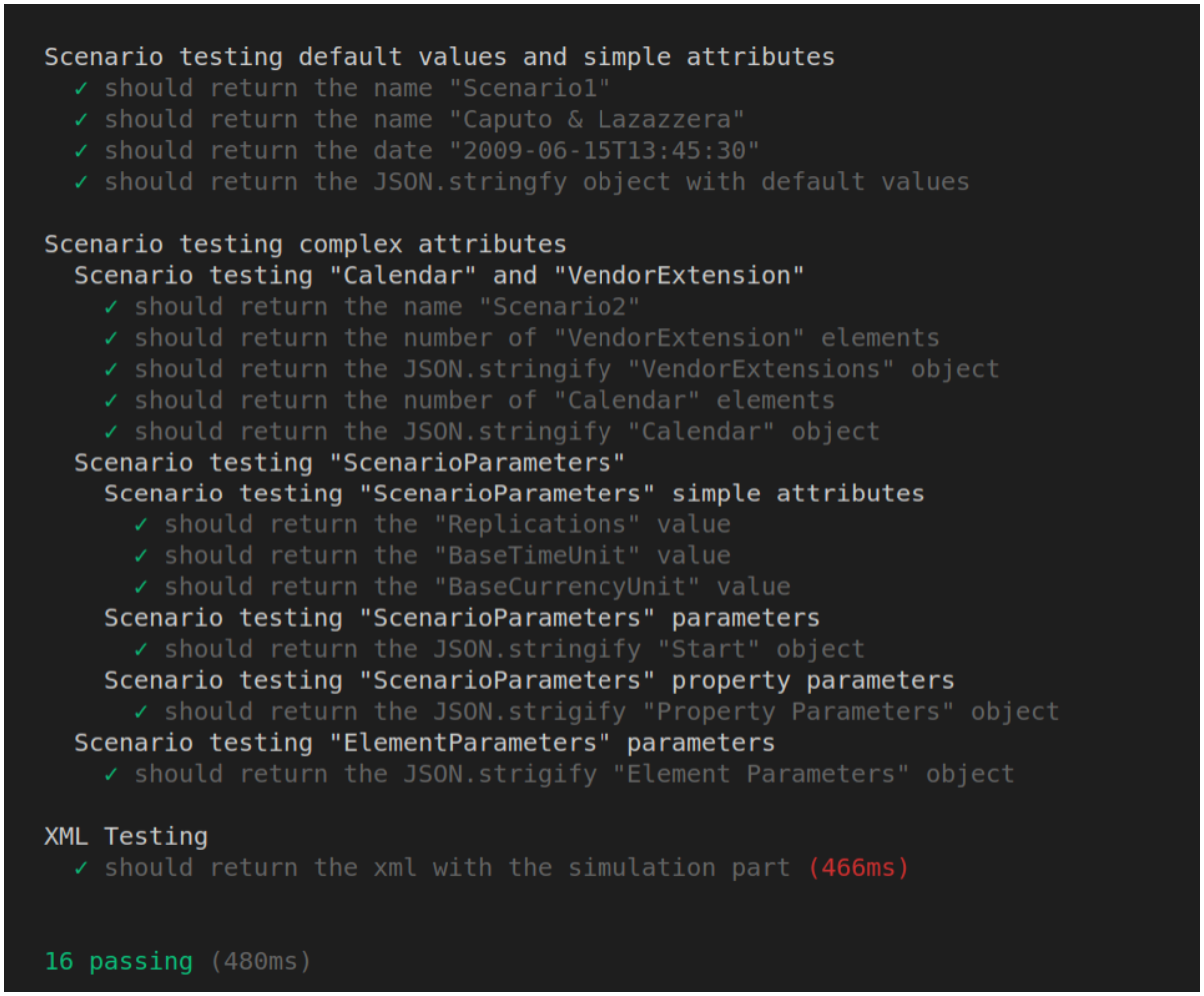



Figura 3.13: Resoconto del test effettuato con Mocha sulla struttura dati

# Conclusioni

In conclusione, in questo elaborato si è, in un primo momento, definito il concetto di simulazione, illustrandone vari esempi e sono emersi una serie di vantaggi che la simulazione porta in una qualsivoglia realtà aziendale, in particolare l'abbattimento di costi e tempi.
In un secondo momento è stato 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 manipolare i parametri di simulazione: nello specifico ci si è focalizzati sullo sviluppo di una struttura dati dinamica che salva opportunamente le manipolazioni che l'utente fa sulla parte grafica della Desktop PWA.
Per la realizzazione di questo elaborato di tesi, ovvero la creazione di una modellazione dinamica per diagrammi BPMN, sono stati utilizzati TypeScript, JavaScript e JQuery per quel che concerne la realizzazione vera e proprio della modellazione dinamica e Mocha e Chai per la fase di testing che ha portato a validare la struttura dati creata a supporto della Desktop PWA.
La ragione principale che ha portato alla creazione di una struttura dati dinamica è il poter salvare, in maniera consistente, le modifiche svolte dall'utente, evitando che tale compito sia affidato alla sola parte grafica e applicando, quindi, la separazione degli ambiti.
Questo progetto di tesi si integra perfettamente con un altro elaborato che ha portato alla produzione della parte grafica della Desktop PWA.
Il codice di questo progetto è consultabile su Github (https://github.com/mat23795/bpmn-simulation).
Fra gli sviluppi futuri più plausibili, vi è la creazione di un simulatore che sia in grado di interfacciarsi con la Desktop PWA e che completi il percorso di simulazione di diagrammi BPMN. Un ulteriore sviluppo futuro possibile potrebbe essere la possibilità di creare un

diagramma BPMN direttamente sulla PWA e di inserire contestualmente i parametri di simulazione. Interessante sarebbe anche aggiungere dei suggerimenti all'utente circa i parametri di simulazione che lui può selezionare per un determinato elemento del diagramma BPMN.

# Bibliografia


[1] Weske Mathias, *Business Process Management: Concepts, Languages, Architectures*, Springer Science Business Media, pp. 1–24, 2012.

[2] Tumay Kerim, *Business Process Simulation*, Winter Simulation Conference, 1995.

[3] Kumar Akhil, *Business process management*, New York: Routledge, 2018.

[4] Hammer Michael, Champy James, *Reengineering the corporation: A manifesto for business revolution*, New York, HarperBusiness, Capitolo: The experience of process redesign, 1993.

[5] Michele Chinosi, Alberto Trombetta, *BPMN: An introduction to the standard*, Computer Standards Interfaces, pp. 124-134, 2012.

[6] White Stephen A., *Introduction to BPMN*, `https://www.omg.org/bpmn/Documents/Introduction_to_BPMN.pdf`.

[7] *Camunda Modeler Documentation*, `https://docs.camunda.org/manual/7.4/modeler/camunda-modeler/`

[8] *BPMN.io Walkthrough*, `https://bpmn.io/toolkit/bpmn-js/walkthrough/`

[9] *Signavio Modeler*, `https://documentation.signavio.com/suite/en-us/Content/process-manager/userguide/getting-started.htm`

[10] *Bizagi Modeler*, `http://help.bizagi.com/process-modeler/en/`

[11] *Visual Paradigm Modeler*, `https://www.visual-paradigm.com/support/documents/vpuserguide/12_gettingstart.html`

[12] Gagné Denis, *Business Process Simulation Specification*, `http://bpsim.org/specifications/2.0/WFMC-BPSWG-2016-01.pdf`, 2016.

[13] *BIMP Simulator*, `http://bimp.cs.ut.ee/`

[14] *Sparx System Enterprise Architect Simulator*, https://sparxsystems.com.au/enterprise_architect_user_guide/14.0/model_simulation/model_simulation.html

[15] *Simul8 Simulator*, https://www.simul8.com/products/

[16] *Visual Paradigm Simulator*, https://www.visual-paradigm.com/tutorials/process-simulation.jsp

[17] António Paulo Freitas, José Luís Pereira, *Process Simulation Support in BPM Tools: The Case of BPMN*, https://core.ac.uk/download/pdf/55638662.pdf, 2015.

[18] *BP Simulator*, https://www.bpsimulator.com/#productivity

[19] *Signavio Simulator*, https://documentation.signavio.com/suite/en-us/Content/process-manager/userguide/bpmn-simulation.htm

[20] *Adonis Simulator*, https://fr.boc-group.com/uploads/files/_2015__ADONIS_Whitepaper_-_EN_04.pdf

[21] *Bizagi Simulator*, http://help.bizagi.com/process-modeler/en/index.html?simulation_in_bizagi.htm

[22] *Trisotech Simulator*, https://cloud.trisotech.com/help/index.html?da_documentation.htm

[23] *Typescript Documentation*, https://www.typescriptlang.org/docs/home.html.

[24] *Npm Documentation*, https://docs.npmjs.com.

[25] *Mocha Documentation*, https://mochajs.org/api/mocha.js.html.

[26] *Chai Documentation*, https://devdocs.io/chai/.

# Ringraziamenti

Vorrei ringraziare il Prof. Davide Rossi, relatore di questa tesi e fonte inesauribile di conoscenza che mi ha guidato nella stesura di questo elaborato giorno dopo giorno.
Un gigantesco grazie va alla mia fidanzata Arjola. Grazie per avermi sostenuto in questo percorso, per essermi stata vicino sempre e grazie soprattutto per tutto l'amore e l'affetto che mi hai mostrato in questi anni.
Un ringraziamento speciale alla mia famiglia, in particolare ai miei genitori, mio fratello e a mio nonno Luigi: è grazie a loro sostegno e al loro incoraggiamento se oggi sono riuscito a raggiungere questo traguardo.
Un grazie anche ai colleghi Matteo, Domenico, Antonio, Andrea ed Emilio con i quali ho passato due anni meravigliosi.
Infine un grazie agli amici del gruppo fantacalcio di Bari che riescono a rendere le giornate più allegre e spensierate.