BPMN Modeling and Simulation

# Business Process Model and Notation (BPMN 2.0)

Terminologia:

* **Start event**: genera un token
* **End event**: consuma un token
* **Sequence flow**: freccia
* **Gateway**
* **Activity**
* **Intermediate event**: ~attesa di un evento o esecuzione di un evento

**Race pattern**: “corsa” tra intermediate events alternativi dopo un event gateway

A diagram of a mail delivery process

Description automatically generated

- **Interruption pattern**: riutilizzo del sub-process e raggruppamento all’interno di un nuovo processo

(**expanded notation**)

* Aggiunta di un intermediate event “**catch**” (3) sul bordo del sub-process
  + Sarà attivato quando il sub-process non termina correttamente
* “**+**” (4) evidenzia un modello collassato di cui non sono visibili i dettagli

A diagram of a report

Description automatically generated

* Non è possibile connettere con l’esterno gli elementi interni ad un processo espanso
* **BPMN structured modeling**: è possibile rappresentare i processi utilizzando un differente livello di astrazione (da 0 a 3, livello 0 = meno dettagli). [Immagine successiva livello 0]

A diagram of a company

Description automatically generated

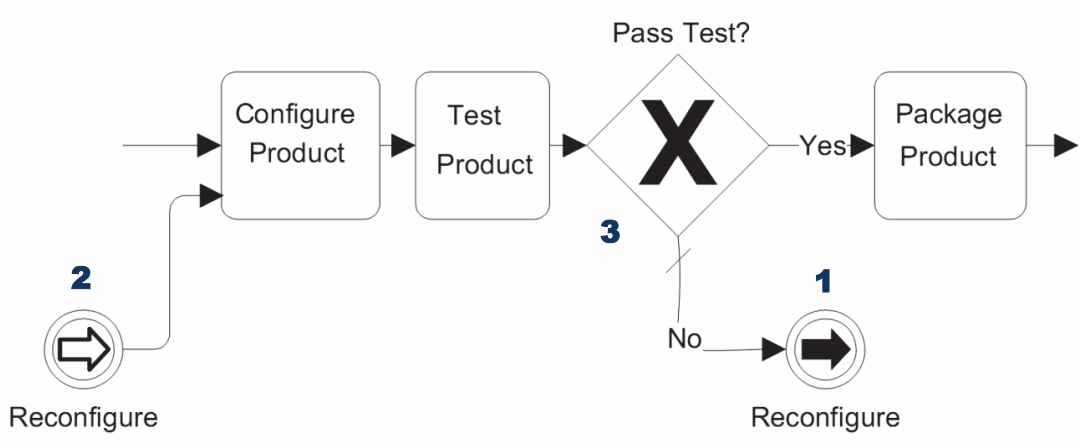
* Livello 0 🡪 rappresentazione ad alto livello 🡪 processo non eseguibile
* **BPMN segmentation** x creazione di differenti segmenti modulari

Come rendere un BPMN eseguibile? Cosa è necessario aggiungere?

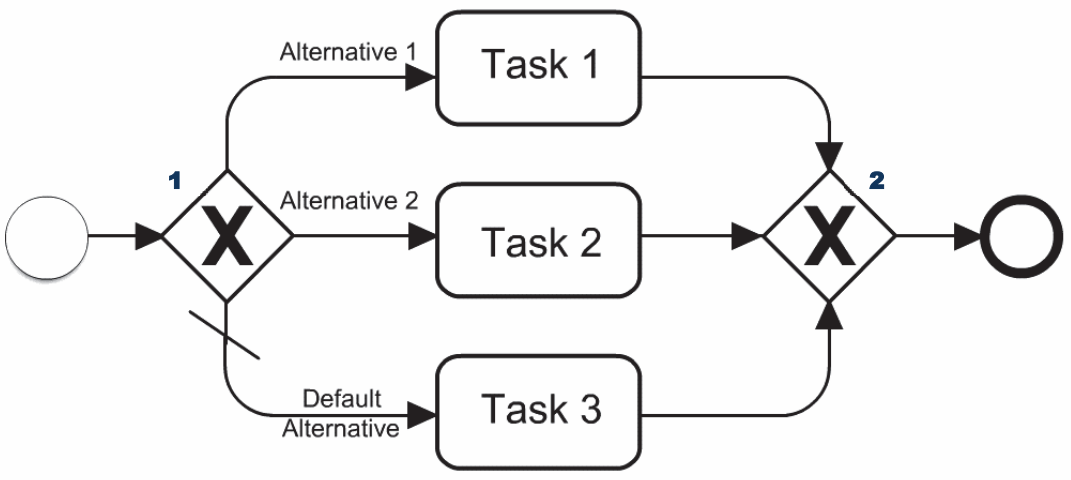
Aggiungere dettagli, arrivare ad activities atomiche per incrementare il livello.

Livello 3 🡪 rappresentazione a basso livello 🡪 processo eseguibile

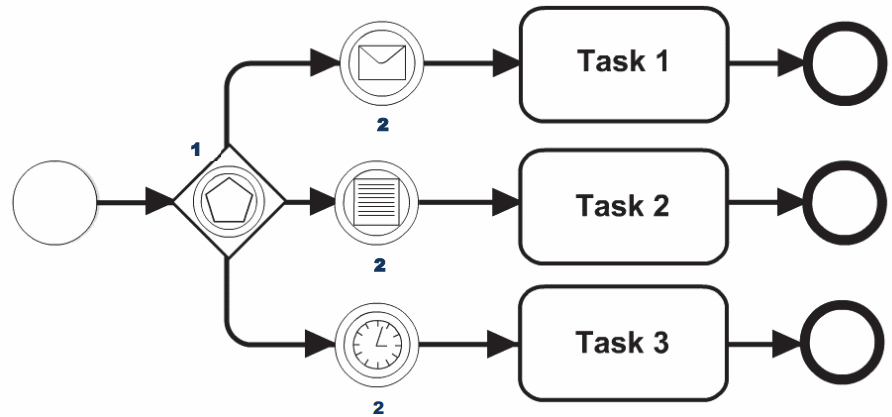
* **Off-page connector**: intermediate event con semantica throw-catch (1,2)



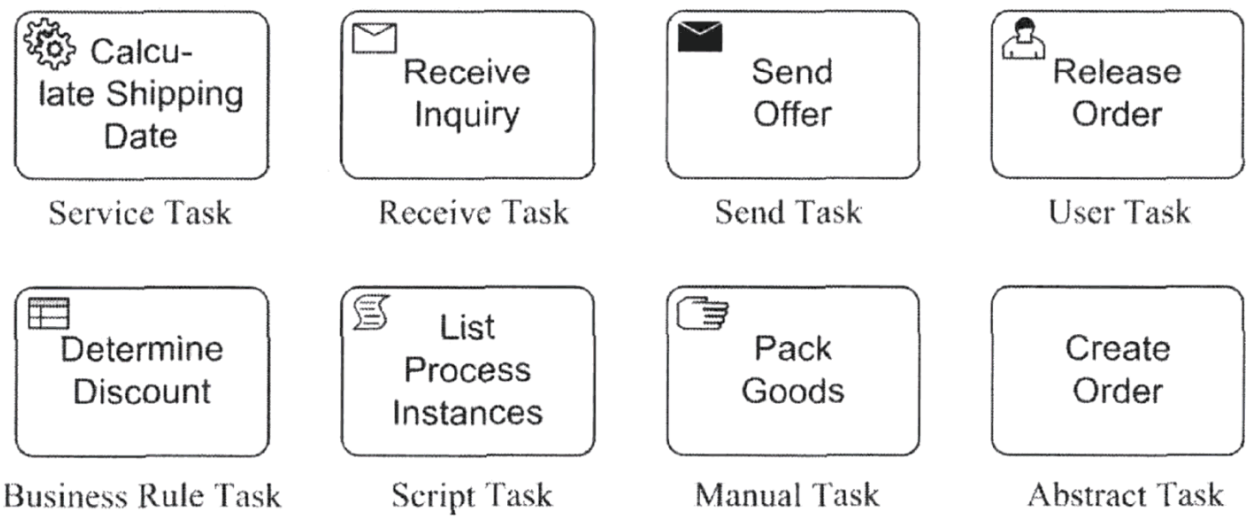
* **Exclusive gateway**: soltanto un ramo in uscita verrà percorso
  + Il ramo barrato indica l’alternativa di default



* **Exclusive event based gateway**: fa partire una “corsa” tra gli intermediate event posti si possibili rami, il primo evento soddisfatto “vince la corsa”



Tipi di task



* **Service**: funzione automatizzata processata da una applicazione esterna
* **Receive**
* **Send**
* **User**: input inseriti da umano tramite user interface
* **Business** **rule**: “trigger” che lancia automaticamente procedure per produrre risultati
* **Script**: task automatica svolta dal processo software
* **Manual task**: azione umana senza supporto IT
* **Abstract**: nessun tipo

**Semi-formal notation** = descrizione tramite elenco numerato (stile tabelle dei costi) della procedura rappresentata nello scenario

* **Inclusive gateway**: più rami in uscita dal gateway possono essere percorsi al verificarsi delle condizioni necessarie (C1, C2). Tali condizioni sono controllate all’inizio.

|  |  |  |
| --- | --- | --- |
| **C1** | **C2** | **Rami** |
| 0 | 0 | Default |
| 0 | 1 | 2 |
| 1 | 0 | 1 |
| 1 | 1 | 1 e 2 |

* + L’inclusive gateway posto in uscita serve a sincronizzare l’esecuzione

A diagram of a task

Description automatically generated

* **Parallel gateway**: no condizioni da soddisfare, esecuzione in parallelo di tutti i rami
  + Il gateway in uscita serve a sincronizzare l’esecuzione (attende terminazione di tutti i rami)

A diagram of a task

Description automatically generated

* **Loop activity**: attività ripetuta (sequenzialmente) ciclicamente, il numero di ripetizione può essere deciso:
  + Definito al momento della progettazione
  + Condizionato a runtime da dati processati
  + Calcolato a runtime
* **Multi-instance activity**: attività ripetuta sequenzialmente o in parallelo, secondo un numero definito in precedenza

A diagram of a sub-process

Description automatically generated

* **Complex gateway**: il suo comportamento può essere definito da uno script (tramite un’espressione), serve a combinare più gateway descritti in precedenza. Può essere usato per ogni situazione. Meglio evitarlo per una questione di leggibilità.

A diagram of a task

Description automatically generated

* **Pool**: rappresenta un partecipante nel business process. Può essere una entità (ex. un dipartimento) oppure un ruolo
* **Lane**: partizione di un pool

A pool chart with text

Description automatically generated with medium confidence

* **Private process**: processo interno ad un pool che non interagisce con pool esterni
* **Public process**: processo interno ad un pool che interagisce con pool esterni tramite scambio di messaggi

Non è possibile avere scambi di messaggi internamente ad un pool o un freccia (sequence flow) tra due task appartenenti a pool differenti

A diagram of a diagram

Description automatically generated

Ci sono due tipi possibili di processi: **orchestration** e **collaboration/choreography**.

· **Orchestration:** un processo con controllo centrale di attività (controllo del direttore d’orchestra), modellato come un private process, and contenente opzionali “scatole nere” presentanti entità esterne.

A diagram of a process flow

Description automatically generated

· Collaboration: ci sono più partecipanti ed il set ordinato di interazioni tra di loro è mostrato dai pool. Una collaborazione è caratterizzata da controllo distribuito. Ogni partecipante reagisce agli eventi generati dagli altri partecipanti. Può contenere sia orchestrazione che black box.

Esempio: pool (1), lane (2), message flow (3), control flow (4)

A diagram of a medical procedure

Description automatically generated

## A diagram of a credit score Description automatically generatedDATA OBJECTS

BPMN permette la modellazione del trasferimento di dati, attraverso data objects, messaggi e data store. I data object, esistono solo all’interno dei processi, mentre i data store sono persistenti. Un’associazione diretta di dati è disegnata come una linea punteggiata per modellare ogni attività quali oggetti dato prende in input o dà in output. Ciò può rappresentare una sorta di dipendenza tra attività.

ESEMPIO SULLE SLIDE: (Insurance company)

In sostanza c’è un modello non BPMN, il boss dà tutti i dettagli e dice di trasformarlo in un BPMN model. Nulla di mortale, ricordarsi che le percentuali tra gateway in serie si moltiplicano e simulare con lo splitting su più percorsi dei token.

BP-MODELING: NOTAZIONE SEMI FORMALE

Essa è fatta di corte frasine, numerate per permettere la specificazione del flusso di controllo. Sostituisce use case dettagliati in UML.

A diagram of a flowchart

Description automatically generated

* Numerazione sequenziale
* Nuovo livello introdotto ogni volta che c’è un flusso interno
* Nuovo livello alfabetico per esprimere branches inclusive (GATEWAY COL +)
* Nuovo livello numerico per esprimere branches esclusive (gli if, gateway col x)
* Il task type è riportato tra parentesi in fondo
* L’end del processo è obbligatorio ed è denotato con ->End (End locale) o con un salto all’end globale

Gli scenari son rappresentati da frecce nere solide (chad frecce). La freccia dotted (puntata) indica un flusso parallelo.

A diagram of a diagram

Description automatically generated

ESEMPIO: Libreria

A diagram of a computer

Description automatically generated

Ricorda: quando rimane un solo token, esso non può dividersi in più parti. Ad esempio, qua alla terza iterazione qua c’è un solo token e va “verso il basso” perché 60% > 40%

**Modellare dal linguaggio naturale**

Viene dato un testo descrittivo di una factory e si deve creare un diagramma di livello handoff.

ESEMPIO DI CIMINO: Manage orders in a manufacturing company

Testo:

Order processing in a manufacturing company takes place after an order has been received from a customer. The order is first taken by the sales office. Both manufacturing capacity and parts availability are then checked, via appropriated business rules and an automated information service, by the production office and the warehouse, respectively. Subsequently, an order rejection is sent to the customer by the sales office if capacity is not available. Otherwise, if parts are also available, an order confirmation is automatically sent to the customer by the sales office. Therefore, both assemblies and invoice are produced, by human operatives at the workshop and by the sales office via an automated information service, respectively. Finally, assemblies and invoice are sent to the customer by the shipping office. In contrast, if there are parts to be ordered, the purchase office procures such parts, purchasing them from suppliers, so as to allow the aforementioned process to continue once such parts are available. However, if some parts are still unavailable (they cannot be procured), a rejection is then sent to the customer by the sales office.

Modellazione:

A diagram of a manufacturing company

Description automatically generated

Definizione Handoff level diagram: particolare tipo di flusso in cui il lavoro viene passato da un attore all’altro

A diagram of a decision

Description automatically generatedImportante: Il caso B è migliore del caso A a livello di tempi, si evita la sincronizzazione tra token nel caso in cui CAPACITY NOT AVAILABLE.

A è più semplice di B a livello di layout

VISUAL PARADIGM FOR UML 11

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Le istruzioni per configurare Visual Paradigm sono nelle slide del Professore.

Immagine che contiene testo, schermata, diagramma, Carattere

Descrizione generata automaticamenteÈ la versione semplifica della Insurance Company, sopra i task sono riportati durata e costo di esecuzione, il gateway di XOR ha probabilità 50/50, i 10 token arrivano tutti insieme ma Claims Handler ha solo 5 istanze disponibili.

Slot Temporale 0-10 minuti

Dei 10 token che arrivano solo 5 vengono processati, gli altri rimangono in attesa, dei 5 processati 3 vanno a finire nel ramo NO e 2 nel ramo YES

Immagine che contiene testo, schermata, linea, diagramma

Descrizione generata automaticamenteSlot Temporale 10-20 minuti

I 5 token che erano in attesa adesso sono in RECEIVE CLAIM.

I 3 token che hanno percorso il ramo NO hanno raggiunto l’end event.

I 2 token che hanno percorso il ramo YES sono invece in attesa perché nessuna istanza è libera, sono tutte occupate con RECEIVE CLAIM.

Immagine che contiene testo, schermata, linea, diagramma

Descrizione generata automaticamente

Slot Temporale 20-25 minuti

Dei 5 token in RECEIVE CLAIM, 2 sono andati nel ramo NO e hanno quindi terminati, gli altri 3 nel ramo YES.

Adesso sono disponibili 5 istanze per processare i 5 token rimanenti in ENTER DATA INTO THE SYSTEM.

Durata Finale: 25 minuti Costo Finale: 125€

Adesso è possibile visualizzare il visual report.

Immagine che contiene testo, schermata, schermo, software

Descrizione generata automaticamente

Completed Input against Time

Come si vede dal grafico, i token che passano dal ramo POLICY NOT VALID (rosa in figura) vengono completati in due tranche, 3 dopo 10 minuti e 2 dopo ulteriori 10 minuti (5 in 20 minuti), mentre invece quelli che passano dal ramo POLICY VALID vengono completati tutti insieme dopo 25 minuti.

Immagine che contiene testo, schermata

Descrizione generata automaticamente

Resource Usage

Rappresenta l’utilizzo delle istanze nel tempo, si ha che l’utilizzo è sempre del 100%, tutte e 5 le istanze sono usate, fino al minuto 25 dove scende a picco allo 0% ovvero quando ogni token è stato processato.

Immagine che contiene testo, software, Rettangolo, schermata

Descrizione generata automaticamente

Queue Time

Ci indica il tempo di attesa medio **nel caso in cui ci sia da attendere,** nel calcolo non vengono conteggiati i token che vengono direttamente processati senza attendere, entrambi i task hanno infatti entrambi 10 minuti di attesa media.

Immagine che contiene testo, software, Icona del computer, Pagina Web

Descrizione generata automaticamenteCost per Flow Object

RECEIVE CLAIMS: 10 token \* 10€ = 100€

ENTER DATA: 5 token \*5€ = 25€

Immagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamente

Cost per Input

Il Costo per Istanza è la somma dei costi delle attività coinvolte in quello scenario considerando un solo token.

POLICY VALID: RECEIVE CLAIMS (10€) + ENTER DATA (5€) = 15€

POLICY NOT VALID: RECEIVE CLAIMS = 10€

Immagine che contiene schermata, testo, Diagramma

Descrizione generata automaticamenteCost over Time

É basato sul numero di token processati durante un Time Slot.

POLICY VALID:

* 0 minuti: 2 token \* 10€ = 20€
* 0-10 minuti: 20€ + (3 token \* 10€) = 50€
* 10-20 minuti: 50€ + (5 token \* 5€) = 75€
* 20-25 minuti: Non ci sono altri incrementi

POLICY NOT VALID:

* 0 minuti: 3 token \* 10€ = 30€
* 0-10 minuti: 30€ + (2 token \* 10€) = 50€
* 10-25: Non ci sono altri incrementi

BP Simulation

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Process Logs: Un log dettagliato per ogni istanza di processo che può essere analizzato utilizzando un framework per il process mining, sono un set finito di transazioni che coinvolgono process item come Customers e Products, ex. Cust1 compra Prod1 e Prod2; Cust2 compra Prod2; Cust3 compra Prod1, Prod2 e Prod3. I file sono salvati generalmente nel formato MXML (Mining XML).

Benchmarking: Tecnica utilizzata dalle aziende per comparare le proprie performance con altre aziende dello stesso settore attraverso l’uso di diversi diagrammi.

KPI: Key Performance Indicator

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

# ADVERTISING AGENCY

Immagine che contiene testo, diagramma, Piano, linea

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Abbiamo 100 token alla partenza, i gateway hanno probabilità:

* TEXT OKAY (90%) / NOT OKAY (10%)
* LAYOUT OKAY (80%) / NOT OKAY (20%)
* PROOF ACCEPTED (99%) / CHANGES REQUESTED (1%)

I 100 token vengono duplicati dal parallel gayeway (+) su entrambi i rami, sul ramo TEXT ne passano 90 mentre 10 vanno in CORRECT TEXT, nel ramo LAYOUT ne passano 80 mentre 20 vanno in REMARK LAYOUT, nel caso migliore (i 10 che vanno in CORRECT TEXT fanno parte dei 20 che vanno in REWORK LAYOUT) in uscita dal merge gateway ci saranno inizialmente 80 token, col passare del tempo anche i restanti 20 arriveranno, al gateway finale 99 passeranno ad ARCHIVE ADVERTISEMENT e termineranno, mentre solo 1 verrà mandato a REWORK ADVERTISEMENT, anche questo token prima o poi arriverà a ARCHIVE ADVERTISEMENT e terminerà.

Identificazione scenari per determinare il numero di ending token per ciascuno di essi.

* Arrivo di 100 token
* Scenario S1 (vengono considerati tutti e 100 i token)
  + Il parallel gateway duplica i 100 token sui due rami
    - Dal gateway del ramo sopra (text) passano 90 token che vanno in attesa sul merge gateway e 10 token entrano nel ciclo
    - Dal gateway del ramo sotto (layout) passano 80 token che vanno in attesa sul merge gateway e 20 token entrano nel ciclo
  + Il merge gateway in uscita si trova con 90 token in attesa provenienti dal ramo sopra e 80 dal ramo sotto. Passano quindi 80 token, considerando il best case secondo cui gli 80 token pronti sul ramo inferiore fanno parte dei 90 pronti su quello superiore. Restano quindi 10 token pronti del ramo superiore in attesa sul merge gateway.
  + Raggiungono l’end event **79 token**, 1 token tornerà alla revisione testo e layout.
* Scenario S2 (vengono considerati soltanto i token che per entrambi le verifiche, testo e layout, sono stati rigettati ad S1)
  + Dei 10 token che hanno fatto nuovamente la verifica del testo 9 token vanno avanti ed 1 token entra ancora una volta nel ciclo.
  + Dei 20 token che hanno fatto nuovamente la verifica del layout 16 token vanno avanti e 4 token entrano ancora una volta nel ciclo.
  + Il merge gateway in uscita si trova con 9 token in attesa provenienti dal ramo sopra e 16 dal ramo sotto. Passano quindi 9 token, considerando il best case secondo cui i 9 token pronti sul ramo superiore fanno parte dei 16 pronti su quello inferiore. Restano quindi 7 token pronti del ramo inferiore in attesa sul merge gateway.
  + Raggiungono l’end event **9 token**, nessun token tornerà alla revisione testo e layout.
* Scenario S3 (vengono considerati soltanto i token rimasti in attesa sul parallel gateway dagli scenari precedenti)
  + Il merge gateway in uscita si trova con 10 token in attesa provenienti dal ramo sopra e 7 token dal ramo sotto. Passano quindi 7 token, considerando il best case secondo cui i 7 token pronti sul ramo inferiore fanno parte dei 10 pronti su quello superiore. Restano quindi 3 token pronti del ramo superiore in attesa sul merge gateway.
  + Raggiungono l’end event **7 token**, nessun token tornerà alla revisione testo e layout.
* Scenario S4
  + Dei 4 token che hanno fatto nuovamente la verifica del layout 3 token vanno avanti e 1 token entra ancora una volta nel ciclo.
  + Il merge gateway in uscita si trova con 3 token in attesa provenienti dal ramo sopra e 3 token dal ramo sotto. Passano quindi 3 token, considerando il best case secondo cui i 3 token pronti sul ramo inferiore sono esattamente i 3 pronti su quello superiore.
  + Raggiungono l’end event **3 token**, nessun token tornerà alla revisione testo e layout.
* Scenario S5
  + Dei 1 token che hanno fatto nuovamente la verifica del testo 1 token va avanti e nessuno entra ancora una volta nel ciclo.
  + Dei 1 token che hanno fatto nuovamente la verifica del layout 1 token va avanti e nessuno entra ancora una volta nel ciclo.
  + Il merge gateway in uscita si trova con 1 token in attesa proveniente dal ramo sopra e 1 token dal ramo sotto. Passa quindi 1 token.
  + Raggiungono l’end event **1 token**.
* Scenario S6
  + Passa 1 token, l’ultimo token che raggiunge l’end event.

## BUSINESS PROCESS SIMULATION: INSURANCE COMPANY

Problema: trovare il livello di personale (staffing level) più efficiente per ognuno dei 5 tipi di risorsa.

Limitazioni:

* Ogni tipo di risorsa ha un limite max di 20 persone
* Il totale di persone non può eccedere 70
* Per 100 richieste (picco) non devo eccedere 200 minuti di tempo

A screenshot of a document

Description automatically generatedA diagram of a process flow

Description automatically generatedPer decidere quale configurazione si allinea meglio col livello service e l’obiettivo di processo va considerato un tradeoff tra il #di personale e la durata totale.

Cimino fa l’esperimento con 20 token per lane. La durata totale è di 3h 26m 5s (206 min). Eccediamo la threshold prefissata.

Si nota che la seconda lane è la più “in difficoltà” con code e i 20 addetti quasi sempre impiegati. Questa coda può anche dipendere dai tempi di arrivo dalla prima.

A screenshot of a graph

Description automatically generated

Cimino prova quindi a ridurre i token nella prima lane, portandoli a 9. Si ottiene:

A screenshot of a graph

Description automatically generated  
L’algoritmo procede così. Eliminiamo per ogni lane istanze partendo dalle 20 iniziali. Finchè i tempi NON aumentano in maniera significativa continuo a togliere, altrimenti mi fermo. Nella riduzione Cimino arriva ad avere (9,20,18,6,16). Il tempo diventa minore di 200 minuti e il numero totale di istanze 69 (<70).

La sensitività dei parametri va ad indagare come l’eliminazione di un ulteriore istanza in ogni singola lane impatta i tempi. Vediamo la tabella nel nostro esempio:

A table with numbers and time

Description automatically generated

Notiamo come le prime due lane siano le più sensibili! Se gli levi una risorsa i tempi SBIRRANO over 200 minuti.

Ordering and Delivering Pizza

Immagine che contiene testo, diagramma, Piano, Disegno tecnico

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Alla partenza ci sono 100 token(pizze), ogni token rappresenta un ordine di una pizza, il 95% delle pizze sono cotte in 60’, il 90% delle pizze sono completate (cotte e consegnate) in 60’.

Dopo 60’ avremo: 90 completate, 5 cotte ma non consegnate, 5 non cotte.

Dopo 120’ delle 10 pizze rimaste avremo: 9 completate, 1 (10 - 9) cotta ma non consegnata, 0 (5 \* 5%) non cotte.

Dopo 180’ l’unica pizza rimasta sarà cotta e consegnata.

Come si nota l’end event del pool PIZZA VENDOR è diverso dall’end event del PIZZA CUSTOMER, infatti il gateway parallelo(+) duplica i token facendone andare 100 al task “BAKE THE PIZZA” e 100 in attesa del messaggio “WHERE IS MY PIZZA?” intrappolandoli in quel loop, quello specifico end event chiamato DOT END EVENT permette di eliminare non solo il token arrivato in fondo ma anche il gemello intrappolato nel loop.

# Job Application

Immagine che contiene testo, diagramma, Carattere, linea

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, informazione

Descrizione generata automaticamente

Considerando 100 token (invio del curriculum) soltanto il 10% risulta interessante per l’ENTERPRISE.

Dopo l’arrivo della conferma di ricezione da parte dell’ENTERPRISE del curriculum l’APPLICANT si mette in attesa di messaggio.

Caso 1: APPLICANT INTERESTING -> INVITATION RECEIVED 10 token

L’APPLICANT invia un messaggio per conferma data/ora del colloquio, entrambi i pool si mettano in attesa fino alla data/ora del colloquio, dopo il colloquio entrambi i pool terminano con un normale end event.

Caso 2: APPLICANT NOT SUITABLE -> REJECTION RECEIVED 90 token

L’ENTERPRISE utilizza il MESSAGGE SENT END EVENT per terminare e conseguentemente inviare il messaggio di rifiuto all’APPLICANT che dopo aver ricevuto il messaggio di rigetto termina con un normale end event (probabilmente non esiste un end event con messaggio ricevuto altrimenti sarebbe stato usato quello).

## Modeling from informal natural language: a Hospital Emergency Center (grattata di palle)

A diagram of a flowchart

Description automatically generated

L’esercizio inizia con un testo descrittivo dell’ospedale. Sono forniti i vari scenari possibili e le relative probabilità. L’ospedale è diviso in stanze, le quali sono individuate come lane della factory (l’ospedale è il pool). Alcune percentuali non sono fornite, ma sono semplicemente ricavabili da aritmetica di base (se non ci riuscite CAZZI vostri).

Gli attori sono:

* Administrative clerk (non il prete)
* Nurse
* Physician
* Patient care technitian

I pazienti possono vedersi assegnati 3 codici (verde giallo o rosso). Chi arriva in ambulanza prende il rosso di default e viene piazzato in una emergency room. Per altri dettagli vedere il grafico, credo siano abbastanza ovvi dal modello sopra (o slide 55 del pacco 2).

L’obiettivo è trovare il numero di token per ogni scenario.

A diagram of a number of numbers and a pie chart

Description automatically generated

In foto vediamo come sono distribuiti i pazienti in base al codice e tipo di arrivo (AMBULANZA YES/NO).

Scenari:

* S1) NO AMBULANCE & NO RED CODE & NO ADMISSION: 100 x .944 & 94 x .9025 & 85 x .9 = 76.5 » 77.
* S2) NO AMBULANCE & RED CODE & NO ADMISSION: 100 x .944 & 94 x 0.0975 & 9 x .9 = 8
* S3) NO AMBULANCE & NO RED CODE & ADMISSION: 100 x .944 & 94 x .9025 & 85 x .1 = 8.5 » 8
* S4) NO AMBULANCE & RED CODE & ADMISSION: 100 x .944 & 94 x 0.0975 & 9 x .1 = 1
* S5) AMBULANCE & RED CODE & ADMISSION: 100 x .056 & 6 & 6 x .1 = 1
* S6) AMBULANCE & RED CODE & NO ADMISSION: 100 X .056 & 6 & 6 x .9 = 5