Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in

Informatica per il management

PROGETTAZIONE E PROTOTIPAZIONE DI UN'APPLICAZIONE WEB PER MODELLARE PROCESSI IN AMBITO MANUFACTURING

Relatore:
Prof. ANGELO DI IORIO

Presentata da:
ROBERTO MITUGNO

Correlatori:
Dott. MARCO FERRATI,
Prof. DAVIDE ROSSI

Sessione I

Anno Accademico 2023/2024

INDICE

CAPITOLO 1 - INTRODUZIONE

L'obiettivo principale di questa tesi è sviluppare e il testare un'interfaccia utente intuitiva ed efficace per la creazione e la modellazione di processi produttivi. Questo strumento sarà utilizzato per definire le componenti e i parametri di un processo produttivo in ambito manifatturiero fornendo un input ad un simulatore di processi. Il progetto nasce dall'esigenza di affrontare e risolvere un problema concreto incontrato in un progetto dei relatori di questa tesi, impegnati nello sviluppo di nuovi metodi per la gestione e l'ottimizzazione dei processi aziendali.

Per raggiungere questo obiettivo, verranno utilizzati due principali approcci alla modellazione dei processi produttivi: centrato sulle risorse (attraverso un framework) e centrato sulle attività (attraverso BPMN). BPMN (Business Process Model and Notation) è uno standard internazionale per la modellazione dei processi aziendali che utilizza una notazione grafica standardizzata per rappresentare i processi aziendali. D'altra parte, un framework offre una struttura organizzativa e una base solida per sviluppare applicazioni software, facilitando la modellazione delle risorse e delle attività coinvolte nei processi produttivi.

L'interfaccia utente sarà costruita utilizzando la libreria bpmn.js basata su JavaScript, con il supporto di React per la creazione di un pannello di proprietà che consenta la definizione dettagliata delle caratteristiche dei processi.

Lo sviluppo di uno strumento in grado di simulare i processi produttivi è fondamentale per affrontare le sfide che le aziende devono superare ogni giorno per rimanere competitive sul mercato. L'ottimizzazione e l’analisi sono essenziale per ridurre i costi, incrementare la qualità dei prodotti e migliorare l'efficienza operativa. Tuttavia, richiedono l'utilizzo di strumenti adeguati alla modellazione e la creazione di scenari reali.

In questo contesto, l'utilizzo di un'interfaccia utente intuitiva ed efficace per la modellazione dei processi produttivi può fare la differenza. Uno strumento ben progettato può facilitare il lavoro degli ingegneri e dei manager, permettendo loro di creare e modificare i modelli di processo in modo rapido e semplice, senza la necessità di conoscenze approfondite di programmazione o di specifici linguaggi di modellazione. Inoltre, può migliorare la collaborazione tra i diversi attori coinvolti nel processo di ottimizzazione, facilitando la condivisione delle informazioni e la comunicazione tra i team.

1.1 BPMN

BPMN [6] è uno standard internazionale per la modellazione dei processi aziendali, nato nel 2004 dalla Business Process Management Initiative (BPMI) e successivamente adottato dall'Object Management Group (OMG) nel 2005. La modellazione, in questo contesto, consiste nella realizzazione di un modello che rappresenti in maniera semplificata, ma accurata, un sistema, un processo o un concetto, catturandone le caratteristiche fondamentali. L'obiettivo principale è fornire una notazione grafica standardizzata per la rappresentazione dei processi aziendali, facilitando la comunicazione tra i diversi stakeholders coinvolti nel processo, come ad esempio i manager, gli analisti di processo, i progettisti e gli sviluppatori di software.

Nel 2011, è stata rilasciata la versione 2.0 di BPMN, che ha introdotto nuove caratteristiche e miglioramenti rispetto alla versione precedente, come ad esempio nuovi elementi grafici o la possibilità per l'utente di definire nuovi componenti e regole.

La necessità di uno standard come BPMN nasce dalla crescente complessità dei processi aziendali e di disporre di uno strumento in grado di rappresentare in modo chiaro e preciso le attività, i flussi di lavoro e le interazioni tra i diversi attori coinvolti. Inoltre, consente di rappresentare i processi in modo indipendente dalla tecnologia utilizzata, facilitando l'integrazione tra i diversi sistemi informativi aziendali e la comunicazione tra i diversi reparti aziendali.

BPMN non è l'unico strumento di modellazione dei processi aziendali disponibile sul mercato. Esistono infatti altri software simili, come ad esempio UML (Unified Modeling Language), EPC (Event-driven Process Chain) e Petri Net. Tuttavia, si distingue dagli altri strumenti per la sua semplicità, flessibilità e versatilità, utilizzando infatti una notazione grafica intuitiva, basata su simboli facilmente comprensibili, che consentono di rappresentare in modo chiaro e preciso i processi, anche quelli più complessi.

Questo strumento è applicabile in vari ambiti aziendali, che spaziano dalla gestione dei processi produttivi a quella amministrativa e di supporto. In particolare, è impiegato per modellare i flussi operativi al fine di automatizzarli, ottimizzarli e monitorarli. Grazie alla sua capacità di rappresentare in modo chiaro e preciso tali flussi, consente di identificare eventuali inefficienze e colli di bottiglia, migliorare la comunicazione tra i diversi attori coinvolti e ridurre i tempi e i costi di realizzazione delle attività.

La rappresentazione del flusso di lavoro viene creata utilizzando alcuni simboli grafici per poter riprodurre i vari elementi di un processo aziendale. Essi sono divisi in diverse categorie principali:

- Eventi: rappresentano qualcosa che accade durante il corso di un processo e possono influenzare il flusso dello stesso. Sono ulteriormente classificati in eventi di inizio, intermedi e finali. La prima tipologia segnala l'inizio di un processo, quelli intermedi rappresentano eventi che accadono durante il processo e quelli finali indicano la conclusione del processo.

Fig. 1: BPMN Eventi

- Attività: rappresentano il lavoro svolto all'interno del processo. Sono compiti singoli (task) o più complessi (sub-process) e possono essere ulteriormente suddivise in base al tipo di lavoro che rappresentano, come attività di servizio, attività umane, attività di script, ecc.

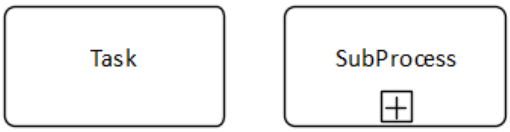


Fig. 2: BPMN Attività

- Gateway: vengono utilizzati per controllare il flusso del processo, determinando il percorso che deve seguire. Possono rappresentare decisioni, parallelismo, unione di flussi e altre logiche di controllo.

Fig. 3: BPMN Gateway

- Connettori**:** le componenti del flusso sono collegati tra loro utilizzando oggetti di connessione, che sono di tre tipologie:
    - Sequence Flow: rappresenta il flusso di controllo tra le attività all'interno di un processo. Viene utilizzato per indicare l'ordine in cui le attività devono essere eseguite.
    - Message Flow: rappresenta lo scambio di messaggi tra due entità diverse, come ad esempio tra due processi o tra un processo e un partecipante esterno. I messaggi possono essere inviati o ricevuti e possono essere utilizzati per sincronizzare l'esecuzione di due processi.
    - Association: rappresenta una relazione tra due elementi del processo, come ad esempio tra un'attività e un oggetto dati. Viene utilizzato per indicare che l'oggetto dati è associato all'attività e può essere utilizzato o prodotto durante l'esecuzione dell'attività.

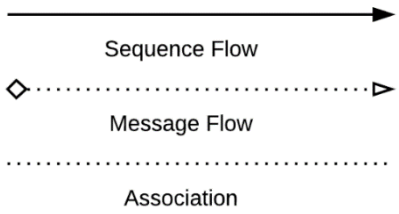


Fig. 4: BPMN Connettori

- Artefatti: sono utilizzati per fornire ulteriori informazioni sul processo. Esistono sono tre tipi di artefatti in BPMN:
    - Data Object: rappresenta un oggetto dati che viene utilizzato o prodotto durante l'esecuzione di un'attività. Può indicare i dati di input o di output di un'attività.

- Group: raggruppa diverse attività all'interno di un processo. Non influisce sul flusso del processo, ma viene utilizzato per migliorare la leggibilità del diagramma.
- Annotation: viene utilizzata per fornire ulteriori informazioni sul processo. Può essere utilizzata per aggiungere note o commenti al diagramma.

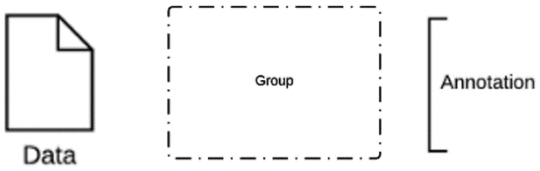


Fig. 5: BPMN Gateway

1.2 Simulazione dei processi produttivi

È importante comprendere come la rappresentazione grafica data da BPMN possa essere utilizzata per migliorare l'efficienza ed efficacia dei processi stessi. Essa fornisce una base per la creazione e l'utilizzo di software per la simulazione che consentono migliorare i processi produttivi.

La simulazione è una tecnica che permette di creare modelli virtuali di sistemi reali [14]. Consente di studiare il comportamento del sistema e prevedere le sue prestazioni in diverse condizioni operative. Nel contesto dei processi, viene utilizzato per replicare il funzionamento di una fabbrica o di una linea di produzione, analizzando l'impatto di diverse modifiche senza dover intervenire sul sistema reale.

Questo metodo è particolarmente utile per:

- Eseguire analisi "what-if": valuta l'impatto di diverse ipotesi e scenari sul sistema. Ad esempio, si può simulare l'effetto di un aumento della domanda di un prodotto, di un cambiamento nella disponibilità delle risorse o di una modifica del layout della linea di produzione.
- Ottimizzare le risorse: identifica la configurazione ottimale del sistema in termini di utilizzo delle risorse, come macchinari, manodopera e materiali.
- Ridurre i tempi di produzione: individua i colli di bottiglia e le inefficienze nel processo produttivo, permettendo di ottimizzare i tempi di ciclo e di ridurre i tempi di produzione complessivi.
- Migliorare la qualità del prodotto: analizza l'impatto di diverse variabili sulla qualità del prodotto, identificando le cause di difetti ed implementa misure per migliorare la qualità complessiva.

1.3 Approcci alla modellazione

Nell'ambito della simulazione, due metodologie prevalenti guidano la progettazione e l'implementazione di modelli: l'approccio centrato sulle risorse e l'approccio centrato sulle

attività. Entrambe mostrano punti di forza e di debolezza distinti, rendendoli adatti a differenti scenari.

L'approccio "resource centric" si focalizza principalmente sugli elementi impiegati nel processo produttivo. Questi possono comprendere macchinari, forza lavoro, materie prime e qualsiasi altro componente indispensabile per svolgere le attività produttive. La priorità viene data alla gestione e all'ottimizzazione di tali elementi, al fine di aumentare l'efficienza operativa e minimizzare i periodi di inattività. Ad esempio, in una fabbrica di produzione di automobili, l'approccio analizzerebbe l'utilizzo delle macchine di assemblaggio, la disponibilità dei pezzi di ricambio e la gestione del personale tecnico. Si potrebbe scoprire che un macchinario critico è frequentemente fermo per manutenzione non programmata, causando ritardi nella produzione. In questo modo, si può pianificare meglio la manutenzione predittiva, riducendo i tempi di fermo e migliorando l'efficienza complessiva.

L'approccio "activity centric", invece, si concentra sulle attività o sui processi specifici che compongono il flusso produttivo. Esse rappresentano le singole operazioni o fasi del processo produttivo, come l'assemblaggio di componenti, il controllo di qualità o il confezionamento. Utilizzando lo stesso esempio della fabbrica di automobili, l'approccio si focalizzerebbe sul dettagliare e migliorare ogni fase del processo di assemblaggio dell'auto. Per esempio, si potrebbe scoprire che l'installazione dei sedili e il montaggio delle porte sono svolti in due stazioni separate, creando un rallentamento nelle fasi successive. Questa tipologia di modellazione potrebbe suggerire una riorganizzazione delle stazioni di lavoro per svolgere le due operazioni in parallelo, riducendo i tempi di ciclo e migliorando la produttività complessiva.

1.4 Framework

Un framework è un insieme di strumenti (librerie, moduli, componenti) che forniscono una struttura organizzativa utilizzata per sviluppare applicazioni software, basato su un approccio resourse centric. Immaginando di costruire una casa, un framework sarebbe come l'impalcatura che sostiene le pareti mentre viene costruita e fornisce i piani su cui lavorare. Offre una base solida e standardizzata su cui sviluppare soluzioni software, aiutando a non dover reinventare tutto da zero ogni volta.

In questo progetto, il framework si suddivide in due "mondi" interconnessi: il mondo dei processi e il mondo delle risorse.

Nel mondo dei processi, viene utilizzato un metamodello (definito nel paragrafo 1.5) generico basato sulle attività. Il metamodello definisce le regole e i componenti generali necessari per creare modelli specifici, senza imporre una struttura rigida. I due concetti fondamentali richiesti sono quelli di Processo e Attività:

- Processo: rappresenta l'unità di base del modello, definendo una sequenza di attività coordinate per raggiungere un obiettivo specifico. Per esempio, in una fabbrica, un processo potrebbe essere l'assemblaggio di un'automobile.
- Attività: è l'unità di lavoro all'interno del processo. Ogni attività descrive un compito specifico che deve essere completato. In un processo di assemblaggio di un'automobile, un'attività potrebbe essere l'installazione del motore. Esse collegano il mondo dei

processi con il mondo delle risorse, definendo quali azioni devono essere compiute (ad esempio, l'installazione del motore) e quali risorse sono necessarie per realizzarle (ad esempio, un operatore specializzato e un motore).

Nel mondo delle risorse, identifichiamo due tipi principali di risorse che partecipano al processo di produzione: esecutori e oggetti.

- Esecutori: Sono gli agenti che svolgono le attività. Possono essere esseri umani, come operatori o tecnici di manutenzione, oppure macchine, come robot o impianti automatizzati. Sono elementi attive e svolgono azioni specifiche nel processo produttivo.
- Oggetti: Sono componenti passivi che supportano il processo. Questa categoria include una vasta gamma di elementi, come materie prime, componenti, prodotti finiti e strumenti. Gli oggetti non eseguono attività ma sono essenziali per completarle.

Il framework funziona collegando il mondo dei processi e il mondo delle risorse. Ogni attività definita nel processo richiede determinate risorse per essere completata.

Questa organizzazione permette di modellare in modo chiaro e dettagliato come i vari elementi di un sistema produttivo lavorano insieme per raggiungere gli obiettivi prefissati.

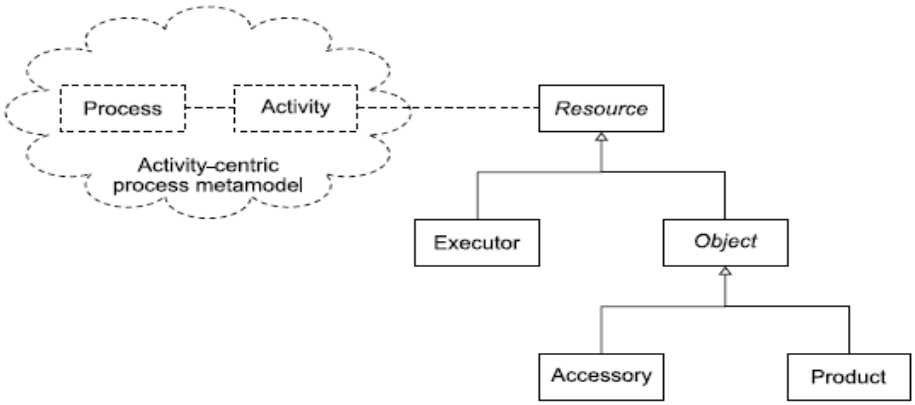


Fig. 6: Struttura del framework

1.5 Metamodello

Un metamodello è uno strumento teorico e pratico che serve a rappresentare, descrivere e analizzare modelli complessi. In altre parole, mentre un modello è una rappresentazione semplificata di un sistema o processo reale, utile per comprenderne, progettarne e comunicarne le dinamiche, un metamodello è un modello dei modelli. Esso definisce le regole, i componenti e le relazioni necessarie per creare modelli specifici.

Immaginando di voler costruire una casa. Il modello della casa includerebbe i dettagli architettonici, come le stanze, le finestre, le porte e i materiali da usare. Un metamodello, invece, stabilirebbe le linee guida generali per costruire qualsiasi casa, non solo una specifica: quali tipi di stanze possono esistere, come le stanze possono essere collegate tra loro, quali materiali sono appropriati e così via.

Nel contesto dell'informatica e dell'ingegneria del software, i metamodelli sono fondamentali per creare linguaggi di modellazione standardizzati, come UML (Unified Modeling Language) o BPMN, che permettono ai professionisti di descrivere in modo coerente e comprensibile sistemi complessi, facilitando la comunicazione e la collaborazione tra i team di sviluppo.

Il punto di partenza è stato l'identificazione delle entità primarie: Attività, Esecutore e Prodotto. Esse rappresentano i concetti fondamentali del dominio che si sta andando a sviluppare. Successivamente, sono state definite le associazioni tra le entità ed i loro specifici attributi per soddisfare i requisiti funzionali emersi dalla fase di analisi (descritti nel paragrafo 2.2), come la gestione delle attività in batch e l'assegnazione di priorità.

Durante la progettazione, si è tenuto conto di due principi fondamentali:

1. Descrivere i processi di produzione: il metamodello doveva essere in grado di rappresentare in modo fedele la struttura e il flusso dei processi produttivi, con tutti i loro elementi e le loro relazioni.
2. Servire come input per un simulatore: doveva fornire una base solida per la simulazione dei processi, fornendo le informazioni necessarie per eseguire le elaborazioni e generare i risultati desiderati.

Le entità vengono definite nel seguente modo:

- Attività: rappresenta un compito elementare all'interno di un processo di produzione. Corrisponde al componente *Task* presente in BPMN ed è stata arricchita con attributi addizionali per supportare la gestione delle attività in batch (attributo *isBatch*) e l'assegnazione di priorità (attributo *priority*).
- Esecutore: rappresenta un tipo di risorsa che eseguirà o sarà responsabile di un'attività. Ognuno di essi è caratterizzato da parametri specifici (attributi) che possono essere utilizzati per assegnare le attività alle risorse disponibili in base alle loro caratteristiche e capacità.
- Prodotto: rappresenta un singolo oggetto che viene lavorato e trasformato durante il processo di produzione. Il suo ciclo di vita è gestito dal simulatore, con la possibilità di creare nuove istanze di Prodotto durante la simulazione. A differenza delle altre entità, le istanze di questo elemento cambiano stato e caratteristiche nel corso della simulazione, riflettendo le trasformazioni subite nel processo produttivo.

Questo approccio permette di avere una rappresentazione flessibile e dettagliata dei processi produttivi, facilitando sia l'analisi che l'ottimizzazione attraverso la simulazione.

La metodologia di ricerca adottata in questa tesi si articola su due concetti cardine:

1. Adottare un approccio basato sulle attività (in particolare BPMN) come punto di partenza per la creazione di modelli per l'analisi di simulazione di processi produttivi.
2. Definire un framework generale per mettere in relazione attività e risorse, ovvero una struttura che mira a fornire un punto di vista coerente per i metamodelli che specializzano questa visione di alto livello con concetti più dettagliati, specifici per il dominio manifatturiero. In particolare, il framework sarà definito con un approccio

basato sulle risorse, al fine di colmare le lacune di BPMN in termini di modellazione delle risorse e delle loro relazioni con le attività.

L'obiettivo generale è quindi di creare una rappresentazione completa e accurata dei processi produttivi, integrando la modellazione delle attività con quella delle risorse attraverso un framework dedicato.

CAPITOLO 2 – BACKGROUND

Il lavoro dai relatori di questa tesi ha rappresentato un'importante fase preliminare per questa tesi, andando ad analizzare in modo dettagliato e approfondito i processi produttivi manifatturieri, identificando le principali sfide e le potenzialità di miglioramento. Questa fase ha costituito la base su cui è stato sviluppato il progetto di tesi, fornendo indicazioni e dati concreti per la modellazione e la simulazione dei processi.

2.1 Limiti degli approcci tradizionali

La descrizione dettagliata dei processi produttivi moderni è essenziale per affrontare le sfide poste dalla competitività del mercato, come l'ottimizzazione delle risorse e la riduzione degli sprechi, senza compromettere la qualità del prodotto. La simulazione si è dimostrata un potente strumento di supporto nella gestione dei processi produttivi, consentendo alle aziende di analizzare diversi scenari e prendere decisioni consapevoli.

Tuttavia, maggior parte delle soluzioni attualmente disponibili, sia prototipi accademici che prodotti commerciali, si basa sull'approccio centrato sulle risorse. Esempi di strumenti che adottano questo sistema sono: FlexSim®, AnyLogic®, Arena® e Warteschlangensimulator [5]. Inoltre, impiegano notazioni di modellazione non standard, spesso composte da una rappresentazione visiva arricchita con codice ad hoc (scritto in una varietà di linguaggi generici o specifici del dominio) per descrivere il processo produttivo con un alto livello di dettaglio.

L'utilizzo del codice, sebbene presentato come opzione per scenari complessi, si rivela spesso un requisito per rappresentare la complessità dei processi produttivi reali. Questo fattore rappresenta un ostacolo significativo nell'applicazione di questi strumenti da parte di utenti con limitate competenze di programmazione, rendendoli accessibili a un pubblico ristretto di esperti.

Al contrario, le notazioni incentrate sulle attività, come BPMN, offrono un metodo più intuitivo e accessibile per descrivere processi complessi, senza la necessità di ricorrere a competenze di nessun linguaggio di programmazione. La loro capacità di rappresentare le sfumature di processi articolati, come dimostrato dalla varietà di modelli di flusso di lavoro supportati, li rende particolarmente adatti alla modellazione di scenari complessi [7]. Inoltre, è ampiamente utilizzato per la creazione di modelli di processo destinati all'analisi di simulazione, consolidandone la sua idoneità per il raggiungimento degli obiettivi. Tuttavia, è principalmente focalizzato su processi aziendali generici e la sua applicazione in ambito manifatturiero si scontra con la mancanza di concetti specifici per questo dominio. Questa limitazione deriva principalmente dalla sua ridotta capacità di modellare le risorse e le relazioni tra risorse e attività.

2.2 Raccolta dei requisiti

Durante l'analisi e la raccolta dei requisiti sono state identificati diversi aspetti e opportunità per la modellazione dello sviluppo produttivo, evidenziando i limiti degli approcci tradizionali

Il primo requisito emerso pone l'attenzione sulla compatibilità prodotto-esecutore: in molti casi, un prodotto può essere realizzato solo da una specifica tipologia di macchina o da un operatore qualificato. Questa relazione non può essere modellata efficacemente con un approccio centrato sulle attività, come BPMN, che non dispone di meccanismi per rappresentare vincoli di questo tipo. Al contrario, gli approcci centrati sulle risorse, sebbene richiedano abilità nella programmazione, offrono maggiore flessibilità nel definire queste relazioni.

Un'altra problematica comune consiste nel poter definire che lo stesso compito può essere eseguito da esecutori diversi, ma ognuno di essi impiega un tempo diverso per completarlo. Ad esempio, un macchinario moderno potrebbe essere più efficiente di uno più vecchio, oppure le loro prestazioni potrebbero essere influenzate da fattori esterni. Anche in questo caso, con BPMN non è possibile modellare questo aspetto, mentre con tutti gli strumenti centrati sulle risorse analizzati, ciò può essere ottenuto solo attraverso la programmazione.

La gestione delle attività in batch rappresenta un'ulteriore esigenza emersa. In alcuni casi, è necessario raggruppare un numero di elementi prima per poter eseguire l'attività, come ad esempio il riempimento di una lavatrice con un numero minimo di capi. Questa tipologia di attività è stata oggetto di studio e proposta come estensione di BPMN, dimostrando la necessità di ampliare le sue capacità in tal senso.

Infine, nel contesto produttivo, è fondamentale poter assegnare priorità diverse alle attività, garantendo che quelle più critiche vengano completate per prime. Anche in questo caso, BPMN non dispone di funzionalità per gestire le priorità, mentre gli approcci centrati sulle risorse offrono maggiore flessibilità in tal senso.

Le sfide identificate evidenziano la necessità di un quadro concettuale più completo e flessibile per la modellazione di processi produttivi complessi. Le quattro sfide emerse:

- Relazione tra attività, tipo di prodotto ed esecutore
- Durata dell'attività influenzata dall'esecutore assegnato
- Gestione delle attività in batch
- Assegnazione di priorità diverse a diverse attività

richiedono un approccio che integri i concetti di risorse e attività in modo efficace.

2.3 Compatibilità

Nel contesto della modellazione e simulazione dei processi produttivi manifatturieri, la compatibilità gioca un ruolo fondamentale. Essa rappresenta un'associazione tra Attività, Prodotto ed Esecutore, descrivendo quale esecutore può lavorare su un determinato tipo di prodotto in un'attività specifica. Può essere considerata come una matrice per ogni attività, dove le colonne rappresentano gli esecutori possibili e le righe rappresentano i prodotti da lavorare.

Ad esempio, le tabelle 2a e 2b mostrano di due matrici di compatibilità per le attività A1 e A2. Per ogni attività, sono presenti gli esecutori nelle colonne che possono essere assegnati a quella attività: E1, E2 e E3 per A1 e E1, E4 e E5 per A2. Nelle righe, sono presenti i prodotti che verranno lavorati in quelle attività: P1 e P2. L'intersezione indica quale esecutore può lavorare su quale prodotto per l'attività: in A1, P1 può essere lavorato da E1 e E2, e P2 può essere lavorato da E2 e E3. In A2, P1 può essere lavorato da E1 e E5, mentre P2 può essere lavorato da E1 e E4. Grazie all'introduzione della compatibilità, il metamodello soddisfa i requisiti:

relazione tra attività, tipo di prodotto ed esecutore e durata dell'attività influenzata dall'esecutore assegnato.

	E1	E2	E3
P1	•	•	
P2		•	•

(a) Compatibility matrix for A1

	E1	E4	E5
P1	•		•
P2	•	•	

(b) Compatibility matrix for A2

Tab 1: Matrice di compatibilità

Inoltre, questo tipo di associazione consente di modellare quelle situazioni in cui un esecutore ha bisogno di risorse aggiuntive per svolgere l'attività. Ciò consente di soddisfare il requisito che richiede delle dipendenze multi-esecutore, ovvero la necessità di avere più esecutori per complementare un'attività.

CAPITOLO 3 – IMPLEMENTAZIONE DEL MODELLO PROPOSTO

Il terzo capitolo descriverà il lavoro svolto per implementare l'interfaccia web per il software di simulazione utilizzando bpmn.js. Saranno illustrati i passi compiuti per creare un modello pratico.

3.1 Analisi del software

È stata condotta un'analisi sugli strumenti software già esistente, capaci di poter fornire modellazione e simulazione di processi produttivi, con l'obiettivo di individuare la soluzione più idonea. Tra le alternative esaminate, figurano strumenti come Camunda [10], Bizagi [11], jBPM [12] e bpmn.js [13], ognuno con caratteristiche peculiari e specifiche.

- Camunda: una piattaforma BPM (Business Process Management) open-source che supporta la modellazione mediante notazione BPMN. Nonostante sia uno strumento potente, la sua integrazione e il suo utilizzo efficace richiedono competenze avanzate in configurazione e programmazione. Questo fattore la rende meno accessibile a utenti privi di conoscenze tecniche approfondite, limitandone l'utilizzo. Inoltre, non è progettato per essere facilmente esteso o personalizzato. Ciò significa che le sue funzionalità di base potrebbero non essere sufficienti per soddisfare i requisiti specifici del progetto, che richiede un elevato grado di flessibilità e personalizzazione.
- Bizagi: questo software offre un ambiente user-friendly e una robusta piattaforma di modellazione e gestione dei processi. Tuttavia, la sua versione completa è a pagamento e può risultare costosa, mentre le funzionalità della versione gratuita sono limitate. Questo restringe le possibilità di personalizzazione necessarie per il progetto.
- jBPM: anch'essa open-source ed utilizza BPMN per la modellazione dei processi. Si distingue per la sua facilità nell'essere personalizzabile e adattabile a integrazioni complesse. Analogamente a Camunda, presenta una curva di apprendimento ripida e richiede competenze tecniche specifiche per la sua configurazione e il suo utilizzo efficiente, rendendolo non idoneo allo scopo.
- bpmn.js: un toolkit open-source specificamente progettato per la modellazione e l'esecuzione di diagrammi BPMN in ambiente JavaScript. La scelta è ricaduta su questa libreria per i seguenti motivi:
    - Facilità d'uso: è concepito per essere utilizzato anche da utenti senza la conoscenza di concetti di programmazione. La sua interfaccia intuitiva e la sua documentazione completa permettono di creare e modificare modelli di processo con una minima formazione, rendendolo uno strumento accessibile a un'utenza più ampia.
    - Flessibilità e personalizzazione: la sua natura open-source, offre un elevato grado di personalizzazione. Questo permette di adattare il tool alle specifiche esigenze del progetto senza costi aggiuntivi, implementando funzionalità e modifiche in base alle necessità concrete.
    - Integrazione web: essendo basato su JavaScript, si integra facilmente con applicazioni web esistenti. In questo modo si favorisce una distribuzione e un

accesso più agevole ai modelli di processo, permettendo di integrarli perfettamente con i sistemi web già in uso.

- Comunità e supporto: vanta una community attiva di sviluppatori, garantendo un supporto costante in caso di problemi o dubbi, facilitando la risoluzione di eventuali criticità e l'implementazione di nuove funzionalità.

3.2 Caso di studio

Per valutare l'approccio scelto, sono stati combinati scenari reali utili a creare un caso unico e più complesso. Il risultato è un processo che descrive come vengono prodotti i telai per occhiali in un ambiente di produzione.

È stato costruito un modello per questo scenario derivato dal metamodello e pronto per essere creato con un modellatore. Il caso può essere riassunto come segue:

> L'obiettivo è produrre 100 telai di tipo A e 100 telai di tipo B, ciascuno composto da due steli (sinistro e destro) e dal supporto per le lenti frontali. I componenti vengono stampati su una stampante a pressione, con tre stampanti disponibili (PP1, PP2, PP3). I gambi di tipo A possono essere stampati solo su PP1, mentre i gambi di tipo B possono essere stampati su PP1 e PP2. Il telaio frontale per entrambi i tipi A e B può essere stampato con uno qualsiasi dei tre stampanti. Ogni stampante richiede 1 minuto per stampare un pezzo e richiede lo stampo corrispondente, con un solo stampo disponibile per ogni pezzo. Le stampanti hanno un tempo di avvio di 10 minuti e la stampa dei telai frontali ha una priorità più elevata. Un operatore assembla manualmente il telaio finale, con tre operatori disponibili (O1, O2 e O3) con tempi di assemblaggio diversi. I telai vengono quindi posizionati in un tumbler per 24 ore, con una capacità di 15 telai alla volta. Un controllo di qualità viene effettuato ogni 20 telai da due operatori (O4 e O5), con 2 minuti di tempo di controllo, quelli risultati difettosi vengono scartati e ristampati, mentre quelli idonei vengono conservati in un magazzino in attesa della verniciatura, che viene effettuata fuori sede e richiede 2 giorni.

La figura 7 rappresenta il processo utilizzando una rappresentazione basata su BPMN. Sono utilizzi esagoni per rappresentare gli esecutori e collega con una linea tratteggiata le attività in cui possono essere utilizzati.

Nel modello, le attività sono arricchite con proprietà per la priorità e le caratteristiche di batch, non mostrate nel diagramma. Ad esempio, "Stampa telaio frontale" ha una priorità più elevata nel contesto esaminato e "Pulisci telaio" viene considerato batch. Anche se non completamente mostrato, il modello si è dimostrato in grado di catturare tutti gli aspetti del processo di produzione dei telai per occhiali.

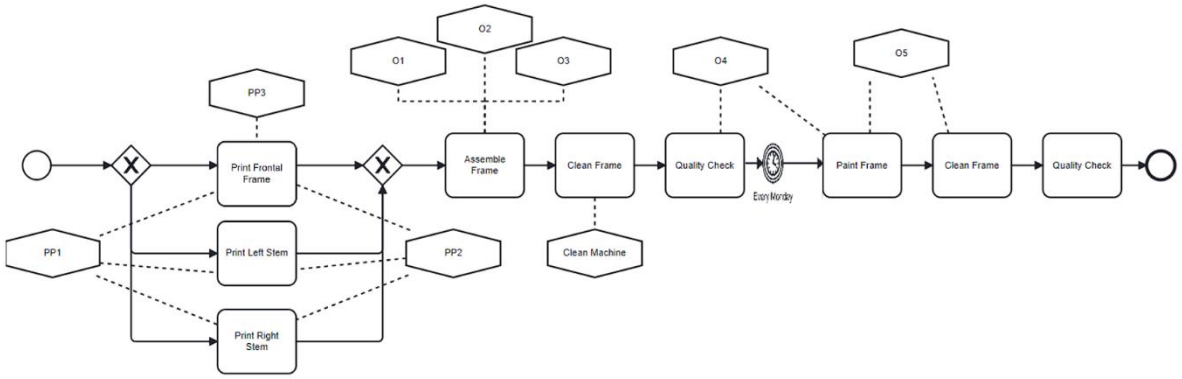


Fig. 7: Diagramma rappresentante lo scenario del caso di studio

Nei prossimi capitoli, verranno descritti dettagliatamente l'interfaccia utente (Capitolo 4), i processi di implementazione (Capitolo 5) e la struttura del codice (Capitolo 6), che insieme costituiscono il nucleo del lavoro di tesi, dimostrando l'applicabilità pratica del framework concettuale e del metamodello proposto.

CAPITOLO 4 – INTERFACCIA UTENTE

In questo capitolo verrà descritta l'interfaccia del progetto di tesi, che utilizza bpmn.js per la modellazione di un processo produttivo. Verranno esaminati in dettaglio gli elementi personalizzati introdotti, le funzionalità del pannello delle proprietà e le modalità con cui queste componenti interagiscono per facilitare la definizione e la gestione dei processi di produzione.

4.1 Panoramica della schermata

L'interfaccia utente dell'applicazione è organizzata in modo da facilitare la modellazione dei processi attraverso una serie di componenti visive e interattive. La schermata principale è divisa in tre sezioni principali:

- Palette degli elementi: situata a sinistra, contenente gli strumenti e gli elementi BPMN che possono essere utilizzati per creare il diagramma di flusso.
- Area di disegno: al centro, dove gli utenti possono trascinare e rilasciare gli elementi della palette per costruire il loro modello di processo.
- Pannello delle proprietà: posizionato a destra, mostra le proprietà dettagliate degli elementi selezionati nel diagramma, consentendo la loro modifica e personalizzazione.

Questa struttura consente agli utenti di accedere rapidamente agli strumenti necessari e di visualizzare le informazioni rilevanti per una modellazione efficiente e intuitiva.

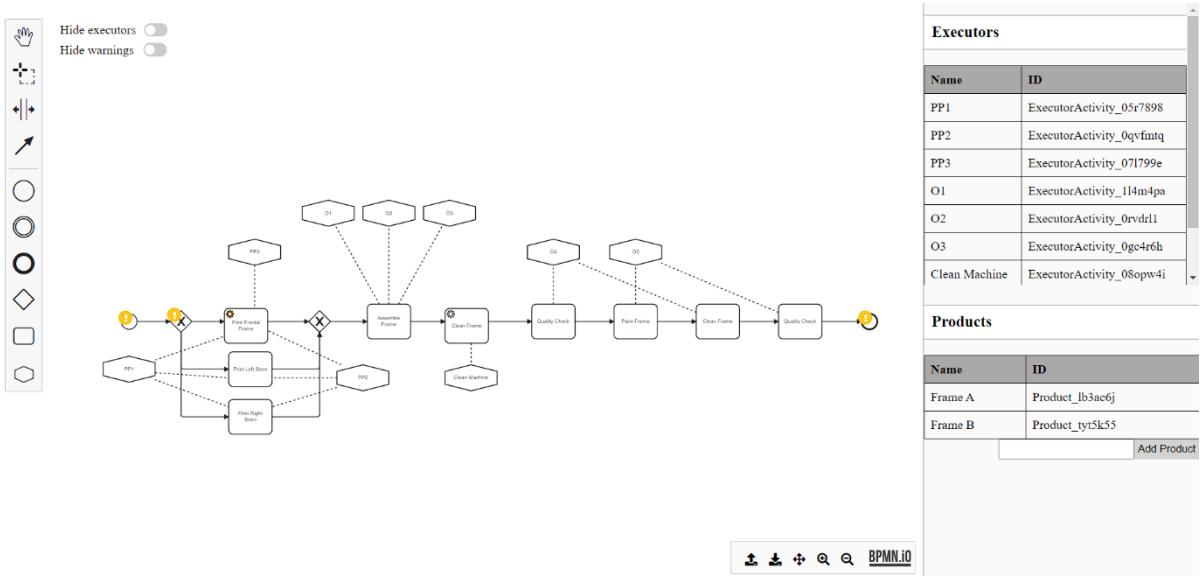


Fig. 8: Panoramica della schermata al momento della creazione

4.2 Elemento Executor

L'elemento *Executor* è stato introdotto nel progetto per rappresentare una componente chiave del processo, ovvero le risorse che eseguono specifiche attività all'interno del flusso di lavoro.

Le proprietà principali dell'elemento esecutore sono ereditate componente *Activity*, descritto nel paragrafo 4.3, garantendo così la compatibilità e l'integrazione con il modello. Tuttavia, per adattarsi alle necessità specifiche del progetto, è stata dichiarata una proprietà aggiuntiva, *Product*, che consente di associare all'*Executor* uno o più prodotti. In questo modo è possibile definire quali prodotti possono essere manipolati dall'esecutore, migliorando così la personalizzazione del modello di processo.

Graficamente, l'esecutore è rappresentato da una forma esagonale, scelta che facilita il riconoscimento visivo e differenzia questo elemento dalle altre componenti all'interno del diagramma di flusso. Inoltre, è stato integrato nella palette degli elementi selezionabili nella parte sinistra della schermata, rendendolo facilmente accessibile per gli utenti durante l'utilizzo del software.

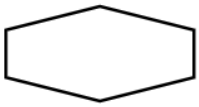


Fig. 9: Rappresentazione grafica dell'esecutore

4.3 Elemento Activity

L'elemento *Activity*, descritto nel paragrafo 1.1 come componente attività, funge da nodo centrale nel modello in quanto può essere connesso sia agli elementi standard di BPMN sia agli esecutori e ai prodotti. Questo lo rende un componente cruciale per la definizione e la gestione dei processi produttivi.

Graficamente è rappresentata, come illustrato nella figura seguente, da un rettangolo con bordi arrotondati.

Fig. 10: Rappresentazione grafica dell'activity

4.3.1 Elemento Batch

Una specializzazione dell'*Activity* è l'elemento *Batch* che viene utilizzato per indicare attività che coinvolgono l'elaborazione di gruppi di prodotti, anziché singoli elementi. In altre parole, un'attività *Batch* si occupa della gestione e lavorazione di un insieme di articoli

contemporaneamente, ottimizzando così l'efficienza produttiva e riducendo i tempi di attesa e i costi operativi.

Graficamente, è rappresentato da una forma rettangolare con bordi arrotondati e un'icona a forma di ingranaggio nell'angolo in alto a sinistra. Questa rappresentazione visiva aiuta a distinguere rapidamente le attività Batch dalle normali attività di processo, citate nel paragrafo precedente.
L'icona dell'ingranaggio può presentarsi in due varianti cromatiche: bianca o arancione. Questa differenziazione è importante poiché fornisce un'indicazione visiva sullo stato e la configurazione del componente.

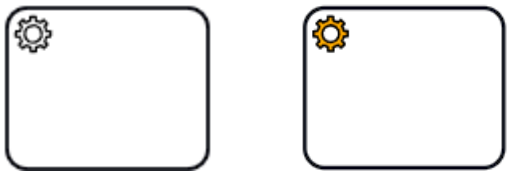


Fig. 11: Rappresentazione grafica delle due tipologie di batch activity

Un ingranaggio bianco indica che l'attività *Batch* è completamente configurata per elaborare gruppi di prodotti, in cui tutti i prodotti associati hanno un valore di batch maggiore di uno. Un ingranaggio arancione, invece, segnala che almeno uno dei prodotti associati all'attività ha un valore di batch uguale a uno, suggerendo che non tutti i componenti sono configurati per il trattamento in lotti e che potrebbe essere necessario un intervento per ottimizzare l'attività.

Le proprietà dell'elemento sono principalmente ereditate dall'*Acitivity*, il che garantisce che tutte le funzionalità di base delle attività siano mantenute, inclusa la possibilità di collegare l'attività a esecutori, risorse e altre attività nel diagramma di flusso.

Per utilizzare questo oggetto, l'utente deve prima creare un elemento *Activity* e poi modificarne il tipo attraverso l'opzione 'change element', accessibile tramite un'icona a forma di chiave inglese. Questa procedura consente di trasformare una normale attività in un'attività *Batch*, applicando le proprietà e la logica di batch processing necessarie per gestire gruppi di prodotti.

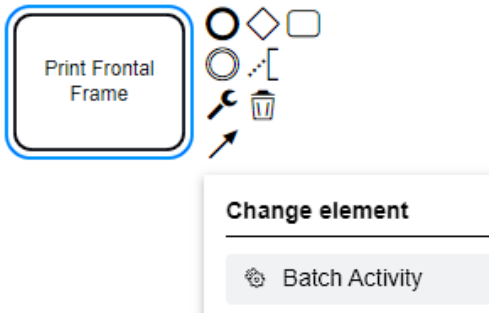


Fig. 12: Rappresentazione grafica della funzionalità 'Change element' per creare l'elemento Batch

4.4 Altri elementi

4.4.1 Elemento Product

L'elemento *Product* non ha una visualizzazione grafica diretta nel diagramma, ma la sua inclusione è stata fondamentale per definire quali prodotti possono essere utilizzati da un Executor.

4.4.2 Elemento Connection

Il componente *Connection* è stato introdotto per fornire un collegamento visivo chiaro tra le *Activity* o i *Batch* e gli *Executor*. È rappresentato graficamente da una linea tratteggiata che connette le due componenti nel diagramma di flusso, andando a migliorare la leggibilità del diagramma e a rendere immediatamente evidente quali esecutori sono associati a quali attività.

4.5 Pannello delle Proprietà

Il pannello delle proprietà, situato nella parte destra della schermata, è stato sviluppato per permettere la definizione e la modifica delle proprietà degli elementi all'interno del diagramma. Esso offre una visione dettagliata e personalizzabile delle caratteristiche di ciascun elemento selezionato, migliorando l'usabilità e la flessibilità della piattaforma.

L'interfaccia del pannello di proprietà presenta un aspetto diverso a seconda che l’utente selezioni o meno un elemento.

4.5.1 Nessun elemento selezionato

Quando nessun elemento è selezionato, viene mostrata una panoramica degli *Executors* e dei *Products* definiti nel sistema. Questa sezione elenca gli ID e i nomi di tutti gli esecutori e i prodotti disponibili, permettendo agli utenti di visualizzare rapidamente i componenti principali già creati nel processo.

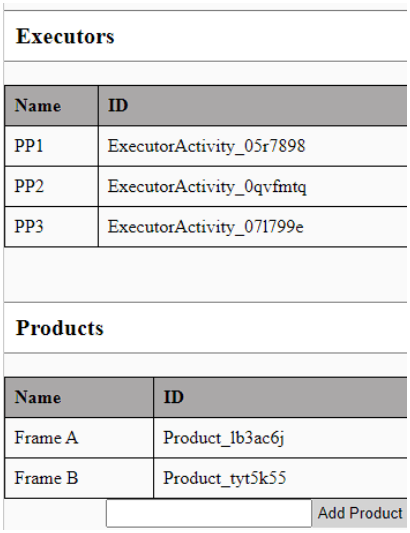


Fig. 13: Pannello di proprietà con nessuno elemento selezionato

4.5.2 Elemento selezionato

Quando viene selezionato un elemento nel diagramma, il pannello cambia dinamicamente per riflettere le proprietà specifiche di quell'elemento. Nella parte superiore viene mostrato:

- Un'etichetta che indica il tipo di elemento selezionato (*Task*, *Executor*, *Batch*, …).
- Un campo di input di sola lettura mostra l'ID dell'elemento selezionato, permettendo all'utente di identificarlo univocamente nel contesto del diagramma.
- Un campo di input con un testo modificabile consente di cambiare il nome dell'elemento, offrendo la possibilità di personalizzare le etichette per una migliore comprensione e gestione del flusso di lavoro.

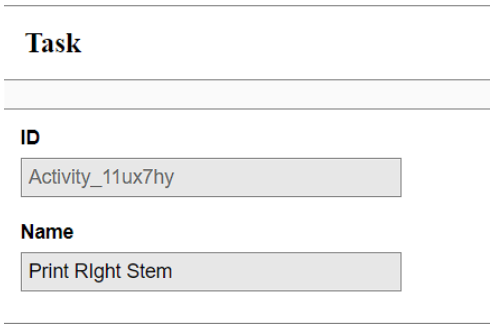


Fig. 14: Pannello di proprietà con un elemento selezionato (esempio con elemento di tipo Activity)

Al di sotto il pannello è organizzato in sezioni che variano a seconda del tipo di elemento selezionato e che hanno una funzionalità di “comprimi ed espandi”.

Elemento Executor:

- Sezione ‘Connected Products’: visualizza la lista dei prodotti associati all'esecutore, includendo il nome del prodotto e l'attività a cui è collegato.

    - Per ogni *Product* viene visualizzata un'area di input per inserire o modificare il tempo di esecuzione e un menù a tendina che consente di selezionare un'unità di tempo (secondi, minuti, ore, giorni). Inoltre, se l’esecutore è connesso ad un elemento di tipo Batch, viene mostrato un campo input che consente di inserire il numero di elementi necessari per poter eseguire la lavorazione in batch.

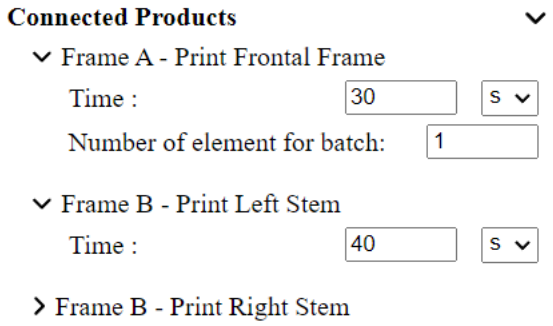


Fig. 15: Pannello di proprietà: sezione ‘Connected Products’

Elemento Activity:

- Sezione ‘Connected Executors’: mostra la lista degli esecutori collegati all'attività selezionata.
    - L’icona con il simbolo “X” permette di rimuovere un esecutore connesso all'*Activity*, eliminando anche la linea tratteggiata che li collega.

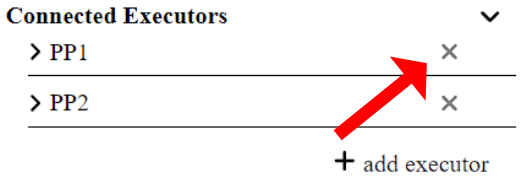


Fig. 16: Pannello di proprietà: sezione ‘Connected Executors’ con puntatore all’icona per rimuovere l’esecutore

    - Il pulsante *add executor* permette di collegare un esecutore non ancora associato all’attività, tramite una barra di ricerca. Nel momento in cui viene selezionato un elemento esecutore, viene creata la connessione tra i due elementi (*Activity-Executor*) attraverso la linea tratteggiata. In questo modo, viene consentito di

creare collegamenti sia direttamente nello spazio di disegno che dal pannello di proprietà.

Andando a selezionare un esecutore si espande la sezione e viene mostrata la lista dei prodotti associati, consentendo di gestire ogni prodotto singolarmente. È possibile:

- Definire il tempo di esecuzione (time) e l'unità di misura (secondi, minuti, ore, giorni).
- Aggiungere un prodotto attraverso il pulsante *add product*, tramite una barra di ricerca.
- Rimuovere un prodotto, utilizzando il pulsante con l'icona "X", posto sulla stessa linea del nome del prodotto.

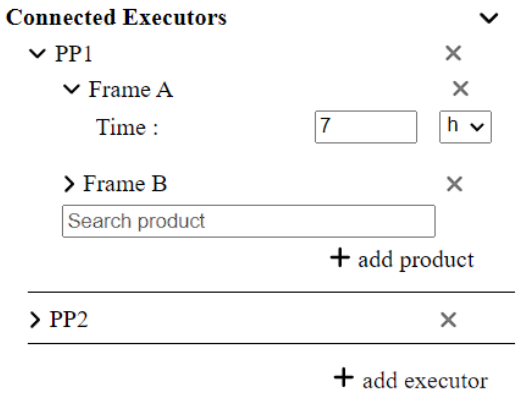


Fig. 17: Pannello di proprietà: sezione 'Connected Executors' espansa

- Sezione 'Activity Priority': permette di definire una priorità numerica per l'Activity.

Fig. 18: Pannello di proprietà: sezione 'Activity Priority'

Elemento Batch:

Il pannello delle proprietà per l'elemento Batch è molto simile all'elemento Activity appena descritto. Tuttavia, all'interno della sezione che mostra i prodotti associati all'attività, è presente l'opzione per impostare il valore del batch.

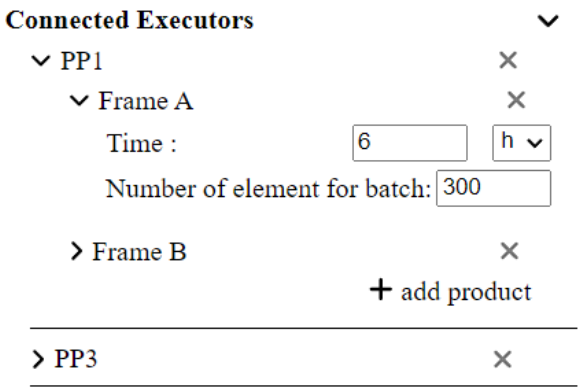


Fig. 19: Pannello di proprietà: sezione ‘Connected Executors’ con batch input

Per altri tipi di elementi nel diagramma, non sono definite sezioni aggiuntive specifiche nel pannello delle proprietà per mantenere l'interfaccia utente semplice e non sovraccaricarla, evitando di renderne più complesso l’utilizzo.

4.6 Palette

La palette di bpmn.js è uno strumento fondamentale per la modellazione dei processi, permettendo agli utenti di selezionare e posizionare le varie componenti all'interno del diagramma di flusso. Essa fornisce una raccolta di componenti predefiniti che possono essere facilmente trascinati e rilasciati nell'area di lavoro, facilitando così la creazione e la modifica dei processi.

Nel contesto del progetto sperimentale, la palette è stata personalizzata per soddisfare esigenze specifiche, in particolare gli elementi selezionabili sono: *Start Event*, *Intermediate Event*, *End Event*, *Gateway*, *Task* ed *Executor*.

Fig. 20: Palette personalizzata

4.7 Visibilità esecutori

Per migliorare l'usabilità dell'interfaccia e garantire una chiara comprensione del diagramma di flusso, è stato introdotto uno switch button posizionato nella parte in alto a sinistra dell'interfaccia, permettendo agli utenti di mostrare o nascondere alcune componenti aggiuntive inserite all'interno di BPMN, in particolare gli esecutori e i connettori tratteggiati.

La scelta di introdurre questa funzionalità è stata motivata dalla necessità di mantenere il diagramma pulito e leggibile. Nei processi industriali complessi, è frequente l'utilizzo di numerosi esecutori. Senza un controllo adeguato della visibilità, questi elementi aggiuntivi possono rendere il diagramma eccessivamente carico e difficile da interpretare. Pertanto, la possibilità di nascondere temporaneamente gli esecutori e i connettori tratteggiati consente agli utenti di concentrarsi sugli aspetti più rilevanti del diagramma.

Di seguito, una figura mostra l'aspetto del diagramma prima e dopo l'utilizzo dello switch button per la visibilità degli esecutori, evidenziando la differenza in termini di chiarezza e semplicità visiva.

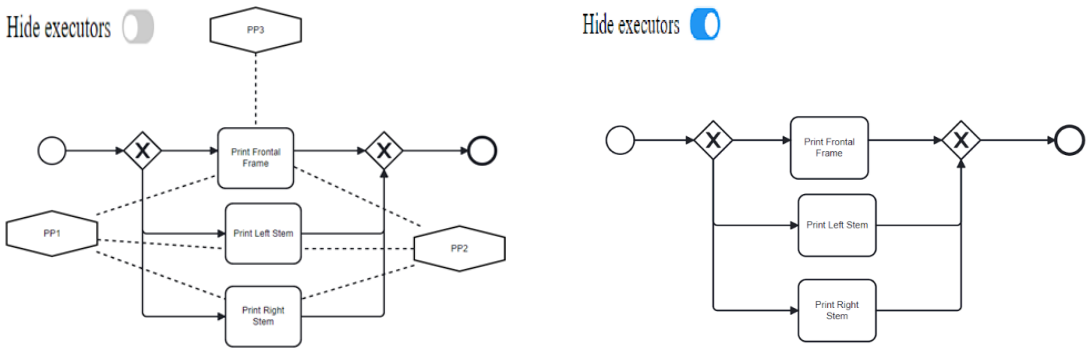


Fig. 21: Rappresentazione del diagramma con esecutori e connettori tratteggiati mostrati (figura a sinistra) e nascosti (figura a destra)

4.8 Visibilità regole BPMN

Analogamente alla gestione della visibilità degli esecutori, è stato implementato uno switch button per controllare la visibilità delle regole di bpmn.js. Questo pulsante permette di mostrare o nascondere le regole di convalida (*linting*) che verificano la correttezza e la conformità del modello BPMN.

La validazione del modello è essenziale per assicurare che il diagramma di flusso rispetti gli standard BPMN e che non vi siano errori o incongruenze.
Sono mostrate due tipologie principali di regole: warning ed error. Le regole di tipo error segnalano violazioni gravi dello standard BPMN, che potrebbero compromettere il funzionamento del processo; invece, quelle di tipo warning indicano situazioni potenzialmente problematiche che meritano attenzione, ma che non impediscono necessariamente il corretto funzionamento del diagramma.

Due regole specifiche sono state aggiunta nel progetto e considerate di tipo warning. Viene controllato che a ogni esecutore sia associato almeno un prodotto e che per ogni elemento *Batch* sia collegato almeno ad un elemento *Executor*. Se queste condizioni non sono soddisfatte, viene emesso un avviso per informare l'utente della possibile presenza di una configurazione non corretta del diagramma. Sebbene queste situazioni non costituiscano un errore grave, è comunque importante prestare attenzione per garantire un funzionamento ottimale del processo.

Di seguito, una figura mostra l'aspetto del diagramma prima e dopo l'utilizzo dello switch button per la visibilità delle regole BPMN, evidenziando come questa funzionalità aiuti a mantenere un ambiente di lavoro più pulito e focalizzato durante la modellazione.

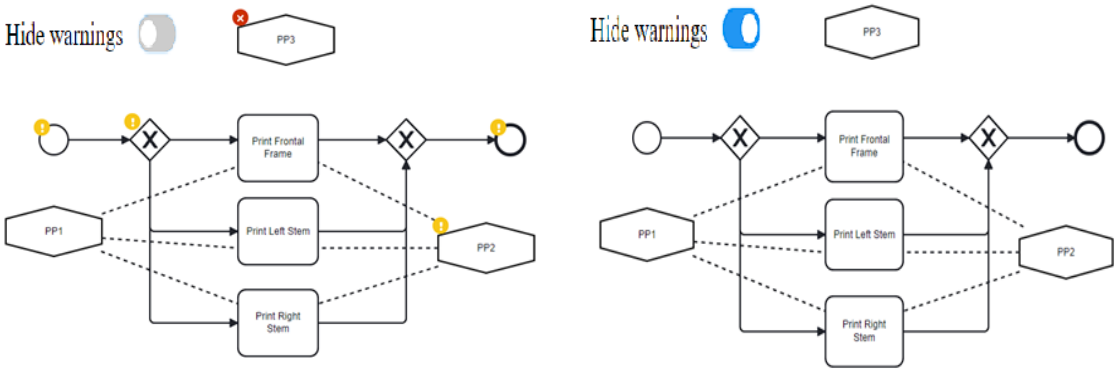


Fig. 22: Rappresentazione del diagramma con regole mostrate (figura a sinistra) e nascoste (figura a destra)

4.9 Controlli

I controlli consentono agli utenti di eseguire rapidamente operazioni chiave durante la creazione e la gestione dei diagrammi di flusso. Di seguito viene fornita una descrizione dettagliata dei controlli implementati, inclusi gli obiettivi e le modalità di utilizzo di ciascuno di essi.

Fig. 23: Rappresentazione dei controlli

- Caricamento del diagramma BPMN: rappresentato da un'icona di caricamento (1), consente agli utenti di importare un diagramma BPMN esistente nel modellatore. Questa funzionalità è stata progettata per facilitare l'importazione di diagrammi creati in precedenza, permettendo agli utenti di continuare a lavorare sui loro modelli senza dover ricreare tutto da zero. La possibilità di caricare un diagramma è fondamentale per garantire la continuità del lavoro e la collaborazione tra diversi utenti che possono condividere i loro diagrammi.
- Download del diagramma BPMN: rappresentato da un'icona di download (2), permette agli utenti di scaricare il diagramma corrente in formato XML. Questo controllo è

essenziale per la conservazione e la condivisione dei diagrammi, consentendo agli utenti di salvare il lavoro svolto e di distribuirlo ad altri membri del team o di archiviare versioni specifiche del modello.

- Centraggio del diagramma: rappresentato da un'icona di centraggio (3), e consente agli utenti di visualizzare l'intero processo al centro dell'area di lavoro del modellatore. Questa funzione è particolarmente utile quando si lavora su diagrammi di grandi dimensioni che possono risultare fuori vista o disallineati. Il centraggio migliora la navigabilità e la visibilità del flusso di lavoro, assicurando che tutti gli elementi siano facilmente accessibili.
- Zoom in: rappresentato da un'icona di ingrandimento (4), consente agli utenti di ingrandire il diagramma per vedere i dettagli con maggiore chiarezza. Questo è particolarmente utile quando si lavora su parti complesse del diagramma che richiedono un'attenzione particolare ai dettagli.
- Zoom out: rappresentato da un'icona di riduzione (5), permette agli utenti di ridurre il diagramma per avere una visione d'insieme più ampia. Questa funzione è utile per esaminare l'intero flusso di lavoro e comprendere come le diverse parti del diagramma si collegano tra loro.

CAPITOLO 5 – DETTAGLI IMPLEMENTATIVI

Questo capitolo descrive in dettaglio i processi di implementazione del progetto, focalizzandosi sull'analisi preliminare, sulla scelta del software, sulla creazione di mock-up e sullo sviluppo delle funzionalità necessarie per il raggiungimento degli obiettivi prefissati.

Ogni iterazione ha contribuito a migliorare la rappresentazione delle relazioni tra attività, esecutori e prodotti, garantendo una maggiore intuitività e facilità d'uso. L'implementazione delle funzionalità ha seguito un percorso metodico, integrando progressivamente gli elementi necessari, tra i quali l'elemento *Executor*, la gestione delle connessioni, lo sviluppo del pannello delle proprietà, il sistema di validazione e i controlli dell'interfaccia, assicurando così un'applicazione robusta e user-friendly.

5.1 Analisi preliminare e scelta del software

Il primo passo del progetto ha riguardato una scomposizione dettagliata del problema per ottenere una visione completa del contesto in cui esso si inserisce. Durante questa fase si è svolta un'analisi approfondita dei diversi software esistenti e idonei a rappresentare adeguatamente il problema.

Innanzitutto, si è optato che la soluzione ottimale fosse la rappresentazione del problema attraverso un diagramma. Successivamente, sono stati valutati i diversi software, concludendo che bpmn.js fosse la scelta ideale per la modellazione di diagrammi BPMN.

La scelta di bpmn.js è stata guidata da fattori quali la facilità d'uso, la flessibilità e personalizzazione, l'integrazione web e il supporto di una comunità di sviluppatori, come discusso approfonditamente nel paragrafo 3.1. La natura open-source di bpmn.js permette modifiche e adattamenti specifici, rendendolo ideale per le esigenze del progetto.

5.2 Creazione dei mock-up

Alla scelta di bpmn.js come software per la modellazione dei processi, è seguita la creazione dei mock-up per visualizzare e perfezionare l'interfaccia utente ideale. L'obiettivo era garantire che essa fosse intuitiva e facile da usare anche per utenti privi di conoscenze di programmazione. Questa fase è stata molto importante per identificare e risolvere sia problemi concettuali che logici nella rappresentazione dei processi produttivi.

Il primo mock-up mostra un pannello di proprietà personalizzato per la gestione degli elementi BPMN, posizionato sulla destra dell'interfaccia, diviso in due aree quando viene selezionato un elemento di tipo *Activity*:

- Sezione 'Executor': permette all'utente di aggiungere gli esecutori associati all'attività e definirne le proprietà.

- Sezione ‘Product’: permette di aggiungere i prodotti associati all'attività e di stabilire le relazioni tra essi e gli esecutori. In particolare, sono mostrate due modalità per mettere in relazione i prodotti con gli esecutori: una lista di elementi selezionabili tramite checkbox oppure una barra di ricerca per poterli associare.

Tuttavia, questo approccio ha presentato diverse criticità:

- Proprietà diverse per prodotto: le proprietà degli elementi variavano da prodotto a prodotto, rendendo inefficace l'inserimento delle proprietà sotto gli esecutori.
- Selezione ridondante e complessa: la combinazione di esecutori e prodotti richiedeva più passaggi, risultando macchinosa e inefficiente. In particolare, la modalità tramite checkbox era problematica, soprattutto con liste lunghe di esecutori.
- Visualizzazione delle relazioni: era necessario selezionare ogni attività e navigare tra le liste per comprendere le relazioni tra gli elementi, rendendo difficile una visione d'insieme immediata.

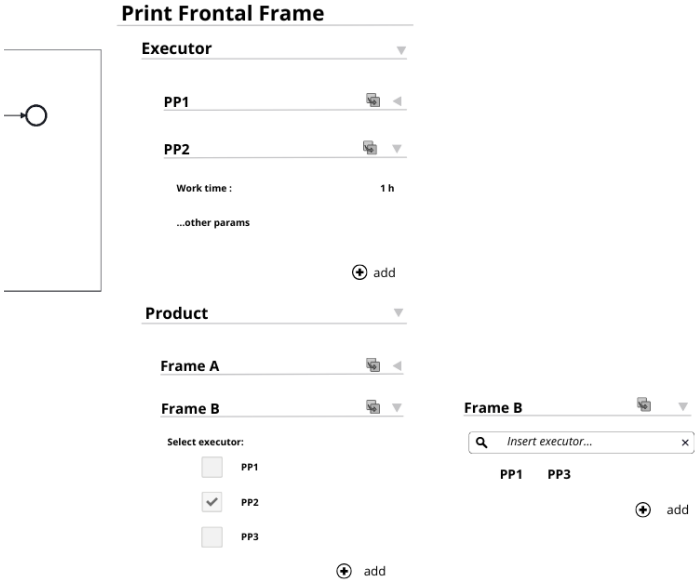


Fig. 24: Rappresentazione primo mock-up

Successivamente, sono state esplorate alcune alternative per migliorare la rappresentazione delle relazioni.

Un secondo mock-up presenta una zona inferiore divisa in due parti, una per gli esecutori e l'altra per i prodotti, ma la gestione degli elementi rimaneva simile al caso precedente, mantenendo le stesse problematiche.

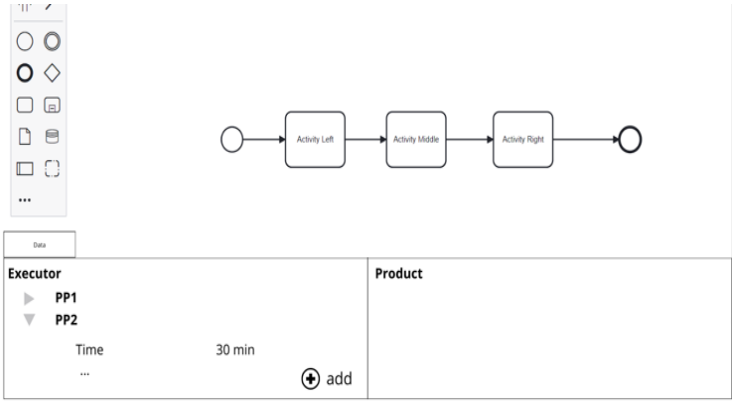


Fig. 25: Rappresentazione secondo mock-up

Un'altra alternativa ha introdotto due tab nella parte inferiore dell'interfaccia, suddivisa come nei casi precedentemente descritti in ‘Executor’ e ‘Product’. Questo approccio, a differenza degli altri modelli, permetteva una visione più chiara delle relazioni esistenti perché consentiva di visualizzare direttamente per ogni esecutore il relativo nome, Id e le connessioni create, e allo stesso modo per i prodotti. Tuttavia, anche questa soluzione aveva dei limiti, specialmente quando le connessioni erano numerose, rendendo la lista difficile da gestire. Inoltre, la gestione delle relazioni era ancora suddivisa tra la tab nella parte inferiore e il pannello di proprietà a destra dell’interfaccia, creando confusione.

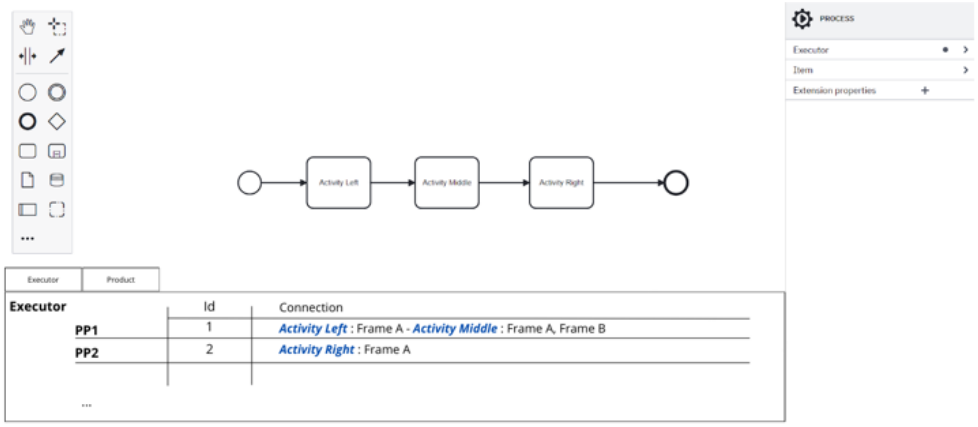


Fig. 26: Rappresentazione terzo mock-up

Un’ulteriore soluzione è stata sperimentata attraverso una tabella bidimensionale [Prodotto, Esecutore] che veniva mostrata quando il puntatore del mouse si posizionava sopra un elemento *Activity* e una 'x' nelle celle indicava dove era presente una relazione.

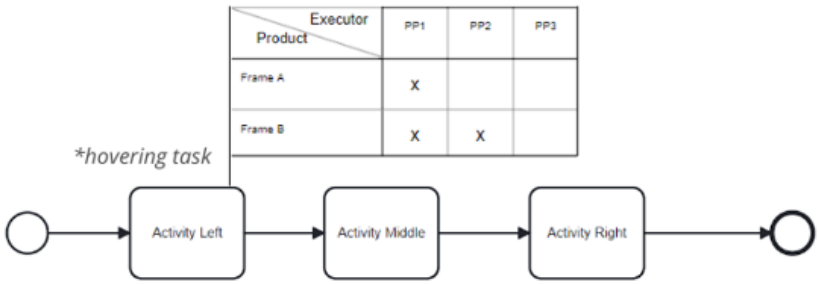


Fig. 27: Rappresentazione quarto mock-up

Il quinto mock-up creato gestisce le relazioni con uno scenario differente. L'utente, in modo molto semplice, avrebbe selezionato uno switch button situato in alto a sinistra della schermata, nascondendo il diagramma e mostrando una matrice tridimensionale [*Activity*, *Executor*, *Product*] che illustrava le relazioni tra i tre elementi. Questa idea era stata presa in considerazione come possibile alternativa o per offrire una visualizzazione aggiuntiva.

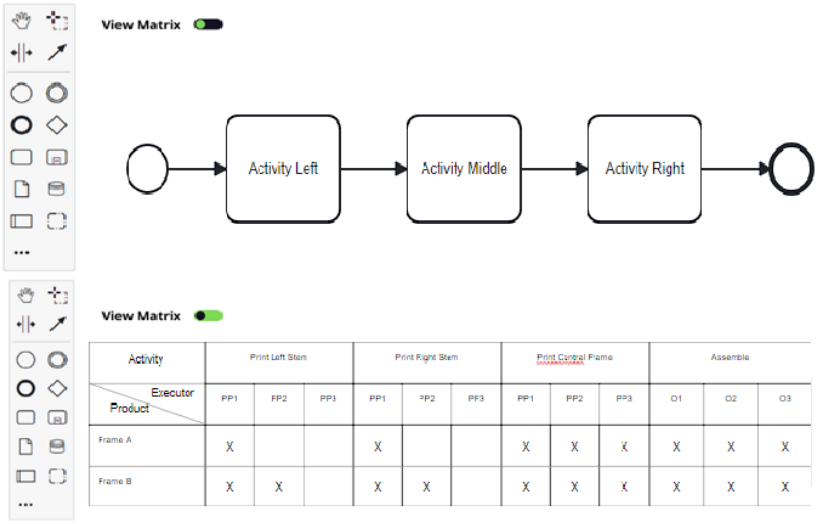


Fig. 28: Rappresentazione quinto mock-up

Dopo varie iterazioni e feedback, è stato creato un mock-up finale che integrava le migliori caratteristiche delle soluzioni precedenti:

- Un pannello di proprietà unificato che mostrava chiaramente le relazioni tra attività, esecutori e prodotti e che consentiva una gestione semplice e intuitiva delle stesse.
- Implementazione di funzioni di ricerca e filtro per facilitare la selezione degli esecutori e dei prodotti in presenza di liste lunghe.

Questo modello è stato utilizzato come base per le successive implementazioni del software, assicurando che le esigenze degli utenti fossero al centro del processo di sviluppo.

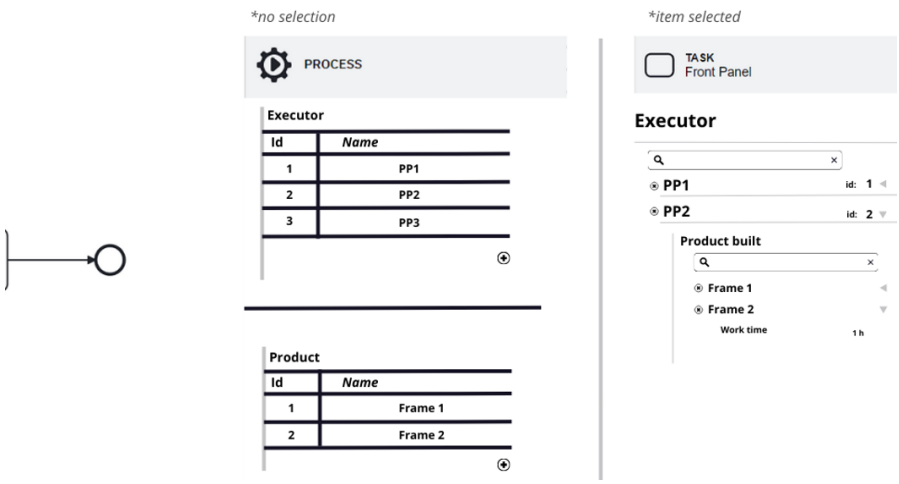


Fig. 29: Rappresentazione sesto mock-up

5.3 Implementazione iniziale

Lo sviluppo del codice è partito da una base di bpmn.js. Inizialmente, è stata fatta una ricerca di una soluzione già esistente che si avvicinasse il più possibile ai requisiti da soddisfare. In questa fase esplorativa, sono stati individuati diversi progetti, strumenti e librerie che potevano velocizzare la fase iniziale del lavoro o addirittura assolvere alcune delle funzionalità desiderate. Tra le opzioni esaminate, una libreria che estende bpmn.js [14] metteva a disposizione un pannello di proprietà con un'interfaccia grafica intuitiva adatto allo scopo. Tuttavia, la sua natura statica richiedeva la compilazione di informazioni non pertinenti, complicando la definizione delle proprietà degli elementi.

Proseguendo la ricerca, è stata trovata un'altra libreria che estende bpmn.js [15], il quale integra un pannello di proprietà personalizzabile ed inoltre sfruttava React. Questa soluzione si è dimostrata ideale, in quanto ha permesso di creare una componente personalizzata da zero, adattandola perfettamente alle esigenze e garantendo la compatibilità con bpmn.js.

5.4 Implementazione delle funzionalità

5.4.1 Introduzione dell'elemento esecutore

Uno dei primi miglioramenti identificati è stato l'inserimento di un elemento esecutore all'interno del diagramma. Dopo aver esaminato le figure già presenti nello standard BPMN, è stato deciso di introdurre una figura esagonale per rappresentare l'*Executor*, migliorando significativamente la comprensione del diagramma.

5.4.2 Problemi di visualizzazione e soluzione

L'aggiunta dell'elemento ha introdotto un problema visivo: le connessioni tra il nuovo componente e le activity venivano mostrate tramite frecce standard, creando confusione e rendendo la visualizzazione generale del diagramma più complessa. Per risolvere questo problema, è stata modificata la rappresentazione delle connessioni.

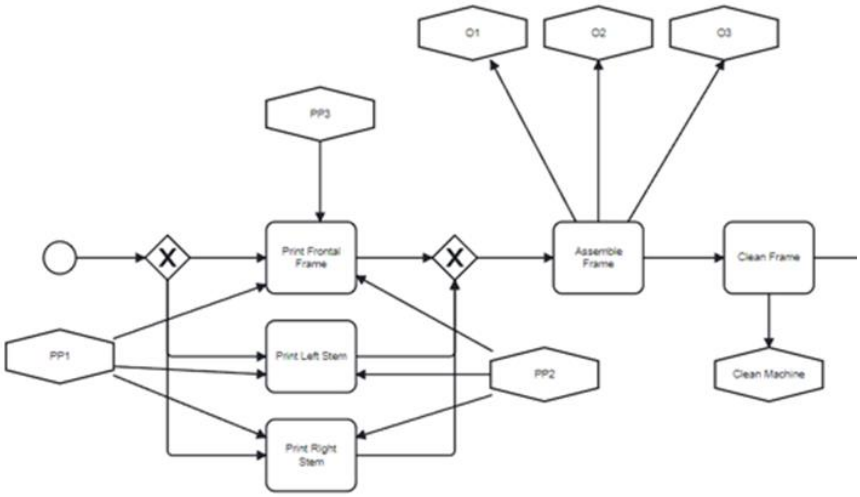


Fig. 30: Rappresentazione del diagramma con connettori standard di bpmn.js

Dopo diverse considerazioni, si è optato per l'introduzione di un nuovo tipo di connessione denominato *factory:Connection*, rappresentato da una linea tratteggiata. Questo ha permesso di chiarire meglio le relazioni tra gli elementi senza compromettere la leggibilità del diagramma. È stata anche aggiunta una funzione per nascondere dalla visualizzazione sia l'esecutore che la *factory:Connection*, offrendo un'opzione di visualizzazione più pulita.

5.4.3 Gestione del file XML

La sfida successiva è stata garantire che i nuovi elementi e le connessioni fossero correttamente definiti nel file XML, che l'applicazione doveva poter importare ed esportare. Questo ha richiesto l'implementazione di logiche per leggere e scrivere gli elementi, le loro dimensioni e le relazioni in modo conforme allo standard BPMN.

5.4.4 Sviluppo del pannello delle proprietà

Con l'implementazione dei nuovi elementi, si è iniziato a lavorare sul pannello delle proprietà per migliorare la gestione delle attività. Il pannello è stato progettato in modo che fosse il più vicino possibile ai mock-up creati, ma è stato necessario definire ulteriori funzionalità per

l'aggiunta, rimozione e modifica degli oggetti. Ogni modifica apportata tramite il pannello delle proprietà si rifletteva immediatamente sia nell'interfaccia utente sia nel file XML sottostante.

5.4.5 Integrazione del validatore

Con tutte le funzionalità di base integrate, si è incentrata l'attenzione sull'implementazione di un sistema di validazione per aiutare gli utenti a progettare processi conformi. Utilizzando la libreria bpmnlint, che evidenzia graficamente errori e avvertimenti nel diagramma (un cerchio rosso con una 'x' bianca per gli errori e un cerchio giallo con un punto esclamativo bianco per gli avvertimenti).

Questa funzionalità è stata progettata in modo che l'utente potesse nascondere o mostrare gli avvisi attraverso un bottone situato in alto a sinistra dell'interfaccia.

Inoltre, sono state aggiunte due nuove regole specifiche per il progetto di tesi: un avvertimento che compare nell'esecutore se non gli è ancora stato associato alcun prodotto e un avvertimento quando non è associato nessun esecutore all'elemento *Batch*.

5.4.6 Gestione delle attività batch

L'elemento *Batch* è stato integrato solo successivamente, essendo una specializzazione dell'elemento *Activity*, ovvero quelle attività che richiedono un numero minimo di elementi per iniziare a lavorare. È stato sostituito l'elemento attività di base con un nuovo tipo che avesse le stesse proprietà, ma una visualizzazione diversa: un rettangolo con gli angoli arrotondati che in alto a sinistra contiene un ingranaggio.

In aggiunta, è stata introdotta una distinzione visiva per le attività batch: se tutti i prodotti associati lavorano in batch, l'ingranaggio è bianco, altrimenti è arancione. In questo modo aiuta l'utente a identificare rapidamente lo stato delle attività *batch*.

5.4.7 Personalizzazione della palette

Durante il processo di miglioramento dell'applicazione, si è resa necessaria una modifica della palette degli strumenti. L'obiettivo principale era quello di semplificare la selezione degli elementi e rendere l'interfaccia utente più intuitiva.

Gli elementi principali inclusi nella palette di bpmn.js comprendono:

- Eventi: rappresentano qualcosa che accade durante un processo. Possono essere di vari tipi, tra cui eventi iniziali, intermedi e finali.
- Attività: sono compiti o azioni che devono essere eseguiti all'interno del processo. Possono essere semplici (attività singole) o composte (sottoprocessi).

- Gateway: definiscono punti di decisione nel flusso del processo, permettendo di deviare il percorso basato su condizioni specifiche.
- Connettori: i connettori (flussi di sequenza, flussi di messaggio, ecc.) collegano i vari elementi del diagramma, rappresentando il percorso e l'ordine delle attività.
- Artefatti: includono annotazioni e gruppi che aiutano a chiarire il diagramma senza influenzare direttamente il flusso del processo.

In particolare, sono stati apportati i seguenti cambiamenti:

- Aggiunta dell'elemento esecutore: rappresenta la risorsa responsabile dell'esecuzione dei processi. La sua inclusione consente di modellare in modo più dettagliato chi esegue le attività all'interno del processo.
- Rimozione di elementi superflui: sono stati eliminati elementi come *Subprocess-expanded*, *Data-object*, *Group*, *Data-store* e *Participant-expanded*.

La personalizzazione della palette è stata integrata con l'obiettivo di migliorare l'esperienza utente, rendendo più intuitiva la creazione dei diagrammi di flusso e agevolando la rappresentazione precisa dei processi industriali.

5.4.8 Implementazione dei controlli dell'interfaccia

L'ultima fase dell'implementazione si è concentrata sull'integrazione di alcune funzionalità già definite da bpmn.js. L'aggiunta ha introdotto cinque pulsanti di controllo posizionati in basso a destra dell'interfaccia consentendo di eseguire le seguenti azioni:

- Caricare un diagramma.
- Scaricare un diagramma.
- Centraggio del diagramma.
- Zoom in.
- Zoom out.

Questi controlli migliorano l'usabilità dell'applicazione, rendendo più semplice la navigazione e la gestione dei diagrammi BPMN.

CAPITOLO 6 – STRUTTURA DEL CODICE

Questo capitolo si propone di offrire una panoramica completa e dettagliata della struttura del codice, basandosi sul repository bpmn-js-example-react-properties-panel [15].

Viene illustrata la ricostruzione da zero di un pannello delle proprietà, sfruttando le potenzialità di React. Partendo da questa base, sono state implementate e modificate diverse funzionalità per rispondere alle esigenze specifiche del sistema di modellazione dei processi produttivi in ambito manifatturiero.

6.1 Strumentazione utilizzata

6.1.1 React

È una libreria JavaScript open-source per la costruzione di interfacce utente, principalmente per applicazioni web a pagina singola. È stata sviluppata da Facebook e viene mantenuta da una comunità di sviluppatori indipendenti. La filosofia alla base di React è quella di suddividere l'interfaccia utente in componenti riutilizzabili, ciascuno dei quali gestisce il proprio stato. Questo approccio consente di creare applicazioni complesse e dinamiche in modo più organizzato e mantenibile.

React utilizza un concetto chiamato *Virtual DOM* per ottimizzare le operazioni di aggiornamento del browser. Invece di aggiornare direttamente il *DOM* del browser ogni volta che cambia lo stato dell'applicazione, React ne aggiorna uno virtuale mantenuto in memoria. Solo quando il *Virtual DOM* ha calcolato le differenze rispetto a quello reale, React effettua le modifiche necessarie, migliorando così le prestazioni.

6.1.2 Npm

Per la gestione delle dipendenze e la configurazione del progetto, è stato utilizzato *npm* (Node Package Manager). È un gestore di pacchetti per l'ambiente di runtime JavaScript *Node.js*. Permette di installare, aggiornare e rimuovere pacchetti di codice in modo semplice ed efficiente, facilitando la gestione delle dipendenze del progetto.

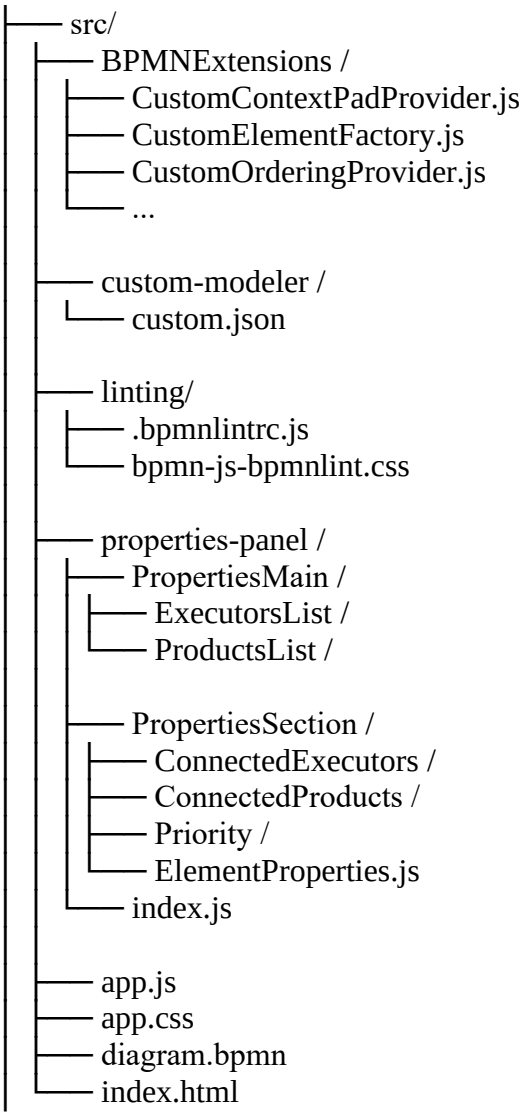
È possibile specificare tutte le librerie e i tool necessari per il progetto in un file chiamato *package.json*. Questo file include informazioni sui pacchetti utilizzati, le versioni richieste e script personalizzati per automatizzare vari compiti di sviluppo. Ad esempio, attraverso *npm* è possibile installare React e altre librerie necessarie con semplici comandi da terminale come *npm install react* o *npm install*, che legge il *package.json* e installa tutte le dipendenze elencate.

Per avviare l'applicazione, viene utilizzato il comando *npm start*. Esso è definito all'interno del file *package.json* del progetto e permette di eseguire un server di sviluppo locale che ospita l'applicazione React. Quando viene eseguito il comando, *npm* legge le configurazioni specificate nel *package.json* e avvia il server di sviluppo configurato per il progetto. Questo server, solitamente basato su Webpack o su un altro bundler JavaScript, consente agli sviluppatori di vedere in tempo reale le modifiche apportate al codice, grazie al live reloading.

In altre parole, ogni modifica al codice sorgente provoca un aggiornamento automatico della pagina web, rendendo il processo di sviluppo più efficiente e interattivo.

Grazie all'uso di questo strumento, è possibile mantenere un ambiente di sviluppo coerente e riproducibile, garantendo che tutti i membri del team utilizzino le stesse versioni delle librerie e degli strumenti necessari per il progetto.

6.2 Struttura del progetto



6.3 BPMNExtensions

La cartella BPMNExtensions contiene tutti i file personalizzati che estendono le funzionalità base di bpmn.js, come la creazione e gestione degli esecutori, definizione delle regole del diagramma, aggiunta di un validatore.

Verrà fornita una breve panoramica sull'utilizzo e la funzione all'interno del progetto. Inoltre, per alcuni passaggi più complessi o fondamentale importanza, saranno forniti ulteriori dettagli e spiegazioni per garantire una comprensione completa del loro ruolo e della loro rilevanza all'interno del sistema.

6.3.1 CustomContextPadProvider

Il ContextPadProvider è un menu che appare automaticamente quando si seleziona un elemento nel diagramma. Il provider fornisce un accesso rapido e intuito a una serie di azioni specifiche relative all'elemento selezionato.

Sono stati definiti due voci nel menù dell'esecutore:

- *Sequence-flow*: consente agli utenti di creare una connessione a partire dall'elemento esecutore.
- *Delete*: consente di eliminare l'elemento.

Fig. 31: Menu dell'elemento esecutore

6.3.2 CustomElementFactory

All'interno di questo componente sono stati definiti una serie di elementi aggiuntivi agli elementi base di bpmn.js. La *factory* è responsabile della creazione di nuovi elementi nel diagramma, sia per gli elementi standard di BPMN sia elementi personalizzati.

La funzione principale di questo file è *create*. Essa prende due argomenti in input:

- *elementType*: indica il tipo di elemento da creare (es. *shape, connection, label*).
- *attrs*: oggetto contenente gli attributi specifici dell'elemento da creare.

viene controllato il tipo di elemento passato in input (*elementType*):

- Se si tratta di un label, delega la creazione alla funzione base ereditata (*_baseCreate*).
- Per gli altri tipi di elemento (*shape* o *connection*), la funzione esegue i seguenti passaggi:
    - Se non si tratta di un elemento personalizzato, viene delegata la creazione alla funzione standard *createElement* fornita dal componente base.
    - Se è un componente personalizzato:
        - Crea l'oggetto *business object* (struttura dati che incapsula sia i dati rilevanti che le operazioni che possono essere eseguite su di essi) corrispondente utilizzando *bpmnFactory* se non è già presente negli attributi.

- Controllate le informazioni relative alla rappresentazione visuale (DI, *Diagram Interchange*) dell'elemento. Il DI descrive come l'elemento dovrebbe essere visualizzato nel diagramma, incluse informazioni come posizione, dimensioni, colori, ecc. Se queste informazioni non sono presenti, vengono generate automaticamente.
- Viene calcolata la dimensione predefinita dell'elemento tramite la funzione *_getCustomElementSize*.
- Gli attributi specifici dell'elemento vengono applicati, filtrando quelli non validi per il tipo di elemento in questione.
- Vengono infine uniti tutti gli attributi necessari (ID, *business object*, informazioni visuale) per creare l'elemento finale.

6.3.3 CustomOrderingProvider

CustomOrderingProvider permette di posizionare in modo specifico gli elementi nel diagramma. In questo caso, assicura che le connessioni personalizzate vengano sempre renderizzate sopra gli altri elementi, garantendo una migliore visualizzazione delle relazioni nel diagramma.

6.3.4 CustomPalette

Il file CustomPalette.js definisce una palette per il modeler BPMN. La palette è l'insieme di icone che compaiono a sinistra del diagramma e che permettono agli utenti di creare nuovi elementi.

La funzione *getPaletteEntries* è responsabile di definire le voci, ovvero icone e azioni, che appariranno nella palette personalizzata. In particolare, è stata definita una funzione *createTask* che si occupa di creare un nuovo oggetto di tipo *factory:Executor* e la relativa forma visuale associata utilizzando *elementFactory*. Inoltre, mediante la stessa funzione sono stati rimossi alcuni elementi predefiniti non necessari per il lavoro di tesi (*subprocess-expanded*, *data-object*, *group*, *data-store*, *participant-expanded*).

Infine, le voci della palette standard sono state unite a quelle personalizzate, al fine di ottenere una palette completa e personalizzata per l'applicazione.

6.3.5 CustomRenderer

Il renderer è responsabile della traduzione degli elementi del modello in rappresentazioni visive in formato SVG visualizzate nell'area di disegno.

Il costruttore della classe CustomRenderer riceve tramite iniezione diverse dipendenze:

- *config.bpmnRenderer*: configurazione del renderer BPMN standard.
- *eventBus*: bus degli eventi utilizzato per comunicare all'interno dell'applicazione.
- *bpmnRenderer*: renderer BPMN standard fornito da bpmn.js.
- *textRenderer*: testo utilizzato per le label degli elementi.
- *styles*: gestione degli stili degli elementi.
- *pathMap*: hashmap utilizzata per accedere a percorsi predefiniti di icone.

La funzione *canRender* stabilisce se un elemento grafico può essere gestito dal sistema. Controlla se appartiene a uno dei tipi personalizzati (*factory:Executor*, *factory:Connection* e *factory:Batch*) e se non ha una label associata.

La funzione *drawShape* è responsabile del disegno delle forme degli elementi. In particolare viene controllato il tipo dell'elemento (*element.type*) e richiama una funzione specifica per il rendering in base al tipo:

- *factory:Executor*: viene disegnato un esagono con la possibilità di visualizzare una label al centro.
- *factory:Batch*: viene utilizzato il renderer standard BPMN per disegnare un task. Successivamente viene analizzato il modello per verificare se tutti gli elementi collegati alla *task* appena disegnata soddisfano alcune condizioni. In base al risultato dell'analisi viene modificato il colore di riempimento del dell'icona a forma di ingranaggio posta nell'angolo in alto a sinistra, utile per evidenziare eventuali incongruenze.
- Altri tipi di elemento: viene utilizzato il renderer standard BPMN ereditato (*this.bpmnRenderer.drawShape*).

6.3.6 CustomRules

Questo componente consente di estende le regole di comportamento all'interno del modeler BPMN. Il suo scopo è quello di limitare le azioni consentite sugli elementi non standard in modo da garantire la coerenza del diagramma di processo.

Vengono definite due funzioni:

- *canCreate*: verifica se un elemento personalizzato può essere creato all'interno dell'area di disegno selezionata. Sono consentite le creazioni solo su processi.
- *canConnect*: viene dichiarata una regola che consente connessioni tra elementi *Executor* ed elementi *Batch* o *Activity*. In questo modo l'esecutore non può collegarsi a nessun altro tipo.

Le due funzioni appena citate vengono richiamate nel momento in cui si verificano alcuni eventi nel modellatore:

- *elements.move*: controlla lo spostamento di più elementi.
- *shape.create*: controlla la creazione di un singolo elemento.
- *shape.resize*: disabilita il ridimensionamento degli elementi custom.
- *connection.create*: controlla la creazione di una connessione.
- *connection.reconnectStart* e *connection.reconnectEnd*: controllano il ricollegamento dell'inizio o della fine di una connessione esistente.

6.3.6 CustomUpdater

La classe CustomUpdater è un intercettatore di eventi utilizzato per sincronizzare automaticamente il *business object* degli elementi con le modifiche visive avvenute sul diagramma BPMN. Questo garantisce la coerenza tra la rappresentazione grafica e i dati sottostanti del processo modellato.

Vengono definite due principali funzioni per gestire l'aggiornamento del *business object* degli elementi personalizzati, in base all'evento intercettato:

- *updateCustomElement*: gestisce la creazione, lo spostamento e l'eliminazione di elementi. Aggiorna la posizione dell'elemento e si occupa di aggiungere o rimuovere l'elemento dall'elenco degli elementi, mantenuto internamente dall'applicazione.
- *updateCustomConnection*: gestisce la creazione, il ricollegamento, l'aggiornamento della posizione e l'eliminazione delle connessioni.

6.3.7 ReplaceConnectionBehavior

Anch'essa è un intercettatore di eventi, ma gestisce il ricollegamento delle connessioni nel diagramma BPMN. Questo comportamento viene attivato quando una connessione esistente viene modificata, riposizionata o collegata a un altro elemento, e non quando viene creata una nuova connessione. Il suo scopo è quello di garantire la coerenza del flusso di processo quando gli elementi vengono riposizionati o collegati ad altri elementi.

La principale funzione è *replaceReconnectedConnection* che viene eseguita quando una connessione viene ricollegata. Controlla se l'elemento sorgente o target è un elemento di tipo Executor o meno, e in base al tipo recupera le regole di connessione consentite, inclusa la visualizzazione corretta della connessione (freccia o linea tratteggia).

6.3.8 ReplaceMenuProvider

Il componente CustomMenuProvider è responsabile nel menu contestuale (*ContextPad*) di eseguire un'operazione di sostituzione degli elementi nel diagramma BPMN, fornendo all'utente la possibilità di trasformare un elemento in un altro elemento compatibile con il flusso di processo.

Se l'elemento selezionato è di tipo *Batch*, verranno visualizzate le opzioni per sostituire il componente con l'elemento *Activity* e viceversa.

6.3.9 ReplaceOptions

Questo file esporta due array, *ACTIVITY_INTO_BATCH* e *BATCH_INTO_ACTIVITY*, che contengono definizioni di sostituzione, utile per il file descritto precedentemente nel paragrafo 6.2.8.

6.3.10 TypeUtils

La classe TypeUtils fornisce una funzione chiamata *isDifferentType* utilizzata per verificare la compatibilità di tipo durante la sostituzione di elementi nel diagramma BPMN. Questa funzione è fondamentale per garantire che la sostituzione di un elemento con un altro mantenga la validità e la coerenza del flusso di processo.

Anche in questo caso il file è stato introdotto per utilizzare il file descritto nel paragrafo 6.3.8.

6.3.11 index.js

Il file index.js funge da punto di ingresso principale per le estensioni personalizzate del diagramma BPMN finora descritte. Esso non implementa alcuna logica specifica, ma si occupa di registrare e configurare i vari componenti personalizzati che modificano il comportamento del modeler. Attraverso l'importazione dei vari moduli, consente di modificare e arricchire le funzionalità in base alle necessità descritte.

6.4 custom-modeler/custom.json

Consente di personalizzare il diagramma BPMN con elementi e proprietà aggiuntive che soddisfano specifiche esigenze di modellazione. Grazie a queste estensioni, è possibile rappresentare flussi di processo complessi con un livello di dettaglio maggiore rispetto allo standard BPMN 2.0.

Informazioni generali:

- name: nome dell'estensione (*Factory*).
- uri: namespace dell'estensione.
- prefix: prefisso utilizzato per gli elementi personalizzati (*factory*).
- xml.tagAlias: impostazione per la conversione dei tag nel file XML degli elementi personalizzati in minuscolo.

Sono stati dichiari quattro nuovi elementi:

- *Executor*: estende l'elemento *bpmn:Task*, ma viene aggiunge la proprietà 'product', consentendogli avere molteplici valori di tipo *Product*.
- *Connection*: estende l'elemento *bpmn:SequenceFlow* senza proprietà aggiuntive.
- *Product*: tipo di elemento personalizzato con proprietà che descrivono un prodotto (identificativo, nome, tempo, unità di tempo, batch e identificativo dell'attività associata).
- *Batch*: estensione dell'elemento *bpmn:Task* senza proprietà aggiuntive.

6.5 linting/.bpmnlintrc

Il modulo bpmnlinting definisce un insieme di regole personalizzate per la verifica della correttezza e della coerenza di un modello BPMN, identificando potenziali errori o problemi che potrebbero ostacolare il corretto funzionamento del flusso di processo.

La libreria include diverse regole predefinite che coprono una vasta gamma di aspetti:

- *conditionalFlows*: verifica che i flussi di sequenza in uscita da un gateway o un'attività di fork condizionale abbiano una condizione allegata o siano flussi predefiniti.
- *endEventRequired* verifica che ogni processo o sottoprocesso abbia un evento di fine.
- *eventSubProcessTypedStartEvent* verifica che gli eventi di avvio all'interno di un sottoprocesso di evento siano tipizzati.
- *fakeJoin* verifica che non venga modellato un join falso tentando di dare a un'attività o a un evento semantiche di join.
- *labelRequired* verifica che gli elementi abbiano un'etichetta.
- *noBpmndi* verifica che non manchi alcuna informazione DI per gli elementi che lo richiedono.
- *noComplexGateway* vieta l'utilizzo di gateway complessi.
- *noDisconnected* verifica che non vi siano elementi di flusso scollegati.
- *noDuplicateSequenceFlows* verifica che non vi siano flussi di sequenza duplicati.
- *noGatewayJoinFork* verifica che un gateway non esegua sia il fork che il join.
- *noImplicitSplit* verifica che non venga modellata una divisione implicita a partire da un'attività.
- *noInclusiveGateway* vieta l'utilizzo di gateway inclusivi.
- *singleBlankStartEvent* verifica che non vi siano più di un evento di avvio vuoto per ambito.
- *singleEventDefinition* verifica che un evento contenga al massimo una definizione di evento.
- *startEventRequired* verifica che vi sia un evento di avvio per ogni ambito.
- *subProcessBlankStartEvent* verifica che gli eventi di avvio all'interno di un sottoprocesso normale siano vuoti (non abbiano una definizione di evento).
- *superfluousGateway* verifica che un gateway non abbia solo una sorgente e un destinatario.

Oltre alle regole predefinite, sono state implementate due regole specifiche per il progetto:

- *noProductsDefined*: verifica che per ogni esecutore sia stato dichiarato almeno un prodotto.
- *noExecutorDefined*: verifica che un elemento di tipo Batch sia connesso almeno ad un elemento di tipo Executor.

Nel lavoro di tesi, alcune regole predefinite (*conditionalFlows*, *noBpmndi*, *noDuplicateSequenceFlows*, *noImplicitSplit*) sono state commentate perché considerate non utili ai fini del progetto.

6.6 properties-panel

6.6.1 PropertiesMain

Questo file è responsabile della gestione del pannello delle proprietà. Agisce come punto di ingresso per l'utilizzo del pannello e renderizza due diverse interfacce utente a seconda della selezione degli elementi. Se non è selezionato alcun elemento, il pannello mostra due sezioni: una sezione superiore con una lista di esecutori e una sezione inferiore con una lista di prodotti. Se invece è selezionato un elemento, viene visualizzato un pannello delle proprietà per gestire l'elemento selezionato. (descritto nel paragrafo 6.5.4).

I metodi principali all'interno di questo file sono:

- *onAddProduct*: quando viene aggiunto un prodotto alla lista, lo inserisce in una variabile globale e ne consente l'associazione con gli esecutori.
- *handleSelectionChange*: viene chiamato quando un elemento è selezionato nel diagramma, mostrando così un pannello delle proprietà con una schermata differente.

6.6.2 ExecutorsList

Questo componente elenca tutti gli esecutori presenti nel diagramma BPMN. Utilizza gli hook *useState* e *useEffect* per gestire lo stato degli esecutori e monitorare i cambiamenti nel modello BPMN. Quando il modello subisce una modifica, l'evento *elements.changed* viene intercettato e la lista degli esecutori aggiornata di conseguenza.

La funzione *updateExecutors* filtra gli elementi del registro interno di bpmn.js per individuare quelli di tipo *Executor*, estraendone ID e nomi e aggiornando la lista con queste informazioni. Il rendering restituito presenta una tabella con i nomi e gli ID degli esecutori, offrendo una visualizzazione chiara e strutturata degli elementi rilevanti nel contesto del modellatore.

6.6.3 ProductsList

ProductsList consente di aggiuntare e alla visualizzazione la lista dei prodotti associati agli esecutori nel modellatore BPMN. Utilizza l'hook *useState* per mantenere e aggiornare lo stato dei prodotti.

L'interfaccia utente consente di inserire nuovi prodotti tramite un campo di testo e un pulsante di aggiunta. Quando un nuovo prodotto viene inserito, *handleAddProduct* crea un nuovo oggetto prodotto e invoca *onAddProduct*, che aggiorna lo stato dei prodotti nel componente genitore, tenendo traccia della lista dei prodotti in una lista globale. Essa viene visualizzata in una tabella, facilitando la gestione e l'espansione dell'elenco dei prodotti nel contesto dell'applicazione.

Un'altra funzione chiamata *generateUniqueId* genera ID unici per ogni nuovo prodotto aggiunto, garantendo l'unicità degli elementi nel sistema.

6.6.4 ElementProperties

Il componente ElementProperties viene richiamato quando un utente seleziona un elemento all'interno del diagramma ed è responsabile della gestione delle proprietà dell'elemento selezionato e presenta due sezioni principali.

La sezione superiore mostra le informazioni generali dell'elemento, quali il tipo, l'ID e il nome associato. La sezione inferiore, invece, presenta alcune sottosezioni che variano in base alla tipologia dell'elemento selezionato:

- Se l'elemento selezionato è un esecutore, verrà visualizzata la sezione *ConnectedProducts* (descritto nel paragrafo 6.5.6).

- Se l'elemento selezionato è un *Activity* o un *Batch*, verranno mostrate le sezioni *ConnectedExecutors* (descritto nel paragrafo 6.5.5) e *Priority* (descritto nel paragrafo 6.5.7).

Le funzioni principali sono:

- *getElementType*: restituisce il tipo dell'elemento selezionato, dopo aver rimosso i prefissi *bpmn* e *factory*, altrimenti restituirà *Process* come tipo predefinito.
- *handleNameChange*: gestisce la modifica del nome dell'elemento selezionato. Quando il nome viene modificato, la funzione aggiorna lo stato del componente e il modello BPMN associato all'elemento.

6.6.5 ConnectedExecutors

Gestisce la visualizzazione e la manipolazione degli esecutori connessi a un determinato elemento nel diagramma BPMN. Utilizza React e vari hook per gestire lo stato e le interazioni dell'utente. Questo componente è particolarmente importante per mantenere aggiornati gli esecutori associati a un'attività o a un batch e per consentire la gestione dei prodotti collegati agli esecutori.

Stati e Riferimenti:

- *searchBarRef*: riferimento per la barra di ricerca dei prodotti.
- *executorSearchResults*, *executorSearchInput*: gestiscono i risultati e l'input della ricerca degli esecutori.
- *selectedExecutors*, *executors*: gestiscono gli esecutori selezionati e l'elenco completo degli esecutori.
- *showInputExecutor*, *showInputProduct*: stato per mostrare o nascondere l'input di ricerca per esecutori e prodotti.
- *productSearchInput*, *selectedProducts*, *productSearchResults*: gestiscono l'input, i prodotti selezionati e i risultati della ricerca dei prodotti.
- *executorDropdownOpen*, *productDropdownOpen*: controllano l'apertura dei dropdown menu degli esecutori e prodotti.
- *isPropertiesExpanded*, *executorExpanded*, *productExpandedExec*: gestiscono l'espansione delle sezioni delle proprietà e dei prodotti.
- *isExecutorConnectedToBatch*: indica se l'esecutore è collegato a un batch.

Funzioni utilizzate:

- *getConnectedExecutors*: estrae gli esecutori collegati all'elemento corrente attraverso le connessioni nel diagramma.
- *handleSearchExecutor*: filtra gli esecutori in base all'input di ricerca e aggiorna i risultati.
- *handleSelectExecutor*: aggiunge un esecutore selezionato alla lista degli esecutori collegati e aggiorna lo stato.
- *handleSearchProducts*: filtra i prodotti in base all'input di ricerca e aggiorna i risultati per un esecutore specifico.
- *handleSelectProduct*: aggiunge un prodotto selezionato all'esecutore specificato e aggiorna lo stato e il modello BPMN.

- *handleDeleteProduct*: rimuove un prodotto selezionato dall'esecutore specificato e aggiorna il modello BPMN.
- *handleDeleteExecutor*: rimuove un esecutore dalla lista degli esecutori collegati e aggiorna lo stato.
- *handleExecutorClick*: gestisce l'espansione delle proprietà degli esecutori e aggiorna la lista di prodotti associati da visualizzare
- *handleProductExpansionExec*: gestisce l'espansione delle proprietà dei prodotti per un esecutore specifico.
- *handleToggleSearchInput*: mostra o nasconde l'input di ricerca per i prodotti di un esecutore.
- *handleTimeUnitChange*, *handleTimeChangeExe*: gestiscono la modifica dell'unità di tempo e del tempo per i prodotti.
- *handleAttachExecutor*: collega un esecutore all'elemento corrente nel diagramma.
- *handleDetachExecutor*: scollega un esecutore dall'elemento corrente e rimuove la connessione.
- *handleBatchChange*: gestisce la modifica del valore batch.

6.6.6 ConnectedProducts

Gestisce la visualizzazione e la modifica dei prodotti associati a un esecutore selezionato nel diagramma BPMN. Questo componente viene richiamato quando un utente seleziona un esecutore e mostra una lista di prodotti collegati, consentendo la modifica di proprietà specifiche: tempo e valore del batch.

Funzioni utilizzate:

- *getConnectedExecutors*: utilizza il registro degli elementi di bpmn.js per ottenere gli esecutori collegati all'elemento corrente.
- *handleProductExpansion*: gestisce l'espansione o la contrazione delle sezioni dei prodotti per mostrare o nascondere i dettagli di un prodotto specifico.
- *getActivityName*: recupera il nome dell'attività associata a un prodotto specifico dal registro degli elementi di bpmn.js.
- *handleTimeChange*: aggiorna il tempo di un prodotto selezionato e riflette questa modifica nel modello BPMN.
- *handleTimeUnitChange*: modifica l'unità di tempo di un prodotto selezionato e aggiorna il modello BPMN di conseguenza.
- *handleBatchChange*: gestisce le modifiche del valore batch e aggiorna il modello BPMN.

6.6.7 Priority

In questo file è definita solo una funzione che consente di aggiornare il valore della priorità, tramite un campo input. La modifica viene salvata nelle proprietà dell'elemento selezionato che può essere di tipo *Batch* o *Activity*.

6.6.8 index.js

Funge da punto di ingresso per il rendering del pannello delle proprietà nell'applicazione BPMN. Utilizzando React e *ReactDOM*, il componente crea e mostra il pannello delle proprietà.

Il costruttore della classe accetta un oggetto *options* che contiene due proprietà:

- *modeler*: un'istanza del modellatore fornita da bpmn.js. Questa variabile è fondamentale perché permette al pannello delle proprietà di interagire con sistema, aggiornando e recuperando informazioni sugli elementi del diagramma.
- *container*: un riferimento al contenitore *DOM* in cui il pannello delle proprietà verrà visualizzato.

La dichiarazione di questo codice è essenziale per integrare l'interfaccia del pannello delle proprietà con l'applicazione, garantendo che gli utenti possano interagire con le proprietà degli elementi direttamente all'interno dell'interfaccia utente.

6.7 src

6.7.1 app.js

Il file app.js è il punto di ingresso principale per l'applicazione. Questo file si occupa dell'inizializzazione del modellatore BPMN e del pannello delle proprietà, oltre a gestire diverse funzionalità dell'interfaccia utente, come il caricamento, il salvataggio e lo zoom del diagramma.

Viene creata un'istanza del modellatore (*modeler*) utilizzando la libreria bpmn.js, configurato con i seguenti valori:

- *container*: il contenitore *DOM* in cui il diagramma verrà mostrato.
- *moddleExtensions*: estensioni personalizzate del modellatore, incluse dal file *custom.json*.
- *linting*: configurazione per la validazione del diagramma BPMN usando *bpmnlint*.
- *additionalModules*: Moduli aggiuntivi, come *bpmnExtension* e *lintModule*.

È inizializzato il pannello delle proprietà (*propertiesPanel*) e viene passato l'oggetto *modeler* per consentire l'interazione diretta con il modellatore.

Inoltre, vengono gestite le funzionalità dei controlli nell'interfaccia utente, ovvero lo zoom in, zoom out, center, upload e download.

Infine, viene gestita la visibilità degli esecutori e degli avvisi nell'interfaccia attraverso due switch button.

6.7.2 diagram.bpmn

Il file diagram.bpmn serve a inizializzare il modellatore BPMN. Al momento del caricamento dell'applicazione, il diagramma presente nel file viene automaticamente caricato nel modellatore, offrendo così all'utente un diagramma di partenza o un esempio da seguire.

6.7.3 index.html

Questo file HTML fornisce la struttura di base per un'applicazione Web che consente agli utenti di visualizzare, caricare, scaricare e interagire con BPMN.

CAPITOLO 7 – CONCLUSIONI

Lo sviluppo di questo progetto di tesi è stata un'esperienza formativa e professionale di grande valore. Sin dalle prime fasi, ho trovato estremamente stimolante l'opportunità di approfondire le mie conoscenze nell'ambito dei modelli BPMN e delle tecnologie web per la loro visualizzazione e interazione.

L'utilizzo di BPMN si è rivelato una scelta vincente, fornendo una base solida e flessibile per la realizzazione dell'applicazione. Ho apprezzato particolarmente la facilità d'uso della libreria, la sua ampia documentazione e la ricca community di sviluppatori.

La sfida più impegnativa è stata senza dubbio la progettazione e l'implementazione delle funzionalità avanzate, come il sistema di validazione dei modelli BPMN e il pannello di gestione delle proprietà. In questa fase, ho potuto mettere a frutto le mie conoscenze di programmazione e di algoritmi, imparando a gestire dati complessi e a sviluppare interfacce utente intuitive.

Nel complesso, il lavoro svolto ha ampiamente soddisfatto le mie aspettative. Ho raggiunto gli obiettivi prefissati, realizzando un'applicazione web completa e funzionale per la gestione di modelli di processi produttivi.

Inoltre, l'esperienza mi ha permesso di acquisire competenze e metodologie di sviluppo che saranno preziose per il mio futuro professionale.

Evoluzioni future:

Il progetto ha un buon potenziale di evoluzione e può essere ampliato in diverse direzioni:

- Integrazione del simulatore nell'interfaccia web: potrebbe essere integrato il simulatore di processi direttamente all'interno dell'interfaccia web per consentire agli utenti di eseguire simulazioni e analizzare i risultati in tempo reale, migliorando l'interattività e l'efficacia dello strumento. Inoltre, sarebbe utile implementare una funzionalità di cronologia per tenere traccia delle simulazioni passate, permettendo agli utenti di rivedere e confrontare i risultati precedenti, facilitando l'analisi delle evoluzioni dei processi nel tempo.
- Realizzazione di funzionalità di collaborazione: potrebbero essere implementate funzionalità di collaborazione per permettere a più utenti di lavorare contemporaneamente sullo stesso modello BPMN.
- Estensione del progetto: ampliare le funzionalità di gestione dei processi. Ad esempio, integrare un inventario associato ai prodotti per tenere traccia delle disponibilità, delle scorte invendute e delle provviste utilizzabili all’interno di un processo. Inoltre, sviluppare un processo di modellazione più esteso che non comprenda solamente il processo di produzione, ma anche la gestione degli ordini, dei processi amministrativi e altre attività correlate, creando una soluzione completa per la gestione aziendale.

In definitiva, questo progetto rappresenta un punto di partenza solido per lo sviluppo di soluzioni software avanzate per la gestione di processi aziendali basati su modelli BPMN.

Considerazioni personali:

A livello personale, questa esperienza mi ha permesso di crescere come professionista e di sviluppare una forte passione per l'ambito dei modelli BPMN e delle tecnologie web. Ho imparato l'importanza di un approccio metodologico al lavoro, della collaborazione con altri professionisti e della continua ricerca di soluzioni innovative.

Sono fiducioso che le competenze acquisite durante lo sviluppo di questo progetto mi saranno di grande beneficio nel mio futuro percorso professionale.

RIFERIMENTI

1. Chryssolouris, G.: Manufacturing systems: theory and practice. Springer Science & Business Media (2013)
2. Law, A.M., McComas, M.G.: Simulation of manufacturing systems. In: Proceedings of the 19th conference on Winter simulation, pp. 631–643 (1987)
3. Mourtzis, D., Doukas, M., Bernidaki, D.: Simulation in manufacturing: Review and challenges. Procedia Cirp 25, 213–229 (2014)
4. Williams, E.J.: Simulation attacks manufacturing challenges. In: Buckley, S.J., Miller, J.A. (eds.) Proceedings of the 2014 Winter Simulation Conference, Savannah, GA, USA, December 7-10, 2014, pp. 81–89. IEEE/ACM (2014). https://doi.org/10.1109/WSC.2014.7019879
5. Herzog, A.: Simulation mit demWarteschlangensimulator. Angenommen zur Veröffentlichung (2021)
6. OMG, Business Process Model and Notation (BPMN), (2011).
7. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow Resource Patterns: Identification, Representation and Tool Support. In: Pastor, O., e Cunha, J.F. (eds.) Advanced Information Systems Engineering, 17th International Conference, CAiSE 2005, Porto, Portugal, June 13-17, 2005, Proceedings. LNCS, vol. 3520, pp. 216–232. Springer, Heidelberg (2005). https://doi.org/10.1007/11431855_16
8. Jablonski, S., Bussler, C.: Workflow management - modeling concepts, architecture and implementation. International Thomson (1996)
9. Camunda Modeler. https://camunda.com/
10. Bizagi Modeler. https://www.bizagi.com/en
11. JBPM Modeler. https://www.jbpm.org/
12. Bpmn.js – Libreria Javascript. https://bpmn.io/toolkit/bpmn-js/
13. Industria 4.0: le 9 tecnologie abilitanti - Automazione industriale. https://www.elteco.it/9-tecnologie-abilitanti/
14. Repository GitHub con properties panel integrato. https://github.com/bpmn-io/bpmn-js-properties-panel
15. Repository GitHub con properties panel custom in React. https://github.com/bpmn-io/bpmn-js-example-react-properties-panel