POLITECNICO DI MILANO

Corso di laurea in Ingegneria dell'Automazione

VALIDAZIONE SPERIMENTALE DI TECNICHE DI SIMULAZIONE DI UN SISTEMA DI PRODUZIONE MISTO REALE E VIRTUALE

Relatore:

Chiar.mo Prof. Luca Ferrarini

Correlatore:

Ing. Alessio Dedè

Luca Smeraldi

Matricola 720946

Anno accademico 2009/2010

RINGRAZIAMENTI

Primo doveroso ringraziamento ai miei genitori, Elvezio e Paola. Oltre a permettermi di intraprendere un percorso formativo importante come la carriera universitaria, mi hanno incoraggiato ed hanno atteso con pazienza il raggiungimento dell'obiettivo finale.

Ringrazio il Prof. Ferrarini, già mio relatore per la laurea triennale, che mi ha nuovamente assegnato un lavoro stimolante da svolgere. Un grazie particolare all' Ing. Alessio Dedè che mi ha seguito ed aiutato in maniera concreta nella realizzazione dell'elaborato. Insieme abbiamo condiviso interminabili ore di lavoro in dipartimento e biblioteca.

Non dimentico gli amici conosciuti in università, Carlo, Paolo, Federico, Simone: ho perso il conto delle ore passate insieme a studiare e delle serate trascorse a divertirci. Termina la mia carriera universitaria ma la nostra amicizia non finisce certo qui.

Infine, non certo in ordine di importanza, ringrazio Valentina per il sostegno in questi ultimi anni di studio. Non c'è persona migliore con la quale condividere questo momento.

SOMMARIO

Negli ultimi anni, la forte concorrenza introdotta dalla globalizzazione del mercato ha costretto le aziende ad elevare la loro velocità di risposta alle rapide modifiche richieste dai prodotti. Contemporaneamente sprechi di denaro e perdite di tempo non sono assolutamente accettabili e tutti i tipi di costi devono essere ridotti al minimo. In questo scenario è evidente che ogni tipo di investimento da parte dell'azienda diventa rischioso e occorre valutarlo con attenzione. È quindi di fondamentale importanza disporre di strumenti che permettano lo studio degli impatti che i possibili investimenti possono avere sui costi finali, ancor prima che questi vengano effettivamente intrapresi, al fine di ottenere una valutazione preventiva di ogni possibile strategia perseguibile.

Scopo di questo lavoro è dimostrare come gli ambienti di simulazione grafica 3D possono aiutare l'azienda nella fase di test di nuove logiche di controllo, senza pericolo di arrecare danni a impianti o persone e, grazie alla modularità di queste risorse grafiche, come sia possibile implementare architetture di Mixed Reality nell'ambito dell'automazione con lo scopo di far interagire parti reali e parti virtuali.

La prova finale di Mixed Reality ha permesso di dimostrare la validità delle idee iniziali e ha introdotto altre possibili applicazioni per progetti futuri.

INDICE

1

INTRODUZIONE

1.1 Contesto

Con il termine Mixed Reality ci si riferisce alla fusione di mondi reali e virtuali con lo scopo di ottenere nuovi ambienti in cui gli oggetti fisici (reali) e gli oggetti digitali (virtuali) coesistono e interagiscono in tempo reale. Nel corso degli anni la Mixed Reality è stata oggetto di numerose ricerche ed è stata applicata in contesti molto diversi fra loro: la didattica, l'imprenditoria, l'industria del cinema e molti altri.

In questo lavoro l'attenzione sarà ovviamente focalizzata nell'ambito dell'automazione industriale. La realtà aziendale moderna prevede spesso di dover modificare impianti e scenari produttivi al fine di aumentarne e migliorarne la produttività o fornire nuove lavorazioni. Acquistare un nuovo Macchinario è sempre un investimento finanziario ad alto rischio se non si stima in anticipo il beneficio che si potrebbe ottenere. Far dunque interagire un modello del Macchinario che si vuole acquistare (parte virtuale), con l'impianto fisico esistente (parte reale) permetterebbe di simulare il nuovo scenario ancor prima di avere l'impianto fisico finale e stimare se l'investimento è più o meno opportuno.

Un'altra applicazione potrebbe essere la ricerca della migliore legge di controllo e di schedulazione prima ancora che la nuova porzione d'impianto sia posizionata, permettendo una minore spesa di riconfigurazione per l'upgrade. Altri interessanti utilizzi potrebbero essere il Monitoring 3D per rappresentare a distanza lo stato di

evoluzione del sistema o l'implementazione di funzioni avanzate di diagnostica dei guasti.

Scopo di questo lavoro è mostrare la metodologia applicata e i risultati ottenuti tramite la simulazione Mixed Reality nel contesto del progetto europeo MEDEIA (Model-Driven Embedded Systems Design Environment for the Industrial Automation Sector).

L'approccio basato su modelli è il cuore di MEDEIA, il cui obiettivo principale è quello di definire un framework e una metodologia integrati per il progetto di sistemi di automazione che prevedono l'integrazione di diversi dispositivi eterogenei pur mantenendo un elevato grado di flessibilità del sistema controllato finale. In particolare, la definizione del sistema come aggregato di tanti sotto-sistemi permette la creazione di librerie di componenti riusabili per la progettazione di impianti.

1.2 Obiettivi dell'elaborato

L'obiettivo principale dell'elaborato è stato la realizzazione dell'architettura necessaria a rappresentare un progetto di Mixed Reality. Tale architettura deve garantire un corretto flusso di informazioni dei numerosi segnali presenti e la coerenza tra le varie parti simulate e reali. Pertinenti al progetto principale, è stata implementata la funzione di monitoraggio della cella di lavorazione, sono state implementate e verificate tutte le logiche di controllo dell'impianto e si sono effettuate prove di diagnostica. Infine è stata svolta la revisione completa delle schede tecniche dell'impianto (diagrammi SFC e descrizione delle funzioni di controllo in linguaggio C.

1.3 Organizzazione dell'elaborato

Il primo capitolo è una breve introduzione al contesto e agli obiettivi dell'elaborato. I successivi due capitoli sono a carattere teorico. In dettaglio, nel secondo si descrivono gli aspetti fondamentali della Mixed Reality, se ne presenta la definizione e il significato che assume nell'ambito dell'automazione. Sempre in questo capitolo si affrontano argomenti strettamente collegati alla Mixed Reality, la diagnostica, il Monitoring 3D, il simulatore SIMBA.

Nel terzo capitolo si espone l'approccio ai sistemi di controllo basati sui modelli riutilizzabili e si descrivono le idee fondamentali del progetto Media.

Il quarto capitolo è dedicato ad una precisa e dettagliata descrizione dei componenti e delle funzioni della Macchina SPI. Si mostrerà inoltre il lavoro di revisione delle schede tecniche.

Il quinto capitolo contiene la parte sperimentale dell'elaborato, viene descritta come è stata organizzata la prova pratica finale che prevede l'interazione fra componenti reali (Pallet Changer) e componenti virtuali (Macchina SPI , robot, stazione di carico/scarico pezzi). Parte del capitolo è inoltre dedicato alla fase di generazione e test delle logiche di controllo.

Chiude l'elaborato il sesto capitolo dove si discutono i risultati ottenuti e le future applicazioni.

2

MIXED REALITY

Il capitolo presenta una visione d'insieme della metodologia Mixed Reality e ne mostra in dettaglio il significato che assume nell'ambito dell'automazione.

2.1 Generalità sulla Mixed Reality

Con il termine Mixed Reality ci si riferisce alla fusione di mondi reali e virtuali con lo scopo di ottenere nuovi ambienti in cui gli oggetti fisici (reali) e gli oggetti digitali (virtuali) coesistono e interagiscono in tempo reale.

Nel 1994 Paul Milgram e Fumio Kishino hanno definito una **realtà mista** come *"*... un qualsiasi punto tra gli estremi del *continuum virtuality"* [(25)], dove il Continuum Virtuality si estende dalla parte completamente reale (Real Environment) alla parte completamente virtuale (Virtual Environment).

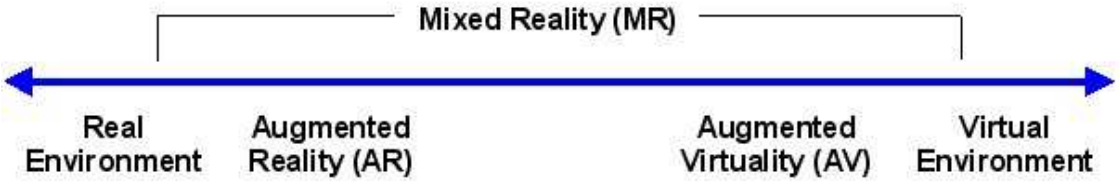

Figura 1: Continuum Virtuality

I due casi intermedi, definiti come Augmented Reality (Realtà aumentata) e Augmented Virtuality (Virtualità aumentata), rappresentano un ambiente reale integrato da oggetti virtuali il primo ed un ambiente virtuale integrato da oggetti reali il secondo. La figura 2 spiega in maniera esauriente tale classificazione.

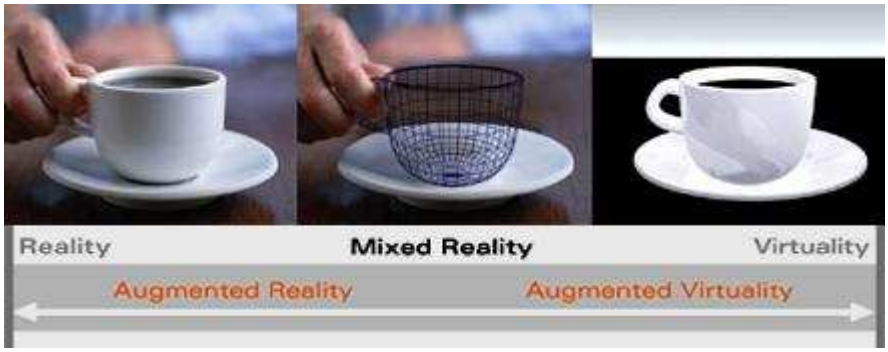

Figura 2: Esempio di Mixed Reality

Come già accennato la visione convenzionale di una realtà mista è un mondo in cui l'osservatore-partecipante è totalmente immerso nell'ambiente ed è in grado di interagire con i vari oggetti presenti (reali o virtuali). Tale ambiente riproduce scenari già esistenti o immaginari e può anche superare i limiti delle leggi della fisica attraverso la creazione di un mondo in cui le proprietà dei materiali, del tempo, dello spazio e della meccanica non vengono rispettate. Questa visione della Mixed reality assume sfumature sicuramente interessanti in contesti quali l'industria cinematografica e nella creazione di videogames; l'applicazione nel mondo dell'automazione potrebbe invece lasciare qualche perplessità. Nel seguito verrà invece mostrato come la Mixed Reality può essere uno strumento utile anche nell'ambito della produzione manifatturiera.

2.2 La Mixed Reality nell'automazione

Nell'ambito dell'automazione industriale le operazioni di test durante lo sviluppo di un software caratterizzano una fase molto importante e spesso costosa. Infatti, l'impatto tra persone e dispositivi fisici può essere estremamente pericoloso in caso di errori. Attualmente, soprattutto nel campo dei sistemi manifatturieri flessibili, non c'è l'abitudine sistematica di utilizzare strumenti o approcci dedicati alla fase di test, un

componente al massimo viene testato in anello aperto senza considerare tutti i problemi che deriveranno dalla sua integrazione col sistema finale. È consuetudine infatti limitarsi ad un test finale in anello chiuso che comprende tutti i componenti fisici della Macchina e il controllo software applicato, con l'inevitabile insorgere di numerosi problemi spesso di difficile soluzione. Nel corso degli ultimi dieci anni, l'evoluzione e il miglioramento della tecnologia, soprattutto nell'ambito delle prestazioni grafiche, ha catturato l'attenzione dei ricercatori ed ha infatti reso possibile la rappresentazione di un sistema manifatturiero flessibile tramite una simulazione grafica 3D. Da qui nasce l'idea di utilizzare tali strumenti nella fase di test realizzando dunque un sistema in anello chiuso dove il controllo è il reale software e hardware di controllo mentre l'impianto controllato è un modello grafico 3D simulato di quello reale. Si pensi ora ad un sistema in anello chiuso, dove il controllo è costituito da un software e la parte controllata può essere:

1. un Macchinario attualmente installato in azienda adibito alla produzione (cella);
2. una simulazione grafica 3D;
3. una ben definita combinazione di parti reali e modelli simulati.

Se consideriamo il terzo caso e supponiamo una fase di test in cui ci sono n celle di produzione manifatturiera, n-p reali e p simulate, ad esempio attraverso un tool di simulazione grafica 3D, si possono identificare tre tipi di componenti, la parte reale di controllo (RC), i componenti reali (RP) e i componenti virtuali simulati (VP). Questo semplice esempio ci riconduce alla definizione di Mixed Reality. Il problema principale è ora quello di formalizzare correttamente l'interfaccia tra i tre tipi di componenti.

Nel dettaglio:

- tra RC e RP avviene uno scambio di segnali reali, i classici segnali I/O oppure richieste di comandi per gli attuatori o ancora richieste di lettura sui sensori. Queste connessioni sono fatte con particolari protocolli dedicati allo scambio di informazione in tempo reale (fieldbus come: Modbus/TCP-Serial, Profibus, Profinet, EtherCat, etc.) e la scelta dipende dal caso particolare;
- tra RC e VP i segnali reali di controllo devono essere associati ai segnali di I/O di simulazione. In questo caso il protocollo di comunicazione diventa parte integrante della simulazione;

- tra RP e VP: questo è il caso più delicato, tale interfaccia infatti non prevede solo uno scambio di informazioni ma deve anche rappresentare il passaggio di pezzi fisici da una parte reale ad una simulata e viceversa. In entrambi i casi i pezzi possono essere modificati e tale flusso di informazione deve essere modellizzato.

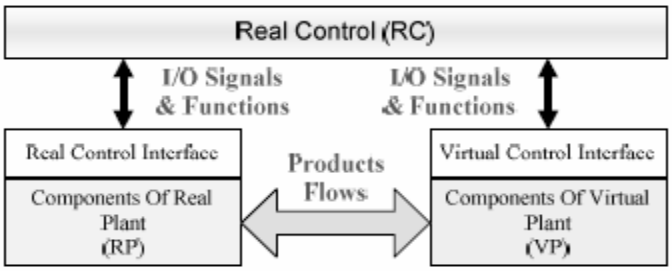

Figura 3: Architettura per la Mixed Reality

La configurazione presa in esame in questo elaborato presenta una Macchina di lavorazione per carter motore di motociclette completamente simulata, un sistema di carico e scarico pezzo reale posizionato davanti alla Macchina, un'unica stazione di carico/scarico pezzi grezzi/lavorati con operatore simulata ed un robot per movimentare i pezzi dalla stazione al pallet e viceversa simulato.

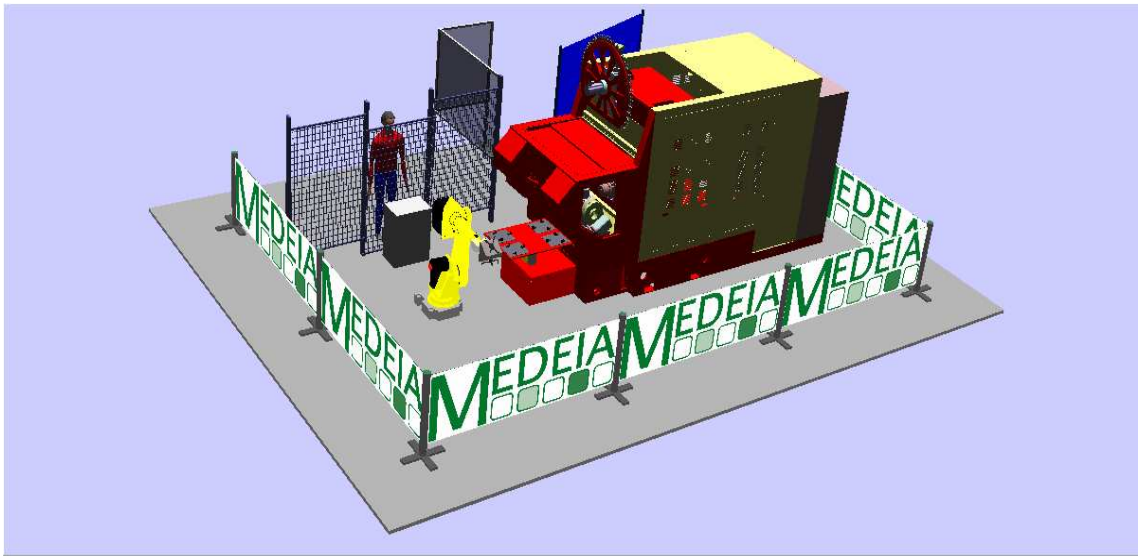

Figura 4: Ambiente di simulazione

Nel corso della prova sperimentale si sostituiranno alcuni dispositivi simulati con dei componenti fisici reali. A disposizione per il lavoro mostrato in questo elaborato si avrà il Pallet Changer, successivamente per la prova finale verrà aggiunto un robot.
Si possono dunque identificare diversi scenari.

Caso 1: completa simulazione
Con questa configurazione non vi è la presenza di parti reali, il Pallet Changer, la Macchina SPI, il robot e la stazione di carico/scarico pezzi sono simulati. È la condizione ideale per testare nuove logiche di controllo.

- RP: nessuno;
- VP: Macchina SPI, stazione carico/scarico, robot, Pallet Changer;
- RC: software di controllo.

Caso 2: Pallet Changer reale, Macchina SPI , stazione carico/scarico e robot simulati
È la configurazione utilizzata in questo elaborato per dimostrare l'interazione di parti reali e parti simulate.

- RP: Pallet Changer;
- VP: Macchina SPI, stazione carico/scarico, robot;
- RC: software di controllo.

Caso 3: robot reale, Macchina SPI, stazione carico/scarico, Palle Changer simulati
Configurazione mai utilizzata, in futuro si potrebbe testare tale architettura per rafforzare i risultati ottenuti.

- RP: robot;
- VP: Macchina SPI, stazione carico/scarico, Pallet Changer;
- RC: software di controllo.

Caso 4: Pallet Changer e robot reali, , Macchina SPI e stazione carico/scarico simulati
Sarà l'architettura finale per la chiusura del progetto MEDEIA.

- RP: robot, Pallet Changer;
- VP: Macchina SPI, stazione carico/scarico;
- RC: software di controllo.

Non sempre le alternative alla Mix Reality ottengono risultati meno soddisfacenti. Nel dettaglio, si potrebbero utilizzare modelli per simulare totalmente l'impianto oppure dei prototipi. La prima soluzione risulta sicuramente più economica della seconda ma spesso i simulatori sono troppo semplificati rispetto alla reale complessità dell'impianto. I prototipi eguagliano maggiormente le caratteristiche reali dell'impianto ma risultano molto costosi e lunghi da implementare. L'idea della Mixed Reality "usare la parte reale disponibile e simulare il resto" risulta comunque vantaggioso in moltissimi casi, soprattutto nei sistemi manifatturieri modulari e flessibili. Permette infatti di selezionare gli investimenti economici necessari e aumenta la competitività migliorando flessibilità e produttività. È da escludere il suo utilizzo con l'intero impianto reale o parte di esso nei casi di test di nuovi controlli quando essi implichino condizioni di pericolo per persone o danni a Macchinari o semplicemente quando tale prova risulta troppo onerosa.

2.3 I Simulatori

Con il termine *simulazione* s'intende un insieme di metodi e applicazioni che costituiscono un modello in grado di replicare, con un determinato grado di fedeltà, il comportamento di un sistema reale; solitamente la simulazione è fatta attraverso un computer ed un particolare software.

La simulazione, come la maggior parte dei metodi d'analisi, si basa sul concetto di *modello* come particolare rappresentazione del sistema reale che si vuole studiare. Per modello in questo caso s'intende un'architettura che è capace di interagire con il mondo esterno tramite degli ingressi e delle uscite imitando il comportamento di un sistema reale, cioè applicando gli stessi ingressi al modello e al sistema reale si vogliono ottenere le stesse uscite (o comunque uscite sufficientemente simili tra loro).

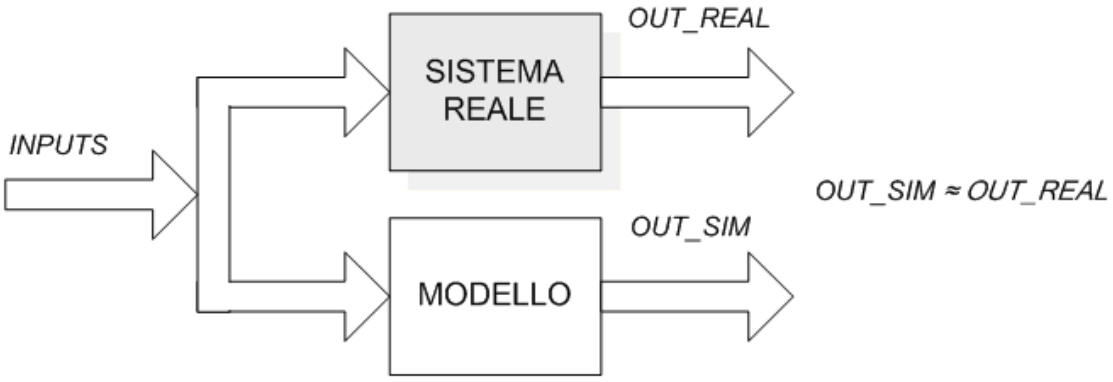

Figura 5: Confronto tra modello e sistema reale

La simulazione ha numerosi vantaggi, impossibili da ottenere eseguendo dei test nella realtà, e permette di comprendere e conoscere meglio un sistema reale senza effettuare nessun esperimento su di esso. La realizzazione di modelli di simulazione diventa necessaria ogni volta che la fase di testing risulta impossibile sui sistemi reali, ad esempio perché troppo costosa o troppo pericolosa, oppure perché il sistema reale non esiste e si vuole utilizzare la simulazione per seguirne e testarne la progettazione. In tutti questi casi la simulazione è il mezzo che permette di ridurre i costi e i fattori di pericolo che a volte renderebbero impossibile qualsiasi manipolazione del sistema; si pensi al settore aerospaziale o a quello nucleare nei quali è quasi sempre impensabile eseguire dei test direttamente sulle parti reali. Un'altra caratteristica fondamentale della simulazione, soprattutto quella software, è la facilità di modifica di un modello rispetto al sistema reale; questo fattore è molto rilevante nell'utilizzo della simulazione per la fase di progettazione dei sistemi. Il costo da sostenere per modificare una parte di software è, infatti, sicuramente inferiore, nella maggior parte dei casi applicativi, ai costi di modifica di impianti e strutture già progettate o realizzate. Nel modello è inoltre possibile eliminare i disturbi che influenzano i sistemi reali e che talvolta ne rendono difficili la comprensione e l'analisi. Tutti i vantaggi appena descritti evidenziano l'importanza della simulazione in tutte le operazioni d'analisi che richiedono un'interazione diretta con l'impianto e che spesso risultano, per questo motivo, impossibili nella realtà.

2.3.1 L’ambiente di simulazione SIMBA 3D

Come descritto in precedenza, una realtà mista prevede una parte reale e una simulata. Per supportare questa tecnica è dunque necessario un ambiente specifico di simulazione che offra possibilmente una rappresentazione grafia tridimensionale della parte di impianto simulata.

Nel corso degli ultimi dieci anni sono stati sviluppati molti software di simulazione ma le varie restrizioni in campo commerciale e l’alta capacità computazionale richiesta per le simulazioni dinamiche hanno portato allo sviluppo di un nuovo software capace di migliorare la simulazione cinematica real time in un approccio modulare denominato SIMBA (SImulation of Model Based Automatic system).

SIMBA definisce un ambiente per una simulazione modulare e gerarchica dei sistemi manifatturieri. Poiché le tipologie di impianti da simulare sono infinite, il modello proposto da SIMBA sfrutta il concetto di modularità con l’obiettivo di permette una più facile costruzione ed un maggiore riutilizzo dei modelli di simulazione creati precedentemente. Ciò consente di creare un sistema complesso attraverso l’unione di sistemi più piccoli e semplici in grado di funzionare anche autonomamente.

L’immagine grafica fornisce una chiara e semplice comprensione della simulazione ma non ne rappresenta l’obiettivo principale. In SIMBA i modelli sono cinematici, generalmente meno accurati ma più leggeri di quelli dinamici che risultano complessi da implementare e pesanti dal punto di vista della potenza di calcolo da utilizzare in un ambiente real time. Tale scelta è supportata dal fatto che lo scopo principale è testare le logiche di controllo: in questo modo i movimenti possono essere approssimati con semplici relazioni cinematiche riducendo la potenza di calcolo necessaria per la simulazione.

2.3.2 Il meta-modello SIMBA

Le parti basilari del meta-modello SIMBA sono i *Simulation Element (SE),* porzioni ben definite dell’impianto con associate un set di funzioni di controllo. Il meta-modello definisce due sotto tipi di Simulation Element, il Basic 3dSimulation Element (B3SET) che rappresenta il componente base di simulazione e il Composite 3dSimulation

Element (C3SET) che rappresenta l'aggregazione di più SE per creare componenti più complessi.

Il meta-modello SE definisce un'interfaccia standard comune tra i sottotipi B3SET e C3SET e permette l'aggregazione e lo scambio di informazioni tra dispositivi eterogenei simulati.

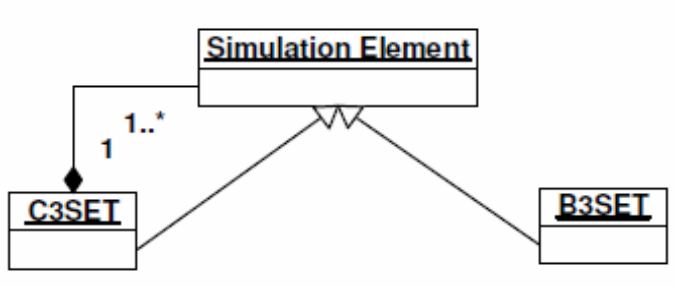

Figura 6: Simulation Element

Questa interfaccia è principalmente composta da:

- set di frames: sistemi di riferimento per operare l'aggregazione grafica in termini di saldature meccaniche;
- set di gates: usati per scambiare informazioni (segnali) tra gli elementi aggregati in un C3SET.

Un *B3SET* è composto da una parte grafica/cinematica e da una parte comportamentale/funzionale. La parte grafica è composta a sua volta dall'albero cinematico, formato da nodi (*Link*) che rappresentano le varie parti grafiche. L'albero cinematico contiene anche nodi privi di grafica, chiamati *Frame*, che costituiscono parte dell'interfaccia del *B3SET* e sono utilizzati per l'aggregazione grafica/meccanica dei vari elementi. Ogni nodo può avere un grado di libertà rispetto al nodo padre. Ciascun grado di libertà è descritto da un oggetto *DOF* (Degree Of Freedom) che riporta la descrizioni del grado di libertà, cioè una 4-tupla che indica la posizione iniziale e le posizioni di minimo e massimo e il tipo di grado di libertà (rotazione o traslazione) ed è inserito all'interno della parte grafica/cinematica del *B3SET*. La parte comportamentale/funzionale viene descritta attraverso l'uso dell'oggetto *BehaviorBlock*

(*BB*). Ciascun *BB* ha una propria implementazione e una propria interfaccia composta da un insieme di porte di input e di output.

Per costruire SE più complessi occorre aggregare più B3SET in un C3SET. Tale aggregazione eseguita dal *C3SET* riguarda sia la parte comportamentale sia la parte grafica. Nel dettaglio le connessioni grafiche avvengono per mezzo di matrici di roto-traslazione. Questo paradigma di aggregazione definisce un'architettura ad albero nella quale ogni nodo è connesso con i suoi nodi figli e con il suo nodo padre attraverso un insieme di connessioni.

All'interno del *C3SET*, come si può facilmente intuire, è presente l'insieme dei componenti di simulazione che porteranno il *C3SET* a rappresentare un determinato funzionamento di un sistema. Inoltre è compito del *C3SET* eseguire i collegamenti sia cinematici sia funzionali fra i vari componenti di simulazione interni.

Il meta-modello utilizzato è stato implementato in Java ([23]) ed è caratterizzato da 4 elementi:

1. ***SIMBA core:*** è la libreria java che implementa il meta-modello e un gruppo di classi utili per archiviare e caricare i modelli creati in file XML.
2. ***B3SET Editor:*** è un applicativo che permette di definire l'intero modello di un *B3SET* usando solamente l'interfaccia grafica senza scrivere alcuna riga di codice. Questo tool permette quindi, anche a persone non esperte, di definire un oggetto di simulazione base (*B3SET*). L'applicazione fa uso di un'unica finestra al fine di rendere più semplice e veloce l'uso dell'applicativo.

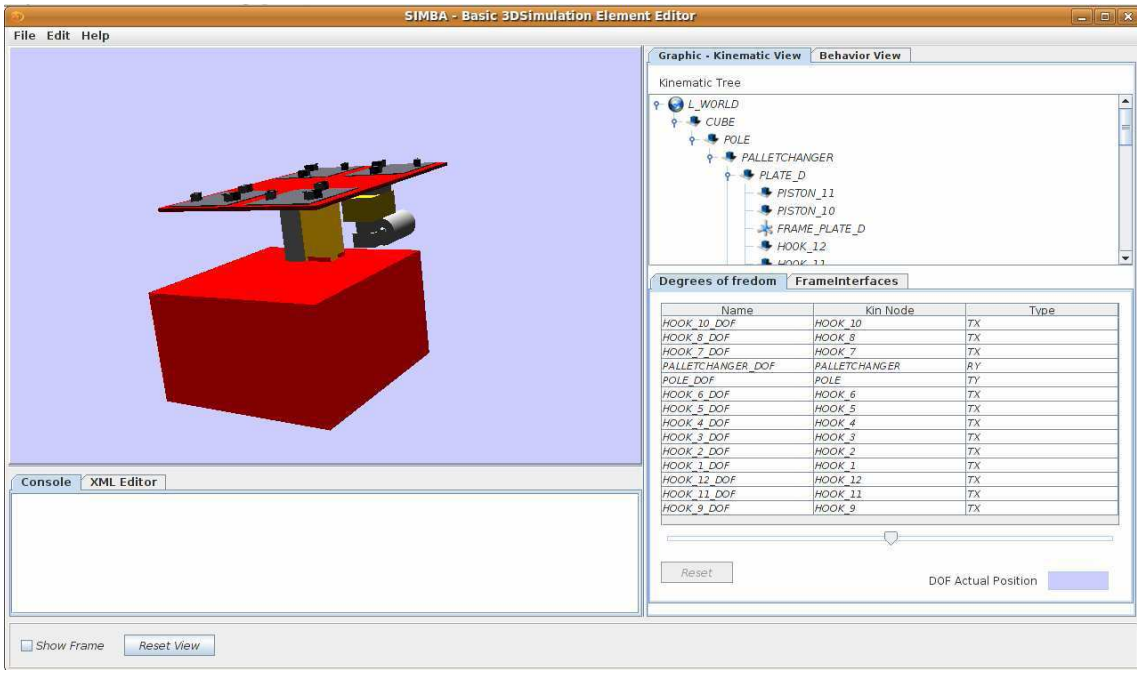

Figura 7: B3SET Editor – Kinematic View

La configurazione della finestra cambia a seconda di quale parte, grafica o comportamentale, del *B3SET* si vuole definire. Quando si definisce la parte grafica del *B3SET*, in alto a sinistra si vede un pannello che riporta lo stato attuale della rappresentazione tridimensionale del *B3SET*. A destra si vedono i pannelli che permettono l'aggiunta o l'eliminazione di un *Link* o un *Frame* (pannello in alto a destra) oppure l'aggiunta o l'eliminazione dei *DOF* e dei *Frames* appartenenti all'interfaccia del *B3SET* (pannello in basso a destra). Ogni modifica può essere apportata usando opportunamente i tasti del mouse sui rispettivi pannelli. In basso a sinistra troviamo i pannelli che riportano la descrizione XML del *B3SET* in costruzione e gli avvisi di errore che possono essere generati dai metodi definiti nella libreria SIMBA3D. Quando si passa alla definizione della parte comportamentale, alcuni pannelli cambiano visualizzazione.

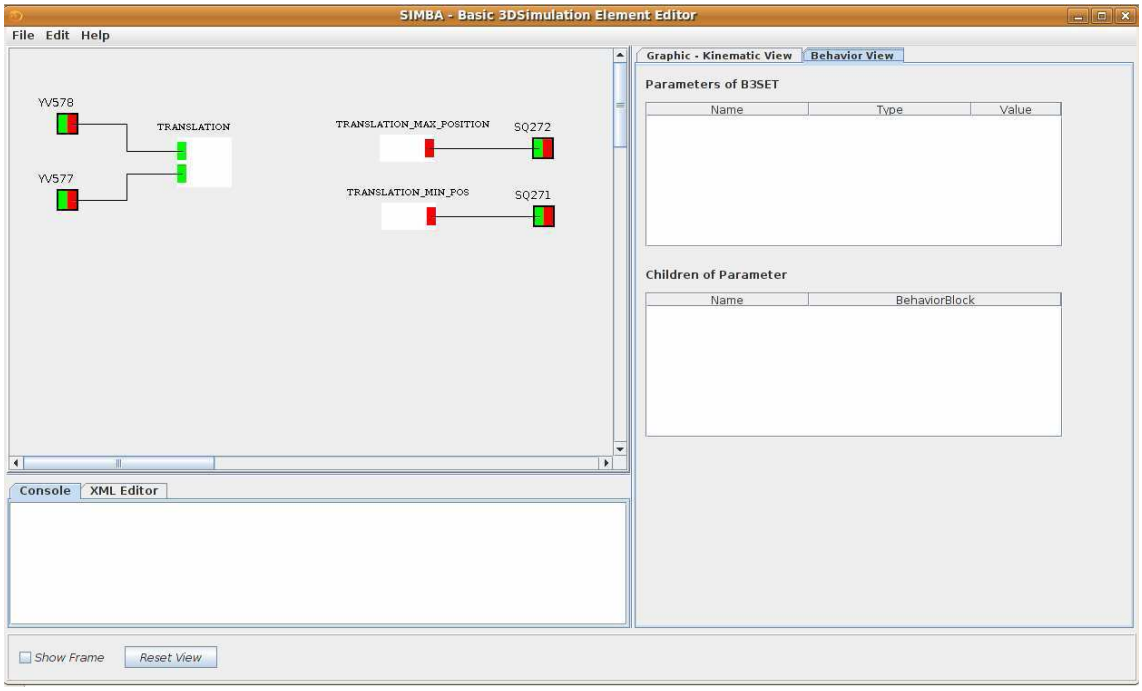

Figura 8: B3SET Editor - Behavior View

In alto a sinistra troviamo un pannello con grafica 2D che permette l'aggiunta o l'eliminazione dei vari elementi di simulazione che compongono il *B3SET*. Cliccando opportunamente con il mouse su questo pannello è possibile:

- inserire ed eliminare *BehaviorBlocks*, *InputGates* ed *OutputGates*;
- realizzare le connessioni fra i vari elementi;
- spostare i *BehaviorBlocks* e le connessioni all'interno del pannello al fine di rendere più chiara la visualizzazione dello schema.

Nella parte destra della finestra vengono ora visualizzati due pannelli. Il pannello in alto riporta i parametri definiti all'interno del *B3SET* in costruzione, mentre il pannello in basso riporta i parametri figli associati al parametro selezionato nel pannello precedentemente descritto. L'applicativo fornisce inoltre le funzioni necessarie per:

- creare un nuovo *B3SET*;
- salvare il *B3SET*;
- aprire un precedente *B3SET*.

3. ***C3SET Editor:*** permette di creare modelli di *C3SET* partendo dai *Simulation Elements* che si vogliono aggregare. L'editor è formato da una singola finestra

all'interno della quale ci sono diverse cartelle. Cliccando su ciascuna cartella vengono presentate delle tabelle che permettono di apportare modifiche al *C3SET* in costruzione. Tramite il *C3SET Editor* è possibile:

- aggiungere o eliminare *Simulation Elements*;
- aggiungere o eliminare i *Frames* che dovranno far parte dell'interfaccia del *C3SET*;
- aggiungere o eliminare gli *InputGates* e *OutputGates*;
- definire le connessioni fra i vari *Gates* dei *Simulation Elements* e i *Gates* del *C3SET* in costruzione;
- definire le connessioni grafiche fra i vari *Simulation Elements* che compongono il *C3SET*.

L'applicativo *C3SET Editor* offre inoltre funzioni per la persistenza dei modelli creati o modificati:

- creazione di un nuovo *C3SET*;
- salvataggio su file di un *C3SET*;
- caricamento di un *C3SET* precedentemente salvato su file.

4. ***SIMBA Viewer:*** è stato creato con l'intento di fornire un'interfaccia grafica all'utente per il controllo dell'esecuzione delle simulazioni. La simulazione può essere effettuata per un singolo *B3SET* o per un insieme di *Simulation Elements* raggruppati in un *C3SET*. L'applicativo fornisce anche altre funzioni quali:

- la gestione del tempo con la possibilità di accelerare e rallentare la simulazione in corso;
- stoppare, avviare e mettere in pausa la simulazione.

L'applicativo è composto da una singola finestra, all'interno della quale si trovano:

- un pannello grafico 3D per la visualizzazione del sistema in simulazione;
- tre bottoni che servono per l'avvio, stop e pausa della simulazione;
- uno barra orizzontale a scorrimento che permettere all'utente di scegliere opportunamente la velocità di simulazione;
- un pannello dove vengono riportate le informazioni provenienti dall'esecuzione dei metodi implementati nella libreria SIMBA3D ed eventuali errori.

Il tool offre anche la possibilità di muoversi all'interno dello spazio tridimensionale quando la simulazione è in corso, offrendo all'utente la possibilità di visualizzare anche i piccoli particolari che, dall'inquadratura iniziale del mondo virtuale non sono invisibili.

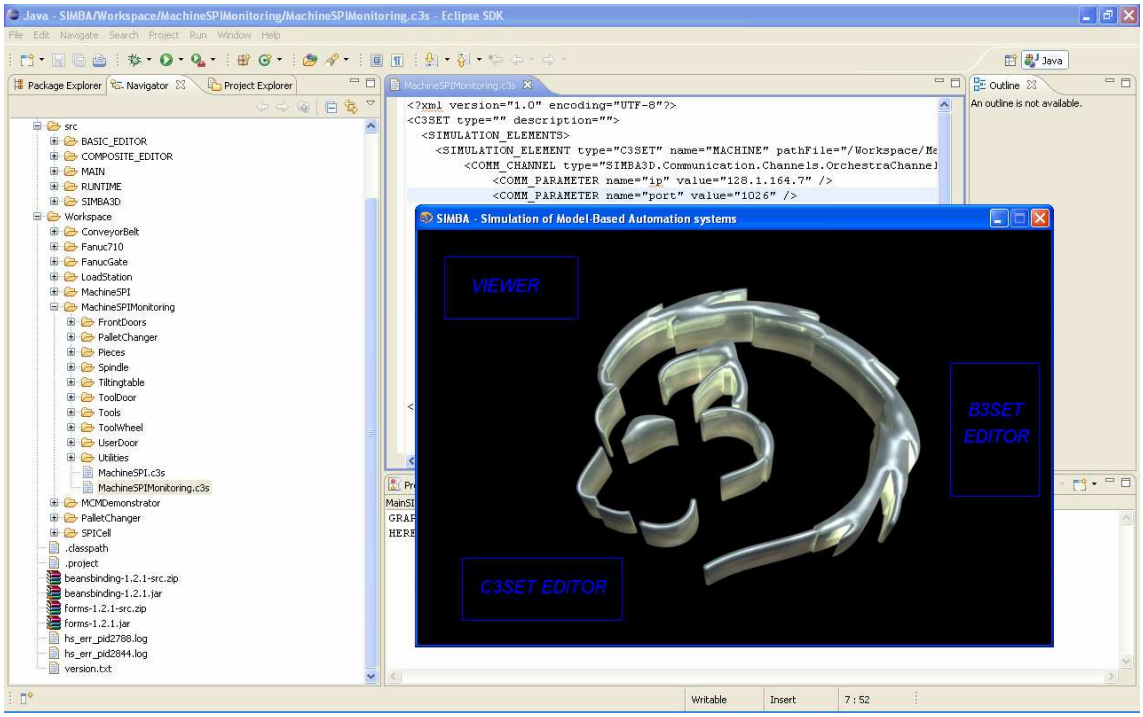

Figura 9: Overview dell'interfaccia grafica

2.4 Il Monitoring 3D

Il modello grafico di simulazione, in cui l'impianto viene riprodotto fedelmente e nella maggioranza dei suoi componenti e funzionalità, può essere impiegato anche per rappresentare a distanza il comportamento dell' impianto reale durante il suo effettivo esercizio, svolgendo quindi una funzione di monitoraggio. Questa architettura prevede di collegare i segnali dell'intero impianto controllato al simulatore, potendo così monitorare l'evoluzione dello stato dell'intero sistema reale tramite la sua rappresentazione grafica tridimensionale.

In questo modo è inoltre possibile introdurre alcune funzionalità avanzate per la diagnostica dei guasti e per il rilevamento di malfunzionamenti che possono verificarsi nell'impianto in tempo reale. Semplici implementazioni grafiche potrebbero evidenziare

sul monitor i componenti affetti da malfunzionamenti con un colore differente permettendo all'operatore di individuarli con rapidità evitando lunghi fermi dell'impianto.

I vantaggi di tale soluzione sono evidenti, ogni ora di fermo Macchina comporta gravi perdite economiche per l'azienda e non sempre risulta facile identificare il componente guasto in tempi rapidi. Sapendo invece quasi istantaneamente quale pezzo va sostituito o riparato il compito dell'operatore risulta agevolato. Le informazioni fornite dal sistema di monitoraggio potrebbero inoltre essere integrate da video per lo smontaggio e la sostituzione delle parti guaste evidenziate a monitor, facilitando ulteriormente il compito di chi deve intervenire per ripristinare il normale e corretto funzionamento dell'impianto.

2.5 Diagnostica

Un sistema automatico industriale deve garantire ottime prestazioni, sicurezza e rispetto di norme ambientali. Esso è progettato a priori su un sistema nominale, i malfunzionamenti non sono considerati; deve dunque essere equipaggiato di un sistema di diagnostica in grado di rilevare il mancato rispetto dei vincoli sopraelencati, semplice da sviluppare, implementare e validare e chi sia per quanto possibile economico.

Nell' ambito industriale si evidenziano:

- perdite funzionali sui componenti (es. blocco di una valvola);
- perdite di performance e sicurezza;
- perdite di informazioni (es. guasti su un host);
- numero di sensori limitati;
- componenti difficilmente accessibili;
- guasti a volte intermittenti.

Da decenni in ambito accademico la diagnostica è oggetto di studio e si basa sulla rilevazione di dinamiche anomale mediante l'osservazione delle misure a disposizione. È possibile riconoscere due approcci:

- ***Non model based***: si utilizzano reti neurali e tecniche di intelligenza artificiale;

- ***Model based***: si confronta l'evoluzione attuale del sistema con un modello (Osservatori, filtri di Kalman, ridondanze analitiche e analisi strutturale).

Il modello del controllo sintetizza ciò che l'impianto dovrebbe fare ed è definito come una sequenza ben precisa di comandi che specificano l'evolversi del ciclo nominale. In questo paradigma una ben definita sequenza di eventi generati dal controllo (comandi) è seguita da un insieme, generalmente non ordinato, di eventi osservabili generati dal plant (misure). In questo modo è possibile riconoscere ed isolare un guasto tutte le volte che il sistema non esegue il ciclo nominale.

2.5.1 Approccio Diagnoser

Si modellizza il sistema mediante un automa e i guasti vengono considerati persistenti e modellati come eventi non osservabili. Il cuore di tale approccio è la dinamica del sistema e gli stati raggiunti.

Il primo passo è la costruzione del modello con la composizione parallela tra il modello del plant e il modello del controllo:

- si esprime la concorrenza dei componenti;
- eventi comuni tra componenti necessitano di sincronizzazione;
- eventi privati possono avvenire sempre;
- lo spazio degli stati risultanti è un prodotto cartesiano.

Nel modello ottenuto si riconoscono:

- **eventi osservabili:** comandi dal supervisore e lettura del cambiamento di stato dei sensori;
- **eventi non osservabili:** cambiamenti nel sistema non registrati dai sensori e guasti.

Il secondo passo è la costruzione del diagnoser, un automa in grado di stimare lo stato del sistema, e quindi in grado di rilevare uno o più eventi di guasto, la cui evoluzione è data solo dal sottoinsieme di eventi osservabili del sistema.

La presenza di eventi non osservabili rende il modello ad automi del sistema non deterministico, occorre dunque affrontare il problema di stimare lo stato attuale del sistema stesso e costruire un automa deterministico in grado di farlo (l'osservatore diagnostico stesso). Con la definizione di deterministico, l'automa risultante presenta

transizioni dovute solo ad eventi osservabili e lo stato contiene una stima dello stato originale. Ad ogni nuovo step si associa allo stato raggiunto l'insieme raggiungibile non osservabile. Una volta definito l'automa diagnostico occorre arricchire lo stato dell'osservatore con informazioni sull'occorrenza del guasto (evento di interesse), affiancando ai nomi degli stati delle etichette che rappresentano "guasto avvenuto" oppure "guasto non avvenuto". Quando si costruisce l'insieme raggiungibile non osservabile si aggiornano tali etichette solo se per raggiungere lo stato la stringa non osservabile di azioni contiene il guasto. Dato che per ipotesi il guasto è considerato persistente, l'etichetta "guasto avvenuto" si propaga inalterata.

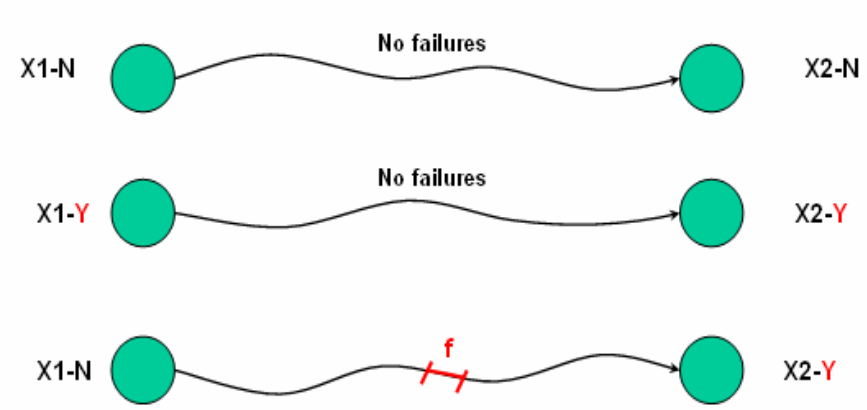

Figura 10: Propagazione del guasto

Condizione necessaria e sufficiente per l'isolamento del guasto è che il sistema risulti diagnosticabile per il guasto considerato, ovvero che dopo il suo accadimento si ha solo una stringa di eventi osservabili e di lunghezza finita che permetta di isolarlo.
Per questo motivo se l'automa diagnostico presenta un ciclo di stati incerti il guasto non è diagnosticabile. Riassumendo il discorso, condizione necessaria e sufficiente affinché un automa sia diagnosticabile è che il diagnoser non contenga cicli indeterminati.
Per quanto appena descritto se il linguaggio è diagnosticabile, il diagnoser è in grado di rilevare in un numero finito di passi l'occorrenza di un guasto entrando in uno stato certo.
L'approccio a diagnoser può quindi essere utilizzato in due modi:

- Off-line: per verificare la diagnosticabilità di un dato sistema;

- On-line: per identificare effettivamente l'accadimento di un guasto nel sistema durante il suo esercizio.

L'implementazione del diagnoser è un punto un po' delicato dell'intero approccio, infatti un numero di stati spesso troppo grande e la tipica non diagnosticabilità dei sistemi reali rendono questa fase particolarmente critica. Tipicamente per avere una versione eseguibile del diagnoser si possono seguire due strade:

Off-line: si memorizzano tutte le transizioni del sistema sotto diagnosi. Questo approccio richiede molta memoria ma una ridotta capacità di calcolo;

On the fly: gli eventi del sistema vengono processati in tempo reale. Ovviamente questo secondo approccio richiede poca memoria, non dovendo memorizzare tutti gli eventi del sistema, ma necessita di una capacità di calcolo elevata per poter processare in tempo reale ogni singola transizione del sistema sotto analisi.

Se un sistema risulta non diagnosticabile è possibile renderlo tale con diversi approcci:

- approccio passivo: si aggiungono dei sensori e dunque eventi osservabili. In questo caso occorre prestare attenzione al numero e al tipo di sensori da aggiungere (costi elevati);
- approccio attivo: modificare la strategia di supervisione per evitare l'instaurarsi di cicli indeterminati introducendo dei vincoli nel progetto del supervisore stesso.

2.5.2 Approccio TiDiaM

TiDiaM è l'acronimo di Timed Diagnoser Model; tale approccio è un'estensione del classico diagnoser ed è specializzato nell'isolamento on line dei guasti nei sistemi manifatturieri. Come per l'automa diagnostico classico, l'approccio TiDiaM segue l'evoluzione del sistema osservando gli eventi generati dal controllo e le misure registrate dai sensori ma aggiunge una funzione di time-out che rappresenta il massimo tempo disponibile per completare un'azione. Se durante l'esecuzione scade un time-out significa che è avvenuto un guasto. Con questa soluzione è possibile isolare i guasti senza un indefinito tempo di attesa. In questo caso il modello del plant contiene anche tutti i possibili stati osservabili di guasto (sia per gli attuatori che per i sensori).

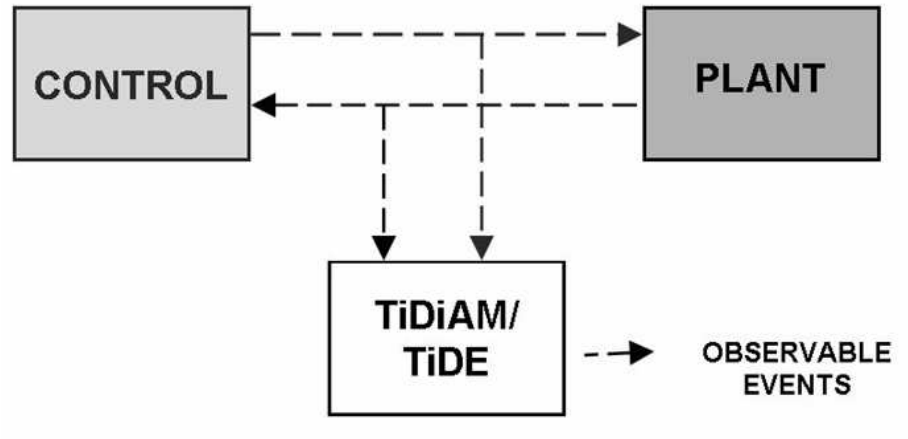

Figura 11: Schema di applicazione del metodo TiDiaM

Dopo la composizione parallela tra il modello dell'impianto (P) e il modello del controllo (C), il passo per ottenere l'automa diagnostico è la definizione di gruppi di possibili guasti simili detti partizioni dei guasti (fault partition).

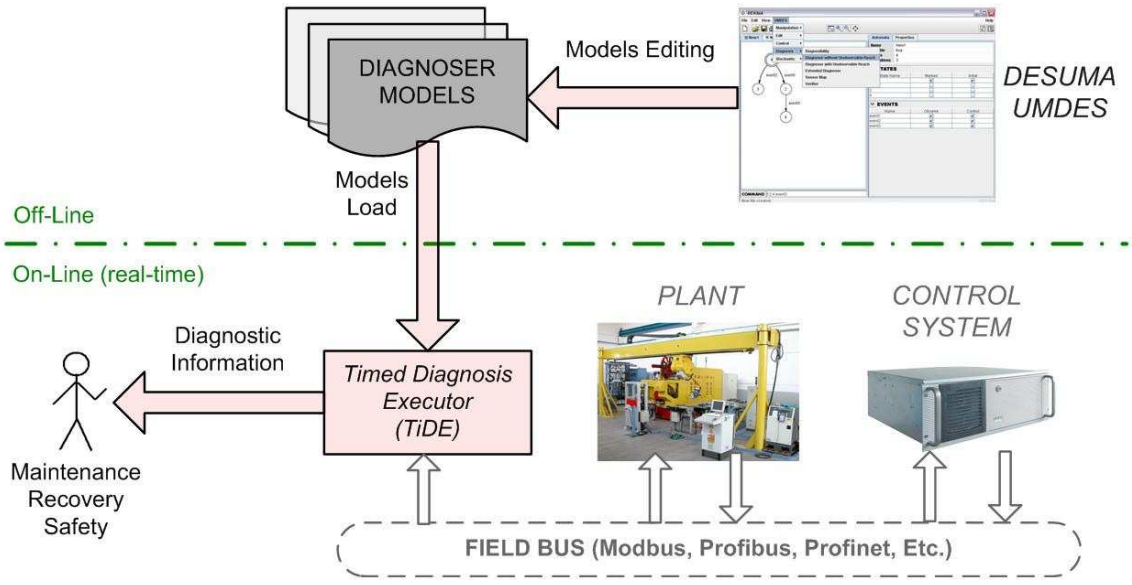

Figura 12: Architettura dell'applicazione del metodo TiDiaM in uno scenario industriale

La parte superiore rappresenta i passi eseguiti off-line, cioè la generazione dei modelli di diagnosi ottenuti con software come DESUMA-UMDES ([20]). La parte inferiore mostra invece l'esecuzione in real time del modello diagnostico TiDE, il quale è

connesso al plant tramite un bus di campo (Modbus, Profinet…) e fornisce le informazioni di diagnosi ad un operatore oppure ad un altro sistema automatico. Tali informazioni saranno utilizzate per la manutenzione o la riparazione dell'impianto.
TiDE ha un'architettura modulare composta da tre interfacce, da una rappresentazione interna del modello TiDiaM e da un esecutore dell'automa diagnostico real time.

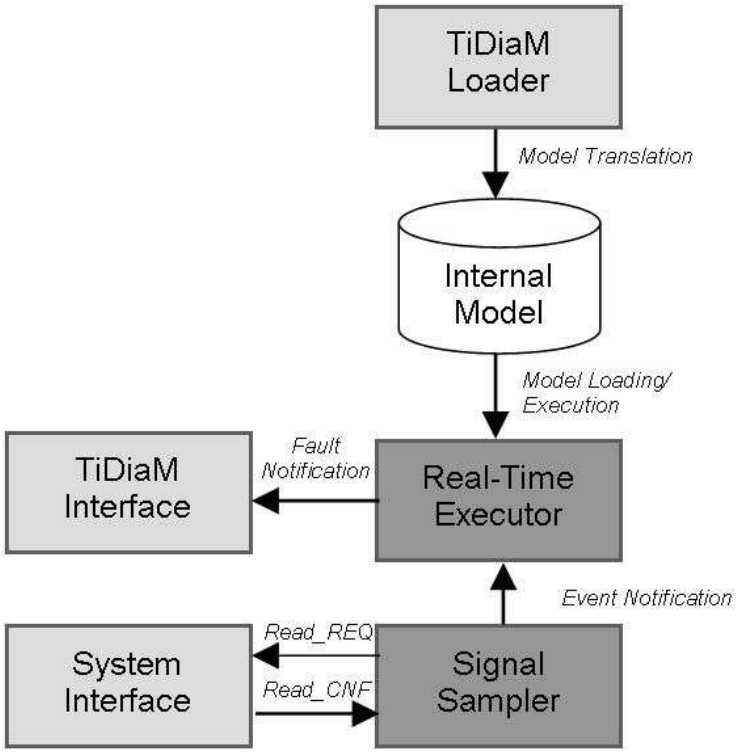

Figura 13: Architettura interna dell'esecutore real-time TiDiaM

Il modello interno rappresenta l'automa diagnostico e la sua struttura contiene gli stati, le transizioni, i modelli dei guasti e i gruppi di guasti. Ogni stato dell'automa ha un set di etichette che rappresentano gli stati stimati del sistema e le relative informazioni sui time-out per i guasti.
Il processo è composto da due differenti entità interagenti:

- i segnali campionati (Signal Sampler) che vengono interfacciati col sistema di controllo dall'interfaccia di sistema (System Interface);
- l'esecutore real time, cioè un software che implementa l'automa diagnostico specificato nel modello TiDiaM.

L’esecutore attende le notifiche osservate dal signal sampler ed esegue l’evoluzione dell’automa diagnostico. Dopo ogni transizione il real time executor effettua l’analisi diagnostica e controlla il manifestarsi di un guasto o lo scadere di un time-out. Se si verifica una delle due condizione parte l’avviso di occorrenza di guasto tramite l’interfaccia diagnostica verso l’operatore o il sistema automatico di gestione guasti.

2.6 PLC

Il controllore logico programmabile o Programmable Logic Controller (PLC) è un computer industriale specializzato in origine nella gestione dei processi industriali. Il PLC esegue un programma ed elabora i segnali digitali ed analogici provenienti da sensori e diretti agli attuatori presenti in un impianto industriale.

La caratteristica principale è la sua robustezza estrema, infatti normalmente il PLC è posto in quadri elettrici in ambienti rumorosi, con molte interferenze elettriche, con temperature elevate o con grande umidità. In certi casi il PLC è in funzione 24 ore su 24, per 365 giorni all'anno, su impianti che non possono fermarsi mai.

La prima azione che il PLC compie è la lettura degli ingressi e con questo si intende tutti gli ingressi sia digitali che analogici, on board o su bus di campo (schede remote collegate al PLC o con una rete di comunicazione). Dopo aver letto tutti gli ingressi, il loro stato viene memorizzato in una memoria che è definita "Registro immagine degli ingressi". A questo punto le istruzioni di comando vengono elaborate in sequenza dalla CPU e il risultato viene memorizzato nel "Registro immagine delle uscite". Infine, il contenuto dell'immagine delle uscite viene scritto sulle uscite fisiche ovvero le uscite vengono attivate. Poiché l'elaborazione delle istruzioni si ripete continuamente, si parla di elaborazione ciclica; il tempo che il controllore impiega per una singola elaborazione viene detto tempo di ciclo (solitamente da 10 a 100 millisecondi).

Un PLC è composto da un alimentatore, dalla CPU che in certi casi può avere interna o esterna una memoria RAM o ROM o EPROM o EEPROM, da un certo numero di schede di ingressi digitali e uscite digitali, e nel caso in cui sia necessario gestire grandezze analogiche, il PLC può ospitare delle schede di ingresso o di uscita sia analogiche che digitali.

La CPU durante il funzionamento a regime, colloquia con tutte le schede connesse sul BUS del PLC, trasferendo dati e comandi da e verso il mondo esterno (input e output).
Una delle caratteristiche peculiari di molte CPU è la capacità di poter gestire le modifiche del programma di gestione del processo durante il normale funzionamento. Questa possibilità è estremamente utile nel caso di impianti che devono essere sempre attivi, come ad esempio nel controllo di processo e nella produzione industriale in serie.
All'interno della CPU sono varie parti, tra cui:

- unità di gestione, ovvero informazioni di gestione del PLC stesso, impostate dal costruttore e trasparenti all'utente;
- archivio di temporizzatori e contatori funzionali all'operatività del PLC;
- memorie immagine del processo, cioè le informazioni in ingresso ed i comandi in uscita del processo;
- memoria utente, in cui vengono scritti i programmi che il PLC deve eseguire;
- interfaccia per il dispositivo di programmazione, che comunica con gli strumenti di programmazione;
- bus dati, comando, indirizzi per la veicolazione dei dati fra le varie parti e con l'esterno della CPU.

Il PLC durante il suo funzionamento può comunicare con computer, con altri PLC oppure con altri dispositivi come le macchine CNC (i torni e/o le frese a controllo numerico delle aziende). La comunicazione con computer e altri dispositivi avviene tramite tipi di connessione standard come RS232 o TCP/IP. La comunicazione con altri PLC avviene tramite protocolli standard, ad esempio: Profibus, TCP/IP, Modbus.

2.6.1 Programmazione di un PLC

Il PLC per ottemperare ai suoi compiti deve essere programmato. La programmazione del PLC è effettuata normalmente con un PC sul quale un software specializzato permette di creare programmi da scaricare nella memoria della CPU del PLC. Questi software di programmazione possono leggere il programma direttamente dalla memoria della CPU, e visualizzare il programma sul PC.

Normalmente il programma viene scritto su PC, quindi scaricato sul PLC, e salvato sul PC stesso, per ulteriori modifiche o per sicurezza.
La normativa IEC 1131-3 del 1993 ha standardizzato 5 linguaggi di programmazione, di cui 3 grafici e 2 testuali.

Linguaggi grafici

- Ladder Diagram detto Linguaggio a contatti, è il linguaggio più usato fino a pochi anni fa, in quanto era la trasposizione informatica dei circuiti elettrici usati daglielettrotecnici. L'automazione industriale infatti era basata su sistemi a logica cablata, il PLC (controllore di logica programmabile) ha permesso di trasportare i concetti della logica cablata nel linguaggio Ladder. Il programmatore semplicemente utilizza simboli logici corrispondenti a segnali di ingresso e di uscita per implementare la logica non più cablando i relè, ma disegnando gli schemi elettrici nel software di programmazione.
- Sequential function chart (SFC), detto diagramma funzionale sequenziale, viene usato anche come strumento di specifica. Tale linguaggio permette di implementare facilmente una Macchina (o automa) a stati finiti.
- Function Block Diagram (FBD o FUP) detto Diagramma a blocchi funzionali, è analogo ai diagrammi circuitali.

Linguaggi testuali

- Instruction List (IL o AWL) detto Lista di istruzioni. È un linguaggio di basso livello molto simile all'Assembler. Può essere facilmente ricavato dal Ladder.
- Structured Text (ST) detto Testo strutturato è un linguaggio di alto livello simile al Pascal.

2.6.2 Orchestra Control Engine

Per questo elaborato si è utilizzato Orchestra Control Engine ([21]). Esso è una suite di componenti software per la progettazione, sviluppo e realizzazione di applicazioni di controllo in tempo reale per macchine utensili, celle di lavorazioni, robot e altri complessi sistemi industriali. Oltre ad eseguire applicazioni critiche in tempo reale, Orchestra è dotato di programmi per lo sviluppo, il debug e le prove che supportano

l’utente nella realizzazione di programmi controllo. Il vantaggio principale dell’intero ambiente è il suo costo limitato, basta infatti disporre di un personal computer: installando la suite, il PLC utilizzerà le risorse fisiche del computer stesso (CPU e Memoria) evitando l’acquisto di componenti esterni. Il softPLC e gli ambienti di sviluppo implementano funzioni di editing per i linguaggi definiti dalla normativa IEC61131, è quindi possibile implementare le funzioni di controllo in SFC, LD, FBD, ST e C. Una serie di menu a tendina permettono di gestire le librerie, di aggiungere, eliminare o modificare variabili ma soprattutto di visualizzare la fase di debug del codice con riferimenti alle righe dove sono presenti istruzioni errate e il tipo di errore commesso.

La gestione della comunicazione dei segnali di input e output può avvenire tramite gli standard più diffusi, come porte seriali, Profibus o CanOpen.

L’utilizzo di tale software presenta inoltre altri vantaggi:

- un framework open source che permette di inserire il proprio know-how specifico e di interfacciarsi con entità esterne sviluppando applicazioni sia per il controllo di moto sia per il controllo logico;
- la modularità, offrendo la possibilità di riutilizzare l’algoritmica sviluppata attraverso la sua suddivisione in unità funzionali distinte;
- è possibile dislocarlo su risorse hardware fisicamente separare (anche a diverse centinaia di metri);
- la riconfigurabilità permette di modificare i parametri del sistema, sia offline che runtime;
- Approccio visivo all’ambiente di sviluppo.

3

IL PROGETTO MEDEIA

3.1 Sistemi di controllo: modelli riutilizzabili

La progettazione di sistemi di controllo per impianti automatizzati è oggi guidata principalmente da un approccio centralizzato in base al quale un controllore, tipicamente a logica programmabile (PLC – Programmable Logic Controller), gestisce i segnali di un insieme di dispositivi hardware che compongono una porzione più o meno grande di un impianto. La maggior parte dei produttori si basa quindi su quanto dettato dalla normativa IEC-61131 che definisce uno standard per l'hardware e per la programmazione di logiche di controllo per PLC. Negli ultimi anni, l'espansione dei mercati internazionali e l'aumento dei mix produttivi ha portato allo sviluppo di impianti di produzione, soprattutto nel campo manifatturiero, sempre più complessi e flessibili, in modo da avere a disposizione soluzioni modulari facilmente modificabili e riconfigurabili. Un'architettura modulare e un'organizzazione flessibile per questi nuovi sistemi di produzione non possono che entrare in contrasto con il paradigma utilizzato fino ad ora per lo sviluppo di controllori centralizzati, e proposto nella stessa normativa IEC-61131. Il controllista si trova a dover gestire autonomamente e senza adeguati supporti concettuali e pratici l'onere di inserire nelle logiche e nelle architetture di controllo quel grado di flessibilità che permette di raggiungere la modularità e la riconfigurabilità richieste per i moderni sistemi di produzione.

Un'evoluzione dello standard IEC-61131 è stata pubblicata nel 2004 con la normativa IEC-61499 che introduce molti dei concetti chiave dei linguaggi di programmazione orientati agli oggetti nella progettazione dei sistemi di controllo. In particolare, viene

introdotto il concetto di blocco funzionale (FB, Function Block) come entità base per l'implementazione delle funzioni di controllo. Ciascun blocco ha una propria interfaccia e un proprio corpo interno costituito essenzialmente da un automa (Ecc, Execution Control Chart) che evolve secondo eventi notificati in ingresso. A ciascuno stato dell'automa può essere quindi associato un insieme di algoritmi che, elaborando i dati in ingresso al blocco, definiscono i valori delle sue variabili di uscita. Questa evoluzione permette di esprimere meglio e più facilmente il grado di flessibilità richiesto oggi per i sistemi di produzione automatici. Resta tuttavia da risolvere il problema dell'integrazione di numerosi sistemi eterogenei che spesso sono utilizzati insieme per ottenere nuove funzionalità sfruttando le peculiarità di diversi fornitori e produttori. Occorre quindi trovare un modo rapido e possibilmente standardizzato per lo sviluppo di software di controllo per diverse piattaforme e tecnologie. Inoltre, c'è l'esigenza del riutilizzo delle soluzioni di controllo sviluppate con tecnologie hardware diverse. Una possibile soluzione è quella di utilizzare un approccio per la progettazione basato sui modelli (model-driven engineering) che preveda cioè la definizione di modelli del controllo e del sistema da controllare come primo passo e da questi, con l'aiuto di traduttori e generatori automatici di codice per ciascuna particolare piattaforma, permetta di ottenere le implementazioni dei diversi blocchi di controllo. In effetti, la programmazione e la progettazione ad oggetti nascono come desiderio di riutilizzo di codice, ma il reale vantaggio in automazione nasce dal riutilizzo dei modelli. Infatti, i modelli permettono non solo di riusare gli algoritmi incapsulati negli oggetti, ma anche le relazioni e i vincoli tra gli oggetti stessi, il che tipicamente rimane sostanzialmente immutato in vari progetti. In questo modo, si estende quasi automaticamente anche il riutilizzo al di fuori di un singolo progetto di automazione, all'interno di famiglie di interi progetti. Il riutilizzo nasce dunque dall'adozione di modelli. Ciò aumenta anche il livello di manutenibilità del sistema e la sua evoluzione e modifica nel tempo.

3.2 Il progetto MEDEIA

L'approccio basato su modelli è il cuore di MEDEIA (Model-Driven Embedded Systems Design Environment for the Industrial Automation Sector), progetto triennale cofinanziato all'interno del settimo programma quadro (FP7) della Comunità Europea e

avviato il 1° gennaio 2008. L'obiettivo principale del progetto è quello di definire un framework e una metodologia integrati per il progetto di sistemi di automazione che prevedono l'integrazione di diversi dispositivi eterogenei pur mantenendo un elevato grado di flessibilità del sistema controllato finale. L'idea è quella di ottenere un modello centralizzato del sistema da controllare composto da diversi componenti, definiti Automation Component (AC), ciascuno dei quali è rappresentato da un insieme di modelli che definiscono diagnostica, simulazione e controllo del singolo elemento considerato. L'intero sistema controllato è quindi ottenuto come aggregazione di più componenti a diversi livelli gerarchici. L'approccio modellistico diventa quindi il cuore della progettazione del sistema automatico e identifica il vero know-how del controllista e del produttore del sistema stesso. In particolare, la definizione del sistema come aggregato di tanti sotto-sistemi permette la creazione di librerie di componenti riusabili per la progettazione di impianti anche diversi tra loro, velocizzando così sia la fase di progetto sia quella successiva di messa in opera.

L'ambiente e i modelli MEDEIA sono stati implementati in un software costituito da un ESB (Enterprise Service Bus), un'architettura software che permette di connettere diverse applicazioni attraverso uno scambio di messaggi ottenendo quindi una struttura multipiattaforma. L'accesso all'ESB avviene attraverso le API JMS che sono state implementate all'interno di un editor con interfaccia "user-friendly" fornito direttamente da MEDEIA che facilita la gestione delle informazioni contenute all'interno di un repository nel quale vengono memorizzati tutti i modelli realizzati. L'architettura, resa scalabile dalla presenza dell'ESB, può essere integrata con software esterni in grado di interfacciarsi e di editare particolari parti dei modelli presenti nel repository del progetto.

L'ambiente di sviluppo realizzato in MEDEIA è in grado di descrivere, oltre alle parti di controllo, anche il comportamento e la cinematica di ogni componente reale che l'utente vuole realizzare. A tale scopo, vista la molteplicità di componenti presenti nella realtà, MEDEIA usa un approccio modulare.

Gli elementi base di MEDEIA che vengono usati per costruire sistemi più complessi sono:

- *AutomationComponent*: può essere sia Composite che Basic. Permette di descrivere la parte di controllo dell'impianto;
- *DiagnosticComponent*: può essere sia Composite che Basic. Permette di descrivere quei componenti addetti alla diagnostica;
- *PlantComponent*: permette di descrivere l'impianto facendo uso anche della cinematica.

Per ottenere una struttura modulare ogni *PlantComponent* deve contenere al suo interno altri *PlantComponent* collegati fra di loro che svolgono delle funzioni generalmente meno complesse del *PlantComponent* esterno. MEDEIA esprime la parte puramente cinematica attraverso l'elemento *KinematicPort* che permette di connettere due *KinematicPort* sfruttando i parametri di Denavit-Hartenberg. Questi parametri esprimono il passaggio da un sistema di coordinate al successivo e, combinando delle matrici opportune con la tabella dei parametri, si ottiene una matrice che trasforma un punto espresso nel sistema finale in quello iniziale. Rispetto ai collegamenti cinematici, MEDEIA non permette di specificare direttamente la grafica di un componente.

Le porte fisiche presenti nei componenti reali non possono essere descritte attraverso le *KinematicPort* quindi MEDEIA fornisce un ulteriore elemento, chiamato *PlantPort*, che permette anche di definire dei flussi di dati. Attraverso informazioni aggiuntive si può definire anche la direzione, il valore iniziale e il tipo dei dati nel flusso.

Per descrivere la parte comportamentale del componente, MEDEIA sfrutta una versione semplificata del diagramma degli stati di UML chiamato *Timed State Chart*. Questo diagramma descrive il comportamento attraverso l'uso di stati che possono contenere delle azioni. Quest'ultime possono essere eseguite quando si arriva nello stato (*OnEntryActionList*) oppure quando lo si abbandona (*OnExitActionList*) passando in un altro stato. Nel caso in cui l'azione abbia conseguenze sulla parte cinematica, l'azione deve contenere un riferimento esplicito alla *KinematicPort* che viene modificata mentre, se le conseguenze riguardano l'output, l'azione deve contenere un riferimento esplicito alla *PlantPort* interessata. Gli stati posso essere collegati fra di loro attraverso delle transizioni alle quali sono associate delle condizioni e nel caso in cui la condizione risulti vera l'esecuzione passa da uno stato all'altro.

All'interno del quadro formale, il modello dell'Automation Component ha bisogno di essere rappresentato in una varietà di altri modelli.

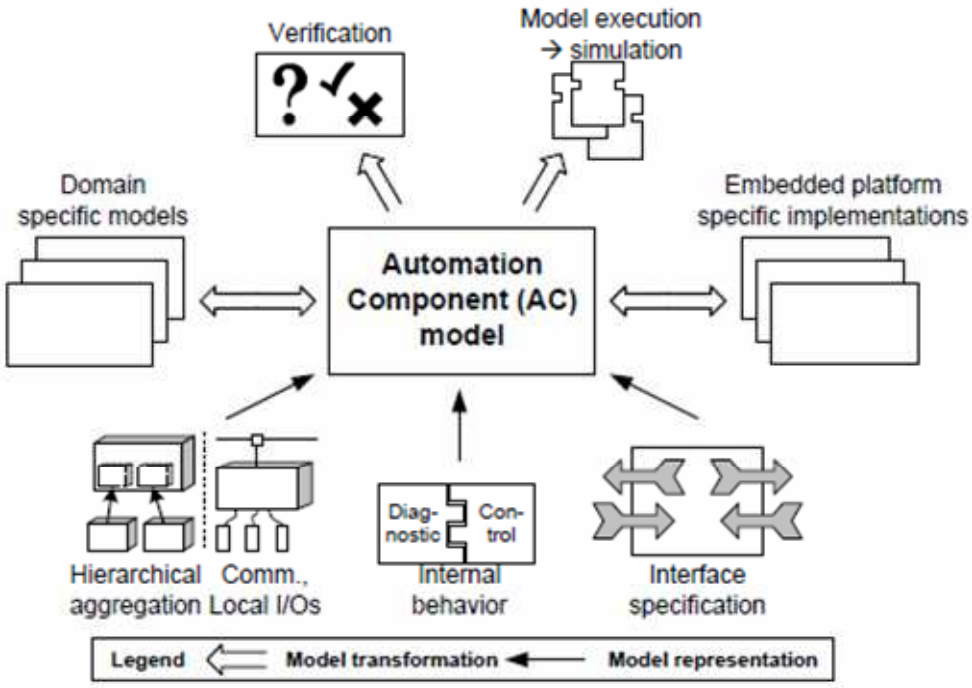

Figura 14: Quadro formale dell'approccio MEDEIA

Nel dettaglio:

- modelli di dominio specifico: nell'ambito dell'automazione industriale e dei sistemi di controllo esistono una grande varietà di linguaggi e tecniche di specifica. Il modello AC non sostituirà nessuno di questi, infatti ogni tecnica sarà considerata come un dominio specifico e utilizzando trasformazioni bidirezionali sarà possibile specificare ed analizzare il modello dell'impianto da ogni specifica prospettiva. In questo modo il progettista ha la possibilità di comprendere e specificare gli Automation Component nel suo modo più familiare;
- implementazioni integrate della piattaforma: per le stesse ragioni sopra descritte, come per i modelli di dominio specifico, l'implementazione di un AC (indipendentemente dal livello all'interno della struttura gerarchica di un impianto) è ancora una volta caratterizzata da una grande varietà di diversi sistemi e linguaggi di programmazione. L'approccio MEDEIA introduce di nuovo modelli specifici per

questi diversi sistemi e linguaggi di programmazione. Con l'uso di trasformazioni bidirezionali del modello, un AC può essere trasferito per essere eseguito su hardware reale e anche i componenti predefiniti possono essere integrati nel modello;

- simulazione dei modelli: il progetto provvede alla trasformazione del modello di un Autamation Component in un framework di simulazione per integrarlo in ogni momento in una parte dell'impianto già esistente;
- verifica: oltre alla simulazione, la verifica dei componenti e della loro aggregazione fornisce una maggiore efficienza; pertanto è inclusa la trasformazione del modello di un AC in una descrizione formale per controllare eventuali contrasti tra l'interfaccia e l'ambiente oppure per verificare che le specifiche dell'interfaccia non siano in contrasto con l'effettiva implementazione.

3.2.1 Automation Component

L'Automation Component (AC) è l'elemento principale dell'approccio MEDEIA. Lo si può definire come una combinazione di software e hardware integrato (embedded). Si consideri un robot industriale come esempio di Automation Component: esso è caratterizzato sia da una parte hardware (le parti meccaniche del robot e l'architettura di controllo fisica) sia da una parte software (il software applicativo di controllo). Il concetto gerarchico di Automation Component permette un'aggregazione multipla di AC allo stesso livello o ad un livello più alto, aggiungendo via via delle nuove funzionalità. Nell'approccio MEDEIA non c'è limite al numero di livelli di AC.

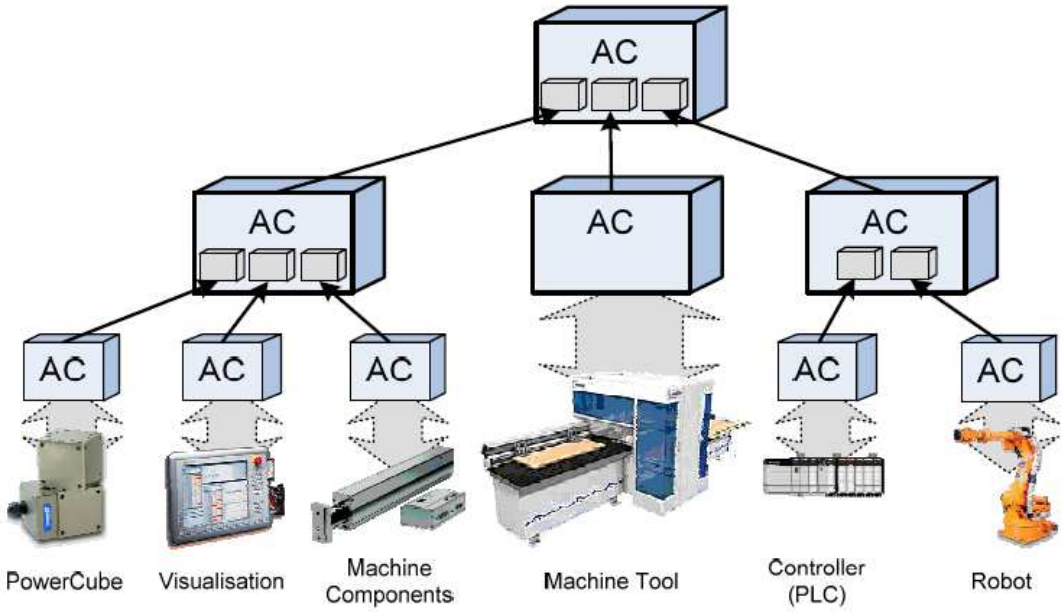

Figura 15: Struttura gerarchica di un Automaion Component

L'implementazione di Automation Component è composta da tre parti essenziali.

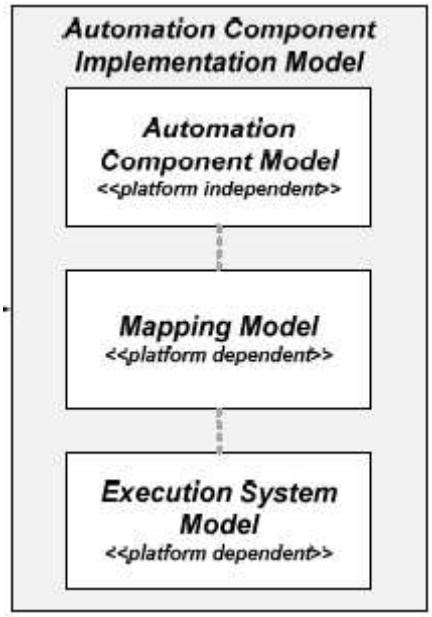

Figura 16: Implementazione di un Automation Component

1. Modello dell'Automation Component

Contiene le informazioni sulle interfacce, sugli aspetti funzionali, sulla diagnostica (identificazione dei guasti) e sul modello dell'impianto. Questa parte è indipendente dalla piattaforma implementativa che si sceglierà in seguito.

2. Modello del sistema di esecuzione

Rappresenta il modello dell'hardware che verrà utilizzato. Contiene informazioni sul sistema operativo, sulla memoria ecc. Questa parte dipende dalla piattaforma specifica di implementazione.

3. Modello del Mapping

Permette di colmare il divario tra il modello dell'Automation Component, indipendente dalla piattaforma utilizzata, al momento della traduzione per una specifica piattaforma. Contiene le informazioni sui segnali di input e output e su come questi si mappano ad esempio in aree di memoria della piattaforma scelta.

Non appena un Automation Component viene collegato ad un dispositivo hardware integrato, si ottengono informazioni aggiuntive dalla comunicazione e dagli input/output locali. Le prime risultano importanti soprattutto in fase di progettazione per l'integrazione di dispositivi hardware diversi. Gli I/O locali diventano importanti durante la l'implementazione della piattaforma specifica dato che introducono segnali nelle fasi di verifica e simulazione.

4

MACCHINA SPI

In questo capitolo viene descritta nel dettaglio la Macchina SPI, le operazioni che svolge, i componenti principali (sensori e valvole) che caratterizzano i vari dispositivi e si mostra il lavoro di revisione delle schede tecniche (diagrammi SFC e linguaggio che descrive le funzioni) riportando l'esempio del Pallet Changer.

4.1 Architettura gerarchica funzionale

Per generare funzioni di controllo non banali per sistemi manifatturieri complessi è consigliabile organizzare la progettazione e quindi l'implementazione in maniera gerarchica funzionale come mostrato nel diagramma UML di figura 17.

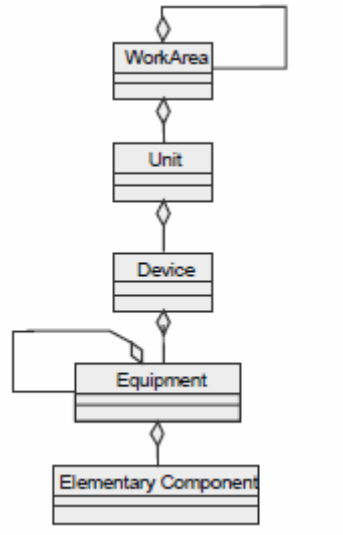

Figura 17: Diagramma UML di un'architettura gerarchica funzionale

Questa organizzazione può essere applicata, per esempio, a tutti quei centri complessi di lavorazione la cui scomposizione funzionale del plant è rappresentabile come in figura 18.

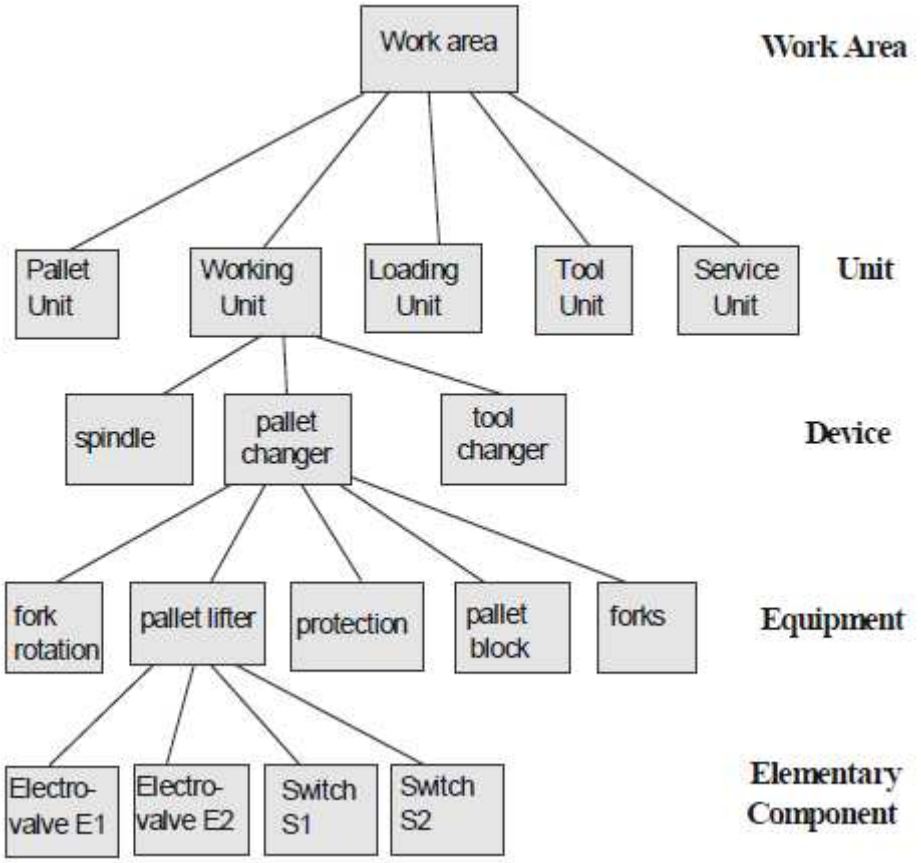

Figura 18: Esempio di decomposizione funzionale per un impianto di produzione manifatturiera

Tuttavia, una più chiara organizzazione delle funzioni di controllo si ottiene separando nettamente tutte le funzioni che operano direttamente sul sistema e non si preoccupano del pezzo specifico da realizzare da tutte le altre funzioni che interagiscono direttamente col prodotto. Le seconde sono legate direttamente al prodotto e hanno l'obiettivo di:

- imporre delle specifiche al flusso di un prodotto (per esempio, una zona definita del buffer B deve contenere solo pezzi grezzi, mentre un'altra pozione solo pezzi semi-lavorati in attesa che una Macchina libera li finisca);
- risolvere possibili conflitti (per esempio, quando la lavorazione su due prodotti è completata, la navetta di trasporto deve decidere quale prelevare per primo);

- gestire le code (cioè quale pezzo ha la priorità di venire lavorato);

Le prime invece, dipendenti dalla configurazione del plant, non cambiano in base al particolare pezzo da produrre, basti pensare al movimento di un pezzo dal buffer B alla Macchina di lavorazione M; il percorso sarà eseguito sempre nella stessa maniera. Per questo motivo, l'approccio progettuale ai due diversi strati è concepito in maniera differente:

a)leggi di controllo legate al prodotto: sono progettate con un approccio di vigilanza e sono inclini al paradigma agent-based quando la complessità e la flessibilità dell'impianto sono elevate;

b)leggi di controllo legate al plant: sono progettate con criteri in cui il comando rilasciato va eseguito obbligatoriamente e non come un insieme di comandi possibili.

4.2 Descrizione della Macchina SPI

La Macchina SPI è una Macchina utensile a 5 assi per l'asportazione di truciolo ad alta velocità su carter per motori di motociclette. I cinque gradi di libertà sono dati da un sistema cartesiano che permette il posizionamento del mandrino nell'area di lavoro, da una tavola inclinabile situata di fronte al mandrino e da due fixture presenti sui due piatti rotanti fissati alla tavola inclinabile. Le fixture vengono utilizzate per fissare saldamente i pezzi durante le lavorazioni.

Sulla parte superiore dell'area di lavoro è presente un magazzino utensili locale a ruota, capace di contenere al massimo 30 utensili. Il magazzino può ruotare e traslare, permettendo il cambio utensile con il mandrino senza ricorrere ad alcun meccanismo esterno. L'area di lavoro è chiusa da due porte frontali che la separano dall'ambiente esterno ed è rifornita mediante un dispositivo a buffer per il carico dei pezzi grezzi e lo scarico dei semilavorati.

Figura 19: Macchina SPI

Le operazioni che compongono una lavorazione sono le seguenti:

1. i pezzi vengono caricati sul buffer di carico e scarico pezzi dal robot manipolatore;
2. le porte della Macchina si aprono;
3. il buffer di carico e scarico pezzi ruota di 180° e si alza;
4. la tavola tiltante e le fixture fissate sui due piatti ruotano fino ad arrivare ad agganciare i pezzi posti sul buffer di carico e scarico pezzi;
5. le fixture si muovono fino a portare i pezzi nella corretta posizione di fronte al mandrino;
6. il buffer di carico e scarico pezzi scende e le porte si chiudono.

Dopo la lavorazione i pezzi vengono scaricati con la seguente sequenza di operazioni:

7. le porte della Macchina si aprono;
8. il buffer di carico e scarico pezzi si alza;
9. la tavola tiltante e le fixture fissate sui due piatti ruotano fino ad arrivare a lasciare i pezzi sul buffer di carico e scarico pezzi;
10. le fixture si muovono fino a riportarsi nella corretta posizione di fronte al mandrino;
11. il buffer di carico e scarico pezzi ruota di 180° e si abbassa;
12. le porte si chiudono;
13. i pezzi vengono prelevati dal buffer di carico e scarico pezzi dal robot manipolatore.

È possibile riconoscere 7 sotto-parti principali che compongono la Macchina SPI

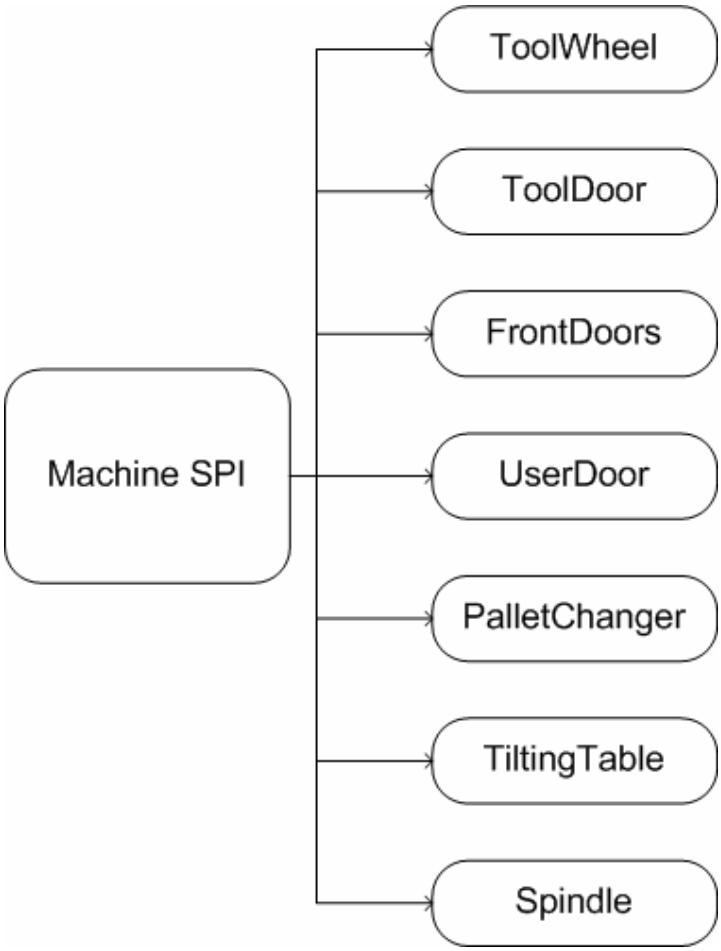

Figura 20: Componenti della Macchina SPI

4.2.1 Scomposizione modulare della Macchina SPI

Questo paragrafo mostra la scomposizione gerarchica funzionale della Macchina SPI, in particolare, i componenti possono essere classificati in tre livelli:

- **Unit** → radice dell'albero, corrisponde alla Macchina SPI completa;
- **Device** → dispositivi utilizzati dalla Macchina per lo svolgimento delle lavorazioni;
- **Equipment** → foglie dell'albero ed elementi costitutivi dei device.

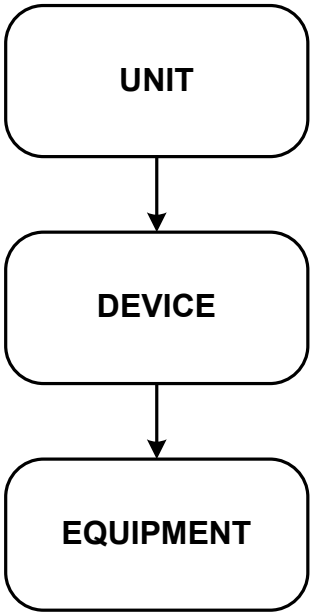

Figura 21: Gerarchia funzionale dei componenti della Macchina SPI

Nel seguito del capitolo si presentano e descrivono le variabili che caratterizzano i device della Macchina SPI.

4.3 Revisione delle schede tecniche

Le schede tecniche della Macchina SPI sono un documento che descrive ogni componente funzionale in dettaglio, riportando le variabili che lo caratterizzano, le funzioni che lo descrivono, i diagrammi SFC di tali funzioni e tutte le condizioni logiche necessarie ad implementare il controllo.

Nel corso dell'elaborato è stato svolto un lavoro di completa revisione dell'intero documento, si sono rielaborate 26 funzioni e i relativi diagrammi SFC.

La prima parte del lavoro si è concentrata sulla correzione concettuale dei diagrammi SFC con le seguenti azioni:

- eliminazione degli stati inutili o ridondanti;
- eliminazione e/o correzione di condizioni di transizione errate;
- correzione dei tempi di attesa.

Terminato il lavoro concettuale sono state riprodotte tutte le immagini grafiche degli SFC.

La seconda parte della revisione ha permesso di esprimere tutte le funzioni in un linguaggio formale simile al linguaggio di programmazione C.

Oltre ad avere una funzione informativa, le schede tecniche sono state utilizzate per implementare manualmente il controllo della Macchina SPI in Orchestra e chiudere la retroazione col simulatore con la tecnica Hardware in The Loop allo scopo di verificare che le informazioni sulle logiche di controllo riportate su di esse fossero corrette e per una fase iniziale di test del simulatore stesso.

4.3.1 Device 1: ToolWheel

La ToolWheel permette il carico e lo scarico dei vari utensili utilizzati dal mandrino. Si utilizzano due funzioni, la prima ***Move*** permette di ruotare la ruota nella posizione desiderata in base allo strumento da utilizzare, la seconda ***IsTop*** permette di sapere se la ruota è nella posizione TOP (in alto) in modo da poter chiudere la porta per iniziare la lavorazione. La ruota è guidata da un attuatore elettrico verticale.
Il livello gerarchico inferiore di questo device è l'equipment ***WheelCartesianAxis.***

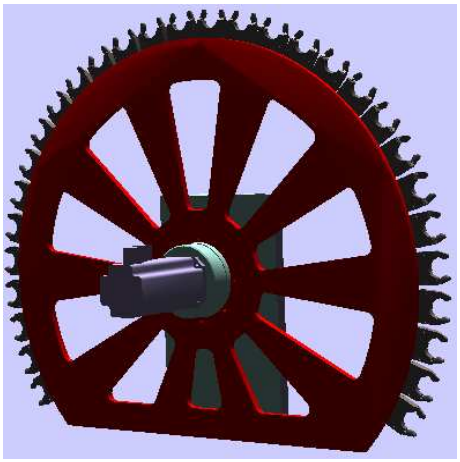

Figura 22: Toolwheel

Lista segnali:

WCOMMAND: output di tipo double, descrive la posizione angolare.
***WSPEED*:** output di tipo double, descrive la velocità di rotazione.
***VCOMMAND*:** output di tipo double, descrive la posizione verticale.
***VSPEED*:** output di tipo double, descrive la velocità di traslazione.
***WPOSITION*:** input di tipo double, descrive la corrente posizione angolare.
***VPOSITION*:** input di tipo double, descrive la corrente posizione verticale.
SQ369: input di tipo double, rappresenta lo stato del sensore (ruota in standby).
***SQ370*:** input di tipo double, rappresenta lo stato del sensore (ruota in movimento).

4.3.2 Device 2: ToolDoor

Il compito di questa porta è di isolare la zona di lavorazione della Macchina SPI dalla ruota porta utensili. Anche in questo caso si utilizzano due funzioni, ***Open*** permette di aprire la porta. Restituisce il valore 1 quando la porta è aperta, 0 altrimenti. La seconda funzione ***Close*** permette di chiudere la porta, restituisce il valore 1 quando la porta è chiusa, 0 altrimenti.

Il livello gerarchico inferiore di questo device è l'equipment ***UpDoor.***

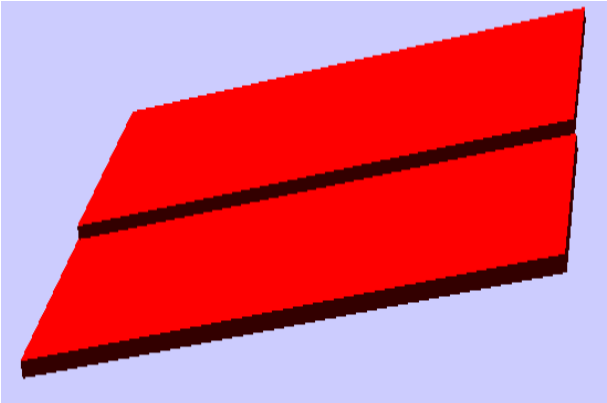

Figura 23: ToolDoor

Lista segnali:

KM53A: variabile booleana di output, rappresenta l'elettrovalvola per l'apertura porta.
KM53B*:* variabile booleana di output, rappresenta l'elettrovalvola per la chiusura porta.
RT1: variabile booleana di input, rappresenta il surriscaldamento delle elettrovalvole.
SQ72: variabile booleana di input, rappresenta il sensore di porta superiore aperta.
SQ73: variabile booleana di input, rappresenta il sensore di porta superiore chiusura.
SQ171: variabile booleana di input, rappresenta il sensore di movimento (strobe).
SQ172: variabile booleana di input, rappresenta il sensore di movimento (strobe).

4.3.3 Device 3: FrontDoors

Questo dispositivo consiste in due porte che separano l'area di lavorazione della Macchina SPI dallo scambiatore pezzi (Pallet Changer). Si utilizzano due funzioni per la porta di sinistra, e due per la porta di destra, ovviamente uguali. La prima ***Open*** per aprire la porta, la seconda ***Close*** per chiuderla. A livello superiore le funzioni di apertura e chiusura porte vengono chiamate in simultanea.

Il livello gerarchico inferiore di questo device sono gli equipments ***RightFrontDoor*** e ***LeftFrontDoor.***

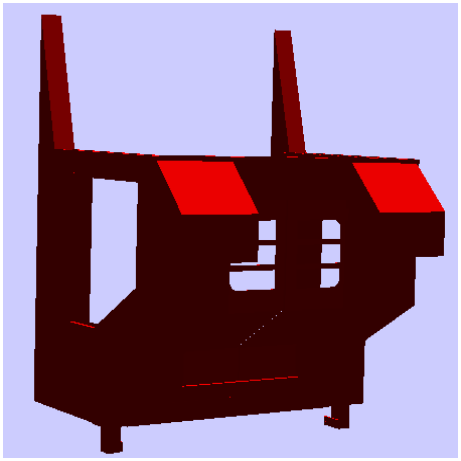

Figura 24: FrontDoors

Lista segnali per la porta di sinistra:

KM82A: variabile booleana di output, rappresenta l'elettrovalvola per l'apertura porta di sinistra.
KM82B: variabile booleana di output, rappresenta l'elettrovalvola per la chiusura porta di sinistra.
SQ20: variabile booleana di input, rappresenta il sensore di porta sinistra aperta.
SQ21: variabile booleana di input, rappresenta il sensore di porta sinistra chiusa.
SQ24: variabile booleana di input, rappresenta il sensore di movimento (strobe).
SQ25: variabile booleana di input, rappresenta il sensore di movimento (strobe).

Lista segnali per la porta di destra:

KM83A: variabile booleana di output, rappresenta l'elettrovalvola per l'apertura porta di destra.
KM83B: variabile booleana di output, rappresenta l'elettrovalvola per la chiusura porta di destra.
SQ202: variabile booleana di input, rappresenta il sensore di porta destra aperta.

SQ212: variabile booleana di input, rappresenta il sensore di porta destra chiusa.
SQ242: variabile booleana di input, rappresenta il sensore di movimento (strobe).
SQ252: variabile booleana di input, rappresenta il sensore di movimento (strobe).

4.3.4 Device 4: UserDoor

Questa porta separa l'area di lavoro della Macchina SPI dall'ambiente esterno. In fase di simulazione non è stata considerata ma si sono comunque revisionate le schede tecniche. Si utilizzano due funzioni, ***Lock*** che permette di bloccare la porta aperta e restituisce 1 mentre la funzione ***Unlock*** permette di sbloccare la porta aperta.

Il livello gerarchico inferiore di questo device è l'equipment ***SideDoor.***

Lista segnali:

YV22: variabile booleana, rappresenta l'elettrovalvola per lo sblocco della porta laterale.
SQ19: variabile booleana, rappresenta il sensore di blocco della porta laterale.
MPORT: variabile booleana, rappresenta il sensore di chiusura della porta laterale.

4.3.5 Device 5: Pallet Changer

Lo scambiatore pezzi rappresenta l'unica parte reale della Macchina che verrà utilizzata nella fase di simulazione Mixed Reality. Il sistema Pallet Changer ha la funzione di buffer locale per lo scarico di parti lavorate e per il carico di parti grezze. Il device può essere suddiviso in tre equipments, ***ZRotator, ZTranslator, LockPieces.***

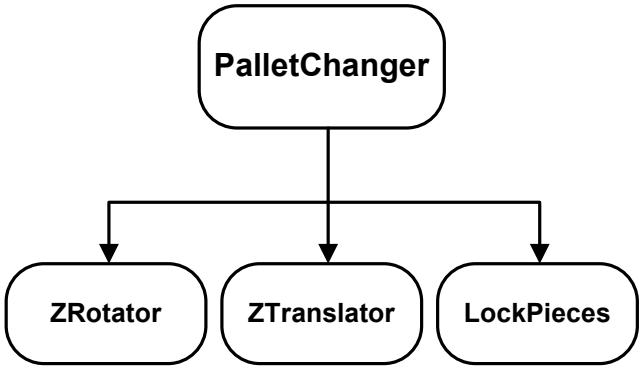

Figura 25: Schema componenti Pallet Changer

- Translator: componente che permette la traslazione lungo l'asse verticale della tavola del LockPieces (o LockParts). È costituito da una base ed un pistone comandato da due elettrovalvole;
- Rotator: componente che permette la rotazione della tavola del LockPieces. È costituito da un motore che permette la rotazione nei due sensi;
- LockParts: tavola sulla quale sono presenti quattro piatti per il bloccaggio dei pezzi. Ciascun piatto dispone di tre pinze per il bloccaggio, comandate ognuna da una elettrovalvola.

Figura 26: Pallet Changer reale

4.3.5.1 Translator

Il Translator è composto da due parti. La prima è la base, che rappresenta anche l'appoggio del Pallet Changer, la seconda è la cassa dove alloggia lo stelo del pistone. Al pistone è associato un grado di libertà di traslazione. Per gestire il movimento del pistone, il Translator è dotato di due elettrovalvole e due sensori che ne determinano la posizione. Le due funzioni che lo caratterizzano sono ***MoveUp*** che implementa la traslazione verso l'alto e ***MoveDown*** che implementa la traslazione di discesa.

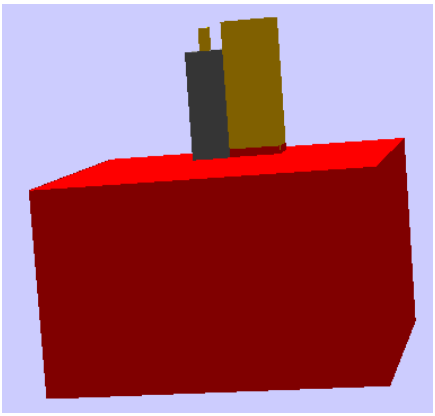

Figura 27: Translator

Lista segnali:

YV577: variabile booleana di output, rappresenta l'elettrovalvola per la discesa dello scambiatore pezzi.
YV578: variabile booleana di output, rappresenta l'elettrovalvola per la salita dello scambiatore pezzi.
SQ271: variabile booleana di input, rappresenta il sensore per lo scambiatore in posizione alta.
SQ272: variabile booleana di input, rappresenta il sensore per lo scambiatore in posizione bassa.

Funzioni:

- ***MoveUp***

```
1.  YV577 = 0; YV578 = 1;
2.  while(! (SQ272 == 0 && SQ271 == 1))
3.  if (time >10s)
4.          return false;
5.  return true;
```

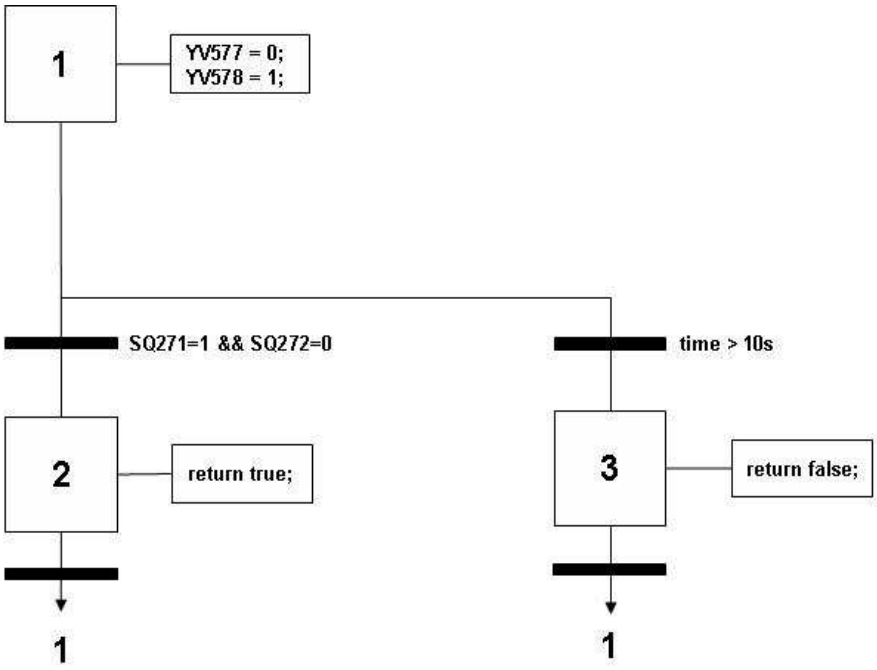

Figura 28: Diagramma SFC della funzione MoveUp

- ***MoveDown***

```
1.    YV577 = 1 && YV578 = 0;
2.    while(!(SQ272 == 1 && SQ271 == 0));
3.        if (time >10s)
4.            return false;
5.    return true;
```

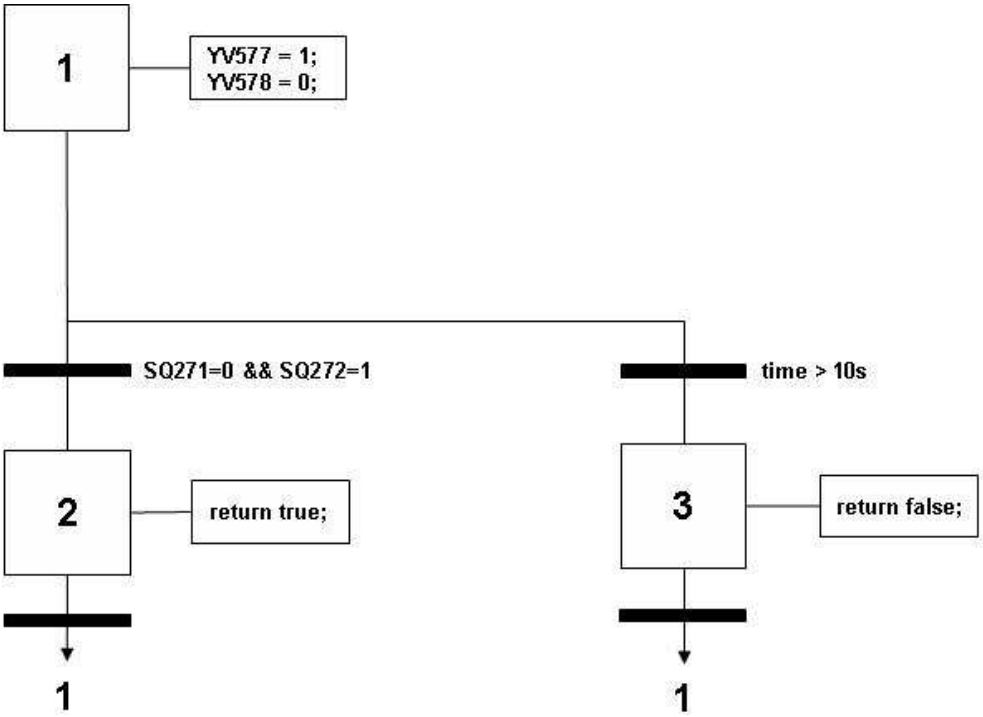

Figura 29: Diagramma SFC della funzione MoveDown

4.3.5.2 Rotator

Il Rotator permette la rotazione della tavola del Pallet Changer di 180 gradi. È composto da due funzioni, ***RotateSide1ToMachine*** e ***RotateSide2ToMachine*** che permetto alla tavola su cui vengono posizionati i pezzi di ruotare verso la Macchina per il carico e lo scarico.

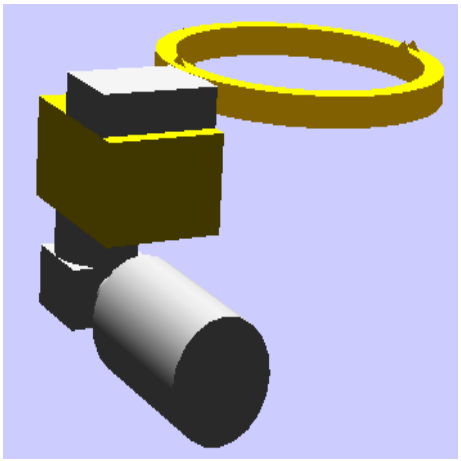

Figura 30: Rotator

Lista segnali:

KM99A: variabile booleana di output, rappresenta l'elettrovalvola per la rotazione del Side 1.
KM99B: variabile booleana di output, rappresenta l'elettrovalvola per la rotazione del Side 2.
SQ269: variabile booleana di input, rappresenta il sensore di posizione per il Side 1.
SQ270: variabile booleana di input, rappresenta il sensore di posizione per il Side 2.
SQ277: variabile booleana di input, rappresenta il sensore di movimento (strobe).
SQ278: variabile booleana di input, rappresenta il sensore di movimento (strobe).

Funzioni:

- ***RotateSide1ToMachine***

```
1.  if (SQ270 == 0 && SQ269 == 1 && SQ277 == 1 && SQ278 == 1 && RT1 == 0)
2.          return true;
3.  else if (RT1==1)
4.          return false;
5.  else if((SQ270 == 1 && SQ269 == 0 && SQ277 == 1 && SQ278 == 1 && RT1 == 0))
6.  {       KM99B = 0; KM99A = 1;
7.          while (!(SQ277==0&&SQ278==0)
8.          {
9.                  if  (RT1 == 1)
10.                 {
11.                         KM99A=0; KM99B=0;
12.                         return false;
13.                 }
14.                 if  (time>10s)
15.                 {
16.                         KM99A=0; KM99B=0;
17.                         return false;
18.         }
19.    while (!(SQ270 == 0 && SQ269 == 1 && SQ277 == 1 && SQ278 == 1 && RT1 == 0))
20.         {
21.                 if  (RT1 == 1)
22.                  {
23.                         KM99A=0; KM99B=0;
24.                         return false;
25.                 }
26.                 if  (time>10s)
27.                 {    KM99A=0; KM99B=0;
28.                         return false;
29.                 }
30.         }
31.         KM99A=0; KM99B=0;
32.         return true;
33. }
```

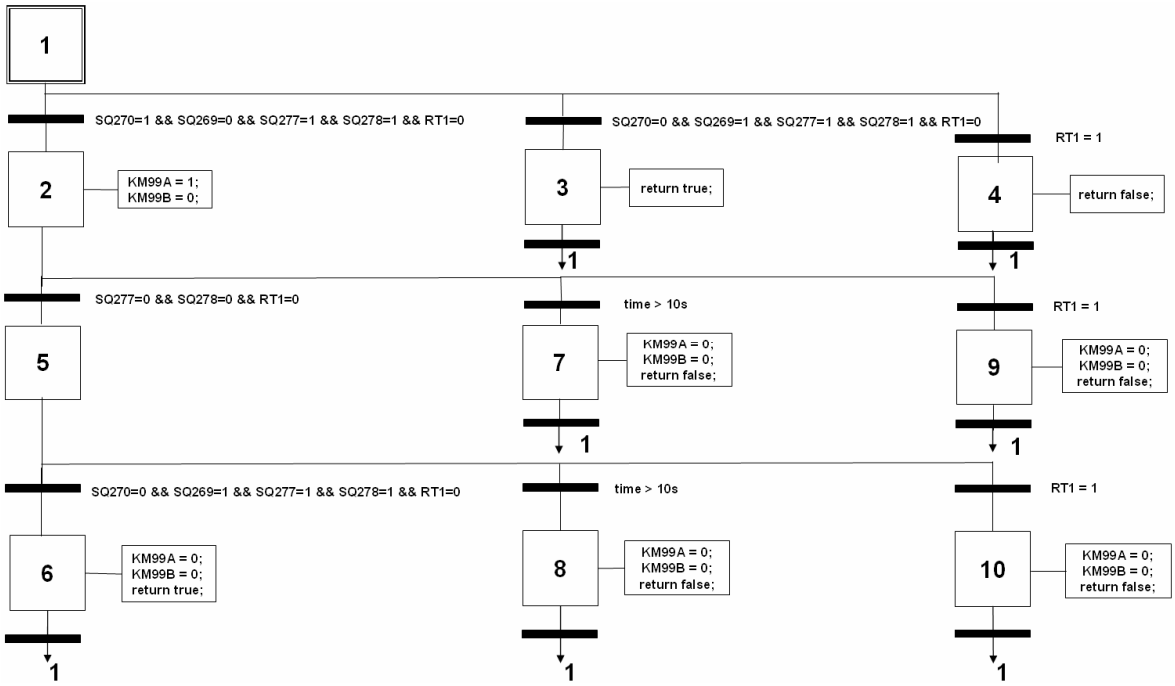

Figura 31: Diagramma SFC della funzione RotateSide1ToMachine

- ***RotateSide2ToMachine***

```
1. if (SQ270 == 1 && SQ269 == 0 && SQ277 == 1 && SQ278 == 1 && RT1 == 0)
2.         return true;
3. else if (RT1==1)
4.         return false;
5. else if((SQ270 == 0 && SQ269 == 1 && SQ277 == 1 && SQ278 == 1 && RT1 == 0))
6. {       KM99B = 1; KM99A = 0;
7.         while (!(SQ277==0&&SQ278==0)
8.         {
9.                 if  (RT1 == 1)
10.                {
11.                        KM99A=0; KM99B=0;
12.                        return false;
13.                }
14.                if  (time>10s)
15.                {
16.                       KM99A=0; KM99B=0;
17.                        return false;
18.        }
19.  while (!(SQ270 == 1 && SQ269 == 0 && SQ277 == 1 && SQ278 == 1 && RT1 == 0))
20.        {
21.                if  (RT1 == 1)
22.                {
23.                        KM99A=0; KM99B=0;
24.                        return false;
25.                }
26.                if  (time>10s)
27.                {
28.                       KM99A=0; KM99B=0;
29.                        return false;
30.                }
31.        }
32.        KM99A=0; KM99B=0;
33.        return true;
34. }
```

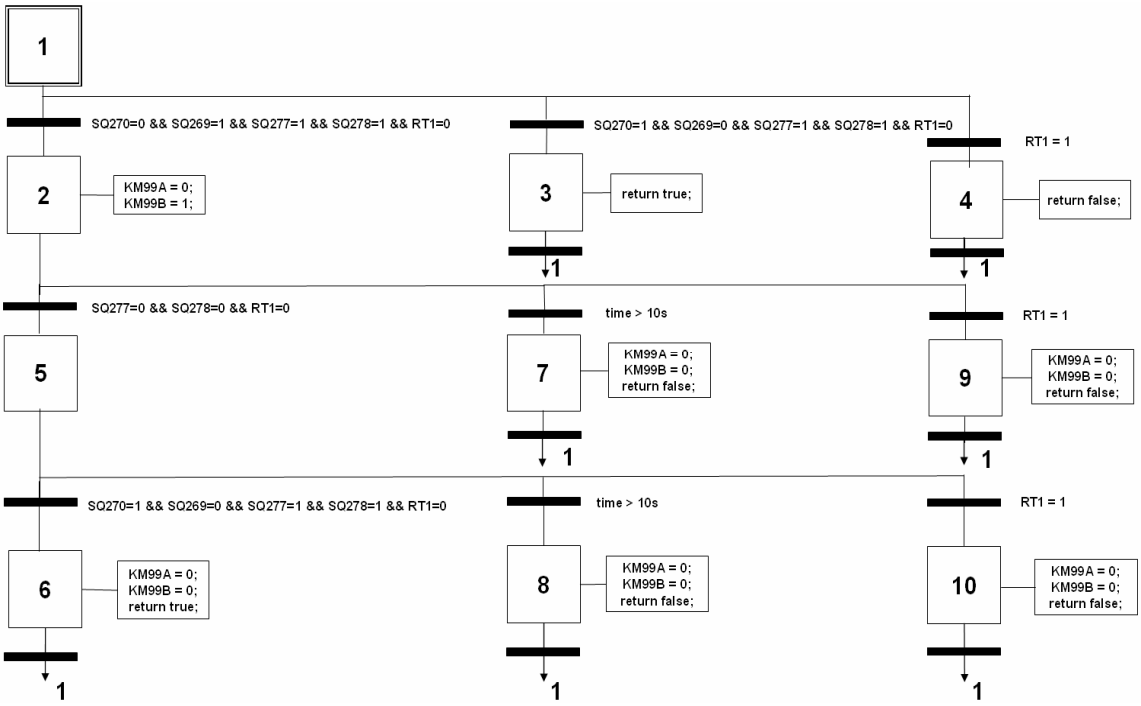

Figura 32: Diagramma SFC della funzione RotateSide1ToMachine

4.3.5.3 LockParts

Il LockParts è formato da una tavola contenente quattro piatti. I piatti sono controllati a coppie dato che la Macchina SPI processa due pezzi alla volta. Ogni piatto è composto da tre pistoni che permettono di bloccare saldamente il pezzo alla tavola. Per questo componente sono previste 4 funzioni:

- 2 di blocco, ***LockPartSide1*** e ***LockPartSide2***: bloccano rispettivamente i due piatti del Side1 (1B-1C) e i due piatti del Side2 (2B-2C);
- 2 di sblocco, ***UnlockSide1*** e ***UnlockSide2***: coma sopra ma in questo caso sbloccano i piati corrispondenti.

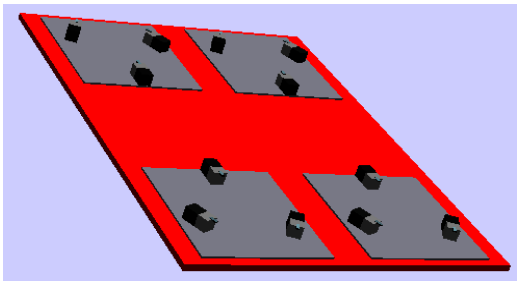

Figura 33 : LockParts

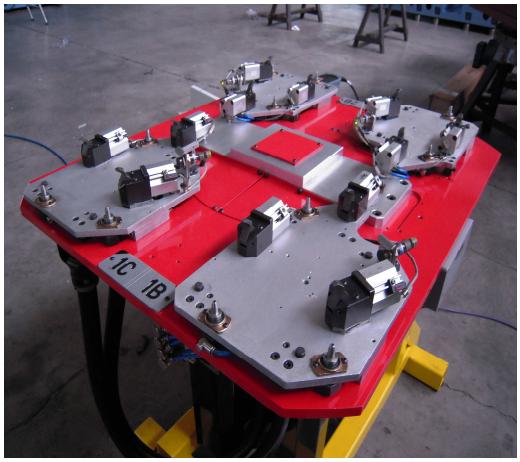

Figura 34: LockPieces reale e identificazione dei piatti

Lista segnali LockParts1:

YV583: variabile booleana di output, rappresenta l'elettrovalvola per il blocco dei pezzi del Side 1.
YV584: variabile booleana di output, rappresenta l'elettrovalvola per lo sblocco dei pezzi del Side 1.
SQ484: variabile booleana di input, rappresenta il sensore per il blocco (valore a 1) o sblocco (valore a 0) del pistone 1 del piatto 1B.
SQ485: variabile booleana di input, rappresenta il sensore per il blocco (valore a 1) o sblocco (valore a 0) del pistone 2 del piatto 1B.
SQ486: variabile booleana di input, rappresenta il sensore per il blocco (valore a 1) o sblocco (valore a 0) del pistone 3 del piatto 1B.
SQ487: variabile booleana di input, rappresenta il sensore per il blocco (valore a 1) o sblocco (valore a 0) del pistone 1 del piatto 1C.
SQ488: variabile booleana di input, rappresenta il sensore per il blocco (valore a 1) o sblocco (valore a 0) del pistone 2 del piatto 1C.
SQ489: variabile booleana di input, rappresenta il sensore per il blocco (valore a 1) o sblocco (valore a 0) del pistone 3 del piatto 1C.
SP289: variabile booleana di input, rappresenta il sensore di sblocco per la valvola a pressione del Side 1
SQ273: variabile booleana di input, rappresenta il sensore di presenza pezzo per il Side 1 (Piatto 1C)
SQ275: variabile booleana di input, rappresenta il sensore di presenza pezzo per il Side 1 (Piatto 1B)

Funzioni:

- ***LockPartSide1***

```
1. YV583 = 1 && YV584 = 0;
2.         while(!(SQ484 == 0 && SQ485 == 0 && SQ486==0 && SQ487 == 0
   && SQ488 == 0 && SQ489==0 && SP289==0));
3.                 if (time >10s)
4.                         return false;
5.         return true;
```

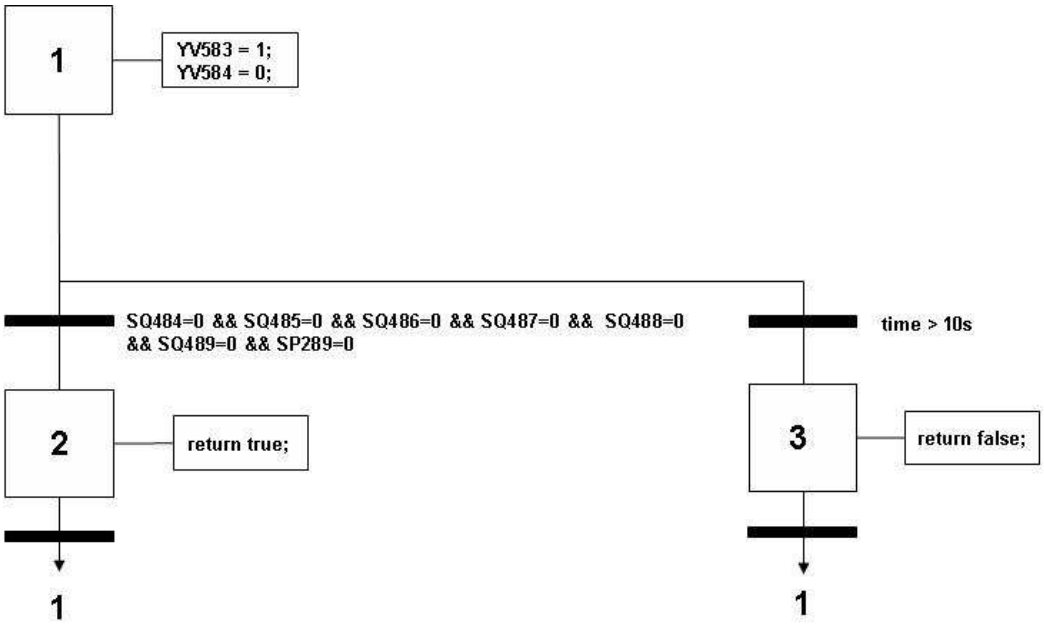

Figura 35: Diagramma SFC della funzione LockPartSide1

- ***UnlockSide1***

```
1. YV583 = 0 && YV584 = 1;
2.         while(!(SQ484 == 1 && SQ485 == 1 && SQ486==1 && SQ487 == 1
   && SQ488 == 1 && SQ489==1 && SP289==1));
3.                 if (time >10s)
4.                         return false;
5.         return true;
```

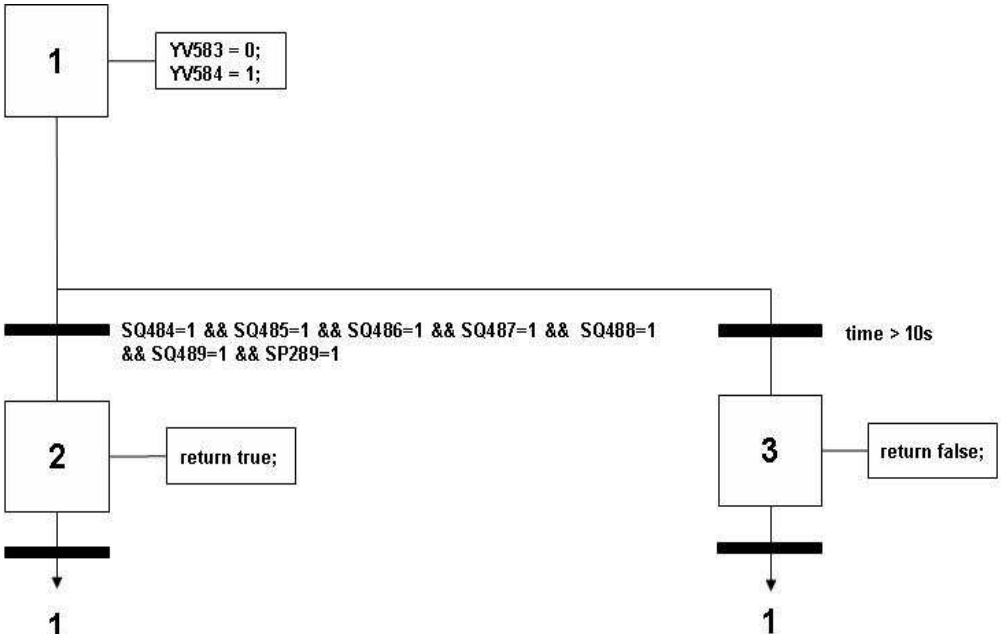

Figura 36: Diagramma SFC della funzione UnlockSide1

Lista segnali LockParts2:

YV585: variabile booleana di output, rappresenta l'elettrovalvola per il blocco dei pezzi del Side 2.
YV586: variabile booleana di output, rappresenta l'elettrovalvola per lo sblocco dei pezzi del Side 2.
SQ490: variabile booleana di input, rappresenta il sensore per il blocco (valore a 1) o sblocco (valore a 0) del pistone 1 del piatto 2B.
SQ491: variabile booleana di input, rappresenta il sensore per il blocco (valore a 1) o sblocco (valore a 0) del pistone 2 del piatto 2B.
SQ492: variabile booleana di input, rappresenta il sensore per il blocco (valore a 1) o sblocco (valore a 0) del pistone 3 del piatto 2B.
SQ493: variabile booleana di input, rappresenta il sensore per il blocco (valore a 1) o sblocco (valore a 0) del pistone 1 del piatto 2C.
SQ494: variabile booleana di input, rappresenta il sensore per il blocco (valore a 1) o sblocco (valore a 0) del pistone 2 del piatto 2C.
SQ495: variabile booleana di input, rappresenta il sensore per il blocco (valore a 1) o sblocco (valore a 0) del pistone 3 del piatto 2C.
SP290: variabile booleana di input, rappresenta il sensore di sblocco per la valvola a pressione del Side 2.
SQ274: variabile booleana di input, rappresenta il sensore di presenza pezzo per il Side 2 (Piatto 2C).
SQ276: variabile booleana di input, rappresenta il sensore di presenza pezzo per il Side 2 (Piatto 2B).

Si omettono le funzioni e i diagrammi SFC del LockParts 2 in quanto uguali a quelli sopra presentati.

4.3.6 Device 6: TiltingTable

Questo componente aggiunge gli ultimi due gradi di libertà per le lavorazioni. Comprende 5 funzioni, ***Move*** per ruotare la tavola, ***LockParts*** e ***UnlockParts*** funzioni che rispettivamente permettono di bloccare e sbloccare i pezzi sulle pinze mentre le ultime 2 funzioni, ***RotateFixture1*** e ***RotateFixture2*** permettono la rotazione delle fixture. Gli equipments di questo device sono ***Tilting*** e ***Fixture.***

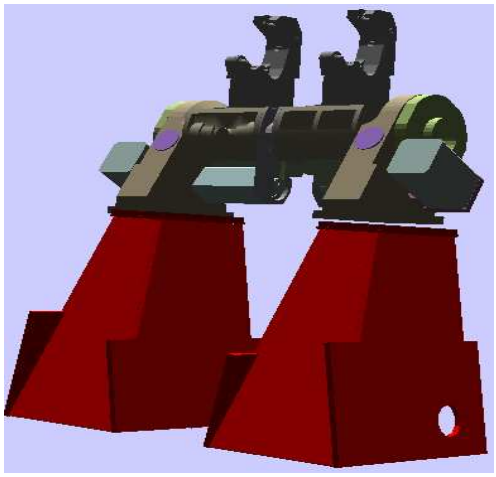

Figura 37: TiltingTable

Lista segnali Tilting:

YV67: variabile booleana di output, rappresenta l'elettrovalvola per lo sblocco della tavola.
ACOMMAND: variabile di output di tipo double, rappresenta la posizione angolare della tavola.
ASPEED: variabile di output di tipo double, rappresenta la velocità di rotazione.
APOSITION: variabile di input di tipo double, rappresenta l'attuale posizione angolare.
SP67: variabile booleana di input, rappresenta il sensore di pressione per tavola sbloccata.
SP68: variabile booleana di input, rappresenta il sensore di pressione per tavola bloccata.

***Lista segnali LeftFixture*:**

YV9: variabile booleana di output, rappresenta l'elettrovalvola per lo sblocco della fixture.
BCOMMAND: variabile di tipo double di output, rappresenta la posizione angolare della fixture.
BSPEED: variabile di tipo double di output, rappresenta la velocità di rotazione.
BPOSITION: variabile di input di tipo double, rappresenta l'attuale posizione angolare.
SP9: variabile booleana di input, rappresenta il sensore di pressione per fixture sbloccata.
SP10: variabile booleana di input, rappresenta il sensore di pressione per fixture bloccata.

***Lista segnali RightFixture*:**

YV92: variabile booleana di output, rappresenta l'elettrovalvola per lo sblocco della fixture.
CCOMMAND: variabile di tipo double di output, rappresenta la posizione angolare della fixture.
CSPEED: variabile di tipo double di output, rappresenta la velocità di rotazione.
CPOSITION: variabile di input di tipo double, rappresenta l'attuale posizione angolare.
SP92: variabile booleana di input, rappresenta il sensore di pressione per fixture sbloccata.
SP102: variabile booleana di input, rappresenta il sensore di pressione per fixture bloccata.

***Lista segnali LockWorkParts Fixture* :**

YV133: variabile booleana di output, rappresenta l'elettrovalvola per il blocco dei pezzi sulle fixture.
YV134: variabile booleana di output, rappresenta l'elettrovalvola per lo sblocco dei pezzi sulle fixture.
SP78: variabile booleana di input, rappresenta il sensore pezzi blocchi sulle fixture.
SP79: variabile booleana di input, rappresenta il sensore pezzi blocchi sulle fixture.

4.2.7 Device 7: Spindle

L'ultimo componente analizzato è il mandrino. È rappresentato da 3 funzioni, la prima ***Move*** per descrivere la traiettoria del mandrino, ***Lock*** e ***Unlock*** per rappresentare il blocco e sblocco utensile. Gli equipments sono: ***SpindleCartesianAxis*** e ***LockTool.***

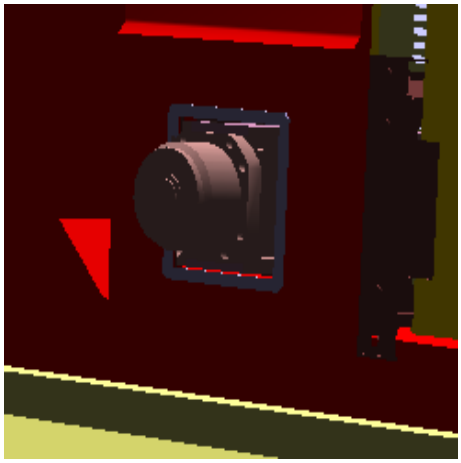

Figura 38: Spinale

Lista segnali SpindleAxes:

XCOMMAND: variabile di tipo double di output, rappresenta la posizione sull'asse x da raggiungere del mandrino.
XSPEED: variabile di tipo double di output, rappresenta la velocità sull'asse x.
XPOSITION: variabile di input di tipo double, rappresenta l'attuale posizione sull'asse x.
SQ26: variabile booleana di input, rappresenta il sensore di posizione di riposo.
SQXS: variabile booleana di input, rappresenta il sensore di left overtravel per l'asse x.
SQXD: variabile booleana di input,rappresenta il sensore di right overtravel per l'asse x.
USXS: variabile booleana di input, rappresenta il sensore sinistro di protezione motore per l'asse x.
USXD: variabile booleana di input, rappresenta il sensore destro di protezione motore per l'asse x.

YCOMMAND: variabile di tipo double di output, rappresenta la posizione sull'asse y da raggiungere del mandrino.
YSPEED: variabile di tipo double di output, rappresenta la velocità sull'asse y.
YPOSITION: variabile di input di tipo double, rappresenta l'attuale posizione sull'asse y.
SQ27: variabile booleana di input, rappresenta il sensore di posizione di riposo.
SQYS: variabile booleana di input, rappresenta il sensore di left overtravel per l'asse y.
SQYD: variabile booleana di input,rappresenta il sensore di right overtravel per l'asse y.

USYS: variabile booleana di input, rappresenta il sensore sinistro di protezione motore per l'asse y.
USYD: variabile booleana di input, rappresenta il sensore destro di protezione motore per l'asse y.

ZCOMMAND: variabile di tipo double di output, rappresenta la posizione sull'asse z da raggiungere del mandrino.
ZSPEED: variabile di tipo double di output, rappresenta la velocità sull'asse z.
ZPOSITION: variabile di input di tipo double, rappresenta l'attuale posizione sull'asse z.
SQ28: variabile booleana di input, rappresenta il sensore di posizione di riposo.
SQZS: variabile booleana di input, rappresenta il sensore di left overtravel per l'asse z.
SQZD: variabile booleana di input,rappresenta il sensore di right overtravel per l'asse z.
USZS: variabile booleana di input, rappresenta il sensore sinistro di protezione motore per l'asse z.
USZD: variabile booleana di input, rappresenta il sensore destro di protezione motore per l'asse z.

Lista segnali LockTool:

YV4: variabile booleana di output, rappresenta l'elettrovalvola per lo sblocco pneumatico della fixture.
YV66: variabile booleana di output, rappresenta l'elettrovalvola per lo sblocco idraulico della fixture.
SP32: variabile booleana di input, rappresenta il sensore di pressione troppo bassa per sbloccare il mandrino.
SP33: variabile booleana di input, rappresenta il sensore di pressione minima per bloccare/sbloccare il mandrino.
SP34: variabile booleana di input, rappresenta il sensore di pressione massima per bloccare/sbloccare il mandrino.
SQ3: variabile booleana di input, rappresenta il sensore di attrezzo bloccato sul mandrino.
SQ4: variabile booleana di input, rappresenta il sensore di attrezzo sbloccato.

5

ORGANIZZAZIONE SPERIMENTALE

5.1 Implementazione del Monitorng 3D

Il monitoraggio riproduce fedelmente a video lo stato della Macchina SPI e del Pallet Changer durante le varie fasi di lavorazione. La realizzazione del Monitoring 3D è stato il primo passo della generazione dell'architettura per la prova finale. Tale implementazione ha richiesto di prendere dimestichezza con l'utilizzo dell'ambiente di simulazione SIMBA, un software di simulazione mai utilizzato in precedenza. Per ognuna delle 7 sottoparti della Macchina SPI è stato dunque creato un componente simulato come spiegato nel seguito.

Per ottenere l'interfaccia 3D del sistema di monitoraggio si sono quindi utilizzate le stesse grafiche che compongono la Macchina SPI simulata in quanto il Monitoring riprodurrà esattamente la stessa Macchina. Come esempio si riporta un solo componente, la ToolDoor.

- Creazione del Behavior Block (BB): scelto il nome, che sarà sempre il nome del componente con il suffisso Monitoring si abbina il grado di libertà corrispondente. Al BB vengono poi collegati i gate di input; nel caso specifico i sensori SQ73 e SQ72. Tutti i componenti avranno solo dei segnali in ingresso in quanto il Monitoring è una rappresentazione e non interagisce con l'esterno.

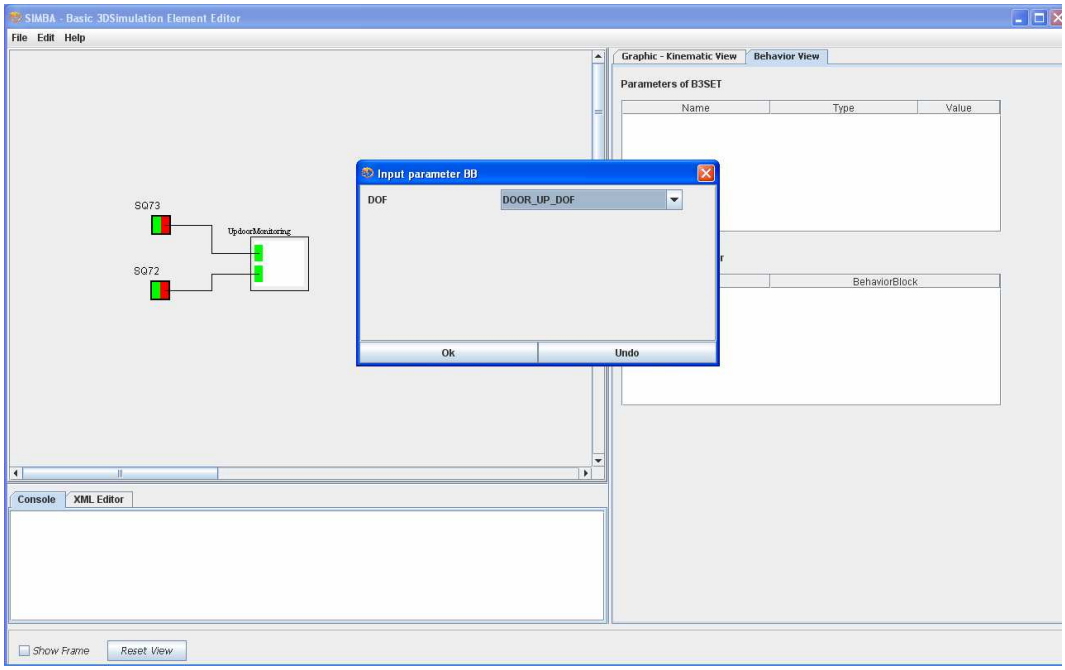

Figura 39: Behavior View Monitorig

- Creazione del Communcation BB: la creazione del BB necessita dell'implementazione del canale di comunicazione. Si allega un pezzo del codice necessario. Ovviamente tutti i segnali sono in lettura.

```
public class UpdoorCommMonitoring extends CommunicationBlock{

OutputPort SQ73 = new OutputPort("SQ73",Boolean.class);
OutputPort SQ72 = new OutputPort("SQ72",Boolean.class);

CommPoint SQ72comm =
newCommPoint8Bit("OrchestraSPISQ72In",24,CommPoint.READ,CommPoint8Bit.BOOL);
CommPoint SQ73comm = new
CommPoint8Bit("OrchestraSPISQ73In",25,CommPoint.READ,CommPoint8Bit.BOOL);

   public UpdoorCommMonitoring(String behaviorName, B3SET myB3SET)
   {
         super(behaviorName, myB3SET);

      outputPorts.put(SQ72.getPortName(), SQ72);
      outputPorts.put(SQ73.getPortName(), SQ73);
   }

         @Override
         public void finalizeBehavior() {
                  // TODO Auto-generated method stub
                  myCommunication.removePoint(SQ72comm);
                  myCommunication.removePoint(SQ73comm);
```

```
}

@Override
public void initializeBehavior() {
	try {
	myCommunication.insertNewPoint(SQ72comm);
	myCommunication.insertNewPoint(SQ73comm);

	} catch (PointPositionDuplicatedException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	}
}

@Override
public void executionBehavior(long time) {

	if(SQ72comm.getData()[0]==1)
		SQ72.writeValue(true);
	else
		SQ72.writeValue(false);

	if(SQ73comm.getData()[0]==1)
		SQ73.writeValue(true);
	else
	SQ73.writeValue(false);
	}}
```

5.2 Implementazione dell'interfaccia di fault

Subito dopo il monitoraggio è stata implementata una semplice interfaccia che comunica con un task del PLC interposta prima del controllo in modo da permettere il mascheramento con un possibile stato di errore di alcuni segnali del Pallet Changer allo scopo di testare la diagnostica. Con questo artificio, evitando di creare collegamenti fisici hardware, durante la fase di simulazione è stato possibile verificare l'efficacia dei diagnoser e l'isolamento dei guasti.

La parte di diagnosi è stata sviluppata in altri lavori di ricerca paralleli a quello qui presentato e prevede la costruzione di un automa diagnostico mediante la metodologia TiDiaM per ciascun dispositivo del Pallet Changer. Nel dettaglio sono stati implementati 4 diagnoser diversi uno per ciascuna della parti dello scambiatore :

1. Diagnoser Rotator;
2. Diagnoser Translator;
3. Diagnoser Lock Side 1;
4. Diagnoser Lock Side 2.

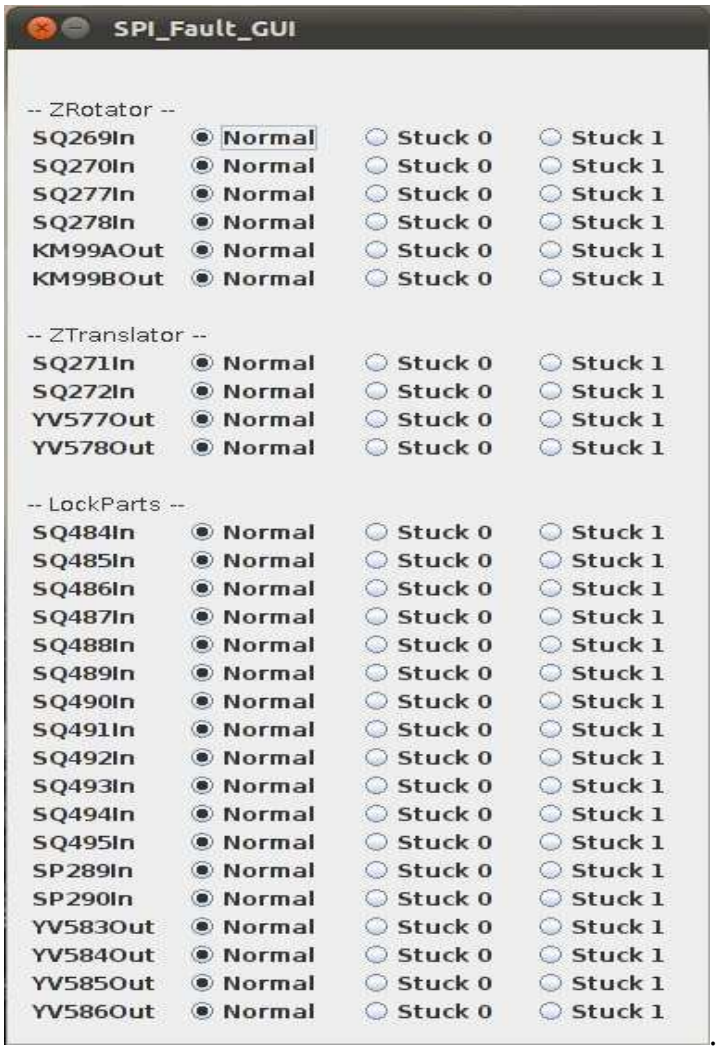

Figura 40: Interfaccia dei fault

Le variabili di stuck rappresentano lo stato di guasto. Trattandosi di sole variabili booleane, è possibile “simulare” un guasto forzando il segnale di una variabile o a 0 oppure a 1. Scopo del diagnoser è di riconoscere il malfunzionamento osservando i cambiamenti dei segnali provenienti dal sistema controllato.

5.3 Modellistica e generazione delle logiche di controllo

Questa è stata sicuramente la parte più cospicua del lavoro. La fase di modellistica e generazione delle logiche di controllo è stata condotta secondo la metodologia e gli strumenti messi a disposizione dal progetto MEDEIA. Il testing è stato quindi condotto non solo sui modelli e sul codice di controllo della Macchina SPI ma anche sull'intero ambiente di progettazione e sviluppo MDEF. Testare le logiche di controllo di un modello richiede molto tempo e tantissima pazienza. Il controllo generato da MEDEIA è stato in fine testato tramite la simulazione Mixed Reality dell'intera cella SPI.

5.3.1 Controllo

In questo paragrafo verrà presentato il procedimento per la generazione del controllo della cella di lavorazione. Il primo passo è stato concettuale, creare cioè il modello del sistema secondo la vista UML proposta da MEDEIA. Tale compito è stato svolto in collaborazione con altri lavori di tesi tramite i quali è stato sviluppato un modello del controllo a Timed State Chart UML utilizzando lo strumento software StarUML. Come esempio si mostra il diagramma del controllo per l'AC Translator del Pallet Changer.

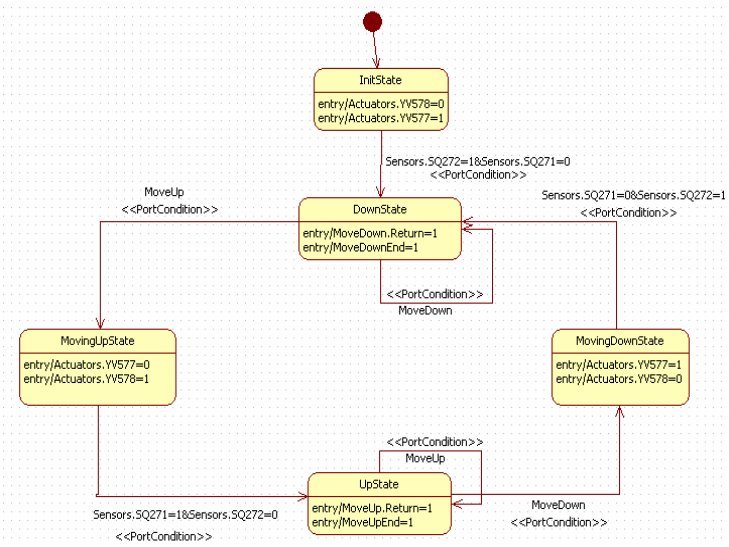

Figura 41: Diagramma UML per il Translator

L'insieme degli AC basic sono stati poi aggregati in una gerarchia modulare tramite AC composite introducendo inoltre una serie di appositi AC in grado di implementare funzionalità di livello superiore.

Tale modello ha permesso di testare il corretto funzionamento del simulatore in attesa che fosse pronto il modello definitivo e maggiormente dettagliato della Macchina SPI realizzato dal partner industriale del progetto MCM ([19]).

Il secondo passo per la generazione del controllo è l'utilizzo del tool messo a disposizione dal progetto MEDEIA. Questa interfaccia permette di importare il modello in formato UML nel repository centralizzato degli AC e quindi di istanziare i tipi di AC importati generando in fine il codice di controllo per diverse piattaforme. In particolare per il controllo della cella SPI si è scelto di utilizzare la piattaforma softPLC Orchestra, quindi il sistema di generazione MEDEIA ha permesso di ottenere il progetto softPLC in formato XML (con la creazione di tutti gli Automation Components) pronto per essere importato e compilato nel particolare ambiente di controllo.

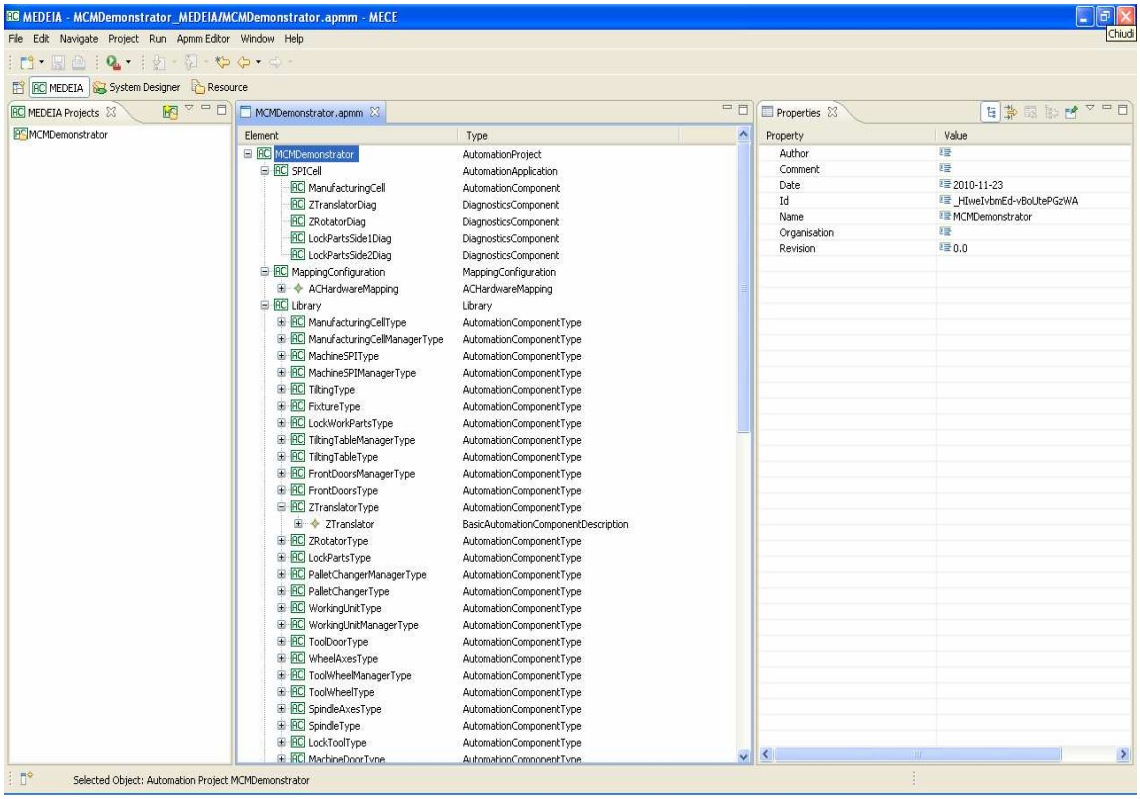

Figura 42: Creazione degli ACs

Tutta la parte di diagnostica, è stata anch'essa importata nel software MEDEIA ed esportata pronta ad essere utilizzata in parallelo al controllo della Macchina.

Il terzo ed ultimo passo prevede di importare il file in formato XML generato dal software MEDEIA all'interno dell'IDE del softPLC Orchestra (Logic Programming). Questo ambiente grafico per la programmazione del softPLC permette di gestire le diverse POU (Program Organization Unit) e creare le istanze di configurazione (Task). All'interno delle istanze è possibile settare il tempo di ciclo e le priorità di schedulazione di ciascuna istanza. Sempre dal Logic Programming è possibile gestire tutte le variabili globali, le librerie necessarie e i tipi di dati. Occorre infine configurare a mano i canali di input e output. Nel progetto si sono utilizzate due configurazioni:

1. I/O Profibus: gestisce la comunicazione dei segnali della parte reale del sistema (Pallet Changer e Robot);
2. I/O Seriale: gestisce la comunicazione dei segnali per tutta la parte simulata (Macchina SPI e stazione operatore).

Tramite lo stesso tool è possibile compilare ed eseguire il progetto importato sulla piattaforma scelta.

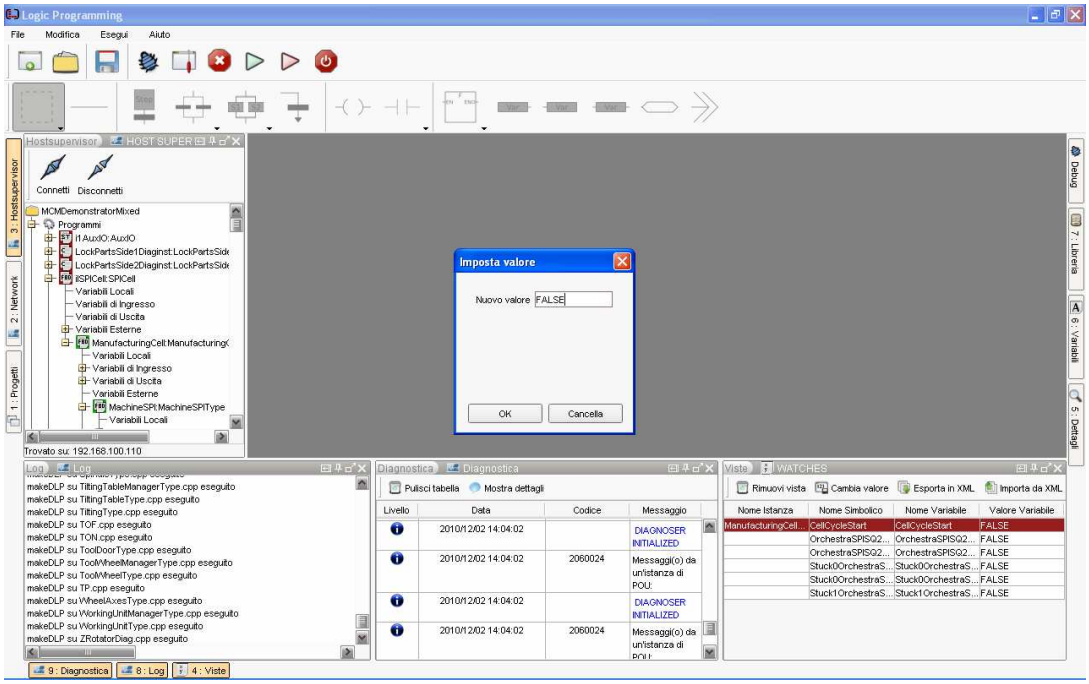

Figura 43: Overview del componente Orchestra Logic Programming

5.3.2 Test del controllo

La generazione del controllo non implica che il lavoro sia terminato, anzi ora occorre verificare che tutte le scelte modellistiche e tutte le logiche siano corrette e coerenti con il comportamento nominale desiderato del sistema.

La verifica non ha comportato alcun pericolo per le parti meccaniche della Macchina reale o per gli utenti, infatti è stata svolta utilizzando l'impianto simulato o una combinazione di dispositivi reali e simulati come definito dall'approccio Mixed Reality. Così facendo, tutti gli errori, le dimenticanze o le sviste sono emersi producendo malfunzionamenti nell'ambiente simulato che non hanno quindi comportato particolare rischi o danni. Se la stessa operazione di verifica si fosse effettuata sulla Macchina SPI reale si sarebbero potute verificare delle conseguenze indesiderate come:

- collisioni e danneggiamenti di componenti;
- cablaggi strappati;
- rischi per l'operatore;
- tempistiche lunghe per il rifasamento della cella in caso di logica errata.

Simulando, quando si presenta una logica errata, è possibile fermare l'esecuzione del programma e isolare il motivo del comportamento non nominale. Per fare ciò occorre effettuare il debug del codice di controllo o del modello UML, individuando attraverso i manager dei componenti lo stato di blocco della simulazione. Trovato lo stato in cui si ferma l'automa, si verificano le condizioni che permettono il passaggio allo stato successivo e si correggono gli eventuali errori che bloccano il flusso delle operazioni.

Se il problema non deriva da una logica errata, bisogna concentrarsi sulla comunicazione fra controllo e simulatore. Questa considerazione non è affatto banale, infatti durante i test molto spesso le logiche funzionavano a dovere e i problemi derivavano dallo scambio di informazioni. Durante i test per facilitare la retroazione HIL (Hardware In the Loop) è stata utilizzata anche una connessione TCP tra simulatore e PLC che tuttavia non ha portato a buoni risultati, molto spesso alcune variabili subivano infatti dei ritardi di lettura inaccettabili per le tempistiche del controllo producendo blocchi della simulazione o addirittura portando all'isolamento di guasti inesistenti da parte del sistema automatico di diagnosi. Successivamente si è utilizzata una porta seriale. Questa soluzione, dopo diversi giorni di prove e scelte dei parametri di

configurazione ha migliorato e reso accettabile lo scenario di simulazione scongiurando la perdita dei dati.

5.4 Architetture per la prova sperimentale di Mixed Reality

Viene ora descritta l'architettura utilizzata nella prova sperimentale. Per evitare pericoli derivanti da azioni che non fanno parte del ciclo nominale, si è deciso di sostituire i pezzi reale in metallo sul Pallet Changer con delle lampadine che evidenziano (luce accesa) o meno (luce spenta) l'effettiva presenza del pezzo.

Figura 44: Lampadine per simulare la presenza pezzo

5.4.1 Configurazioni

La prima configurazione utilizzata è stata la simulazione dell'intera cella SPI. Nessuna parte reale è impiegata in questa situazione: è ideale per testare la sequenza delle lavorazioni e controllare l'esattezza delle operazioni svolte dai vari componenti. Tale configurazione prende il nome di Hardware In the Loop (HIL), l'uso cioè di un simulatore interfacciato ad un PLC hardware di controllo. L'uso che ne è stato fatto è spiegato nel paragrafo 5.3.2 dove viene spiegato il test logiche.

Successivamente, si è sostituito il Pallet Changer simulato con il componente reale per ottenere la configurazione di prova utilizzata a presso l'azienda MCM. Grazie alla

modularità di SIMBA tale operazione è risultata semplice e veloce. È sufficiente infatti commentare nel codice del simulatore, la parte che non deve essere più rappresentata nella simulazione: la figura 45 mostra il “Simulation Element” Pallet Changer e la corrispondente connessione grafica che una volta commentati ne impediscono la visualizzazione e l’esecuzione nell’ambiente di simulazione.

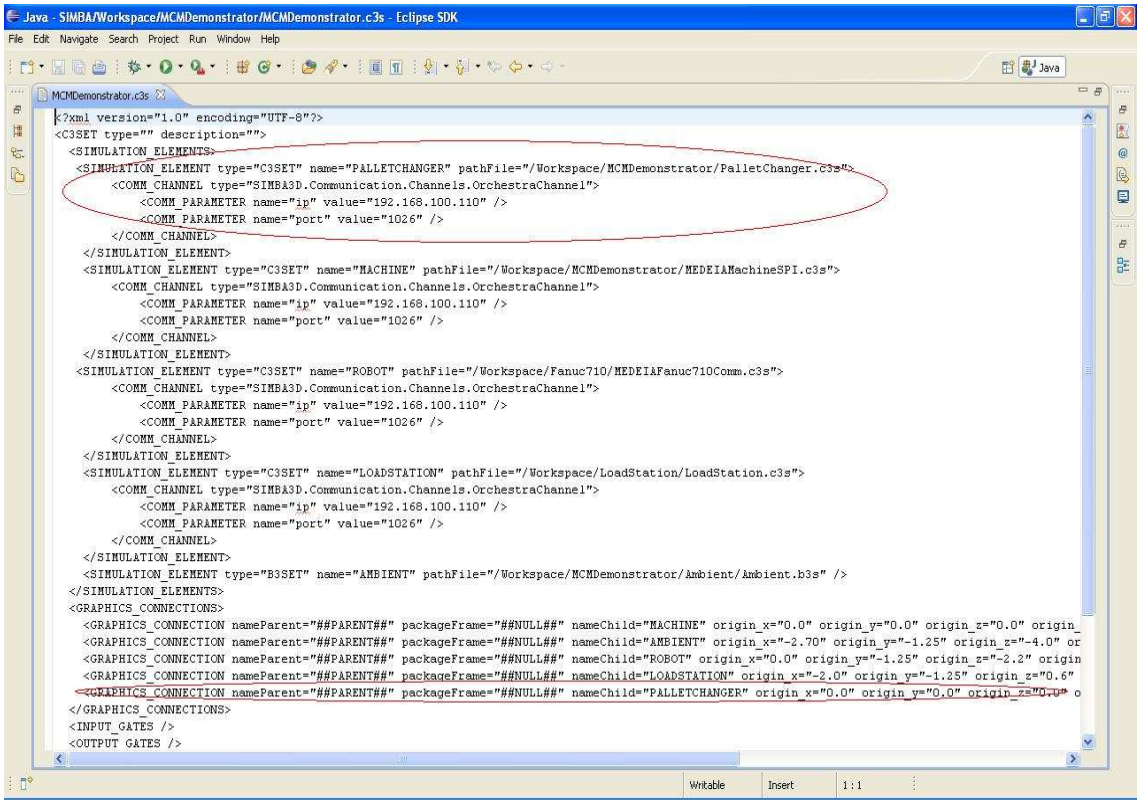

Figura 45: Parte di codice da commentare

È semplice intuire che grazie a questa peculiarità, quando è possibile avere a disposizione un altro pezzo reale dell’impianto, non si ha alcun tempo morto di revisione del simulatore, semplicemente basta commentare la sua copia simulata.

Con lo stesso ragionamento, è immediato capire i passi necessari per integrare l’ultimo componente reale per la simulazione (il robot) e ottenere la terza configurazione: si commenta il “Simulation Element” denominato robot e la sua connessione grafica. Per completare l’operazione occorre riconfigurare i segnali di ingresso e uscita. Tutti quelli

che fanno parte della simulazione saranno contenuti nella comunicazione seriale, mentre i segnali della parte reale saranno gestiti dalla configurazione profibus.

5.4.2 Implementazione del robot

Ad arricchire lo scenario della prova è stato inserito un robot fanuc dotato di 6 gdl atto a movimentare i pezzi. La pinza del manipolatore è di tipo rotante a due posti, ciò gli permette di prelevare due pezzi alla volta. e svolge due funzioni:

- preleva dalla stazione di carico/scarico i pezzi grezzi da lavorare e li depone sul Pallet Changer;
- preleva dal Pallet Changer i pezzi lavorati dalla Macchina e li depone sulla stazione di carico/scarico.

Anche per esso è stato creato un modello di simulazione in SIMBA. Per prima cosa, un'immagine prelevata da internet di un manipolatore fanuc è stata elaborata con 3D Studio Max ([22]) per renderla fedele al vero robot che sarà a disposizione per la prova sperimentale.

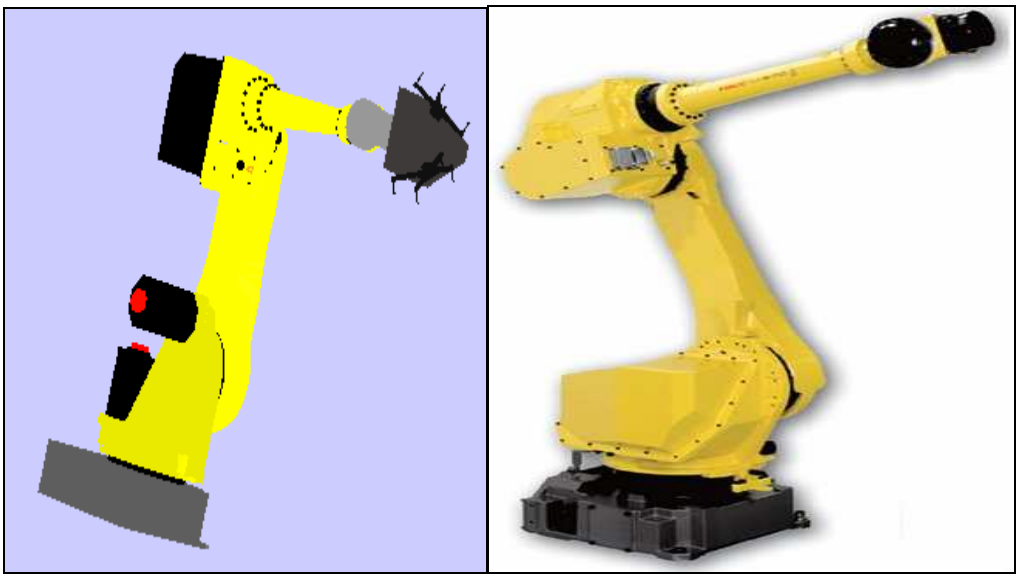

Figura 46: Similitudine fra robot reale e robot simulato

Successivamente si sono calcolate tutte le posizioni per generare la traiettoria simulata del robot.

Si mostra una parte del codice java che implementa la traiettoria del robot.

Legenda delle variabili dell'interfaccia di comunicazione del controllore del robot:

- ***RBRUN:*** variabile booleana indica quando il robot è in movimento;
- ***RBRDY:*** variabile booleana indica quando il robot è pronto a svolgere un'operazione;
- ***RBRST:*** variabile booleana indica il reset della posizione del robot;
- ***RBSL:*** variabile booleana indica l'inizio del carico (start load);
- ***RBSU:*** variabile booleana indica l'inizio dello scarico (start unload);
- ***RBLC:*** variabile booleana indica il completamento dell'operazione di carico (load completed);
- ***RBUC:*** variabile booleana indica il completamento dell'operazione di scarico (unload completed);

```
case 0: moveRobot(0.0,-0.816,0.6,0.0,-0.5,0.0);
                              break;

case 1: if(ENDMOVE.getValue()!=null)
        if(!ENDMOVE.getValue())
                  {
                  RBRUN.writeValue(false);
                  RBRDY.writeValue(true);
                  RBRST.writeValue(true);

      if(RBSL.getValue()!=null&&RBSU.getValue()!=null)

      if(!RBSL.getValue()&&!RBSU.getValue())
                                                {
      RBLC.writeValue(false);
      RBUC.writeValue(false);
      state = state + 1;
                  }
                        }
      break;

case 2:if(RBSL.getValue()!=null)
       if(RBSL.getValue())
       state = 3;
       if(RBSU.getValue()!=null)
       if(RBSU.getValue())
       state = 36;
       break;

case 3: moveRobot(0.0,-0.816,0.6,0.0,-0.5,0.0); // START LOAD (ROBOT
IN HOME)
      break;

case 4: if(!ENDMOVE.getValue())
        state = state + 1;
      break;
```

```
case 5: moveRobot(3.11,-0.816,0.6,0.0,-0.5,0.0); // MOVE TO OPERATION
STATION
        break;

case 6: if(!ENDMOVE.getValue())
        state = state + 1;
        break;

case 7: moveRobot(3.11,0.4,-0.3,0.0,0.4,0.0); // APPROACH TO PIECE 1
(LEFT)
        break;

case 8: if(!ENDMOVE.getValue())
        state = state + 1;
        break;


case 9: moveRobot(3.11,0.46,-0.2,0.0,0.2,0.0); // CATCH PIECE 1
OPERATOR STATION (LEFT)
        break;

case 10: if(!ENDMOVE.getValue())
         state = state + 1;
         break;

case 11: moveRobot(3.11,0.4,-0.3,0.0,0.4,0.0); // UP PIECE 1
         break;

case 12: if(!ENDMOVE.getValue())
         state = state + 1;
         break;

case 13: moveRobot(2.91,0.4,-0.3,0.0,0.4,3.14); // APPROACH TO PIECE 2
(RIGHT)
         break;

case 14: if(!ENDMOVE.getValue())
         state = state + 1;
         break;
```

5.4.3 Logiche per la Mixed Reality

Definito lo scenario della prova, sono state ideate e successivamente implementate in una POU in linguaggio ST del controllo le logiche per gestire lo scambio di informazioni e pezzi per la simulazione Mixed Reality.

La sequenza di operazioni dell'intero ciclo di produzione nominale è la seguente:

1. l'operatore carica due pezzi grezzi sulla stazione di carico/scarico;
2. il robot dalla posizione di riposo in home si dirige alla stazione di carico/scarico e preleva i pezzi per poi deporli sul Pallet Changer;
3. il Pallet Changer ruota di 180 gradi e si alza per permettere alle fixture della Macchina SPI di prelevare i pezzi grezzi da lavorare;
4. il Pallet Changer si abbassa;
5. la Macchina lavora i pezzi e al termine del ciclo abbassa le fixture;
6. il Pallet Changer si alza e riceve i pezzi lavorati;
7. il Pallet Changer si abbassa e ruota di 180 gradi per permettere lo scambio dei pezzi;
8. il robot dalla posizione di riposo recupera i pezzi dal Pallet Changer e li deposita sulla stazione di carico/scarico;
9. l'operatore scarica le parti lavorate.

Dalla sequenza di lavoro appena presentata si identificano due tipologie di trasferimento pezzo:

- il passaggio tra i diversi dispositivi (per esempio, quando il robot preleva i pezzi dalla stazione, scompariranno da quest'ultima e appariranno sulle pinze del robot);
- il passaggio tra mondo reale e mondo virtuale: quando il pezzo arriva sul Pallet Changer o sul robot (cioè sui dispositivi reali che verranno sostituiti dall'interfaccia di simulazione), deve scomparire dall'ambiente di simulazione e apparire nella realtà con l'artificio visivo della lampadine.

Nel seguito si descrivono nel dettaglio le logiche implementate e le variabili utilizzate.

Legenda variabili:

- ***PartPos:*** variabile booleana, indica la presenza pezzi sulla stazione operatore;
- ***PartPRb:*** variabile booleana, indica la presenza pezzi sulle pinze del robot;

- ***PartPC1:*** variabile booleana, indica la presenza pezzi sul side 1 del Pallet Changer;
- ***PartPC2:*** variabile booleana, indica la presenza pezzi sul side 2 del Pallet Changer;
- ***PartFix:*** variabile booleana, indica la presenza pezzi sulle fixture.

Logiche Mixed Reality:

➢ **Se arriva il comando di carico pezzi sulla stazione, i pezzi sono abilitati a comparire.**

```
IF OrchestraSPIOSLCIn THEN
   PartPOs := TRUE;
END_IF;
```

➢ **Se arriva il segnale di fine scarico pezzi, i pezzi scompaiono dalla stazione.**

```
IF OrchestraSPIOSUCIn THEN
   PartPOs := FALSE;
END_IF;
```

➢ **Se arriva il segnale di carico del robot, se il robot non ha le pinze occupate e ci sono dei pezzi disponibili sulla stazione di carico, allora i pezzi scompaiono dalla stazione di carico e compaiono sulle pinze del robot.**

```
IF OrchestraSPIRBSLOut AND PartPRb AND NOT RBPart AND PartPOs THEN
   PartPOs := FALSE;
   RBPart := TRUE;
END_IF;
```

➢ **Se arriva il segnale di scarico del robot e se il robot ha i pezzi sulle pinze, allora i pezzi compaiono sulla stazione di scarico e scompaiono dalle pinze del robot.**

```
IF OrchestraSPIRBSUOut AND NOT PartPRb AND RBPart THEN
   RBPart := FALSE;
   PartPOs := TRUE;
END_IF;
```

- **Se arriva il segnale di scarico del robot, se il robot non ha le pinze occupate, e se è verificata la condizione di presenza pezzo sul side 1 (oppure sul side2) del pallet, allora i pezzi scompaiono dal pallet e compaiono sulle pinze del robot .**

```
IF OrchestraSPIRBSUOut AND NOT RBPart AND PartPRb THEN
    IF OrchestraSPISQ269In AND PartPC2 THEN
    PartPC2 := FALSE;
   RBPart := TRUE;
    END_IF;
  IF OrchestraSPISQ270In AND PartPC1 THEN
    PartPC1 := FALSE;
    RBPart := TRUE;
    END_IF;
END_IF;
```

- **Se arriva il segnale di inizio carico del robot e ci sono dei pezzi sulle pinze del robot, allora i pezzi scompaiono dalle pinze del robot e vengono depositati sul side corrispondente del pallet.**

```
IF OrchestraSPIRBSLOut AND NOT PartPRb AND RBPart THEN
  RBPart := FALSE;
  IF OrchestraSPISQ269In THEN
    PartPC2 := TRUE;
    END_IF;
  IF OrchestraSPISQ270In THEN
    PartPC1 := TRUE;
    END_IF;
END_IF;
```

➢ **Se le fixture sono nella posizione di aggancio pezzi, i pezzi compaiono sulle pinze e scompaiono dalla parte corrispondente del Pallet Changer.**

```
IF OrchestraSPIAPOSITIONIn = -1.57 AND OrchestraSPIBPOSITIONIn = 1.57 AND OrchestraSPICPOSITIONIn = 1.57 AND OrchestraSPISP78In THEN
   IF OrchestraSPISQ270In AND OrchestraSPISQ271In AND PartPC2 AND NOT PartFix THEN
      PartFix := TRUE;
      PartPC2 := FALSE;
      END_IF;
   IF OrchestraSPISQ269In AND OrchestraSPISQ271In AND PartPC1 THEN
      PartFix := TRUE;
     PartPC1 := FALSE;
      END_IF;
END_IF;
```

➢ **Se le fixture sono in posizione bassa e ruotata, se ci son i pezzi sulle fixture, allora i pezzi lavorati vengono depositati sul pallet (si accendono le lampadine) e scompaiono dalle fixture.**

```
IF OrchestraSPIAPOSITIONIn = -1.57 AND OrchestraSPIBPOSITIONIn = 1.57 AND OrchestraSPICPOSITIONIn = 1.57 AND OrchestraSPISP79In AND PartFix THEN
   IF OrchestraSPISQ270In AND OrchestraSPISQ271In AND NOT PartPC2 THEN
      PartFix := FALSE;
      PartPC2 := TRUE;
      END_IF;
    IF OrchestraSPISQ269In AND OrchestraSPISQ271In AND NOT PartPC1 THEN
      PartFix := FALSE;
      PartPC1 := TRUE;
      END_IF;
END_IF;
```

6

CONCLUSIONI

6.1 Risultati ottenuti

Lo scopo di questo lavoro è stato organizzare un'architettura software e hardware atta a dimostrare la realizzabilità e l'efficacia di un progetto di simulazione Mixed Reality nell'ambito dell'automazione. La simulazione finale di una cella mista di lavorazione effettuata presso il gruppo MCM di Piacenza è stata la "prova" generale di quanto verrà mostrato ai revisori del progetto europeo MEDEIA in occasione del meeting di febbraio 2011. I risultati ottenuti dimostrano che è possibile far interagire correttamente e in real time parti fisiche reali con parti simulate, ottenendo una realtà mista che rispecchia perfettamente il comportamento dell'intero sistema reale. Con l'implementazione di opportune logiche è stato possibile rappresentare il passaggio di pezzi da lavorare dal mondo virtuale a quello reale. La configurazione presentata nell'elaborato è solo un esempio tra gli infiniti scenari realizzabili. Implementando ed arricchendo le librerie degli ambienti di simulazione e simulando graficamente i vari componenti industriali più o meno standard (pistoni, porte, robot, ecc.) è possibile creare e far interagire diverse realtà in ogni tipo di situazione.

Si è quindi dimostrata la modularità di tale approccio aiutata anche dal meta-modello SIMBA: partendo infatti da uno scenario completamente simulato, si sono integrate via via le parti reali a disposizione in quel momento (il Pallet Changer e/o il robot) in maniera rapida ed efficiente. L'utilizzo della Macchina SPI completamente simulata ha permesso il test delle logiche di controllo in Harware In the Loop (HIL), senza alcun pericolo di danni al Macchinario o incidenti a persone. L'integrazione del robot e della

stazione di carico nello scenario di simulazione hanno permesso di valutare le tempistiche per la sequenza carico-lavorazione-scarico pezzi.
Si è implementata la funzione di Monitoring 3D: sebbene la velocità di esecuzione non è ancora soddisfacente a causa del collegamento con il PLC (tramite protocollo TCP), si potrà superare tale problema utilizzare una comunicazione di tipo seriale oppure EtherCat. Sempre durante la prova finale si è dimostrato come il metodo di diagnostica integrato sia in grado di isolare i guasti in maniera veloce ed efficiente.

6.2 Sviluppi futuri

Considerando l'esperienza accumulata durante lo svolgimento del lavoro di tesi e alla luce di tutti i risultati ottenuti, è facile convincersi della validità dell'idea di Mixed Reality. Un concreto investimento permetterebbe ulteriori prove di scenari riconfigurabili rafforzando i risultati qui presentati. Gli aspetti più interessanti da approfondire vengono riportati nei paragrafi successivi.

6.2.1 Evoluzione del Monitoring 3D

Come spiegato in precedenza, la funzione di monitoraggio di un impianto industriale o di un semplice Macchinario è di particolare interesse. Se inoltre vi fossero aggregate funzioni di diagnostica avanzata sui componenti si può affermare che il Monitoring 3D diventerebbe quasi una condizione necessaria per ogni realtà industriale moderna, flessibile e competitiva. Per prima cosa, potendo usufruire di protocolli di comunicazione più veloci e sicuri, si otterrebbe una visualizzazione nitida e poco ritardata delle condizioni dell'impianto. Di pari passo si potrebbe pensare di inserire una serie di segnalatori grafici in grado di aumentare l'informazione presentata all'operatore. Si potrebbe ad esempio riportare il rilevamento di un guasto, oltre al messaggio su monitor di allarme, evidenziando componente non funzionante ed emettendo un particolare avviso sonoro per attirare l'attenzione dell'operatore umano. Tutti accorgimenti che all'apparenza potrebbero sembrare inutili ma che con un' analisi più profonda fanno comprendere l'enorme risparmio di tempo (e quindi di denaro considerando i tempi morti di fermo Macchina) nel ripristino delle condizioni normali di produzione.

6.2.2 Livello superiore della diagnostica

Compresa la fondamentale utilità della diagnostica, sarebbe opportuno in futuro investire anche su questo aspetto. Al momento infatti è possibile isolare i guasti a livello del singolo componente questo lascia aperte alcune mancanze, si possono infatti identificate le seguenti prospettive:

1. isolamento dei guasti ad un livello superiore del singolo componente;
2. generazione automatica del metodo TiDiaM partendo dalle informazioni disponibili sul sistema da diagnosticare;
3. implementare il metodo TiDiaM su alte piattaforme (es. PLC fisici o softPLC di altre marche).

6.2.3 Comunicazione EtherCat

Durante la lunga fase di testing immancabilmente si sono verificati malfunzionamenti, i più frequenti possono essere riassunti così:

- mancata apertura delle porte;
- non sincronismo di apertura delle porte frontali;
- movimenti incongruenti del Pallet Changer;
- blocchi improvvisi durante la lavorazione.

La maggior parte delle volte, all'incorrere di un problema il primo pensiero era orientato a cercare un errore nelle logiche di controllo oppure un errore nell'implementazione del simulatore. Spesso la soluzione era davvero correggere uno dei due aspetti sopraccitati, altre volte il malfunzionamento rimaneva un mistero e riavviando la simulazione tutto procedeva nel giusto modo. Verso la fine del lavoro, dopo tantissime ore di test è maturata una nuova consapevolezza riguardante i vari malfunzionamenti: le logiche di controllo e il simulatore erano stati testati e funzionavano bene, mancavano invece conferme sulla comunicazione dei dati. In altre parole, non si aveva la certezza che il flusso di informazioni tra controllo e simulatore avvenisse perfettamente senza la perdita di alcun segnale. Nel dettaglio, la comunicazione tramite TCP risulta insoddisfacente perché si ha un'elevata perdita di dati, infatti il controllo real time impegna notevolmente le risorse della Macchina impedendo la corretta comunicazione dei dati causata anche da un eccessivo intasamento del canale. Motivo principale per la

scarsa visibilità del Monitoring è proprio questo, una lenta e non costante comunicazione dati. La comunicazione via seriale migliora notevolmente le cose a patto di un'attenta e meditata scelta dei tempi di schedulazione dei vari componenti ma anche in questo caso non sempre si sono ottenuti risultati soddisfacenti, soprattutto a causa della scarsa reperibilità di porte seriali nei moderni calcolatori e a causa dell'inaffidabilità dei convertitori seriale-usb.
Dall'esigenza di avere uno scambio dati affidabile e veloce è dunque nata l'idea di utilizzare un altro protocollo di comunicazione, l'EtherCat.

Lo standard EtherCat (Ethernet Control Automation Technology) è un protocollo di comunicazione ad elevate prestazioni per connessioni Ethernet deterministiche. Esso estende lo standard Ethernet IEEE 802.3 al trasferimento dati con una temporizzazione prevedibile ed esatta sincronizzazione. Il protocollo EtherCat implementa un'architettura master/slave appoggiandosi su di una connessione Ethernet standard. Il protocollo EtherCat trasferisce i dati appoggiandosi direttamente ai dataframe Ethernet standard senza la necessità di apportare modifiche alla loro struttura.

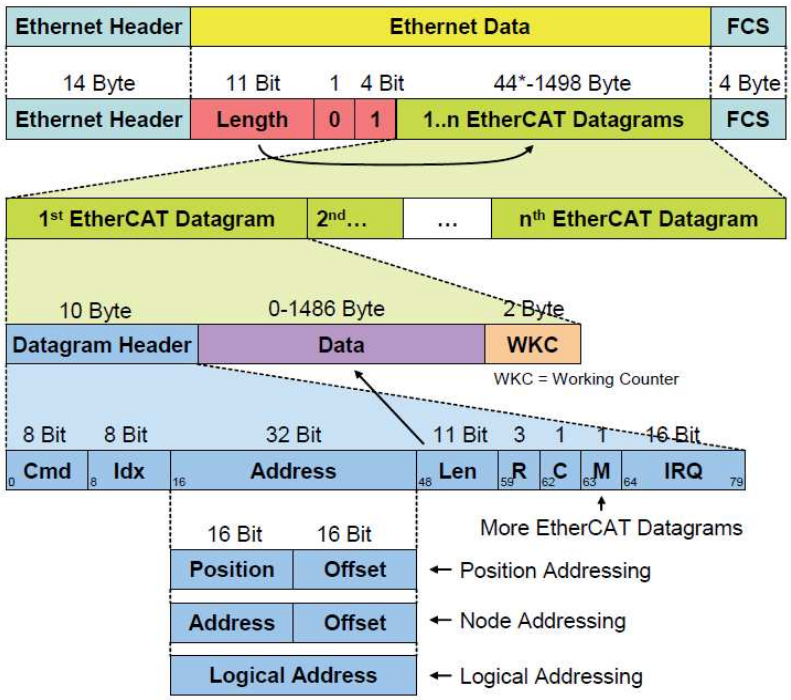

Figura 47: Architettura EtherCat

I dati vengono trasmessi tra master e slave in forma di PDO (process data objects). Ciascun PDO ha un indirizzo verso uno o più slave; questa combinazione di "dati e indirizzi" (con l'aggiunta del conteggio di validazione) forma un telegramma EtherCat. Il protocollo EtherCat è progettato per raggiungere elevate prestazioni e per gestire un alto numero di canali in applicazioni di controllo single-point. Poichè la fase di lettura e scrittura dello slave può avvenire nello stesso frame, la struttura del telegramma EtherCat è ottimizzata per gestire sistemi di I/O decentralizzati. Inoltre, la completa elaborazione del protocollo avviene all'interno dell'hardware e pertanto è indipendente dalle prestazioni della CPU e da ogni componente software.

In ambito industriale esistono diverse aziende che vendono schede di comunicazione EtherCat ma il materiale pubblico a disposizione è davvero limitato. Dopo varie ricerche è stato possibile trovare alcuni documenti che spiegano il funzionamento di questo protocollo ma nessuna fonte con una descrizione minuziosa su come implementare un master o uno slave. In particolare, gli slave vengono unicamente forniti come componenti esterni dotati di una memoria eeprom e riconosciuti dal master tramite dei dati identificativi (numero di serie, tipo di scheda, id del venditore…) contenuti in tale memoria. Il primo passo è stato dunque quello di comprendere la struttura di un pacchetto EtherCat. Si è scritto un programma in C che prendesse in ingresso un pacchetto EtherCat e leggendolo lo suddividesse bit a bit per riconoscere le varie parti di cui è composto. Il passo successivo prevedeva di simulare la comunicazione del master con i vari slave, implementando i dati identificativi in zone ben definite della memoria del pc (emulando le mancanti schede fisiche dotate di memoria eeprom). Causa i tempi stretti per la chiusura del progetto purtroppo le prove per emulare il comportamento degli slave è risultato molto ridotto e si è deciso di rinviare l'implementazione EtherCat. Ne rimane comunque una discreta conoscenza teorica acquisita e la possibilità di sviluppare l'implementazione impostata in lavori futuri, con lo scopo di migliorare le performance della comunicazione tra il softPLC Orchestra e l'ambiente di simulazione SIMBA.

BIBLIOGRAFIA

[1] Ferrarini L., Dedè A. 2007, *A model-Based Approach for Mixed Hardware In the Loop Simulation of Manufacturing System.*

[2] Ferrarini L., Dedè A. 2009, *A Mixed-Reality Approach to Test Automation Function for Manufacturing System.*

[3] Ferrarini L., Castelnuovo A., *A Modular Formal Model for Pallet Transportation System in Machining Centres Automation.*

[4] Ferrarini L., Dedè A., Allevi M. 2010, *Design and implementation of an automatic on-line diagnosis with TiDiaM and TiDE.*

[5] Ferrarini L., Dedè A., Cerrada M. 2010, *Modular fault diagnosis using temporized analysis for a class of discrete event systems.*

[6] Ferrarini L., Fogliazza G., Weber C. 2005, *IEC 61499 implementation of a Modular Control Model for Manufacturing Systems.*

[7] Paoli A. 2009, *Diagnostica di sistemi dinamici mediante modelli ad eventi discreti, appunti del corso.*

[8] Pattavina A. 2003, *Reti di telecomunicazioni, McGraw-Hill.*

[9] Ceri S., Mandrioli D., Sbattella L., 2000, *Informatica programmazione, McGraw-Hill.*

[10] H. Deitel, P. Deitel, 2000, *Corso completo di programmazione, Apogeo.*

[11] MEDEIA Project 2010, *MEDEIA White Paper, www.MEDEIA.eu*.

[12] MEDEIA Project 2008, *MEDEIA ProjectPresentation, www.MEDEIA.eu.*

[13] MEDEIA Project 2008, *Medeia Deliverable D33*.

[14] Panizza P. 2010, *Elaborato di laurea, Definizione di un meta-modello per la simulazione grafica tridimensionale di sistemi manifatturieri, Politecnico di Milano.*

[15] Lupo M. 2009, *Elaborato di laurea, Simulazione grafica tridimensionale Model-Based per sistemi automatici di produzione discreta, Politecnico di Milano.*

[16] Gianazza A. 2010, *Elaborato di laurea, Modellizzazione ed implementazione di un sistema di generazione automatica di codice per il controllo di centri di lavoro, Politecnico di Milano.*

[17] Dedè A. 2008, *Tesi di laurea, Simulazione grafica 3D modulare per impianti di produzione discreta, Politecnico di Milano.*

[18] *Technical Folder Machine SPI, versione 2010.*

[19] MCM, *www.mcmspa.it.*

[20] University of Michigan,*DESUMA-UMDES, www.eecs.umich.edu/umdes/toolboxes.html*

[21] Sintesi 2010, *Orchestra Control Engine, www.orchestracontrol.com*.

[22] 3D Studio Max 2010, *http://usa.autodesk.com/.*

[23] Java 2010, *http://www.java.com.*

[24] Ferrarini L., Dedè A. 2010, *Progetto Medeia, il futuro del controllo è model-based, automazione industriale.*

[25] Paul Milgram, Haruo Takemura, Akira Utsumi, Fumio Kishino 1994, *Augmented Reality: A class of displays on the reality-virtuality continuum.*

[26] EtherCat, *EtherCat slave configuration, http://www.ethercat.org/*