Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Corso di Dottorato in Ingegneria dell'Informazione
Curriculum in Ingegneria Informatica, Gestionale e dell'Automazione

---

Automazione e controllo di macchine per packaging in ambito fabbrica intelligente

Dottorando:

Emanuele Lorenzoni

Tutor:

Prof. Andrea Bonci

Coordinatore:

Prof. Franco Chiaraluce

19° ciclo – nuova serie

Alla mia famiglia,

Sommario

Introduzione **4**
Capitolo 1. ..... **7**
Industria 4.0 ..... **7**
1.1 Origini 7
1.2 Barriere e sfide 9
1.3 Pipeline digitale in manifattura 10
1.4 CPS (Cyber-Physical-System) 14
1.5 Fabbriche e prodotti intelligenti 15
1.6 Gli agenti 17
1.7 Agenti industriali – Architettura a due layers 19
1.8 Gli standard per il controllo reattivo 21
1.9 Standard IEC 61499 21
Modello del blocco funzionale 22
Modello di Sistema 25
Modello Applicazione 26
Modello del dispositivo 27
Modello della risorsa 27
Modello di gestione del dispositivo 28
1.10 Architettura RMAS 29
Relazioni IEC 61499 e Architettura RMAS 31
Mapping RMAS e Risorsa IEC 61499 32
Capitolo 2. ..... **34**
Produzione snella **34**
2.1. Origini e principi 34
2.2 Strumento Kanban 37
2.3 Strumento e-Kanban 45

Capitolo 3. 47
Progetto Shop-Floor 47
3.1. Obiettivi 47
3.2. Scenario di produzione 48
3.3. Semantica di comunicazione tra dispositivi 53
3.4. Applicazione Shop-Floor in standard IEC-61499 62
System Configuration – Configurazione di sistema 62
Applicazione: **INT_CLIENT** 65
Applicazione: **OUT_STORE_FULL** 69
Applicazione: **BASIC_PROCESS_CELL1** 81
Applicazione: **OUT_SUP** (Supermarket B) 83
Applicazione: **TRANSPORT_OPERATOR** 87
Applicazione: **IN_SUP** (Supermarket A) 89
Scenari di simulazione 90
Conclusioni 97
Appendice 101
Modelli semantici di esecuzione dei blocchi funzionali 101
Bibliografia 103

Introduzione

Siamo sicuramente in un momento particolare della storia industriale, probabilmente di una unicità tale che nemmeno riusciamo a realizzare, nel quale non possiamo definire con precisione o predire quali saranno i fattori veramente dirompenti che cambieranno drasticamente i nostri sistemi produttivi. Ciò perché numerosi sono i possibili approcci, le nuove idee e gli strumenti tecnologici che potremmo utilizzare per gestire la complessità da questi stessi generata. Numerosi saranno i cambiamenti che porteranno modifiche, di sicuro questi fattori saranno legati alla digitalizzazione sempre più pervasiva, all'incremento dell'automazione, al numero delle possibili alternative per il progetto di sistemi di controllo non solo industriali, ma anche sociali.
Le transizioni dirompenti sono avvenute nella prima e nella seconda rivoluzione industriale grazie a nuovi modi di produrre energia legati alla scoperta di nuove fonti, poi dalla seconda alla terza rivoluzione si è assistito alla transizione in nuove tipologie di controllo grazie alla integrazione spinta della potenza di calcolo. Dalla terza alla quarta rivoluzione si assiste alla transizione di un nuovo modo di usare le tecnologie conosciute. Lo studio di questo scenario industriale pone il problema dell'integrazione delle conoscenze tra le varie parti, organizzazione transdisciplinare e condivisione della conoscenza, specie in ambito accademico. La risoluzione di problematiche produttive nell'industria richiede conoscenza consolidata nell'uso di strumenti collaudati ma è anche vero che lo studio di questa complessità del mondo industriale in ottica futura, che deve essere affrontato anche a livello accademico, non dovrebbe porsi limiti o essere forzato dalla pressione del mercato, o almeno, sarebbe conveniente che si pensi un po' ad alto livello e cercare di affrontare con criterio la transizione capendo bene quali sono le sfide.

In questo scenario nasce questa tesi che esplora una parte delle attività svolte dall'autore nei tre anni di dottorato cofinanziato da una azienda operante nel settore *packaging*. Queste attività hanno interessato diverse aree con l'obiettivo di offrire un contributo teorico e pratico per quanto concerne l'efficientamento hardware delle macchine e i flussi di produzione/informativi dello shop-floor in ambito I4.0.
Nell'ottica di un efficientamento I4.0 dei consumi energetici del piatto vibrante di una macchina multi-teste prodotta dalla stessa azienda, si cita brevemente la proposta di utilizzo di un dispositivo "Smart" costituito da materiali di nuova generazione (elastomeri dielettrici) in sostituzione del tradizionale attuatore elettromeccanico (pesante ed energicamente poco efficiente). È stato dimostrato in uno studio preliminare e comparativo che l'attuatore smart garantisce stessa frequenza di vibrazione ma con consumi energetici ridotti, nonché notevoli guadagni in termini di peso. Questo studio, svolto dal dottorando in collaborazione con l'Università di Saarland e il Politecnico di Bari non sarà riportato in questa tesi, il lettore interessato può riferirsi all'articolo [1].

Industria 4.0 non è più solo efficientamento hardware, il nuovo paradigma spinge prepotentemente verso l'efficientamento dei flussi informativi e produttivi con l'integrazione di tecnologie che oggi possono permeare ogni livello dell'organizzazione aziendale fino allo shop-floor. Spinti da questa concezione, gli argomenti descritti in questa tesi sono il risultato della messa a sistema di diversi filoni di studi di dottorato che contribuiscono a iniettare conoscenza circa le possibilità offerte dal paradigma I4.0, sia in ambiti aziendali delle PMI (*Piccola e Media Industria*) e orientata a proporre nuove soluzioni implementabili a breve termine, sia in quello accademico dove la visione di I4.0 può spingersi oltre i vincoli contingenti e proporre soluzioni di concetto interessanti con sviluppi futuri. Un fattore comune ad entrambi gli ambiti è sicuramente la conoscenza delle possibilità offerte dal mondo I4.0, la digitalizzazione pervasiva in ambito manifatturiero e le tecnologie abilitanti come i sistemi Cyber-fisici. Nel capitolo 1 di questa tesi è fatta una panoramica su queste possibilità e sfide future.

Durante l'esperienza diretta in azienda sono state analizzate in loco le aree di attività aziendali e le problematiche di produzione e logistica dello shop-floor. Sono stati identificati prodotti, processi produttivi e le modalità di trasmissione dei flussi informativi. Questo contesto ha ispirato lo studio di base della produzione Snella (*Lean Manufacturing*) alla ricerca di uno strumento che potesse essere utile nel contesto produttivo citato ma anche in generale da applicarsi in un contesto di PMI e non solo. Uno strumento trovato interessante della produzione *Lean* è lo strumento Kanban, per l'ottimizzazione del controllo di produzione in un contesto *pull*. Il capitolo 2 introduce, quindi, i concetti della produzione Snella, il meccanismo del controllo di produzione Kanban tradizionale, cartaceo, e la tipologia di carta Kanban elettronica, o virtuale, E-Kanban.

Sicuramente lo standard IEC 61499 per la progettazione di sistemi di controllo industriali offre un approccio molto interessante nell'ambito dei controlli distribuiti e riconfigurabili perché offre adattabilità, flessibilità ed è orientato alla interoperabilità. In ottica I4.0 offre caratteristiche molto interessanti per quanto concerne le interazioni con gli *agenti software,* tra cui la *configurazione dinamica* dei blocchi funzionali (FB o *Function Block*) che incapsulano il codice del controllo. Questo standard mette a disposizione del progettista un livello applicazione nel quale si definisce la funzionalità dell'applicazione e l'ordine di esecuzione delle FBs, indipendentemente dall'ambiente in cui la funzionalità verrà eseguita, quindi indipendentemente dall' hardware che si avrà a disposizione. Rappresenta quindi un approccio orientato all'applicazione e non orientato al dispositivo, ragion per cui fornisce molta flessibilità al problema del controllo.
In aggiunta, un relativo software di programmazione grafica 4Diac con la libreria *run-time* FORTE (ambiente per l'esecuzione del codice) sono risorse *Open source*, confidano in una comunità sempre più vasta di utenti e il numero dei controllori industriali commerciali che ne supportano l'installazione nei loro sistemi operativi è in continuo incremento. Per questi motivi, nel Capitolo 3 si è scelto di utilizzare questa piattaforma per il progetto di un controllo di produzione Kanban distribuito, riconfigurabile e orientato a I4.0, applicabile in una tipica catena produttiva di una PMI per l'ottimizzazione del flusso informativo e del flusso dei materiali tra i processi. Il progetto prevede la definizione dei dispositivi fisici essenziali in una catena del valore che include un magazzino di prodotto finito, un processo di produzione e un processo di trasporto materiale che coinvolge due supermarket o isole di stoccaggio. È

quindi definita una semantica di comunicazione base per lo scambio degli ordini di produzione e trasporto materiali tra i sistemi embedded della catena e che potrà essere adattata in futuro per specifiche applicazioni e specifici hardware. La piattaforma rappresenta inoltre una base concettuale e implementativa per l'integrazione di agenti supervisori nella catena di produzione.

Capitolo 1.

Industria 4.0

1.1 Origini

Il concetto che sta alla base di *Industria 4.0* (*I4.0*) può assumere diversi significati anche tra gli addetti ai lavori. La Quarta Rivoluzione Industriale che è in atto, con l'applicazione dei paradigmi I4.0, è la realizzazione che ci sono molte tecnologie nate, sviluppate e già adottate in altri settori o domini industriali che possono realmente potenziare le capacità delle industrie manifatturiere, spingendole a adottare nuove soluzioni per le proprie attività produttive e in generale suggeriscono nuovi modi di fare impresa. Se percorriamo la storia delle rivoluzioni industriali possiamo ben identificare le novità tecnologiche che hanno prodotto effetti dirompenti sul sistema produttivo industriale e la sensibile metamorfosi dei modelli sociali.
Indicativamente tra il 1780 e il 1820 possiamo collocare la prima rivoluzione industriale, esplosa con lo sviluppo dei sistemi a vapore che convertivano energia termica in energia meccanica. Questa tecnologia permise la creazione di linee produttive che abilitavano l'industria a produzioni su larga scala grazie alla standardizzazione sia del processo produttivo che del prodotto. Per la prima volta si cominciava a parlare del concetto di riusabilità delle macchine o parti di macchine. Tra il 1860 fino a circa il 1920 si colloca la seconda rivoluzione industriale che prese piede con l'avvento dell'elettricità e la possibilità di generare energia meccanica da quella elettrica. Potevano essere progettate stazioni di lavoro più compatte, macchine più piccole, più potenti e più sicure, facili da mantenere ed installare. Anche la trasportabilità via cavo dell'elettricità ne rendeva la gestione più semplice rispetto ai mezzi di trasmissione necessari per l'energia a vapore, ne conseguiva quindi che anche la progettazione del layout di fabbrica era dettato dalla nuova tecnologia. La seconda rivoluzione portò all'esplosione della produzione di massa e nello stesso periodo Frederick W. Taylor sviluppava la teoria della organizzazione del lavoro, i cui principi furono applicati con successo da Henry Ford per la gestione del lavoro nella catena di produzione automobilistica.
Indicativamente a partire dal 1950 collochiamo la terza rivoluzione industriale. In questo periodo si verificava la transizione dal controllo analogico delle macchine a quello digitale. I dispositivi di controllo impiegavano transistori e circuiti integrati sempre più spinti con tasso di complessità descritta dalla legge di Moore e potenze di calcolo sempre più elevate in dispositivi sempre più piccoli. Le macchine erano riprogrammabili con crescente grado di automazione ed erano controllate da un computer. La terza rivoluzione vede lo sviluppo e

l'impiego massiccio della robotica, robotica autonoma (AMR – Autonomous Mobile Robotics) e l'integrazione tra i vari sistemi.
L'effetto dirompente associato alla Quarta Rivoluzione Industriale è un fenomeno più complesso, non omogeneo, siamo ancora in una fase di transizione e la sua evoluzione nel lungo termine è legata ad un ampio spettro di tecnologie inclusivo di vecchie e nuove soluzioni che devono essere messe a sistema. In particolare, la realizzazione di un contesto I4.0 fattuale nella manifattura pone non pochi rischi e sfide aggiuntive che derivano dalla natura conservativa di questo tipo di industria. La migrazione verso il nuovo paradigma industriale o l'integrazione con tecnologie della vecchia e nuova generazione deve quindi essere sempre attentamente valutato.

Industria 4.0 indica la via per i nuovi sistemi di produzione**,** ma perché si sta andando in questa direzione? una prospettiva data dall'osservatorio dell'Unione Europea indica che le "fabbriche moderne sono più efficienti in un mercato sempre più competitivo, più veloci, responsive al cambiamento del mercato globale e più vicine al cliente". In un siffatto scenario la sostenibilità economica richiede la riconfigurabilità e l'adattabilità delle fabbriche, capaci di produzioni su scale minori, ciò significa piccoli lotti fino al lotto 1, alta variabilità del tipo di prodotto (personalizzazione del prodotto o *mass customization*) e veloce variazione del processo di produzione per soddisfare la richiesta del cliente (*pull production*). Il ciclo di vita del prodotto si accorcia, ciò significa che è necessario adattare le risorse di produzione ai nuovi prodotti e considerare che alcune risorse devono essere usate temporaneamente, suggerendo che alcune debbano essere affittate occasionalmente piuttosto che acquistate.

Quali sono i fattori che favoriscono lo sviluppo di I4.0?

Come osservato in [2] i fattori tecnologici che stanno favorendo e favoriranno lo sviluppo di *I4.0* saranno le solide infrastrutture di comunicazione sempre più economiche e introdotte in modo pervasivo, la possibilità di avere macchine, dispositivi a livello di campo, impianti, fabbriche ma anche i singoli prodotti connessi in una rete, individuabili, esplorabili e analizzabili in rete in quanto oggetti che ospitano dati, informazioni e *conoscenza* (CPS o *Cyber Physical System*).
L'industria manifatturiera è quindi coinvolta nell' adozione di tecnologie che abilitano la produzione intelligente, le cosiddette *Smart Manufacturing Technologies* tra le quali, IIoT (*Industrial Internet of Things*), *Big Data Analysis*, *Additive Manufacturing*. La tecnologia IIoT connette i dispositivi intelligenti che possono condividere informazioni, orizzontalmente tra i dispositivi dello shop-floor, condividere informazioni di diagnosi piuttosto che informazioni di produzione verticalmente fino ai software ERP (*Enterprise Resource Planning*) favorendo i processi decisionali, il miglioramento dei modelli di business esistenti o favorire la creazione di nuovi. Gli *Industrial Analytics* sono strumenti che processano e analizzano i *Big Data*, ossia una grande quantità di dati proveniente dai dispositivi IIoT disseminati nello shop-floor o tra sistemi informativi (IT) dell'organizzazione come i sistemi gestionali, ad esempio, per l'estrazione di informazione utile ai processi decisionali.

1.2 Barriere e sfide

Da fonte (http://www.ilconsortium.org/high-speed-network.htm) si possono invece individuare le principali potenziali barriere allo sviluppo di I4.0, e cioè:

- Il 73% delle compagnie non ha piani concreti per I4.0 e l'internet industriale o IoT, hanno solo idee generali o direttive generiche.
- Il 59% dei professionisti IT dicono che non si sono preparati a gestire l'incremento di dati aspettato, molti dipartimenti non hanno idea di come sfruttare la grande quantità di dati disponibili dai processi.
- Molti paesi offrono insufficienti condizioni per supportare una larga adozione di I4.0.
- Il 36% dei dirigenti ammette che le barriere tra dipartimenti (di carattere sociale e umano dei lavoratori) ostacola la raccolta e la correlazione di dati.
- C'è urgenza di rifocalizzare l'educazione, di rieducare a pensare differente, per capire quali sono e saranno i fattori che cambieranno drasticamente gli scenari di produzione.
- Gli standard sono proprietari e per integrare occorre che i sistemi parlino la stessa lingua. Da questo punto di vista i lavori di integrazione tra processi sono ancora sospesi o non esistenti.

I dati indicano che l'effetto dirompente di I4.0 non è ancora stato recepito, probabilmente sarà necessario altro tempo per vedere la concreta applicazione di un ecosistema eterogeneo di componenti perfettamente integrati. Anche la ricerca nell'internet industriale non è abbastanza matura da questo punto di vista.

1.3 Pipeline digitale in manifattura

Nel settore manufatturiero il processo di collezione dei dati non è così semplice dal momento che tutte le risorse di una organizzazione aziendale possono essere potenziali sorgenti di dati e spesso queste risorse sono molto differenti tra loro, possiamo cioè avere differenti macchinari, diversi processi produttivi, diversi prodotti che in generale supportano differente connettività: ad esempio, abbiamo sistemi di localizzazione in tempo reale (*Real Time Localization Systems o RTLS),* sistemi IoT, PLC (*Programmable Logic Controller*), sistemi

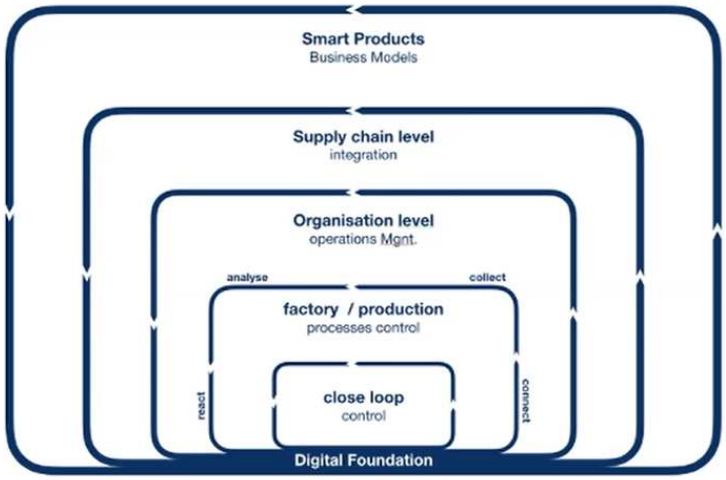


Fig. 1.3.1: *Pipeline* digitale

software che sono usati per la gestione della produzione, MES (*Manufacturing Execution Systems*), WMS (*Warehouse Management System*) e DBMS (*Database Management System*) per l'immagazzinamento delle informazioni di produzione.
Nello specifico, in figura 1.3.1 è mostrato un modello di controllo di un sistema industriale che si fonda sulla digitalizzazione di processi e componenti (*digital foundation*). Per ogni livello (in astratto è compreso anche il prodotto intelligente) il loop procede attraverso quattro fasi: la fase di connessione tra i dispositivi/processi di quel livello (*connect*), la fase di raccolta dati (*collect*), la fase di analisi *(analyse*) e la fase di reazione (*react*).
La prima fase del loop prevede il raggiungimento della connettività tra le risorse di produzione. Una volta connessi i dispositivi o i processi, o in generale le risorse, la sfida sarà quella di collezionare o raccogliere i dati forniti da queste risorse in una forma usabile (seconda fase). Infatti, ogni industria manufatturiera colleziona dati in differenti formati, tipi, a volte strutturati e a volte no, o non contestualizzati. Questo significa che anche se i dati sono raccolti, spesso non sono usabili.

Una volta collezionati i dati, e ammesso che possano essere interpretati, occorre capire come utilizzarli, ossia capire se sia possibile estrarre informazione che abbia valore per i processi decisionali e l'ottimizzazione dei processi. Questo aspetto concerne l'analisi dei dati che può avvenire in diverse forme, si va da una semplice visualizzazione e analisi, al processamento e analisi (*processing and analytics*) o alla generazione di modelli per la simulazione dei processi manifatturieri lungo tutto il ciclo di vita del processo stesso (*life cycle*). In questo contesto c'è anche la possibilità di usare i dati forniti dal sistema reale come feedback per la costruzione di un modello digitale del sistema manifatturiero (*digital twin*).

Una volta estratta informazione dai dati, segue quindi la fase di reazione (*react*). Questa reazione è intesa come una segnalazione informativa (*reporting* and *informing*) ai sistemi di decisione (effettuabile anche attraverso una semplice dashboard dove sono visualizzate le operazioni dello shop floor e utile al processo decisionale di operatori umani che sono presenti nel loop di produzione, ad esempio). L'opportunità offerta dalla condivisione di queste informazioni è sempre l'ottimizzazione del processo manufatturiero che può realizzarsi anche attraverso la costruzione di modelli predittivi. L'utilizzo di modelli predittivi di processo o modelli predittivi per l'identificazione di guasti consentono l'implementazione di sistemi ancora più autonomi perché possono essere addestrati con le informazioni raccolte.

La sfida è quella di considerare tutte le componenti della pipeline che abilitano il *digital manufacturing*, cioè identificare le zone produttive che possono generare informazioni utili che possono dare valore al business aziendale e consentire l'ottimizzazione dei processi all'interno di tutti i processi dell'organizzazione (scambio dati verticale tra i diversi livelli dell'organizzazione per l'ottimizzazione del sistema globale). Quando si scala questo modello tra tutti i livelli dell'organizzazione il problema dell'integrazione dei dati tra sistemi si fa importante. Si può arrivare sino alla connettività tra i prodotti intelligenti venduti al cliente e l'organizzazione produttiva (relativamente ai dati di uso del prodotto al cliente, se possibile per questioni di privacy) per l'estrazione di valore per l'ulteriore ottimizzazione del processo produttivo.

Per l'integrazione bisogna iniziare dal basso, cioè dai processi che devono essere connessi ai sistemi di rilevamento (*sensing*) e attuazione, cioè tecnologie OT (*Operational technology*) dello shop floor, rappresentate dall'insieme di hardware e software che rileva o causa cambiamenti attraverso il monitoraggio o il controllo diretto di dispositivi fisici e processi. Questi processi sono supervisionati da sistemi di controllo che definiscono il limite per il quale abbiamo requisiti real-time della rete dello shop-floor, cioè la rete industriale che deve essere connessa ad un livello più alto ai sistemi IT. A questo proposito c'è molto lavoro di ricerca in questo ambito per quanto riguarda la sicurezza dei dati nel passaggio tra questi livelli. Questa connessione tra sistemi IT/OT permette di estrarre dati dallo shop-floor e inviarli ai differenti livelli software dell'organizzazione, che possono essere i sistemi operativi del sistema manufatturiero (SCADA, MES, ERP, WMS), database e sistemi informativi che sono impiegati per la manipolazione e la gestione dei dati di produzione, o i modelli digitali che utilizzano il feedback dei dati del processo reale per ottimizzarsi e che servono a progettare il processo stesso.

Il fattore chiave per questa integrazione è che i dati devono essere ingegneristicamente strutturati in maniera tale che possono essere utilizzati. Si definiscono quindi dei modelli di dati (*Data models*) per far si che questi siano consistenti.
Un altro livello che è emergente nella manifattura è quello di come connettere l'organizzazione ai sistemi IT/CLOUD e i relativi servizi che questi mettono a disposizione. Infatti, c'è la tendenza di migrare i software nella piattaforma cloud come servizi, ad esempio i CAD sono oggi componenti di servizi clouds, ma anche i sistemi ERP, MES, PLM (*product lifecycle management*). Una grossa sfida è quella di esternalizzare nel cloud questi sistemi di gestione.

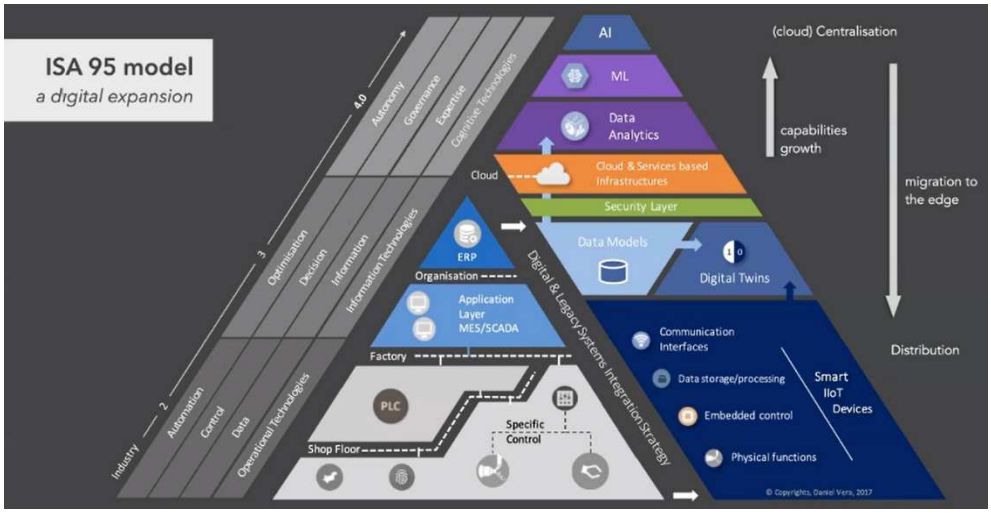


Fig. 1.3.2: Estensione piramide ISA-95 (Source: D.Vera, Warwick Univ.)

In Figura 1.3.2 è mostrato come la tradizionale piramide di automazione ISA-95 di una organizzazione aziendale manifatturiera è estesa dall'utilizzo delle nuove tecnologie. La porzione colorata in blu evidenzia che i dispositivi impiegati nello shop-floor hanno maggiori capacità rispetto alla classica automazione PLC. I PLC attuali offrono molte opzioni di connettività, enorme potenza di calcolo e possono ospitare agenti.
Tutti i modelli dei dati e i digital twins sono costruiti sopra e andiamo verso una piramide con un'operatività orientata ai dati. Si nota che c'è un incremento di capacità/abilità nello shop-floor perché ci sono dispositivi intelligenti, c'è una migrazione di queste capacità verso l'*edge* fino allo shop floor piuttosto che avere un sistema centralizzato che supervisiona l'intero shop floor (le capacità sono *distribuite* tra tutti i dispositivi). La tendenza è quella di provare a incorporare conoscenza (*knowledge*) e intelligenza nei dispositivi embedded che operano in un sistema distribuito, piuttosto che avere un sistema gerarchico e centralizzato.

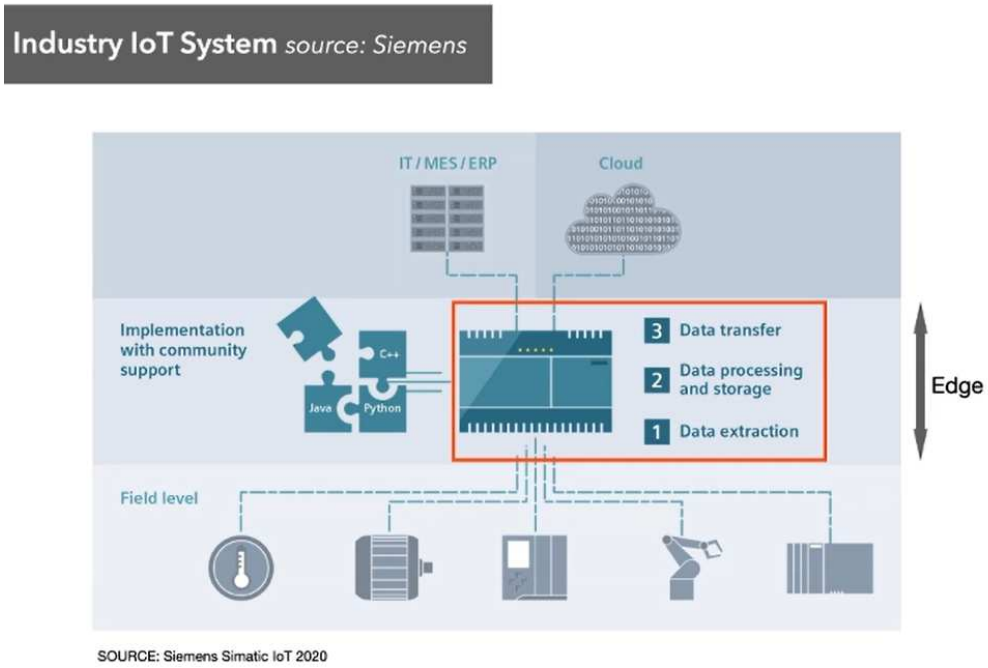


Fig. 1.3.3: *Analytics* fatto da dispositivi Simatic IoT nell'*Edge*

In Figura 1.3.3 è mostrata un'architettura Siemens in cui i dispositivi del livello di campo (*Field level*) sono connessi e inviano dati ad un dispositivo posto nelle immediate vicinanze (*Edge*), non distante dai controllori PLC ad esempio. Questo dispositivo esegue l'estrazione dei dati offerti dai dispositivi del livello di campo, li pre-processa e li invia al cloud dove sono svolte le analisi più onerose.

1.4 CPS (Cyber-Physical-System)

Il CPS è un sistema di elementi computazionali *collaborativi* che controllano entità fisiche e rappresenta uno dei principali strumenti tecnologici abilitanti I4.0.
La differenza tra un componente di un sistema CPS e un componente embedded è che quest'ultimo è costituito dall'oggetto fisico e un computer in qualsiasi forma che ne controlla il movimento o il processo fisico in generale. Un componente embedded non è un componente intelligente o autonomo. Possiamo avere differenti componenti embedded, differenti reti e tecnologie, differenti protocolli di interazione e interfacce, ciò che non abbiamo è una esplicita definizione di *collaborazione* perché i componenti non sono progettati per essere intelligenti, piuttosto sono progettati per avere un computer a bordo. Il concetto di cooperazione è un qualcosa che dobbiamo programmare a più alto livello.
Quindi, se abbiamo un componente embedded (robot+computer) e aggiungiamo una entità/parte software nel computer che si occupa di interagire e collaborare in modo intelligente, in una struttura dove è richiesto un certo pattern comportamentale, con altri componenti dotati delle stesse caratteristiche, otterremo un componente cyber-fisico (CPS).
Quello che è necessario capire, quando si parla di CPS, è che le reti e le tecnologie per la loro comunicazione ci sono già, manca uno sforzo nel definire e standardizzare i protocolli di interazione comuni per la collaborazione.
La collaborazione è normalmente incorporata nei protocolli di interazione, i CPS sono autonomi, localmente intelligenti e capiscono le funzioni di altri CPS in rete.

Un altro elemento abilitante di I4.0 è il *Digital Twin,* spesso confuso con il CPS ma che non è la stessa cosa.

Fonte wikipedia verificata: "il *Digital Twin* si riferisce ad una replica digitale di una risorsa fisica, un processo e sistema, che può essere usato per vari scopi. Il gemello digitale integra intelligenza artificiale, *Machine Learning* e *software Analytics* con i dati forniti dalla risorsa fisica per creare modelli di simulazioni digitali che si aggiornano quando la controparte fisica cambia. Un *Digital Twin* impara continuamente e si aggiorna da sorgenti multiple per rappresentare lo stato quasi real-time, la sua condizione di lavoro o posizione. Questo sistema di apprendimento impara da sé stesso utilizzando dati da sensori che veicolano vari aspetti della sua condizione operativa, da umani esperti come gli ingegneri che hanno una profonda conoscenza del dominio industriale, da altre macchine simili, dai sistemi che fanno parte del suo ambiente. Il digital twin integra anche i dati storici dall'uso della macchina per la *fattorizzazione nel modello digitale*."

A volte si fa confusione tra un sistema CPS e il *Digital Twin*, ma la simulazione del Digital Twin avviene dai dati reali che colleziona in real-time (utilizzati per predire guasti ai macchinari ad esempio) e potenzialmente può mandare dati al sistema reale. Quando parliamo di Digital twin parliamo di una copia virtuale della risorsa fisica, il CPS non è una copia virtuale della risorsa, è la risorsa intelligente stessa (processo fisico + intelligenza).

Quando parliamo di collaborazione parliamo di collaborazione tra CPS, cioè tra i processi *reali,* non ci sono copie.
Un digital twin colleziona dati reali per fare predizioni sullo stato del processo fisico che simula e può mandare un riscontro al processo fisico ma non obbligatoriamente.
Un sistema convenzionale può avere digital twins ma questo non necessariamente implica o preclude la collaborazione intelligente tra componenti. Tuttavia, non c'è nessuna ragione per cui i concetti di CPS e Digital Twin non possano essere armonizzati in futuro.

1.5 Fabbriche e prodotti intelligenti

I prodotti intelligenti (*prodotti smart*) generano *valore* nella rete dei servizi ai quali sono connessi e possono generare valore all'interno della fabbrica stessa. Questo valore è rappresentato dalle informazioni utili (*knowledge* = conoscenza = dato utile) che il prodotto invia alla fabbrica come *feedback* durante il suo ciclo d'uso.
È importante capire che gli strumenti (*tools*), le macchine e i componenti *smart* utilizzati in una fabbrica intelligente (*smart factory*) possono essere stati prodotti da altre fabbriche *smart*, quindi ci sono delle auto-similarità nella progettazione, se una *smart factory* produce un prodotto *smart* e questo prodotto è utilizzato da un'altra *smart factory* per produrre un altro prodotto *smart* abbiamo un prodotto finale che è la composizione di prodotti *smart.*
Questa tipologia di produzione ha una struttura simile a quella frattale, ed è importante capirlo quando parliamo di produzioni a livello globale [3], più formalmente si può parlare di modelli di sistemi di produzione cyber-fisici (CPPS) [4].
Il prodotto genera conoscenza che può essere condivisa con altri sistemi e fornisce un riscontro sul sistema di produzione cyber-fisico o chi ha il diritto di sfruttare questa conoscenza. In futuro il valore del prodotto sta nella conoscenza che è in grado di fornire la fabbrica, tecnicamente generata dall'uso del prodotto da parte del consumatore che essendo il proprietario potrebbe bloccare questo trasferimento di conoscenza.

Non necessariamente un insieme di prodotti intelligenti è una fabbrica intelligente (CPPS), esiste la seguente classificazione:

Livelli non CPPS:

Livello -1

Ci sono sistemi produttivi in cui è praticamente impossibile apportare evoluzioni (ad esempio per impedimenti di legge).

Livello 0:

Il sistema non è progettato per una evoluzione *smart* o modulare, servirebbero grossi sforzi di ingegnerizzazione per renderlo *smart* e *adattivo*, sarebbe più conveniente partire da zero che renderlo smart. *Questo è un problema perché molti sistemi oggi lavorano in questo modo.*

Livello 1:

Il sistema è meccanicamente modulare ma non logicamente modulare, cambiamenti del sistema richiedono uno stop del sistema, cioè certe parti meccaniche possono essere sostituite con altre ma non c'è una flessibilità tale da non consentire lo stop del sistema.

Livello 2:

Il sistema è meccanicamente modulare e un po' modulare dal punto di vista logico. Le parti intercambiabili sono molto simili ma occorre sempre fermare il sistema.

Componenti collaborativi e intelligenti

Livello 3:

Modularità meccanica e logica (capacità di ricerca dinamica della logica). Il sistema ha funzioni minime di orchestrazione. Si passa dal paradigma di *riprogrammazione* al paradigma della *riconfigurazione*, c'è una lingua comune che specifica cosa fa il sistema e il sistema comprende questa lingua. La riconfigurazione riduce gli errori e generalizza il sistema.

Livello 4:

Alta modularità, il sistema interpreta autonomamente la sua struttura quando avvengono cambiamenti (la struttura del sistema è progettata per modificarsi, non modifichiamo solo codice, la cosa importante di questi sistemi è che se introduciamo uno o più componenti fisici il sistema interpreta il cambiamento e si riconfigura autonomamente). Le risorse sono dinamicamente allocate e il sistema si auto-orchestra. Parliamo di sistemi con struttura cyber fisica.
Il sistema di livello 4 rappresenta il focus della ricerca odierna.

Il futuro:

Livello 5:
Sono sistemi altamente modulari, autonomi e capaci di pianificare la struttura del sistema stesso.
Sono sistemi in grado di acquisire in modo autonomo nuove funzionalità da diversi "fornitori di funzionalità (marketplace)" per requisiti futuri.

1.6 Gli agenti

Architetture di sistemi CPS possono essere basate sul concetto di *agenti.* Gli agenti non sono importanti grazie alle tecnologie con cui sono implementate (JADE, IEC 61499 etc..), il loro valore sta nella filosofia utilizzata per esprimere i sistemi, che è quella di pensare al sistema come un insieme di moduli (agenti) che controllano parti del sistema stesso e come questi moduli interagiscono. In letteratura sono presenti moltissime definizioni di agente, tra cui:

- Un agente è qualcosa che agisce [5].
- Un agente razionale è quello che agisce per ottenere il miglior risultato, e laddove ci fosse incertezza, il miglior risultato aspettato [5].
- Un agente è qualcosa che percepisce l'ambiente circostante attraverso sensori e agisce sull'ambiente attraverso attuatori [6].
- Un agente è un sistema computerizzato in grado di compiere azioni indipendenti a favore del suo proprietario, cercando di capire cosa deve essere fatto per raggiungere l'obiettivo [6]

Molti ricercatori convengono su alcune caratteristiche fondamentali che gli agenti devono soddisfare, e cioè:

- Autonomia
- Abilità sociali
- Razionalità
- Reattività
- Proattività
- Adattabilità

Un agente è un'entità autonoma con proprietà razionali, quindi, ha una certa libertà nel prendere decisioni, si dovrebbe considerare l'autonomia come una condizione necessaria ma

non sufficiente per un agente, perché se consideriamo che ogni pezzo di software è autonomo, allora ogni cosa è autonoma, e si perde la capacità di razionalizzare o prendere decisioni. L'agente non decide in base ad una pura esecuzione di un set di "if-then-else", ma a seguito di una certa sua analisi fatta dell'ambiente, questa è la differenza tra un sistema basato su agenti e un sistema governato da un set di regole.

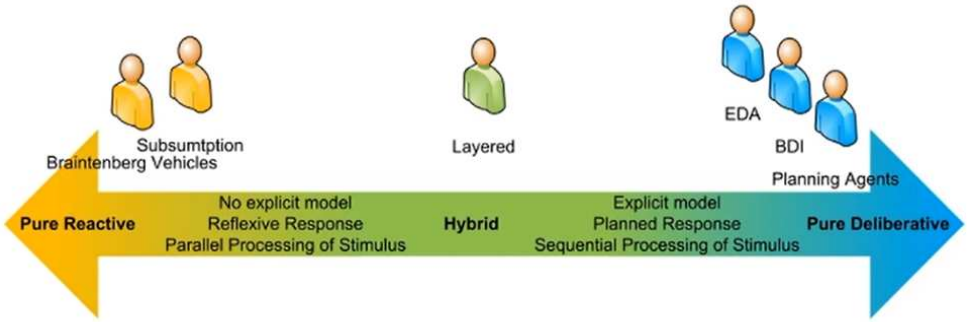


Fig. 1.6.1: I diversi approcci per la progettazione comportamentale dell'agente

Le caratteristiche degli agenti sopracitate hanno dato il via ad una serie di approcci nella progettazione comportamentale di un agente, mostrati in Figura 1.6.1. Si va dal comportamento reattivo puro in cui l'agente non ha nessuna proprietà "cognitiva", ai deliberativi puri governati da processi cognitivi complessi. Nel secondo caso si utilizza il classico approccio AI simbolico, che ha bisogno di un modello esplicito e completo dell'ambiente esterno per operare (traduce l'ambiente in un set di simboli) e la logica del ragionamento è basata sul set di simboli. In questo caso l'agente osserva il mondo, lo processa, pianifica ed esegue l'azione ottima che verifica enunciati logici. Non è un approccio molto buono per un mondo variabile e incerto, ciò implicherebbe un nuovo modello per ogni configurazione dell'ambiente in tempo reale.
L'approccio BDI (*Belief-Desire and Intention*) è più flessibile del precedente, l'agente ha la sua visione del mondo esterno (*Belief*) e ha degli obiettivi da raggiungere (*Desires*) per i quali deve concepire una strategia in un dato istante di tempo e contesto dell'ambiente esterno. Le intenzioni (*Intentions*) fanno parte dell'insieme delle azioni che l'agente decide di eseguire per soddisfare un dato obiettivo [6].

Nel comportamento reattivo puro si considera invece che "il mondo è il miglior modello di sé stesso" [7] quindi l'agente non ha un modello del mondo ma percepisce il mondo costantemente tramite sensori e reagisce ogni volta che arriva nuova informazione. Il comportamento globale del sistema di agenti reattivi è una proprietà *emergente* del sistema globale, dato dall' interazione tra i comportamenti dei singoli agenti. L'intelligenza del sistema è una proprietà emergente che è visibile guardandolo dall'alto, non guardando il singolo agente, un po' come quando si guarda il moto di uno stormo di uccelli.

1.7 Agenti industriali – Architettura a due layers

In ambito industriale un agente può fare parte di un sistema multi-agente MAS (*Multi-Agent System*).
Un sistema MAS (Figura 1.7.1) è usato in ambiente industriale perché è un approccio consistente e interessante per la progettazione di sistemi di controllo distribuiti su larga scala dato che offrono intelligenza, flessibilità, proprietà di auto-adattamento (*self-adptation*), auto-organizzazione (*self-organization*), resilienza e robustezza.

I MAS usano i principi di progettazione che sono utilizzati per gli agenti software ma in questo caso devono essere applicati in ambiente industriale, il ché richiede flessibilità e il soddisfacimento di vincoli temporali.
Gli agenti devono interagire di solito con un hardware fisico (controparte fisica) e ciò ne aumenta la complessità di realizzazione pratica.

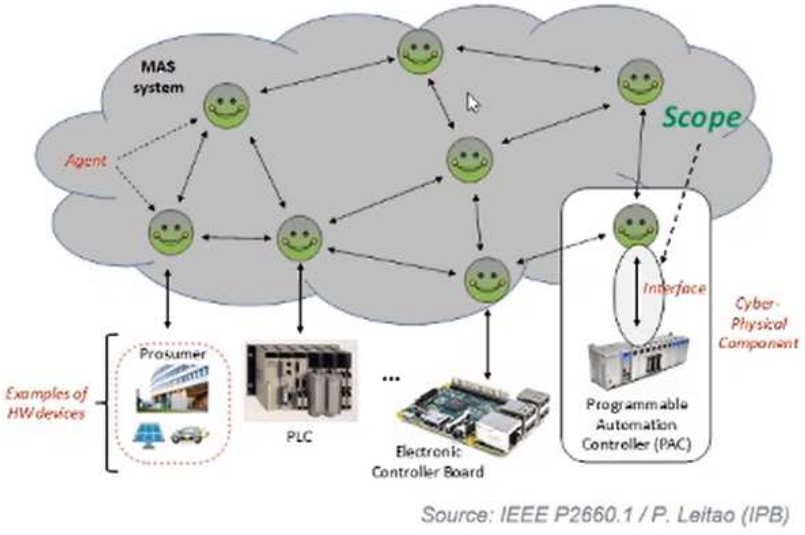


Fig.: 1.7.1: rete MAS e componente CPS

Un tipico modello di architettura per agenti industriali è mostrato in Figura 1.7.2. A livello più basso si trova il componente fisico, cioè un dispositivo automatico come un robot, un nastro trasportatore, una macchina utensile etc. Tipicamente abbiamo anche un livello reattivo (*reactive layer*) in grado di soddisfare requisiti *real-time*, costituito da un controllore (ad esempio un *PLC*) che controlla il componente fisico (*robot*).

A livello più alto abbiamo il sistema di controllo basato sugli agenti che prende decisioni strategiche di produzione e pianifica o riconfigura il controllo reattivo di basso livello per l'ottimizzazione del sistema globale. I due livelli devono interagire per raggiungere un obiettivo comune. Questo agente potrebbe comunicare con sistemi di supervisione (*supervisory tools*), sistemi SCADA, sistemi di visualizzazione (*HMI*), o sistemi di pianificazione (*MES*) dell'organizzazione aziendale.

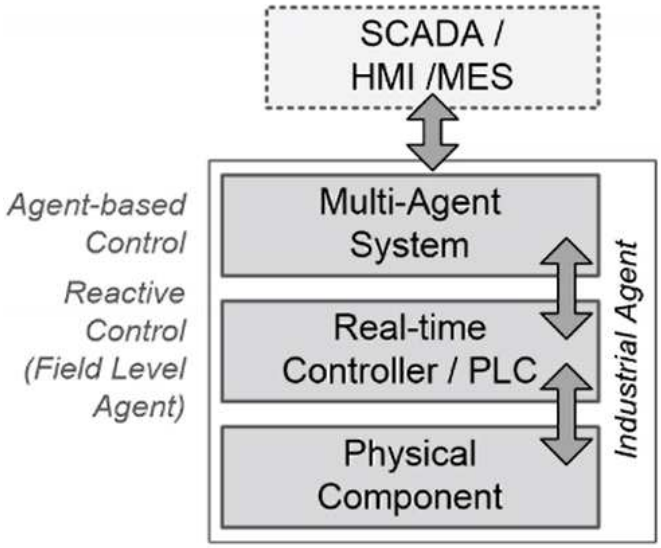


Fig. 1.7.2: Modello dell'agente industriale

L'agente software può essere implementato localmente nella stessa scheda del controllore reattivo (accoppiamento *embedded*) o essere ospitato in remoto [8]. In entrambi i casi, l'ente di standardizzazione FIPA (*Foundation for Intelligent Physical Agent*) che definisce le linee guida riguardo l'implementazione di agenti software e la loro l'interazione, non ha deliberato specifiche riguardo l'interazione (interfaccia) verticale tra l'agente software e il controllo reattivo che di norma è realizzata da software proprietari, quindi non interoperabili. Una soluzione per l'interoperabilità potrebbe essere rappresentata dallo standard di comunicazione OPC-UA (*Open Platform Communications-Unified Architecture*) che offre un modello di dati e un modello informativo completi. Un contributo a questa definizione potrebbe anche essere fornito dallo standard IEEE P2660.1 che si occupa di definire raccomandazioni e pratiche per gli agenti industriali.

1.8 Gli standard per il controllo reattivo

IEC 61131 Programmable Logic Controller (PLC)	IEC 61804 Distributed control System (DCS)
Freely programmable	Configurable
Centralized	Distributed

Fig. 1.8.1: Proprietà degli standard diffusi in industria

Lo standard IEC 61131-3 [9][10] è molto diffuso in industria per la programmazione dei PLC ma definisce un'architettura software orientata ad un modello centralizzato (Figura 1.8.1). Lo standard IEC 61804 definisce un'architettura distribuita, permette la configurazione di vari blocchi funzionali (FB – *Function Block*) ma questi sono oggetti predefiniti.

Lo standard IEC 61499 [11] definisce una architettura sempre basata su FB ma combina le caratteristiche di 61131-3 e 61804 per offrire un modello di controllo distribuito con FB personalizzabili, quindi flessibilità e possibilità di riutilizzo del codice. Inoltre, offre caratteristiche molto interessanti per quanto riguarda l'interazione con *agenti software,* tra cui, molto importante è la *configurazione dinamica* delle FB. Tuttavia, è uno standard ancora poco utilizzato in industria, esistono applicazioni industriali di nicchia, e questo perché serve molto tempo prima che un cambio di paradigma venga percepito in industria dati gli ingenti investimenti fatti per le configurazioni attuali, e l'industria richiede stabilità. Lo standard IEC 61499 è comunque un modello molto promettente vista la diffusione via via crescente di dispositivi industriali IoT nello *shop-floor* di sistemi di produzione *Cyber*.

1.9 Standard IEC 61499

Lo standard IEC 61499 definisce specifiche per l'automazione adattiva distribuita e ha origine dai sistemi olonici degli anni '90 quando si cominciava a parlare di manifattura agile e requisiti di adattabilità, flessibilità e distribuzione del controllo [12] [13]. Si propone come modello di riferimento per l'implementazione di un controllore reattivo associato ad un componente fisico dello *shop-floor*. Se anche un agente software è incorporato in questo controllore si parla di un sistema CPS.

Lo standard IEC 61499 propone una architettura aperta e standardizzata dei blocchi funzionali (BF o FB-Function Block) e offre una certa flessibilità data dalla possibilità di riconfigurazione basica del codice di controllo che può essere modificato *run-time*. Quando

si parla di architettura aperta in automazione ci si riferisce ai seguenti requisiti che devono essere soddisfatti:

- *Portabilità*: Differenti tools di differenti produttori sono in grado di interpretare e accedere le librerie dei componenti e le applicazioni (ad oggi la portabilità tra differenti produttori non esiste e IEC 61131 non è portabile in nessun modo).
- *Configurabilità*: differenti tools/produttori differenti possono configurare i dispositivi.
- *Interoperabilità*: dispositivi di diversi produttori possono interagire nel controllo distribuito o nel *monitoring*.

IEC 61499 REFERENCE MODEL

Il modello di riferimento dello standard 61499 è descritto principalmente in tre parti:

Parte 1: Architettura (*Architecture*) [11] [14]
Parte 2: Requisiti degli strumenti software – (*Software tools Requirements*) [15]
Parte 4: Regole per la conformità dei profili – (*Rules for compliance Profiles*)

La prima parte dello standard definisce l'architettura e gli elementi della modellazione che si basa principalmente sul concetto di blocco funzionale (FB), l'elemento atomico la cui funzionalità non più essere distribuita.

Modello del blocco funzionale

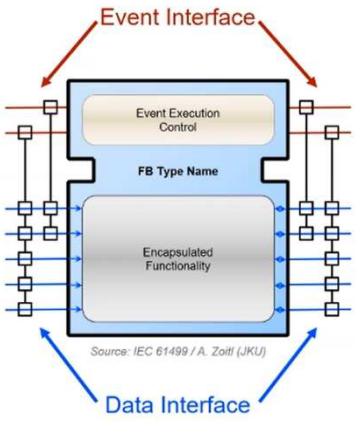


Figura 1.9.1: Modello del blocco funzionale di base (BFB) (Fonte: A. Zoitl – Johannes Kepler University)

Per il modello del blocco funzionale di base (Fig. 1.9.1) valgono le seguenti osservazioni:

- Le classiche FB sono estese con una interfaccia ad eventi
- L'esecuzione delle FB è guidata da eventi (event-driven)
- I tipi di dati sono ereditati da IEC 61131-3
- Il focus è l'incapsulamento e il riuso del codice
- Non è previsto l'uso di variabili globali
- L'accesso all'hardware e alle comunicazioni di rete è eseguito tramite speciali FB dette SIFB (*Service Interface Function Block*)

Lo standard IEC 61499 eredita i tipi e i linguaggi di programmazione ad alto livello definiti in IEC 61131-3. Il blocco funzionale dello standard IEC 61131-3 è però esteso con una interfaccia per gli eventi ed è ridefinito il modello di esecuzione del blocco. In un PLC con software IEC 61131-3 l'esecuzione di un FB avviene all'interno di un tempo di ciclo predefinito, ad ogni istante di campionamento gli ingressi dalle periferiche sono resi disponibili a tutti i blocchi funzionali dell'applicazione che sono eseguiti sequenzialmente nel tempo di ciclo. Al termine di questo tempo i risultati sono resi disponibili in uscita. In IEC 6149 l'esecuzione è controllata dagli eventi di ingresso che sono poi gestiti dal Modulo di Controllo di Esecuzione degli Eventi interno (EEC – *Event Execution Control*). Lo standard supporta quindi l'esecuzione asincrona dei blocchi funzionali e delle reti di blocchi funzionali ma con un FB dello standard IEC61499 è possibile emulare anche la modalità sincrona (ad esempio lo scan-cycle di un PLC).

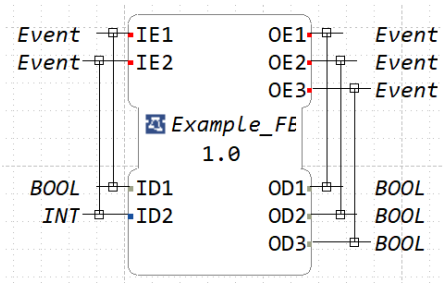


Fig. 1.9.2: Blocco funzionale di base

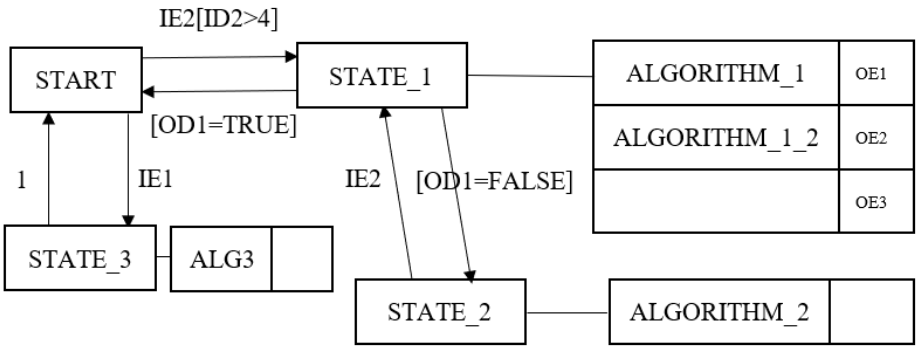


Fig. 1.9.3: Grafo del controllo di esecuzione - esempio

In Figura 1.9.2 è mostrata la rappresentazione di un semplice blocco funzionale di base con un'interfaccia che presenta due eventi di ingresso IE1, IE2 associati rispettivamente alla variabile dato in ingresso ID1 di tipo booleano e alla variabile dato in ingresso ID2 di tipo intero. Sono presenti tre eventi di uscita OE1, OE2, OE3 associati rispettivamente a tre variabili di uscita di tipo booleano, rispettivamente OD1, OD2, OD3.
Le parti algoritmiche del blocco funzionale di Figura 1.9.2 sono incapsulate all'interno del blocco e sono elaborate nella macchina a stati dell'ECC (Figura 1.9.3) che definisce il comportamento del blocco funzionale. La macchina a stati è descritta da un grafo denominato ECC (*Execution control Chart*). Le variabili interne e i dati di ingresso/uscita possono essere letti dall'ECC, sono accessibili per la modifica nello scope degli algoritmi e sono ritenute internamente nel passaggio tra i vari stati della macchina.
Un blocco funzionale è attivato dall'arrivo di un nuovo evento in ingresso. In sequenza, succede che:

1) Arriva un evento in ingresso.
2) Il dato in ingresso associato all'evento è campionato e aggiornato per uso interno. Internamente, i valori degli altri ingressi non associati all'evento mantengono il vecchio valore.
3) L'evento è passato a ECC.
4) Sono eseguite le funzionalità di ECC, cioè transizioni tra stati e il lancio degli algoritmi.
5) Al termine degli algoritmi le uscite sono aggiornate con nuovi valori.
6) Un evento è posto in uscita e i dati ad esso associati diventano validi per la lettura da parte dei dispositivi a valle.

In riferimento alla figura 1.9.3, per il componente ECC vale il seguente comportamento: il sistema ECC parte dallo stato *start* attivo. La transizione tra stati può avvenire a seguito del verificarsi di un evento in ingresso, un evento con un'espressione booleana, a seguito di una espressione booleana, oppure incondizionatamente (presenza di un 1 sul grafo della

transizione). All'arrivo dell'evento IE2 è campionato l'ingresso ID2 e valutata la condizione di transizione.
La transizione allo stato *state_1* avviene se l'espressione booleana associata alla transizione è vera. In questo caso, se il dato ID2 è maggiore di 4 avviene la transizione a *state_1* che diventa lo stato attivo attuale. L'attivazione di *state_1* eccita in sequenza le elaborazioni *Algorithm_1* e *Algorithm_1_2*.
Al termine degli algoritmi sono posti in uscita gli eventi OE1, OE2, OE3 e aggiornate le variabili di uscita OD1, OD2, OD3. L'attivazione degli algoritmi e la generazione degli eventi di uscita è comunemente denominata fase di *azione.*
Al termine della fase di azione, nello stato *state_1* si rivalutano le espressioni booleane delle transizioni. Se OD1 è vera si attiva lo stato *start* altrimenti è attivato lo stato *state_2*. Parte quindi l'elaborazione *Algorithm_2* senza produzione di eventi.
È riattivato lo stato *state*_1 con l'arrivo dell'evento EI2. Dallo stato *state_3* si riattiva lo stato *start* incondizionatamente.

Un modello con blocchi funzionali consente l'incapsulamento e il riuso del codice. Speciali FB denominate SIFB consentono di gestire i servizi di comunicazione con i dispositivi dello standard ma anche di parlare con dispositivi che non ne fanno parte (supervisori, agenti o altri), in questo caso incapsulano i *drivers* o i protocolli per queste comunicazioni.
Per l'aggregazione di funzionalità esiste il blocco funzionale composito (CFB) in cui più FB possono essere aggregate a formare un'unica FB.

Lo standard definisce una architettura granulare che si compone dei seguenti modelli:

- Modello di Sistema (*System model)*
- Modello Applicazione (*Application model)*
- Modello del Dispositivo (*Device model*)
- Modello della Risorsa (*Resource model*)
- Modello di Gestione del Dispositivo (*Device management model*)

Modello di Sistema

In Figura 1.9.4 sono mostrati i componenti considerati nei modelli che costituiscono l'architettura definita dallo standard.
In particolare, il Modello di Sistema prevede che i dispositivi (*Devices*) siano connessi tramite una generica infrastruttura di comunicazione (*Inter-Devices communication links*). I dispositivi possono essere PLCs, computers, dispositivi embedded etc. Lo standard cerca di essere il più generale possibile per cui non definisce i links delle comunicazioni di rete tra dispositivi, né la tipologia, né i protocolli utilizzati, ogni segmento di comunicazione di rete può rappresentare qualsiasi tipologia di rete la cui interfaccia deve essere incapsulata in

speciali SIFB. Si sceglierà il tipo di protocollo di comunicazione che sarà in grado di soddisfare i requisiti dell'applicazione, in termini di tempi di risposta, sicurezza etc.

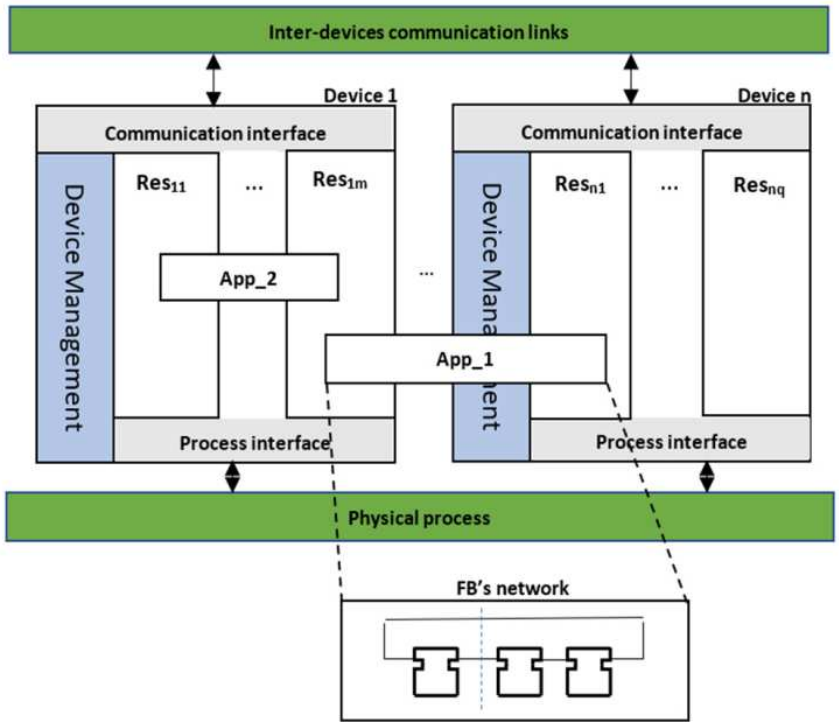


Figura 1.9.4: IEC 61499 – modello generale

Modello Applicazione

Il Modello Applicazione definisce l'applicazione di controllo (che può essere distribuita tra vari dispositivi in base ad un modello di distribuzione) come un framework di blocchi funzionali (FBs network di Figura 1.9.4) connessi in modo opportuno per la realizzazione di una certa funzionalità di controllo, diagnostica etc. L'applicazione può controllare qualsiasi dispositivo tipico di uno shop floor come nastri trasportatori o robots ma è sviluppata indipendentemente dall'hardware che si avrà a disposizione. Ciò definisce un approccio di progetto orientato alla funzionalità dell'applicazione anziché verso il componente fisico offrendo quindi una certa flessibilità al problema del controllo.

Modello del dispositivo

Il Modello del Dispositivo definisce il dispositivo (Figura 1.9.4) come il contenitore di risorse (*Resources*), di una interfaccia di rete (*Communication interface o Network Interface*) che permette l'interazione con altri dispositivi (anche non IEC 61499) e di una interfaccia di processo (*Process Interface*) per la comunicazione con dispositivi di I/O analogico o digitali o con PLCs, ad esempio.

Modello della risorsa

La risorsa è l'ambiente per l'esecuzione del framework dei blocchi funzionali che costituisce l'applicazione o una porzione di questo framework ed eredita dal dispositivo l'interfaccia di rete e l'interfaccia di processo. Per le comunicazioni di rete e di processo abbiamo bisogno di particolari FB, dette SIFB, che incapsulano il codice di interfaccia verso specifici protocolli di rete o l'hardware del processo fisico (attuatori, sensori etc). Tipicamente il *vendor* di un dispositivo hardware può fornire il proprio supporto per la progettazione della SIFB fornendo i propri drivers o direttamente la SIFB stessa. È desiderabile dal punto di vista dell'ingegnere software che l'inserimento di queste interfacce possa essere fatto in modo automatico dal tool software utilizzato per l'applicazione. Inoltre, potremmo anche utilizzare una SIFB per incapsulare lo stack di comunicazione con un agente software per la configurazione on-the-fly dei parametri dell'applicazione. Lo standard IEC 61131 mappa una risorsa con una CPU, cioè una risorsa di computazione, per lo standard IEC 61499 la risorsa è invece un concetto più alto per il quale una CPU può ospitare più risorse, non c'è un mapping diretto CPU->risorsa.
Porzioni di un framework di FB che costituisce un'applicazione di controllo globale possono essere allocate (distribuite) in differenti dispositivi che concorrono al controllo e in diverse risorse di uno stesso dispositivo.
In Figura 1.9.4 è mostrata l'applicazione App_1 costituita da una rete FB suddivisa tra due risorse, *Res1m* e *Resn1* ospitate in due dispositivi diversi, *Device 1* e *Device n*.

La risorsa è responsabile di diversi compiti (Figura 1.9.5):

- fornisce la corretta schedulazione degli algoritmi incapsulati in un FB e attivati dagli eventi di ingresso.
- conserva i valori delle variabili interne delle FB.
- propaga eventi e dati tra i blocchi funzionali all'interno della risorsa e fornisce servizi di comunicazione verso risorse remote attraverso l'interfaccia di comunicazione.

- mappa le richieste di lettura/scrittura verso l'interfaccia di comunicazione (*Communication interface o Network Interface)* o verso l'interfaccia di processo (*Process Interface*) attraverso le SIFB (blocchi funzionali A,D e B,E rispettivamente di Figura 1.9.5).

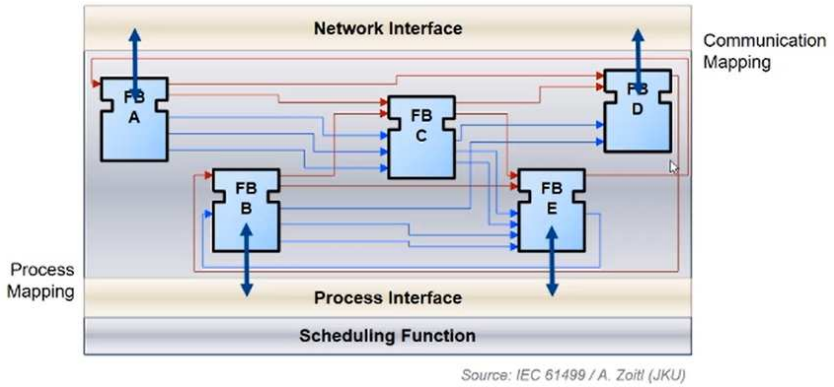


Figura 1.9.5: Risorsa IEC 61499 di un componente

Per quanto riguarda la schedulazione delle FB lo standard definisce un modello di esecuzione delle FBs che da comunque dei gradi di libertà riguardo le modalità implementative rispetto ai principi generali che sono stati stabiliti dal modello, quindi il comportamento di un'applicazione può differire tra le diverse implementazioni, sostanzialmente bisogna tenere a mente che ogni ambiente software IEC 61499 implementa il modello di esecuzione secondo una propria semantica [17] che può differire da quella utilizzata da altri strumenti per ragioni prestazionali etc. Cenni sulle diverse semantiche implementative e le caratteristiche dei principali strumenti di progettazione IEC 61499 sono descritti in Appendice.

Modello di gestione del dispositivo

Lo standard IEC 61499 definisce anche un modello di gestione del dispositivo (*Device Management Model*) (Figura 1.9.4) mappato su una risorsa speciale che fornisce al sistema di automazione caratteristiche di riconfigurabilità e adattabilità. Questo gestore è utilizzabile da altri dispositivi della rete (ad esempio un supervisore con privilegi, un tool, un agente) per la riconfigurazione in fase di esecuzione delle reti FB del dispositivo. Sostanzialmente è possibile modificare a run-time il codice di controllo, generare nuovi FB ed eliminarne altri. Questa caratteristica dello standard lascia aperta la possibilità per l'*autonomic computing* (calcolo autonomo).

1.10 Architettura RMAS

L'architettura RMAS (*Relational Model Multi-Agent System*) è stata introdotta nell'articolo [18] come evoluzione di un precedente *framework* database-centrico, basato su eventi e paradigma di comunicazione *Publish/Subscribe* [19]. L'obiettivo era quello di introdurre il progetto e l'impiego di elementi e componenti distribuiti olonici compatibili con i requisiti dell'automazione industriale. Per questi motivi è interessante fare delle valutazioni sul rapporto esistente tra l'architettura RMAS e i modelli proposti per il controllo in automazione, tra i quali IEC 61499 che, orientato alla distribuzione e integrazione in strutture CPS per la parte real-time del controllo, offre approcci di modellazione e caratteristiche che si relazionano a RMAS e che rappresentano uno spunto per integrazioni future tra le due piattaforme non solo concettuali. Di seguito è introdotto RMAS e saranno descritte alcune delle relazioni con componenti IEC 61499.

L'architettura RMAS rappresenta una base di concetto consistente per la configurazione dinamica, la programmazione, la simulazione, la manutenzione e l'implementazione di sistemi multi-agente (MAS) e sfrutta le proprietà del modello relazionale come base per la programmazione logica nei CPS [20]. Il sistema è stato progettato con l'idea di incorporare *reasoning* e controllo CPS attraverso la combinazione di linguaggi che si definiscono *host languages* (imperativi, object-oriented, linguaggi funzionali e linguaggi di manipolazione di database (ad esempio SQL).

Il nucleo base dell'architettura è la definizione di una unità RMAS germinale, che può essere considerata come il contenitore del *genotipo* di RMAS, come mostrato in Figura 1.10.1.

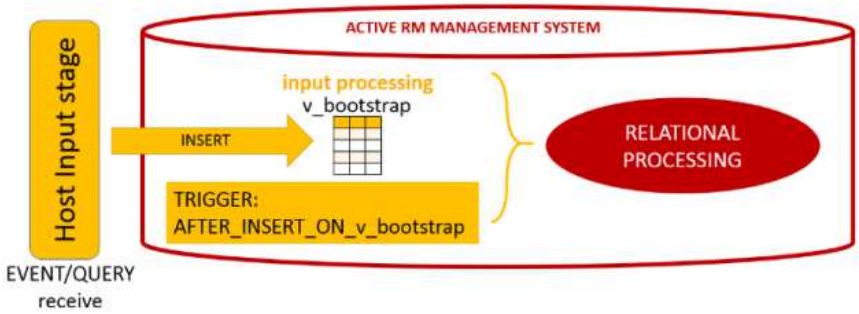


Figura 1.10.1: Genotipo dell'unità RMAS

Questa unità ha inizialmente solo disposizioni minime per essere contattata, configurata e programmata, ovvero consta di:

- una fase (*stage*) di ingresso, tipicamente attuata mediante un linguaggio *host* (cioè un linguaggio ospite, come ad esempio il codice per le connessioni socket di Internet e per l'interpretazione del messaggio in ingresso all'unità)
- *v_bootstrap*, variabile di database
- il trigger AFTER_INSERT_ON_V_bootstrap, che esegue il codice del linguaggio host o del linguaggio del database dopo che un nuovo valore viene inserito nella variabile relazionale *v_bootstrap* di una tabella del database.

Questo meccanismo di bootstrap è costituito da una variabile del database e un trigger che è in grado di innescare ulteriori elaborazioni relazionali. Si osserva che la creazione, la cancellazione o modifica dello schema del database è considerato parte dell'elaborazione relazionale.
Quando arriva un nuovo valore da attribuire alla variabile *v_bootstrap*, questo è inserito nella tabella delle variabili. Questo inserimento è il trigger per l'interpretazione o l'esecuzione di qualsiasi logica connessa al valore della variabile, logica che è espressa nel linguaggio *host* o nel linguaggio del database. In questo modo, un valore di variabile trasmessa dalla query INSERT può eseguire manipolazioni tipiche (ad esempio creare o eliminare oggetti o tabelle del database, inserire o aggiornare contenuti delle tabelle) o, attraverso un linguaggio *host* può agire su altre parti del software o su parti hardware.
Questo significa che un'unità RMAS può essere modificata o meglio "riempita" di contenuti da un oggetto esterno che possiamo chiamare genitore, cioè un altro software o qualsiasi altro sistema. Questo concetto rende RMAS una struttura completa (nel senso di di Turing) fatta da una combinazione di un linguaggio di manipolazione del database e linguaggi di programmazione *host*. In Figura 1.10.2 è espressa la struttura completa di una unità RMAS *full-Fledge.*

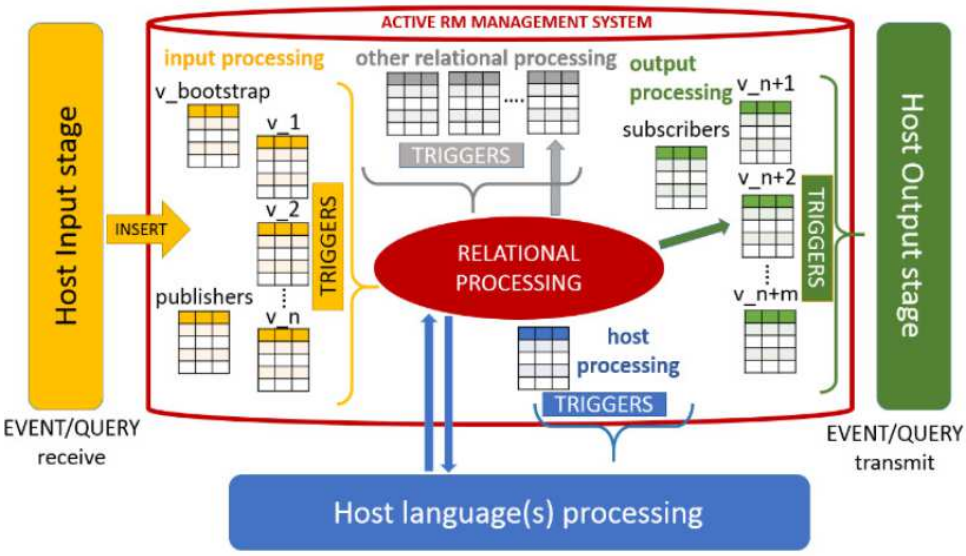


Figura 1.10.2: unità RMAS Full-Fledged

Tenendo presente il funzionamento di una unità germinale, possiamo dedurre che l'RMAS completo Full-Fledged di Figura 1.10.2 è una architettura che consente di ricevere query dallo stadio host di ingresso (giallo), di comunicare dati ai *subscribers* remoti attraverso una fase host di uscita (verde), comunicare con processi software o hardware attraverso un processamento eseguito da un linguaggio host (blu). Internamente tutti i processi sono processi relazionali.
La scelta di una struttura relazionale per RMAS consente la compatibilità con i patterns di comunicazione *publish/subscribe* tipico del networking, con il modello di computazione ad attori (*actor model*) e l'elaborazione guidata dagli eventi (*event driven*). Queste caratteristiche sono anche condivise con quelle definite dallo standard IEC 61499.

Relazioni IEC 61499 e Architettura RMAS

Come descritto in [21], possiamo sostenere che alcuni dei modelli basici dello standard IEC 61499 sono compatibili con i concetti RMAS. In particolare, qui si richiama il lavoro di concetto svolto mappando RMAS nel modello della risorsa e nel modello del blocco funzionale definiti in IEC 61499.

Nello standard IEC 61499 la risorsa è l'elemento che gestisce l'esecuzione della rete di FB ospitata (l'applicazione) e che consente all'applicazione di comunicare con il processo fisico e con la rete. In Figura 1.10.3 è mostrata una risorsa contenente un blocco funzionale FB1 e le SIFB necessarie alla comunicazione di rete e di processo. Si nota sempre in Figura 1.10.3 che la parte algoritmica incapsulata in FB1 è stata utilizzata per incorporare il processo relazionale di RMAS e sappiamo anche che le tabelle e gli oggetti del database, con i trigger associati, possono essere considerati equivalenti a fasi di programmazione logica o algoritmica. Questa è una proprietà che è stata dimostrata e discussa in [20].

L'FB1 esegue questa parte quando arriva una query sotto forma di evento (EI1 o EI2 in Figura 1.10.3) trasportando alcuni dati (il testo della query, ingressi QI1 o QI2 in Figura 1.10.3). Il modello *event-driven* di IEC 61499 è quindi conforme con il paradigma *event-driven* per lo scambio di dati verso RMAS e da RMAS. Ciò si ottiene semplicemente associando un evento all'esecuzione di una query di aggiornamento nella sezione di processamento relazionale di RMAS (per esempio le query INSERT, UPDATE, CREATE e DELETE in SQL).

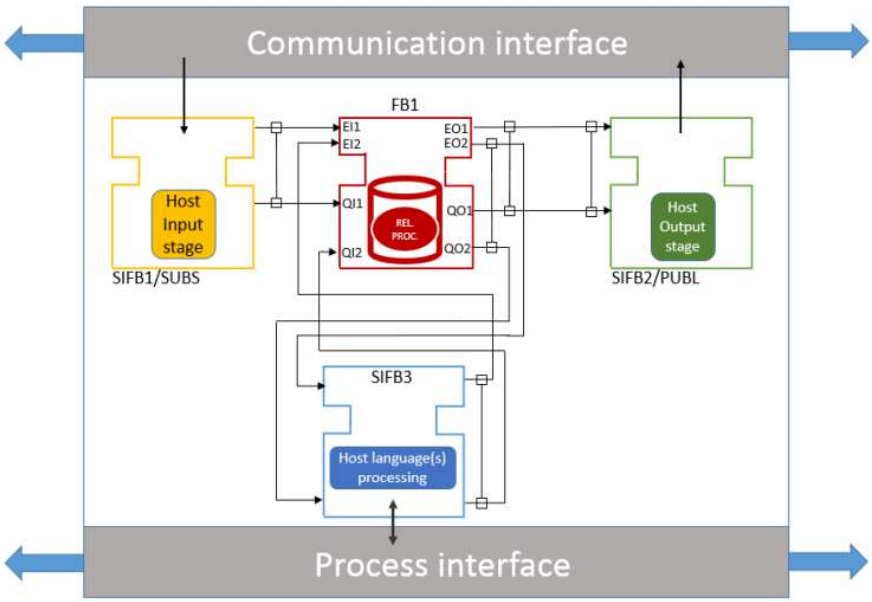


Figura 1.10.3: risorsa IEC 61499 – RMAS mapping

Dopo il *fetching* del primo evento in FB1, è eccitato il processo relazionale e possono essere emessi altri eventi da un'opportuna cascata di trigger interni, fino a quando eventualmente

FB1 genera un evento di uscita con il dato associato (ad esempio EO1 e QO1. Gli eventi e i dati in uscita da FB1 si propagheranno alle altre FB della risorsa connesse a FB1. Le informazioni elaborate da FB1 sono inviate ai *subscribers* tramite SIFB2 (colorato in verde) che incorpora lo stadio di uscita host dell'RMAS. D'altra parte, FB1 è in grado di ricevere eventi e query di dati da *publishers* remoti attraverso SIFB1 (di colore giallo) che rappresenta l'host RMAS della fase di ingresso.
In SIFB3 è mappato lo stadio di processamento di un linguaggio host di RMAS che è il codice relativo alle chiamate a procedure e funzioni per la connessione di RMAS verso attuatori, sensori etc. Possiamo quindi dire che RMAS rappresenta perfettamente una risorsa IEC 61499 che incorpora FB1, SIFB1,2,3.

Capitolo 2.

Produzione snella

Questo capitolo offre una sintesi sulle origini, i principi e gli strumenti che caratterizzano la produzione snella (*Lean production*) nell'industria manifatturiera. In particolare, è descritto più dettagliatamente lo strumento Kanban nella sua concezione tradizionale perché si ritiene possa essere utile come base per la comprensione delle versioni più moderne, inseribili in contesti manifatturieri orientati a I4.0. Questa descrizione è propedeutica al controllo di produzione Kanban la cui implementazione in IEC-61499 sarà descritta nel capitolo successivo.

2.1. Origini e principi

Dopo il secondo dopoguerra, fu l'obiettivo di non soccombere dinanzi alla crescente competitività richiesta in un mondo manifatturiero dominato dalla produzione di massa e dall'industria occidentale, che gli ingegneri giapponesi Kiichiro Toyoda, Taiichi Ohno e colleghi cominciarono ad implementare il sistema TPS (*Toyota Production System*) nella gestione dei processi di produzione Toyota. Dopo la Seconda guerra mondiale la via per la competitività doveva necessariamente passare per la fornitura tempestiva di lotti di prodotto più contenuti, con un certo grado di varietà e qualità, proprietà che non erano gestibili efficacemente con l'applicazione grezza del modello Fordista. Gli ingegneri giapponesi non aggiungevano solo un nuovo gusto al modello produttivo fordista, ma introducevano un cambio di pensiero, quello che poi verrà conosciuto come il pensiero snello (*Lean Thinking*) applicato al sistema produttivo Toyota. Nasceva quindi il concetto di *produzione snella* (*Lean production*) che sarà poi abbracciato anche dall'industria occidentale a partire dagli anni '70.

Il mantra del pensiero Lean è la soddisfazione del cliente finale, il profitto è la realizzazione di questo obiettivo perché il cliente è disposto a pagare di più per un prodotto al quale attribuisce maggior valore. Questo concetto si pone in contrasto con l'approccio standard per il quale il profitto è al centro di ogni speculazione, laddove si determina il livello di produzione necessario per abbattere i costi e raggiungere un certo margine di profitto che conduce a prodotti standard e produzioni di massa.
Quindi per prima cosa occorre definire ciò che ha valore per il cliente, una volta definito ciò si può procedere all'organizzazione dei processi e delle attività che conducono alla

realizzazione del prodotto tenendo bene a mente che l'obiettivo è sempre quello di ridurre i costi.
Al fine di abbassare questi costi, la concezione *Lean* del sistema fabbrica parte da una visione olistica dell'organizzazione aziendale che mira all'efficientamento di ogni processo dell'insieme delle attività di produzione. Il paradigma '*do more with less*' suggerisce di utilizzare la metà delle risorse impiegate in ogni attività, che siano ore uomo per la progettazione di un prodotto o l'inventario (scorte) a disposizione [22]. Ciò si ottiene analizzando le singole attività, le connessioni tra le attività, con l'obiettivo di identificare quelle operazioni che rappresentano un valore per il flusso produttivo ed eliminare quelle che rappresentano sprechi (*Muda*). Tipici sprechi si possono identificare nelle attese, i trasporti, la sovrapproduzione, eccessive scorte, la movimentazione, i difetti, presenza di operazioni inutili nel processo. Ad esempio, le attese possono essere dovute agli accodamenti del prodotto non necessari nel ciclo (o flusso) produttivo, cioè la differenza fra il tempo totale di attraversamento (*Lead Time*) del flusso produttivo di un bene/servizio e il suo tempo di produzione reale dovuto al processo tecnologico. La riduzione degli sprechi deve seguire uno schema iterativo per un miglioramento continuo che deve ridurli a zero. L'efficientamento globale deve essere raggiunto migliorando l'organizzazione dei processi favorendo connessioni e *scambio informativo* tra processi, responsabilizzando ed addestrando personale più consapevole delle interazioni tra le attività e abile a identificare quelle operazioni necessarie ad innalzare il valore per il cliente, intercettare sprechi e problemi.

La riduzione degli sprechi è fattibile ripetendo in maniera ciclica le cinque *azioni* o *principi* fino al raggiungimento della perfezione:

1. Definire il valore percepito dal cliente in riferimento al prodotto o servizio
2. Identificare o mappare il flusso di valore (*mapping the value stream o VSM)* (Identificare l'insieme degli steps di produzione che portano valore o portano valore aggiunto)
3. Realizzare un flusso continuo di produzione (*continuous flow*)
4. Realizzare un flusso di produzione guidato dal cliente (*pull production)*
5. Ricerca della perfezione (*striving for perfection*)

La definizione del valore percepito dal cliente è importante per capire ciò che il cliente si aspetta, e ciò consente di eliminare le attività che non sono utili per soddisfare queste aspettative. L'identificazione del flusso di valore tra i processi dell'organizzazione si espleta fisicamente in fabbrica dove si raccolgono le informazioni sui processi e le relazioni tra processi, facendo visita agli operatori, analizzando gli impianti, discutendo con i responsabili e gli operatori di reparto sul come e perché vengano eseguite delle determinate operazioni. È consigliata la realizzazione di mappe visuali del flusso di valore che identifichino lo stato attuale e lo stato futuro. Il fine del VSM è quello di avere l'ottimizzazione del flusso globale, non dello specifico processo.

Uno strumento utile e diffusamente utilizzato in letteratura per integrare la descrizione della filosofia *Lean* è la *House of Lean* [23] (Fig. 2.1.1) che definisce due pilastri di una struttura

il cui tetto è formato dagli obiettivi, cioè qualità, costi minimi, minimi tempi di consegna, miglior sicurezza e alta morale, ottenuti minimizzando il flusso produttivo con l'eliminazione degli sprechi. La struttura è solida se ogni elemento è solido a partire dalle basi e i pilastri.

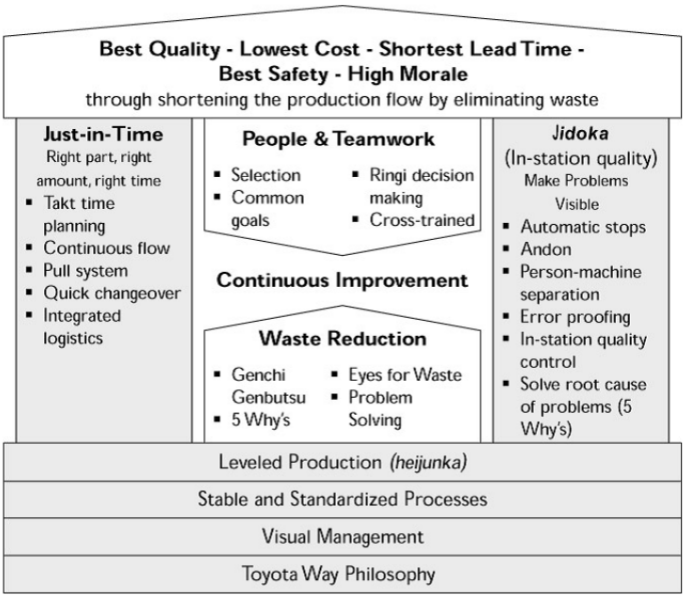


Fig. 2.1.1: House of Lean (Liker, 2004)

Alla base troviamo il concetto di produzione livellata (*heijunka*) intesa come flusso produttivo che si svolge fluido rispetto alla domanda. Una produzione livellata vuole evitare picchi del carico di lavoro che possono inficiare sulla qualità e la sicurezza. Un modo per ottenerla è l'applicazione del *Just-in-Time* (uno dei pilastri della struttura), cioè facendo in modo che i componenti per la produzione siano sempre disponibili, nella giusta quantità e sequenza al momento giusto. Questa tecnica comprende l'applicazione di un *Takt-time* al flusso produttivo, cioè un passo di produzione costante per un flusso stabile di produzione. Ma il *Just-in-Time* significa anche zero inventario*,* zero scorte in magazzino e tra processi*,* cioè si produce solo ciò che deve essere venduto in accordo con il sistema di produzione *pull.* Uno strumento spesso associato al metodo *Just-in-Time* e il sistema *pull* è il metodo Kanban che abilita maggior controllo e tracciamento di ciò che è prodotto in ogni processo della catena produttiva con conseguente contenimento degli inventari di processo.
Il secondo pilastro della struttura (*Jidoka*) comprende tutte quelle metodologie orientate a garantire la qualità di ciò che è prodotto facendo in modo che un pezzo difettoso venga intercettato prima che crei problemi ad una fase successiva di lavorazione e disturbi il flusso continuo di produzione. *Andon* e *Poka-Yoke* sono due strumenti tipici di *Jidoka,* il primo di solito è associato ad un pannello in cui sono visualizzate le informazioni sullo stato della

linea produttiva, il secondo fornisce strumenti o macchine in grado di assistere l'operatore nell'assemblaggio al fine di evitare la produzione di pezzi difettosi.

Nel paragrafo seguente lo strumento Kanban è descritto in maniera più dettagliata.

2.2 Strumento Kanban

Le aziende manifatturiere occidentali, tra gli anni '70 e '80, iniziarono a utilizzare lo strumento Kanban per il controllo di produzione nell'ambito della gestione JIT (*Just in Time*) dei flussi di produttivi. Ciò avvenne dopo che questo metodo di gestione e lo strumento Kanban erano già stati implementati con efficacia dalle aziende giapponesi.
Il successo di questo strumento favorì l'interesse non solo del mondo industriale [24],[25] ma anche quello della ricerca con numerosi studi empirici che ne dimostravano la validità [26],[27].
Implementare un controllo Kanban in maniera efficace presuppone la determinazione di una configurazione del controllo ottima o sub-ottima con metodi in grado di determinare misure di indici di prestazione (KPI) come i tassi di riempimento medi dei buffer di prodotto o livelli medi degli inventari nei processi (anche detti WIP=*Work in Progress*) della catena produttiva dello *shop-floor*. Simulazioni al computer possono essere utili ma impiegano molto tempo al contrario dei metodi analitici che sono presenti in letteratura specie per impianti con una sola tipologia di prodotto [28], [29].

Il sistema Kanban tradizionale è basato sulla movimentazione di carte contenenti informazioni per ogni stadio della catena produttiva. Il suo obiettivo è quello di stabilire un flusso continuo (stabile) del materiale che attraversa la catena, un livello stabile di scorte nei processi e tempi di consegna adeguati alla domanda. È un metodo semplice da implementare ed ha costi ridotti. Tuttavia, presenta alcune limitazioni dovute al lavoro improduttivo causato dallo spostamento manuale delle carte da parte dell'operatore umano, infatti, il movimento delle kanban può non verificarsi sempre nel momento in cui c'è l'effettivo consumo dei materiali mettendo in crisi la scorrevolezza del flusso di materiale e il *Just-in-time*. Questa criticità si presenta con l'aumento del ritmo delle operazioni di produzione e con la dimensione del lotto di produzione perché aumenta il numero dei movimenti delle carte con la conseguenza che queste possano andare smarrite. Altri aspetti che mettono in crisi il Kanban tradizionale, o addirittura inapplicabile, è il grado di varietà del prodotto, una domanda di prodotto con ampie fluttuazioni, lunghi tempi nel cambio dei settaggi dei processi di produttivi (*changeovers*) e operazioni non standardizzate. Questo scenario indica che il sistema tradizionale deve essere adattato o modificato in base alle condizioni di produzione citate. In letteratura si può fare riferimento ad una review sulle varianti metodologiche che potranno essere valutate e applicate al sistema Kanban tradizionale in diverse condizioni produttive [30].

Queste limitazioni sono superate con l'ausilio delle kanban elettroniche (sistema e-Kanban) che, sfruttando la logica Kanban tradizionale in un ecosistema digitalizzato offre ampie possibilità di ottimizzazione e robustezza del controllo.

Il Kanban tradizionale si basa su quattro concetti fondamentali:

- produzione pull
- controllo decentralizzato
- quantità stabile dell'inventario dei processi produttivi (scorte o WIP)
- uso di due segnali di comunicazione (o sistema duale, cioè con carte kanban di produzione e carte di trasporto)

Lo strumento di controllo Kanban applicato alla catena dei processi di uno shop floor si basa sul metodo di produzione *pull*: un processo di una catena di produzione può iniziare a produrre un nuovo pezzo solo se il suo processo cliente consuma quello disponibile in uscita, in sostanza, *il processo deve fornire ciò che è necessario solo quando è necessario, della quantità e qualità necessaria*. Quando un processo (detto anche venditore) ha disponibile il pezzo in uscita, il processo a valle (detto consumatore) può ritirarlo e "pagare" il primo con una "moneta", cioè la kanban che funge da segnale o ordine per il venditore che può iniziare a produrre un nuovo pezzo. Nella concezione tradizionale una carta kanban è una etichetta informativa fisica che viaggia con il contenitore contenente il prodotto o direttamente con il prodotto stesso. In Figura 2.2.1 è rappresentata una catena del valore semplificata con le principali fasi di una catena produttiva dal fornitore al cliente finale, in particolare sono utilizzate le frecce direzionali blu per rappresentare i flussi di prodotto e quelle grigie per rappresentare i flussi dei segnali/carte kanban. In un sistema *pull* è la domanda del cliente che detta il ritmo di produzione: il cliente ritira il pezzo disponibile nel magazzino di uscita (in una visione semplificata il pezzo ritirato può già rappresentare il segnale kanban per il magazzino che un nuovo rifornimento è necessario) e il magazzino è autorizzato a rifornirsi di un nuovo pezzo disponibile dal processo produttivo. Il processo produttivo a questo punto può cominciare la produzione di un altro pezzo ed estrae il materiale necessario dal magazzino a monte, se disponibile, e così via fino al fornitore. Il paradigma di produzione *pull* è un paradigma decentralizzato che consente di ovviare ad una serie di problemi connessi al tradizionale sistema *push* (Figura 2.2.1). Il regime push è tipico di un sistema centralizzato in cui i processi della catena sono indipendenti, eseguono solo ordini di lavoro che sono programmati ai vertici dell'organizzazione aziendale nell'ambito di una programmazione globale e istruiti dal sistema MES (*Manufacturing Execution System*), quindi la quantità programmata per un processo è prodotta indipendentemente da ciò che accade in un altro processo a valle. Sempre in un sistema *push,* il MES fissa anche l'inventario interno di ogni processo (o scorte o WIP) (Figura 2.2.1). Questa rigidità può condurre a inventari che eccedono la quantità prevista e questi eccessi possono rappresentare importanti fattori di costo, oltre che mettere in crisi il flusso produttivo. In queste ipotesi può succedere che le stime circa la data di consegna del prodotto finito rimanga disattesa e saranno necessarie continue ripianificazioni. Ciò non si verifica in un sistema *pull* dove le produzioni dei singoli processi sono connesse orizzontalmente nello shop-floor e ciò che accade in un punto della

catena si riflette sempre in tutti i processi a monte. È un solo un processo a dirigere la sinfonia produttiva, il cliente consumatore. Infine, un ordine di produzione è autorizzato da una carta kanban di produzione, un ordine di trasporto è autorizzato da una carta di trasporto (sistema Kanban duale).

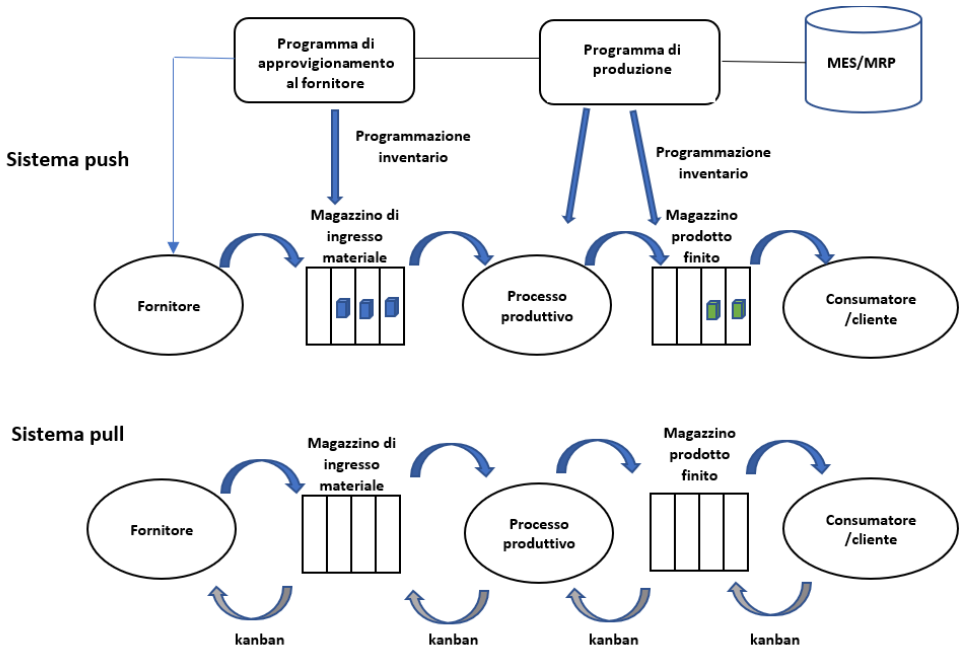


Fig. 2.2.1: Sistemi di produzione push e pull

In questo paragrafo è descritta la logica di controllo Kanban tradizionale singolo stadio e monoprodotto, propedeutica all'implementazione del controllo automatico Kanban proposta nel capitolo 3.

Il controllo è applicato ad una generica catena produttiva che considera alcune attività tipiche di base, ossia, la gestione degli ordini del cliente, la produzione e il trasporto materiali nello *shop-floor* tra due supermarket e il processo produttivo. Trascuriamo l'approvvigionamento al fornitore in quanto procedura di trasporto ridondante ai fini della comprensione del meccanismo di controllo e che può essere considerata successivamente riutilizzando logiche e modelli illustrati in questa descrizione. Per quanto riguarda il modello produttivo, il metodo *pull* prevede che gli ordini del siano evasi con logica *first-in-first out*, per cui i magazzini e i componenti kanban fisici sono modellati con buffers FIFO a capacità finita. Gli elementi che costituiscono il sistema sono mostrati in Figura 2.2.3 e Figura 2.2.4 mentre in Figura 2.2.5 è

mostrata la catena complessiva composta dallo stadio produttivo e da quello di trasporto materiale tra due supermarket A e B. In particolare, abbiamo le seguenti componenti:

- Una sorgente di domanda/ordini del cliente dotato del buffer B2.
  È considerata la possibilità di avere ordini inevasi (*backorders)* che si considerano persi. L'ordine del cliente è inviato al magazzino di uscita, quest'ultimo modellato con il buffer B1.

- Il magazzino di prodotto finito a monte della sorgente di ordini, rappresentato dal buffer B1.

- Un generico processo produttivo, o cella di produzione. Il processo riceve ordini di produzione attraverso il *board/scheduler* B3 e si approvvigiona di materiale attingendo nel magazzino/supermarket B. Rende disponibile il prodotto finito al magazzino B1.

- Un magazzino di approvvigionamento materiale o supermarket rappresentato dal buffer B a monte del processo produttivo.

- In generale, in presenza di processi produttivi multiprodotto, è utilizzata una lavagna (*board/scheduler*) per la visualizzazione delle carte kanban di produzione (una carta kanban di produzione inserita nel board rappresenta un ordine di produzione in attesa che dovrà essere consegnato al processo produttivo) e che rende più chiara, condivisa e monitorabile la situazione attuale degli ordini e la loro schedulazione per gli operatori.
  Tradizionalmente questo supporto di visualizzazione facilita la gestione degli ordini cartacei in presenza di produzioni multiprodotto. La tabella di Figura 2.2.2 mostra un *board* le cui colonne identificano tre differenti prodotti P1, P2, P3. Ogni riga per prodotto contiene una carta/ordine kanban in attesa di essere servita e durante la produzione il board si popola dall'alto verso il basso. Si notano degli indicatori rossi posti a diversi livelli per ogni prodotto che segnalano situazioni critiche di scarsità di quel prodotto in magazzino e che quindi, è necessario provvedere subito a servire gli ordini che popolano questa zona. Per ogni prodotto si utilizza un numero di carte Kanban deciso in base all'analisi dei costi (collegati anche alla quantità di scorte/giacenza di prodotto nel buffer di materiale in magazzino), dei tempi di produzione, numero delle differenti produzioni e profilo della domanda del cliente per quel prodotto [31]. Maggiore è il numero di carte utilizzate per prodotto, maggiore sarà il materiale circolante in magazzino (o in generale in un buffer WIP tra due processi produttivi della catena) che si è deciso di tollerare come costo per sopperire a fenomeni imprevisti in produzione o variazioni del profilo della domanda cliente.
  Le carte destinate al *board*, in genere, sono quelle estratte da un *box* (o *collector*) posto nelle vicinanze del magazzino di uscita (nel nostro caso il magazzino di uscita B1) a valle del processo produttivo. Una carta kanban è inserita nel *collector* al

consumo di una certa quantità di prodotto finito nel magazzino B1, cioè a seguito del ritiro di un prodotto (o un contenitore di prodotto) da parte di un operatore dal magazzino B1. In questa trattazione, non prevedendo uno scheduling tra più prodotti, in questa trattazione accorpiamo le funzionalità del *board* e del collector nel loop di produzione definendo un solo oggetto che chiamiamo board o alternativamente collector, rappresentato dal buffer B3.

P1	P2	P3
K1	K2	K3
K1	K2	K3
K1	K2	K3
K1	K2	K3
K1	K2	K3
K1	K2	K3
K1	K2	K3
K1		K3
K1		

Fig. 2.2.2: Kanban Board

- Un raccoglitore generico (*box* o *collector*) delle carte kanban di trasporto o di produzione. Ogni carta kanban di trasporto rappresenta un ordine di trasporto di materiale tra supermarket (o isole o magazzini) ed è tradizionalmente inserita nel collector al consumo di prodotto nel supermarket (o magazzino). Una carta è estratta dal collector da operatori attivi del loop di trasporto quando l'ordine di trasporto associato alla carta deve essere espletato. Il *collector* è rappresentato in Figura 2.2.4 dal buffer BC.

- Un magazzino di ingresso o supermarket rappresentato dal buffer A.

- Un set finito di carte kanban di produzione da utilizzare nel loop kanban di produzione. Una carta kanban di produzione viaggia con il contenitore e in genere contiene le seguenti informazioni sul prodotto contenuto: numero della parte, descrizione della parte, quantità della parte, processo produttivo che evade l'ordine, magazzino che riceve il prodotto. Una carta può anche viaggiare con il singolo prodotto/componente quando non sono utilizzati i contenitori.

- Un set finito di carte kanban (ordini di trasporto) da utilizzare nel loop di trasporto. Una kanban di trasporto in generale contiene le seguenti informazioni: numero della parte, descrizione della parte, quantità della parte, magazzino di partenza, magazzino di arrivo del materiale.

- Contenitori di prodotto.

Sempre in riferimento alle figure 2.2.3, 2.2.4, 2.2.5, il controllo di produzione procede come segue:

Lato client:

Una generica sorgente genera gli ordini di prodotto finito che si accodano nel buffer B2. Un operatore è addetto al ritiro dell'ordine dal buffer quando svincolato da altre mansioni. Ritirato l'ordine l'operatore controlla se un contenitore di prodotto finito è disponibile in uscita al magazzino B1. In caso negativo l'operatore deve attendere e può dedicarsi ad altro. Se il prodotto è presente in B1 e lo scheduler B3 non è già saturo di ordini precedenti in coda, l'operatore può staccare la kanban posta sul contenitore ed inserirla nello scheduler B3 come nuovo ordine di produzione in attesa di essere evaso, può ritirare il prodotto dal magazzino e consegnarlo per l'invio al cliente (qui potrebbe instaurarsi un loop kanban con il gestore del trasporto al cliente esterno alla fabbrica che non è considerato in questa descrizione).

Loop di produzione:

Se il processo produttivo ha terminato una produzione a seguito di un precedente ordine il contenitore è reso disponibile dall'operatore in ingresso al magazzino B1 che può accettarlo nel caso quest'ultimo non sia saturo (in teoria B1 non può essere saturo in quanto ogni quantità è prodotta a seguito di un consumo che fa partire l'ordine di produzione). Se B1 fosse saturo il processo non può comunque produrre ulteriori pezzi. Se B1 non è saturo accetta l'inserimento del prodotto in magazzino e l'operatore del processo produttivo può consentire una nuova produzione se sono presenti ordini di produzione kanban nello scheduler B3 e se contemporaneamente sia disponibile il prodotto o componente necessario alla lavorazione in uscita al buffer B (il prodotto/componente è disponibile in uscita al buffer B e può essere ritirato solo se il relativo collector BC non è già saturo di ordini). Nel caso i due controlli abbiano esito affermativo, l'operatore stacca la kanban di trasporto dal container in B, la inserisce nel collector BC del loop di trasporto come nuovo ordine di trasporto in attesa, attacca la kanban di produzione fornita dallo scheduler del loop di produzione sul container ritirato in B e lo invia in lavorazione, altrimenti il processo produttivo rimane in attesa.

Loop di trasporto:

Il trasporto del materiale tra un supermarket A (in generale un'isola di stoccaggio) e un supermarket B è effettuato in base agli ordini kanban di trasporto presenti nel collector BC. L'operatore di trasporto controlla se sono presenti carte kanban nel collector e in caso affermativo esegue il rifornimento in A della quantità indicata nella carta. Al contenitore di prodotto/componente fornito in A verrà applicata la kanban di trasporto e consegnato in B se quest' ultimo non è saturo.

Il supermarket A sarà eventualmente rifornito da un'altra isola di stoccaggio, da un magazzino di ingresso o direttamente da un fornitore esterno con i quali è consigliabile instaurare ulteriori loop Kanban di trasporto.

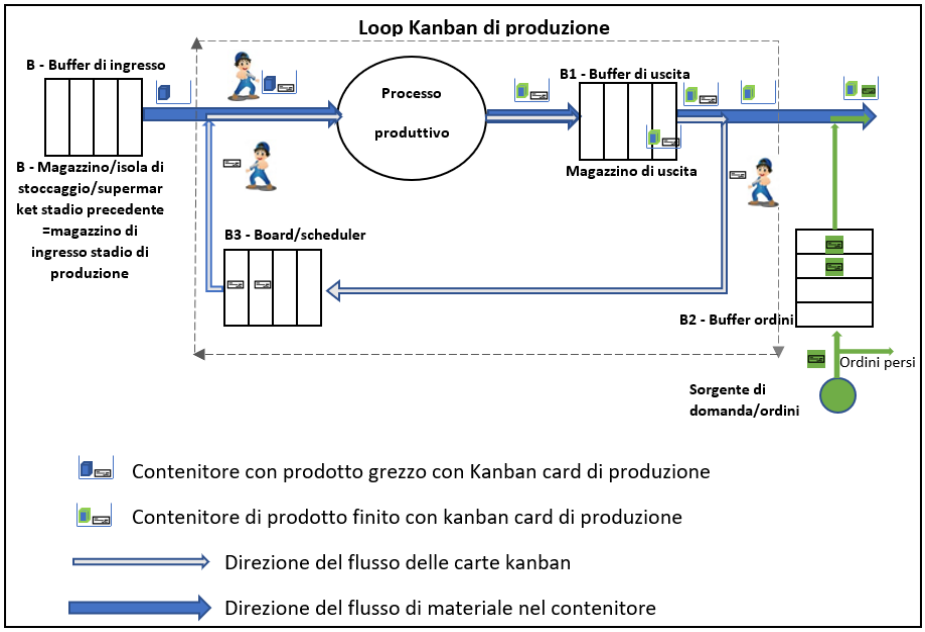


Fig. 2.2.3: Sorgente di domanda e Loop Kanban di produzione

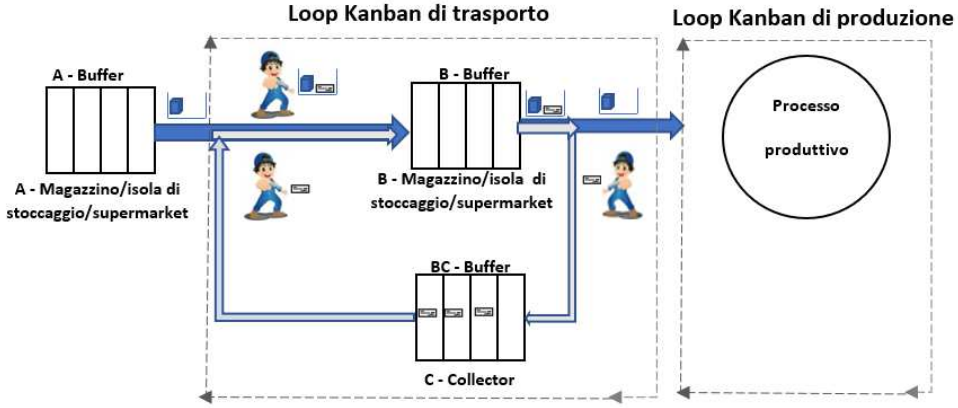


Fig. 2.2.4: Loop Kanban di trasporto

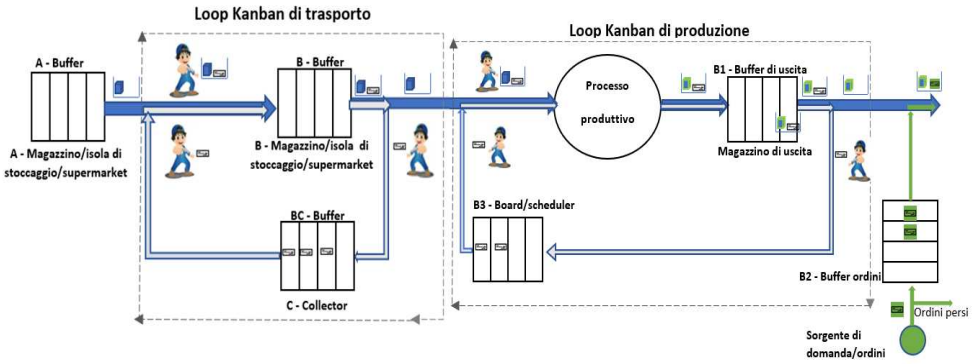


Fig. 2.2.5: Composizione del loop di trasporto e produzione

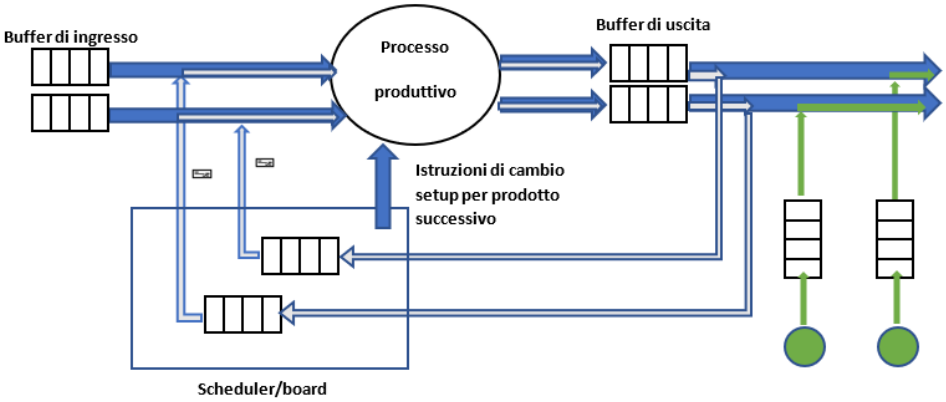


Fig. 2.2.6: Configurazione multiprodotto (2 prodotti)

Per configurazioni multiprodotto, cioè nel caso il processo possa produrre più tipologie di prodotto e componenti, le criticità per il sistema kanban tradizionale aumentano. Nel caso di due prodotti (Figura 2.2.6) ad esempio, avremo un modello con due buffer per magazzini/supermarket di ingresso e uscita, uno scheduler/board con due buffer (o due buffer collectors), due buffer per gli ordini cliente. In questo caso l'operatore che gestisce lo scheduler/board dovrà adottare una politica per il cambio di setup del processo produttivo (cambio prodotto) che può avvenire in maniera ciclica o in base a certe priorità per prodotto decise in base alle condizioni di produzione. Quale che sia la configurazione di produzione, ciascun prodotto dovrà avere il proprio loop di produzione e trasporto e nel caso di una certa varietà delle tipologie dei prodotti potrebbe non essere agevole gestire tanti set di carte

kanban diversi senza incorrere in errori o rallentamenti che pregiudicano il flusso continuo. Anche nel caso di un aumento repentino della domanda, ad esempio, l'operatore sarà costretto ad aumentare il numero delle carte kanban da utilizzare nel loop (aumento delle dimensioni dei buffer) e probabilmente anche la quantità di prodotto riportata nelle carte, quindi ulteriore aumento della tipologia di carte da gestire. Per queste ragioni un controllo Kanban tradizionale è consigliabile nel caso in cui la domanda sia stabile o subisca deboli variazioni.

2.3 Strumento e-Kanban

Già con le possibilità offerte dalle tecnologie I3.0, in generale, i problemi tradizionali legati alla gestione cartacea dei flussi informativi sono superati. Le tecnologie ICT hanno permesso infatti di integrare le informazioni di produzione generate nello shop-floor con i sistemi software dei livelli superiori dell'organizzazione, i flussi di materiale possono essere tracciati con dispositivi RFID (*Radio frequency Identification*) o con la lettura di codici a barre [32] incorporati al prodotto o ai contenitori e queste informazioni sono trasmesse al database centrale dal quale i sistemi ERP attingono per fare le analisi di produzione. La decisione su cosa produrre e quando produrre, le dimensioni degli inventari, gli ordini di rifornimento materiale e di produzione sono stabiliti in base a previsioni che derivano dall'analisi della domanda del cliente e dei dati storici di produzione presenti nel database centrale. In questo contesto le operazioni sono governate da una logica push pura. Vi sono organizzazioni in cui si fa un passo in più, cioè si decide di investire in moduli Lean incorporati nel software ERP come il *Just In Time* e e-*Kanban*, in questi casi siamo in presenza di una logica push intrecciata con quella pull di una visione Lean.

In generale si può parlare di tecnologia e-Kanban quando le carte tradizionali e i tracciamenti sono in forma digitale ed elaborati da un sistema computerizzato. Un sistema computerizzato ad hoc potrebbe implementare il metodo pull puro del kanban tradizionale per ereditarne i benefici ed eliminare le note problematiche connesse all'uso di strumenti cartacei. Per controllo pull puro si intende che le quantità dei materiali nei buffer WIP tra processi e nei buffer dei supermercati sono regolate dal controllo stesso una volta definita (tipicamente su base giornaliera) la scorta di materiale nei buffer e le quantità kanban (numero di carte utilizzate) per ogni loop di produzione e trasporto, senza ingerenze di sistemi push che vorrebbero forzare dall'esterno queste quantità. Nel caso, ad esempio, di un loop di trasporto materiali instaurato con un fornitore esterno, il numero delle carte utilizzate può essere, in generale, maggiore della quantità ottenuta moltiplicando il "*lead time"* tipico del fornitore (cioè il tempo che il fornitore impiega nella consegna di una certa quantità di materiale associata ad una carta kanban), per il consumo giornaliero (o domanda giornaliera) compreso di un fattore di sicurezza (che caratterizza le scorte) e dividendo per la capacità di un contenitore.

In sistemi ERP Lean si combinano invece le logiche push tradizionali con logiche pull che caratterizzano la produzione snella e il controllo Kanban, non siamo, quindi, in presenza di un pull puro.
In sostanza, il Kanban elettronico è un sistema Kanban che riduce a zero le possibilità di errore umano che si verificano nelle movimentazioni di materiali e nei processi decisionali, specie in contesti di produzioni variegate e domanda fluttuante o random. In questi contesti un Kanban cartaceo sarebbe ingestibile. Sarebbe invece da valutare, per ragioni di costi, l'implementazione di e-Kanban nel caso di domanda stabile e pochi prodotti, in questo caso potrebbe essere ancora vantaggioso un Kanban cartaceo ma la questione va valutata caso per caso. Come già introdotto, tipicamente le tecnologie RFID e tag con codici a barre possono essere incorporate con il prodotto o con i contenitori, quindi, possono essere utilizzate per generare i segnali delle movimentazioni e del consumo di materiale lungo il flusso di produzione. Questi dati sono analizzati dai livelli superiori per adattare i parametri di produzione. Il sistema e-Kanban eredita il metodo del controllo tradizionale, ovvero, quando un prodotto con tag RFID o barcode è consumato in un buffer WIP, in un supermarket o in un magazzino, il segnale e-kanban (carta digitale, menzionata in letteratura come carta kanban virtuale [30]) che rappresenta un nuovo ordine di rifornimento o di produzione elettronico, è inviato al sistema computerizzato. Ad esempio, semplificando, un sistema computerizzato svolgerà la funzione del board B3 del loop di produzione di Figura 2.2.3: alla ricezione del segnale kanban dal tag di prodotto (prodotto ritirato dal magazzino), la e-kanban è inserita in una coda interna del sistema. Da questa coda il sistema preleva le e-kanban e le invia come nuovi ordini di produzione al processo specifico secondo politiche di priorità stabilite tra diverse produzioni. Simile logica vale per un loop di trasporto: ad esempio, in un loop di trasporto con un fornitore esterno, il consumo di prodotto da un magazzino o supermarket interno, genera una e-kanban di trasporto nella coda del sistema computerizzato. Quando il numero delle e-kanban supererà una soglia critica tale per cui la quantità del materiale in magazzino sarà inferiore a quella indicata precedentemente in questo paragrafo, che tiene conto della domanda e del lead time del fornitore, partirà un nuovo ordine kanban di rifornimento verso il fornitore stesso (e-mail etc..).

Capitolo 3.

Progetto Shop-Floor

3.1. Obiettivi

Il contributo di tesi espresso in questo capitolo tratta la progettazione di un controllo Kanban decentralizzato che sfrutta tecnologie e-kanban elettroniche e applicabile in un tipico shop-floor di una azienda manifatturiera. Sono considerati componenti fisici tipici di una generica catena produttiva che tratta la lavorazione di un generico prodotto attraverso i quali sono realizzati un loop Kanban di produzione e un loop Kanban di trasporto tra processi. A questi elementi si associano le interfacce software embedded che permetteranno la comunicazione di ordini di produzione o di ordini di trasporto materiale elettronici (e-Kanban di produzione e trasporto) tra gli elementi della catena produttiva sfruttando l'utilizzo di una minima semantica di comunicazione ad eventi. I componenti di questo sistema effettuano un controllo distribuito e decentralizzato dei flussi di materiale secondo il principio del Kanban tradizionale descritto nel Capitolo 2.
Come già descritto, un controllo Kanban consente di ovviare a tutti i problemi connessi alla pratica di scambio informativo ancora effettuato per via cartacea tra operatori nello shop-floor di molte aziende manifatturiere (ad esempio ordini di produzione/trasporto e tracciamento del flusso di materiale che vengono persi). Inoltre, il sistema implementato andrebbe ad eliminare i rallentamenti che si verificano tipicamente in un sistema Kanban tradizionale dovuti al fatto che per mille motivi, un prodotto non è mai ritirato dal magazzino l'esatto momento in cui è disponibile, introducendo inefficienze che nei casi critici di ritmi di produzione sostenuti possono aumentare e mettere in crisi il flusso continuo di produzione.
Tipicamente in aziende provviste di ERP con moduli Kanban, questo controllo è implementato in modo centralizzato, è L'ERP che organizza il flusso di ordini Kanban di produzione e richieste materiale ed in genere un sistema *pull* non è supportato pienamente, ad esempio in [33] è descritto un approccio con interfaccia web per l'inclusione di ordini cliente emessi in magazzino in un sistema pull.
Il controllo Kanban proposto in questa tesi è decentralizzato e usabile in shop-floor di organizzazioni aziendali prive di software ERP o, laddove esista un ERP, non sia previsto di investire in moduli aggiuntivi di tipo Lean ERP. Ciò non significa che non possa esserci un'integrazione con questi sistemi dotando i dispositivi di funzionalità aggiuntive cyber o non cyber per ulteriori ottimizzazioni del processo globale. L'idea è quella di organizzare una architettura di controllo basata su un paradigma di comunicazione ad eventi, modulare, flessibile e distribuita tra i dispositivi di uno shop-floor, in grado in primis di risolvere

automaticamente un controllo Kanban di tipo *pull* e che sia adattabile a situazioni di produzione diverse e dispositivi diversi; come detto, potrebbero coesistere sezioni di produzione con logica pull autonome per particolari prodotti e altre che fanno riferimento ad un sistema informativo centralizzato a logica push che potranno comunque essere integrate.

Per la progettazione dell'applicazione è scelto lo standard IEC-61499, concepito per superare i limiti tipici dei sistemi di controllo centralizzati e monolitici basati su standard IEC 61131. Si pensi alla riusabilità delle componenti software (riusabilità dei blocchi funzionali o FB), al supporto OpenSource per la comunicazione tra i dispositivi, alla funzionalità di *distribuzione* del controllo e alla *riconfigurazione dinamica* (a *run-time*) del codice dei dispositivi, *tutti elementi che forniscono un'infrastruttura compatibile con i principali requisiti di I4.0 e le applicazioni IoT industriali*. Fatto importante è che la caratteristica di riconfigurabilità dinamica dell'applicazione può permettere ad un supervisore (ad esempio un CPS) di modificare i parametri del controllo e adattarli run-time alle variabilità della domanda del cliente. Ad esempio, il supervisore può decidere se cambiare dinamicamente il numero delle "carte" e-kanban circolanti nei loop di trasporto in base all'ottimo ricalcolato. In quanto alla riusabilità del codice, questa implementazione offre la possibilità di aggiungere altri loop Kanban di produzione e trasporto qualora altri processi o dispositivi entrino a far parte della catena produttiva (nei loops di trasporto e produzione esistenti o nuovi) riutilizzando il codice già esistente o adattarlo a nuove specifiche esigenze, con minime modifiche data la modularità del progetto. La funzionalità globale dell'applicazione è anche direttamente simulabile su un computer, non è necessario conoscere l'infrastruttura hardware sottostante. Questi sono tutti vantaggi che derivano di uno standard orientato al modello applicazione piuttosto che al componente.

3.2. Scenario di produzione

In Figura 3.2.1 è rappresentato lo schema di una catena produttiva di una generica azienda manifatturiera dove sono evidenziati i tipici flussi informativi riferiti agli ordini di produzione o trasporto materiale in sistemi di controllo della produzione di tipo *push* e *pull.* Il primo è evidenziato con frecce verdi e blu, il secondo con frecce grigie. Come già descritto nel Capitolo 2, si ricorda che in un sistema di produzione push, i sistemi MES/MRP, se presenti, gestiscono l'approvvigionamento di materiali, definiscono le quantità delle scorte nei magazzini, supermarket, le scorte di materiale tra i processi produttivi (WIP) e gli ordini di lavorazione per le linee produttive. Moduli aggiuntivi Lean ERP, incluso quello Kanban, sono orientati nell' adattare sistemi che nascono con filosofia push, ad approcci di tipo pull permettendo di prendere decisioni ottimali circa le attività di produzione e logistica con l'obiettivo di ridurre le scorte e rendere il flusso produttivo più snello, efficiente e reattivo grazie all'analisi dei dati di produzione (attuali e storici) provenienti dallo shop-floor.

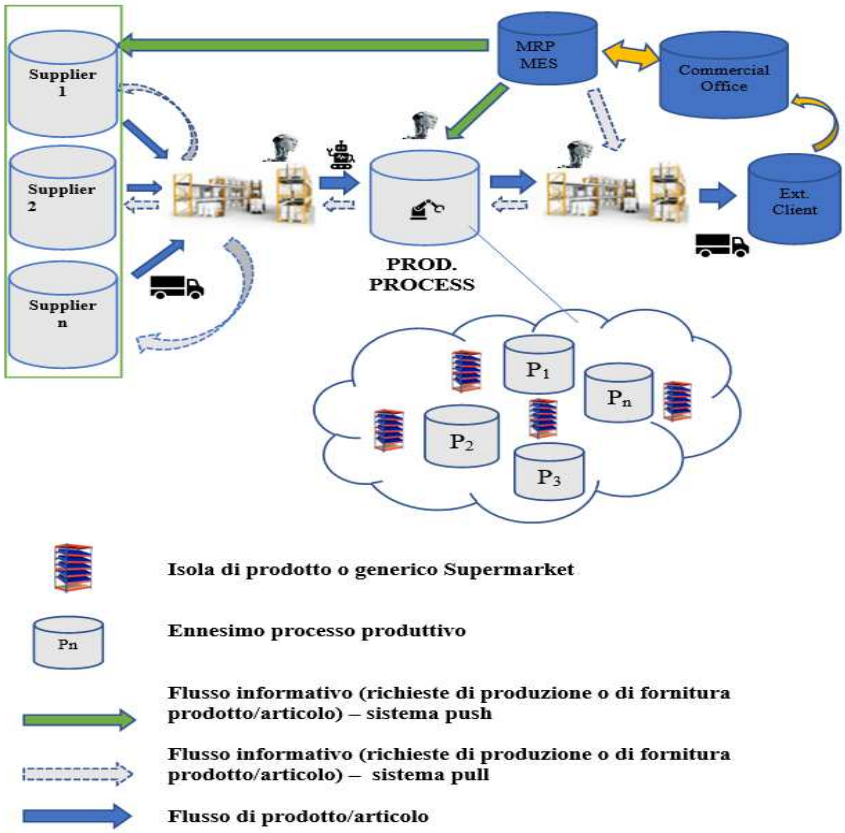


Fig. 3.2.1: Generico processo di fornitura, produzione e distribuzione

Lo stato dei magazzini, le movimentazioni e la domanda del cliente sono monitorati in tempo reale e disponibili in una base di dati centrale.
In generale il processo o i processi di produzione e, tra questi, la gestione degli ordini di trasporto dei materiali, possono essere molto complessi e le metodologie Kanban sono gestite dai sistemi software dei livelli più alti dell'organizzazione, sopra lo shop-floor.

L'obiettivo è quello di realizzare un controllo di produzione Kanban pull, per quanto possibile atomico rispetto alle funzionalità produttive e logistiche che devono essere svolte nello shop-floor, e autonomo rispetto ai software e moduli sopracitati. Il controllo è espletato dai componenti di una linea produttiva minima e in condizioni di domanda stabile (condizioni ideali per il metodo Kanban tradizionale) mantiene livelli stabili dei materiali in magazzino o nei supermarket e stabili le scorte tra processi, tendenti a zero in virtù della logica pull applicata, cioè si produce e si trasporta materiale solo quando si consuma prodotto in

magazzino a seguito di un ordine del cliente. Situazioni critiche come cambiamenti repentini della domanda possono essere affrontate dai software dei componenti con l'adeguamento del numero di carte Kanban utilizzate nei loop di trasporto e produzione (in soldoni le dimensioni dei buffer collectors descritti nel capitolo precedente), numero tipicamente proporzionale alla quantità di scorta necessaria per assorbire tali cambiamenti.

In particolare, facciamo riferimento alla figura 3.2.2 che semplifica la catena produttiva di Figura 3.2.1. Nello specifico sono considerati, da sinistra a destra, il magazzino interno di ingresso materiale (*Warehouse*), un supermarket A, un supermarket B, il processo produttivo, un magazzino o isola di prodotto finito in uscita (*OutputStore*), un cliente interno che genera gli ordini di prodotto finito (*Int Client/MRP*) verso il magazzino di uscita. Nella stessa figura sono anche riportati i flussi informativi tra i dispositivi (in grigio) e i flussi di prodotto (in blu). Senza perdere di generalità, facciamo le seguenti ipotesi:

- Esiste un flusso informativo (ordini e-kanban), evidenziato dalle frecce colorate in grigio, tra il cliente interno (*Int. Client/MES*) e il magazzino di uscita (*Output Store*), tra gli operatori di trasporto (robots colorati in nero) e il magazzino di uscita (*Output Store*), tra il processo produttivo e il magazzino di uscita, tra il processo produttivo e il supermarket B, tra il supermarket B e l'operatore di trasporto (robot colorato in giallo), ed eventualmente tra il supermarket A e un operatore di trasporto a monte.

- Escludiamo dalla catena di produzione il cliente finale esterno e consideriamo un dispositivo client interno alla fabbrica (*Int. Client/MES*) come sorgente degli ordini di prodotto inviati al magazzino di uscita. Questa sorgente ha un suo buffer interno a dimensione finita e può essere prevista o non prevista la perdita degli ordini generati. Nel caso non si preveda questa perdita, la sorgente emette un nuovo ordine solo all'avvenuta evasione dell'ordine precedente.

- Assumiamo che i supermarket e il magazzino di uscita possano trattare più tipologie di prodotti tenendo presente, come descritto nel capitolo precedente, che deve esistere un loop Kanban per ciascuna tipologia di prodotto. Un supermarket ospita un proprio database interno di prodotto/i e un buffer collector per l'immagazzinamento degli ordini (e-Kanban) di trasporto di uno specifico prodotto in attesa di essere emessi. Non appena un prodotto del database di un supermarket è consumato su richiesta di un dispositivo a valle, è generato un relativo nuovo ordine di trasporto nel collector del supermarket. Gli ordini in attesa nel collector saranno evasi da un dispositivo robot o in generale qualsiasi dispositivo rispettante la semantica di comunicazione stabilita. In questo caso un robot riceve gli ordini dal collector ed esegue il pick-up del materiale relativo all'ordine nel supermarket o magazzino a monte, poi il trasporto e l'inserimento del prodotto nel supermarket che ha generato l'ordine di trasporto.

- Anche il magazzino di uscita ospita un proprio database interno di prodotto/i e un buffer collector per l'immagazzinamento degli ordini (e-kanban) di produzione di uno specifico prodotto in attesa di essere emessi.

- Escludiamo dal sistema i fornitori esterni e il magazzino interno o Warehouse. Ciò significa che non consideriamo eventuali loop Kanban di trasporto tra il supermarket A e il magazzino di ingresso (*o Warehouse*); quindi, consideriamo risorse infinite (prodotto sempre disponibile) per il supermarket di ingresso A. Il supermarket A approvvigiona un supermarket B per mezzo del robot introdotto al punto precedente, ad esempio un AGV (*Automated Ground Vehicle*), e il supermarket B rifornisce l'unico processo produttivo (cella 1). Il processo produttivo rifornisce un'isola di prodotto finito o magazzino d'uscita *(Output Store)* tramite un robot di pick-up e trasporto gestito dal software del magazzino. Dal magazzino fisico, il prodotto fisico sarà disponibile al client tramite l'ausilio di un secondo robot per pick-up e trasporto sempre gestito dal magazzino.

- Al termine di una produzione il processo produttivo richiede al magazzino di uscita l'inserimento del prodotto fisico con associata kanban di prodotto. Se il magazzino può contenere il prodotto (database interno non saturo) richiede al primo Trobot (colorato in nero) di eseguire il pick-up e il trasporto del prodotto nella postazione fisica. A trasporto concluso il magazzino aggiorna il database interno con la kanban del prodotto acquisito ricevuta dal processo produttivo. Dal lato client, il magazzino riceve ordini kanban di ritiro di un prodotto. Alla ricezione dell'ordine, il magazzino controlla nel proprio database se il prodotto è presente. In caso negativo il client rimane in attesa. In caso affermativo il magazzino controlla che il proprio buffer collector non sia già saturo di ordini. Nel caso non lo fosse invia la kanban del prodotto al collector come nuovo ordine di produzione e richiede al secondo TRobot un pick-up in magazzino per il ritiro del prodotto. Conclusa questa operazione il magazzino invia al client la kanban di ordine eseguito. Se il collector fosse saturo l'ordine del client rimane in attesa.

- Se una richiesta di inserimento prodotto nel magazzino di uscita è andata a buon fine il processo produttivo può iniziare una nuova lavorazione: a tal fine il processo controlla se sono presenti kanban/ordini di produzione in attesa nel magazzino di uscita (forniti dal collector di magazzino). In caso affermativo può iniziare una nuova lavorazione solo se è anche disponibile il prodotto dal supermarket B.

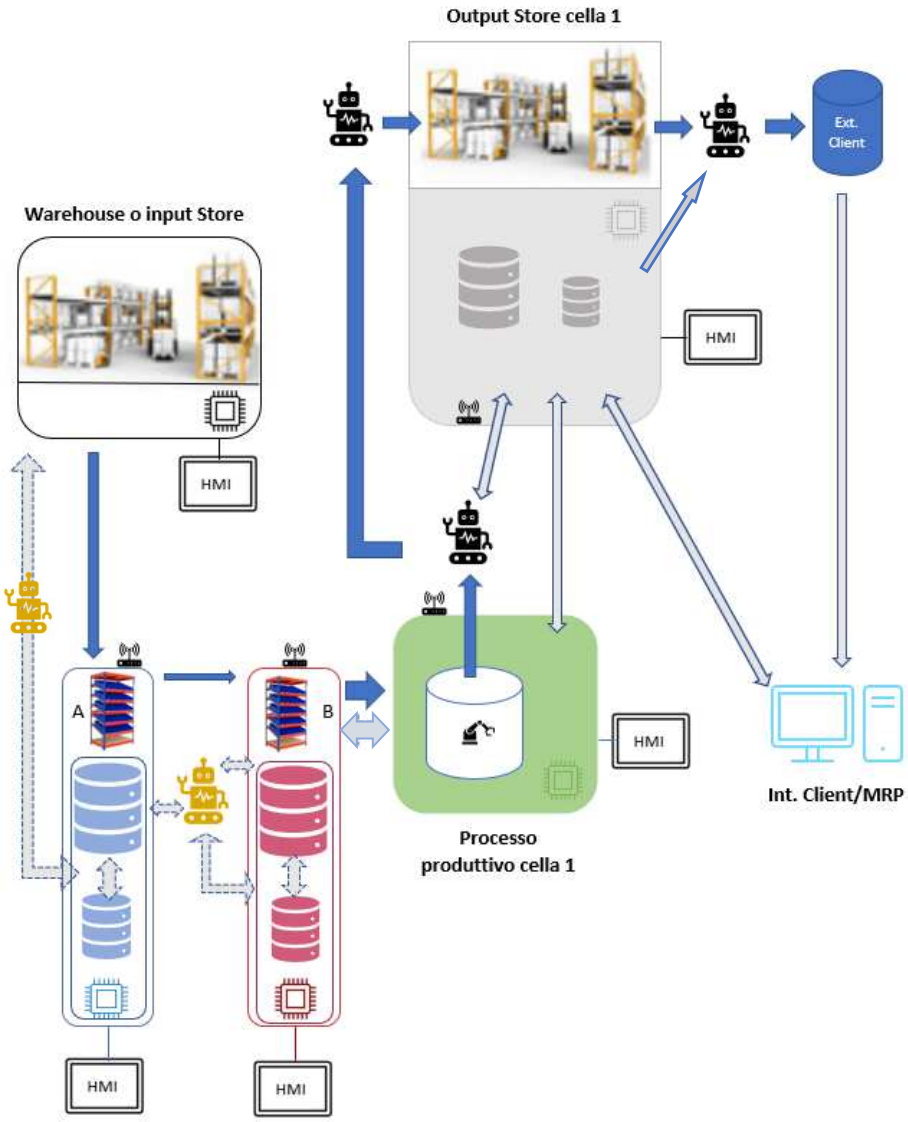


Fig. 3.2.2: Dispositivi e flussi informativi (frecce in grigio) tra i dispositivi connessi e flussi di prodotto (frecce in blu) della catena produttiva.

Nella legenda **L1** sono descritte le funzionalità delle componenti software associate ai dispostivi della catena di produzione di figura 3.2.2.

3.3. Semantica di comunicazione tra dispositivi

Definiamo una semantica di base per la comunicazione tra le interfacce embedded dei dispositivi.
Ogni richiesta di ordine di produzione e di trasporto prodotto, di inserimento di un prodotto in un supermarket o un magazzino, ritiro prodotto da un supermarket o magazzino avviene tramite l'emissione di un segnale/evento *REQ* sempre accoppiato ad un segnale *DATA* contenente l'informazione *e-kanban* dell'ordine di produzione, trasporto o inserimento e ritiro prodotto. La conferma della presa in carico di una richiesta e la sua successiva evasione da parte di un dispositivo avviene con l'emissione di un segnale *CNF* accoppiato ad un dato.

Una e-Kanban di produzione è una struttura dati che in generale contiene: numero della parte o del lotto prodotti (o *id*), descrizione della parte, quantità della parte, processo produttivo che esegue l'ordine, processo/dispositivo che riceve l'ordine.

Una e-Kanban di trasporto è una struttura dati che in generale contiene: numero della parte o *id* della parte, descrizione della parte, quantità, magazzino di partenza, magazzino di arrivo del materiale.

Legenda L1 dei componenti software dei dispositivi di figura 3.2.2

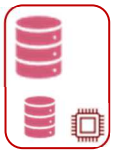
Simbolo grafico che rappresenta la componente software embedded di un dispositivo associato ad un componente fisico della catena di produzione.

kanban digitale (e-kanban): variabile strutturata come definita nel paragrafo 3.3 utilizzata dal software embedded di un componente come dato associato ad un ordine di produzione (kanban di produzione) o trasporto (kanban di trasporto). In generale, nella catena di produzione, l'informazione kanban viaggia insieme ad uno specifico prodotto fisico e ne traccia quindi lo spostamento. Nel database di un componente questa variabile è utilizzata per rappresentare i dati circa il prodotto stesso.

IN_SUP: Rappresenta il software embedded completo associato al supermarket A. È la composizione degli oggetti **IN_SUP_DB** e **IN_SUP_TBOARD** che gestiscono rispettivamente un database interno di prodotti del supermarket A e la struttura dati preposta ad immagazzinare gli ordini di trasporto/rifornimento (o e-kanban di trasporto)**.** Presenta tre interfacce esterne, una per la gestione delle richieste di prodotto da parte di dispositivi *consumers* (cioè l'operatore di trasporto lato consumer che nel caso specifico è il robot colorato in giallo tra i supermercati A e B), un'altra per le richieste di inserimento prodotto nel supermarket da parte di dispositivi *producers* (operatore di trasporto lato producer che in Fig. 3.2.2 è il robot lato Warehouse del loop di trasporto che non consideriamo in questa trattazione). La terza interfaccia invia gli ordini di trasporto (kanban di trasporto) sempre all'operatore preposto lato Warehouse.

IN_SUP_DB: Componente di **IN_SUP** che gestisce un database/struttura dati dei prodotti fisici presenti nel supermarket di ingresso **A**. Un prodotto è rappresentabile nel database o nella struttura dati da una variabile tabella o variabile strutturata e-kanban contenente le informazioni del prodotto. Incorpora le interfacce lato *producer* e *consumer* descritte in **IN_SUP**. Inoltre, presenta una terza interfaccia per mezzo della quale invia al collector **IN_SUP_TBOARD** gli ordini kanban di trasporto prodotto.

IN_SUP_TBOARD: Componente Board/Collector **di IN_SUP**, software che partecipa al generico loop e-kanban di trasporto tra Warehouse e supemarket **A**. Il Collector immagazzina le e-kanban trasmesse da **IN_SUP_DB** e le invia al dispositivo di warehouse o ad un operatore di trasporto come richiesta di trasporto di un nuovo prodotto/componente (e-kanban di trasporto). In genere un collector può eseguire operazioni generiche quali il conteggio delle kanban circolanti, cioè il numero degli ordini di trasporto emessi per quel particolare prodotto.



OUT_SUP: rappresenta il software embedded completo associato al supermarket **B**. È la composizione degli oggetti **OUT_SUP_DB** e **OUT_SUP_TBOARD** che gestiscono rispettivamente un database interno di prodotti del supermarket, la struttura dati preposta ad immagazzinare gli ordini di trasporto (o e-kanban di trasporto) e le comunicazioni lato *consumer* e *producer***.** Presenta tre interfacce esterne, una per la gestione delle richieste di prodotto da parte di dispositivi *consumers* (nel caso specifico il processo produttivo), una per le richieste di inserimento prodotto nel supermarket da parte di dispositivi *producers* (nel caso specifico il robot colorato in giallo tra A e B), la terza interfaccia invia le richieste/ordini di trasporto (kanban di trasporto) all'operatore preposto (nel caso specifico sempre il robot colorato in giallo tra A e B). Le prime due interfacce sono incorporate da **OUT_SUP_DB**, la terza è incorporata in **OUT_SUP_TBOARD.**

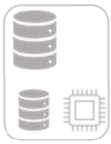
OUT_SUP_DB: Rappresenta il software embedded di gestione del database/struttura dati di prodotti ospitati nel supermarket **B** con le informazioni e-kanban di prodotto/articolo. Incorpora le interfacce lato *producer* e *consumer* descritte in **OUT_SUP**. Inoltre, presenta una terza interfaccia interna per mezzo della quale invia al collector **OUT_SUP_TBOARD** gli ordini kanban di prodotto (anche dette kanban di trasporto non attive).

OUT_SUP_TBOARD: software embedded del Board/Collector per la gestione e memorizzazione delle e-kanban di trasporto provenienti da **OUT_SUP_DB.** Esegue l'invio di ordini di trasporto (o e-kanban di trasporto) all'operatore preposto (nel caso specifico il robot colorato in giallo, come descritto per la terza interfaccia di OUT_SUP).

BASIC_PROCESS _CELL1: Processo produttivo fisico.

Interfaccia embedded del processo produttivo **BASIC_PROCESS _CELL1** che gestisce le comunicazioni con le interfacce dei dispositivi connessi al processo stesso.

TRANSPORT_OPERATOR: Interfaccia embedded di un Robot o di un operatore umano aumentato per le comunicazioni di trasporto prodotto tra supermarket A e B e ordini di trasporto e-kanban.



OUT_STORE: componente software associata al magazzino di uscita (*Output store cella 1*). È la composizione degli oggetti **OUT_STORE_DB** e **OUT_STORECollector** che gestiscono rispettivamente un database interno di prodotti del magazzino e la struttura dati preposta ad immagazzinare gli ordini di produzione (e-kanban di produzione)**.** Presenta tre interfacce esterne, una per la gestione delle richieste di prodotto da parte

dell'interfaccia *consumer* (*If_cons)*, un'altra per le richieste di inserimento prodotto in magazzino da parte dell'interfaccia *producer* (*If_prod*) (Figura 3.2.5). Una terza interfaccia esterna invia le e-Kanban di produzione al processo produttivo.

OUT_STORE_DB: Software per la gestione del database/struttura dati di prodotti ospitati nel magazzino di uscita o isola esterna al processo produttivo. Riceve richieste di inserimento prodotto in magazzino da parte di *If_prod* e richieste di estrazione prodotto effettuate da *If_cons.* Invia le e-Kanban di produzione al collector **OUT_STORECollector.**

OUT_STORECollector: Collector del loop kanban di produzione tra magazzino di uscita e processo produttivo. Il collector immagazzina le e-kanban di produzione inviate da **OUT_STORE_DB** che poi invia al processo produttivo.

OUT_STORE_FULL: software completo del magazzino di uscita *Output Store cella 1* di Figura 3.2.2. Comprende le interfacce *If_prod* e *If_cons* (Figura 3.2.5), **OUT_STORE_DB** e **OUT_STORECollector**. *If_prod* gestisce le richieste di inserimento prodotto in magazzino provenienti dal processo produttivo, richiede l'inserimento della e-kanban del prodotto in **OUT_STORE_DB** e richiede a **Trobot** l'inserimento del prodotto nel magazzino fisico**.** L'interfaccia *If_cons* gestisce le richieste di prodotto da parte del Cliente, l'estrazione della e-kanban di prodotto da **OUT_STORE_DB** e richiede a Trobot l'estrazione del prodotto dal magazzino fisico.

TRobot: robot per per pick-up del prodotto finito dalla cella 1 di produzione con destinazione magazzino fisico oppure prelievo prodotto dal magazzino fisico di uscita per la consegna al cliente. È il magazzino **OUT_STORE_FULL** che gestisce i Trobot.

INT_CLIENT, Software Client interno alla fabbrica per la gestione degli ordini del cliente esterno da inviare al magazzino di uscita.

Pannello HMI

Flusso informativo/dati/richieste di prodotto

Flusso di prodotto

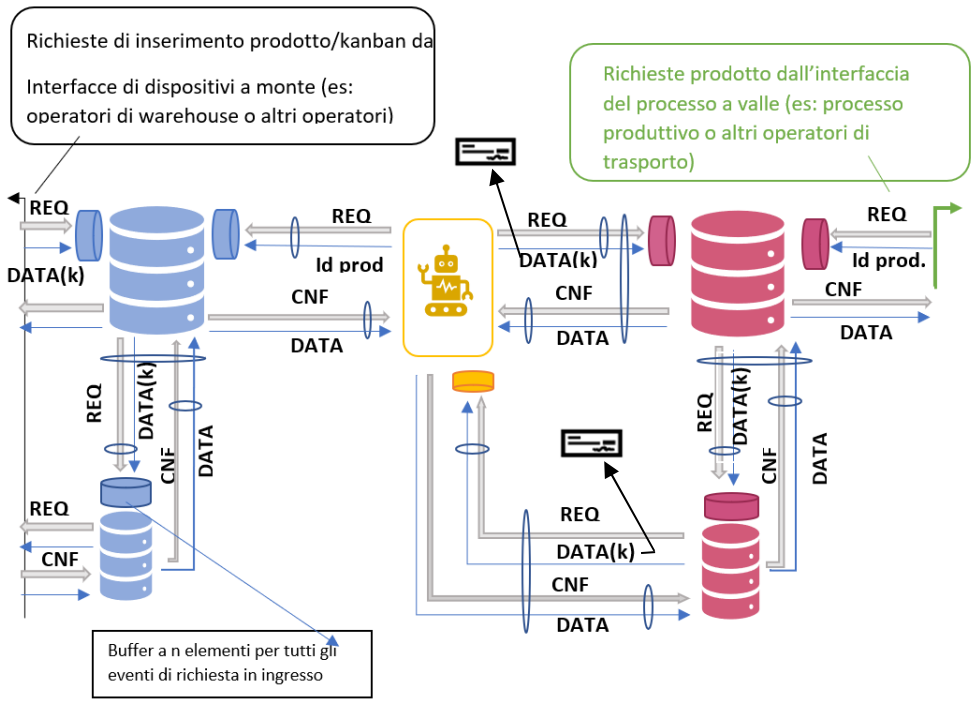


Fig. 3.2.3: Kanban loop di trasporto tra supermarket A (viola chiaro) e B (viola scuro), relativa semantica dei segnali di comunicazione tra dispositivi

In Figura 3.2.3 sono rappresentate le componenti software dei supermarket A (viola chiaro) e B (viola scuro) e i segnali evento per il controllo di un loop di trasporto Kanban tra B e A al cui interno è inserito il Robot per il pick-up e trasporto di materiale tra A e B. In riferimento alle simbologie grafiche di legenda **L1** si evidenziano, da destra in alto, il componente **OUT_SUP_DB** e il collector **OUT_SUP_TBOARD** che costituiscono il software completo **OUT_SUP** del supermarket B.
Sul lato sinistro troviamo il componente **IN_SUP_DB** e il collector **IN_SUP_TBOARD** che costituiscono il software completo **IN_SUP** del supermarket A. Tutti gli ingressi per gli eventi REQ sono "bufferizzati" per richieste asincrone e contemporanee.

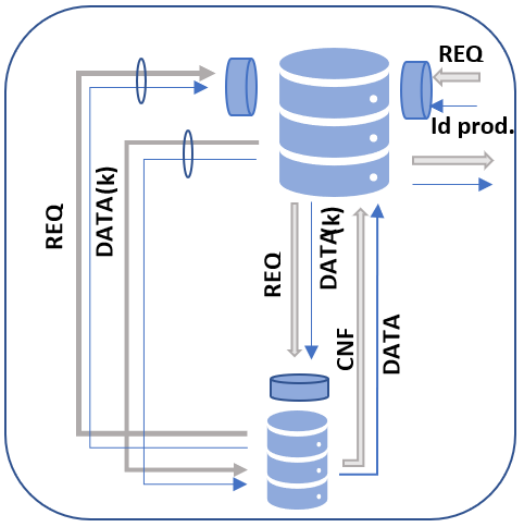


Fig. 3.2.4: Supermarket A in loop chiuso auto-alimentante per la simulazione di risorse infinite (prodotto sempre disponibile) nel supermarket.

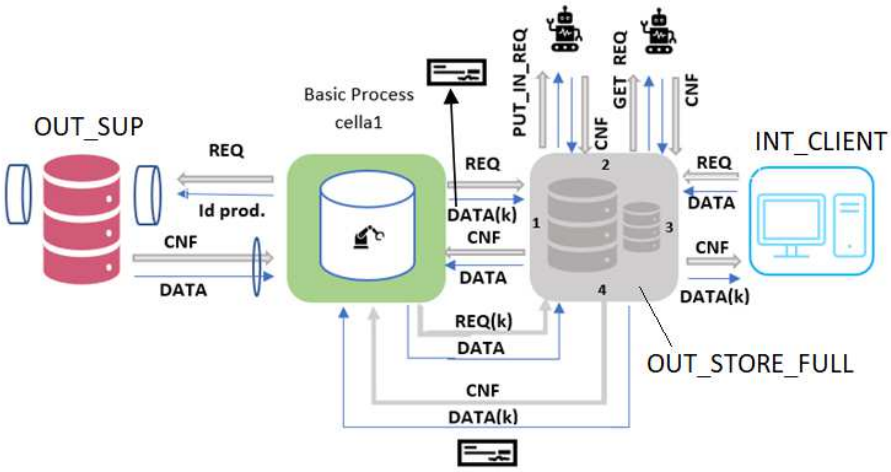


Fig. 3.2.5: Componenti software dei dispositivi inseriti nel Kanban loop di produzione (componente del magazzino di uscita OUT_STORE_FULL - grigio, processo produttivo -verde)

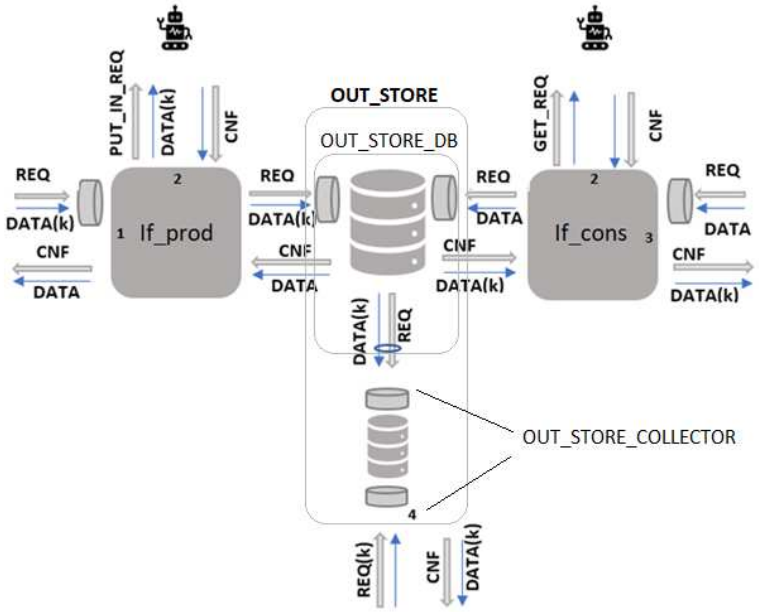


Fig. 3.2.6: Componenti software di OUT_STORE_FULL (If_cons, OUT_STORE_DB, OUT_STORECollector, If_prod)

In Figura 3.2.5 sono mostrate le componenti software dei dispositivi inseriti nel Kanban loop di produzione e cioè le componenti del magazzino di uscita (colore grigio) e del processo produttivo (colore verde). La componente software **OUT_STORE_FULL** del magazzino in oggetto è esploso ulteriormente in Figura 3.2.6 nelle sue componenti **If_cons**, **If_prod**, **OUT_STORE_DB** e il collector **OUT_STORECollector** di legenda **L1.**

Con riferimento alla legenda **L1** dei componenti, alle Figure 3.2.5, 3.2.6**,** le corrispondenze date dai colori e la semantica di comunicazione definita, il controllo dettagliato agisce nel modo seguente**:**

Comunicazione lato Client

Similmente a quanto descritto nel capitolo 2 il controllo segue la logica del metodo Kanban tradizionale. La sorgente di ordini **INT_CLIENT** recupera un ordine dal suo buffer interno, emette questo ordine tramite un *REQ* al magazzino, recepito dall'interfaccia *If_cons*, e attende una sua conferma *CNF*. Durante questa attesa, l'interfaccia *If_cons* emette una REQ bloccante al database **OUT_SUP_DB** del magazzino. Se nel database il prodotto è disponibile, **OUT_SUP_DB** esegue una richiesta non bloccante a **OUT_STORECollector** di inserimento di una e-Kanban relativa ad una nuova richiesta di produzione. È solo al CNF di **OUT_STORECollector (**emesso se il collector non è saturo) che **OUT_SUP_DB** può inviare il suo CNF a *If_cons,* che a sua volta può richiedere al robot (con un GET_REQ) di ritirare il prodotto richiesto dal magazzino fisico. Quest'ultima richiesta è bloccante. Alla conferma del robot, *If_cons* può trasmettere a **INT_CLIENT** la conferma con i dati di prodotto relativo all'ordine richiesto.

Loop Kanban di produzione

Se il processo produttivo ha terminato una produzione emette un *REQ* bloccante al magazzino per l'inserimento del prodotto finito attraverso l'interfaccia *If_prod*. A questo punto *If_prod* emette una *REQ* bloccante al database del magazzino **OUT_STORE_DB**. Se il database del magazzino non è saturo, **OUT_STORE_DB** emette il CNF all' interfaccia *If_prod* che può emettere una richiesta bloccante di **PUT_IN_REQ** al robot per l'inserimento del prodotto nel magazzino fisico. Al CNF del robot può essere emesso il CNF al processo produttivo. Quest'ultimo può così iniziare una nuova produzione, quindi invia una REQ bloccante a **OUT_STORECollector** per ricevere una e-Kanban di produzione se disponibile nel collector. Al CNF di **OUT_STORECollector** può emettere una REQ bloccante a **OUT_SUP_DB** (supermarket B) per una nuova fornitura di prodotto. Al CNF di **OUT_SUP_DB** il processo produttivo può iniziare una nuova lavorazione.

Loop Kanban di trasporto

Quando il processo produttivo emette una REQ di prodotto a **OUT_SUP_DB** (supermarket B) e il prodotto è disponibile, **OUT_SUP_DB** emette una REQ non bloccante di inserimento e-Kanban di trasporto verso il collector **OUT_SUP_TBOARD.** Solo al **CNF** di **OUT_SUP_TBOARD** potrà rilasciare un CNF al processo produttivo**.**

Se il collector **OUT_SUP_TBOARD** contiene delle e-kanban emette una REQ di trasporto materiale non bloccante al robot (colorato in giallo in Figura 3.2.3). Il robot si dirige al supermarket A ed emette una REQ di prodotto a **IN_SUP_DB (**supermarket A). Al CNF di **IN_SUP_DB** fa il pick-up e si dirige a rifornire il supermarket B, **OUT_SUP_DB.**

Non ci sono altri loop di trasporto a monte di **IN_SUP_DB** (supermarket A). Connettendo l'ingresso producer di **IN_SUP_DB** con l'uscita del collector **IN_SUP_TBOARD** si genera un loop di e-kanban che auto-alimenta **IN_SUP_DB** senza necessità di altri componenti (Figura 3.2.4). In questo caso la e-Kanban di trasporto proveniente da **IN_SUP_TBOARD** emula una richiesta di inserimento prodotto in **IN_SUP_DB**. Altri loop kanban di trasporto o processi produttivi possono comunque essere inseriti a monte della catena riutilizzando gli stessi componenti.

Nel paragrafo a seguire sarà implementata l'applicazione distribuita in standard IEC 61449 del controllo Kanban appena descritto che potrà essere calata in futuro su dispositivi hardware reali e modificata per scenari specifici. Questa fase presupporrà un secondo step di progettazione, cioè quello di progettare le SIFB per i driver dei dispositivi hardware scelti, se non già forniti dai produttori.

3.4. Applicazione Shop-Floor in standard IEC-61499

In questo paragrafo il modello del controllo Kanban distribuito con semantica ad eventi descritto nel paragrafo precedente è implementato a livello applicazione in standard IEC 61499 utilizzando l'ambiente di progettazione grafica 4Diac accoppiato alle librerie di run-time FORTE per la generazione del codice in ambiente Windows. La libreria Shop-Floor dell'applicazione distribuita realizzata in questo progetto consta di FB progettate da zero. (Uno svantaggio dell'ambiente di programmazione incontrato nella progettazione, almeno nella versione 1.13 del tool, è l'impossibilità di compilare a run-time, tramite il preposto interprete LUA, i blocchi funzionali che utilizzano variabili "custom" strutturate (ad esempio quelle create in questo progetto). Ciò comporta la necessità di esportare in FORTE il codice C++ dei blocchi funzionali creati in 4Diac, e ricompilarli insieme al codice FORTE).

System Configuration – Configurazione di sistema

In Figura 3.4.1 è mostrata la configurazione del sistema distribuito che implementa le funzionalità dei dispositivi dello Shop-floor descritti nel paragrafo precedente, dei quali è mantenuta la stessa codifica dei colori. L'applicazione di ogni dispositivo è ospitata nella rispettiva risorsa IEC 61499 gestita dalla rispettiva sessione FORTE, che gestisce anche le comunicazioni di rete, in questo caso basate su protocollo Ethernet (indirizzi localhost:port). In futuro potranno essere utilizzati e testati diversi protocolli offerti da questo ambiente di progettazione, come le principali librerie dei protocolli di livello di campo, ma anche OPC-UA, MQTT etc. per le comunicazioni con supervisori, agenti o sistemi informativi.

Note implementative:

- Ogni applicazione relativa a ciascun dispositivo utilizza SIFB *publish/subscribe* per l'invio e la ricezione in rete degli eventi di richiesta e conferma REQ/CNF.

- Come primo approccio è considerato il controllo di una tipologia di prodotto; i database di magazzini e supermarket, come definiti nel paragrafo precedente, sono implementati con FB che realizzano code FIFO a dimensione finita impostabile da interfaccia. Una configurazione multiprodotto è comunque ottenibile con modifiche minimali del codice del blocco funzionale FIFO fornito, in modo da aggiungere internamente una gestione multi-Array e accoppiando all'evento POP dell'interfaccia FIFO un dato contenente l'*id* del prodotto per la selezione interna dello specifico array. L'evento PUSH è già associato al dato kanban che contiene l'informazione *id* di prodotto necessaria. Il sistema continuerà a lavorare in modo trasparente con la nuova tipologia di coda FIFO.
  Vista la modularità del progetto, in un prossimo step di progettazione si potrà testare la possibilità di sostituire le attuali FB FIFO con database di terze parti posti su un server remoto o in cloud con minime modifiche al codice. A tale scopo potranno

essere utilizzate SIFB per le comunicazioni del magazzino con il database sfruttando i protocolli forniti dallo stesso tool di progettazione.

- Code FIFO sono utilizzate dai Collectors per l'immagazzinamento delle e-kanban degli ordini di produzione o trasporto scambiate tra i dispositivi. Il numero delle e-kanban in un collector definisce la dimensione del buffer FIFO associato, impostabile allo start-up del sistema (attualmente non in modalità run-time).
  In prossimi steps di progettazione dovrà essere testata la caratteristica di riconfigurazione dinamica del codice a run-time (cancellazione di FB, creazione di nuove FB).

- È definita la variabile strutturata di tipo *kanbanc* per la memorizzazione delle informazioni relative alle e-kanban di trasporto e produzione con campi definiti come descritto nei paragrafi precedenti.
  È inoltre definita un'altra variabile strutturata di tipo *cordersinfo* utilizzata da INT_CLIENT contenente i dati completi di un ordine evaso e cioè, numero *id* dell'ordine, numero *id* della parte richiesta, descrizione della parte, provenienza, dimensioni del lotto e numero del lotto prodotto.

- Sono stati realizzati buffer a 1 elemento (tipizzati per dati interi, stringa e kanbanc di tipo strutturato) per evitare la perdita dei messaggi REQ contemporanei nelle comunicazioni asincrone.

- La circolazione delle e-kanban è tracciabile in ogni dispositivo grazie a SIFB che permettono di stampare su file le informazioni relative alle e-kanban.

- Per ogni applicazione esistono moduli FB per l'inizializzazione dei buffer FIFO relativi ai supermarket, il magazzino e rispettivi Collectors.

- Gli algoritmi dei grafi di esecuzione degli eventi (ECC) sono scritti in ST (Structured Text), unica opzione possibile nella versione del tool 4Diac utilizzata per questo progetto.

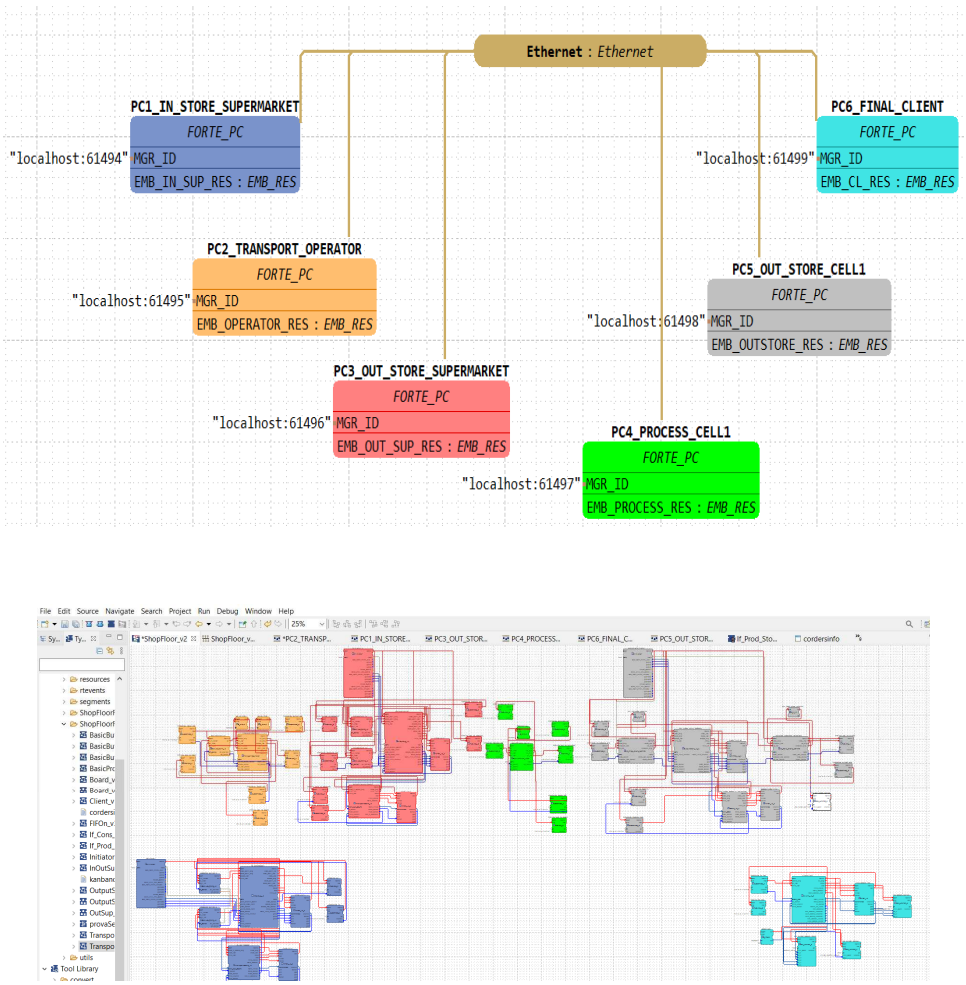
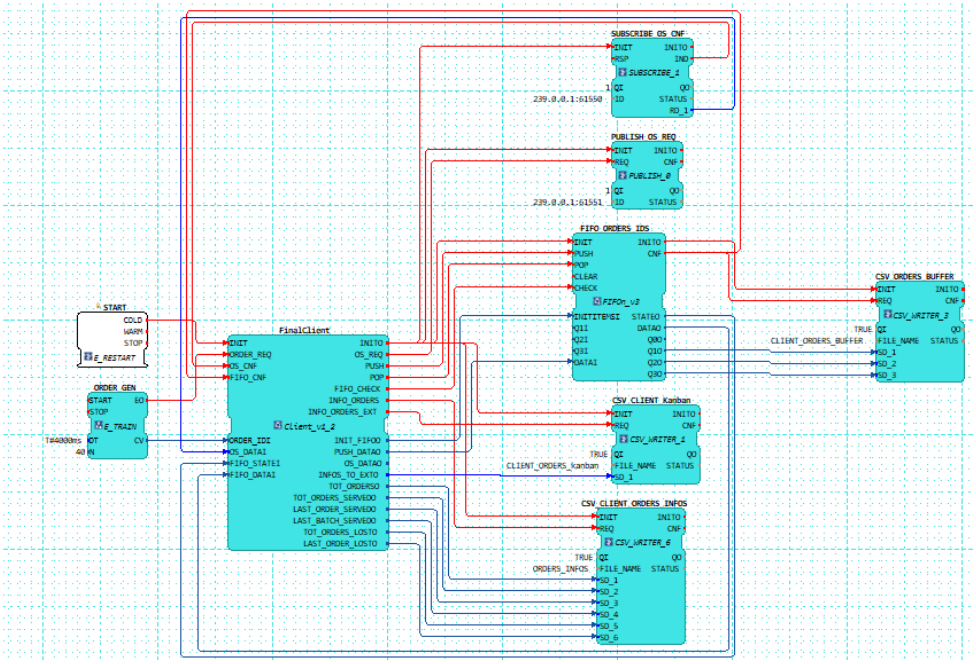


Fig. 3.4.1: Applicazione Shop-Floor – Configurazione di sistema

Di seguito sono descritte le applicazioni dei singoli componenti di Figura 3.4.1 che mappano le funzionalità dei dispositivi dello shop floor elencati in legenda L1, sono quindi indicati i blocchi funzionali con la descrizione delle interfacce e i rispettivi grafi di esecuzione degli eventi o macchine a stati.

Applicazione: ***INT_CLIENT***

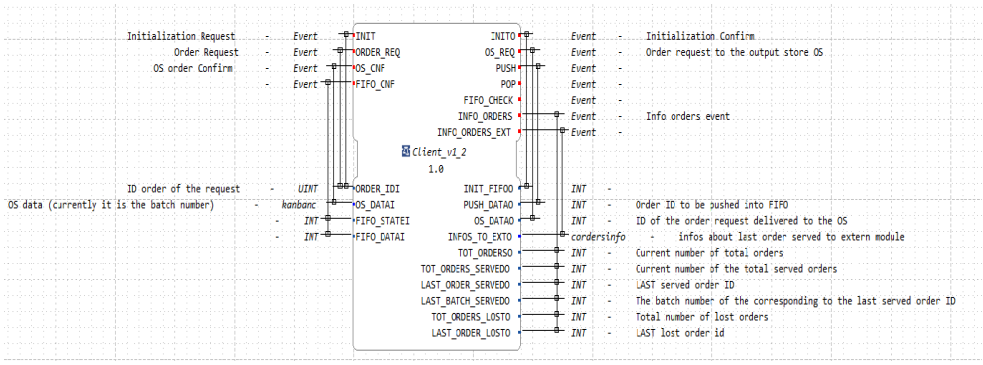
Dispositivo: **PC6_FINAL_CLIENT**
Tipo: **FORTE_PC**
Risorsa: **EMB_CL_RES**



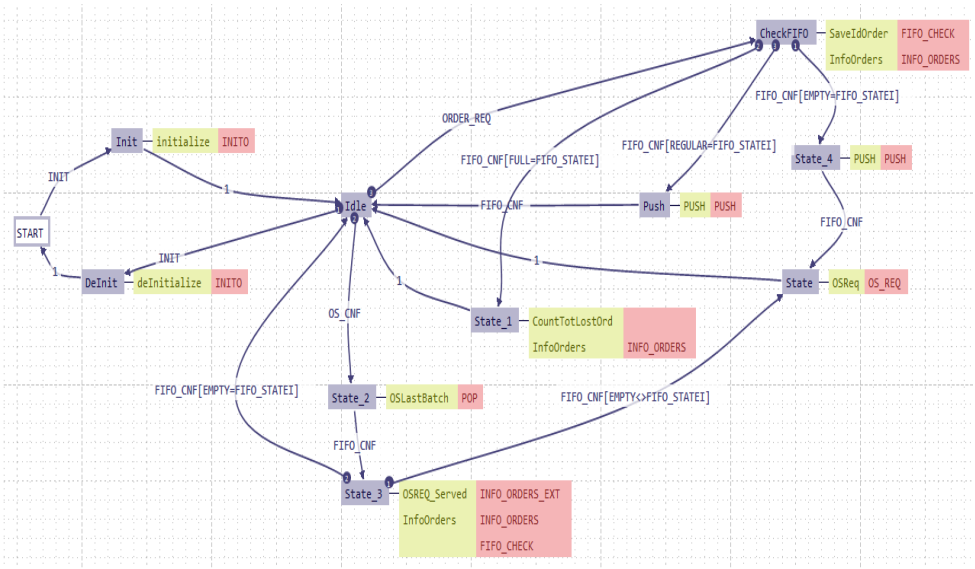
Dispositivo **PC6_FINAL_CLIENT**, risorsa **EMB_CL_RES**

PC6_FINAL_CLIENT è il dispositivo client che mappa **INT_CLIENT** di legenda L1**,** preposto alla generazione degli ordini di prodotto da inviare al magazzino. È composto da un blocco funzionale (*ORDER_GEN*) che genera un numero totale N di eventi (impostabile, in questo caso 40) ad un certo *rate* (impostabile, in questo caso posto a 1 ordine ogni 4 secondi), inviati al modulo *FINALCLIENT,* il quale genera l'ordine vero e proprio da inviare in magazzino. Il modulo FINALCLIENT gestisce le informazioni in tempo reale relative allo stato degli ordini (che sono inviate ai 2 moduli per la stampa CSV su file) e la FIFO

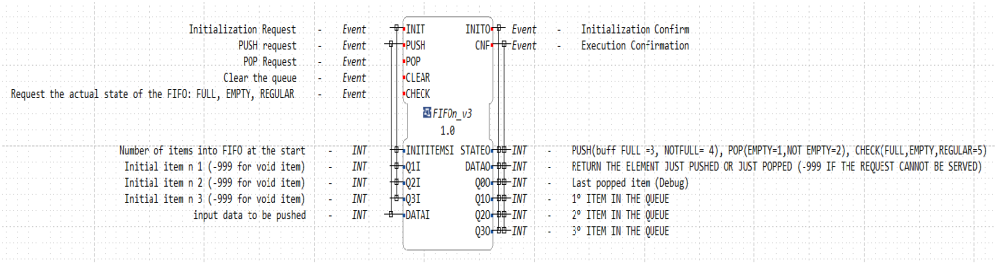
(FIFO_ORDERS_IDS) per la memorizzazione degli ordini emessi in attesa di essere inviati al magazzino. È ammessa perdita degli ordini qualora la FIFO sia piena. Un nuovo ordine è emesso a seguito dell'evasione del precedente da parte del magazzino.
Un modulo CSV (CSV_CLIENT_ORDERS_INFOS) stamperà gli ordini totali correnti, gli ordini serviti correnti, il numero di *id* relativo all'ultimo ordine servito, il numero di *id* dell'ultimo lotto (batch) di prodotto servito, il totale degli ordini persi e l'*id* dell'ultimo ordine perso.
Inoltre, all'evasione di un ordine da parte del magazzino (prodotto disponibile per il ritiro in magazzino), quest'ultimo invia a FINALCLIENT un CNF con il numero del lotto prodotto, che FINALCLIENT aggiunge ad una variabile struttura di tipo "*cordersinfos*" contenente i dati completi dell'ordine evaso.
Un altro modulo CSV (CSV_Client_kanban) stamperà su file le informazioni e-kanban relative all'ordine servito dal magazzino corrente. In versioni future, alla stampa CSV su file sarà aggiunta un'interfaccia utente.
Sono inoltre utilizzate, una SIFB per il *publishing* del segnale REQ in rete e diretto al magazzino e una SIFB per la ricezione del segnale CNF dal magazzino (il cui dato associato è l'informazione dell'ordine evaso).
Di seguito sono indicati i moduli FB con i rispettivi diagrammi degli stati (ECC).



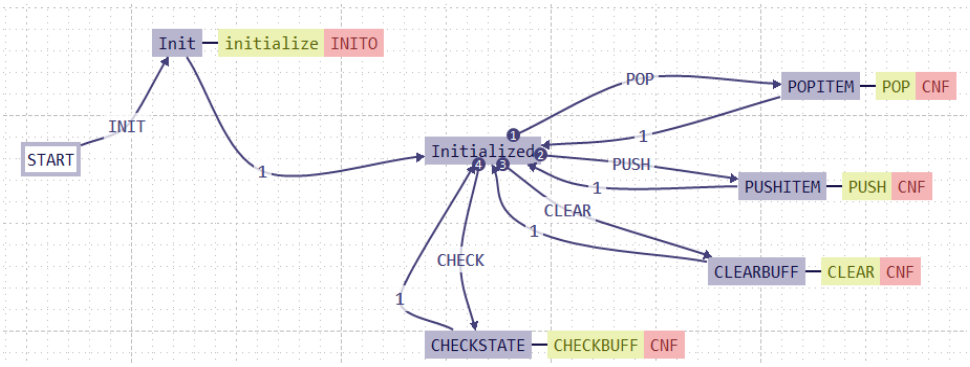
Function Block **FinalClient,** interfaccia di tipo **Client_v1_2**



Function Block **FinalClient** – Diagramma degli stati (ECC)



Function block **FIFO_ORDER_IDS** – Interfaccia di tipo **FIFOn_v3**



Function block **FIFO_ORDER_IDS** – Diagramma degli stati (ECC)

Applicazione: ***OUT_STORE_FULL***

Dispositivo: **PC5_OUT_STORE_CELL**
Tipo: **FORTE_PC**
Risorsa:**EMB_OUTSTORE_RES**

È l'applicazione relativa al magazzino di uscita che mappa OUT_STORE_FULL (Fig. 3.2.6). Si compone di blocchi funzionali per i quali vale la seguente mappa con i dispositivi software di legenda L1:

OUTSTORE_CONSUMER_INTERFACE -> ***If_cons***

Responsabilità:

- Riceve dal *consumer* (INT_CLIENT) un ordine di prodotto finito attraverso il modulo *subscriber* SUBSCRIBE_CLIENT_REQ.
- Richiede al modulo di gestione dei prodotti del magazzino (OUT_STORE_CELL1_MANAGER) il nulla osta per il ritiro del prodotto riferito all'ordine cliente.
- Richiede al robot di magazzino il ritiro fisico del prodotto riferito all'ordine. Tale richiesta è modellata come servizio generico espletato istantaneamente dal modulo GET_FROM_ STORE_SERVICE (Flip Flop tipo T).
- Invia al cliente, tramite il modulo PUBLISH_CONF_TO_CLIENT, la conferma di prodotto disponibile in magazzino per il ritiro con i dati relativi al prodotto.

PRODUCER_OUTSTORE_INTERFACE -> ***If_prod***

Responsabilità:

- Riceve la richiesta di inserimento prodotto finito in magazzino da parte del processo *producer* a monte (il processo produttivo). Tale richiesta è corredata dai dati associati al prodotto (e-kanban di produzione) e sarà girata al modulo OUT_STORE_CELL1_MANAGER.
- Richiede al modulo di gestione dei prodotti del magazzino (OUT_STORE_CELL1_MANAGER) il nulla osta per l'inserimento in magazzino del prodotto fornito dal processo produttivo.
- Richiede al robot di magazzino l'inserimento fisico in magazzino del prodotto finito. Tale richiesta è modellata come una richiesta di servizio generico espletato istantaneamente dal modulo PUT_IN_ STORE_SERVICE (Flip Flop tipo T).

OUT_STORE_CELL1_MANAGER + STORE_FIFO_Db -> OUT_STORE_dB

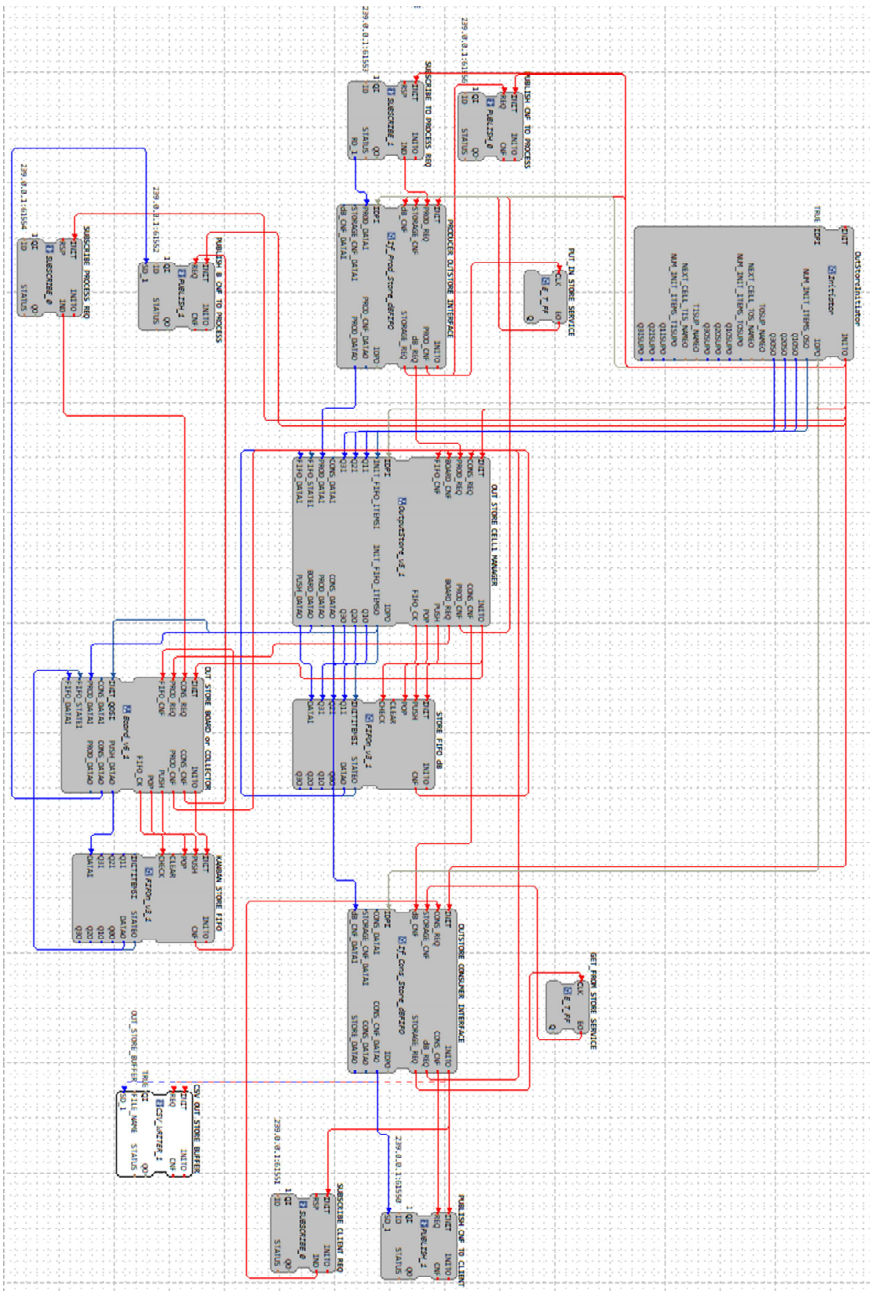
Responsabilità:

- Il modulo OUT_STORE_CELL1_MANAGER gestisce le richieste di inserimento nuovo prodotto in magazzino emesse da PRODUCER_OUTSTORE_INTERFACE e le richieste di estrazione di un nuovo prodotto emesse da OUTSTORE_CONSUMER_INTERFACE. Utilizza il modulo STORE_FIFO_dB per immagazzinare i dati dei prodotti presenti in magazzino e la loro estrazione.
  La gestione lato producer e consumer è "bufferizzata" quindi sono consentite richieste di inserimenti ed estrazioni asincrone e concorrenti provenienti rispettivamente dai moduli PRODUCER_OUTSTORE_INTERFACE e OUTSTORE_CONSUMER_INTERFACE.
  In particolare, alla ricezione di un evento di richiesta inserimento (PUSH), controlla se FIFO_dB è satura. In caso affermativo controlla se c'è una richiesta pendente di estrazione prodotto lato consumer, nel caso evade la richiesta (POP) e può portare a termine il PUSH. In caso negativo esegue il PUSH e controlla se ci sono richieste di consumo pendenti nel buffer lato consumer. In caso affermativo esegue un POP. Ad ogni POP richiede l'inserimento della relativa kanban prodotto (risultato del POP) al modulo OUTSTORE_Board_or_Collector. Questa richiesta non è bloccante, il modulo può continuare a servire richieste di inserimenti lato producer (PUSH) o gestire altri eventi aggiuntivi ma non può servire atre richieste lato consumer finché non è emesso un CNF da OUTSTORE_Board_or_Collector.
- Alla ricezione di una richiesta di inserimento prodotto in magazzino provenienti dall'interfaccia PRODUCER_OUTSTORE_INTERFACE emette un CNF non appena l'operazione è possibile.
  Stesso paradigma alla ricezione di richieste di estrazione prodotto da parte dell'interfaccia CONSUMER_OUTSTORE_INTERFACE.
- Richiede al modulo STORE_FIFO_dB l'inserimento (PUSH) dei dati e-Kanban del prodotto che dovrà essere fisicamente inserito in magazzino.
- Richiede al modulo STORE_FIFO_dB l'estrazione (POP) dei dati e-Kanban del prodotto che dovrà essere fisicamente ritirato dal magazzino.
- Richiede al collector OUTSTORE_Board_or_Collector l'inserimento di un nuovo ordine di produzione e-Kanban al consumo o estrazione di un prodotto dalla struttura dati STORE_FIFO_dB.

OUTSTORE_Board_or_Collector+KANBAN_STORE_FIFO-> OUT_STORECollector

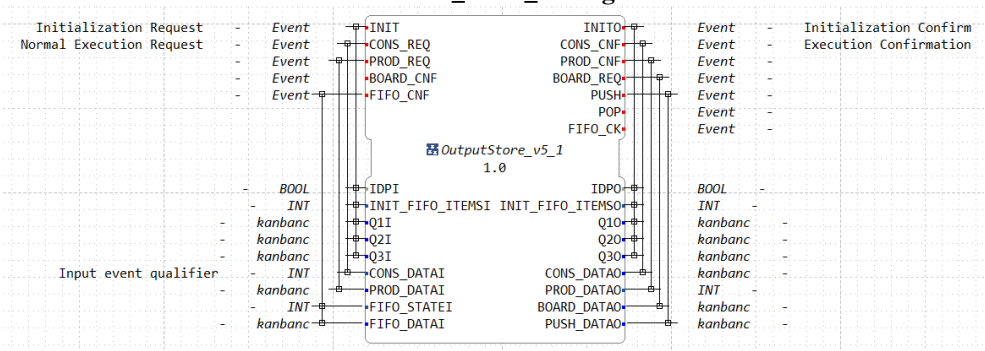
Responsabilità:

- Il modulo OUTSTORE_Board_or_Collector gestisce le richieste di inserimento nuovi ordini di produzione (e-Kanban) ricevute da OUT_STORE_CELL1_MANAGER e le richieste di estrazione ordini di produzione da parte del processo produttivo. OUTSTORE_Board_or_Collector utilizza il modulo KANBAN_STORE_FIFO per immagazzinare gli ordini kanban di produzione.
  OUTSTORE_Board_or_Collector lavora in modalità "bufferizzata" quindi sono consentite richieste di inserimenti ed estrazioni ordini asincrone e concorrenti come visto per OUT_STORE_CELL1_MANAGER**.**
- La ricezione di una richiesta lato processo produttivo avviene attraverso il modulo *subscriber* SUBSCRIBE_PROCESS_REQ. La conferma di ricezione avviene con l'utilizzo del modulo *publisher* PUBLISH_CNF_TO_PROCESS.



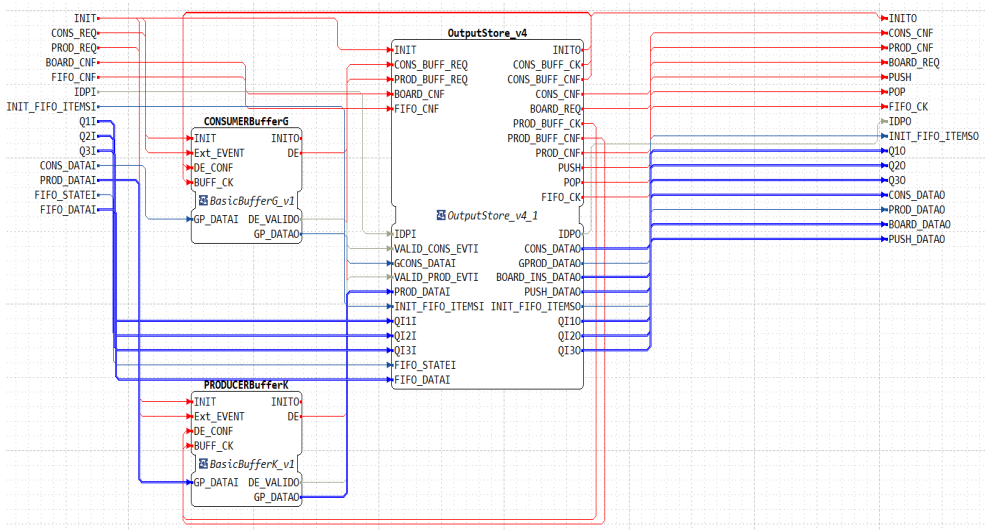
Applicazione dispositivo **PC5_OUT_STORE_CELL1**, risorsa **EMB_OUTSTORE_RES**

OutStore_Cell1_manager



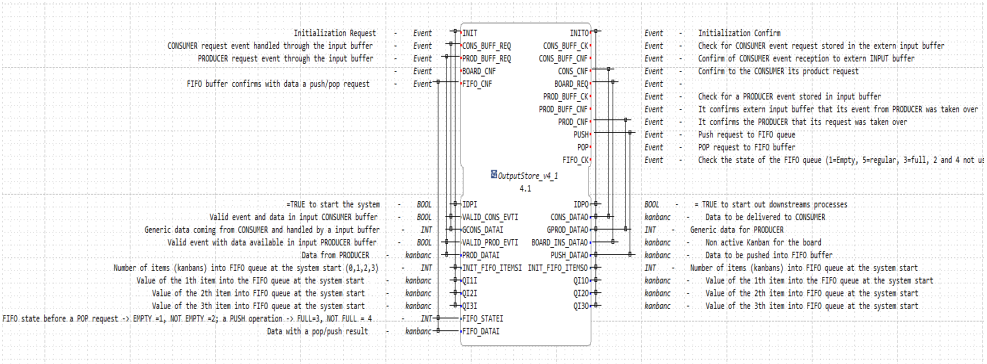
Function Block **OutStore_Cell1_manager** – FB di tipo **OutputStore_v5_1**

OutputStore_v5_1

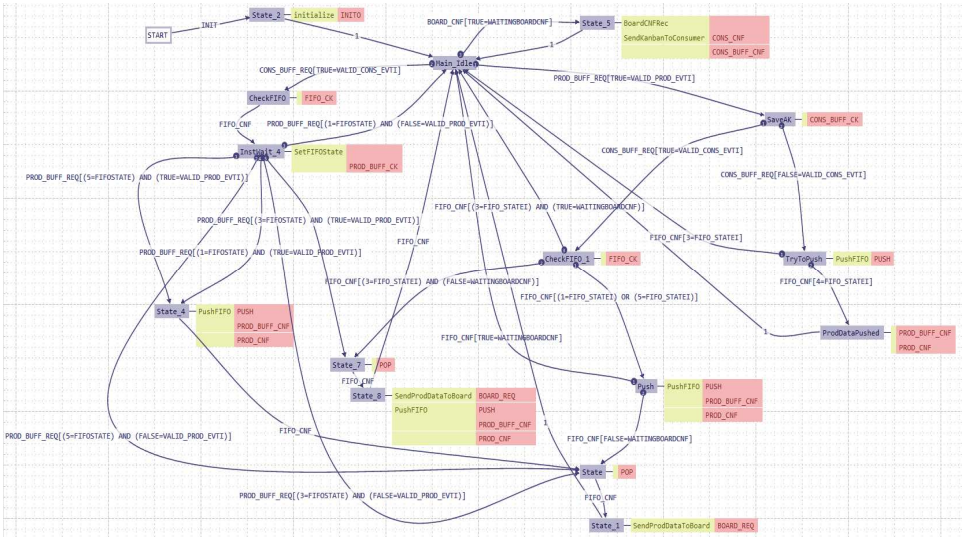


Function block **OutputStore_v5_1**, FB composita per il wrapping dei buffer in ingresso

OutputStore_v4

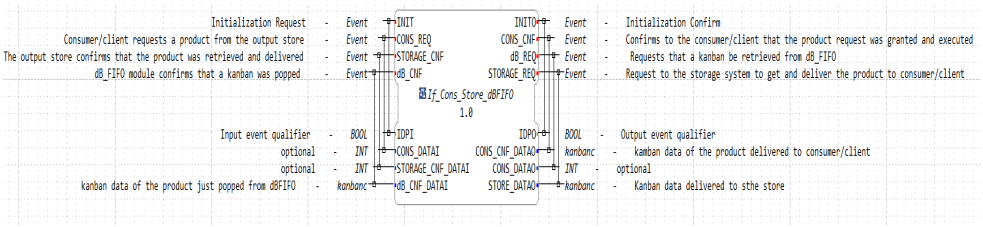


Function Block **OutputStore_v4** – FB di tipo OutputStore_v4_1

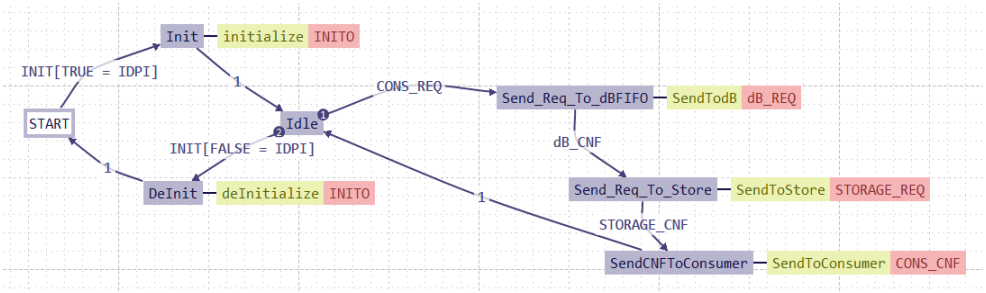


Function Block **OutputStore_v4** – diagramma degli stati (ECC)

OUTSTORE_CONSUMER_INTERFACE

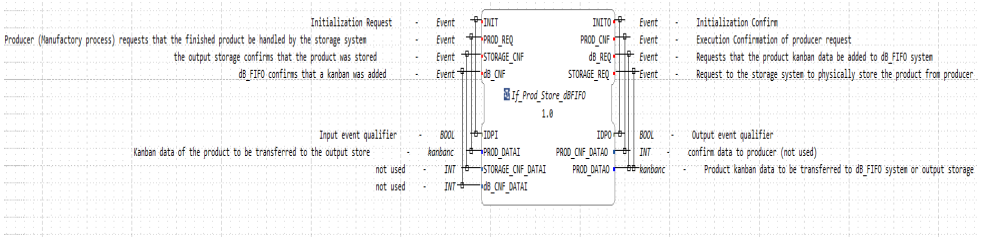


Function block **OUTSTORE_CONSUMER_INTERFACE** – FB di tipo **If_Cons_Store_dBFIFO**

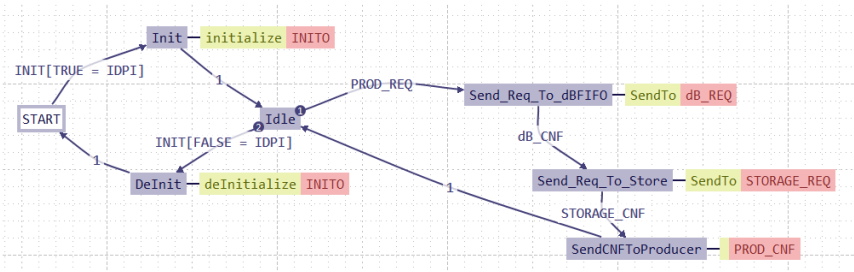


Function block **OUTSTORE_CONSUMER_INTERFACE** – diagramma degli stati (ECC)

PRODUCER_OUTSTORE_INTERFACE

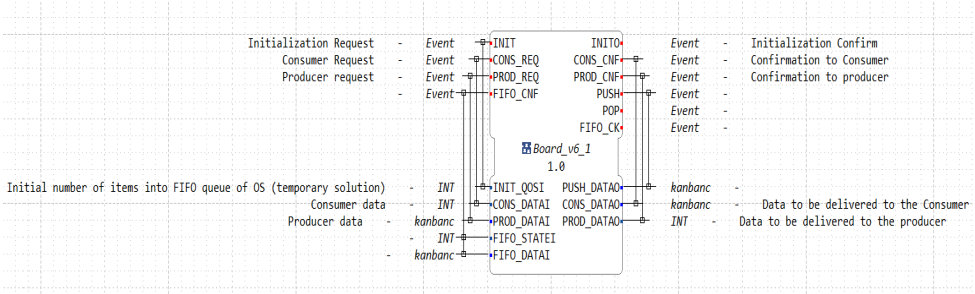


Function block **PRODUCER_OUTSTORE_INTERFACE** – FB di tipo **If_prod_store_dBFIFO**



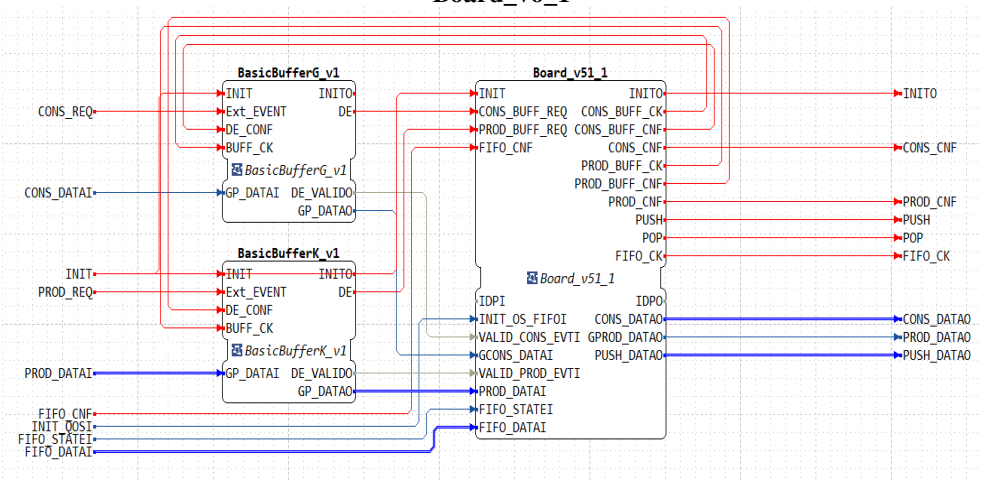
Function block **PRODUCER_OUTSTORE_INTERFACE** – diagramma degli stati (ECC)

OUTSTORE_BOARD_or_COLLECTOR



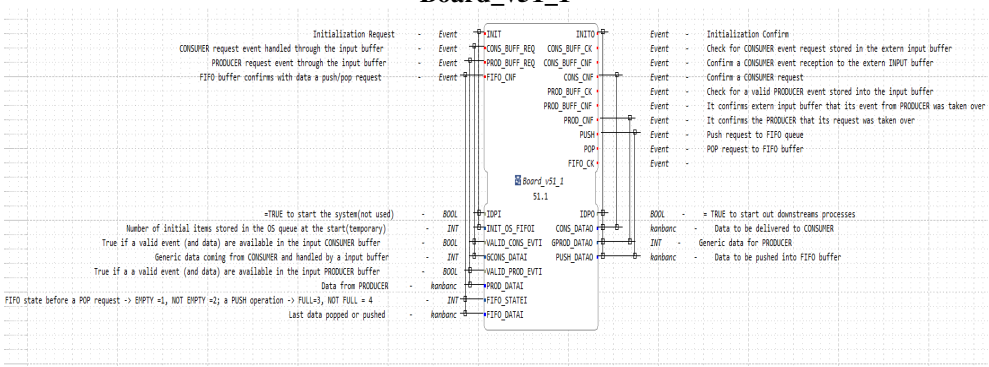
Function block **OUTSTORE_BOARD_or_COLLECTOR** – FB composita di tipo **Board_v6_1**

Board_v6_1

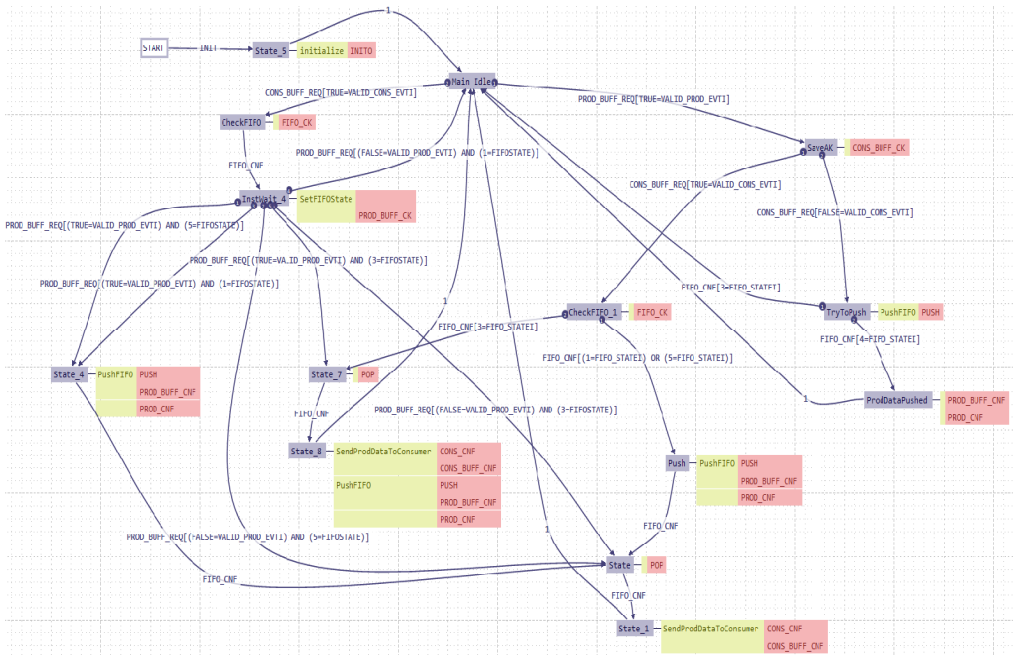


FB **Board_v6_1** - FB composita per il warping dei buffers BasicBufferG/K_v1

Board_v51_1

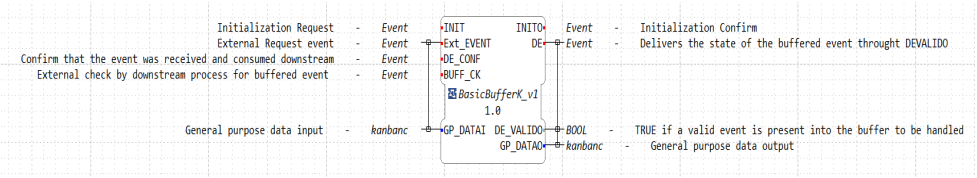


FB **Board_v51_1** – Interfaccia



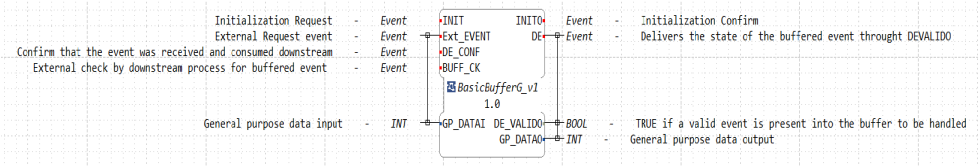
FB **Board_v51_1** – Diagramma degli stati (ECC)

BasicBufferK



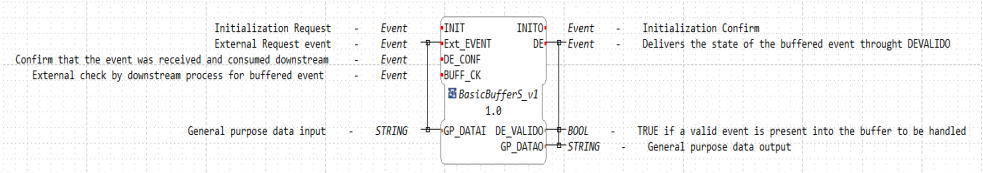
FB **BasicBufferK** – tipo **BasicBufferK_v1**, buffer per dato **kanbanc**

BasicBufferG

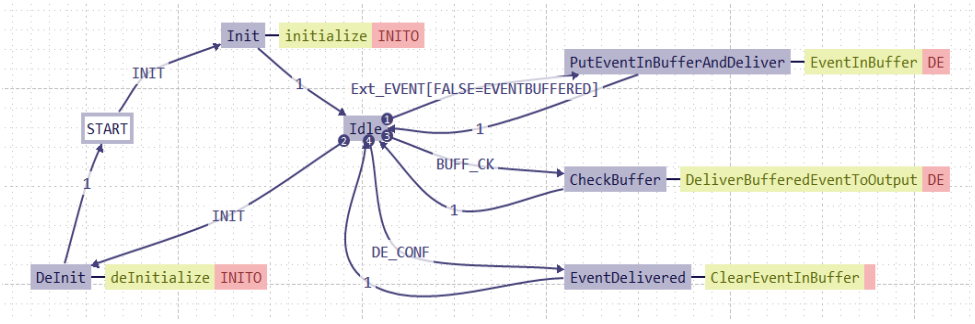


FB **BasicBufferG** – tipo **BasicBufferG_v1,** buffer per dato **INT**

BasicBufferS

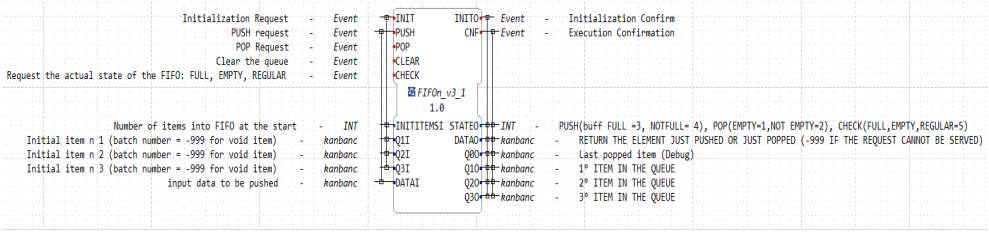


FB **BasicBufferS** – tipo **BasicBufferS_v1**, buffer per dato **STRING**

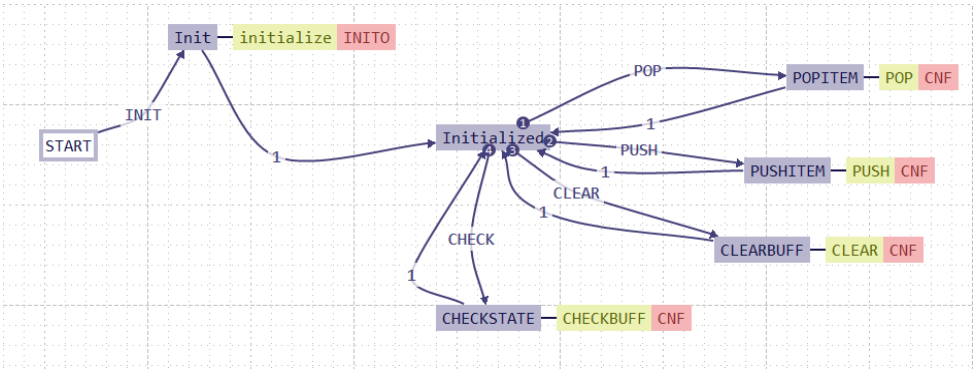


FB **BasicBufferG/K/S_v1** – ECC

STORE_FIFO_Db e KANBAN_STORE_FIFO



FBs **STORE_FIFO_Db, KANBAN_STORE_FIFO** – FB di tipo **FIFO_v3_1**

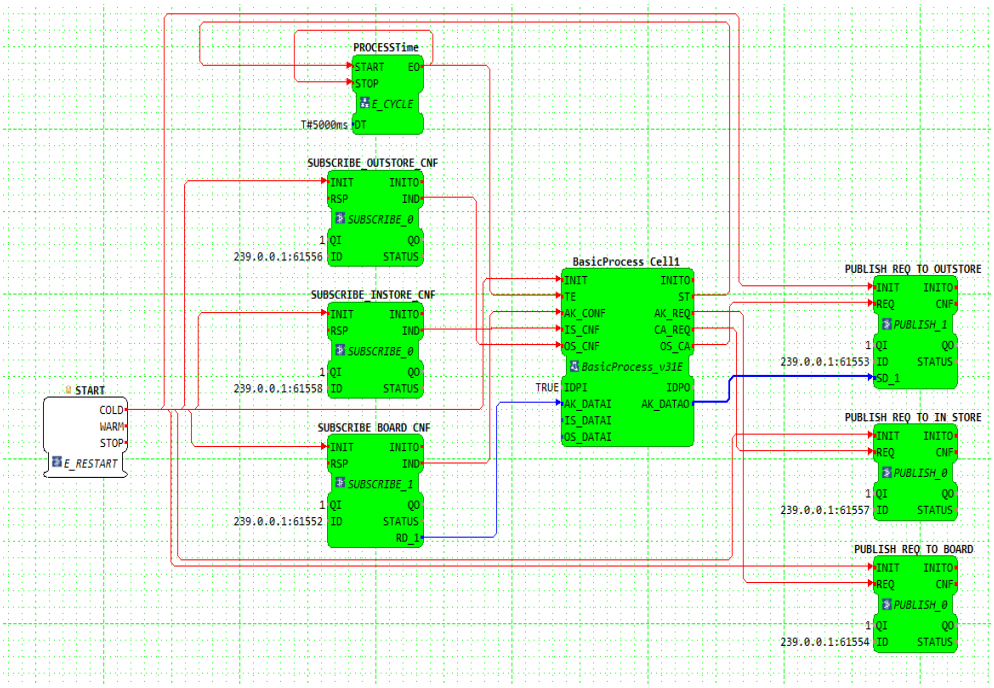


FB **FIFO_v_3_1** – Diagramma degli stati (ECC)

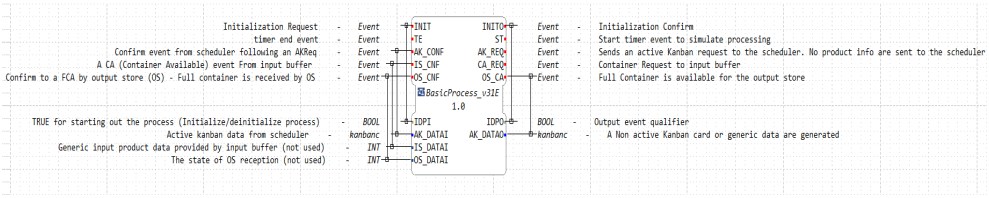
Applicazione: ***BASIC_PROCESS_CELL1***

Dispositivo: **PC4_PROCESS_CELL1**
Tipo dispositivo: **FORTE_PC**
Risorsa: **EMB_PROCESS_RES**

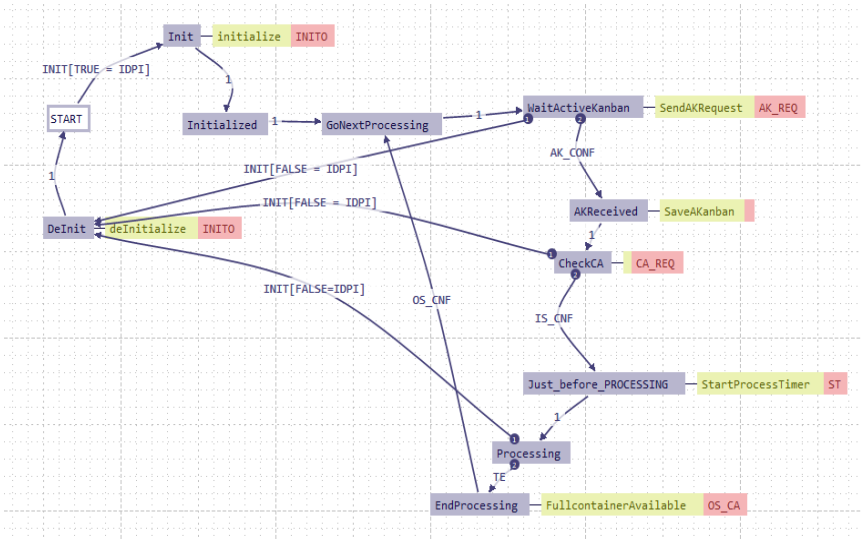
Questa applicazione simula le funzionalità del generico processo produttivo **BASIC_PROCESS_CELL1** di legenda L1. Il blocco funzionale B*asicProcess_cell1* invia le richieste di inserimento prodotto finito al magazzino di uscita (**PC5_OUT_STORE_CELL**), richiede le e-kanban di produzione al collector/board del magazzino di uscita e richiede il prodotto necessario alla lavorazione al supermarket B a monte. Il tempo di lavorazione è simulato dal blocco funzionale PROCESSTime.



BasicProcess_Cell1



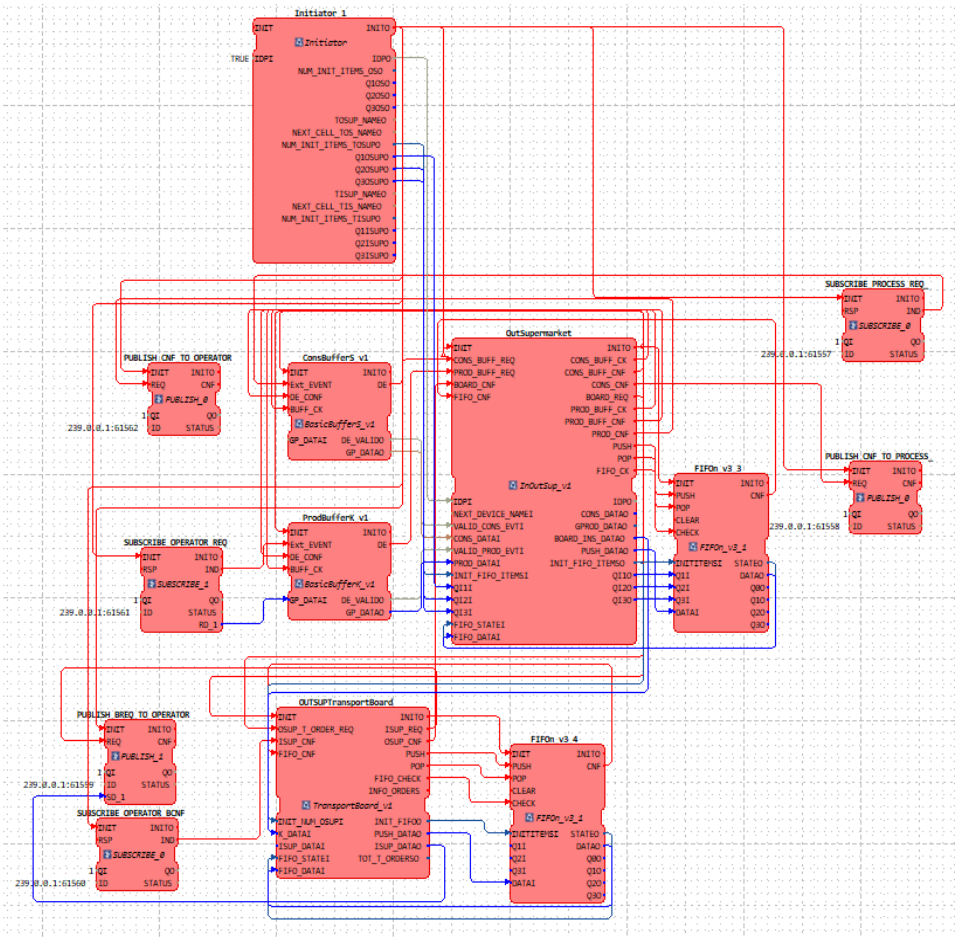
FB **BasicProcess_Cell1** – Interfaccia



FB **BasicProcess_Cell1** – Diagramma degli stati (ECC)

*Applicazione: **OUT_SUP** (Supermarket B)*

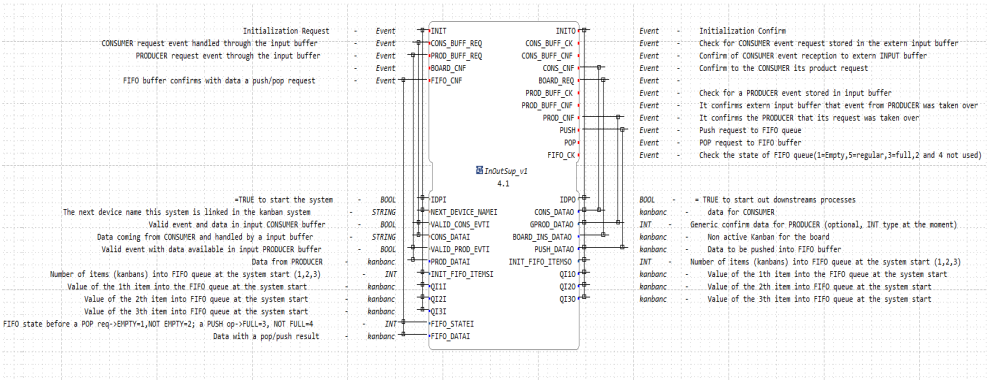
Dispositivo **PC3_OUT_STORE_SUPERMARKET**
Tipo: **FORTE_PC**
Risorsa: **EMB_OUT_SUP_RES**



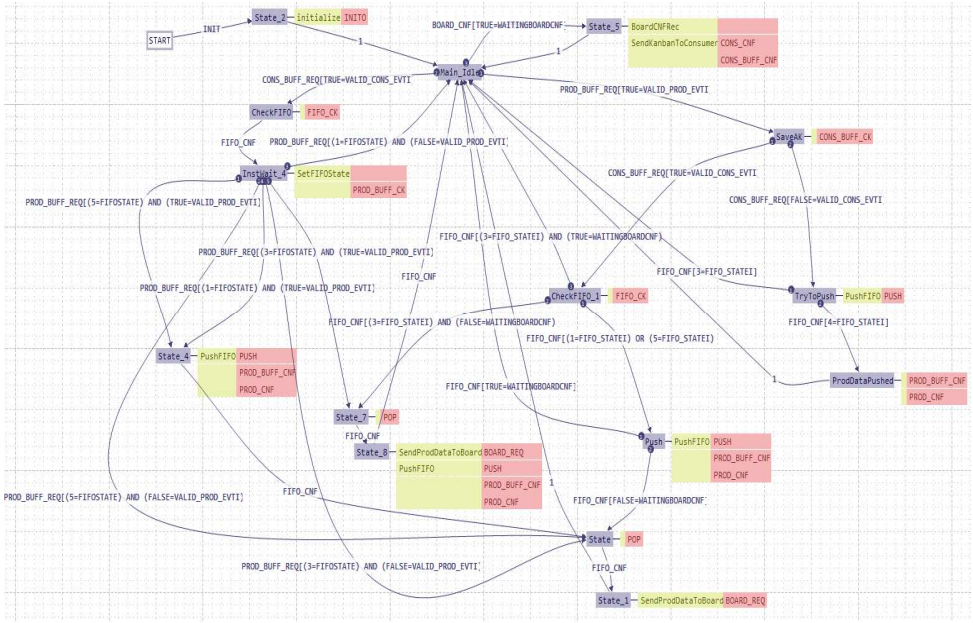
Applicazione dispositivo **PC3_OUT_STORE_SUPERMARKET**,
risorsa **EMB_OUT_SUP_RES**

È l'applicazione che mappa OUT_SUP, software del supermarket B: consta del blocco funzionale OutSupermarket che gestisce (sempre in modalità "buffered" lato produttore e consumatore per eventi asincroni e concorrenti) le richieste di prodotto del processo produttivo e le richieste di inserimento prodotto emesse dal robot di trasporto. Gestisce la coda FIFO che immagazzina le informazioni e-kanban dei prodotti presenti nel supermarket (FB FIFOnv3_3) e l'inserimento degli ordini di trasporto nel collector OutSupTransportBoard. Il blocco funzionale OutSupTransportBoard riceve le richieste di inserimento degli ordini di trasporto da OutSupermarket e le immagazzina nel buffer FIFO FIFOnv3_4. OutSupTransportBoard gestisce l'invio di questi ordini all'operatore di trasporto (il robot) e lavora in modo bufferizzato lato producer.

OutSupermarket

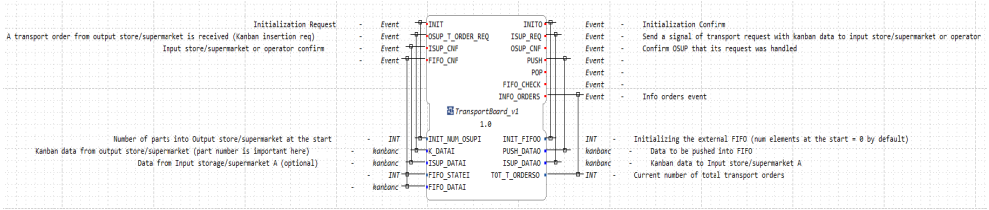


FB **OutSupermarket** – Interfaccia di tipo **InOutSup_v1**

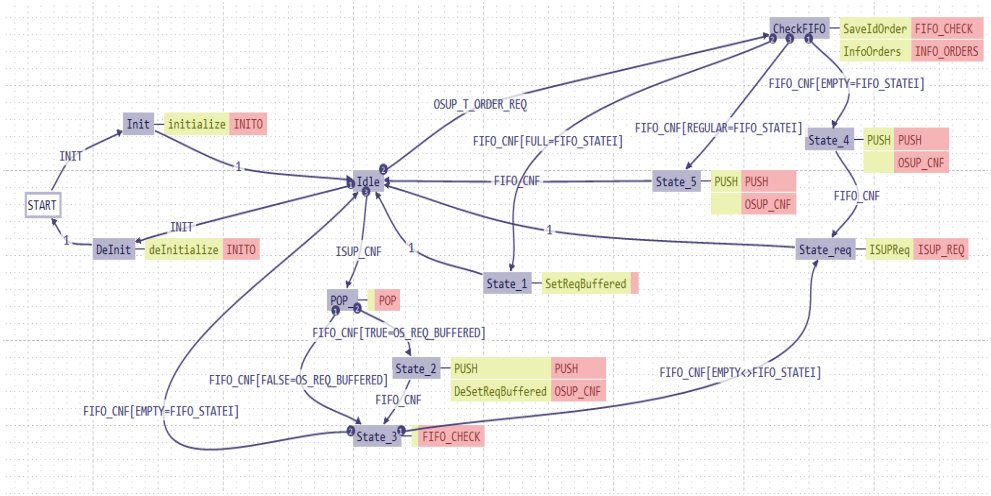


FB **OutSupermarket** – Diagramma degli stati (ECC)

OUTSUPTransportBoard



FB **OUTSUPTransportBoard** – interfaccia di tipo **TransportBoard_v1**

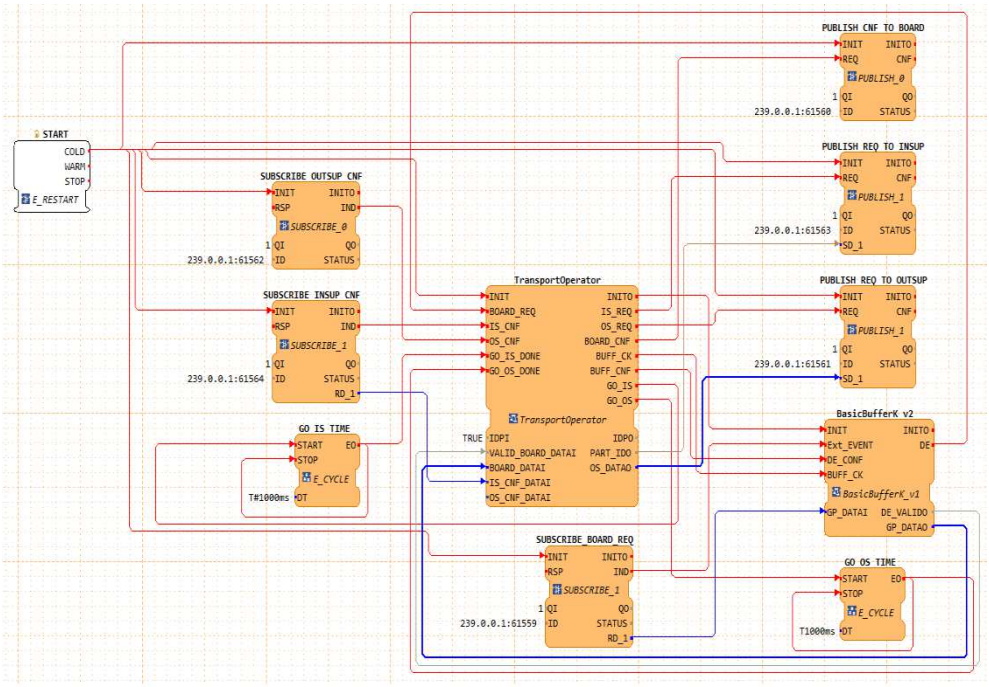


OUTSUPTransportBoard, tipo **TransportBoard_v1 -** ECC

Applicazione: ***TRANSPORT_OPERATOR***

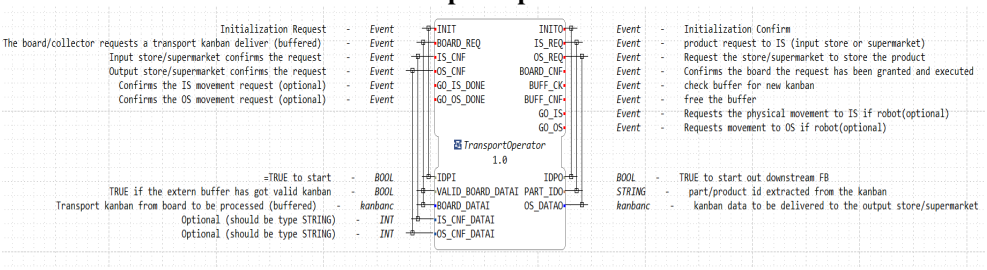
Dispositivo: **PC2_TRANSPORT_OPERATOR**
Tipo: **FORTE_PC**
Risorsa: **EMB_OPERATOR_RES**

Applicazione che mappa **TRANSPORT_OPERATOR**, il robot di trasporto materiale tra supermarket A e B. Il robot riceve le richieste/ordini e-kanban di trasporto (*ingresso buffered*) dal collector/board del supermarket B, simula il tempo di spostamento verso il supermarket A (blocco funzionale *GO_IS_TIME*), richiede il prodotto al supermarket A, simula la durata dello spostamento verso il supermarket B (FB *GO_OS_TIME*) e richiede l'inserimento del prodotto nel supermarket B. Il tempo di trasporto totale da A a B in questo esempio è di due secondi.

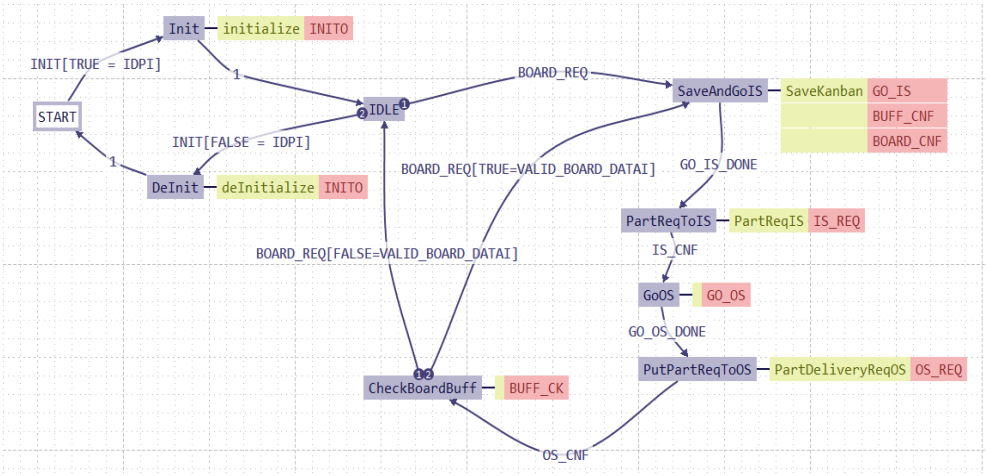


Applicazione dispositivo **PC2_TRANSPORT_OPERATOR,** risorsa **EMB_OPERATOR_RES**

TransportOperator



FB **TransportOperator** – Interfaccia di tipo **TransportOperator**

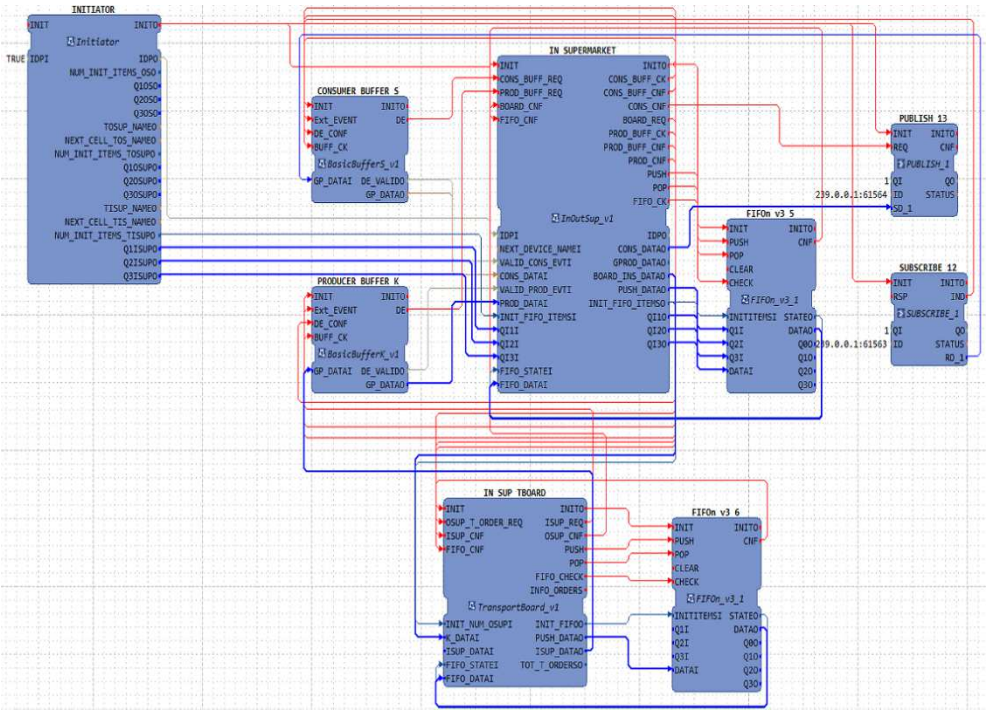


TransportOperator, tipo **TransportOperator -** ECC

*Applicazione: **IN_SUP** (Supermarket A)*

Dispositivo **PC1_INSTORE_SUPERMARKET**
Tipo**: FORTE_PC**
Risorsa**: EMB_IN_SUP_RES**

Applicazione che mappa IN_SUP, software del supermarket A: consta di un blocco funzionale IN_SUPERMARKET che gestisce la coda FIFO FIFOv_3_5 che immagazzina i dati dei prodotti contenuti nel supermarket. IN_SUPERMARKET gestisce le richieste di prodotto effettuate dal robot trasportatore (TRANSPORT_OPERATOR) attraverso il buffer CONSUMER_BUFFER_S e invia al collector/board IN_SUP_TBOARD le e-kanban di trasporto da utilizzare in un loop di trasporto a monte. In questo caso specifico non esiste un altro loop di trasporto a monte del dispositivo, in particolare, il collector emette le sue e-kanban di trasporto in ingresso a IN_SUPERMARKET attraverso il buffer PRODUCER_BUFFER_K: l'ordine di trasporto è associato dal modulo IN_SUPERMARKET ad un nuovo ordine di inserimento prodotto nel supermarket, ciò permette la simulazione di risorse infinite nel supermarket B, Figura 3.2.4) e la continuità della simulazione in assenza di elementi a monte.



Applicazione dispositivo **PC1_ INSTORE_SUPERMARKET,** risorsa **EMB_IN_SUP_RES**

Scenari di simulazione

È stato simulato uno scenario in cui il modulo *Int_Client* stressa la catena di produzione inviando al magazzino 40 ordini con tasso di emissione pari ad un ordine ogni 4 secondi. Il processo di produzione della cella 1 lavora 1 pezzo ogni 5 secondi e il tempo di trasporto totale del prodotto da parte dell'operatore robot (per il pick and place dal supermarket A a B) è di 2 secondi. Il supermarket A lavora a risorse infinite (prodotto sempre disponibile per il consumatore a valle) ed il tempo di trasporto di un prodotto finito dalla cella produttiva al magazzino da parte dei robot di magazzino è nullo, così come il tempo necessario all'altro robot di magazzino nel rendere un prodotto del magazzino disponibile per la consegna al cliente. La dimensioni del buffer FIFO dei collectors, dei buffers prodotti dei supermarket e del magazzino di uscita sono impostate a 3 elementi (nella concezione tradizionale significa che stiamo utilizzando un numero di e-kanban pari a 3 sia per il loop di produzione che di trasporto).
Il buffer FIFO interno del componente *Int_Client*, che immagazzina gli ordini da inviare al magazzino, è impostato a tre elementi ed è permessa la perdita di ordini nel caso il tasso di emissione di Int_Client fosse più alto della capacità di servizio della catena produttiva, come nel caso in esame. Ad ogni ordine servito da parte del magazzino (prodotto ordinato disponibile al cliente per il ritiro), il magazzino rilascia a Int_Client una informativa che comprende il numero di lotto del prodotto reso disponibile (o *batch_num*). I lotti sono numerati a partire dal lotto numero 0. Alla ricezione di questa informativa del magazzino, *Int_Client* la integra con il relativo *id* del suo ordine (*order_id*) e forma la variabile strutturata di tipo *corderinfos*, che a sua volta rende disponibile in uscita al modulo per la visualizzazione o per l'utilizzo da parte di un processo a valle. Questa variabile strutturata contiene l'*id* dell'ordine effettuato da *Int_Client*, l'*id* della parte consegnata, la descrizione della parte, la provenienza (magazzino di uscita), la dimensione del lotto e il numero di lotto.

Di seguito è mostrato un trace su file eseguito dal componente CSV connesso a Int_Client relativo alla variabile d'ordine di tipo *corderinfos* appena descritta. Si nota dal *trace* che dei 40 ordini emessi sono stati persi gli ordini con *id* (*order_id*) 26,31,36 e i numeri dei lotti corrispondenti consegnati dal magazzino (colonna *batch_num*) al client sono ovviamente contigui. Era triviale aspettarsi la perdita di qualche ordine visti i tempi della lavorazione e il *rate* della sorgente di ordini.

```
(order_id:=0,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=0);
(order_id:=1,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=1);
(order_id:=2,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=2);
(order_id:=3,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=3);
(order_id:=4,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=4);
(order_id:=5,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=5);
(order_id:=6,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=6);
(order_id:=7,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=7);
(order_id:=8,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=8);
(order_id:=9,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=9);
(order_id:=10,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=10);
(order_id:=11,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=11);
(order_id:=12,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=12);
(order_id:=13,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=13);
(order_id:=14,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=14);
(order_id:=15,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=15);
(order_id:=16,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=16);
(order_id:=17,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=17);
(order_id:=18,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=18);
(order_id:=19,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=19);
(order_id:=20,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=20);
(order_id:=21,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=21);
(order_id:=22,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=22);
(order_id:=23,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=23);
(order_id:=24,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=24);
(order_id:=25,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=25);
(order_id:=27,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=26);
(order_id:=28,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=27);
(order_id:=29,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=28);
(order_id:=30,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=29);
(order_id:=32,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=30);
(order_id:=33,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=31);
(order_id:=34,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=32);
(order_id:=35,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=33);
(order_id:=37,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=34);
(order_id:=38,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=35);
(order_id:=39,part_id:='',part_descr:='',from:='Output Store cell 1',batch_size:=0,batch_num:=36);
```

Il *trace* che segue conferma i dati risultanti da quello precedente e mostra lo stato degli ordini aggiornato dal sistema *Int_Client* ad ogni nuovo evento emesso dal generatore di eventi interno (che emette un evento ogni quattro secondi). Il trace presenta righe con i campi seguenti:
Totale ordini inviati da *Int_Client* in magazzino; Totale ordini serviti dal magazzino; numero *Id* dell'ultimo ordine servito; Ultimo numero di lotto servito; Totale ordini persi; Numero *Id* dell'ultimo ordine perso;

1; 0; -999; -999; 0; -999;
1; 1; 0; 0; 0; -999;
2; 1; 0; 0; 0; -999;
2; 2; 1; 1; 0; -999;
3; 2; 1; 1; 0; -999;
3; 3; 2; 2; 0; -999;
4; 3; 2; 2; 0; -999;
4; 4; 3; 3; 0; -999;
5; 4; 3; 3; 0; -999;
5; 5; 4; 4; 0; -999;
6; 5; 4; 4; 0; -999;
6; 6; 5; 5; 0; -999;

7; 6; 5; 5; 0; -999;
7; 7; 6; 6; 0; -999;
8; 7; 6; 6; 0; -999;
8; 8; 7; 7; 0; -999;
9; 8; 7; 7; 0; -999;
9; 9; 8; 8; 0; -999;
10; 9; 8; 8; 0; -999;
10; 10; 9; 9; 0; -999;
11; 10; 9; 9; 0; -999;
11; 11; 10; 10; 0; -999;
12; 11; 10; 10; 0; -999;
12; 12; 11; 11; 0; -999;
13; 12; 11; 11; 0; -999;
13; 13; 12; 12; 0; -999;
14; 13; 12; 12; 0; -999;
14; 14; 13; 13; 0; -999;
15; 14; 13; 13; 0; -999;
15; 15; 14; 14; 0; -999;
16; 15; 14; 14; 0; -999;
17; 15; 14; 14; 0; -999;
17; 16; 15; 15; 0; -999;
18; 16; 15; 15; 0; -999;
18; 17; 16; 16; 0; -999;
19; 17; 16; 16; 0; -999;
19; 18; 17; 17; 0; -999;
20; 18; 17; 17; 0; -999;
20; 19; 18; 18; 0; -999;
21; 19; 18; 18; 0; -999;
22; 19; 18; 18; 0; -999;
22; 20; 19; 19; 0; -999;
23; 20; 19; 19; 0; -999;
23; 21; 20; 20; 0; -999;
24; 21; 20; 20; 0; -999;
24; 22; 21; 21; 0; -999;
25; 22; 21; 21; 0; -999;
25; 23; 22; 22; 0; -999;
26; 23; 22; 22; 0; -999;
27; 23; 22; 22; 0; -999;
27; 23; 22; 22; 1; 26;
27; 24; 23; 23; 1; 26;
28; 24; 23; 23; 1; 26;
28; 25; 24; 24; 1; 26;
29; 25; 24; 24; 1; 26;
29; 26; 25; 25; 1; 26;
30; 26; 25; 25; 1; 26;
30; 27; 27; 26; 1; 26;

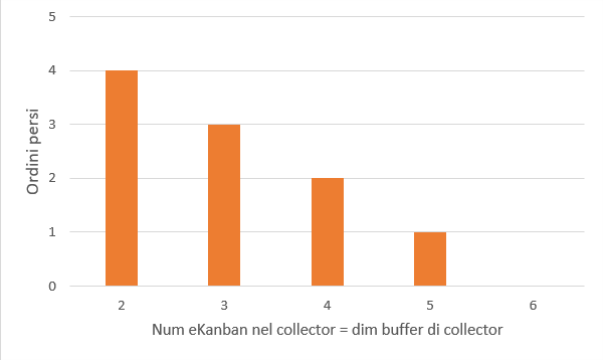
31; 27; 27; 26; 1; 26;
32; 27; 27; 26; 1; 26;
32; 27; 27; 26; 2; 31;
32; 28; 28; 27; 2; 31;
33; 28; 28; 27; 2; 31;
33; 29; 29; 28; 2; 31;
34; 29; 29; 28; 2; 31;
34; 30; 30; 29; 2; 31;
35; 30; 30; 29; 2; 31;
35; 31; 32; 30; 2; 31;
36; 31; 32; 30; 2; 31;
37; 31; 32; 30; 2; 31;
37; 31; 32; 30; 3; 36;
37; 32; 33; 31; 3; 36;
38; 32; 33; 31; 3; 36;
38; 33; 34; 32; 3; 36;
39; 33; 34; 32; 3; 36;
39; 34; 35; 33; 3; 36;
40; 34; 35; 33; 3; 36;
40; 35; 37; 34; 3; 36;
40; 36; 38; 35; 3; 36;
40; 37; 39; 36; 3; 36;

I valori della quinta e sesta colonna del trace confermano il totale degli ordini persi (3) e i relativi *id* (26,31,36).

Inoltre, si ricorda che si è ipotizzato il supermarket A a risorse infinite quindi questo componente ha sempre un prodotto disponibile per il consumatore a valle, cioè per il supermarket B. Il robot tra supermarket A e B impiega due secondi per il trasporto, cioè un tempo inferiore al tempo di lavorazione, quindi il processo produttivo trova sempre prodotto disponibile nel supermarket B. L'unico collo di bottiglia è il processo di lavorazione, visto che i robot del magazzino di uscita eseguono le operazioni in tempo nullo e il tempo del ciclo di lavorazione (5 secs) è superiore al tempo tra due ordini consecutivi (4 secs).
Con un tasso di emissione ordini inferiore, ad esempio un ordine ogni 6 secs (intervallo di arrivo ordini maggiore del tempo di lavorazione) il sistema non può perdere ordini, nemmeno con una dimensione minima del buffer di collector del magazzino, ad esempio pari a 1, poiché si produce solo quando un prodotto è consumato. In queste condizioni, e in una situazione di domanda stabile, converrebbe proprio lavorare con un buffer pari a 1, cioè la minima quantità di materiale circolante in magazzino. Il problema è che la domanda può non essere stabile quindi occorre cautelarsi mantenendo opportune dimensioni dei buffer dei collectors. Maggiore è la dimensione di questi buffer (ricordiamo essere pari al numero di e-kanban circolanti nel loop) maggiore sarà la capacità di servizio del sistema in caso di un aumento del tasso degli ordini emessi, per contro, maggiore sarà la quantità di materiale circolante in magazzino. L'ottimizzazione delle dimensioni dei buffer dei collectors rispetto al profilo della domanda e capacità di servizio, (quindi l'ottimizzazione della quantità di materiale

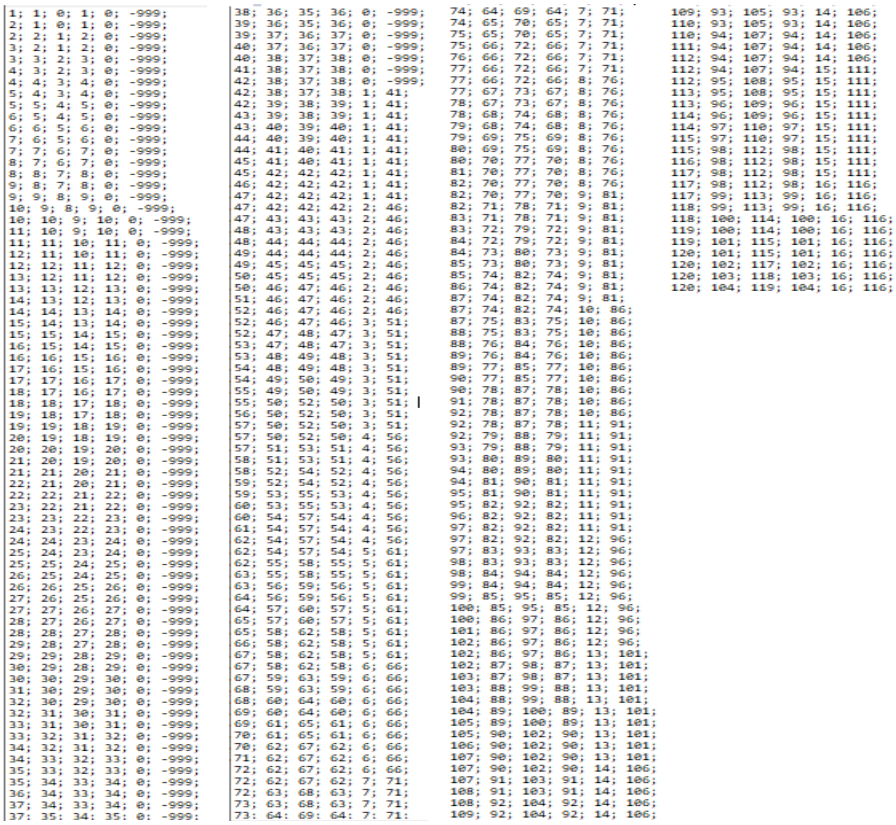
circolante in magazzino, che in generale rappresenta un costo) non è oggetto di questo lavoro ma rappresenta un ulteriore step di studio nell'ambito dei sistemi produttivi. La modularità del sistema progettato, insieme alla caratteristica di configurazione run-time del codice potranno consentire l'adattamento dei parametri dei buffer in tempo reale da parte di un supervisore che li imposterà in base al profilo di domanda del cliente e ai requisiti di capacità di servizio e costi degli inventari.

Per testare il comportamento dell'applicazione sono stati simulati altri cinque scenari per i quali il profilo di emissione ordini da parte della sorgente Int_Client rimane invariato (un ordine ogni 4 secondi contro un ciclo di lavorazione di 5 secondi) ma aumentando per ogni scenario la dimensione del buffer del collector di magazzino da 2 a 6. Si trova il risultato in figura:



Aumentando la dimensione del buffer di collector del magazzino, gli ordini persi decrescono progressivamente nel tempo di simulazione considerato (N=40 ordini, un ordine ogni 4 secondi, tempo di simulazione t_s = 160 secondi) fino a nessuna perdita con una dimensione pari a 6.
Dilatando il tempo di simulazione si riscontra che con t_s >160 secondi, quindi con N>40, anche con dimensione del buffer di collector del magazzino pari a sei, comincerà la perdita degli ordini, precisamente a partire dal quarantunesimo.
Questa situazione è mostrata nel *trace* che segue con N = 120 ordini > 40, t_s =480 secondi e indica che il primo ordine perso è il quarantunesimo (seconda macro-colonna delle quattro, ottava riga, quinta e sesta colonna) dopo 42 ordini emessi (164 secondi):

Totale ordini inviati da *Int_Client* in magazzino; Totale ordini serviti dal magazzino; numero *Id* dell'ultimo ordine servito; Ultimo numero di lotto servito; Totale ordini persi; Numero *Id* dell'ultimo ordine perso;



Il *trace* che segue mostra invece il contenuto del buffer FIFO a tre elementi di *Int_Client* contenente gli ordini generati che entrano nel buffer da destra e vengono estratti da sinistra al momento dell'invio in magazzino. Il *trace* conferma la perdita dell'ordine con *order_id* = 41 che è in arrivo quando lo stato del buffer si satura per la prima volta (terza macro-colonna) e contiene gli ordini 38;39;40 (-999 = elemento del buffer vuoto):

```
                14; -999; -999;  29; -999; -999;  44; 45; -999;  62; 63; 64;  79; 80; 82;  98; 99; 100;  117; 118; -999;
                -999; -999; -999;  -999; -999; -999;  44; 45; 47;  63; 64; -999;  80; 82; -999;  99; 100; -999;  117; 118; -999;
                -999; -999; -999;  -999; -999; -999;  45; 47; -999;  63; 64; -999;  80; 82; -999;  99; 100; -999;  117; 118; -999;
-999; -999; -999;  -999; -999; -999;  -999; -999; -999;  45; 47; -999;  63; 64; -999;  80; 82; -999;  99; 100; -999;  117; 118; 119;
0; -999; -999;  15; -999; -999;  30; -999; -999;  45; 47; -999;  63; 64; 65;  80; 82; 83;  99; 100; 102;  118; 119; -999;
-999; -999; -999;  -999; -999; -999;  30; -999; -999;  45; 47; 48;  63; 64; 65;  82; 83; -999;  100; 102; -999;  118; 119; -999;
-999; -999; -999;  -999; -999; -999;  30; 31; -999;  47; 48; -999;  64; 65; -999;  82; 83; -999;  100; 102; -999;  119; -999; -999;
-999; -999; -999;  -999; -999; -999;  31; -999; -999;  47; 48; -999;  64; 65; -999;  82; 83; -999;  100; 102; -999;  119; -999; -999;
1; -999; -999;  16; -999; -999;  31; -999; -999;  47; 48; -999;  64; 65; -999;  82; 83; 84;  100; 102; 103;  -999; -999; -999;
-999; -999; -999;  -999; -999; -999;  31; -999; -999;  47; 48; 49;  64; 65; 67;  83; 84; -999;  102; 103; -999;  -999; -999; -999;
-999; -999; -999;  -999; -999; -999;  31; 32; -999;  48; 49; -999;  65; 67; -999;  83; 84; -999;  102; 103; -999;
2; -999; -999;  17; -999; -999;  32; -999; -999;  48; 49; -999;  65; 67; -999;  83; 84; 85;  102; 103; 104;
-999; -999; -999;  -999; -999; -999;  32; -999; -999;  48; 49; 50;  65; 67; 68;  83; 84; 85;  103; 104; -999;
-999; -999; -999;  -999; -999; -999;  32; 33; -999;  48; 49; 50;  67; 68; -999;  84; 85; -999;  103; 104; -999;
-999; -999; -999;  -999; -999; -999;  33; -999; -999;  49; 50; -999;  67; 68; -999;  84; 85; -999;  103; 104; -999;
3; -999; -999;  18; -999; -999;  33; -999; -999;  49; 50; -999;  67; 68; -999;  84; 85; -999;  103; 104; 105;
-999; -999; -999;  -999; -999; -999;  33; -999; -999;  49; 50; -999;  67; 68; 69;  84; 85; 87;  103; 104; 105;
-999; -999; -999;  -999; -999; -999;  33; 34; -999;  49; 50; 52;  68; 69; -999;  85; 87; -999;  104; 105; -999;
-999; -999; -999;  -999; -999; -999;  34; -999; -999;  50; 52; -999;  68; 69; -999;  85; 87; -999;  104; 105; -999;
4; -999; -999;  19; -999; -999;  34; -999; -999;  50; 52; -999;  68; 69; -999;  85; 87; -999;  104; 105; -999;
-999; -999; -999;  -999; -999; -999;  34; -999; -999;  50; 52; -999;  68; 69; 70;  85; 87; 88;  104; 105; 107;
-999; -999; -999;  -999; -999; -999;  34; 35; -999;  50; 52; 53;  68; 69; 70;  87; 88; -999;  105; 107; -999;
-999; -999; -999;  -999; -999; -999;  34; 35; -999;  52; 53; -999;  69; 70; -999;  87; 88; -999;  105; 107; -999;
5; -999; -999;  20; -999; -999;  34; 35; 36;  52; 53; -999;  69; 70; -999;  87; 88; -999;  105; 107; -999;
-999; -999; -999;  -999; -999; -999;  35; 36; -999;  52; 53; -999;  69; 70; -999;  87; 88; 89;  105; 107; 108;
-999; -999; -999;  -999; -999; -999;  35; 36; -999;  52; 53; 54;  69; 70; 72;  88; 89; -999;  107; 108; -999;
-999; -999; -999;  -999; -999; -999;  35; 36; -999;  53; 54; -999;  70; 72; -999;  88; 89; -999;  107; 108; -999;
6; -999; -999;  21; -999; -999;  35; 36; 37;  53; 54; -999;  70; 72; -999;  88; 89; -999;  107; 108; -999;
-999; -999; -999;  -999; -999; -999;  36; 37; -999;  53; 54; -999;  70; 72; -999;  88; 89; 90;  107; 108; 109;
-999; -999; -999;  -999; -999; -999;  36; 37; -999;  53; 54; 55;  70; 72; 73;  88; 89; 90;  108; 109; -999;
-999; -999; -999;  -999; -999; -999;  36; 37; -999;  53; 54; 55;  72; 73; -999;  89; 90; -999;  108; 109; -999;
7; -999; -999;  22; -999; -999;  36; 37; 38;  54; 55; -999;  72; 73; -999;  89; 90; -999;  108; 109; -999;
-999; -999; -999;  -999; -999; -999;  37; 38; -999;  54; 55; -999;  72; 73; -999;  89; 90; -999;  108; 109; 110;
-999; -999; -999;  -999; -999; -999;  37; 38; -999;  54; 55; -999;  72; 73; 74;  89; 90; 92;  108; 109; 110;
-999; -999; -999;  -999; -999; -999;  37; 38; -999;  54; 55; 57;  73; 74; -999;  90; 92; -999;  109; 110; -999;
8; -999; -999;  23; -999; -999;  37; 38; 39;  55; 57; -999;  73; 74; -999;  90; 92; -999;  109; 110; -999;
-999; -999; -999;  -999; -999; -999;  38; 39; -999;  55; 57; -999;  73; 74; -999;  90; 92; -999;  109; 110; -999;
-999; -999; -999;  -999; -999; -999;  38; 39; -999;  55; 57; -999;  73; 74; 75;  90; 92; 93;  109; 110; 112;
-999; -999; -999;  -999; -999; -999;  38; 39; -999;  55; 57; 58;  73; 74; 75;  92; 93; -999;  110; 112; -999;
9; -999; -999;  24; -999; -999;  38; 39; 40;  57; 58; -999;  74; 75; -999;  92; 93; -999;  110; 112; -999;
-999; -999; -999;  -999; -999; -999;  38; 39; 40;  57; 58; -999;  74; 75; -999;  92; 93; -999;  110; 112; -999;
-999; -999; -999;  -999; -999; -999;  39; 40; -999;  57; 58; -999;  74; 75; -999;  92; 93; 94;  110; 112; 113;
-999; -999; -999;  -999; -999; -999;  39; 40; -999;  57; 58; 59;  74; 75; 77;  93; 94; -999;  112; 113; -999;
10; -999; -999;  25; -999; -999;  39; 40; -999;  58; 59; -999;  75; 77; -999;  93; 94; -999;  112; 113; -999;
-999; -999; -999;  -999; -999; -999;  39; 40; 42;  58; 59; -999;  75; 77; -999;  93; 94; -999;  112; 113; -999;
-999; -999; -999;  -999; -999; -999;  40; 42; -999;  58; 59; -999;  75; 77; -999;  93; 94; 95;  112; 113; 114;
-999; -999; -999;  -999; -999; -999;  40; 42; -999;  58; 59; 60;  75; 77; 78;  93; 94; 95;  113; 114; -999;
11; -999; -999;  26; -999; -999;  40; 42; -999;  58; 59; 60;  77; 78; -999;  94; 95; -999;  113; 114; -999;
-999; -999; -999;  -999; -999; -999;  40; 42; 43;  59; 60; -999;  77; 78; -999;  94; 95; -999;  113; 114; -999;
-999; -999; -999;  -999; -999; -999;  42; 43; -999;  59; 60; -999;  77; 78; -999;  94; 95; -999;  113; 114; 115;
-999; -999; -999;  -999; -999; -999;  42; 43; -999;  59; 60; -999;  77; 78; 79;  94; 95; 97;  113; 114; 115;
12; -999; -999;  27; -999; -999;  42; 43; -999;  59; 60; 62;  78; 79; -999;  95; 97; -999;  114; 115; -999;
-999; -999; -999;  -999; -999; -999;  42; 43; 44;  60; 62; -999;  78; 79; -999;  95; 97; -999;  114; 115; -999;
-999; -999; -999;  -999; -999; -999;  43; 44; -999;  60; 62; -999;  78; 79; -999;  95; 97; -999;  114; 115; -999;
-999; -999; -999;  -999; -999; -999;  43; 44; -999;  60; 62; -999;  78; 79; 80;  95; 97; 98;  114; 115; 117;
13; -999; -999;  28; -999; -999;  43; 44; -999;  60; 62; 63;  78; 79; 80;  97; 98; -999;  115; 117; -999;
-999; -999; -999;  -999; -999; -999;  43; 44; 45;  62; 63; -999;  79; 80; -999;  97; 98; -999;  115; 117; -999;
-999; -999; -999;  -999; -999; -999;  43; 44; 45;  62; 63; -999;  79; 80; -999;  97; 98; -999;  115; 117; -999;
-999; -999; -999;  -999; -999; -999;  44; 45; -999;  62; 63; -999;  79; 80; -999;  97; 98; 99;  115; 117; 118;
14; -999; -999;  29; -999; -999;  44; 45; -999;  62; 63; 64;  79; 80; 82;  98; 99; -999;  117; 118; -999;
```

Lo scopo di questi scenari è stato quello di testare il comportamento dell'applicazione che simula una catena produttiva minima dello shop-floor, verificando la coerenza dei dati collezionati dal client; inoltre, lo scenario di produzione globale è stato circoscritto assumendo risorse infinite per i supermarket, al fine di mettere in luce una caratteristica critica dello strumento Kanban. In particolare, si verifica che qualora fossimo in presenza di incrementi della velocità di emissione degli ordini, discostandoci dalle condizioni di domanda stabile e in accordo con i tempi del processo di lavorazione, un adattamento dinamico delle dimensioni dei collectors (numero di e-Kanban utilizzate) deve essere auspicato in funzione dei requisiti di capacità di servizio desiderati. Ciò comporta anche che la capacità produttiva disponibile dovrebbe essere maggiore di quella richiesta in media per fronteggiare picchi di richieste.

Conclusioni

Un valore propedeutico per gli obiettivi di questa tesi è stato quello di capire in che contesto industriale ci stiamo muovendo. In un tipico shop-floor di produzioni manifatturiere di piccole e medie imprese c'è infatti la continua necessità di adottare soluzioni orientate all'efficientamento dei flussi di materiale e dei flussi informativi, laddove, in una buona percentuale, il *monitoring* delle attività di produzione e lo scambio informativo tra processi è ancora eseguito per via cartacea con tutte le problematiche del caso. Inoltre, ci si rende conto che siamo in una fase di transizione in industria, una fase in cui esperienze aziendali reali insegnano che molti sistemi non sono progettati per una evoluzione *smart* o modulare, servirebbero grossi sforzi di ingegnerizzazione per renderli smart o adattivi e che a volte sarebbe più conveniente partire da zero che renderli tali; quando si parla di sistemi I4.0 si parla infatti di sistemi eterogenei decentralizzati, distribuiti, con intelligenza distribuita, adattabili e riconfigurabili. La digitalizzazione pervasiva e l'utilizzo di componenti intelligenti che si inseriscono nel sistema in modalità plug-in, che cooperano e condividono informazioni e servizi orizzontalmente nello shop-floor e verticalmente fino ai sistemi ERP rappresenta il modo con il quale il paradigma I4.0 si propone di affrontare i requisiti di agilità, flessibilità e adattabilità del processo produttivo. Sebbene gli scenari, le modalità e le difficoltà di integrazione tra le nuove possibilità e gli strumenti produttivi *legacy* siano molteplici, la tendenza nella produzione manifatturiera è quella di considerare il cliente in posizione centrale (dalla produzione di massa alla customizzazione di massa) e sempre più aziende stanno migrando verso una filosofia di produzione *pull.*

In questa tesi, mettendo a sistema concetti di base tradizionali derivanti dalla filosofia Lean, ossia metodo di produzione pull orientato al cliente, controllo decentralizzato della produzione, obiettivi di quantità stabili degli inventari nei magazzini e tra processi produttivi (WIP), si è considerato lo strumento Kanban tradizionale come primo possibile approccio al problema del controllo e tracciamento dei flussi di produzione in uno shop-floor manifatturiero di piccole medie imprese in presenza o meno di sistemi ERP. Se l'organizzazione aziendale prevede un software ERP, generalmente è connesso ad un database centrale che contiene lo storico dei dati di produzione, l'inventario o scorte nei magazzini/supermarket, lo stato dei buffer tra processi produttivi (WIP) e quindi il flusso dei materiali. Queste informazioni possono essere analizzate dal software ERP per fare previsioni su cosa, quanto e quando produrre, definire e ordinare la quantità dei materiali necessari allo shop-floor nei magazzini e nei buffer tra processi per l'ottimizzazione della produzione secondo uno schema tipicamente push. La condivisione dei dati dallo shop-floor può essere realizzata in tempo reale in modo più o meno automatico con tecnologie che

permettono il tracciamento dei materiali dello shop-floor (RFID, codici a barre) e la connessione delle macchine, ma non di rado l'avanzamento della produzione giunge al sistema informativo attraverso operatori umani per via cartacea. Nel caso di reporting cartaceo dell'avanzamento della produzione e inventari, possono verificarsi ritardi (a volte di un'intera giornata o più) tra la situazione attuale e quella aggiornata nel sistema ERP, quindi può succedere che quando l'inventario WIP viene creato nel sistema ERP, è già stato consumato fisicamente in produzione, quindi, non sempre le previsioni si basano su dati accurati.
In questo scenario sembrerebbe che il concetto stesso di "previsioni" alla base dei software ERP si scontri con i benefici di un possibile sistema kanban applicato alla linea produttiva per il quale i rifornimenti di materiali e produzioni sono attivati dall'utilizzo effettivo delle parti piuttosto che da previsioni teoriche soggette a errori (il livello di WIP è sotto controllo perché non supererà mai il numero di carte kanban o e-kanban in circolazione e gli inventari sono controllati fisicamente con metodo pull lungo la linea, possibilmente dal cliente al fornitore, anziché tramite un sistema push ERP). Tuttavia, dotare una linea produttiva di capacità di controllo e-Kanban automatico non si scontra con l'utilizzo di sistemi ERP ma rappresenta una capacità aggiuntiva, in quanto gli inventari sono tenuti sotto stretto controllo da un metodo Lean che lavorerebbe in parallelo al sistema ERP senza la necessità di ricorrere a moduli Lean ERP aggiuntivi, basati su un mix di metodo push e pull. Possibili Integrazioni e sincronizzazioni tra i due mondi potranno essere oggetto di approfondimenti in sviluppi futuri.

In questa tesi è quindi fornito un modello di controllo e-Kanban basato sulla concezione tradizionale Kanban di tipo pull, implementabile dai dispositivi tipici di uno shop-floor manifatturiero, estensibile e orientato ai requisiti di I4.0. Il modello generale del controllo è infatti un modello decentralizzato, distribuito e con semantica di comunicazione ad eventi, compatibile con lo standard IEC61499. Magazzini e supermarket sono moduli che ospitano i propri prodotti in database locali, con possibilità di analisi locale di consumi/domanda realizzabile in futuro da un ipotetico agente software del dispositivo o un agente supervisore per possibili ottimizzazioni successive.

La definizione del modello proposto ha comportato le seguenti fasi: 1) Descrizione dello scenario produttivo, ossia identificazione dei processi e dei componenti considerati nella catena produttiva; 2) Produzione di ipotesi circa le funzionalità generali dei processi/componenti e descrizione delle funzionalità generali del software embedded associato; 3) Proposizione di una semantica di comunicazione generica ad eventi/dati per lo scambio dei messaggi tra i componenti; 4) Descrizione dettagliata delle funzionalità del metodo Kanban attraverso la descrizione della semantica di scambio dei messaggi tra i componenti lato cliente, nel loop di produzione e nel loop di trasporto.

In particolare, è stato descritto uno scenario generale in cui sono individuati i componenti fisici che partecipano nella produzione e le relative componenti software embedded: sono considerati un cliente generatore di ordini prodotto, un magazzino di uscita che riceve questi ordini, un processo produttivo che richiede ordini di produzione e richiede l'inserimento del prodotto finito in magazzino, due isole di prodotto o supermarket per il rifornimento di materiale al processo produttivo, e operatori (robot) per il pick-and-place di materiale dal

processo produttivo al magazzino, dal magazzino al cliente e un robot per il trasporto tra i supermarkets. Il magazzino e i supermarket sono dotati di un database interno di prodotti. Le componenti software dei dispositivi realizzano comunicazioni di ordini di produzione o di ordini di trasporto materiale (e-Kanban di produzione e trasporto), di richieste di inserimento/estrazione dei prodotti fisici nel/dal magazzino, nei/dai supermarket e nei/dai database interni. Per queste comunicazioni è stata definita una semantica minima orientata agli eventi/dati che stimola a pensare in termini di richieste di "servizi" tra i processi/dispositivi che svolgono il controllo di basso livello. I database sono embedded nei dispositivi ma la modularità del modello e la semantica generale di comunicazione utilizzata consentiranno implementazioni nelle quali questi database potranno essere localizzati, in modo trasparente, in un server esterno, un server dello shop-floor, in cloud o coincidere con un database centrale aziendale. Tra il magazzino di uscita e il processo produttivo è realizzato un loop Kanban di produzione, tra i due supermarket è realizzato un loop Kanban di trasporto materiale. È possibile l'espansione della catena con altri loop di trasporto e produzione. Nel loop di produzione è presente un dispositivo collector che immagazzina le e-Kanban o ordini di produzione ed è un componente del software di magazzino. Nel loop di trasporto è previsto un collector (componente del software di un supermercato) che immagazzina gli ordini di trasporto materiale; la dimensione del buffer collector del magazzino/supermarket rappresenta la quantità massima di prodotto circolante nel magazzino/supermarkets.
Sono state realizzate due tipologie di collectors, una versione per i supermarkets che segue la logica tradizionale (è il collector ad inviare l'ordine di trasporto e-Kanban all'operatore robot preposto) e una versione per il magazzino, in quest'ultima si è previsto che fosse il processo produttivo a richiedere l'ordine e-kanban al collector, Ad ogni modo, può essere sempre utilizzata la tipologia del collector di trasporto anche in produzione e in modo trasparente, a meno di minime modifiche alla macchina a stati del processo produttivo.

Il modello proposto è stato poi mappato in una applicazione distribuita in standard IEC 61499 utilizzando il software di progettazione OpenSource 4Diac accoppiato alla libreria di run-time FORTE. Rispetto a IEC 61131, lo standard IEC 61499 definisce un linguaggio per la modellazione più che un linguaggio di programmazione, il suo focus è rivolto a creare codice modulare, comprensibile e riconfigurabile *on-the-fly*. L'ambiente scelto per la programmazione dell'applicazione che implementa il controllo, è il tool 4DIAC che può connettersi ai dispositivi, programmarli e distribuire le applicazioni, cioè programmare un set di dispositivi (differentemente da IEC 61131), monitorarli, riconfigurarli e lo può fare in un ambiente eterogeneo. Inoltre, è orientato alla comunicazione tra dispositivi distribuiti (fornisce implementazioni di MQTT, OPC-UA, interazioni cloud e orientate ai servizi) quindi si presta bene, in prospettiva, per la parte di esecuzione real-time degli agenti industriali grazie ad una forte connettività IoT e cloud offerta dal codice open-source già fornito in 4DIAC-FORTE.

Per ogni componente del sistema produttivo è stata descritta la relativa applicazione IEC 61499, i blocchi funzionali utilizzati, le relative interfacce e macchine a stati (ECC) che descrivono il comportamento dei blocchi funzionali.
La validazione dell'applicazione è stata eseguita con la verifica della coerenza dei dati collezionati dal client in più scenari di simulazione e hanno messo in evidenza un aspetto critico nell'utilizzo di un controllo Kanban tradizionale (tipico di tutti i sistemi con code)

cioè la perdita di ordini nel caso di variazioni repentine della domanda cliente che superino per un certo Δt la capacità di servizio della produzione. In questi casi sarà necessario adattare (aumentare) opportunamente le dimensioni dei buffer di collector per garantire uno specifico requisito di capacità di servizio. Sfruttando le possibilità di riconfigurazione offerte dallo standard IEC61499, l'adattamento dei parametri è possibile a run-time. Le analisi circa il dimensionamento ottimo dei buffer di collector dati il profilo statistico della domanda, i requisiti di quantità media di scorte in magazzino per una data capacità di servizio e tempi di consegna, non sono oggetto di questo lavoro ma sono temi che devono essere affrontati in successivi step di analisi per un corretto dimensionamento del sistema produttivo.

In conclusione, è stato proposto un modello di controllo decentralizzato e distribuito Kanban basato su eventi e fornita un'implementazione di base in standard IEC 61499 da applicarsi ai dispositivi tipici di uno shop-floor che può essere esteso e modificato, anche per situazioni multiprodotto con minime modifiche del codice già esistente e multi-processo. L'attenzione è stata riposta nella costruzione dei blocchi funzionali che gestiscono le code di magazzino, supermarkets e collectors (in ambiente asincrono con possibilità di eventi concorrenti che potrebbero non essere gestiti dai blocchi funzionali) in modo di disporre di codice riutilizzabile in questo o altri progetti.

Appendice

Modelli semantici di esecuzione dei blocchi funzionali

Nel capitolo 1 paragrafo 1.9 sono stati descritti gli aspetti del modello di esecuzione event-driven dei blocchi funzionali definiti dal modello standard IEC 61499. Ci sono però altri aspetti, detti semantici, che lo standard lascia aperti e responsabiltà dei differenti tools per la progettazione che generano il codice per i dispositivi reali di controllo. Ogni implementazione reale del modello IEC 61499 può infatti adottare una specifica semantica implementativa che rappresenta l'insieme delle regole adottate per l'esecuzione degli eventi e delle connessioni tra i blocchi funzionali e ciò spiega perché le stesse applicazioni progettate con tools differenti possono produrre risultati diversi anche in termini di consumo di memoria e prestazioni. Nello specifico, alcuni strumenti di progettazione sfruttano particolari ambienti di run-time (simili ad uno scheduler di un sistema operativo) che in generale gestiscono o schedulano l'esecuzione degli eventi, dei blocchi funzione e il trasferimento dei dati tra i blocchi, tramite un meccanismo di accodamento che serve gli eventi multipli uno alla volta, pertanto, il comportamento dell'intero sistema dipende dalla coda degli eventi e dalla sua gestione. In particolare, tra i tools di progettazione più conosciuti troviamo il software FBDK *(Function Block Development Kit)* [34], che genera codice Java per l'ambiente run time FBRT, l'IDE OpenSource di progettazione 4-Diac [35] che genera codice per il run-time FORTE [36], e il run-time FUBER [37]. Gli ambienti run-time citati adottano una semantica event-driven pura.
In particolare, FBRT è basato su Java, quindi ha il requisito/svantaggio di dover caricare una macchina virtuale Java nel dispositivo target, il ché può implicare un consumo di risorse troppo oneroso, ad esempio per sistemi target embedded. Il codice generato da FBRT è semplice ma presenta un altro svantaggio riguardo la gestione della propagazione degli eventi che avviene utilizzando un modello *depth-model*, meno efficiente del *breadth-first* model adottato ad esempio dal run-time FORTE. Quest'ultimo è basato su librerie Opensource scritte in C++ e presenta lo svantaggio di generare codice target generalmente più pesante a causa del multithreading ma con focus rivolto all'efficienza al fine di soddisfare i requisiti più stringenti dei sistemi embedded. Infine, FUBER è un progetto con vantaggi e svantaggi molto simili a FORTE, adotta FIFO locali per ciascun blocco funzionale, a differenza della FIFO globale adottata in FORTE.
Il tool ISaGRAF [38] adotta invece un ambiente run-time con semantica sincrona [39]. Ciò significa che i blocchi funzionali sono eseguiti in modo sequenziale all'interno del tempo ciclo di un programma PLC che gira a parte ed in parallelo con il programma principale. I blocchi funzionali sono eseguiti nel caso siano campionati eventi in ingresso ad ogni ciclo e il comportamento del sistema può variare in base all'ordine con il quale sono stati schedulati i blocchi funzionali all'interno del ciclo stesso. Una comparazione delle prestazioni tra le implementazioni ISaGRAF e FBDK è reperibile in [40].

Approcci alternativi ai sistemi run-time, molto più efficienti in termini di consumo di memoria e prestazioni e particolarmente adatti a sistemi embedded real-time con poche risorse, incorporano la gestione dell'esecuzione dei blocchi funzionali direttamente nel

codice dell'applicazione e utilizzano semantiche sincrone che introducono determinismo. L'utilizzo di semantiche sincrone per l'implementazione del modello di esecuzione dei blocchi funzionali non deve confondere, "esternamente" il modello è sempre event-driven, ma "localmente" alla specifica risorsa la semantica delle attivazioni delle FB di un framework applicativo può essere sincrona, come proposto in [41] (in un sistema di controllo complesso e distribuito in rete avrebbe poco senso utilizzare una implementazione sincrona globale tra tutti i dispositivi).

Bibliografia

[1] Bonci A., Longhi S., Pirani M., Lorenzoni E., Rizzello G., Naso D., Seelecke S., (2018), Simulation Analysis and Performance Evaluation of a Vibratory Feeder Actuated by Dielectric Elastomers, 2018, 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), Oulu, Finland.

[2] Drath R., Horch A., (2014), Industrie 4.0: Hit or Hype?, IEEE Industrial Electronics Magazine, Vol 8, issue 2, p. 56-58.

[3] Ribeiro L, (2017), Cyber-physical production systems design challenge, IEEE International Symposium on Industrial Electronics (ISIE 2017), Edinburgh, Scotland.

[4] Ribeiro L. and Hochwallner M., (2018), On the design complexity of Cyberphysical production systems", Complexity vol 2018, article ID 4632195, 13 pages.

[5] Russel S. and Norvig P., (2003), Artificial Intelligence a modern approach, Prentice Hall, New Jersey, 2003.

[6] Wooldridge M., (2002), An Introduction to Multi-Agent Systems, John Wiley & Sons, 2002.

[7] Brooks R.A., (1990), Elephant don't play chess, Robotics and Autonomous Systems, Vol 6, pp. 3-15, 1990.

[8] Leitão P., Barbosa J., (2015), Building a Robotic Cyber-Physical Production Component, SOHOMA'15, Cambridge.

[9] International Electrotechnical Commission (2003), International Standard IEC 61131-3:Programmable Controllers – Part 3: Programming Languages. Geneva, 2nd edn.

[10] Lewis RW. Programming industrial control systems using IEC 1131-3, 2nd edition. Stevenage (UK): IET; 1998.

[11] International Electrotechnical Commission (2005), International Standard IEC 61499-1:Function blocks-Part 1: Architecture. Geneva, 1st edn

[12] Van Leeuwen E.H., Norrie D., (1997), Holons and holarchies [intelligent manufacturing systems]. Manufacturing Engineer,1997; 76(2): 86-88.

[13] Ribeiro L., (2015), The design, deployment, and assessment of industrial agent systems. In: Industrial Agents. Morgan Kaufmann; 2015. p. 45-63.

[14] IEC-61499-1. International and Electrotechnical Commission. IEC 61499-1, Function Blocks - Part 1: Architecture. Edition 2.0. Geneva: IEC; 2012.

[15] International Electrotechnical Commission (2005)
International Standard IEC 61499-2:Function blocks -Part 2: Software Tool Requirements. Geneva, 1st edn

[16] Zoitl A., Strasser T., (2017), Distributed control applications: guidelines, design patterns, and application examples with the IEC 61499. Boca Raton: CRC.

[17] Ferrarini L., Veber C., (2005), Lo standard IEC 61499 per sistemi distribuiti di automazione industriale. Automazione e strumentazione.

[18] Bonci A., Pirani M., Bianconi C., Longhi S., (2018). RMAS: Relational Multiagent System for CPS Prototyping and Programming. 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems andApplications (MESA). IEEE:2018. p. 1-6.

[19] Bonci A., Pirani M., Longhi S., (2018). A database-centric framework for the modeling, simulation, and control of cyber-physical systems in the factory of the future. Journal of Intelligent Systems 2018; 27(4):659-679.

[20] Bonci A., Pirani M., Dragoni A.F, Cucchiarelli A., Longhi S., (2017). The relational model: In search for lean and mean CPS technology. In: IEEE 15th International Conference on Industrial Informatics (INDIN). IEEE; 2017. p. 127-132

[21] Bonci A., Longhi S, Lorenzoni E., Pirani M., (2019). RMAS Architecture for Industrial Agents in IEC 61499, International Conference on Industry 4.0 and Smart Manufacturing, (ISM 2019).

[22] Womack J. P., Jones D., Roos D., (1990). *The machine that changed the world: The story of Lean production – Toyota's secret weapon in the global car wars that is revolutionizing world industry*, New York, Free Press.

[23] Liker, J.K. (2004). *The Toyota Way, 14 Management Principles from the World's Greatest Manufacturer.* McGraw-Hill, New York.

[24] O.hno, T., (1988), Toyota Production System: Beyond Large-Scale Production. Cambridge, Productivity Press.

[25] Schonberger, R. J., (1982), Japanese Manufacturing Techniques: Nine Hidden Lessons in Simplicity. New York: The Free Press.

[26] White, R. E., J. and N. Pearson, and J. R. Wilson, (1999), JIT manufacturing: a survey of implementations in small and large U.S. manufacturers, Management Science 45,1-15.

[27] White, R. E. and V. Prybutok, (2001), The relationship between JIT practices and type production system, Omega 29, 113-124.

[28] Shobha N. S., Subramanya K. N., (2018), A Review of Kanban-based Manufacturing Systems, International Journal of Research and Scientific Innovation (IJRSI) | Volume V, Issue IV.

[29] Krieg G. (2005), Kanban-Controlled Manufacturing Systems. Springer Berling

[30] Junior M.L., Filho M.G., (2010), Variations of the kanban system: Literature review and classification, Int. J. Production Economics 125 (2010) 13–21.

[31] Murino T., Naviglio G., Elpidio R., Zoppoli P., (2009), Single stage multi-product kanban system. Optimization and parametric analysis. Proceedings of the 8th WSEAS International Conference on System Science and Simulation in Engineering, ICOSSSE '09. 313-318.

[32] Reza A. Maleki1 and Gretchen Meiser, (2011), Managing Returnable Containers Logistics - A Case Study Part II - Improving Visibility through Using Automatic Identification Technologies, International Journal of Engineering Business Management, Vol. 3, No. 2 (2011).

[33] Menanno P., Ragno M., Savino M. and S. Muhammad, (2019), Implementing industry 4.0 technologies in lean production through e-kanban automotive production, In Proceedings of the Summer School Francesco Turco, vol. 1, 2019, pp. 458–463.

[34] Function Block Development Kit (2008), Holobloc Inc. http://www.holobloc.com.

[35] 4Diac IDE (Integrated development environment). https://www.eclipse.org/4diac/.

[36]4DIAC-FORTE,RunTime Environment, IEC6149.
https://www.eclipse.org/4diac/en_rte.php

[37] Cengic G. (2009), Function Block Execution Runtime (Fuber),
http://fuber.sourceforge.net.

[38] ISaGRAF (2008), http://www.isagraf.com.

[39] Yoong L.H., Roop P.S., Salcic Z. (2009). Efficient Implementation of IEC 61499 Function Blocks, IEEE International Conference on Industrial Technology (ICIT), Gippsland.

[40] Vyatkin V., Chouinard J., (2008). On Comparisons of the ISaGRAF Implementation of IEC-61499 with FBDK and Other Implementations, In 6th IEEE International Conference on Industrial Informatics (INDIN), Daejeon, pp 289–294.

[41] Yoong L.H., Roop P.S., Vyatkin V., Salcic Z. (2007). Synchronous Execution of IEC 61499 Function Blocks Using Esterel. In: 5th IEEE International Conference on Industrial Informatics (INDIN), Vienna, pp 1189–1194.