Modellizzazione e implementazione modulare di funzioni di automazione per sistemi di produzione flessibili

Luca Ferrarini, Carlo Veber, Valentina Schirò

In questo articolo viene proposto un modello di controllo modulare per lo sviluppo delle funzioni di automazione per sistemi di produzione flessibili. In particolare, viene formalmente specificato un nuovo Modulo di Controllo ad oggetti e ne viene discusso l'utilizzo all'interno di un'architettura funzionale gerarchica e vengono quindi descritte le fasi della metodologia dell'applicazione dei Moduli di Controllo a sistemi manifatturieri. Inoltre viene indagata una sua implementazione con lo standard IEC 61499. Questo modello, e la sua implementazione, vengono infine utilizzati per lo sviluppo di funzioni di controllo di un impianto di laboratorio che emula un generico sistema manifatturiero.

Keyword

Standard IEC 61499, Macchina a Stati Finiti Modulare, Sistema di Produzione Flessibile

Il problema della definizione formale delle funzioni che un sistema di controllo deve eseguire ha rappresentato per molti anni una stimolante sfida. Molte proposte in questo senso si possono trovare in letteratura [1, 4]. Il problema diventa più critico se si ha a che fare con applicazioni reali [12], dal momento che esse pongono una serie di problematiche che sono "speciali" per costruzione, e che devono essere prima valutate e strutturate, e infine risolte con qualche tipo di tecnica di modellazione e analisi. Tuttavia, tale approccio risulta difficile se si affronta il problema della definizione di modelli standard sia per le funzioni che per il codice, che si possano adottare in scenari reali e che risultino facilmente adattabili alle richieste degli utenti e alle specifiche di prestazioni più efficaci [13]. In questo ultimo decennio, mentre lo sviluppo di componenti elettrici e meccanici di sistemi manifatturieri ha seguito questa tendenza verso la modularità e la standardizzazione, lo stesso non è accaduto per i componenti di controllo. Questo determina costi elevati, anche se talvolta celati, per quanto riguarda la progettazione, l'implementazione, la verifica, l'installazione e la manutenzione del sistema di controllo, e riduce le possibilità di ottenere delle reali caratteristiche di flessibilità, riconfigurabilità e riuso delle soluzioni di controllo [4].

L. Ferrarini, C. Veber, V. Schirò - Dipartimento di Elettronica e Informazione, Politecnico di Milano

Chiaramente, ciò richiede nuovi paradigmi di modellazione che possano catturare la descrizione dell'intero sistema e favorire la traduzione delle sempre più rigorose specifiche di controllo nella progettazione e implementazione del controllo. I modelli e gli approcci progettuali tradizionali utilizzati dagli ingegneri di controllo sono basati su paradigmi di modellizzazione di basso livello (spesso linguaggi di programmazione), approcci centralizzati, strumenti e soluzioni proprietarie.
Una possibile soluzione da un punto di vista metodologico e pratico è la separazione delle funzioni dal codice, che permette la generazione (semi-)automatica del codice di controllo, la precisa corrispondenza tra il modello di progettazione e il modello di implementazione, e la possibilità di introdurre significative proprietà diagnostiche. Alcune proposte notevoli stanno emergendo sia nella letteratura scientifica [1, 2, 8] sia dell'ambito della standardizzazione [14], incoraggiando l'adozione di metodi di progettazione più formali e concetti orientati ad oggetti [9, 10]. Ancora più promettente per quanto riguarda la tecnica di progettazione e implementazione per il codice di automazione nel campo manifatturiero è il concetto di agente [2, 3, 5, 10]. Gli agenti sono componenti software creati come estensione naturale di oggetti caratterizzati da controllo autonomo della propria esecuzione, comportamenti sia reattivi che "proattivi", che cooperano per il raggiungimento di un obbiettivo comune.
Il modulo di controllo presentato in [3] vuole rappresentare il comportamento di un agente evidenziando quelle caratteristiche funzionali rilevanti per il controllo di sistemi manifatturieri e non solo. Il lavoro qui presentato riconsidera la proposta fatta in [3] cercando di specificare questi modelli astratti, attraverso il formalismo degli automi modulari chiamato MFSM (Modular

Finite State Machine) definito in [6, 7], e propone una loro possibile implementazione secondo lo standard IEC 61499 [14].
Il documento è organizzato nel modo seguente. Nella prossima sezione (Definizione del modello) viene presentato il modello di controllo e il suo principale elemento chiamato Modulo di Controllo (CM) e ne viene data una specifica tramite il formalismo degli automi modulari. La terza sezione presenta una possibile implementazione dei concetti del CM usando il blocco funzionale (FB) definito nello standard IEC 61499. Nella quarta sezione viene quindi descritta la metodologia di progettazione di un'applicazione di controllo basata sui CM. Tale metodologia è stata applicata per la progettazioni delle funzioni di controllo di un impianto da laboratorio, i cui risultati sono riassunti nella penultima sezione (Applicazione ad un impianto di laboratorio). Nell'ultima sezione vengono quindi riassunti i principali risultati del lavoro qui proposto e tracciate linee guida per possibili sviluppi futuri.

Definizione Del Modello

Il modello proposto in [3] è basato su un elemento principale detto Modulo di Controllo (CM), rappresentato nella figura 1. L'idea del CM riproduce il comportamento di un agente in un sistema basato su agenti. La specifica del CM è riportata qui di seguito per ragioni di completezza.
Fondamentalmente, il modulo di controllo può essere concepito come realizzato dall'opportuna aggregazione di diversi sottomoduli che lavorano in parallelo: uno State Manager, un Alarm Manager e una serie di operazioni (Oi). Le operazioni dei moduli di controllo corrispondono ai metodi degli oggetti nel modello procedurale. Le operazioni sono rappresentate come due sottomoduli: uno, generico, per gestire la richiesta dell'operazione (Operation Manager), e il secondo per la specifica del corpo dell'Operation Body. Ovviamente c'è un Operation Manager e un Operation Body per ogni operazione che il CM può eseguire. Lo State Manager, invece, incapsula lo stato interno dell'oggetto di automazione. Qui lo stato dell'oggetto è concepito come una serie di restrizioni all'esecuzione delle operazioni, i.e. le regole secondo le quali un oggetto può rispondere alla richiesta di un servizio. Per esempio, un oggetto "shuttle" non può eseguire due operazioni di rilascio una di seguito all'altra, dato che tra le due deve essere eseguita un'operazione di ripresa. Il significato di "stato" è di fondamentale importanza. In realtà tutti gli oggetti, a qualsiasi livello gerarchico, hanno il proprio stato, che non deve necessariamente essere coerente, dal momento che essi rappresentano soltanto restrizioni all'esecuzione dell'operazione. Infine, l' Alarm Manager confronta lo stato logico di un agente (rappresentato dallo State Manager) con quello effettivo misurato con i sensori. Per esempio, l'agente "shuttle" può trovarsi in uno stato che rappresenta uno stato dello shuttle fisso davanti alla posizione del buffer: se un sensore dovesse mostrare un movimento dello shuttle, viene subito diramato l'allarme.
Sia il CM che i sotto-moduli sono blocchi (Block) che hanno una o più interfacce e un unico comportamento. Esistono cinque tipi di sotto-moduli: *State Manager, Alarm Generator, Operation Manager, Operation Body e State Operation Manager*. Un CM è composto da: un comportamento Composite ovvero l'aggregazione dei sottomoduli, delle connessioni di dati e di eventi tra le interfacce dei sotto-moduli; 4 tipi di interface. Ogni sottomodulo è composto da: un comportamento MFSM, descritto dal formalismo MFSM [6]; un'interfaccia determinata univocamente una volta conosciutone il comportamento. Una modellizzazione di questi concetti secondo la specifica UML è rappresentata nella figura 2.

Interfacce del Modulo di Controllo
Il Modulo di Controllo ha 4 interfacce: Input Interface, Output Interface, Error Interface e Alarm Interface (cfr figura 2).
- *Input Interface*: tramite questa interfaccia il CM riceve le richieste di operazioni dagli altri CM. Questa interfaccia è composta da porte di ingresso (Input Ports), una per ogni Operazione eseguita dal CM; ogni porta è una serie di eventi e dati I/O secondo il diagramma per classi (Class Diagram) mostrato nella figura 2.
- *Output Interface*: tramite questa interfaccia il CM richiede le operazioni agli altri CM. È composta da una serie di porte di

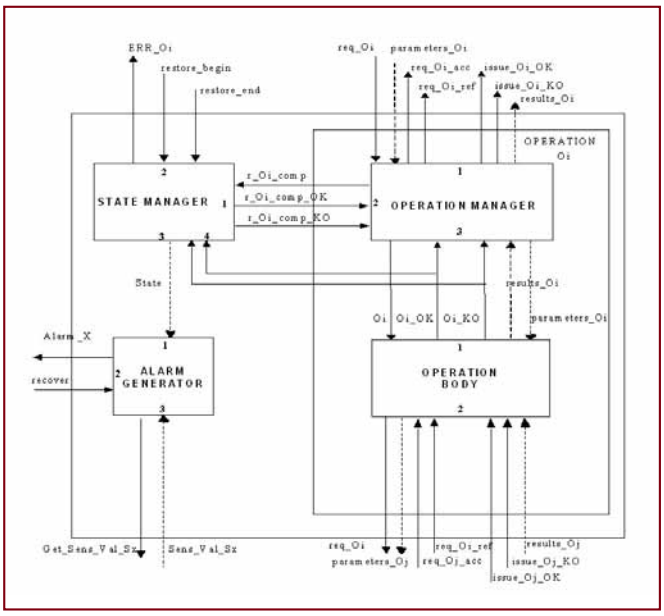

Figura 1 - Schema a blocchi del CM

req_Oi	operazione Oi richiesta da un modulo di livello superiore
r_Oi _comp	richiesta di compatibilità da parte dell'Operation Manager al lo State Manager
r_Oi_comp_OK	lo State Manager comunica all'Operation Manager che la richiesta è compatibile con lo stato del sistema
r_Oi_comp_KO	lo State Manager comunica all'Operation Manager che la richiesta non è compatibile con lo stato del sistema
req_Oi_acc	l'Operation Manager inoltra la compatibilità della richiesta al livello superiore
req_Oi_ref	l'Operation Manager inoltra la non-compatibilità della richiesta al livello superiore
Oi	l'esecuzione dell'Operation Body viene richiesto dall'Operation Manager
Oi _OK	l'esecuzione dell'operazione Oi è stata completata con successo
Oi _KO	l'esecuzione dell'operazione Oi non è stata completata o è fallita
issue_Oi _OK	il completamento dell'operazione Oi è comunicato dal Operation Manager al livello superiore
issue_Oi _KO	il fallimento dell'operazione Oi è comunicato dall'Operation Manager al livello superiore
ERR_Oi	lo State Manager rileva un errore quando Oi fallisce e lo invia al livello supervisore
restore_begin	l'attività del sistema nominale è stata ripristinata, tornando allo stato iniziale dell'operazione in corso
restore_end	l'attività del sistema nominale è stata ripristinata andando allo stato finale dell'operazione in corso
Alarm_X	l'Alarm Manager dirama un allarme per una non corrispondenza dei valori dei sensori
recover	un livello supervisore comunica che l'allarme è rientrato

uscita (Output Ports), una per ogni operazione che il CM richiede; ogni porta è una serie di eventi e dati I/O secondo il diagramma per classi (Class Diagram) mostrato nella figura 2.
- *Error Interface*: tramite questa interfaccia il CM comunica ad un blocco di livello superiore il fallimento di un'operazione (Err_Oi), e riceve l'evento di ripristino (Restore).
- *Alarm Interface*: tramite questa interfaccia il CM comunica ad un blocco di livello superiore l'occorrenza di allarmi (Alarm_X), e riceve l'evento di allarme rientrato (Recover).

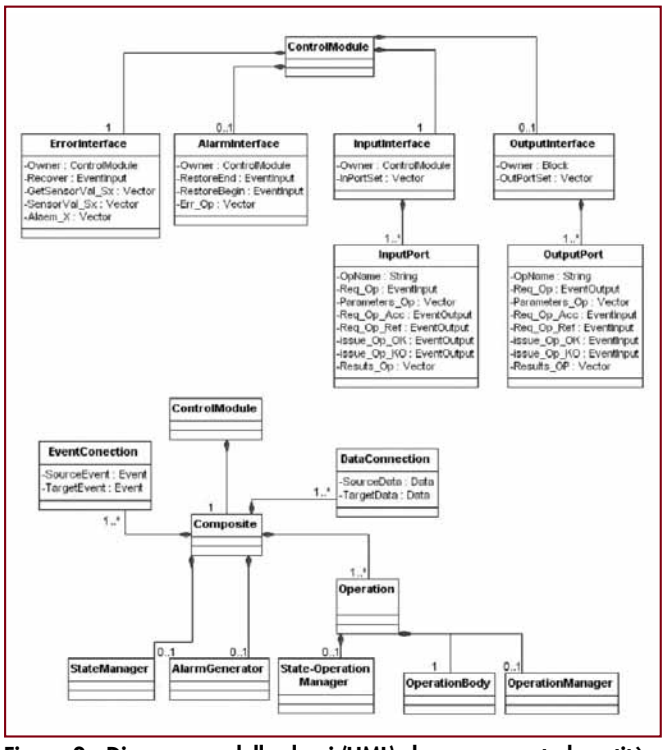

Figura 2 - Diagramma delle classi (UML) che rappresenta le entità che compongono un CM

Comportamento del Modulo di Controllo
Il Comportamento del Modulo di Controllo (cfr figura 2) è composto da:
- 1 o 0 State Manager;
- 1 o 0 Alarm Generator;
- n Operazioni. Per ciascuna di esse si ha: 1 o 0 Operation Body; 1 Operation Manager o 1 State-Operation Manager (questo sotto-modulo è presente solo se non c'è uno State Manager);
- un insieme di Connessioni di dati ed eventi tra le interfacce dei sotto-moduli e le interfacce del CM.

Comportamento dei sotto-moduli
Il comportamento dei sotto-moduli è descritto dal formalismo di MFSM [6]. Per ogni tipo di sotto-moduli sono stati definiti degli automi di riferimento che ne modellizzano i comportamenti più comuni.

Operation Manager
L'Operation Manager riceve la richiesta dell'operazione associata, richiede la verifica di compatibilità allo State Manager e, se la risposta è positiva, comanda l'esecuzione dell'operazione. Il modello MFSM che descrive il comportamento di un generico Operation Manager è delineata nella figura 3.

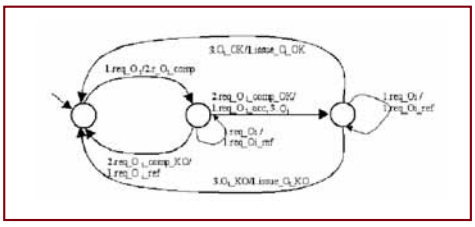

Figura 3 - Modello MFSM del comportamento di un generico Operation Manager

State Manager
Lo State Manager include le restrizioni sull'esecuzione dell'operazione. Generalmente è necessario utilizzare MFSM diversi per i diversi CM. Nella figura 4 viene mostrata la descrizione MFSM di uno State Manager in un CM con n operazioni che non possono essere parallele (tutti i diversi stati possono essere ridotti in un'unica rappresentazione simbolica di un singolo stato). Nel modello proposto è possibile distinguere tre diversi tipi di stati i cui significati vengono spiegati qui di seguito:
- *Waiting state*. Stato di Attesa (cerchio vuoto): il modulo sta aspettando una richiesta di operazione da parte dell'Operation Manager; può essere associato alla configurazione di un impianto in cui il componente fisico è immobile.
- *Motion state*. Stato di Moto (cerchio a sfondo grigio): l'operazione è in corso e il modulo è in attesa della sua conclusione; può essere associato alla configurazione di un impianto in cui il componente fisico è in movimento.
- *Error state*. Stato di Errore (cerchio a sfondo tratteggiato): stato associato al fallimento di un'operazione che viene trasmesso dallo stesso Operation Body e inviato ad un livello supervisore ed allo State Manager.

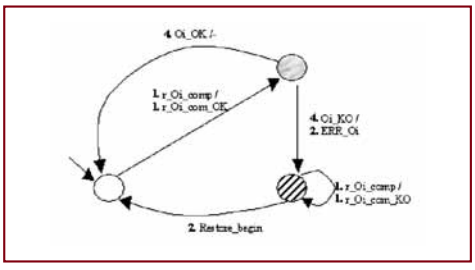

Figura 4 - Modello MFSM del comportamento di un State Manager con n operazioni

State-Operation Manager
Quando un CM gestisce solo un'operazione o n operazioni senza restrizioni reciproche, lo State Manager non è necessario, ma il CM ha un sotto-modulo che agisce in funzione di State Manager e di Operation Manager: lo State-Operation Manager. Il suo comportamento è rappresentato nella figura 5.

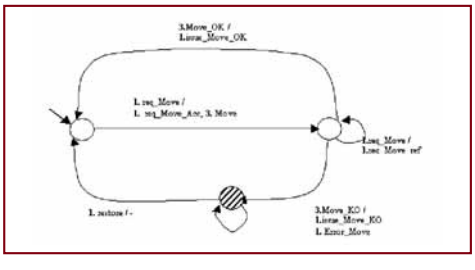

Figura 5 - Modello MFSM del comportamento di un generico State-Operation Manager

Operation Body
Ovviamente vi è un diverso MFSM per ciascuna operazione. Degno di nota è lo schema di richiesta dell'operazione, mostrato nella figura 6, che è presente nell'Operation Body ogniqualvolta questo richiede un'operazione ad un modulo (di livello inferiore). In particolare, la transizione marcata con il trigger event expired permette la trasmissione del fallimento di un'operazione quando nessun segnale di risultato appare entro un tempo DT prefissato. Un tale evento può essere emesso da un timer, non rappresentato nel modello complessivo, i cui input event sono Start_timer e Stop_timer, e il cui unico output event è scaduto/terminato/concluso.

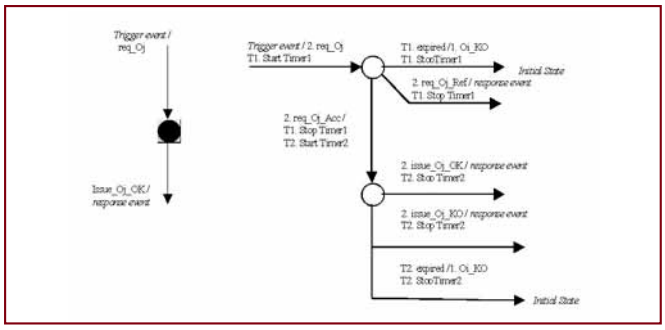

Figura 6 - Modello MFSM che rappresenta la richiesta di operazione

Alarm Generator
Ha il compito di procurarsi continuamente informazioni sia dallo State Manager che dai sensori, confrontarle e, se è il caso, diramare un allarme. In particolare, per ciascuno stato nello State Manager, una serie di valori previsti dei sensori è racchiuso nell'Alarm Generator e il confronto viene effettuato tra questi valori e quelli reali. Quando non coincidono, viene diramato un evento alarm ALARM_X finché non viene attuato un evento recover da parte di un Alarm Manager.

Canali di Comunicazione
Due Moduli di Controllo si scambiano dati ed eventi attraverso canali di comunicazione. Un Canale si stabilisce tra due moduli: un modulo chiamante e un modulo chiamato. Attraverso il canale, il modulo chiamante richiede al modulo chiamato l'operazione (Req_Oi). Il modulo chiamato risponde tramite il canale (Req_Oi_Acc, Req_Oi_Ref, issue_Oi_OK, issue_Oi_KO).
Il canale è una serie di Port Connection, una per ogni operazione richiesta dal modulo chiamante al modulo chiamato. Una Port Connection si compone di una coppia di porte: la porta di uscita (Output Port) legata all'operazione dell'Output Interface del modulo chiamante e una porta d'ingresso (Input Port) legata all'operazione dell'Input Interface del modulo chiamato. Tra le due porte connesse da una Port Connection vengono stabilite opportune connessioni di dati ed eventi.

Applicazione basata sui Moduli di Controllo
Un'applicazione dei Moduli di Controllo è composta da: 1 o più CM, 1 o più Canali, 1 Rete Complementare composta da Blocks. La Rete Complementare deve realizzare le seguenti funzioni:
gestione allarmi: la rete riceve gli eventi di allarme (Alarm X) dagli Alarm Generator dei CM, e quando l'allarme è rientrato, la rete lo comunica al CM generando l'evento Recover_Mi;
gestione errori: la rete riceve gli eventi di errore (Err_Oi) dallo State Manager, e quando il problema è riconosciuto e risolto, la rete lo comunica al CM generando l'evento Restore_Mi;
richiesta di servizio ai CM: la rete interagisce con i CM richiedendo i loro servizi (Req_Oi) e ricevendo le loro risposte (Req_Oi_Acc, Req_Oi_ref, issue_Oi_OK, issue_Oi_KO).
La rete deve esportare tre interfacce adeguate per realizzare le tre funzioni sopra descritte: Alarm Manager Interface, Error Manager Interface e Output Interface. Queste tre interfacce comunicano rispettivamente con le Alarm Interface, le Error Interface and le Input Interface dei CM, attraverso appositi canali.

Implementazione del CM utilizzando i blocchi funzionali di IEC 61499

Il CM può essere implementato usando i blocchi funzionali di IEC 61499 [14]. Un'implementazione di un CM secondo lo standard IEC 61499 sarà chiamato CMFB. In particolare, un CMFB è un Composite FB come definito in IEC 61499. La sua interfaccia (cfr figura 7) è composta da eventi e dati I/O delle interfacce del CM. Il CMFB contiene i Basic FB che rappresentano i sotto-moduli del CM.
Ogni sotto-modulo è implementato come un Basic FB la cui interfaccia è composta da eventi e dati I/O dell'interfaccia del relativo sotto-modulo.

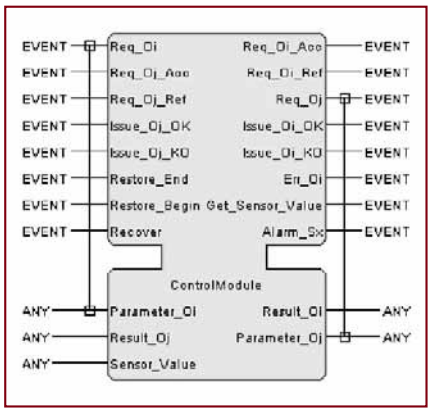

Figura 7 - Interfaccia di un CMFB

L'MFSM che rappresenta il comportamento del sotto-modulo è tradotto nello ECC del FB. Ad esempio, nella figura 8, viene mostrato lo ECC di un generico Operation Manager. Le Connessioni di dati ed eventi dei Basic FB che implementano i sotto-

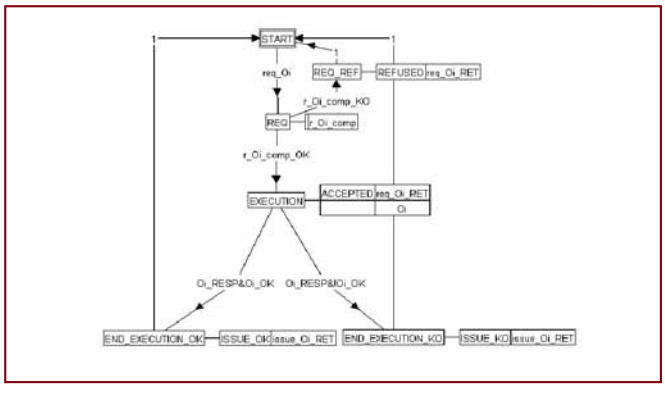

Figura 8 - ECC di un generico Operation Manager

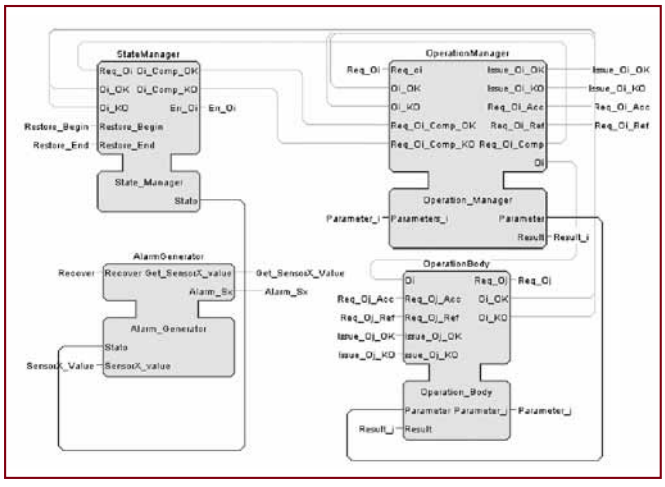

Figura 9 - FB che implementano un Basic CMFB

moduli sono le stesse connessioni che si trovano tra i sotto-moduli del CM. L'implementazione del comportamento di un generico CMFB è mostrata nella figura 9.

Metodologia di progettazione di un'applicazione basata sui CM

La metodologia è divisa principalmente in 5 fasi sequenziali di seguito descritte.

Modello fisico
L'impianto è concepito come l'aggregazione di componenti fisici. Un componente è concepito come l'aggregazione di componenti di livello inferiore. In particolare:
- *Elementary Component*: modellizzano i componenti fisici che interagisco con l'impianto (sensori e attuatori);
- *Equipment*: aggrega uno o più Elementary Component. A sua volta un Equipment può essere definito come aggregazione di due o più Equipment;
- *Device*: aggrega uno o più Equipment;
- *Unit*: aggrega uno o più Device;
- *WorkArea*: aggrega uno o più Unit. A sua volta una WorkArea può essere definita come aggregazione di due o più WorkArea.

Modello funzionale
Le funzioni di controllo vengono identificate e associate a componenti fisici. Ogni funzione di controllo è associata ad una classe astratta con la dichiarazione dei metodi che realizzano quella funzione. Una classe usa classi di livello inferiore per realizzare la propria funzione.
Il modello ottenuto è un'organizzazione gerarchica secondo il paradigma ad oggetti, strutturata su 4 livelli costituiti da classi astratte e dalle loro relazioni di "uso" che riflette il modello dell'impianto.
Le funzioni di controllo di alto livello (come le unità di coordinazione, diagnosi, arresto) non possono essere associate a componenti fisici, così vengono introdotti oggetti funzionali che non sono localizzati in nessun livello della gerarchia.

Modello dell'applicazione a Moduli di Controllo
Lo sviluppo del modello dell'applicazione a CM viene fatto attraverso i passi qui elencati:
- *Individuazione dei CM*: ogni classe del modello di controllo fino al livello delle Unità è associata ad un CM; ogni metodo di questa classe è un'operazione del CM.
- *Specifica dei sotto-moduli*: per ogni sotto-modulo di un CM è necessario specificarne il comportamento; una volta conosciuto il comportamento, l'interfaccia del sotto-modulo è univocamente determinata.
- *Specifica del CM*: una volta specificati i sotto-moduli, le interfacce e la connessione interna del CM sono univocamente determinate.
- *Canali*: si stabilisce tra due moduli un Canale che è collegato a due classi in relazione di uso; una volta conosciuta la Porta connessa, il canale viene univocamente specificato.

Modello della Rete Complementare
Gli oggetti funzionali introdotti nel modello funzionale vengono realizzati da blocchi della Rete Complementare. Le interfacce Alarm ed Error della Rete Complementare vengono univocamente determinate una volta conosciuti gli eventi di allarme ed errore generati dai moduli. L'interfaccia di Output è univocamente determinata una volta conosciute le operazioni richieste dai blocchi. I comportamenti dei blocchi devono essere specificati.

Implementazione con IEC 61499
Innanzitutto, devono essere definiti i FB necessari per l'applicazione: devono essere implementati i CMFB che rappresentano i CM e il/la rete di FB che rappresenta/no la Rete Complementare. Successivamente, l'applicazione deve essere distribuita su Dispositivi e Risorse come definito dallo standard IEC 61499.

Applicazione ad un impianto di laboratorio

Il modello di controllo modulare descritto è stato utilizzato per controllare un impianto di laboratorio (cfr figura 10) riproduce un tipico sistema manifatturiero. L'impianto è costituito da 4 macchine, 1 magazzino dei pezzi, 1 shuttle robotizzato, 1 stazione di carico e 1 di scarico. Le parti che devono essere lavorate hanno un codice costituito da un colore che ne rappresenta il piano di processo (ricette) e i colori rappresentano tanto le proprietà quanto le operazioni della macchina. Le operazioni di lavorazione sono rappresentate dal posizionamento di anelli colorati sopra ciascun pezzo (ogni macchina lavora anelli dello stesso colore).
Il processo di lavorazione si svolge nel modo seguente: un pezzo non lavorato viene caricato e la sua ricetta viene letta da un sensore di colore e, seguendo la sequenza specificata nella ricetta è portato dallo shuttle alla/e macchina/e. Se la macchina che lo deve lavorare è ancora occupata, il pezzo viene immagazzinato in un buffer (se disponibile). Il pezzo lavorato viene scaricato e controllato con un secondo sensore di colore per confrontare la sua lavorazione con la ricetta iniziale.
L'architettura hardware del sistema di controllo è composta da 1

JavaPLC, fornito da Elsist e 2 PC industriali collegati via Ethernet TCP/IP. I PC industriali sono equipaggiati con una JVM standard cosicché l'intero sistema può essere controllato usando il codice Java come principale linguaggio di programmazione.

Applicando la metodologia descritta nel paragrafo 4, prima viene creato il modello dell'impianto, poi il modello funzionale. In seguito, i moduli di controllo vengono applicati al modello funzionale fino al livello delle Unità. Al di sopra del livello delle Unità sono state introdotte due classi che gestiscono le funzioni di controllo di più alto livello: Prodotto (Product), che si occupa del coordinamento dell'Unità e Supervisore (Supervisor), che realizza le altre funzioni di controllo di alto livello.

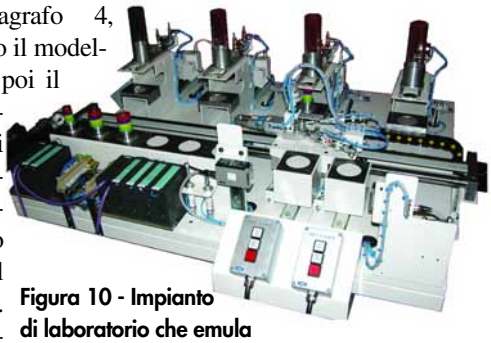

Figura 10 - Impianto di laboratorio che emula un sistema di produzione flessibile

Il modello progettato è stato implementato usando concetti IEC 61499 come descritto nella sezione precedente. In particolare, tutti i CM sono stati implementati come CMFB. Per quanto concerne la distribuzione del modello ottenuto (secondo la specifica IEC 61499) si ha che: ad ogni Unità corrisponde un dispositivo con una singola risorsa; ad ogni Prodotto corrisponde un risorsa, ma tutte queste risorse risiedono su un unico dispositivo. L'applicazione di controllo è stata progettata usando l'editor FBDK fornito da Rockwell Automation.

In questo Modello di Controllo il prodotto è caratterizzato dal fatto di avere un controllo autonomo dell'esecuzione della propria ricetta, che mostra un comportamento sia reattivo che proattivo. È un CM e interagisce con le unità del modello (che sono a loro volta dei CM) per eseguire la sua ricetta richiedendo ad esse le loro operazioni e reagendo alle differenti risposte inviate dalle unità.

Conclusioni

In questo articolo viene presentato un modello modulare ad oggetti di automazione particolarmente utile per sviluppare le funzioni di controllo per i sistemi manifatturieri. Oltre a presentare una specifica formale dei componenti di tale modello, ne viene altresì discussa l'implementazione con blocchi funzionali IEC 61499, mostrandone la fattibilità pratica e la conformità con un nascente standard internazionale dedicato ai sistemi di automazione distribuiti.

Il modello e la sua implementazione si sono dimostrati facilmente adattabili in un'architettura funzionale di controllo gerarchica. Infine, vengono mostrate un'applicazione di tale modello e la sua implementazione per le funzioni di controllo per un impianto di laboratorio che riproduce un generico sistema manifatturiero.

Lavori futuri includeranno la definizione formale della Rete Complementare; la progettazione di un ambiente di sviluppo integrato in grado di editare e gestire i CM e le loro implementazioni; infine lo studio di un'implementazione dei CM basata su agenti.

Riferimenti

[1] E. Almeida, D.M. Tilbury, "Automatic Logic Generation for Reconfigurable Cell-Based Manufacturing Systems", *Wodes'04, Workshop on Discrete Event Systems 2004*, Reims, France, 22-24 settembre, 2004.

[2] R.W. Brennan, M. Fletcher, D.H. Norrie, "An Agent-Based Approach to Reconfiguration of Real-Time Distributed Control Systems", *IEEE Trans. On Robotics And Automation*, vol. 18, no. 4, pp. 444-451, 2002.

[3] A. Castelnuovo, L. Ferrarini, "A Modular Formal Model for Pallet Transportation System in Machining Centres Automation", *CDC-ECC05*, 2005.

[4] A.W. Colombo, R. Schoop, P. Leitão, F. Restivo, "A Collaborative Automation Approach to Distributed Production Systems", *2nd IEEE Int. Conf. on Industrial Informatics*, pp. 27-32, 2004.

[5] S.M. Deen (editor), *Agent-based manufacturing*, Springer Verlag, Berlino, 2003.

[6] E.W. Endsley, *Modular Finite State Machines for Logic Control: Theory, Verification and Applications to Reconfigurable Manufacturing Systems*, PhD thesis, University of Michigan, 2004.

[7] E.W. Endsley, D.M. Tilbury "Modular Finite State Machines for Logic Control", in *Proceedings of the IFAC Workshop on Discrete Event Systems*, 2004.

[8] L. Ferrarini, G. Fogliazza, "Advanced Control System Design for Machining Centres", *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM '01)*, 8–11 July 2001, Como, Italia, pp. 671-676, vol. I.

[9] L. Ferrarini, C. Veber, G. Fogliazza, "Modelling, Design and Implementation of Machining Centres Control Functions with Object-Oriented Techniques", *IEEE/Asme International Conference on Advanced Intelligent Mechatronics (AIM 2003)*, 20-24 Luglio, 2003, Kobe, Giappone, p. 1037-1042.

[10] B. S. Heck, L. M. Willis, G. J. Vachtsevanos, "Software Technology for Implementing Reusable, Distributed Control Systems", *IEEE Control Systems Magazine*, pp. 21-35, Febbraio 2003.

[11] Holobloc.com, Function Block-Based, Holonic Systems Technology. Available on: www.holobloc.com.

[12] E. Park, D. M. Tilbury, P. P. Khargonekar, "Modeling, Analysis, and Implementation of Logic Controllers for Machining Systems using Petri Nets and SFC", *Wodes 2002*.

[13] S.S. Shah, E.W. Endsley, M.R. Lucas, D.M. Tilbury, "Reconfigurable logic control using modular FSMS: design, verification, implementation, and integrated error handling", *ACC 2002*, p. 4153, 2002.

[14] Standard IEC 61499, *Function Blocks for Industrial-Process Measurements and Control System*, IEC TC65/WG6, Draft, 18/9/96.

[15] TORERO Project, Total life cycle web-integrated control, IST-2001-37573. Available on: http://www.uni-magdeburg.de/iaf/cvs/torero. ■