

UNA PROPOSTA DI METODOLOGIA INTEGRATA DI SPECIFICA PER IL CONTROLLO IN TEMPO REALE: IL CASO DEGLI AZIONAMENTI ELETTRICI

L. Baresi ^(*), S. Carmeli ⁽⁺⁾, A. Monti ⁽⁺⁾ e M. Pezzè ^(*)

⁽⁺⁾ *Dipartimento di Elettrotecnica - Politecnico di Milano
Piazza Leonardo da Vinci 32 - 20133 Milano*

Tel: +39-2-23993714 - Fax: +39-2-23993703 - e-mail: [carmeli,anto]@etec.polimi.it

^(*) *Dipartimento di Elettronica e Informazione - Politecnico di Milano
Piazza Leonardo da Vinci 32 - 20133 Milano*

Tel: +39-2-23993400 - Fax: +39-2-23993411 - e-mail: [baresi,pezze]@elet.polimi.it

Sommario

L'uso di metodi formali nello sviluppo del software in ambito industriale è tuttora assai limitato. La difficoltà intrinseca e l'abitudine a tecniche di specifica informali sono gli ostacoli principali alla loro diffusione.

In quest'ottica, si propone un approccio innovativo per la progettazione di sistemi ibridi, coniugando i metodi formali operazionali con le notazioni solitamente utilizzate nella pratica industriale. La metodologia, sviluppata nell'ambito del progetto CEE ESPRIT chiamato INFORMA, è applicata alla realizzazione del controllo degli azionamenti elettrici.

Parole chiave

Sistemi ibridi, Software in tempo reale, Reti di Petri, Functional Block Diagram

1. IL SOFTWARE IN TEMPO REALE

Il software in tempo reale richiede metodologie e processi di sviluppo più complessi rispetto alle applicazioni tradizionali. Oltre alla correttezza funzionale del prodotto, assumono particolare importanza i requisiti temporali. La "validità" della computazione è subordinata al soddisfacimento dei vincoli temporali. Nel caso di applicazioni di controllo, la correttezza del software determina la *sicurezza* dell'impianto. Eventuali errori del software di controllo potrebbero provocare danni - irreparabili - alle strutture controllate o alle persone (gli operatori).

L'elevata criticità delle applicazioni in questione giustificherebbe l'utilizzo di tecniche di specifica e di progettazione che consentano la verifica (automatica) del sistema in esame. I metodi formali potrebbero, quindi, essere impiegati proficuamente per migliorare la qualità del codice prodotto. Purtroppo, però, nella pratica industriale si preferiscono tecniche di specifica informali o semi-formali anche per applicazioni critiche. L'uso di metodi formali è limitato ad applicazioni pilota ed a studi di fattibilità [8, 10]. Difficilmente i metodi formali sono utilizzati per la verifica e la validazione dell'intero sistema.

Le ragioni della scarsa diffusione vanno ricercate nella difficoltà dei metodi formali. Senza una buona preparazione matematica di base queste tecniche sono difficilmente utilizzabili [11]. Attualmente, si preferiscono tecniche immediatamente "disponibili", che spesso forniscono linguaggi grafici di programmazione semplici ed immediati e non richiedono conoscenze specifiche. Date queste premesse, i costi da affrontare per l'istruzione del personale, uniti ad un certo scetticismo ed alla reticenza a modificare procedure di sviluppo ampiamente collaudate, precludono l'effettiva diffusione dei metodi formali a livello industriale.

In letteratura esistono diverse proposte [12, 16] che coniugano le notazioni di specifica solitamente utilizzate in ambito industriale con metodi formali operazionali. Definendo una corrispondenza fra le due notazioni (informale e formale) si rendono disponibili all'utente i benefici dei metodi formali, senza richiedere alcuna conoscenza specifica dei metodi stessi. Gli utenti usano i metodi formali in modo "trasparente": si giovano dei benefici senza dover imparare nuovi linguaggi (formali) o cambiare il processo di progettazione.

Nessuna delle proposte presenti in letteratura si rivolge, in modo specifico, al settore dell'automazione. Queste applicazioni sono caratterizzate da un legame stretto - ed imprescindibile - fra il *controllore* (il prodotto software) ed il *controllato* (l'impianto). Non sarebbe corretto pensare alla verifica del software di controllo come se fosse un componente isolato. È necessario vedere l'applicazione software all'interno dell'anello di controllo, come schematizzato in Figura 1. La verifica del controllore deve essere fatta in funzione dei dati prodotti dalla richiusura diretta dell'anello di controllo.

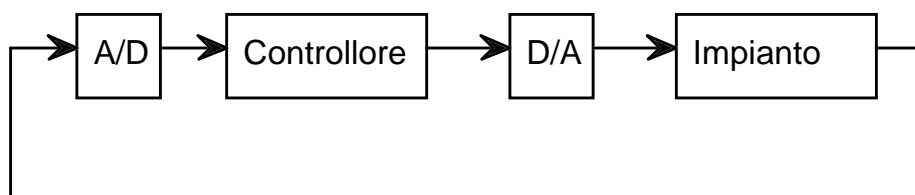


Figura 1: Struttura di un sistema di controllo a microprocessore

L'intero sistema (Figura 1) è dato dall'unione di due sottosistemi, caratterizzati da un comportamento tempo continuo (l'impianto) e tempo discreto (il controllore). Questo tipo di applicazioni, note anche come *sistemi ibridi*, sono stati ampiamente studiati in letteratura, sia proponendo modelli formali ad-hoc [14, 2, 1], sia in funzione della progettazione del software di controllo [17]. Due quesiti restano tuttora irrisolti:

- come avvicinare i metodi formali di specifica del software alla pratica di progettazione industriale.
- come adattare il concetto di verifica e simulazione del software ai problemi dell'automazione descritti in precedenza.

L'articolo presenta un approccio innovativo per la progettazione di sistemi ibridi. La metodologia, sviluppata nell'ambito del progetto CEE ESPRIT denominato INFORMA, unendo i metodi formali operazionali e le notazioni di specifica utilizzate in ambito industriale, fornisce una risposta ai quesiti precedenti. In questo articolo, l'approccio è applicato ad un preciso settore applicativo dell'automazione: gli azionamenti elettrici.

Il resto dell'articolo è organizzato come segue. Il Paragrafo 2 illustra brevemente gli azionamenti elettrici ed i problemi connessi. Nel Paragrafo 3 si presentano i risultati già conseguiti dalla pratica industriale e si discutono i problemi tuttora irrisolti. Nel Paragrafo 4 si introducono le reti di Petri e si spiega la corrispondenza fra *Functional Block Diagram* e reti di Petri. Il Paragrafo 5 introduce i problemi relativi e le soluzioni trovate per quanto riguarda la simulazione. Il Paragrafo 6 dà una breve descrizione delle finalità e degli obiettivi del progetto CEE ESPRIT chiamato INFORMA. Infine, il Paragrafo 7 riassume i propositi dell'articolo e presenta alcune conclusioni.

2. GLI AZIONAMENTI ELETTRICI

La struttura di controllo di un azionamento elettrico è composta da un insieme di anelli di controllo nidificati. In particolare, facendo riferimento alla Figura 2, si hanno - dall'interno verso

l'esterno - la regolazione di corrente o coppia, la regolazione di velocità e l'eventuale regolazione di posizione.

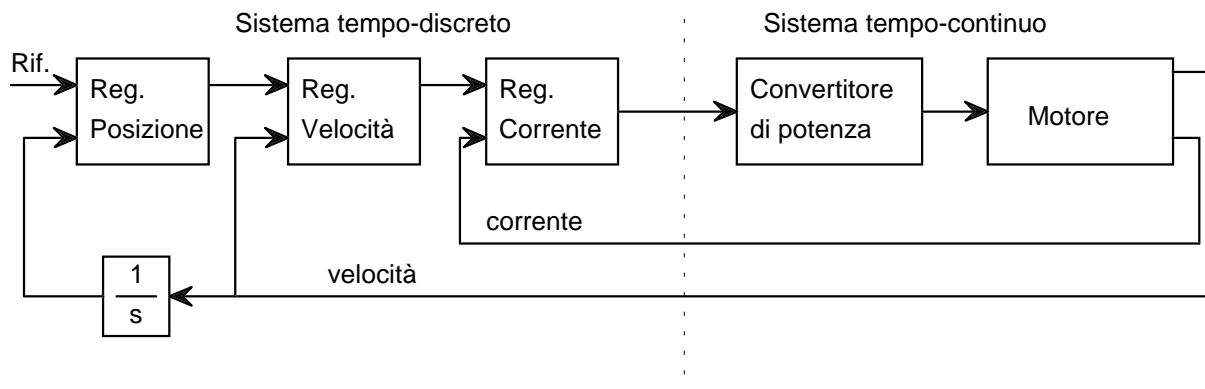


Figura 2: Schema di regolazione di un azionamento elettrico

I diversi anelli impongono vincoli temporali di granularità diversa, che richiedono soluzioni specifiche. Considerando l'hardware disponibile, è più facile gestire l'anello di velocità, che l'anello di corrente. Il problema tempo diventa ancora più rilevante quando si considera l'interfacciamento del sistema per il controllo diretto dell'accensione delle valvole (tiristori, transistor, IGBT), cioè si considera il convertitore di potenza. Spesso, i problemi vengono risolti utilizzando unità centrali separate e con caratteristiche diverse: un microprocessore *general purpose*, o un DSP, per l'anello di velocità e l'orientamento di campo, compresi, quindi, gli anelli di regolazione su assi di Park (tempo di ciclo pari ad 1 millisecondo) ed un microcontrollore per il controllo diretto delle valvole (tempo di ciclo pari a 400 microsecondi). Questa scelta è dovuta alle caratteristiche intrinseche del microcontrollore, che grazie alle sue elevate capacità di controllo dell'*input/output* può agire direttamente sul sistema controllato in tempi assai ridotti.

È inoltre possibile progettare l'anello di velocità avendo come obiettivo la riusabilità dei componenti e la riconfigurabilità del prodotto. La standardizzazione dell'hardware e del software favoriscono la rapidità nella produzione, pur mantenendo elevata la qualità del prodotto. In questo modo, l'esperto di controllo può realizzare il progetto del regolatore per il sistema in esame facendo riferimento a "schemi mentali" (blocchi funzionali) ormai standardizzati. Il progettista si pone ad un livello d'astrazione in cui le specifiche si definiscono connettendo blocchi funzionali predefiniti, prescindendo dai problemi di livello inferiore (hardware e software). I vantaggi di questo approccio sono:

- La rapidità nella produzione. Si salta la fase di codifica.
- La corrispondenza tra la specifica del controllore ed il codice realizzato (su questo punto si tornerà nel seguito).
- La qualità e la manutenibilità del codice (anche se ciò non garantisce la corrispondenza con le specifiche) garantita dal riuso di componenti "standard".

L'approccio descritto in precedenza, anche se ben radicato nella pratica industriale, presenta lacune, che devono essere rimosse al fine di ottenere una metodologia di specifica completa ed affidabile. Le deficienze principali sono:

- L'impossibilità di compiere verifiche di consistenza temporale sulle specifiche prodotte. Non è, cioè, possibile provare che l'esecuzione dei programmi definiti rispetti i tempi di ciclo richiesti.

- La mancanza di un approccio alla progettazione che integri il controllore (software) ed il controllato, sin dalle prime fasi del lavoro. Una metodologia completa ed integrata potrebbe servire anche come “linguaggio comune” tra i diversi progettisti.

3. LA PRATICA INDUSTRIALE

In ambito industriale si è cercato di ovviare alle lacune, evidenziate al paragrafo precedente, introducendo strumenti “specifici” di supporto alla progettazione del software [19]. Questi ambienti sono ancora lontani dal coprire l'intero processo di sviluppo. Fra questi, vanno ricordati gli strumenti di supporto alla metodologia chiamata *Functional Block Diagram* (FBD)¹. Il linguaggio a disposizione consente le operazioni fondamentali proprie dei linguaggi standard per PLC (blocchi predefiniti) e permette all'utente di programmare le funzionalità di blocchi particolari, chiamati *building block*, per risolvere problemi specifici.

Una specifica (programma) FBD è organizzata in sezioni. Ogni sezione è composta da uno a più blocchi. Spetta poi al progettista definire l'ordine di esecuzione delle diverse sezioni suddividendole in gruppi (*task*). Una sezione può essere eseguita all'interno del ciclo principale (*main task* PLC), che viene eseguito con una cadenza fissa programmabile dall'utente (tipicamente 100 millisecondi), oppure all'interno di altri *task* ad alta priorità eseguiti anch'essi con una cadenza fissa (al massimo ogni millisecondo). Un semplice programma FBD (Figura 3) potrebbe essere composto da tre *task*:

- *Main task* PLC per l'*input/output* digitale e la logica di start/stop.
- *Task 1* (ogni millisecondo con massima priorità) per l'anello di corrente.
- *Task 2* (ogni 4 millisecondi) per l'anello di velocità.

È poi possibile generare automaticamente il codice della specifica prodotta e condurre la simulazione direttamente sul microprocessore *target*. La corretta esecuzione dei diversi *task* deve essere assicurata da un sistema operativo *multitasking* ed in tempo reale.

Una metodologia di progetto quale quella descritta in precedenza può essere vista come il primo passo verso un approccio “standardizzato” per la progettazione del software. Non si risolve completamente il problema della produzione del software per gli azionamenti e, più in generale, il problema del progetto integrato dell'azionamento. La certezza, grazie alla generazione automatica del codice, della corrispondenza tra la specifica del controllore ed il codice prodotto non è sufficiente. Non si ha la possibilità, durante la progettazione, di provare il soddisfacimento dei requisiti temporali richiesti. È richiesto, quindi, un metodo di specifica che, sfruttando i metodi formali, consenta la verifica dei requisiti temporali già in fase di specifica, senza dover ricorrere all'esecuzione del codice direttamente sul *target*.

La metodologia richiesta deve consentire l'esecuzione (simulazione) della specifica, ponendo particolare attenzione alle grandezze di interesse per gli utenti. Deve essere possibile l'analisi automatica delle specifiche prodotte al fine di evidenziare errori e lacune sin dalla fase di definizione dei requisiti. Infine, deve essere possibile trasformare automaticamente la specifica prodotta in codice conforme ai requisiti iniziali.

¹ Metodologia standardizzata sia dall'ISO, che dall'IEC.

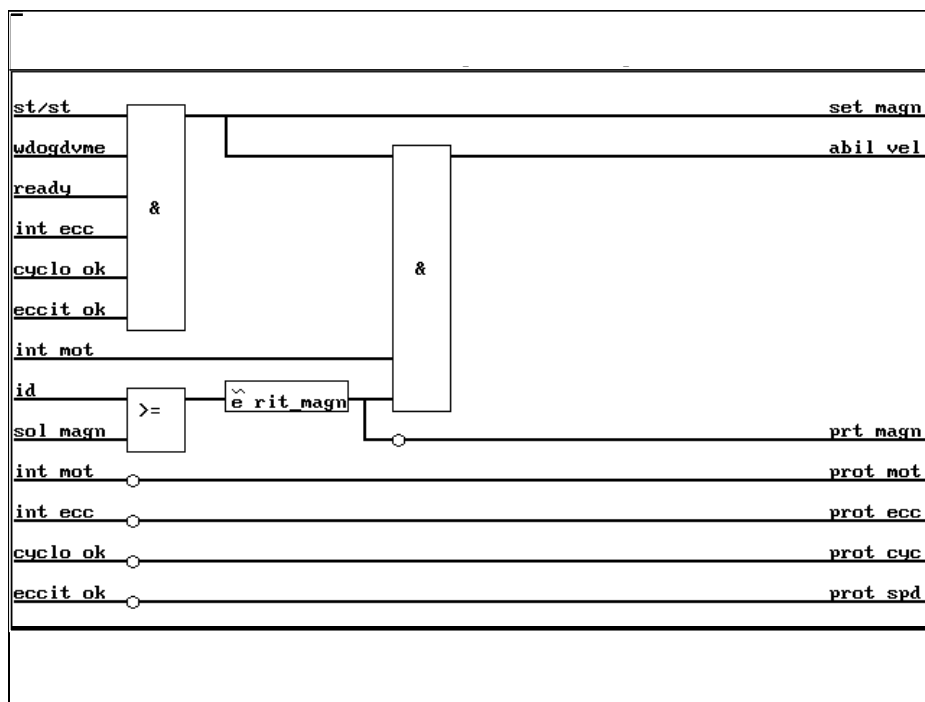


Figura. 3: una schermata tipo di FBD

I sistemi di controllo (degli azionamenti) impongono un ulteriore vincolo: il software prodotto non deve essere verificato come un'entità isolata, ma deve essere "provato" in funzione del sistema controllato. In altri termini, si ritiene che la progettazione di un sistema di controllo ad alta dinamica richieda una metodologia globale di progetto, che consenta al progettista di integrare tecnologie diverse per ottenere una specifica completa e formale del sistema (controllore-controllato) definito.

Nel seguito si illustrerà, quindi, la metodologia proposta, che risponde alle caratteristiche identificate in precedenza. La tecnica può fornire un valido ausilio al miglioramento della qualità del prodotto finale azionamento, facendo riferimento in particolare al software di controllo, ed al rispetto delle specifiche di sistema.

4. I FUNCTIONAL BLOCK DIAGRAM E LE RETI DI PETRI

Un modo per non stravolgere la pratica progettuale ed, al tempo stesso, facilitare la diffusione dei metodi formali nell'industria, consiste nell'integrazione delle notazioni informali utilizzate con metodi formali operazionali [4]. Nel nostro caso, è quindi possibile definire una corrispondenza fra i diagrammi FBD e le reti di Petri. In altri termini si traduce ogni costrutto della prima notazione in un'opportuna rete di Petri.

Al fine di facilitare il lettore non esperto, una rete di Petri (Figura 4) è un grafo bipartito orientato [20]. Questo significa che i nodi del grafo sono divisi in due gruppi: posti e transizioni, e gli archi non possono connettere due nodi che appartengono allo stesso gruppo. Graficamente, i posti sono rappresentati mediante cerchi e le transizioni mediante barre o rettangoli. Informalmente, i posti definiscono le condizioni necessarie perché si verifichi un evento (modellato da una transizione). La validità di una condizione è rappresentata per mezzo di un token - un piccolo cerchio nero - all'interno del posto opportuno. Continuando con la terminologia propria delle reti di Petri, l'insieme dei posti collegati con archi entranti ad una transizione t viene chiamato il preset di t . In modo analogo, l'insieme dei posti collegati con archi uscenti da t è detto il postset di t .

L'evoluzione di una rete di Petri può essere descritta nel modo seguente: in ogni istante si suddivide l'insieme delle transizioni in due sottoinsiemi: le transizioni abilitate e quelle non abilitate. Una transizione è abilitata in un certo istante se esiste almeno un token in ogni posto del preset della transizione. Lo scatto di una transizione abilitata t comporta la rimozione del token abilitante da ogni posto del preset di t e l'aggiunta di un token in ogni posto del postset di t . Ad esempio, considerando lo scatto della transizione $T1$ di Figura 4, si rimuoverebbe il token dal posto $P1$ e si creerebbe un nuovo token nel posto $P3$.

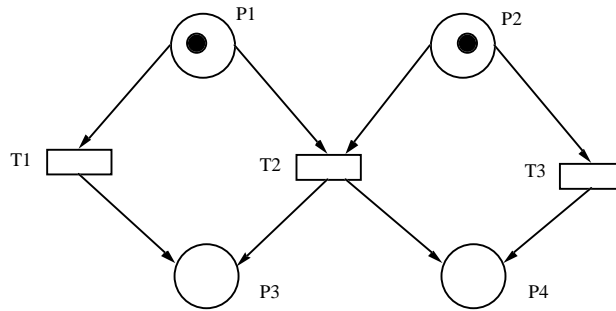


Figura 4: Un esempio di rete di Petri con marcatura

La “storia” di una rete di Petri è caratterizzata, quindi, sia dalla sequenza di scatti delle transizioni, che dalla disposizione dei token (marcatura della rete) nei diversi istanti.

Nel modello di reti di Petri appena descritto i token sono entità anonime. L'informazione non è contenuta nel token, ma è data dalla sua posizione all'interno della rete: la marcatura della rete di Petri. Utilizzando una notazione così povera da un punto di vista espressivo non è semplice descrivere un sistema reale e non è possibile modellarne gli aspetti temporali.

Per ovviare alle suddette limitazioni, in letteratura sono stati proposti diversi modelli di reti di Petri estese (o di alto livello), che consentono di gestire direttamente l'informazione e/o il tempo. Fra questi, le reti di Petri ER (Environment-Relationship, [13]) riassumono in un unico modello concettuale diversi approcci esistenti. Nelle reti ER:

- È possibile memorizzare delle informazioni all'interno dei token. Si passa dal token anonimo all'*environment*, cioè alla coppia identificatore, valore. Il contenuto informativo di una rete ER non è solamente la marcatura della rete, ma anche i valori associati ad ogni token.
- Ad ogni transizione t è associato un predicato ed un'azione. Il predicato definisce la condizione che deve valere fra i token selezionati dai posti del preset di t (tupla abilitante), affinché la transizione sia abilitata. L'azione stabilisce il contenuto informativo dei token prodotti dallo scatto della transizione (token nei posti del postset di t), in funzione della tupla abilitante.
- La regola di scatto diventa più complessa. Una transizione t è abilitata se esiste almeno un token in ogni posto del preset di t e questi token soddisfano il predicato associato a t . Lo scatto della transizione produce un token in ogni posto del postset di t , associando ad essi i valori definiti dall'azione di t .

Le reti di Petri ER possono essere ulteriormente estese per gestire il tempo (TER: Time Environment-Relationship, [13]). Ad ogni token si associa una variabile particolare *chronos*, che rappresenta l'istante di creazione del token, mentre ad ogni transizione si associa un intervallo di abilitazione temporale. Il tempo comporta un'ulteriore complicazione della regola di scatto. Informalmente, una transizione è abilitata (allo scatto) se è abilitata la transizione ER equivalente e sono rispettati i vincoli temporali. Affinché una rete temporizzata sia corretta è necessario, anche, imporre “vincoli” sul comportamento della rete: le variabili *chronos* devono rispettare le caratteristiche di monotonicità tipiche del concetto di tempo.

I vantaggi derivanti dall'uso di un modello formale (reti di Petri) vanno ricercati soprattutto nella possibilità di validazione automatica delle specifiche definite. Una rete di Petri può essere eseguita, permettendo all'utente di simulare il sistema modellato in casi particolari, ma può anche essere analizzata. Le tecniche di analisi (invarianti, raggiungibilità) consentono di ricavare una serie di informazioni dalla rete di Petri, che permettono di valutare la "bontà" della specifica prodotta. È possibile provare la mancanza di *deadlock*, il raggiungimento (o il non raggiungimento) di particolari stati del sistema e - in caso di modelli temporali - è possibile provare, anche, il rispetto dei vincoli temporali associati alla specifica.

Considerando la potenza espressiva, le reti di Petri TER sarebbero un modello formale adatto per la rappresentazione dei problemi descritti in precedenza. Non ci si può ritenere soddisfatti, invece, considerando la facilità d'uso e la capacità di modellazione offerta (dalle reti di Petri in generale). Per queste ragioni, dopo aver descritto brevemente ed informalmente le reti di Petri e le possibilità d'analisi offerte, si devono ora definire le regole di traduzione che consentono di derivare automaticamente il modello formale (rete di Petri) dalla specifica informale (diagramma FBD). L'automazione del processo di traduzione (descritto in [3]), consente all'utente di continuare a lavorare in maniera tradizionale, impiegando gli stessi strumenti di specifica e gli stessi linguaggi. In più, il modello formale consente la simulazione e l'analisi delle specifiche definite. Affinché l'approccio sia "completo", i risultati della validazione (simulazione o analisi) del modello formale devono essere tradotti in una forma comprensibile agli utenti, esperti del dominio applicativo, ma digiuni di metodi formali. Un meccanismo di traduzione completa nei due sensi (dai blocchi FBD alle reti di Petri e viceversa) consentirebbe agli utenti di godere dei benefici dei metodi formali, restandone completamente ignari.

Seguendo l'approccio a regole descritto in [3] è possibile creare un vero e proprio archivio di regole di traduzione da utilizzarsi per definire il modello formale che corrisponde ai diversi blocchi FBD. Ad esempio, ricordando la semantica delle reti di Petri descritta in precedenza, è possibile definire la corrispondenza illustrata in Figura 5. Si supponga di avere un blocco PLC caratterizzato da tre ingressi e due uscite. L'ipotesi che l'esecuzione inizi solo quando sono presenti tutti gli ingressi consente di rappresentare un singolo blocco attraverso una transizione con tre posti in ingresso. Il codice associato al blocco PLC è trasformato nell'azione della transizione. L'ipotesi, poi, che le uscite siano prodotte contemporaneamente permette di collegare direttamente alla transizione i due posti che corrispondono alle uscite del blocco.

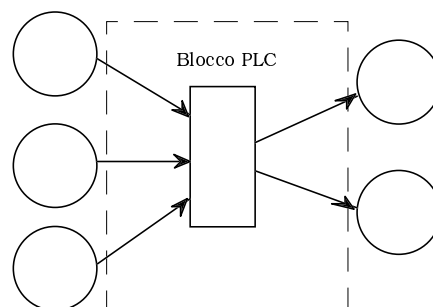


Figura 5: Mappatura di un blocco PLC

L'intera specifica PLC si ottiene unendo le sottoreti di Petri simili a quella di Figura 5. La connessione fra i diversi blocchi avviene "fondendo" i posti in uscita da un blocco con quelli in ingresso al blocco successivo.

Ponendosi ad una granularità inferiore, sarebbe possibile dettagliare ulteriormente la rappresentazione formale (rete di Petri). Le funzionalità di un blocco potrebbero essere descritte da sottoreti

di Petri composte da più transizioni, evidenziando le “sottoazioni” principali svolte dal blocco PLC. Un maggior livello di dettaglio fornisce un modello formale più rigoroso, ma comporta anche un notevole aumento delle dimensioni della reti di Petri corrispondente. La maggiore analizzabilità ha come effetto negativo l’aumento della complessità della specifica prodotta.

L’effetto immediato della tecnica descritta in questo paragrafo è la possibilità di sfruttare tutti i benefici offerti dalle reti di Petri (temporizzate) senza conoscerle e senza dover modificare il proprio modo di lavorare, sia in termini di metodo di progettazione, che di notazioni usate.

5. LA SIMULAZIONE

Un modello formale - eseguibile ed analizzabile - del controllore software consentirebbe di valutare la validità ed il soddisfacimento dei requisiti (temporali) richiesti prima della codifica e della validazione sul target. D'altra parte, per una verifica significativa del software di controllo è necessario produrre dati che siano logicamente, e matematicamente, compatibili con la chiusura dell’anello di regolazione. Questo implica un ambiente di simulazione in grado, da una parte, di gestire modelli tempo discreti e, dall’altra, modelli tempo continui caratterizzati da equazioni differenziali, magari non lineari e tempo varianti [5-18].

Gli ambienti di simulazione tradizionali sono carenti nelle potenzialità di rappresentazione dei componenti discreti. Si limitano a fornire la possibilità di descrivere parti del sistema in termini di trasformata Z. La soluzione ideale, invece, dovrebbe consentire di:

- integrare equazioni differenziali, anche complesse, che descrivono il sistema controllato.
- gestire modelli del sistema tempo discreto - il controllore - utilizzando metodi formali quali le reti di Petri.

Una soluzione di questo tipo può essere realizzata introducendo nuove funzionalità all’interno dell’ambiente Matlab/Simulink della Mathworks. Questo strumento, grazie anche all’interfaccia grafica, è particolarmente adatto per la descrizione dei sistemi da controllare (componenti tempo continui), ma carente per quanto riguarda la definizione dei componenti tempo discreti. La sua struttura di ambiente aperto consente, però, l’aggiunta di nuovi elementi all’interno dei componenti standard. Interfacciando Matlab/Simulink ed un esecutore di reti di Petri è possibile attuare la seguente logica di simulazione:

- Ad ogni istante di integrazione del modello tempo continuo, si risolvono le equazioni differenziali del modello utilizzando l’algoritmo scelto dall’utente.
- Ad ogni istante di campionamento delle variabili tempo discrete, si campionano i valori e li si inviano all’esecutore di reti di Petri.
- I risultati ottenuti dall’esecutore di reti di Petri sono considerati validi per tutto il periodo di campionamento successivo.

Una logica di questo tipo è già stata realizzata per l’esecuzione tempo discreta di procedure di controllo scritte in linguaggio C o linguaggio di Matlab. La realizzazione della connessione Simulink - esecutore di reti di Petri è attualmente in corso [6]. Questa attività è parte integrante del progetto CEE ESPRIT denominato INFORMA, descritto al paragrafo seguente.

6. INFORMA

Le finalità del progetto INFORMA [9] mirano a colmare le lacune evidenziate nei paragrafi precedenti. Il consorzio INFORMA è costituito da: Ansaldo Industria, Ansaldo Ricerche, IFAD (Danimarca), Odense Steel Shipyard (Danimarca), Politecnico di Milano - Dipartimento di Elettronica e Informazione e Dipartimento di Elettrotecnica e Scientific Software Group (Francia)

INFORMA si propone di fornire un supporto multilinguaggio in grado di assistere il progettista nelle diverse fasi del progetto: dalla definizione dei requisiti fino alla codifica, consentendo ad

ogni livello verifiche accurate. Gli elementi fondamentali dell'ambiente INFORMA sono schematizzati in Figura 6:

Matlab/Simulink viene utilizzato per la stesura delle specifiche del sistema e l'analisi e la simulazione dei componenti tempo continui. Matlab/Simulink costituisce anche l'infrastruttura per l'integrazione dei vari componenti: tutti gli altri elementi dell'ambiente possono essere visti come aggiunte a Matlab/Simulink.

Editor ISO-PLC viene utilizzato per il progetto dei componenti *hard real time*. Lo standard ISO-PLC costituisce, di fatto, un tipico linguaggio per la descrizione delle logiche di controllo per sistemi d'automazione ed in particolare per i task *hard real time* (chiusura degli anelli).

Traduttore ISO-PLC-HLTPN viene utilizzato per definire la semantica degli oggetti ISO-PLC in termini di reti di Petri d'alto livello temporizzate (HLTPN: High Level Timed Petri Nets).

HLTPN Kernel viene utilizzato per l'esecuzione ed il *debugging* simbolico delle reti di Petri È, quindi, la base per l'animazione (esecuzione/validazione) delle specifiche ISO-PLC.

VDM⁺⁺ viene utilizzato per la progettazione dei componenti *soft real time*. Il VDM⁺⁺ [15] è un linguaggio formale di specifica orientato agli oggetti particolarmente adeguato per la definizione di tutti i componenti con vincoli temporali non stringenti. Ad esempio, l'interfaccia utente.

Generatore di codice C e C++ viene utilizzato per la generazione automatica di codice C a partire dalle specifiche ISO-PLC e codice C++ dai componenti VDM⁺⁺.

L'integrazione di diversi formalismi permette, anche, di sfruttare le caratteristiche degli strumenti che compongono l'ambiente INFORMA: Matlab/Simulink per quanto riguarda l'interfaccia utente e le capacità grafiche e matematiche; VDM⁺⁺ Toolbox² per la definizione e l'esecuzione delle specifiche VDM⁺⁺; Cabernet³ per la gestione delle reti di Petri.

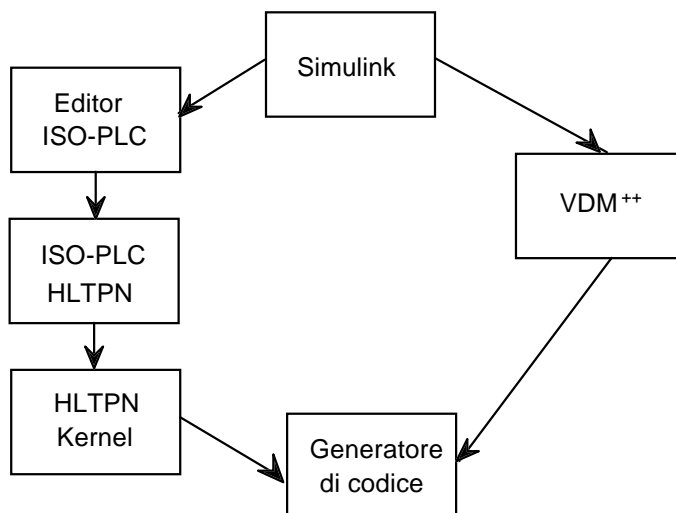


Figura 6: L'architettura di INFORMA

L'ambiente INFORMA si presenta come un sistema multiformale in grado di coprire il ciclo di vita del software per applicazioni di controllo in tempo reale, in cui è possibile:

- simulare completamente il sistema integrato (controllato - controllore).
- specificare il software con vincoli temporali *hard real time* usando il formalismo ISO-PLC.

² Ambiente di supporto allo sviluppo di specifiche VDM⁺⁺ realizzato dall'IFAD.

³ Ambiente di supporto allo definizione e esecuzione ed analisi di reti di Petri d'alto livello realizzato presso il Dipartimento di Elettronica e Informazione del Politecnico di Milano.

- definire i componenti con requisiti temporali non stretti (*soft real time*) attraverso il VDM⁺⁺.
- generare automaticamente il codice a partire dalla specifica eterogenea.

L'organizzazione di Figura 6 suggerisce, anche, un "ipotetico" processo di sviluppo, che sfrutti completamente le possibilità offerte dall'ambiente integrato:

1. Definizione del modello del sistema controllato, o *embedding*, per mezzo di blocchi Simulink. Le caratteristiche di questo strumento di simulazione consentono la creazione di una libreria proprietaria per le diverse applicazioni (si consideri, come esempio, l'ambiente AZIO, sviluppato presso il Dipartimento di Elettrotecnica del Politecnico di Milano [8]).
2. Individuazione dell'architettura del controllo, identificando l'insieme dei task *hard real time* (vincoli temporali stringenti) e l'insieme dei task *soft real time*. Questa suddivisione facilita lo sviluppo, in parallelo, dei due insiemi di task, utilizzando i diversi formalismi di specifica.
3. Progettazione della specifica PLC, per la descrizione delle parti del controllore critiche rispetto al tempo. L'ambiente INFORMA garantisce la traduzione automatica verso una rete di Petri avanzata e, quindi, l'eseguibilità della specifica PLC. Inoltre, il modello PLC è integrato, nell'ambiente INFORMA, nella simulazione con Matlab/Simulink per garantire il dialogo diretto tra controllo e impianto.
4. Sviluppo della specifica VDM⁺⁺ per i componenti *soft real time*. Anche in questo caso, INFORMA integra, attraverso un opportuno blocco Simulink, l'interprete VDM⁺⁺ per consentire un dialogo diretto con i dati provenienti dal sistema simulato.
5. Simulazione del sistema "completo". È possibile osservare, sulla scala del tempo simulato, le interazioni tra sistema di controllo e sistema controllato.
6. Codifica automatica delle specifiche prodotte e, quindi, verifica sul sistema *target*.

Dall'utilizzo dell'ambiente INFORMA, l'esperto di controllo dovrebbe, quindi, trarre i seguenti benefici:

- La disponibilità di un ambiente integrato di sviluppo per le diverse fasi del progetto. L'uniformità dell'interfaccia utente e la standardizzazione delle procedure di progettazione facilitano l'esperto di controllo nel passaggio attraverso le diverse fasi del ciclo di vita del prodotto controllato.
- Una nuova accezione del concetto di simulazione di sistema. Si mira a fornire modelli, quanto più realistici possibile, del sistema digitale di controllo, consentendo di effettuare verifiche non solo algoritmiche, ma anche temporali grazie all'utilizzo delle reti di Petri d'alto livello.
- Un elevato livello di automazione delle diverse fasi del processo di progettazione ed, in particolare, della fase di codifica.
- L'adeguamento a standard internazionali riconosciuti: il linguaggio *Functional Block Diagram* per quanto riguarda il PLC ed il linguaggio VDM per il VDM⁺⁺, ed a standard "de facto", come Matlab/Simulink, che è molto diffuso in ambito industriale.

7. CONCLUSIONI

L'articolo ha descritto i problemi e le aspettative esistenti per quanto riguarda la progettazione del software di controllo nel campo dell'automazione. Come soluzione ai problemi tuttora irrisolti, si è descritta la metodologia proposta nell'ambito del progetto CEE ESPRIT chiamato INFORMA. Si sono illustrate le finalità del progetto e la sua possibile collocazione in ambito industriale.

I punti di forza della metodologia possono essere riassunti in:

- Capacità, grazie alla corrispondenza tra FBD e reti di Petri, di "mascherare" i metodi formali agli utenti finali, consentendo loro di goderne i benefici senza avere alcuna conoscenza specifica.

- Possibilità di usare formalismi diversi e particolari per la definizione dei requisiti di componenti diversi: *hard* e *soft real-time*.
- Modularità dell'ambiente proposto in INFORMA per poter utilizzare - eventualmente - altre notazioni di progetto o altri metodi formali operazionali.

In conclusione, in questo articolo la metodologia è stata applicata alla definizione di azionamenti elettrici, ma l'approccio è da ritenersi valido per la progettazione di sistemi ibridi in generale.

BIBLIOGRAFIA

- [1] R. Alur, T.A. Henzinger e E.D. Sontag, "*Hybrid Systems III*". Volume 1066 di *Lecture Notes in Computer Science*. Springer-Verlag, 1996.
- [2] P.J. Antsaklis, "*Hybrid Systems II*". Volume 999 di *Lecture Notes in Computer Science*. Springer-Verlag, 1995.
- [3] L. Baresi, "*Personalizzazione Formale di Notazioni Grafiche*". Tesi di Dottorato. Dipartimento di Elettronica e Informazione - Politecnico di Milano, Gennaio 1997.
- [4] L. Baresi, A. Orso e M. Pezzè. "*Introducing Formal Methods in Industrial Practice*". Proceedings of the 19th International Conference on Software Engineering, IEEE-CS Press. Maggio 1997.
- [5] M. Carpita, A. Monti. "*Voltage Source Converters and Drives Simulation at System Level for Control design Applications*". EPE Journal, vol. 5 n. 3 e 4, pag. 49-55. Gennaio 1996.
- [6] F. Castelli Dezza, A. Monti. "*Utilizzo di Matlab/Simulink come Sistema di Specifica per Controlli Digitali: Teoria ed Esempi Applicativi*". I Conferenza Italiana Utenti Matlab - Bologna. Novembre 1995.
- [7] F. Castelli Dezza, E. Chiesa, A. Monti e M. Riva. "*A New Interactive Matlab Toolbox for Teaching and Testing Electrical Drives*". Verrà presentato a EPE97 - Trodheim (Norvegia). Settembre 97
- [8] E.M. Clarke e J.M. Wing, "*Formal Methods: State of the Art and Future Directions*". Rapporto Tecnico CMU-CS-96-178. Carnegie Mellon University, Settembre 1996.
- [9] Consorzio INFORMA. *Technical Annex*
- [10] J. Crow e B.L. de Vito, "*Formalizing Space Shuttle Software Requirements*". First Workshop on Formal Methods in Software Practice, ACM, Gennaio 1996.
- [11] K. Finney, "*Mathematical Notation in Formal Specification: Too Difficult for the Masses?*". IEEE Transactions on Software Engineering. vol. 9, n. 6, pag. 733-744. Giugno 1996.
- [12] R. France e M. Larrondo-Petrie, "*From Structured Analysis to Formal Specifications: State of the Theory*". Proceedings of the ACM Computer Science Conference, pag. 249-256. Marzo 1994.
- [13] C. Ghezzi, D. Mandrioli, S. Morasca, M. Pezzè, "*A Unified High-level Petri Net Formalism for Time Critical System*". IEEE Transactions on Software Engineering. vol. 17, n. 2, pag. 160-172. Febbraio 1991.
- [14] R.L. Grossman, A. Nerode, A.P. Ravn e H. Rischel, "*Hybrid Systems*". Volume 736 di *Lecture Notes in Computer Science*. Springer-Verlag, 1993.
- [15] IFAD, "*VDM++ Toolbox: User Manual*", 1996
- [16] C. Kronlöf, "*Method Integration - Concepts and Case Studies*". John Wiley & Sons, 1993.
- [17] A. Monti. "*Circuits as Concurrent Systems: Through a New Approach to the Simulation*". IEEE ISCAS 96 - Atlanta (USA).
- [18] A. Monti, F. Noli, "*Towards a Flexible Environment for Developing and Testing Software-based electrical Drives*". SPEEDAM 94, Taormina (Italy)

- [19] M. Matuonto, E. Mazzocut, A. Monti, “*Graphic Software Designers: a Quick Way to Develop and Standardise Electrical Drives Digital Control*”. SPEEDAM 94, Taormina (Italy)
- [20] T. Murata, “*Petri Nets: Properties, Analysis, and Applications*”. Proceedings of the IEEE, vol. 77, n. 4, pag. 541-580. Aprile 1989.