

Informatica Industriale

Parte prima: cap 1-8 dispense

1. Introduzione

1.1 Computer Integrated Manufacturing

CIM: si coordinano e si gestiscono le attività di manifattura industriale tramite sistemi informatizzati. Non sarà mai possibile integrare totalmente tutti gli aspetti della produzione (fattori umani, certe operazioni non sono riconducibili a strutture informatiche).

Il CIM è una trama che correla gestione di informazioni (Basi di Dati), comunicazione tra calcolatori (networking) e l'automazione delle macchine di produzione.

Suddiviso in livelli gerarchici (ISO TC184). Generalmente il livello 0 (fisico) è poco significativo. Ogni livello corrisponde ad una serie di nodi (elaboratori) che coordinano i nodi gerarchicamente inferiori, fornendo report informativi (con diverse granularità di informazione) ai nodi di livello gerarchicamente superiore.

Scendendo di livello gli orizzonti temporali sono sempre più brevi e le esigenze real-time sono sempre più stringenti. Le informazioni sono più dettagliate nei livelli inferiori, mentre nei livelli più alti si utilizzano aggregazioni di dati in modo da generare informazioni di tipo consuntivo adatte alla gestione dell'impianto. Il corso si concentra sui livelli 1-2-3

1.2 Sistemi di automazione

Obiettivo: fare in modo che i processi fisici evolvano secondo le nostre necessità in modo efficiente. Si interviene gestendo il processo fisico tramite calcolatori.

Il **Processo Industriale** è un complesso di trasformazioni di materia ed energia, che produce e consuma informazioni ed è soggetto a disturbi. I **disturbi** includono fenomeni che interferiscono indesideratamente con il processo (modificano le trasformazioni in modo non voluto). Sono classificate anche come disturbi le semplificazioni del modello che applichiamo per renderlo più semplice (il modello non aderisce perfettamente con la realtà). In realtà i fenomeni disturbanti sono tali perché non sono codificati nel modello. Se lo fossero non sarebbero disturbi (bisogna raffinare il modello). Si fa un tradeoff tra la semplicità del calcolo (legata ad un modello semplice) ed il controllo ottenibile sul processo (più raffinato con modelli più complessi che tengono conto di un maggior numero di variabili).

Per far evolvere le trasformazioni nella direzione desiderata, si fanno interagire le informazioni scambiate con un **controllore**. Il controllore interagisce anche con collaboratori e supervisori tramite un ulteriore scambio di informazioni.

1.3 Schema Funzionale del sistema d'automazione

Il sistema d'automazione viene visto da una prospettiva funzionale, con blocchi funzionali che producono / utilizzano i dati in schemi di interazione (modello *DataFlow*). Lo schema presenta numerosi anelli chiusi, oltre al classico anello feedback di regolazione, e coinvolgono l'operatore umano (comandi di supervisione).

Ho un solo anello critico per i tempi di reazione, ed è quello del controllo.

Le informazioni acquisite possono essere trasmesse in modo diverso a seconda di chi le riceve

(tipicamente un operatore umano legge i dati ad una frequenza molto inferiore rispetto al controllore che regola il processo, e si accontenta di dati consuntivi non necessariamente precisi, diversamente dal controllore).

L'operazione di *monitoraggio* fa parte della supervisione. Grazie alla funzione di *interpretazione* si estraggono, dai singoli dati, informazioni consuntive adatte all'operatore o all'utilizzo da parte di altri calcolatori (che generalmente non hanno bisogno di un livello di dettaglio eccessivamente elevato). Il modulo di *gestione operatore* fornisce un'interfaccia comoda e adatta all'uomo. Similmente il modulo di *comunicazione* con gli altri calcolatori gestisce i protocolli di scambio messaggi con le altre macchine. Il blocco *comandi* trasforma i "comandi consuntivi" in informazioni più adatte al controllo. Questi dati vengono infine tradotti nell'attuazione vera e propria dal blocco *interventi*.

2. I sistemi industriali

2.1 Acquisizione dei dati

Raccolta di dati al fine di analisi ed elaborazione dei dati per controllare i processi. Oppure verificare ipotesi sul modello, collezionare andamenti temporali per analisi future non ancora ben definite. Bisogna tenere conto della precisione (scostamenti dal valore reale) che vogliamo ottenere, della frequenza di campionamento (in particolar modo riguardo alle applicazioni real-time, non ci si basa semplicemente sul teorema di campionamento), della necessità di filtri (che influenzano le tecniche di acquisizione), delle modalità di presentazione dei dati acquisiti. Se ho usi diversi, devo effettuare un disaccoppiamento tra lo stadio di acquisizione (aggiornare l'immagine dell'insieme di variabili, tenendo conto della realtà fisica) e lo stadio di elaborazione (controllare l'accesso all'immagine dati di variabili globali). Questo mi aiuta anche nel collaudo della mia applicazione, in quanto separo la logica di regolazione dalla logica di acquisizione, e posso simulare il mondo fisico con un programma, con grande beneficio sulle modalità di debugging (riproduzione esatta di una seduta di prova che producono anomalie). Sezione di disaccoppiamento è composta principalmente da 2 componenti:

- Immagine globale degli stati: insieme di variabili che rappresentano la realtà fisica
- Code di eventi: situazioni istantanee nel tempo, che vanno accodati

2.2 Monitoraggio

Verifiche di alto livello in base ai dati acquisiti. Correlazioni nello spazio/tempo delle varie grandezze. Queste elaborazioni non interessano comandi diretti ma forniscono informazioni ad altri responsabili di reagire alla particolare situazione generando azioni correttive. Occorre avere una buona conoscenza del modello dei fenomeni, in modo da distinguere situazioni di pericolo dall'immagine di stato corrente.

2.3 Controllo e regolazione

Produzione di comandi che hanno effetto sui processi fisici (funzioni attive), che comandano attuatori. Regolazione (controllo modulante); Controllo (controllo logico). A differenza di altri oggetti, il calcolatore non può realizzare fisicamente il controllo continuo perchè non segue leggi fisiche intrinseche per effettuare il controllo. Bisogna invece virtualizzare il continuo, rendendolo comprensibile anche per un calcolatore, ripetendo ciclicamente gli algoritmi di calcolo su immagini aggiornate degli stati.

2.4 Gestione allarmi

E'una funzione legata al monitoraggio. Segnalazione di anomalie: un'anomalia è il valore di verità di una formula logica che rileva una particolare condizione dell'impianto. Un allarme è uno stato associato alla storia sequenziale di un'anomalia. Il suo scopo è la segnalazione di anomalie al supervisore, producendo azioni di blocco e generando eventi da memorizzare.

Per rappresentare le situazioni di allarme si usano automi a stati (ISA-1). Possono esserci azioni di segnalazione o anche di bloccaggio nelle situazioni più critiche (quelle che possono generare un disastro). In genere il blocco automatico genererebbe situazioni più gravi di quanto l'anomalia stessa potrebbe produrre, per cui ci si limita alla segnalazione. L'operatore umano infatti ha a disposizione un insieme di informazioni di contesto per cui può stabilire se il blocco va effettuato immediatamente oppure in modalità differita, pur tenendo conto della gravità o meno dell'anomalia. In molti casi le anomalie in sequenza sono dovute ad effetti a catena di un'unico evento scatenante. E'importante quindi in questi casi capire qual'è stata la prima anomalia ad essersi verificata (per scoprire il guasto).

E'importante segnalare anche la fase di rientro dall'anomalia, in modo da avvisare che l'impianto sarà presto rimesso in funzione.

2.5 Visualizzazioni

Un sistema d'automazione presidiato dovrà presentare agli operatori le informazioni relative all'impianto. Bisogna visualizzare le informazioni nel modo più efficace possibile (precisione, chiarezza, correttezza, ergonomia). E'importante la conoscenza delle modalità di percezione psicofisica dell'uomo, così come evidenziare le correlazioni tra fenomeni (affiancando le indicazioni).

2.6 Memorizzazione Storica

Tenere uno storico degli eventi significativi, con indicazioni cronologiche. Ogni tanto si memorizza la situazione dell'impianto (log). Si possono memorizzare dati precisi oppure informazioni consuntive. Bisogna effettuare compromessi a seconda della destinazione operativa dei dati che memorizziamo. Fare attenzione alla perdita di informazione in memorie volatili (che potrebbero andare perdute prima dell'effettiva memorizzazione storica)

2.7 Stampa di rapporti

Funzione obsoleta. Utile laddove non sono disponibili altre modalità di consultazione immediata dei dati (no video)

2.8 Comunicazione con i livelli superiori

Complementano o sostituiscono le funzioni di memorizzazione. Difatti spesso i calcolatori di controllo non hanno memorie di massa (tipicamente sono microcontrollori).

2.9 Interfaccia uomo-macchina

Bisogna evidenziare gli ruoli umani relativi al progetto, con le rispettive attitudini: Committente, Progettista, Costruttore, Installatore, Supervisore, Conduttore, Manutentore

Nelle applicazioni industriali abbiamo una maggior semplicità delle interfacce, ma con esigenze superiori di efficacia, tempistica e sicurezza (rispetto ad esempio ad interfacce di applicativi gestionali). Grande importanza riveste la percezione sensoriale degli eventi in corso nell'impianto (occorre quindi evidenziare quelli più importanti e renderli facilmente e velocemente assimilabili,

nonchè evidenziare le correlazioni tra i vari eventi). Eco dei comandi: reazione del sistema (audio-visiva o di altro genere) ad una mia azione sul sistema stesso. Bisogna vedere l'effetto del comando, anche a livello di interfaccia, che ho dato per essere sicuro che sia stato recepito dal sistema ed attuato, e bisogna fare in modo che l'effetto non sia eccessivamente dilazionato ("comando recepito" è diverso da "comando eseguito"). Riferimenti spaziali: l'uomo si forma una topologia di tutte le cose che incontra, in modo da non ri-verificare ogni volta le informazioni che abbiamo percepito, sempre che la collocazione spaziale sia rispettata nel tempo. Devo rispettare la topologia dell'impianto nella collocazione delle informazioni sul video. Visualizzazione delle alternative: menù e scelte multiple che ricordano all'utente che cosa può fare. Propensione agli errori dell'uomo (non sa, non ricorda, sbaglia) e del programma (sbaglia perchè il suo comportamento si discosta dalle specifiche), segnalazione di errori avvenuti nell'impianto o nel calcolatore. Il sistema deve scoprire gli errori (verificando comunque in automatico i comandi dati dagli operatori e vedere se sono palesemente errati) e segnalarli tempestivamente ed efficacemente (chi legge la segnalazione capisce che errore è avvenuto e può porvi rimedio: il programma può anche presentare le scelte che si potevano fare, e aiutare l'operatore ad uscire dalla condizione di errore segnalando la via d'uscita). Separare tra scelta e conferma: l'effetto dei comandi, quando possibile, non deve essere irreversibile. Visualizzo la scelta effettuata prima di fornire l'opportunità di conferma (posso riflettere e verificare sulla scelta che ho fatto, magari aiutato dal sistema che può prefigurare l'evoluzione prevista del sistema se attuo quel comando). Livelli di astrazione: dominare le situazioni complesse, scegliere con attenzione i parametri da evidenziare ai vari livelli. Rendere omogenee (come tipo di astrazione) le entità presentate in una unità di presentazione. Utilizzare colori, lampeggi e acustica per arricchire e connotare in modo evidente le informazioni, utilizzandoli però in modo da non ottenere l'effetto opposto a quello cercato (seguendo le norme e le convenzioni già adottate e a cui gli operatori sono abituati). Per i lampeggi sono importanti frequenza e duty cycle (probabilità di vedere il segnale con una vista occasionale, differenziare informazioni differenti), fase (di più entità lampeggianti: fase = aggregazione, controfase = separazione). Orientamento al problema: tutte le interfacce possono essere orientate al problema oppure alla struttura realizzativa (la struttura che supporta la soluzione del problema, fanno parte di essa ma non appartengono al dominio applicativo del problema). Ricerca di standardizzazione e coerenza in tutti i contesti della nostra applicazione. Le **unità di I/O** per realizzare le interfacce sono eterogenee e dipendono dall'applicazione considerata.

2.1 Requisiti di elaborazione nell'informatica industriale

Modalità di esecuzione dei programmi

1. Esecuzione batch: interazione poco interattiva. L'utente fa sapere al centro di calcolo programma e dati da eseguire (via rete). Si massimizza il *throughput* (eseguo i programmi che in quel momento hanno bisogno delle risorse e sono disponibili a sfruttarle al meglio).
2. Esecuzione on-line: interattiva. L'utente ed il programma interagiscono tra di loro. Il tempo di risposta è interessante ma non è un vincolo rigido, posso tollerare ritardi nella comunicazione dovuti a sovraccarichi momentanei.
3. Esecuzione real-time: il programma interagisce con le macchine e deve adeguarsi a vincoli temporali governati dalla fisica del processo. Il processo fisico evolve con determinati vincoli, il programma deve adeguarsi ai tempi del processo e deve calibrarsi precisamente su questi vincoli. Mi interessa la predicibilità dell'andamento temporale del sistema di elaborazione.

Integrazione HW/SW

C'è una ricchezza di dispositivi HW e di interazioni possibili con il SW nei vari contesti. I programmi che gestiscono le interazioni devono integrarsi con la scelta fatta nei dispositivi hw

presenti nell'impianto, e devono prevedere variabilità nelle soluzioni adottate (adattabilità). Mancanza di standardizzazione nelle applicazioni industriali, non è detto nemmeno che esista la barriera di visualizzazione rappresentata dal Sistema Operativo, si usano sistemi dedicati HW/SW. Alcune operazioni infatti, normalmente gestibili con un sistema operativo, non hanno requisiti di efficienza sufficienti come quelli richiesti dalle applicazioni.

Classi di configurazioni

1. Sistemi embedded
2. Sistemi di controllo macchine
3. Sistemi di automazione di impianti

Interfacce specifiche

Esigenze variabili (**velocità** di trasferimento, **precisione** delle misure, **insensibilità** ai disturbi, **resistenza** alle condizioni ambientali avverse, **robustezza** e **tolleranza** ai guasti, **sicurezza** intrinseca per ambienti con pericoli particolari legati all'elettronica utilizzata) portano alla costruzione di interfacce specifiche per ogni tipo di applicazione.

Cpu

Quelli adatti per queste applicazioni privilegiano la **predicibilità** dei tempi d'esecuzione e la **semplicità** per le operazioni orientate ai bit e agli interi. Si rinuncia alle caratteristiche che migliorano la velocità media per privilegiare le prestazioni nel caso pessimo (a differenza delle CPU general purpose). Devo garantire prestazioni istantanee sufficienti all'elaborazione dei dati che mi arrivano in continuo, preferisco una CPU con prestazioni più basse ma con worst case garantito in un minimo accettabile (per quanto mediamente questi processori siano mediamente meno performanti di quelli general purpose).

Memorie

Le memorie di lavoro hanno capacità ridotte, e sono organizzate in aree distinte per capacità e tecnologia.

1. **ROM / EPROM:** memoria non volatile a sola lettura. Programmi/costanti che devono essere immediatamente disponibili all'accensione del sistema e che non devono essere modificati. Può contenere anche i valori di inizializzazione delle variabili (in questo caso deve esserci un modulo che carica l'inizializzazione delle variabili dalla memoria non volatile alla cella allocata su RAM della variabile considerata per inizializzarla).
2. **RAM:** memoria volatile, va inizializzata (specialmente per le variabili che corrispondono a misure fisiche, non posso lasciare il valore casuale senza inizializzare effettivamente ad un valore accettabile).
3. **RAM non volatili:** per le informazioni modificabili a runtime ma che non devono essere volatili (coefficienti, parametri, valori di taratura...). Flash, EEPROM, RAM Tampone (in batteria). Bisogna progettare il controllo di accesso a memoria, per assicurarsi che non sia facile sovrascrivere queste aree di memoria (a causa di errori di programma oppure di esecuzione in condizioni perturbate)
4. **Memory mapped I/O:** si associano indirizzi di memoria a registri delle porte di I/O in maniera trasparente. Se una porta è collocata in area di memoria, vi posso accedere con tutte le istruzioni possibili nell'area di memoria. Per contro i bit di indirizzamento dell'area di memoria sono 20-24bit, devo decodificare una grande quantità di bit di indirizzamento che sono inutili perchè è difficile arrivare a un migliaio di porte di I/O (basterebbero 10-12 bit).

Strumenti informatici e SW di base

Gli strumenti tendono a nascondere i dettagli “incapsulamento opaco”. Eterogeneità dei linguaggi di programmazione, standard poco diffusi (IEC-1131-3). Si cerca di evitare il ricorso all'assembler. In ogni caso il dettaglio deve trasparire (perchè devo comunque gestire i bit), ma non voglio arrivare alla complessità dell'assembler che espone anche i dettagli implementativi della CPU, ad esempio), ma solo i dettagli significativi vengono fatti trasparire.

Peculiarità delle architetture costruttive

Bisogna facilitare l'accesso, l'ispezione e la sostituzione dei componenti, perchè tipicamente i calcolatori industriali sono mantenuti da personale anche non familiare con un computer. Organizzazione più razionale della struttura dei componenti del calcolatore.

Collaudo e messa a punto

E'difficile attribuire le responsabilità dei malfunzionamenti perchè ci sono un sacco di fattori da considerare in un impianto industriale, non legati solo al software ed ai calcolatori ma anche ai componenti stessi dell'impianto. Inoltre se non siamo in simulazione ma stiamo facendo un collaudo effettivo dell'impianto ci sono anche i fattori legati alla preoccupazione di danni all'impianto (ad esempio non è facile provare gli allarmi, mandando volontariamente l'impianto in una condizione critica per vedere se l'allarme è riconosciuto). Importanza dei sistemi di simulazione anche durante il ciclo di vita dell'impianto, per testare le modifiche e gli aggiornamenti prima di implementarle effettivamente.

Tempo Reale

I sistemi real-time sono adatti a reagire con fenomeni asincroni. Nei sistemi industriali si eseguono elaborazioni in tempo reale, ovvero reagire a fenomeni esterni con risultati corretti, rispettando le costanti temporali che vincolano l'elaborazione. Il tempo reale va valutato rispetto ai vincoli del processo: i risultati devono “arrivare in tempo” perchè il programma si dica in tempo reale.

Hard real-time: tempo reale stretto, i fallimenti sono *catastrofici*.

Soft real-time: tempo reale lasco, i fallimenti sono *degradanti*.

La correttezza dei sistemi deve essere logica (risultati corretti) e temporale (risultati arrivano in tempo utile).

Guasti

Un sistema che non produce dati errati in presenza di guasto si dice **integro** (proprietà di **integrità** del sistema). In certe situazioni sono disposto ad accettare valori inesatti e/o errati, ma non voglio che il sistema di controllo si fermi (proprietà di **persistenza** del sistema). Questo perchè certi fenomeni fisici in assenza di controllo possono evolvere verso stati pericolosi, per cui è meglio comunque proseguire in condizioni di efficienza ridotta causate dal guasto. I calcolatori **integrati** e **persistenti** sono detti **affidabili**. In ogni caso le tipologie di impianti possono far privilegiare un aspetto su un altro.

1. Impianti continui (che generalmente si modellano con filtri passa basso), in genere sono reversibili se rimangono in condizioni di sicurezza. Valori occasionali errati (che rientrano rapidamente) non portano a situazioni critiche per cui sono tollerabili. Si privilegia la persistenza, e l'assenza di controllo può essere tollerata indefinitamente se l'impianto ha un comportamento che lo porta autonomamente ad una situazione sicura (es. Un motore che si ferma – ovviamente dipende da che cosa muoveva!).
2. Impianti discreti invece sono caratterizzati da fasi a bassissima reversibilità (scaricamento di silos, linee di montaggio). E'sufficiente un solo comando errato per provocare un disastro. Si privilegia l'integrità.

Obiettivi di un controllore affidabile

- Safety: Prevenire danneggiamenti a cose e persone
- Availability: ripristinare prontamente il funzionamento nominale
- Reliability: massimizzare la disponibilità nel lungo periodo.

Tutte queste proprietà sono strettamente collegate alla tipologia del processo considerato.

3. Caratterizzazioni di tempo ed evoluzione

3.1 Caratterizzazione degli errori

Distinguo errore vero e proprio (scostamenti di ampiezza arbitraria) da approssimazioni (scostamenti in intorno dei valori corretti).

Errori formali: mancato rispetto delle regole sintattiche del linguaggio usato (rendono inutilizzabile il programma).

Errori di comportamento: possono essere o meno accettabili a seconda della loro ampiezza.

- **Errore di valore**: scostamenti del valore di una informazione dal valore vero. Possono essere introdotti in diverse fasi (**acquisizione**, dovuti ad un comportamento non nominale del dispositivo di misurazione; **rappresentazione**, errori di quantizzazione, non linearità, errori di guadagno; **elaborazione**, limitatezza dei dispositivi di calcolo e uso di approssimazioni, overflow; **emissione**). Gli errori di valore si sommano, per cui è inutile avere dispositivi iper precisi quando poi trascuri la correzione di errori negli altri dispositivi della catena. Devo avere dei dispositivi omogenei riguardo alla precisione.
- **Errore temporale**: l'intervallo di tempo tra la richiesta di un servizio e la sua effettiva esecuzione (problemi real-time). A volte si manifestano come errori di valore.

Accanto ai valori nominali devo sempre indicare le fasce di tolleranza sulla misura! Gli scostamenti, a parità di valore, sono da considerarsi accettabili o meno a seconda del campo applicativo in cui stiamo valutando questi aspetti.

3.1.1 Errori di misura

Fare una misura: trovare il modo di conoscere un'informazione numerica che rappresenta un'entità fisica. **V**: valore vero che si esprimerebbe in assenza di errori. **M**: valore misurato, quello che abbiamo ottenuto effettivamente. M è soggetto a scostamenti da V solamente a causa della tecnica di misura utilizzata.

- Errore assoluto $E_A = M - V$
- Errore relativo $E_R = (M-V)/V = E_A/M$

L'errore relativo è adimensionale, spesso viene espresso in percentuale. Gli errori relativi sono indipendenti dall'entità della misurazione. Quello che importa è il rapporto tra lo scostamento assoluto e la misura effettuata, per valutare l'entità dell'impatto dell'errore sulla misura. L'errore assoluto non mi permette di valutare la reale entità dello scostamento. Lo scostamento assoluto inoltre è legato alle unità di misura, cosa che non avviene valutando l'errore relativo.

In certe situazioni invece sono interessanti solamente gli errori assoluti. Alcune grandezze come il tempo, lo spazio e l'energia assoluti hanno un'origine arbitraria. Fissando alcuni obiettivi, si vuole che sia l'errore assoluto su cui do le specifiche, mentre non ho molto vantaggio a considerare

l'errore relativo. Stiamo parlando di punti, non di intervalli (per i quali resta interessante l'errore relativo perchè l'origine è implicitamente ben fissata).

3.1.2 Distribuzione degli errori rispetto ad un valore V

Tante misure ripetute nel tempo assumono la forma di una distribuzione gaussiana. Il *valore medio* (E_{AM}) rappresenta la **componente sistematica dell'errore**, mentre lo *scostamento* (della singola misura rispetto al valore medio rilevato) è detto **componente accidentale dell'errore**.

1. Le componenti sistematiche sono dovute a taratura imperfetta, e sono problematiche in calcoli di tipo integrativo. Si riducono con taratura (HW e SW). Nei calcoli integrativi continuo a sommare le misure, e si sommano anche le componenti sistematiche.
2. Le componenti accidentali sono dovute ai disturbi, sono problematiche nei calcoli derivativi e si riducono con operazioni di filtraggio (es. media tra N campioni: occorre una certa potenza di calcolo, in tempo reale posso solo fare la media dei campioni precedenti, ovvero introduco un ritardo).

Proprietà delle tecniche di misura

Ripetibilità (precision): misura che presenta una ridotta componente accidentale (stabilità: ripetibilità nel lungo periodo). **Accuratezza (unbias)**: misura che presenta una ridotta componente sistematica. **Precisione (accuracy)**: tecnica di misura che presenta ridotti valori degli errori assoluti globali.

3.1.2 Distribuzione degli errori rispetto a diversi valori V

E'utile analizzare l'andamento degli errori al variare del valore V (che prima abbiamo supposto costante). Questa stima è utile nel caso di misure d'errore relativo:

1. Errore assoluto costante: la fascia in cui cadono i valori di misura ha un'ampiezza costante al variare del valore di V. In questo caso l'errore relativo è una costante fratto il valore V (K/V iperbole). *Maggiore è il valore di V, più piccolo è l'errore relativo*. In questo caso la tecnica di misurazione è accettabile solo al di sopra di una certa **soglia**, in quanto per valori troppo piccolo l'entità dell'errore continua a crescere (idealmente fino all'infinito).
2. Errore relativo costante: ho errori assoluti *proporzionali* al valore di V.

3.1.3 Errori di quantizzazione

Ho un valore reale di una misurazione M che deve essere rappresentato con simboli finiti. **Granularità (risoluzione)** : grandezza che misura gli intervalli di valori mappati su una singola configurazione di simboli. La quantizzazione è ripetibile (ho sempre lo stesso valore legato alla stessa configurazione), però si hanno situazioni di valori per cui ho rappresentazioni diverse dovute a piccoli errori accidentali.

Questi errori dipendono dal modo con cui effettuo la quantizzazione, che mi danno luogo a granularità differenti. (es. Rappresentazioni *integer*, granularità costante; Rappresentazioni *float*)

- rappresentazione intera senza troncamento;
- rappresentazione intera con troncamento (es. Orologi digitali);
- rappresentazione float: errore relativo costante.

Generalmente comunque i sensori lavorano o con gli integer o con una misura analogica da convertire con un ADC (anche questo funziona con rappresentazione integer). In un sistema di misura si usano gli amplificatori prima di effettuare la conversione, in modo da raggiungere la

soglia minima per ridurre l'entità degli errori di quantizzazione: esso porta le misure significative nella parte alta della scala in modo da non incorrere nell'errore elevato in zone prossime allo zero.

Si sceglie rappresentazione integer oppure float a seconda dell'ambito di lavoro e dei requisiti di precisione del problema.

4 Caratterizzazioni di tempo ed evoluzione

4.1 Stati continui a tempo continuo

Il comportamento del sistema è visto come una descrizione di stati che cambiano in modo continuo nel tempo. Si adottano degli stati continui a tempo continuo per descrivere un fenomeno fisico (una situazione presente a prescindere dal mio interesse per essa)(Nota: stato fisico \neq stato degli automi. Gli stati sono associati alle grandezze fisiche intrinseche nel processo).

Un aspetto fondamentale è rappresentato dalla gamma di valori che uso per definire lo stato (fatto che andrà preso in considerazione dalle tecniche di acquisizione che dovranno recepire i valori misurati). La gamma di valori non comprende la descrizione dell'effettivo comportamento del sistema. Le descrizioni di tipo dinamico vengono date da altre grandezze, con granularità descrittive differenti. Ad esempio le grandezze analogiche sono caratterizzate da uno spettro, da costanti di tempo e da slew rate (massima derivata). Tutti questi parametri mi danno idea anche delle variazioni delle grandezze, per sapere come devo organizzarmi nell'acquisizioni dati in modo da avere misurazioni affidabili e valide. (scostamenti accettabili e slew rate mi indicano le temporizzazioni sulle frequenze di campionamento).

4.2 Stati continui a tempo discreto

Sono stati caratterizzati da valori stabili, finchè non scatta il prossimo valore temporale (es. Misure derivate dal campionamento S&H). Ho una transizione linearmente istantanea da un valore all'altro. Ad ogni scadenza di intervallo discreto potrei avere una modifica dello stato. Ho una rappresentazione discreta nel tempo di grandezze continue, per motivi di comodità. In casi di questo tipo è inutile considerare grandezze come slew rate nel senso stretto del termine. Si può parlare di rapporto incrementale massimo e di periodo.

4.3 Stati discreti a tempo continuo

Possono assumere un numero finito di valori in tempi non periodici (es. # pezzi prodotti in catena di montaggio). A volte i valori sono enumerativi. Le transizioni di stato possono avvenire in qualsiasi istante. Vengono caratterizzati dalla minima durata significativa in uno stato, per capire come ammettere le transizioni di stato quando durano un tempo infinitesimo. Ha senso contare una transizione in uno stato che è durata al di sotto di una certa soglia minima?

Uno stato è stato descrittivo se è durato un certo intervallo di tempo superiore ad un limite fissato. Tutto ciò che è più breve, non viene considerato dal mio sistema. Se capita, allora quello stato viene considerato spurio: *irrilevante* se non cambia nulla contarli oppure no, *da ignorare* se considerarlo porterebbe ad errore, *malfunzionamento* se sono fatti di cui si deve tenere conto.

- Gli spurii che sono irrilevanti sono quelli che non mi inducono alcuna alterazione significativa dello stato del sistema, in alcun caso.
- Gli spurii che vanno ignorati sono quelli che, se non ignorati, provocherebbero cambiamenti irreversibili nello stato.
- Certi spurii corrispondono a malfunzionamenti: non solo li devo ignorare (nella catena di controllo), ma devo segnalarli all'operatore come condizioni di guasto.

L'intervallo minimo su cui decido se un valore è spurio o meno è legato al minimo tempo di decisione sulle mie intenzioni (stati che permangono per meno di questo tempo prima di commutare non possono essere imputati al mio tempo di decisione – o alle costanti di tempo del processo coinvolto – quindi sono spurii). Gli spurii sono un concetto relativo alla singola applicazione, ed alla significatività delle informazioni.

4.4 Stati discreti a tempo discreto

Ho stati caratterizzati da un set finito di tipo enumerativo, e possono effettuare transizioni solo in istanti di tempo prefissato (es. circuito logico sincrono: 2 valori true/false, tempo di clock). Le transizioni sono dette eventi. La caratterizzazione si basa sui valori ammissibili e sulla temporizzazione degli intervalli di transizione

4.5 Eventi

4.5.1 Flussi di eventi

Sono un'astrazione comoda che però non esiste nella realtà. Un **evento** è un passaggio da uno stato ad un altro, che avviene in un tempo estremamente breve e di cui non interessa la dinamica della transizione.

Uno **stato** può essere visto come un flusso di eventi.

Gli eventi possono essere caratterizzati in base alla sorgente che li produce.

- **Eventi Temporal** (es. Tempo discreto, caratterizzato dal tempo di clock). Certe scadenze temporali costituiscono eventi cui vale la pena considerare.
- **Eventi Esterni** (prodotti da fenomeni esterni al calcolatore e raccolti tramite interfacce – generalmente ad interrupt. Non derivano dall'esecuzione dei programmi. Acquisisco solo quelli rilevanti per la mia applicazione)
- **Eventi Interni** (causati dall'esecuzione delle istruzioni, sono intrinsecamente percepiti dall'esecuzione dei programmi)

L'evento in genere è un'astrazione significativa a seconda della granularità temporale di interesse. La stessa caratterizzazione a stati discreti, se si diminuisce il granulo temporale considerato potrebbe apparire come una caratterizzazione a stati continui. In definitiva quindi bisogna ricordare che le astrazioni si fanno se è lecito e sensato farle, ovvero possono essere viste in modi differenti a seconda dei contesti.

4.5.2 Contenuto informativo degli eventi

- Eventi incrementali: si ricava il nuovo stato dall'evento solo se è noto lo stato precedente all'evento. La perdita di un evento incrementale è un errore fatale che va recuperato. Il nuovo stato è corretto se a loro volta lo erano gli stati precedenti. Si cerca di evitare la ridondanza nell'informazione
- Eventi assoluti: contengono al loro interno tutto il necessario per stabilire qual'è il prossimo stato che si raggiunge, a prescindere dalla correttezza dell'informazione precedente. Comportano ridondanza, ma implicano robustezza nell'informazione (sopperire ad errori)

In genere la scelta va commisurata al tasso di errore sulla misura. Più è alto, più ridondanza è meglio avere.

4.5.3 Proprietà temporali degli eventi

Gli eventi sono caratterizzati da una densità di probabilità rispetto al susseguirsi di eventi in un flusso temporale.

- **Flussi di eventi APERIODICI:** avvengono in istanti imprevedibili. Si dà un'idea della probabilità dell'evento in tempi brevi, qual'è il valor medio della probabilità. Se non ho un tempo minimo di distanza dall'evento precedente, perderò degli eventi (perché dimensiono il sistema di acquisizione in un certo modo). La probabilità mi può indicare la percentuale di eventi (errori) che posso perdere in un certo lasso di tempo (e magari compensarla)

....

Azioni: unità di lavoro di durata finita e breve, attivate da eventi. Tipiche delle funzionalità programmate. Il trigger non “attiva” un'azione, la “prenota”: è lo scheduler che effettivamente attiva un'azione. Il trigger è una richiesta di eseguire un'azione. *Scopo Temporale* di un'azione: intervallo tra trigger di start e trigger di stop. In genere non ci interessa la modalità di svolgimento dell'azione, ci interessano solamente gli istanti di prenotazione e quello di fornitura del risultato (ammettendo che diversi svolgimenti non portino a modifiche nello stato finale, a parità di condizioni iniziali). I calcolatori sono predisposti per eseguire azioni. Attivazione di volta in volta delle azioni di interesse.

Attività: comportamento continuo nel tempo, tipico delle funzionalità intrinseche in un processo (attività fisiche). Caratterizzate da una funzione di trasferimento, questi comportamenti si stagliano in modo indefinito nel tempo, come se non avessero inizio o fine. L'uscita è funzione dello stato interno raggiunto e degli ingressi. I calcolatori non sono in grado di svolgere attività in senso stretto, però si riesce a ottenere un comportamento equivalente in modo accettabile. Si può ridurre il tutto ad una sequenza di azioni cicliche (acquisizione dei dati, elaborazione delle uscite, trasmissione dei valori calcolati) che, se eseguite con una tempistica sufficiente, possono essere indistinguibili da un'elaborazione continua ideale. Inibizione di tutte le attività che non sono di interesse.

Prestazioni temporali richieste: i requisiti temporali vanno selezionati dagli esperti dell'impianto non dall'informatico. Si dice qual'è il massimo tempo di risposta accettabile (*deadline*). Quelle che arrivano prima sono corrette, quelle dopo no. Però non è detto che questa distinzione così netta sia valida. Inoltre in un momento di sovraccarico parecchi eventi diverrebbero inaccettabili a causa dei ritardi.

Si può quindi stabilire che alcune risposte, anche se non accettabili riguardo alla *deadline*, possono comunque essere usate senza gravi danni, altre invece sono della massima criticità ed è un errore grave qualsiasi sfioramento rispetto alla *deadline*.

Si definisce **funzione di validità temporale della risposta** una funzione (raramente espressa in forma analitica) che ci fornisce informazioni: dov'è la *deadline*, quant'è rapido il degrado dopo la *deadline* a breve termine, che cosa succede se si finisce parecchio oltre la *deadline* (*frequenze di campionamento, scadenza temporale di risposte ad eventi, validità temporale della risposta*).

Bisogna classificare le esigenze real-time in almeno due categorie:

- **Hard real time:** esigenze per cui i ritardi provocano danni catastrofici;
- **Soft real time:** esigenze per cui i ritardi provocano danni comunque accettabili e quindi hanno priorità rispetto alle esigenze hard.

Si adottano quindi, in caso di sovraccarico, tecniche di scheduling in modo da eseguire subito (lontano dalla *deadline*) quelle hard mentre posticipo più tardi quelle soft (in modo che possano

sforare dalla deadline solamente quelle risposte che non portano a danni catastrofici.)

Per quanto riguarda invece le attività, è più interessante accertare la periodicità della risposta (regolare e con basso GID -???-). E'preferibile una frequenza più bassa ma regolare a una elevata ma molto irregolare.

In questo caso si è fatta un'analisi in forma disaggregata.

Prestazioni temporali offerte: espresse dall'esperto di informatica, e dice quali sono le prestazioni temporali che il programma è in grado di offrire. Si suppone comunque che l'algoritmo lavori in modo netto, ovvero che abbia a disposizione la CPU, che non venga interrotto ... Nelle applicazioni real-time si considera l'esecuzione worst-case (riguardo ai dati)

- **Algoritmo normale singolo:** si ha, per l'esecuzione di un'azione, un algoritmo classico che produce la risposta alla fine dell'elaborazione.
- **Algoritmi normali multipli:** un risultato è prodotto dall'azione di più algoritmi: alcuni sono veloci e producono una risposta approssimata, alcuni sono lenti ma producono risultati precisi. Utile in situazioni di sovraccarico: lo scheduler può decidere che risposta attivare in modo da usare l'algoritmo veloce in situazioni di sovraccarico, oppure di usare l'algoritmo più preciso quando non sono in situazione di sovraccarico.
- **Algoritmi ad approssimazioni successive:** producono risultato approssimato in tempi brevi, e se hanno possibilità di essere eseguiti ciclicamente possono migliorare l'approssimazione calcolata utilizzando più cicli di elaborazione. Al termine di ogni ciclo elaborazione l'algoritmo è fatto in modo da essere interrompibile da parte del sistema operativo.

Prestazioni complessive (carico di lavoro dell'elaboratore): dipende da 3 componenti. Quante sono le cose da fare (n azioni potenziali da compiere). Qual'è il tempo di elaborazione netto di ogni azione (uno per ogni azione potenziale, Ci). Ogni quanto viene richiesta una particolare azione(Ti).

$U = \text{Somatoria } (Ci/Ti)$

U:fattore di utilizzazione; Ci:computation time; Ti:time period.

Ci non è costante, Ti non è costante, dipende dal carico che vogliamo calcolare (medio, di punta). Il corretto dimensionamento richiede che il carico di lavoro previsto deve corrispondere ad un valore minore di uno. Valori superiori di uno indicano che la CPU lavora sempre al 100% e si introduce ritardo: per smaltire i ritardi accumulati la CPU deve lavorare nominalmente molto al di sotto del 100% in modo da poter smaltire velocemente le situazioni di sovraccarico. Un carico è accettabile se la computazione non produce danni irreversibili.

Il carico di lavoro si suddivide in componenti periodiche (attività), aperiodiche (eventi) e di sottofondo (processi batch senza esigenze di realtime). Le CPU per il controllo realtime sono molto spesso scariche, per cui posso utilizzare i tempi morti per queste operazioni di sottofondo.

Descrizioni tecniche: efficacia valutata con quanta informazione viene trasmessa con il minor sforzo

..... manca un pezzzzzzzo

5 Tecniche di interfacciamento

5.1 Stati ed eventi

Stati (perdurano nel tempo, acquisibili in qualsiasi istante), Eventi (concetto che evidenzia passaggi tra due stati, supposti abbastanza veloci, caratterizzati dall'istante in cui si verificano e dalla transizione di stato che li contrassegnano). La modalità di acquisizione dell'informazione cambia a seconda di che cosa stiamo acquisendo: in ogni caso il meccanismo usato percepisce sempre degli stati, e quando sono interessato agli eventi devo dedurre l'evento dalle informazioni che ho ricavato dagli stati (es. Evento “premo il pulsante” è una transizione da stato “contatto aperto” a “contatto chiuso”

- con il **campionamento (SW)** acquisisco stati e non eventi. Posso ricavare eventi in un secondo tempo confrontando informazioni relative a diversi stati
- con gli **interrupt (HW)** percepisco eventi e non stati. Gli eventi hanno un ruolo attivo, forzando la CPU a eseguire la routine di gestione dell'interrupt, ovvero modificando il normale flusso del programma (esecuzione sequenziale e istruzioni di salto)
- con tecniche eterogenee HW/SW posso ricostruire stati da eventi e viceversa.

5.1.1 Acquisizione di stati

Bisogna definire una politica di campionamento, scegliendo un opportuno *periodo di campionamento* rispetto alle specifiche del problema. Si imposta anche una certa tolleranza temporale per approssimare l'istante in cui si è verificato l'evento. Ecco alcuni possibili aspetti da includere nelle specifiche

- *Massimo ritardo accettabile*: voglio conoscere il cambiamento del valore entro un ritardo max stabilito
- *Durata minima degli stati da rilevare / eliminazione degli spurii*: denota la differenza tra stati consistenti e segnali spurii (quelli che durano meno del minimo)
- *Frequenza minima di rilevamento*: tradeoff tra carico di lavoro della CPU e necessità di sufficiente precisione del rilevamento.
- *Precisione della base temporale*:
- *Correlazione tra diversi segnali*:

Bisogna preservare la parte significativa dell'informazione che voglio acquisire, non preoccupandomi eccessivamente di errori che non mi rovinano la parte significativa.

Per l'emissione degli stati, si fanno discorsi simili all'acquisizione, generalmente con vincoli meno stringenti perchè l'emissione delle informazioni avviene in modo “volontario” e non ho forzature date dal processo.

5.1.2 Emissione di eventi

Essi sono o *dedotti dagli stati* oppure che venga inoltrata una *richiesta di interrupt* (o tramite DMA per informazioni di tipo burst). E'importante non perdere tempi incrementali eseguendo l'acquisizione entro il tempo morto del fenomeno (se l'informazione è assoluta, questo è meno grave perchè l'informazione errata viene corretta al ricevere della successiva informazione assoluta); in caso di eventi con brevissimo tempo morto ma ampio tempo medio bisogna partizionare con accortezza la parte di sistema che risponde all'evento e la parte che lo propaga al resto del sistema, non duplicare eventi e riconoscere sequenze di eventi.

5.1.3 Rappresentazioni interne

Vengono adottate variabili *immagine* del mondo esterno. Rappresentano un'area dove chi percepisce gli eventi associano loro un valore, mentre chi è interessato agli stessi le legge e ne fa calcoli. E' necessario disaccoppiare il momento di acquisizione (aggiornamento) delle variabili immagine rispetto al momento di elaborazione delle stesse variabili, operazioni che verranno gestite ognuna da un set di processi e procedure appositi.

Immagini di stato: per ogni stato del mondo esterno ho una variabile interna valorizzata in modo tale da rappresentare il valore fisico reale entro una certa fascia di tolleranza. Scritta da un processo di interfaccia, tipicamente attivato ciclicamente quando serve. Generalmente c'è un solo processo che scrive lo stato, altrimenti bisogna garantire sia la mutua esclusione tra i processi, sia l'atomicità dell'operazione di aggiornamento del valore, in modo da evitare inconsistenze nei dati.

Eventi assoluti: ho una variabile aggregata per ogni evento. Tuttavia ogni evento deve “mantenere la sua personalità” (ovvero una sua rappresentazione separata) fino al momento in cui esso viene assimilato nella variabile aggregata.

Eventi incrementali: In genere si crea una lista sequenziale che memorizza gli eventi che poi vengono sincronizzati (struttura *producer-consumer*). La sincronizzazione può essere strtta (rendez-vous): il produttore è obbligato ad aspettare il consumo dell'evento prima di produrne un altro. Oppure la sincronizzazione può essere lasca (introducendo un buffer FIFO). In questo modo ho disaccoppiato produttore e consumatore, che posso dimensionare a seconda del problema da risolvere. Tutto questo dipende dalle condizioni temporali di realizzabilità delle soluzioni.

Si può usare un automa per descrivere le mosse di acquisizione nei vari casi, tenendo conto anche delle possibili anomalie che si possono verificato. **USART**: circuito che cura i meccanismi che servono a trasmettere/ricevere informazioni in forma digitale.

5.1.4 Temporizzazioni/sincronizzazioni nell I/O di informazioni

analisi componenti temporali delle elaborazioni (calcolo del tempo di risposta nei suoi elementi).

...