Architetture di calcolo per applicazioni safety-critical embedded

I guasti dei sistemi elettronici safety-critical possono provocare danni di notevole entità: da qui l'esigenza di utilizzare architetture di calcolo particolari

Manfred Schmitz
Technical director
MEN Mikro Elektronik

I sistemi elettronici safety-critical sono utilizzati, per esempio, in apparecchiature medicali, aeroplani, treni e centrali nucleari. I termini 'safety' e 'security' (che si traducono entrambi con 'sicurezza') sono spesso confusi fra loro. Il termine tecnico 'security' può significare non solo che qualcosa è sicuro, ma anche che è stato reso sicuro. Un sistema 'safe' ha invece un comportamento d'errore determinato.

Si distingue fra due modi fondamentali di comportamento d'errore. I sistemi 'fail-operational' continuano a funzionare anche al verificarsi di un errore: è il caso, per esempio, dei sistemi di controllo degli aeroplani e dei reattori nucleari a sicurezza passiva. I sistemi 'fail-safe' vengono invece disattivati e portati in uno stato sicuro in caso di guasto. Un'automobile o un treno, per esempio, possono essere fermati quando il sistema di controllo si rompe, ma per un aeroplano ciò non è possibile.

I sistemi che continuano a funzionare correttamente quando parte del sistema si rompe sono definiti fault-tolerant. I piloti automatici negli aeroplani o il sistema di controllo di una normale centrale nucleare funzionano in base a tale principio. Un metodo spesso utilizzato per ottenere la fault tolerance è il test permanente di tutti i componenti. In caso di errore, il componente difettoso viene escluso e se ne usa un altro. Nella figura 1 viene riportato lo schema del controllo di processo.

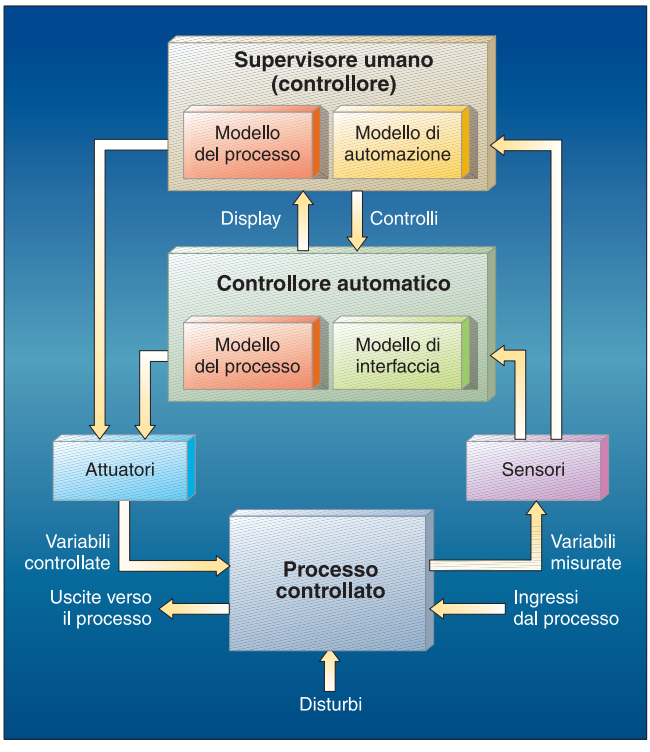

Fig. 1 - Controllo di processo

Pianificazione dello sviluppo

Lo standard di sicurezza (safety) a cui un sistema deve ottemperare è descritto da classi, chiamate livelli SIL (Safety Integrity Level). Per ogni livello SIL, è richiesta una certa affidabilità. Nei vari settori vi sono diversi standard. L'IEC61508 ha un'importanza particolare. Esso definisce 4 livelli, partendo da 1: il livello 4 ha quindi la massima affidabilità. 0ppure significa che non esistono particolari requisiti in termini di sicurezza.

L'affidabilità richiesta dipende dalla frequenza d'uso. Per le applicazioni che sono usate infrequentemente, come gli arresti di emergenza, lo standard definisce una PFD (probabilità di guasto a richiesta) massima di:
- SIL4: da 10-5 a 10-4
- SIL3: da 10-4 a 10-3
- SIL2: da 10-3 a 10-2
- SIL1: da 10-2 a 10-1

Altri standard industriali importanti in questo contesto sono:
- IEC61508 (Sicurezza funzionale di sistemi elettrici/elettronici/a elettronica programmabile legati alla sicurezza).

In avionica:
- D0-178B (Considerazioni sul software nei sistemi avionici e certificazione delle apparecchiature)
- DO-254 (Guida alla progettazione sicura di elettronica per avionica).

Per le applicazioni ferroviarie:
- EN50126 (Applicazioni ferroviarie – specifica e dimostrazione di sistemi di trasporto)
- EN50128 (Applicazioni ferroviarie - software per controllo ferroviario e sistemi di protezione)
- EN50129 (Applicazioni ferroviarie – sistemi elettronici di sicurezza per il segnalamento).

Questi sistemi devono offrire sempre un'estrema affidabilità. Ciò significa che già il processo di viluppo deve soddisfare requisiti elevati. La pianificazione dello sviluppo è quindi molto importante. Il DO-178B, per esempio, prevede 10 tipi di documenti differenti nella fase di pianificazione. Durante lo sviluppo viene seguito il modello a V (Fig. 2). A livello di sistema si formulano i requisiti (che cosa deve essere fatto), che vengono quindi tradotti in un'architettura (come farlo) a livello di sistema. Da ciò si ricavano i requisiti per i singoli componenti e, sulla base di tali requisiti, si sviluppa l'architettura dei componenti. In base alla complessità del sistema, vi possono essere più cascate di documenti dal livello di sistema al livello dei moduli. L'ultimo passo è la concezione effettiva: progetto e implementazione. Questi documenti formano il lato sinistro della 'V'. La controparte della concezione è il test dei moduli, all'architettura corrisponde l'integrazione, ai requisiti corrisponde la validazione e così via da cascata a cascata fino al livello più alto di sistema. Questa parte forma il lato destro della 'V'.

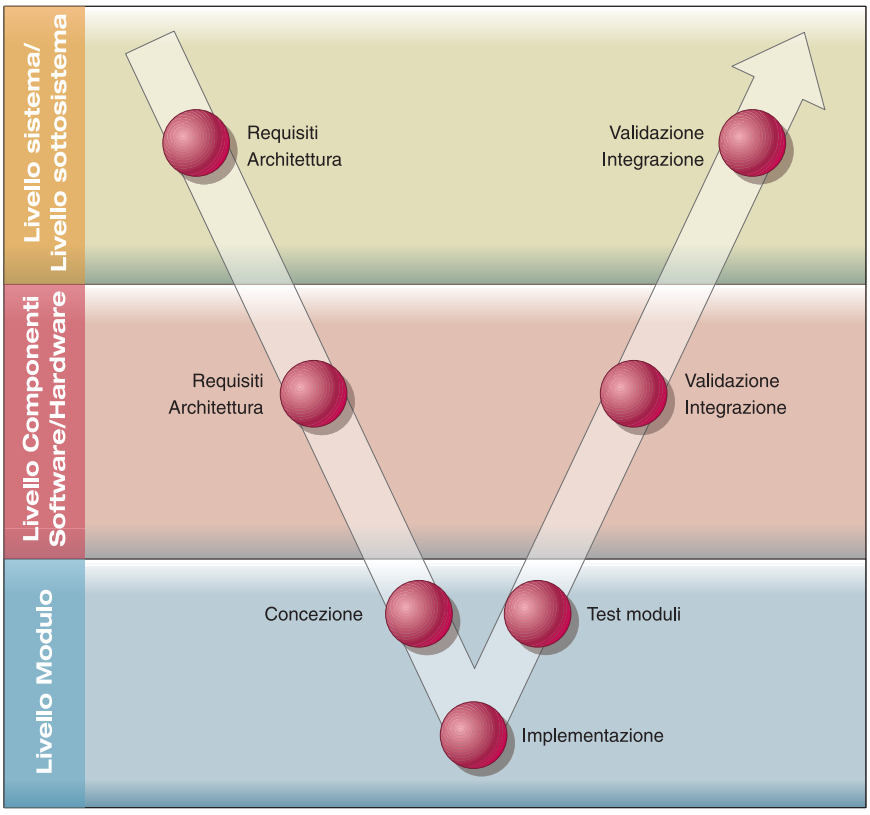

Fig. 2 - Sviluppo secondo il modello a V

Formulazione dei requisiti

Tutti i documenti dei requisiti (inclusi i documenti di architettura e concezione, oltre alla specifica dei requisiti effettivi) devono essere scritti in un chiaro linguaggio per requisiti, da definire. L'uso di espressioni chiare può già garantire la possibilità di verificare e testare ogni requisito. Formulare requisiti chiari e verificabili è molto difficile e richiede molta esperienza nel settore. È importante che il contesto globale rimanga chiaro. Ogni requisito deve essere implementato e testato e deve essere possibile verificare la completezza della catena. Manualmente questo è molto difficile. Spesso, a questo scopo si usano quindi tool basati su data base, come Doors di Telelogic. Questi tool permettono di tracciare i requisiti, i rispettivi casi di test e i rapporti di test in modo relativamente semplice attraverso tutti i documenti (Requirement Tracing). Tuttavia, l'uso di questi tool non garantisce che i requisiti siano formulati correttamente. L'uso dei tool, inoltre, è complesso e spesso una sola persona in un team di progettazione è in grado di inserire i requisiti; per questo, i requisiti spesso non sono aggiornati.

A parte il Requirement Tracing, è necessario utilizzare diversi altri metodi di concezione. Nello sviluppo software e Fpga, si utilizza il 'Code-Rule-Checking', dove il codice è controllato automaticamente in base a certi criteri formali. Il 'Code-

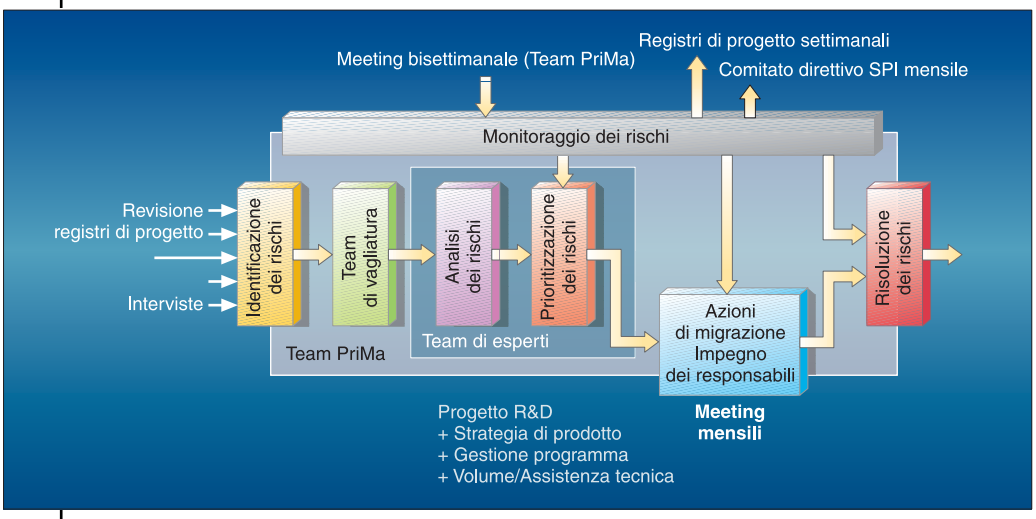

Fig. 3 - Gestione del rischio

Coverage-Analysis' è invece un metodo per provare che ogni linea sia stata verificata almeno una volta. Altri metodi sono i test a scatola nera e scatola bianca, per assicurare che il software risponda a tutti i requisiti senza errori. Naturalmente, è necessaria una gestione della configurazione per riuscire a ritracciare le modifiche del codice e le loro cause. La scelta dei tool da utilizzare è molto important, perché deve essere garantito che l'editor, il compilatore e il linker portino al risultato desiderato. Strettamente parlando, questi tool devono essere prodotti con la stessa cura e la stessa quantità di tempo e denaro. Essi sono effettivamente disponibili; per esempio, sono inclusi in package di numerosi produttori di software, come Wind River (Platform for Safety Critical DO-178B), Green Hills Software (Integrity-178B RTOS) e LynuxWorks (LynxOS-178). Se si usa un software 'normale', la sua affidabilità operativa deve essere provata all'autorità di approvazione, cosa spesso difficoltosa.

Misure dopo lo sviluppo

Dopo lo sviluppo, è richiesta la gestione del rischio, basata sull'identificazione del rischio stesso (Fig. 3). Ciò include tutti i rischi: sia i rischi tecnici o finanziari, sia quelli relativi ai tempi di consegna e al personale coinvolto. È necessario determinare il significato di tali rischi, per riuscire a decidere se occorre analizzarli più in dettaglio. La valutazione dei rischi ha lo scopo di minimizzare i rischi nello sviluppo.
Strategie comuni: il trasferimento di un rischio (per esempio attraverso l'outsourcing di una parte dello sviluppo a un esperto esterno), evitare un rischio (usando un componente provato invece di un nuovo componente) o l'assicurazione di un rischio (ad esempio uno stabilimento contro gli incendi). Nella maggior parte dei casi, l'intero team di sviluppo cerca di identificare tutti i rischi con un brain storming.
Parte del team (gli esperti sui rischi) esegue quindi la valutazione, la prioritizzazione, il controllo regolare (normalmente mensile) e attiva le misure di rimedio se necessario.
È necessario eseguire anche vari test ambientali. Essi sono particolarmente importanti per i sistemi safety-critical, perché questi devono funzionare senza errori anche in condizioni estreme (temperature estreme, vibrazioni estreme e così via). Soprattutto in campo avionico sono richiesti i test Halt (Highly Accelerated Life Test). Nella normale qualificazione ambientale un sistema elettronico viene testato esattamente nelle stesse condizioni per le quali è stato sviluppato. Per esempio, se è richiesto un range di temperatura da -40 a 85°C, la qualificazione viene eseguita esattamente in tale range. Nel test Halt le procedure sono differenti, perché questi limiti vengono superati di proposito. A parte la temperatura, il DUT viene esposto contemporaneamente a vibrazioni estreme. A volte si varia anche la tensione di alimentazione. Lo scopo del test è trovare i limiti di un sistema. Il sistema viene distrutto di proposito; quindi si valuta il danno in modo da potere eseguire un miglioramento costruttivo in situ. Questo processo normalmente richiede da tre a cinque giorni, durante i quali gli ingegneri progettisti devono essere presenti. Con i test Halt è possibile rilevare i punti deboli del sistema ed ottenere un margine di sicurezza rispetto al requisito ambientale: la sicurezza.

Valutazione della sicurezza

Per la valutazione della sicurezza il valore del Mtbf (Mean Time Between Failure) ha un ruolo importante. In base al componente interessato, un errore può essere permanente o

temporaneo. Per esempio, se un resistore presenta un errore, è difettoso nella maggior parte dei casi. Se una memoria presenta un errore, è possibile che sia interessato un solo bit. Il valore del Mtbf indica il tempo medio che trascorre fra due errori e descrive l'affidabilità di un sistema. Il valore del Mtbf è spesso confuso con il valore del Mtbr (Mean Time Between Repair), che corrisponde al tempo medio fra due errori che possono essere rimossi solo con una riparazione. Non vi sono differenze fra il valore del Mtbf e il valore del Mtbr di un resistore, mentre i due valori sono diversi per una memoria.
Il valore del Mtbf è indicato in ore. Per ottenere valori più 'gestibili', si utilizza il valore Lambda o FIT (Mtbf = 1/Lambda). Se si conoscono i valori Lambda di tutti i componenti, si possono sommare tutti i Lambda per ottenere il Lambda totale, quindi il Mtbf di un sistema. Spesso è difficile ottenere il valore Lambda di un componente dal produttore; in questo caso è necessario stimare il valore. Inoltre, il FIT – l'affidabilità – dipende dallo 'stress' a cui il componente è esposto. Se un resistore da 0,5W viene fatto funzionare con una potenza di 0,5W, il suo ciclo di vita non sarà naturalmente lungo come se fosse fatto funzionare con 0,1W. Ciò significa che è possibile eseguire una valutazione solo avendo una conoscenza dettagliata del circuito. I possibili cambiamenti del valore di FIT sono descritti in molti standard concorrenti. Tra questi, i MIL-HDBK-217, Telcordia, Prism e IEC-TR-62380. Lo standard del futuro sembra tuttavia essere l'RDF 2000 (UTE C 80-810).
Il valore di Mtbf è necessario anche per l'analisi Fmea (Failure Mode and Effects Analysis). Durante l'Fmea viene controllato ogni componente dal basso verso l'alto. Un componente viene quindi valutato più in dettaglio se influenza il funzionamento sicuro del sistema, tenendo conto dello stato in cui il sistema viene commutato. Per un sistema con uno stato sicuro (fault-safe) come un treno o un'automobile, è possibile distinguere due tipi di guasti: quelli che portano in uno stato sicuro e quelli che portano in uno stato non sicuro. È necessario considerare solo i guasti pericolosi. Quando si scopre un errore vengono prese le necessarie misure di rimedio. Solo gli errori che non vengono scoperti sono critici; ciò significa che per determinare l'affidabilità di un sistema, ossia il suo livello SIL, occorre sommare solo il valore FIT dei guasti non scoperti.
Soprattutto in campo avionico è necessario tenere conto di effetti come i SEU (Single Event Upsets) e i MEU (Multiple Event Upsets). Si tratta di errori statistici causati dalle radiazioni cosmiche. Tali radiazioni provocano la commutazione delle strutture e dei registri di memorie statiche e dinamiche. È quindi possibile che un'unità processore, che potrebbe avere funzionato senza errori per vari decenni sul livello del mare, mostri un errore dopo poche ore a un'altitudine di 20.000 m. Per evitare i SEU è necessario prendere delle precauzioni, per esempio non usare RAM, in modo da ottenere un valore di Mtbf più elevato.

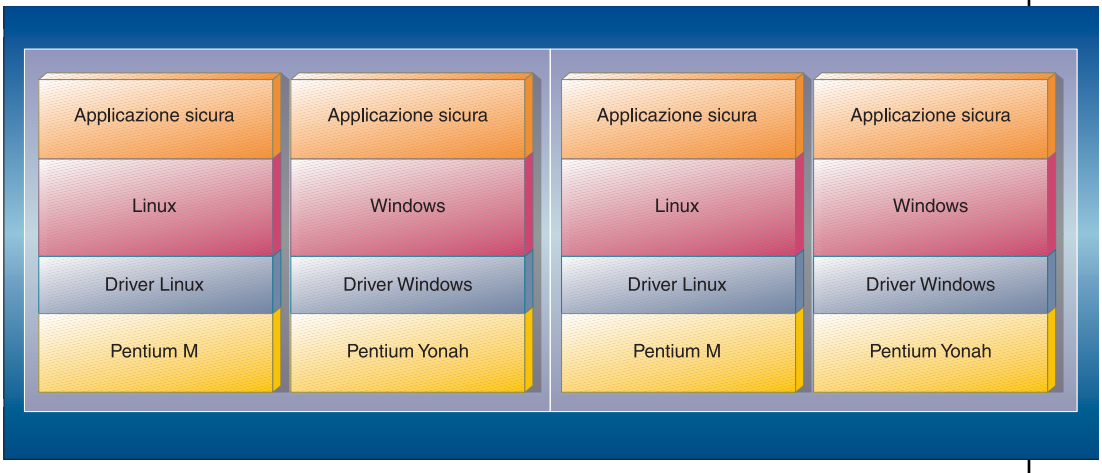

Fig. 4 - Struttura '2 di 4'

Sistemi ridondanti

Per la valutazione della sicurezza e la Fmea, è sempre richiesta una conoscenza del sistema. Se la Fmea/Mtbf non porta a un risultato abbastanza elevato, occorre prendere delle misure nello sviluppo dell'hardware per aumentare il valore FIT. Tali misure o la concezione dell'architettura possono essere eseguite solo considerando il sistema nel suo complesso. In ogni caso è necessario fare di tutto per rilevare gli errori. Il rilevamento degli errori può spesso essere portato avanti in

modo molto semplice, per esempio da una persona che osserva la macchina e la spegne in caso di pericolo; tuttavia, il valore di Mtbf di un essere umano non è particolarmente elevato. Nella maggior parte dei casi si cerca quindi di rendere sicuro il sistema o il controllo.
Un metodo per rilevare gli errori è la ridondanza. I sistemi o una loro parte (per esempio, l'unità CPU) vengono duplicati. Solo i due computer combinati possono portare il sistema nello stato critico. Un 'votante' esegue la valutazione. Se il votante rileva una differenza fra i due sistemi, porta il sistema nello stato sicuro, ossia lo spegne. Questa architettura viene chiamata '1 di 2'. Il metodo funziona solo nei sistemi 'fail safe'. Nei sistemi 'safe operational', il sistema viene tenuto nello stato di 'hot standby': quando una parte difettosa del sistema si autoesclude, l'altra parte ne assume le funzioni. Ora il sistema è funzionale ma non è sicuro. Il software è considerato sicuro se è stato programmato secondo le regole descritte sopra. Come detto sopra, l'hardware non è mai a prova di guasto. Con un'architettura '1 di 2', è possibile rilevare solo i singoli guasti. Gli standard industriali prescrivono, per i livelli SIL3 e SIL4, che sia possibile rilevare anche i doppi guasti. È per questo che le apparecchiature Bite (Build-In Test Equipment) con autotest incorporato sono così importanti. Spesso l'hardware e il software per l'autotest sono molto più complessi dell'apparecchiatura stessa.
In campo avionico l'approccio è differente. Ogni guasto è considerato un guasto locale (Mtbf), ma un sottosistema deve essere in grado di rilevare un difetto da solo (il secondo computer è solo in hot standby), quindi vi è qualche problema.
Nei concetti di ridondanza, particolare attenzione deve essere dedicata ai 'guasti con causa comune'. Se per esempio due computer paralleli hanno lo stesso difetto, nell'unità a virgola mobile tale errore non viene rilevato. Un evento analogo si verifica su uscite binarie dello stesso tipo. Una sovratensione può quindi distruggere contemporaneamente due uscite ridondanti. Altri esempi di guasti con causa comune sono gli alimentatori e spesso anche i cablaggi. Si cerca di evitare i guasti con causa comune rendendo il più possibile indipendenti fra loro i due canali. Due schede CPU, per esempio, vengono equipaggiate con due alimentatori, due timer, e così via. Il principio può essere ulteriormente perfezionato con la diversità. Se si usano due CPU separate non può verificarsi contemporaneamente un singolo guasto, quindi esso non può sfuggire.
Con i classici sistemi ridondanti e un votante è possibile diminuire la probabilità di errore, in quanto il sistema passa nello stato sicuro in caso di errore. Tuttavia diminuisce la disponibilità. Se la disponibilità è importante (così importante che si è disposti a pagare per averla) si usano spesso i tripli sistemi o '2 di 3'. In questo caso si parte dall'ipotesi che i singoli componenti siano sicuri: un sistema operativo sicuro, e così via. Il votante valuta tre voti e la maggioranza decide. Se un componente fornisce un risultato deviante viene escluso. Il sistema rimane sicuro ma la disponibilità diminuisce. Con un altro computer addizionale si ottiene una struttura '2 di 4' (fig. 4): quando un componente si guasta rimangono tre componenti funzionali; il sistema rimane sicuro e disponibile. Se si vogliono utilizzare sistemi operativi standard (non sicuri), si può adottare una struttura modificata con due tuple. Una sola tupla è diversitaria e viene disattivata in caso di errore, mentre la seconda tupla continua a funzionare. Questa struttura accoppiata in modo diversitario non è disponibile come la struttura '2 di 4' descritta sopra, ma in certe circostanze permette l'uso di sistemi operativi standard. Per piccolissimi sistemi controllo o controllori è possibile utilizzare architetture ancora più semplici, purché tutti i componenti siano sviluppati 'in modo sicuro'. Per esempio, è possibile combinare una semplice CPU che ha un valore di Mtbf estremo con un piccolo watchdog intelligente. Il watchdog controlla i risultati dell'autotest della CPU e se tali risultati sono forniti in tempo. Esso attende quindi un certo risultato in un certo istante.
Come descritto sopra, il software è considerato fault-free se è stato sviluppato secondo gli standard richiesti, ma ciò diventa difficile nel caso del sistema operativo. In avionica l'unica soluzione è spesso rendere 'sicuro' anche il sistema operativo; ciò significa che l'intero sistema operativo deve essere testato a fondo. A tale scopo sono necessari i sorgenti software, tempo e denaro. Nell'industria spesso vengono accettati anche altri concetti. Per esempio, l'uso di due sistemi operativi affidabili differenti può essere una soluzione se almeno l'applicazione è stata sviluppata secondo le regole del caso.
Le architetture safety-critical sono molto complesse. Esse non riguardano solo l'hardware e il software in quanto tali, ma anche l'intero processo e di sviluppo e anche i tool utilizzati. I concetti di architettura per i diversi settori – ferroviario, avionico, automobilistico, dell'ingegneria medicale, e così via. – hanno solo piccole differenze, ma è differente nei vari mercati il modo di pensare e sviluppare.

MEN Mikro Elektronik (Goma Elettronica)
readerservice.it n. 37