Ernesto Cappelletti (ErnestoCappelletti)

IL SOFTWARE SECONDO LA NORMA UNI EN ISO 13849-1:2008 (IIA PARTE)

6 April 2012

1. Requisiti per la scrittura del software secondo la norma UNI EN ISO 13849-1:2008

I requisiti per la scrittura del software sono tesi ad evitare la presenza di errori. I requisiti sono più severi per funzioni aventi PLr maggiori. Se il software controlla funzioni di sicurezza aventi differenti PLr, devono essere soddisfatti i requisiti per il PLr maggiore. Un metodo per cercare di evitare errori nel software è fare in modo che le attività di verifica vengano effettuate da persone diverse, ed il più possibile indipendenti, da chi ha scritto il software.

1.1. Requisiti della norma UNI EN ISO 13849-1:2008 per SRASW

SRASW scritti in LVL e compilati secondo le seguenti prescrizioni possono raggiungere PL da **a** a **e**.

1.1.1. SRASW per funzioni con PLr da a a e

Per SRASW di funzioni con PLr da **a** a **e**, devono essere applicate le seguenti misure di base:

- implementazione del ciclo di vita con attività di verifica e validazione;
- documentazione delle specifiche e della progettazione;
- programmazione modulare e strutturata;
- test funzionali;
- appropriate attività di verifica dopo eventuali modifiche.

1.1.2. SRASW per funzioni con PLr da c a e

Per SRASW di funzioni con PLr da **c** fino a **e**, sono richieste o raccomandate le misure aggiuntive ad efficienza crescente riportate nel seguito(efficienza inferiore per PLr pari a **c**, efficienza intermedia per PLr pari a **d** ed efficienza superiore per PLr pari a **e**):

- le specifiche del software legato alla sicurezza devono essere riviste da ogni persona coinvolta nel progetto durante il ciclo di vita e devono contenere le seguenti descrizioni:

– funzioni di sicurezza con i PL richiesti e i modi di operare associati,

– criteri di valutazione delle prestazioni (per esempio tempi di reazione),

– architettura hardware e segnali esterni di interfaccia,

– individuazione e controllo dei guasti esterni;

- selezione di strumenti, librerie e linguaggi:

– strumenti di provata affidabilità: per PL **e** raggiunto con un componente ed il relativo strumento, lo strumento deve essere conforme alla norma di riferimento; se vengono usati due diversi componenti con due diversi strumenti, è sufficiente che siano di provata affidabilità. Devono essere usati accorgimenti tecnici in grado di individuare le condizioni che potrebbero causare errori sistematici. I controlli devono essere effettuati principalmente durante la compilazione e non solo durante l'esecuzione del software;

– ogni volta che è ragionevolmente possibile dovrebbero essere utilizzati librerie e blocchi funzionali (FB) validati,quelli forniti dal produttore dello strumento (altamente raccomandati per PL e)oppure librerie e blocchi funzionali specifiche per l'applicazione validate, in conformità alla norma UNI EN ISO 13849-1;

– per un approccio modulare dovrebbe essere usato un appropriato LVL, per esempio sottoinsiemi dei linguaggi della norma IEC61131-3; linguaggi grafici (ad esempio blocchi funzionali, ladder diagram) sono altamente raccomandati;

- la progettazione del software deve presentare:

– metodi semi-formali per la descrizione dei dati ed il controllo dei flussi, per esempio state diagram o program flow chart;

– programmazione realizzata principalmente attraverso strutture modulari e strutturate, derivate da librerie funzionali validate;

– blocchi funzionali con dimensioni di codifica limitate;

– esecuzione del codice all'interno di un blocco funzionale che dovrebbe avere un solo punto di ingresso e uno di uscita;

– architettura a tre fasi: ingressi → elaborazione → uscite;

– assegnazione di un’uscita di sicurezza ad una sola locazione di programma;

– uso di tecniche per l’individuazione dei guasti esterni e per la programmazione difensiva con blocchi di ingresso, processo e uscita che portano ad uno stato sicuro;

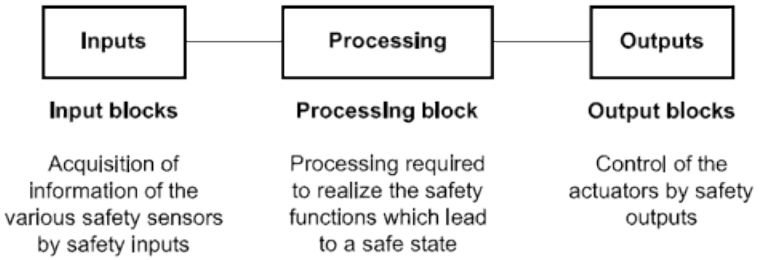

figura 1.png

Figura 1 — Architettura a tre blocchi: ingressi → elaborazione → uscite

- quando in un componente sono combinate SRASW e non-SRASW:

- SRASW e non-SRASW devono essere codificati in blocchi funzionali separati con collegamenti ben definiti;

– non devono esserci combinazioni logiche tra dati legati alla sicurezza e non legati alla sicurezza che potrebbero abbassare il livello di integrità dei segnali di sicurezza (per esempio combinando segnali legati alla sicurezza e segnali non legati alla sicurezza mediante un OR logico il cui risultato controlli i segnali legati alla sicurezza);

- implementazione/codifica del software:

– il codice deve essere leggibile, comprensibile e testabile;

– dovrebbero essere usate delle linee guida dicodifica comuni;

– deve essere verificata l’integrità dei dati;

– il codice deve essere verificato tramite una simulazione;

– per PL **d** o **e** la verifica deve essere effettuata tramite controllo e analisi dei flussi di dati;

- prove:

– il metodo di validazione appropriato è costituito da prove a scatola nera (black box) del comportamento funzionale e dei criteri prestazionali (ad esempio prestazioni in termini di tempo);

– per PL **d** o **e**, è raccomandata l'esecuzionedi test che utilizzino valori limite;

– è raccomandata una pianificazione dei test, che dovrebbe comprendere criteri di completamento e strumenti da utilizzare;

– i test degli ingressi e delle uscite devono assicurare che i segnali di sicurezza siano usati correttamente dal SRASW;

- documentazione:

– tutto il ciclo di vita e le attività di modifica devono essere documentate;

– la documentazione deve essere completa,disponibile e leggibile;

– la documentazione del codice deve contenere le intestazioni dei moduli con le entità ammesse, la descrizione funzionale e degli ingressi e delle uscite, l'indicazione della versione del software edelle librerie di blocchi funzionali utilizzate e sufficienti commenti.

- verifica:

– la verifica è necessaria solo per il codice specifico dell'applicazione e non per funzioni di librerie già validate;

- gestione della configurazione:

– si raccomanda fortemente di stabilire procedure e backup dati per l'archiviazione e l'identificazione di documenti, moduli software, risultati di verifiche e validazioni e configurazioni degli strumentidi sviluppo correlati ad ogni specifica versione del SRASW;

- modifiche:

– dopo modifiche di un SRASW, deve essere eseguita un'analisi di impatto per assicurare il rispetto delle specifiche;

– devono essere effettuate appropriate attività di verifica dopo le modifiche;

– devono essere stabiliti diritti di accesso per effettuare le modifiche e deve essere prodotta una documentazione cronologica delle modifiche.

2. Regole di programmazione (allegato J.4 della norma UNI EN ISO 13849-1:2008)

2.1. Regole di programmazione per la struttura del programma

La programmazione dovrebbe essere strutturata in modo da mostrare uno schema generale coerente e comprensibile che consenta la facile localizzazione delle diverse operazioni di elaborazione; ciò implica per esempio:

- uso di funzioni e blocchi disponibili dal linguaggio in uso;
- partizionare il programma in modo da identificare le diverse sezioni (ingressi, elaborazioni, uscite) che lo compongono;
- commentare ogni sezione del programma per facilitare l'aggiornamento in caso di modifica;
- descrizione delle funzioni richiamate e di quando vengono richiamate;
- mantenere la coerenza dei tipi di dati utilizzati con identificazioni univoche;
- la sequenza di esecuzione del programma deve essere definita e non deve poter saltare alcune parti se non espressamente indicato;
- evitare di avere parti di codice che non vengono mai eseguite (ad esempio prove o retaggi di versioni precedenti);*
- ogni variabile globale (sia di input che di output) deve avere un nome esplicativo del suo significato e deve essere descritto da un commento.

2.2 Regole di programmazione concernenti l'uso di variabili

L'attivazione o la disattivazione di tutte le uscite dovrebbero avvenire una sola volta(condizioni centralizzate).

Il programma dovrebbe essere strutturato in modo che le equazioni per aggiornare una variabile siano centralizzate. Ogni variabile globale,ingresso o uscita, dovrebb avere un nome mnemonico sufficientemente esplicito ed essere descritta mediante un commento.

2.3. Regole di programmazione a livello di un blocco di funzioni

È preferibile utilizzare blocchi precedentemente validati. La dimensione di ogni blocco dovrebbe essere limitata. I blocchi funzionali non devono modificare le variabili globali. Un blocco funzionale dovrebbe poter verificare l'eventuale inconsistenza dei dati in ingresso. Il guasto di un blocco deve essere identificato in modo da poter discriminare tra diversi guasti. Il codice di guasto e lo stato

del blocco a seguito del guasto dovrebbero essere segnalati e accompagnati da commenti. Il ripristino del blocco o il ripristino di uno stato normale dovrebbe essere accompagnato da commenti.

3. Esempio di software per funzioni di sicurezza

Operazioni necessarie:

- acquisizione delle informazioni da parte di diversi sensori;* elaborazione per far funzionare gli attuato ritenendo conto delle prescrizioni di sicurezza;
- controllo degli attuatori.

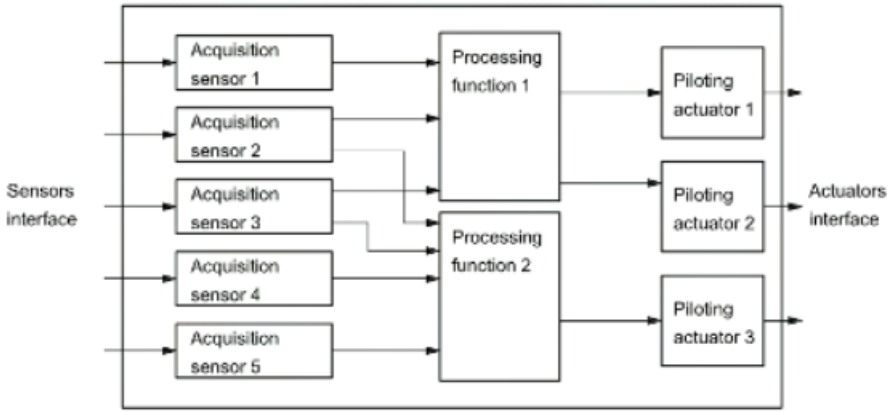

figura 2.png

Figura 2 — Blocchi di funzioni

3.1. Applicazione del modello a V per il ciclo di vita del software

Attività di sviluppo	Attività di verifica	Documentazione associata
Macchina: identificazione delle funzioni che coinvolgono le SRP/CS	Identificazione delle funzioni di sicurezza	Specifiche relative alle funzioni di sicurezza per il sistema di controllo
Architettura: definizione del sistema di comando con sensori e attuatori	Verifica delle caratteristiche di sicurezza degli elementi scelti	Definizione dell'architettura di comando
Specifiche software: trascrizione delle funzioni di sicurezza in funzioni del software	Rilettura e controllo delle descrizioni	Descrizione del software
Architettura del software: scomposizione delle funzioni in blocchi funzionali	Identificazione dei blocchi più critici che necessitano di una revisione e validazione più attenta	Modellazione dei blocchi funzionali
Codifica: stesura delle righe di codice necessarie	Rilettura e controllo del codice, verifica delle funzioni	Inserimento di commenti nel codice
Validazione: esecuzione di test e verifica degli aspetti funzionali e del comportamento a seguito di guasti	Verifica del grado di copertura dei test Verifica dei risultati dei test	Matrice di corrispondenza tra le specifiche e le prove Stesura di documenti che spiegano i test eseguiti e commenti in merito ai risultati ottenuti

figura 3.png

Estratto da "http://www.electroyou.it/mediawiki/index.php?title=UsersPages:Ernestocappelletti:il-software-secondo-la-norma-uni-en-iso-13849-1-2008-iia-parte"