7. Produzione di software critico

Docente: Tullio Vardanega
tullio.vardanega@math.unipd.it

Sicurezza di sistema

- La nozione di sicurezza (safety) di un sistema [software] emana dalla e si arricchisce dell'analisi degli incidenti [software]

Joint Software System Safety Committee
SOFTWARE SYSTEM SAFETY HANDBOOK
A Technical and Managerial Team Approach
Dicembre 1999

Sicurezza: definizioni - 1

- Lo standard MIL-STD 882B (1984) definisce la sicurezza, quale caratteristica di sistema, come:
  - L'assenza di (intesa come libertà da) condizioni capaci di causare danni mortali o severi alle persone e distruttivi o severi alle cose od alle proprietà

Sicurezza: definizioni - 2

- Lo stesso standard definisce la sicurezza di sistema, quale processo ingegneristico, come:
  - L'applicazione di principi, criteri e tecniche ingegneristici e gestionali per ottimizzare le caratteristiche di sicurezza del sistema nel rispetto dei vincoli di efficienza d'uso, di produzione e di costo durante tutte le fasi del ciclo di vita del sistema

Sicurezza: definizioni - 3

- Due termini con significato distinto
  - Sicurezza (*safety*): i requisiti di sicurezza tendono a rendere il sistema incapace di produrre danni catastrofici
  - Affidabilità` (*reliability*): riguarda la prevenzione di ogni tipo di errore che possa condurre ad un guasto di sistema
- Ai fini della sicurezza non è importante prevenire ogni guasto, ma assicurare che quelli che avvengono abbiano conseguenze tollerabili
  - Gli obiettivi di sicurezza e di affidabilità possono entrare in conflitto

Livelli di criticità - 1

- Molti standard di settore assegnano ai sistemi un livello di criticità che dipende da:
  - La severità del danno conseguente ad un guasto di sistema
  - La probabilità di occorrenza di tale guasto
- Al software in esecuzione sul sistema si attribuisce il livello di criticità dei guasti risultanti dal suo malfunzionamento

Livelli di criticità - 2

La **FAA** (*Federal Aviation Authority*) riconosce 5 categorie di guasto ed altrettanti livelli di criticità software

Effetto del guasto	Livello
Catastrofico	A
Rischio elevato	B
Rischio significativo	C
Rischio trascurabile	D
Senza effetto	E

Standard di settore - 1

- Aeronautica
  - **RTCA** (*Radio Technical Commission for Aeronautics*)
    - È a composizione mista industria-governo
    - Emette linee guida e requisiti standard
      - Il nome RTCA ora significa: *Requirements and Technical Concepts for Aviation*
    - Una delle commissioni di lavoro affiliate all'RTCA (la SC-167) è responsabile per la preparazione e la revisione di standard per la certificazione del software aeronautico

Standard di settore - 2

- **RTCA** (segue)
  - Nel dicembre 1992, la commissione SC-167 ha prodotto un documento denominato DO-178B che regola la fornitura di materiale atto ad ottenere l'approvazione di FAA per software di volo
- Difesa
  - Nell'agosto 1997, il dipartimento della difesa inglese (MoD) ha prodotto uno standard denominato Def-Stan 00-55 che regola l'acquisizione di software con caratteristiche di sicurezza

Standard di settore - 3

- Automobilistico
  - **MISRA** (*Motor Industry Software Reliability Association*)
    - È a composizione mista tra industrie automobilistiche, fornitori di componenti ed enti universitari
    - Emana linee guida per la produzione di software con caratteristiche di sicurezza, che applicano al suo intero ciclo di vita
    - Una vasta parte delle linee guida concerne l'uso di linguaggi di implementazione e dei compilatori

Standard di settore - 4

- Nucleare
  - Nel 1986, la **IEC** (*International Electrotechnical Commission*) ha pubblicato uno standard denominato IEC 880 (Software for Computers in the Safety Systems of Nuclear Power Stations)
  - Lo standard fornisce linee guida per la selezione del linguaggio di implementazione

Caratteristiche desiderabili del linguaggio di implementazione - 1

- Un linguaggio adatto per l'implementazione di sistemi con caratteristiche di sicurezza ha (almeno) le seguenti caratteristiche:
  - È definito mediante uno standard internazionale (ISO)
    - Consente di accertare la adeguatezza (*conformance*) del compilatore
  - Ha struttura modulare e supporta progettazione e codifica strutturate
    - Facilita l'astrazione ed il riuso di componenti verificate ed affidabili
  - Accelera la rilevazione di errori
    - Particolarmente a livello di compilazione

Caratteristiche desiderabili del linguaggio di implementazione - 2

- (segue)
  - Consente accesso logico e strutturato agli elementi hardware del sistema
    - Ha costrutti che rappresentano le caratteristiche salienti dell'hardware sottostante, p.es. associazione di componenti di strutture logiche a registri
  - Fornisce controllo rigoroso sulla visibilità di tipi, operazioni e dati ai moduli del programmi
    - Permette di determinare regole di accesso

Certificazione - 1

- Certificazione della sicurezza di una applicazione
  - L'applicazione comprende il supporto a tempo di esecuzione (sistema operativo) del quale fa uso
  - Consiste nel verificare che sviluppo e verifica siano state condotte secondo standard ingegneristici (derivati p.es. da ISO 12207) e di sicurezza (p.es. DO-178B) rigorosi

Certificazione - 2

- I processi software utilizzati determinano la certificabilità dell'applicazione
- Uso preferenziale di un modello di sviluppo sequenziale
- I processi e le relative attività sono fortemente orientati/e alla produzione di documentazione di supporto

Certificazione - 3

- Pianificazione della documentazione di supporto
  - Piano per gli aspetti software concernenti la certificazione
  - Piano di sviluppo del software
  - Piano di verifica del software
  - Piano di gestione della configurazione
  - Piano di assicurazione della qualità

Certificazione - 4

- Uso di standard di processo consolidati in ogni fase di sviluppo
  - Standard per la definizione dei requisiti software
  - Standard per la progettazione (*design*) del software
  - Standard di codifica
  - *Non esistono standard riconosciuti per la verifica, ma solo linee guida ed obiettivi!*

Certificazione - 5

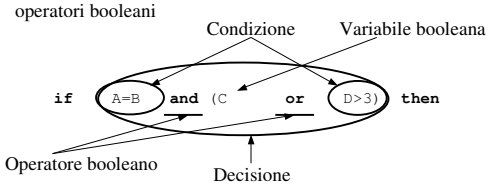
- Una vasta quantità di informazione deve essere collezionata come evidenza di esecuzione dei processi adottati
  - Specifiche ed analisi dei requisiti software
  - Descrizione e definizione del prodotto software
  - Codice sorgente ed eseguibile
  - Definizione, realizzazione e risultato dei test
  - Catalogo del sistema di configurazione
  - Lista dei problemi
  - Resoconto delle azioni di gestione e controllo di qualità

Certificazione - 6

- Strategie di verifica dinamica (test)
  - Black box: verifica che ciascuna funzione produce il risultato atteso in tutte le possibili condizioni in cui essa possa eseguire
    - Il test include l'uso di valori tipici e di valori "ai margini" (condizioni estreme)
  - White box: sulla base della struttura della funzione da testare assicura che tutti i suoi elementi e flussi di esecuzione siano necessari e verificati
    - Il test deve assicurare che il programma esegue correttamente in ogni condizione e che tutte le condizioni logiche in esso contenute eseguano correttamente in ogni ramo

Esempio: test di copertura di condizioni e decisioni - 1

Condizione: espressione booleana che non contiene operazioni booleane
Decisione: contiene una o più condizioni combinate mediante operatori booleani



Esempio: test di copertura di condizioni e decisioni - 2

DO-178B richiede che:

- Tutte le decisioni vengano eseguite e tutti i rispettivi esiti vengano prodotti
- Ciascuna condizione all'interno di una decisione assuma entrambi gli esiti (vero e falso) almeno una volta

Assegnare l'espressione che compone la decisione ad una singola variabile booleana non esime da soddisfare tali condizioni

Esempio: test di copertura di condizioni e decisioni - 3

DO-178 richiede anche di verificare se e come ciascuna condizione può *singolarmente* (senza che le altre varino) determinare l'esito della decisione

Caso	Condizione			Esito
	A=B	C	D>3	
1	T	F	F	F
2	T	T	F	T
3	T	F	T	T
4	F	F	T	F

Certificazione - 7

- Test di copertura
  - Si richiede che il software installato sul sistema destinazione ne soddisfi tutti i requisiti
  - Non si ammette la presenza di componenti software che non corrispondano a requisiti
    - Il rigore di tale prescrizione dipende dal livello di criticità assegnato al software
      - Livello A: corrispondenza tra requisiti e codice eseguibile
      - Livello B: corrispondenza tra requisiti e codice sorgente

Certificazione - 8

- Tecniche di verifica
  - Tracciamento
    - Per determinare completezza, rilevare omissioni, duplicazioni ed elementi superflui
  - Revisioni
    - Formali / informali (come da ISO/IEC 12207)
  - Analisi
    - Statica: applica a requisiti, disegno e codice
    - Dinamica (=test): applica a componenti del sistema od al sistema nella sua interezza

Certificazione - 9

- Analisi statica
  - Analisi di flusso di controllo
    assicura che il programma esegua nell'ordine atteso; che il codice sia ben strutturato; che non vi siano parti del codice irraggiungibili; localizza problemi di terminazione (p.es.: ricorsione, cicli infiniti)
  - Analisi di flusso dei dati
    assicura che nessun cammino di esecuzione acceda a variabili non inizializzate

Certificazione - 10

- Analisi statica (segue)
  - Analisi del flusso dell'informazione
    determina come l'esecuzione di una unità di codice crei dipendenze tra il suo ingresso e la sua uscita

```
X = A+B;        // X dipende da A e B
Y = D-C;        // Y dipende da C e D
if X>0
  Z = (Y+1);    // Z dipende da A, B, C e D
```


Certificazione - 11

- Analisi statica (segue)
  - Verifica formale del codice
    prova che il codice di un programma sia corretto rispetto alla specifica formale dei suoi requisiti (conservazione delle proprietà attese)
  - Verifica di limite (*range check*)
    verifica che i dati manipolati dal programma restino entro i limiti del loro tipo e dell'accuratezza richiesta
    - P.es.: overflow, arrotondamento, limiti di array

Certificazione - 12

- Analisi statica (segue)
  - Analisi d'uso di stack
    [Alcuni linguaggi usano una zona di memoria (detta stack) per consentire a sottoprogrammi di condividere e preservare dati ed informazioni di contesto]
    l'analisi determina se l'ampiezza dello stack disponibile sia sufficiente per l'uso che ne può fare l'esecuzione del programma

Certificazione - 13

- Analisi statica (segue)
  - Analisi temporale
    concerne le proprietà temporali richieste ed esibite dalle dipendenze delle uscite dagli ingressi del programma
  - Analisi di codice oggetto
    dimostra che il codice oggetto da eseguire sia una traduzione corretta del codice sorgente corrispondente e che nessun errore (od omissione) sia stato introdotto dal compilatore