



# **Le Problematiche dell'Object Oriented nel Mondo Safety Critical**

**F. Battini  
IEEE Computer Society  
Italy Chapter**

# Obiettivo dell'Intervento

**Presentazione dei principali problemi sollevati dall'adozione di un disegno ad oggetti per sistemi software safety-critical soggetti a certificazione di prodotto**

**Riferimento principale: la norma  
DO-178B/ED-12B  
*Software Considerations in Airborne Systems  
And Equipment Certification***

**I seguenti punti verranno toccati:**

- **Aspetti relativi allo sviluppo di *Sistemi (Software) safety-critical***
  - **Classificazione degli eventi di avaria**
  - **Fault prevention vs. Fault removal**
- **Le normative di Progetto**
  - **I principi generali alla base delle *consideration* per la safety in campo avionico**
  - **Obiettivi di processo vs requisiti normativi**
- **Le loro implicazioni su un progetto object oriented :**
  - **utilizzo dell'ereditarieta'**
  - **in presenza di polimorfismo**
  - **utilizzo di tool di sviluppo**

# Introduzione al Problema

Normalmente in campo elettronico, si tende a sfruttare al massimo l'ultimo grido tecnologico.

Quando si ha a che fare con sistemi "delicati" dobbiamo sempre porci la seguente domanda:

*" quali tecniche preventive stiamo adottando:  
(1) per evitare di introdurre degli errori di progetto  
(2) e per avere quei margini di sicurezza per cui il sistema rimane "stabile" anche  
in condizioni non previste - e non prevedibili - a priori ?"*

Specialisti di tecnologie devono sapere cosa vuol dire lavorare con dei vincoli diciamo "normativi" che sulla base delle esperienze passate dettano una serie di comportamenti progettuali adatti.

E' questa una situazione tipica in aree di ingegneria piu mature (civile, navale, meccanica, elettrica ecc) dove, per esempio:

- sapere progettare una struttura iperstatica deve essere congiunto con l'applicazione di opportuni criteri di sicurezza, con regole antisismiche delle costruzioni civili ecc.
- sapere disegnare un traghetto veloce deve essere affiancato alle regole di sicurezza marittima del RINA
- saper fare una macchina da 300 km/h deve andare a compromessi con i criteri di viabilita' e del codice stradale

## **L'alternativa e' disporre di un prodotto come:**

- una formula uno ad alto rischio,
- un transatlantico come il Titanic che non ha abbastanza scialuppe.

# **Sviluppo Software di Sistemi Safety-Critical**

# Sistemi Safety Critical

## Sicurezza/Safety & Affidabilita'

**Safety; sicurezza nell'utilizzo di un oggetto**  
*la capacita' di un sistema di essere affidabile rispetto a modalita' critiche di comportamento*

**Security; protezione del contenuto**  
*la capacita' di un sistema di essere protetto da malfunzionamenti intenzionali*

### La Safety ha una definizione probabilistica

Safety. La probabilita' che un sistema non subisca avarie con possibili conseguenze catastrofiche

Il concetto di **Sicurezza/Safety** viene sempre accompagnato dal concetto di **Affidabilita'**.

Tuttavia, esiste una differenza significativa fra i due concetti:

- **l'affidabilita'** e' la probabilita' che un sistema funzioni correttamente (porti a termine il compito per cui e' progettato) per un stabilito intervallo di tempo in condizioni operative stabilite. (ISO 9126)
- **la safety** e' la probabilita' che il prodotto non devii significativamente nel suo comportamento in condizioni complementari a quelle previste

⇒ Deviazioni anche significative dallo scenario operativo di riferimento non innescano comportamenti indesiderati (mishaps).

# Sistemi Safety Critical

## Conseguenze di un'Avaria

La classificazione delle conseguenze e' un punto fondamentale per la definizione della sicurezza

In campo aeronautico, si considerano 5 classi di avaria (failure condition) cosi' descritte:

**Catastrophic:** condizione d'avaria che potrebbe pregiudicare la sicurezza del volo e dell'atterraggio

**Hazardous/Severe-major:** condizione d'avaria che potrebbe ridurre:

- la funzionalità dell'aeromobile per una **drastica** riduzione nei margini di sicurezza; o
- la capacità dell'equipaggio ad ovviare condizioni operative avverse per affaticamento sulle operazioni richieste

**Major:** condizione d'avaria che potrebbe ridurre:

- la funzionalità dell'aeromobile per una significativa riduzione nei margini di sicurezza o
- la capacità dell'equipaggio ad ovviare condizioni operative avverse per incremento del carico di lavoro sulle operazioni richieste.

**Minor:** condizione d'avaria che non riduce significativamente i margini di sicurezza ne' richiede all'equipaggio operazioni al di là del loro addestramento

**No Effect:** condizione d'avaria che non inficia la funzionalità dell'aeromobile ne' richiede all'equipaggio operazioni particolari

# Sistemi Safety Critical

## Analisi di Safety

E' quindi necessario:

- Classificare gli eventi d'avaria per gravita';
- Associare ad ogni classe un livello di criticita' sull'insieme delle funzioni coinvolte;
- Per ogni livello di criticita' definire un profilo di obiettivi di *design assurance* per prevenire errori di progettazione con conseguenze negative sul sistema.

Per le 5 classi di failure condition

Per 5 livelli di criticita'

- **Catastrophic** → **Level A**: funzioni hardware/software il cui malfunzionamento potrebbe causare condizioni d'avaria catastrofiche per l'aeromobile
- **Hazardous/Severe-major** → **Level B**: funzioni hardware/software il cui malfunzionamento potrebbe causare condizioni d'avaria Hazardous/Severe-major per l'aeromobile
- **Major** →
- **Minor** → **Level C**: funzioni hardware/software il cui malfunzionamento potrebbe causare condizioni d'avaria Major per l'aeromobile
- **No Effect** → **Level D**: funzioni hardware/software il cui malfunzionamento potrebbe causare condizioni d'avaria Minor per l'aeromobile
- **Level E**: funzioni hardware/software il cui malfunzionamento non ha conseguenze

Per 4 (+1) profili di design assurance

# Le Normative di Progetto

# Le Normative di Progetto

## La catena di eventi negativi

**Definizioni** [vedi per esempio Laprie].

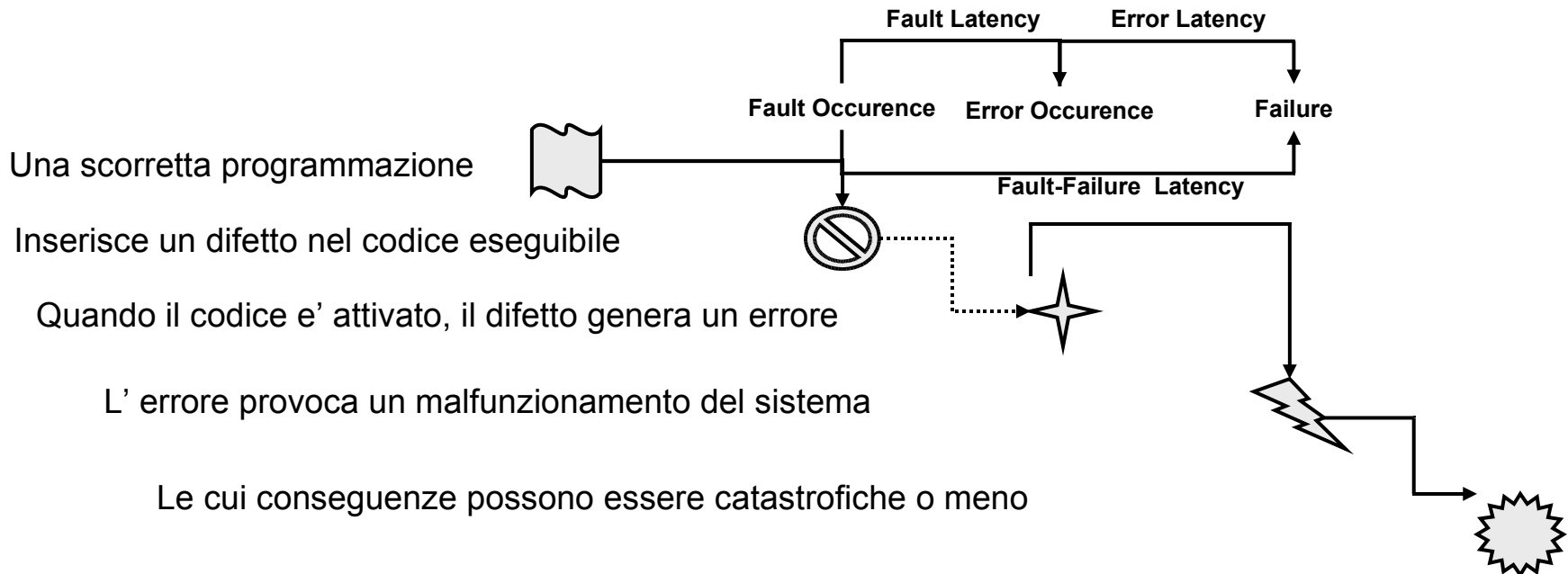
Avaria (Failure). E' la deviazione del sistema dal comportamento specificato (desiderato)

Errore (Error). Inconsistenza fra lo stato specificato di sistema e quello effettivo.

→ Un errore provoca un'avaria

Difetto (Fault). Causa presunta di un errore

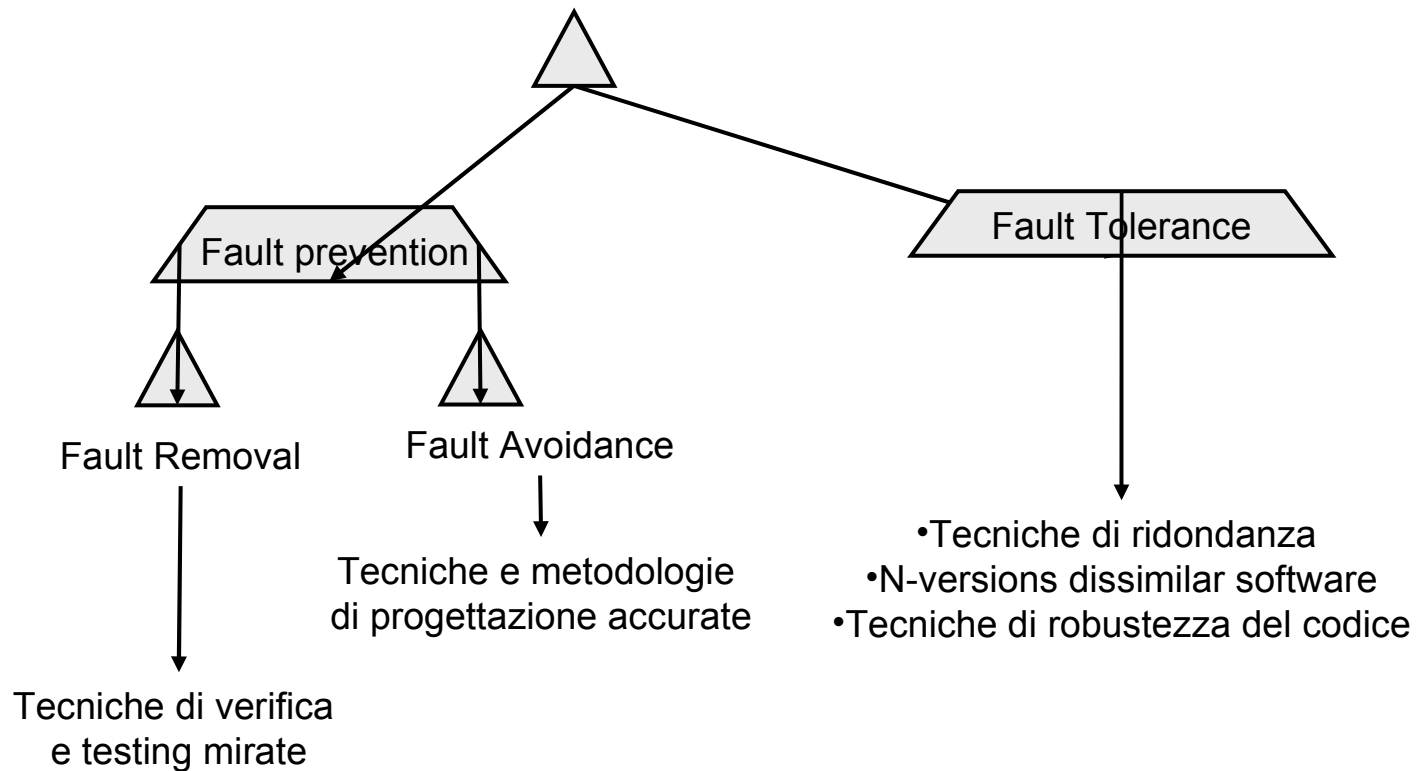
Incidente (Mishaps). Conseguenze di un'avaria



# Le Normative di Progetto

## La catena di eventi negativi

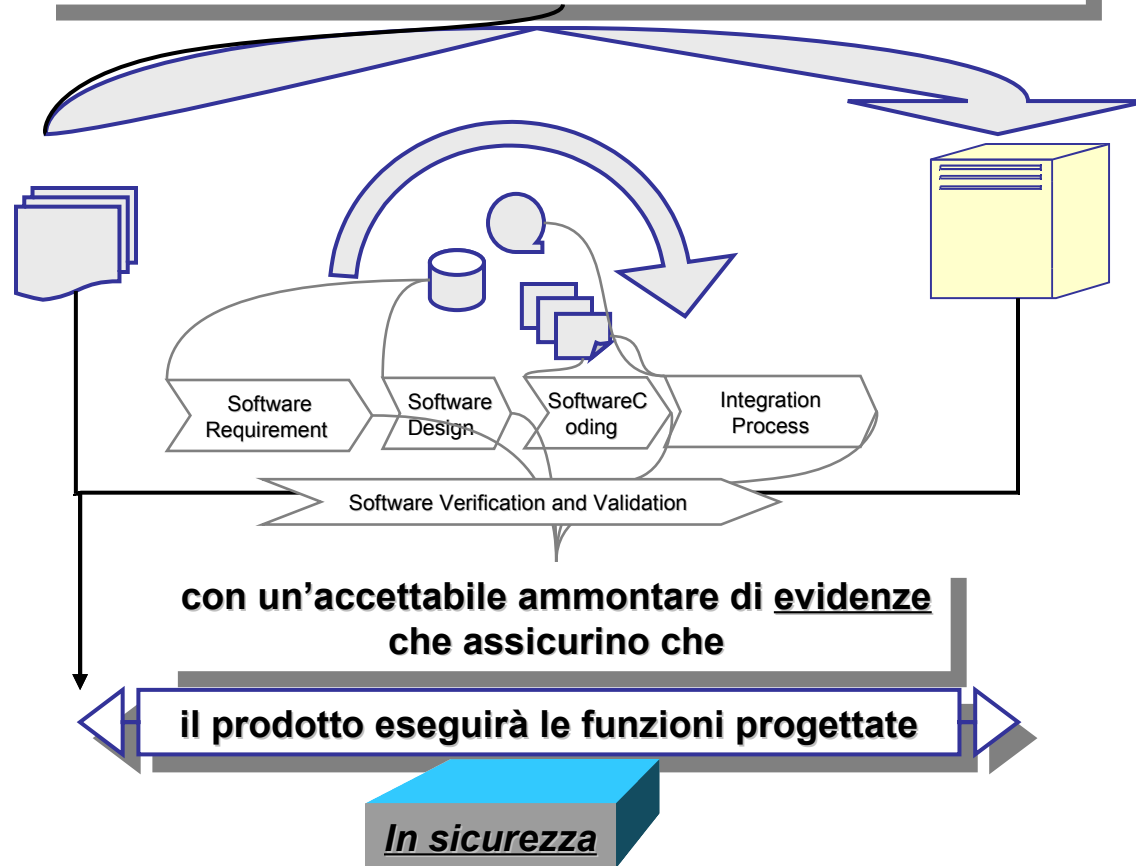
Per spezzare questa catena di eventi negativi una corretta condotta di progetto per i sistemi software safety critical deve dunque basarsi su principi di:



# Le Normative di Progetto sui Processi di Sviluppo

Gli obiettivi dei processi di sviluppo del software sono sintetizzabili come segue:

**convertire requisiti funzionali e normativi in un prodotto**



**con un'accettabile ammontare di evidenze  
che assicurino che**

**il prodotto eseguirà le funzioni progettate**

***In sicurezza***

# Obiettivi di Processo

**La DO-178B fissa una serie di obiettivi da soddisfare**

**Questi sono riferiti all'impostazione dei processi di:**

- **Pianificazione - Software Planning Process**
- **Sviluppo - Software Development Process**
- **Configurazione – S/W Configuration Management Process**
- **Qualità - Software Quality Assurance Process**
- **Certificazione - Certification Liaison Process**

**e alla verifica dei relativi output dai processi di:**

- **Software Requirement Process**
- **Software Design Process**
- **Software Coding & Integration Process**
- **Integration Process**
- **Verification Process**

## I Dati da Produrre

**La DO-178B richiede una serie di dati da produrre come evidenze del raggiungimento degli obiettivi di Design Assurance**

A differenza degli standard basati su DRL (document-driven) dove tutti i progetti devono produrre le stesse tipologie di documentazione, nella DO-178B le evidenze sul processo sono mostrate:

- attraverso documenti a contenuti fissati ma con struttura libera
- attraverso dati estratti da qualche database

ma soprattutto la tipologia dei dati da esporre dipende:

- Dal livello di design assurance prescelto (Level A,B,C,D)
- Dagli obiettivi di processo richiesti per quel livello di design assurance

Passaggio da standard document-driven a guideline process-driven

# Obiettivi di Processo per Livello di Design Assurance

**L'ammontare degli obiettivi di processo da soddisfare, verificare e riportare con adeguate evidenze dipende dal livello di Design Assurance.**

Level A: 66 obiettivi

Level B: 65 obiettivi

Level C: 58 obiettivi

Level D: 28 obiettivi

Con gradi di indipendenza  
fra progettista e verificatore  
via via crescenti

# Impatti Della Metodologia Di Progetto A Oggetti

# Introduzione al Problema

**I vantaggi della tecnologia ad oggetti sono legati essenzialmente:**

- **ad una migliore gestione della complessità**
- **alla riduzione dei costi di sviluppo attraverso il riuso**

**Tuttavia, l'applicazione di un disegno ad oggetti solleva una serie di dubbi (*issue*) sull'effettivo mantenimento degli obiettivi di processo richiesti dal Design Assurance della normativa aeronautica**



# Impatti Della Metodologia Di Progetto A Oggetti

**Nelle fasi di certificazione l'enfasi e' data alle evidenze relative a:**

- **la tracciabilita' dai requisiti fino al codice**
- **la compatibilita' con la piattaforma hardware/software/RTOS**
- **la copertura strutturale dei test**
- **l'individuazione di codice "morto" da eliminare o**
- **la presenza di codice "deattivato" (da segregare)**

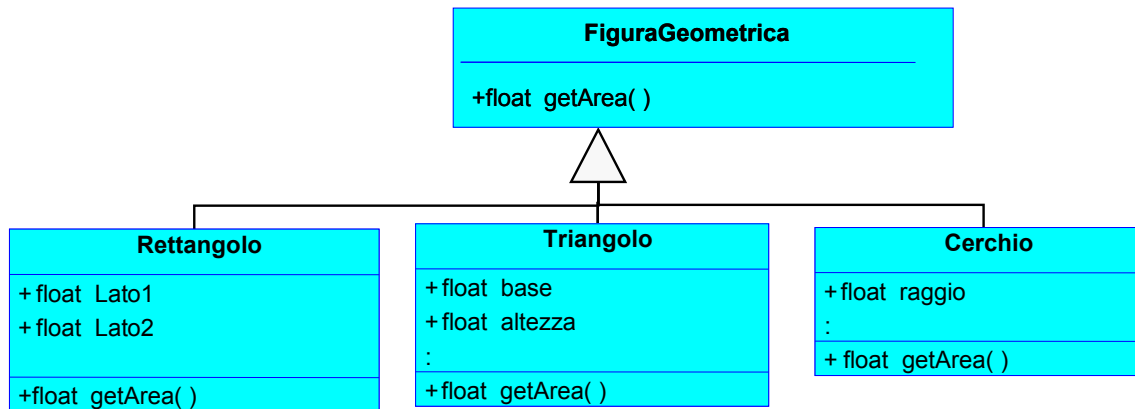
**Le caratteristiche salienti della metodologia ad oggetti devono quindi essere adeguatamente trattate, per evitare:**

- **l'abuso dell'ereditarieta' fra classi**
- **l'ambiguita' introdotta dall'utilizzo di polimorfismo**
- **la presenza di codice non utilizzato da classi "riusate"**
- **la dipendenza da librerie pre-confezionate (foundation class)**
- **meccanismi potenzialmente pericolosi della programmazione ad oggetti da limitare:**
  - ✓ **uso delle `new` : solo in fase di inizializzazione**
  - ✓ **uso del polimorfismo (o dynamic dispatching)**
  - ✓ **uso dell'ereditarieta' multipla**

# Implicazioni dell'Ereditarietà

Ereditarietà: tecnica per estendere la definizione delle interfacce

- ✓ Minimizza la dispersione degli interventi sui vari pezzi di codice che riferiscono un oggetto
- ✓ Concentra le modifiche sul singolo oggetto da utilizzare



*Puntatori ad un'istanza di classe*

```
A: unRettangolo := new Rettangolo(Lato1 => 10; Lato2 => 20);
B: unTriangolo := new Triangolo(base=> 10; altezza => 20);
C: unCerchio := new Cerchio(raggio => 10);
```

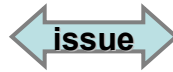
Polimorfismo

```
begin
Surface:=getArea(B.all) + getArea(A.all) +
getArea(C.all);
End;
```

*Reference all'oggetto puntato*

# Implicazioni dell'Ereditarietà

Applicazione dell'ereditarietà

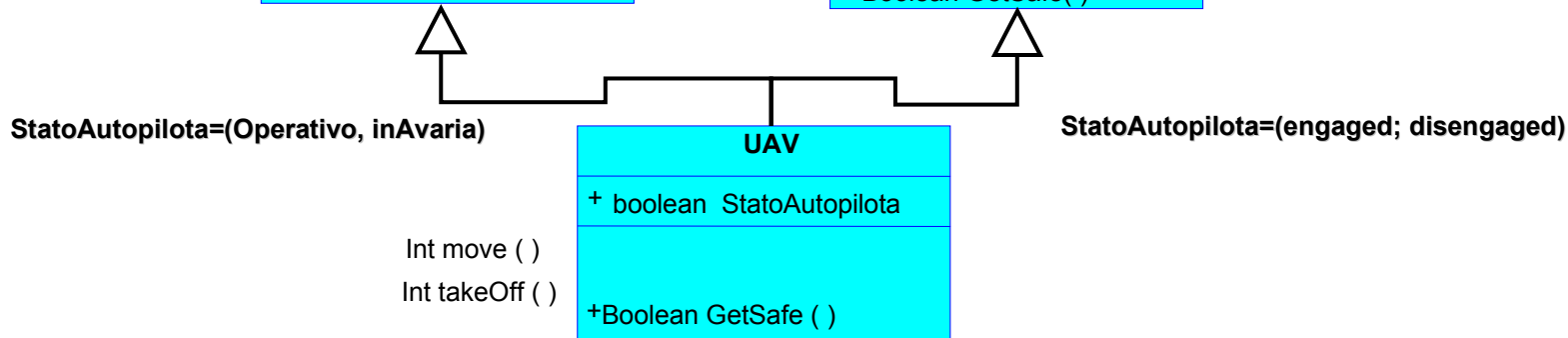
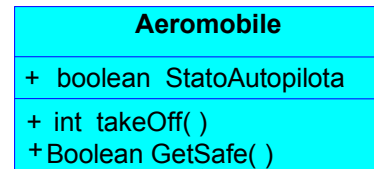
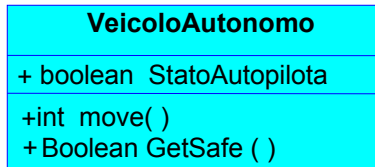


mantenimento della tracciabilità  
Requirements

Design

Code

Test case



Quale StatoAutopilota ?

Quale GetSafe ?

Perche' UAV ha un metodo di move ?

Tracciatura dei requisiti non immediata

Name clashes

Finalita' poco chiare

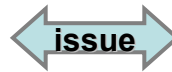
**Che requisiti uso per Testare ?**

# Utilizzo del Dynamic Dispatching

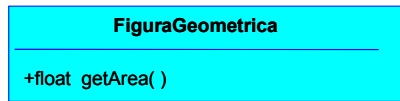
Applicazione dell'ereditarieta'



Polimorfismo



Determinismo del Dynamic Dispatching



Aggiungo una sotto-classe



Polimorfismo



Type pointer is access FiguraGeometrica'Class;  
A : array(1..N) of pointer;

```

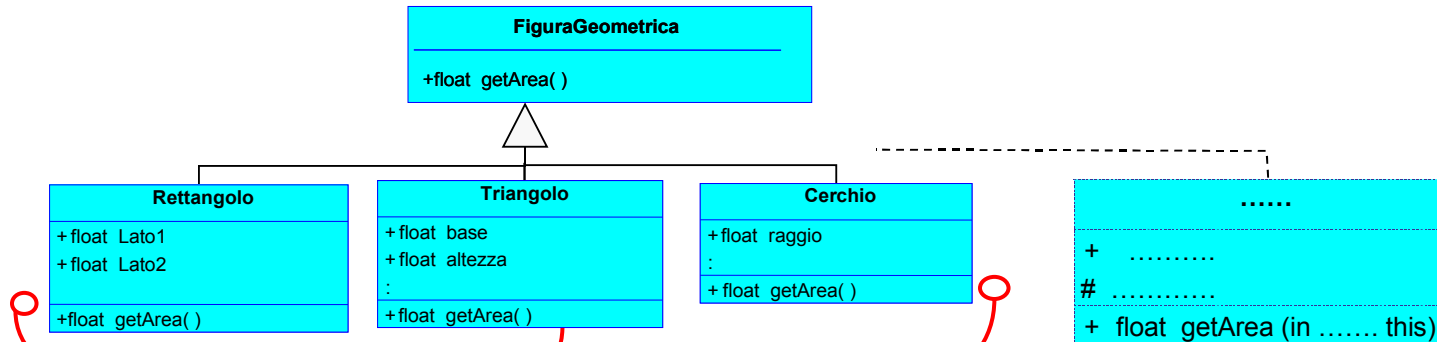
.....
Begin
  for I in A'Range
    Surface := Surface + getArea(A(I).all);
  end loop;
End;
    
```

*Reference all'oggetto puntato*



Funzionera' ancora ?

# Dynamic Dispatching (2)- verifica del determinismo



**Polimorfismo**



**Necessita' di verificare – in isolamento- ogni possibile “branch” di dispatching**

**Necessita' di replicare la verifica in ogni punto di chiamata ?**

**Cosa succede se sto riutilizzando solo il 10% di una superclasse ?**



**Problema del Dead/deactivated code**

# Identificazione di *Dead* o *Deactivated Code*

**Dead Code:** codice eseguibile tale che:

- non puo' essere eseguito in una configurazione operativa dell'ambiente target
- non e' riportabile a nessun requisito software o di sistema

**Se e' presente del dead code, questo e' riconosciuto come conseguenza di un errore di progetto**

**Ma, in un disegno ad oggetti mirato al riuso, l'utilizzo di *foundation class*, *COTS* o *Previously Developed Software* e' una buona practice di progetto**  
**mai conseguenza di un errore progettuale**

**L'introduzione di codice ridondante non utilizzabile (per quello specifico uso) e' certo**



**Trade-off fra:  
benefici introdotti da riuso (riduzione costi e tempi di sviluppo)  
costi aggiuntivi per rimozione del codice in piu'**

## Identificazione di *Dead* o *Deactivated Code* -2

**Deactivated Code:** codice eseguibile tale che:

- viene progettato per essere eseguito in particolari configurazioni operativa dell'ambiente target

(esempi:

codice di bootstrap o manutenzione flash memory

codice di tracing per debugging,

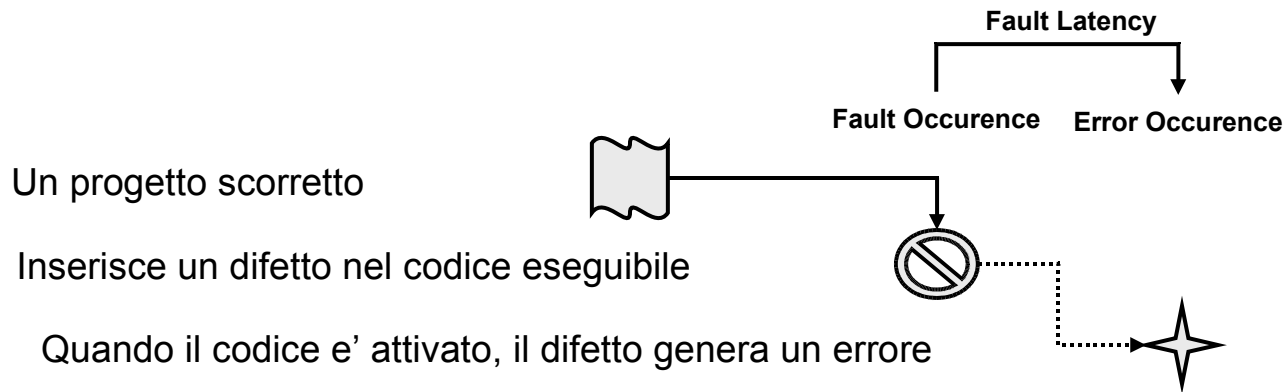
codice che riconosce le diverse configurazioni di hardware)

**Se e' presente del deactivated code, e' necessario dare evidenze che:**

- **non sia codice morto**
- **non ci sia possibilita' di attivarlo per errore**  
(es.: segregazione su banchi di memoria selezionabili)

# Tool di Sviluppo

**Negli ambienti di sviluppo moderni moltissime attività di progetto per software sono attualmente eseguite per mezzo di strumenti informatici che coprono dal disegno concettuale fino alla sintesi di codice sorgente o alla simulazione comportamentale del sistema.**



**Diventa quindi critico accertarsi che gli strumenti automatici non siano essi stessi affetti da errori nella produzione dei lavorati su cui si basera' tutta l'implementazione**

# Tool di Sviluppo: Valutazione E Qualifica

**Rischio da evitare: gli output di questi strumenti contengano un errore che si rifletta in un difetto implementativo che possa causare un avaria**

**La qualifica dei tool e' richiesta quando dei processi descritti dalla norma sono**

- **eliminati,**
- **ridotti o**
- **automatizzati**

**ricorrendo all'uso di strumenti e senza che il loro output sia sottoposto a verifica come descritto dalla sezione 6**

**Obiettivo del processo di valutazione e qualifica dei tool e' di assicurarsi che lo strumento sia capace di fornire la stessa confidenza dei processi eliminati,ridotti o automatizzati**

# La Valutazione e Qualifica Dei Tool

***Software tool*** possono essere classificati come:

Tool di sviluppo software

tool il cui prodotto in uscita entra a far parte del sistema avionico e che possono introdurre difetti (fault)

Esempio: un generatore di codice.

Il tool deve essere qualificato se il codice generato non e' sottoposto a verifica

Tool di verifica software

tool che non possono introdurre difetti. Ma che possono fallire il loro rilevamento

Esempio: un instrumentatore di codice per la copertura strutturale

Il tool deve essere qualificato se l'output generato non e' sottoposto a verifica da un'altra attivita'

**Solo tool deterministici possono essere sottoposti a qualifica allo stesso livello del software prodotto attraverso l'emissione di:**

- un ***Tool Qualification Plan***
- un documento di ***Tool Operational Requirements***
- evidenze di conformita' dello strumento con il TOR
- coperture strutturali e di requisito nonche' test di robustezza appropriati per il livello identificato
- un ***Tool Accomplishment Summary*** che riporti le evidenze del raggiungimento degli stessi obiettivi del Design Assurance Level del software prodotto

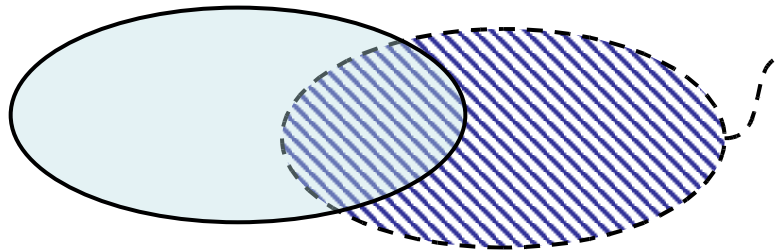
# Domande

# Sistemi Safety Critical

## Safety & Affidabilita'

In realta' per la definizione stessa di Affidabilita' (la probabilita' che un sistema funzioni correttamente per un stabilito intervallo di tempo) fa riferimento a "condizioni operative stabilite"

Ma non e' detto che le condizioni operative stabilite dal progetto corrispondano alle reali condizioni operative del sistema mentre lavora

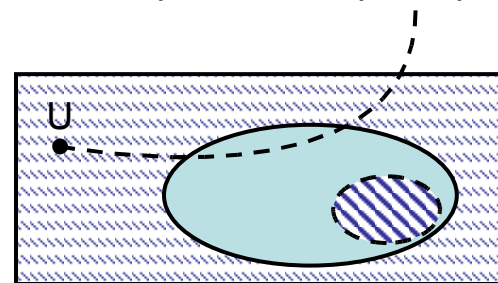


Probabilita' di Avaria  $F(t)$ ;  
malfunzionamento rispetto alle specifiche

Probabilita' di Avaria con  
conseguenze disastrose  $D(t)$

considerando tutti i  
possibili utilizzi !

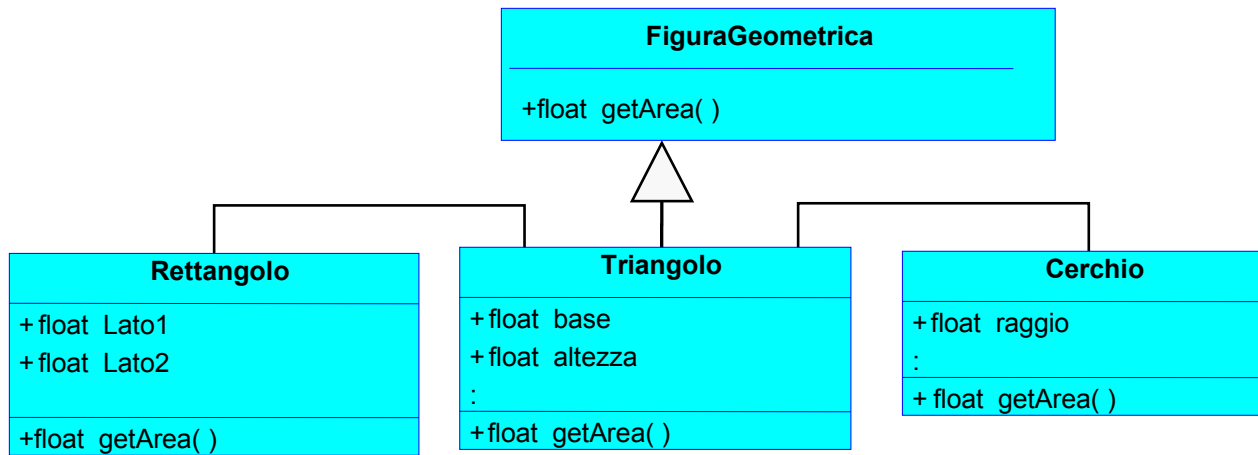
utilizzi non specificati e quindi potenzialmente disastrosi



utilizzi specificati  
potenzialmente disastrosi

utilizzi specificati sicuri

La stessa operazione puo' essere applicata a piu' classi differenti inserite nella stessa gerarchia: *polimorfismo*



In questo caso il *polimorfismo* viene risolto a compile time.

*Puntatori ad un'istanza di classe*

```
A: unRettangolo :=new Rettangolo(Lato1 => 10; Lato2 => 20);
B: unTriangolo := new Triangolo(base=> 10; altezza => 20);
C: unCerchio := new Cerchio(raggio => 10);
```

```
begin
```

```
Surface:=getArea(B.all) + getArea(A.all) + getArea(C.all);
```

```
End;
```

# Sistemi Safety Critical

## Analisi di Safety secondo Def-Stan 00-56

Formulation of the Safety Analysis Tables

Table 1

Example Equivalent Numerical Probabilities

<b>Probability</b>	<b>Numerical Equivalent</b>
Frequent	$10000 \times 10^{-6}/\text{operating hour}$
Probable	$100 \times 10^{-6}/\text{operating hour}$
Occasional	$1 \times 10^{-6}/\text{operating hour}$
Remote	$0.01 \times 10^{-6}/\text{operating hour}$
Improbable	$0.0001 \times 10^{-6}/\text{operating hour}$
Incredible	$0.000001 \times 10^{-6}/\text{operating hour}$

Table 3

Probability Ranges

<b>Accident Frequency</b>	<b>Occurrence</b> during <i>operational life</i> <i>considering all instances of the system</i>
Frequent	Likely to be continually experienced
Probable	Likely to occur often
Occasional	Likely to occur several times
Remote	Likely to occur some time
Improbable	Unlikely, but may exceptionally occur
Incredible	Extremely unlikely that the event will occur at all, given the assumptions recorded about the domain and the system

# Sistemi Safety Critical

## Analisi di Safety secondo Def-Stan 00-56

### Formulation of the Safety Analysis Tables

Table 2  
Accident severity categories

<b>Category</b>	<b>Definition</b>
Catastrophic	Multiple deaths
Critical	A single death; and/or multiple severe injuries or severe occupational illnesses
Marginal	A single severe injury or occupational illness; and/or multiple minor injuries or minor occupational illnesses
Negligible	At most a single minor injury or minor occupational illness

Table 4  
Risk Class Definitions

<b>Risk class</b>	<b>Interpretation</b>
Class A	intolerable
Class B	undesirable, and shall only be accepted when risk reduction is impracticable
Class C	tolerable with the endorsement of the Project Safety Review Committee
Class D	tolerable with the endorsement of the normal project reviews

# Sistemi Safety Critical

## Analisi di Safety secondo Def-Stan 00-56

### Formulation of the Safety Analysis Tables

Table 5  
Example Risk Classification Scheme

	Catastrophic	Critical	Marginal	Negligible
Frequent	A	A	A	B
Probable	A	A	B	C
Occasional	A	B	C	C
Remote	B	C	C	D
Improbable	C	C	D	D
Incredible	C	D	D	D

Table 6  
Claim Limits

Safety Integrity Level	Minimum failure rate
S4	Remote
S3	Occasional
S2	Probable
S1	Frequent