Ingegneria del Software

Verifica e Convalida

Antonino Staiano
e-mail: **antonino.staiano@uniparthenope.it**

Introduzione

- "The software is done. We are just trying to get it to work…"
- Finito di scrivere il codice, possiamo dire di aver finito lo sviluppo del software?
- Guardando il passato, qualche esempio ci dice che le cose non stanno proprio così …
  - Gli errori possono succedere in ogni fase del ciclo di vita!

I bugs…

- Nella storia dell'informatica ci sono innumerevoli casi di sistemi con comportamenti che deviano dalle specifiche
  - Primo caso di bug (e debug) della storia:
    - Una falena (bug) entrò nel Pannello F, Relay #70 del computer Harvard Mark II, nel 1945, creando un cortocircuito. Il computer iniziò a sbagliare addizioni e moltiplicazioni.
    - La falena (ben cotta?) fu presa (de-bug), ed attaccata sul logbook del computer con dello scotch, con la scritta "primo caso di bug trovato" ("first actual case of a bug being found")

Qualche "errore" del errore passato

- Errori di progettazione
- NASA Mars Climate Orbiter
  - La sonda spaziale Mars Climate Orbiter, costata 125 milioni di dollari, fu persa il 23 settembre 1999, mentre entrava nell'orbita di Marte.
  - Si scoprì successivamente che la causa era in un errore insito nelle specifiche del software adottate dalla NASA.
  - Durante il processo di sviluppo, incredibilmente, gli analisti non avevano specificato il sistema di riferimento da usare per la gestione dei motori usati nell'atterraggio
  - Come risultato, un team di sviluppo utilizzò il sistema Imperiale (pounds), mentre un altro il sistema Metrico (Kg)
  - Quando furono passati i parametri da un modulo sw all'altro, non fu operata conversione dei dati, con il risultato di dare direzioni di volo completamente errate, che portarono allo schianto della sonda

Qualche "errore" del errore passato

- Errori di coding
- ESA Ariane 5
  - Il vettore spaziale Ariane 5 esplose durante il suo volo inaugurale, il 4 Giugno 1996.
  - Per la gestione dei motori, l'Ariane 5 utilizzava il codice funzionante correttamente sul suo predecessore Ariane 4.
  - Tra questi, utilizzava una routine aritmetica di conversione da un floating point a 64bit ad un intero a 16 bit
  - I valori generati dai motori dell'Ariane 4 erano gestibili da interi a 16bit.
  - I motori dell'Ariane 5, più potenti, generavano valori più grandi.
  - Il conseguente overflow durante la conversione portò al crash sia del sistema primario che secondario di navigazione, con la conseguente esplosione del razzo

Qualche "errore" del errore passato

- Errori nella gestione dell'input dell'utente
- USS Yorktown CG-48
  - Nel 1998, durante delle esercitazioni in mare aperto, un sottoufficiale dell'incrociatore lanciamissili USS Yorktown inserì per sbaglio uno zero in un form del Data Base Manager di bordo
  - Ciò portò ad una divisione per zero
  - L'errore bloccò tutti i terminali della LAN della nave, arrivando allo shutdown di tutti i sistemi di bordo, comprese le turbine
  - La nave restò completamente immobilizzata per ore, in pieno oceano, perché un form per l'inserimento di dati in un DB non verificava la correttezza dell'input inserito dall'utente

Qualche "errore" del errore passato

- Errori di valutazione sull'utilizzo del sistema
- NASA Spirit Rover
  - Il *robottino* per l'esplorazione di Marte "Spirit" improvvisamente cessò di comunicare con la Terra il 21 Gennaio 2004, qualche giorno dopo l'atterraggio
  - Nei giorni successivi, i tecnici della NASA riuscirono a scoprire che il robot era in Fault mode, e si riavviava in continuazione
  - Poiché l'errore si ripresentava ad ogni reboot, i tecnici capirono che era un problema della Flash Memory, o una rottura hardware
  - Isolando la Flash attraverso un comando speciale, il robot si riavviò senza problemi.
  - Dopo 3 giorni, il team capì l'errore: il robot aveva loggato i principali parametri sia durante il volo fino a Marte, che durante i test al suolo. Come risultato, aveva creato un enorme numero di files, che portava al crash del modulo che si occupava di gestire il File System.
  - Dopo aver riformattato la Flash Memory, il robottino ha funzionato correttamente per altri 2 anni

Scopo della lezione

- Introdurre la verifica e la convalida del software e discuterne la differenza
- Descrivere il processo di ispezione ed il suo ruolo nella verifica e convalida
- Capire cos'è l'analisi statica automatica e come viene utilizzata nei processi di verifica e convalida
- Capire come la verifica statica viene utilizzata nei processi di verifica e convalida

Verifica vs Convalida

- Verifica:
  "Stiamo costruendo il prodotto nel modo giusto?".
  - Il software deve essere conforme alla sua specifica
- Convalida:
  "Stiamo costruendo il prodotto giusto?".
  - Il software dovrebbe fare ciò che realmente chiede l'utente

Il processo di verifica e convalida

- E' un processo che si protrae durante tutto il ciclo di vita – la verifica e la convalida deve essere applicata ad ogni fase del processo software
- Ha due obiettivi primari
  - La scoperta di difetti in un sistema
  - La valutazione dell'utilità e della usabilità del sistema in un contesto operativo

Obiettivi della verifica e della convalida

- La verifica e la convalida dovrebbe stabilire con un certo grado di confidenza che il software è adatto al suo scopo
- Ciò non significa che esso sia completamente privo di difetti
- Piuttosto, deve essere sufficientemente buono per l'uso a cui è destinato ed il tipo di utilizzo determinerà il grado di confidenza necessario

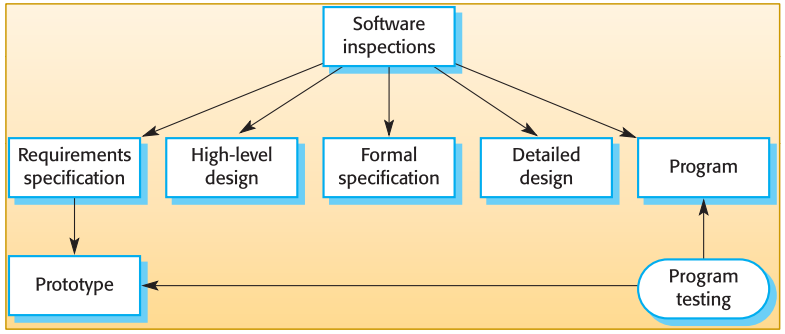
Confidenza nella verifica e nella convalida

- Dipende dallo scopo del sistema, dalle attese dell'utente e dall'ambiente corrente di mercato del sistema
  - Funzione del software
    - Il livello di confidenza dipende dalla criticità del sistema nell'organizzazione
  - Attese dell'utente
    - Gli utenti possono avere scarse attese su certi tipi di software
  - Ambiente di mercato
    - Introdurre un prodotto nel mercato in anticipo può essere più importante che trovare i difetti in un programma

Verifica statica e dinamica

- Ispezioni del software. Riguarda l'analisi della rappresentazione statica del sistema per individuare i problemi (verifica statica)
  - Consiste nell'analisi dei documenti e del codice sorgente
- Testing del software. Riguarda l'uso e l'osservazione del comportamento del prodotto (verifica dinamica)
  - Il sistema viene eseguito con dati di test ed è esaminato il suo comportamento operativo

Verifica - Convalida statica e dinamica



Ispezioni

- Le tecniche di ispezione comprendono
  - Ispezioni del programma
  - Analisi automatica del codice sorgente
  - Verifica formale
- Queste tecniche statiche possono controllare solo la corrispondenza tra un programma e le sue specifiche (verifica)
  - Non possono dimostrare che il funzionamento del programma sia efficiente e controllare aspetti come prestazioni ed affidabilità

Testing del programma

- Può rivelare la presenza di errori e NON la loro assenza
- Rappresenta la sola tecnica di validazione per i requisiti non funzionali poiché il software deve essere eseguito per vedere come si comporta
- Dovrebbe essere usata congiuntamente alla verifica statica per fornire una copertura completa della verifica e della convalida

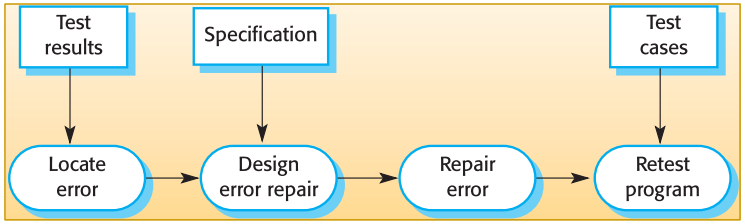
Tipi di testing

- Testing dei difetti
  - Test progettati per scoprire i difetti del sistema
  - Un test dei difetti che ha successo rivela la presenza di difetti nel sistema
- Test di convalida
  - Inteso a mostrare che il software soddisfi i suoi requisiti
  - Un test che riesce mostra che un requisito è stato implementato in modo consono

Testing e debugging

- I processi di testing dei difetti e di debugging sono distinti
- La verifica e la convalida riguardano lo stabilire l'esistenza dei difetti in un programma
- Il debugging riguarda la localizzazione e la riparazione di tali errori
- Il debugging comporta la formulazione di un'ipotesi sul comportamento del programma e successivamente il testing di queste ipotesi per trovare l'errore del sistema

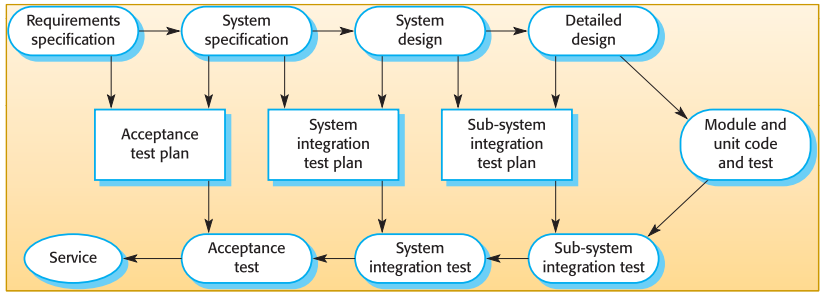
Il processo di debugging



Pianificazione della verifica e della convalida

- Il processo di verifica e convalida è costoso
- E' necessaria un'attenta pianificazione per ottenere il massimo dai processi di ispezione e di testing
- La pianificazione dovrebbe partire nelle fasi iniziali del processo di sviluppo
- Il piano dovrebbe stabilire il giusto equilibrio tra la verifica statica ed il testing
- La pianificazione del test concerne la definizione di standard per il processo di testing non solo la descrizione dei test del prodotto

Il modello a V dello sviluppo



La struttura di un piano di test del software

- Il processo di test
- Tracciabilità dei requisiti
- Unità testate
- Tempistica del test
- Procedure di registrazione dei test
- Requisiti HW e SW
- Vincoli

Il piano di test del software

Il processo di test
Una descrizione delle principali fasi del processo di test

Tracciabilità dei requisiti
Gli utenti sono interessati per lo più al fatto che il sistema soddisfi i requisiti e il test deve essere pianificato in modo che tutti i requisiti siano testati singolarmente
Oggetti testati
Occorre specificare il prodotti del processo software che si stanno testando
Tempistica del test
Una tempistica generale e la relative assegnazione delle risorse, ovviamente collegata alla tempistica generale dello sviluppo del progetto
Procedure di registrazione dei test
Non è sufficiente eseguire semplicemente i test; i risultati devono essere registrati sistematicamente e deve essere possibile controllare il processo per verificare che sia stato eseguito correttamente
Requisiti HW e SW
Questa sezione descrive gli strumenti SW necessari e l'utilizzo stimato dello HW
Vincoli
In questa sezione si prevedono i vincoli che potrebbero influire sul processo di test, come la carenza di personale

Ispezioni del software

- Coinvolge il personale nell'esame della rappresentazione dei codici sorgenti per scoprire anomalie e difetti
- Le ispezioni non richiedono l'esecuzione di un sistema pertanto possono essere usate prima dell'implementazione
- Possono essere applicate a qualsiasi rappresentazione del sistema (requisiti, progetto, dati di configurazione, dati di test, etc.)
- E' stato mostrato che sono efficaci nell'individuazione di errori di programma

Vantaggi delle ispezioni

- Utilizzano la conoscenza del sistema del suo dominio di applicazione e del linguaggio di programmazione in modo che i revisori conoscono i tipi di errori che si verificano solitamente
- Molti difetti differenti possono essere scoperti in una ispezione singola
  - Nel testing, invece, un difetto può mascherarne un altro pertanto sono necessarie molteplici esecuzioni
- E' possibile ispezionare versioni incomplete di un sistema senza costi aggiuntivi
  - In fase di testing, invece, se un programma è incompleto è necessario sviluppare una struttura di test specializzata per testare le parti disponibili
- Le ispezioni possono esaminare anche altri attributi di qualità
  - Conformità agli standard, portabilità e manutenibilità

Ispezioni e testing

- Le ispezioni ed il testing sono tecniche di verifica complementari e non contrastanti
- Entrambe dovrebbero essere usate durante il processo di verifica e convalida
- Le ispezioni possono controllare la conformità ad una specifica ma non la conformità ai requisiti reali del cliente
- Le ispezioni non possono controllare le caratteristiche non funzionali come le prestazioni, l'usabilità, ecc.

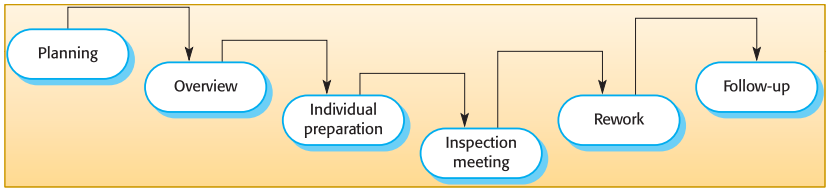
Ispezioni del programma

- Approccio formale alla revisione di documenti
- Intese esplicitamente per l'individuazione dei difetti (non la correzione)
- I difetti possono essere errori logici, anomalie nel codice che potrebbero indicare una condizione erronea (ad esempio, una variabile non inizializzata) o una non conformità agli standard

Pre-condizioni per le ispezioni

- Deve essere disponibile una specifica precisa
- I membri del team devono avere familiarità con gli standard dell'organizzazione
- Deve essere disponibile codice sintatticamente corretto o altre rappresentazioni del sistema
- Dovrebbe essere preparata un lista di controllo degli errori
- Il management deve accettare che l'ispezione incrementerà i costi all'inizio nel processo del software
- Il management non dovrebbe usare le ispezioni per valutare lo staff (trovare chi ha commesso gli errori)

Il processo di ispezione



Procedura di ispezione

- Panoramica del sistema presentata al team di ispezione
- Il codice ed i documenti associati sono distribuiti al team di ispezione in anticipo
- Ha luogo l'ispezione e sono osservati gli errori scoperti
- Sono fatte le modifiche per riparare gli errori scoperti
- Una re-ispezione può non necessariamente essere richiesta

Ruoli di ispezione

Autore o proprietario	Il programmatore o il progettista responsabile di produrre il programma o il documento. E' responsabile anche della correzione dei difetti scoperti durante il processo di ispezione
Ispettore	Trova errori, omissioni e inconsistenze nei programmi e nei documenti; può anche identificare problemi più vasti che esulano dal campo d'azione del team
Lettore	Presenta il codice o il documento in una riunione di ispezione
Segretario	Registra i risultati delle riunioni di ispezione
Chairman o moderator	Gestisce il processo e facilita l'ispezione. Riporta i risultati del processo al moderator capo
Moderatore capo	Responsabile dei miglioramenti nel processo di ispezione, dell'aggiornamento delle liste di controlli, dello sviluppo degli standard, ecc.

Liste di controllo delle ispezioni

- Dovrebbe essere usata una lista di controllo di errori comuni per guidare l'ispezione
- Le liste di controllo degli errori dipendono dal linguaggio di programmazione e riflettono gli errori caratteristici che possono verificarsi verosimilmente nel linguaggio
- In generale, più debole è il controllo di tipo più ampia è la lista di controllo
- Esempi: Inizializzazione, nomi di costanti, terminazione di ciclo, limiti di array, ecc.

Controlli di ispezione 1

Errori nei dati	Tutte le variabili del programma sono inizializzate prima che il loro valore sia utilizzato? Tutte le costanti hanno avuto un nome? Il limite superiore degli array è uguale alla loro dimensione o alla loro dimensione-1? Se si utilizzano stringhe di caratteri, viene assegnato esplicitamente un delimitatore? Ci sono possibilità di buffer overflow?
Errori di controllo	Per ogni istruzione condizionale la condizione è corretta? E' certo che ogni ciclo sarà terminato? LE istruzioni composte sono correttamente messe fra parentesi? Se è necessario un break dopo ogni caso nelle istruzioni case, è stato inserito?
Errori di Input/output	Sono utilizzate tutte le variabili di input? A tutte le variabili di output viene assegnato un valore prima che siano restituite? Input imprevisti possono causare corruzione?

Controlli di ispezione 2

Errori di interfaccia	Tutte le chiamate a funzione e a metodo hanno il giusto numero di parametri? Il tipo di parametri formali e reali corrisponde? I parametri sono nel giusto ordine? Se i componenti accedono alla memoria condivisa, hanno lo stesso modello di struttura per questa?
Errori nella gestione della memoria	Se una struttura collegata viene modificata, tutti i collegamenti sono correttamente riassegnati? Se si utilizza la memoria dinamica, lo spazio è assegnato correttamente? Lo spazio viene esplicitamente deallocato quando non è più richiesto?
Errori di gestione delle eccezioni	Sono state prese in considerazione tutte le possibili condizioni di errore?

Tassi di ispezione

- 500 istruzioni/ora durante la panoramica
- 125 istruzioni/ora durante la preparazione individuale
- 90-125 istruzioni/ora possono essere ispezionate
- L'ispezione è dunque un processo costoso

Analisi statica automatica

- Per alcuni errori ed euristiche è possibile automatizzare il processo di controllo dei programmi rispetto alle liste di controllo degli errori più comuni
- Gli analizzatori statici sono strumenti software per l'elaborazione di codice sorgente
- Effettuano il parsing del testo del programma e cercano di scoprire potenziali condizioni di errore e portarle all'attenzione del team di verifica e convalida
- Sono molto efficaci come supporto alle ispezioni e dunque sono un supplemento delle ispezioni e non un sostituto

Controlli delle analisi statiche

Classe di errore	Controllo di analisi statiche
Errori nei dati	Variabili utilizzate prima dell'inizializzazione Variabili dichiarate e mai utilizzate Variabili assegnate due volte ma mai utilizzate tra le assegnazioni Possibili violazioni dei limiti degli array Variabili non dichiarate
Errori di controllo	Codice irraggiungibile Salti incondizionati all'interno dei cicli
Errori di input/output	Variabili restituite più volte senza un'assegnazione che s'interpone
Errori di interfaccia	Errore nel tipo di parametri Errore nel numero di parametri Non utilizzo dei risultati di una funzione Funzioni e procedure mai chiamate
Errori nella gestione della memoria	Puntatori non assegnati Aritmetica dei puntatori

Fasi di analisi statica

- Analisi del flusso di controllo. Controlla i cicli con uscite o punti di ingresso multipli, trova codice non raggiungibile, ecc.
- Analisi di uso dei dati. Individua variabili non inizializzate, variabili scritte due volte senza un assegnamento, variabili dichiarate ma mai usate, ecc.
- Analisi delle interfacce. Controlla la consistenza delle dichiarazioni di e del loro utilizzo

Fasi di analisi statica

- Analisi del flusso delle informazioni. Identifica le dipendenze delle variabili di output. Non individua anomalie ma evidenzia le informazioni per l'ispezione del codice o le revisioni
- Analisi dei percorsi. Identifica i percorsi attraverso il programma e avvia le istruzioni eseguite nel percorso. Ancora, potenzialmente utile nel processo di revisione

Uso dell'analisi statica

- Particolarmente rilevante quando un linguaggio come il C è usato che ha una tipizzazione debole e quindi molti errori non sono individuati dal compilatore
- Meno efficiente nei costi per linguaggi come Java che hanno una forte controllo di tipo e possono pertanto individuare molti errori durante la compilazione
- Non può sostituire le ispezioni: alcuni tipi di errore non possono essere individuati dagli analizzatori automatici
  - inizializzazioni errate; argomenti di funzioni del tipo giusto ma errati

Verifica e metodi formali

- I metodi formali possono essere usati quando è prodotta una specifica formale del sistema
- Essi sono la tecnica di verifica formale fondamentale
- Comportano analisi matematiche dettagliate della specifica a possono sviluppare argomenti formali sul fatto che che un programma sia conforme alla propria specifica matematica

Argomenti per i metodi formali

- Produrre una specifica matematica richiede un'analisi dettagliata dei requisiti e ciò può portare a scoprire gli errori
- Essi possono individuare errori di implementazione prima del testing quando il programma è analizzato durante la specifica

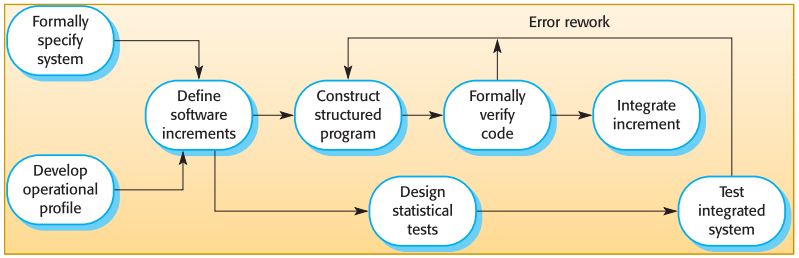
Argomenti contro i metodi formali

- Richiedono notazioni specialistiche che non possono essere capite dagli esperti del dominio
- Molto costoso sviluppare una specifica e anche più costoso mostrare che un programma soddisfa la specifica
- Può essere possbile raggiungere lo stesso livello di confidenza in un programma in maniera più economica usando altre tecniche di verifica e convalida

Sviluppo software Cleanroom

- Il nome deriva dal processo di fabbricazione dei semiconduttori. La filosofia è evitare il difetto piuttosto che la sua rimozione
- Questo processo di sviluppo è basato su:
  - Sviluppo incrementale
  - Specifica formale
  - Verifica statica usando argomenti di correttezza
  - Test statistico per determinare l'attendibilità del programma

Il processo Cleanroom



Caratteristiche del processo Cleanroom

- Specifica formale che usa un modello di transizione di stato
- Sviluppo incrementale dove il cliente impone le priorità degli incrementi
- Programmazione strutturata – sono usati nel programma costrutti limitati di controllo ed astrazione
- Verifica statica che usa ispezioni rigorose. Non c'è un test di unità o dei moduli per il codice dei componenti
- Test statistico del sistema

Specifiche formali ed ispezioni

- Come specifica del sistema si adotta un modello basato sullo stato ed il processo di ispezione controlla il programma rispetto a questo modello
- L'approccio di programmazione è definito in modo che la corrispondenza tra il modello ed il sistema sia chiara
- Sono usati argomenti matematici (non dimostrazioni) per incrementare la confidenza nel processo di ispezione

Team del processo cleanroom

- Team della specifica. Responsabile per lo sviluppo ed il mantenimento della specifica del sistema
- Team di sviluppo. Responsabile per lo sviluppo e la verifica del software. Il software non è eseguito o anche compilato durante questo processo
- Team di certificazione. Responsabile per lo sviluppo di un insieme di test statistici per provare il software dopo lo sviluppo. Sono usati modelli di crescita dell'attendibilità per determinare quando essa è accettabile

Valutazione del processo cleanroom

- I risultati dell'impiego del processo Cleanroom sono stati notevoli. Hanno messo in evidenza la presenza di pochi errori nei sistemi rilasciati
- Diversi tipi di valutazioni mostrano come il processo non sia più costoso rispetto ad altri
- Meno errori rispetto ad altri processi di sviluppo tradizionali
- Tuttavia, il processo non è usato diffusamente. Il problema consiste nell'adattare l'approccio in organizzazioni in cui ci sono meno competenze specifiche e ingegneri meno motivati

Riassumendo (I)

- La verifica e la convalida non sono la stessa cosa. La verifica mostra la conformità alla specifica; la convalida mostra che il programma soddisfi le necessità del cliente
- Dovrebbero essere definiti dei piani di test per guidare il processo di testing
- Le tecniche di verifica statica comportano l'esame e l'analisi del programma per l'individuazione di errori

Riassumendo (II)

- Le ispezioni dei programmi sono molto efficienti nello scoprire gli errori
- Il codice sorgente nelle ispezioni è controllato in modo sistematico da un piccolo team
- Gli strumenti di analisi statica possono trovare anomalie nel programma che sono indicative di errori nel codice
- Il processo di sviluppo Cleanroom dipende dallo sviluppo incrementale, dalla verifica statica e dal testing statistico