

Università degli Studi di Padova

Verifica e validazione: analisi statica

Verifica e validazione: analisi statica



Anno accademico 2007/8
Ingegneria del Software mod. A

Tullio Vardanega, tullio.vardanega@math.unipd.it

Corso di Laurea Triennale in Informatica, Università di Padova 1/28

Università degli Studi di Padova

Verifica e validazione: analisi statica

Premessa – 1

- **Un numero crescente di sistemi *software* svolge funzioni con elevate caratteristiche di criticità**
 - **Sicurezza come *safety***
 - Prevenzione di condizioni di pericolo a persone o cose
 - P.es.: sistemi di trasporto pubblico
 - P.es.: sistemi per il supporto di transazioni finanziarie
 - **Sicurezza come *security***
 - Prevenzione di intrusione
 - P.es.: sistemi per il trattamento dei dati personali
 - P.es.: sistemi per lo scambio di informazioni riservate

Corso di Laurea Triennale in Informatica, Università di Padova 2/28

Università degli Studi di Padova

Verifica e validazione: analisi statica

Premessa – 2

- **Il *software* in uso in tali sistemi deve**
 - **Esibire attributi (o capacità) funzionali**
 - Determinano **cosa** il sistema deve fare
 - **Possedere caratteristiche non funzionali**
 - Determinano **come** ciò deve essere fatto
 - **Soddisfare proprietà o requisiti**
 - Di costruzione
 - D'uso
 - Di funzionamento

Corso di Laurea Triennale in Informatica, Università di Padova 3/28

Università degli Studi di Padova

Verifica e validazione: analisi statica

Premessa – 3

- **Il soddisfacimento di tutte le legittime aspettative deve essere accertato prima dell'abilitazione all'uso del sistema**
- **Accertamento mediante verifica**

Verifica conferma, mediante prova o presentazione di evidenza oggettiva, che determinati requisiti siano stati soddisfatti

ISO 8402:1994 *Quality management and quality assurance - vocabulary*

Corso di Laurea Triennale in Informatica, Università di Padova 4/28

Università degli Studi di Padova

Verifica e validazione: analisi statica

Premessa – 4

- **Nessun linguaggio di programmazione d'uso comune garantisce che *ogni* programma scritto in esso sia verificabile per definizione**
 - La progettazione di un linguaggio di programmazione comporta delicate scelte di bilanciamento tra funzionalità e integrità
 - Potenza espressiva contro onere di verifica
- **Scelto un linguaggio occorre fare estrema attenzione a quali costrutti usare in relazione al loro impatto a tempo d'esecuzione**

Corso di Laurea Triennale in Informatica, Università di Padova 5/28

Università degli Studi di Padova

Verifica e validazione: analisi statica

Tecniche di verifica – 1

- **Tracciamento**
 - **Dimostrare completezza ed economicità dell'implementazione**
 - Soddisfacimento di tutti i requisiti
 - Nessuna funzionalità superflua o componente ingiustificato
 - **Ha luogo**
 - Nel trattamento dei requisiti, tra requisiti elaborati (decomposti) ed originali (di livello utente)
 - Tra procedure di verifica e requisiti, disegno, codice
 - Tra codice sorgente e codice oggetto

Corso di Laurea Triennale in Informatica, Università di Padova 6/28

Università degli Studi di Padova

Verifica e validazione: analisi statica

Tecniche di verifica – 2

Tracciamento (segue)

- Certi stili di codifica (ancor più se facilitati dal linguaggio di programmazione) facilitano la verifica mediante tracciamento**
 - Assegnare **singoli** requisiti di basso livello a **singoli** moduli del programma così da richiedere **una sola** procedura di prova ed ottenere una più semplice corrispondenza tra essi
 - Maggiore l'astrazione di un costrutto del linguaggio maggiore la quantità di codice oggetto generato per esso e maggiore l'onere di dimostrazione di corrispondenza

Corso di Laurea Triennale in Informatica, Università di Padova 7/28

Università degli Studi di Padova

Verifica e validazione: analisi statica

Tecniche di verifica – 3

Revisioni

- Strumento essenziale del processo di verifica**
- Possono essere condotte su**
 - Analisi dei requisiti, progettazione, codice, procedure di verifica, risultati di verifica
- Non sono automatizzabili**
 - Richiedono la mediazione e l'interazione di individui
- Possono essere formali o informali**
 - Richiedono comunque indipendenza tra verificato e verificatore
 - La semplicità e la leggibilità di codice e diagrammi sono sempre molto desiderabili

Corso di Laurea Triennale in Informatica, Università di Padova 8/28

Università degli Studi di Padova

Verifica e validazione: analisi statica

Tecniche di verifica – 4

Analisi

- Statica**
 - Applica a requisiti, progetto e codice
- Dinamica (prova = test)**
 - Applica a componenti del sistema o al sistema nella sua interezza
- Gli standard di certificazione richiedono l'uso, in varie combinazioni, di 10 metodi di analisi statica**
 1. Flusso di controllo – 2. Flusso dei dati – 3. Flusso dell'informazione – 4. Esecuzione simbolica – 5. Verifica formale del codice – 6. Verifica di limite – 7. Uso dello *stack* – 8. Comportamento temporale – 9. Interferenza – 10. Codice oggetto

Corso di Laurea Triennale in Informatica, Università di Padova 9/28

Università degli Studi di Padova

Verifica e validazione: analisi statica

1. Analisi di flusso di controllo

Ha l'obiettivo di

- Accertare che il codice esegua nella sequenza attesa**
- Accertare che il codice sia ben strutturato**
- Localizzare codice non raggiungibile sintatticamente**
- Identificare parti del codice che possano porre problemi di terminazione (i.e.: chiamate ricorsive, iterazioni)**
 - **Esempio**
L'analisi dell'albero delle chiamate (*call-tree analysis*) mostra se l'ordine di chiamata specificato a progetto corrisponda a quello effettivo; rivela la presenza di ricorsione diretta o indiretta
 - **Esempio**
Divieto di modifica di variabili di controllo delle iterazioni

Corso di Laurea Triennale in Informatica, Università di Padova 10/28

Università degli Studi di Padova

Verifica e validazione: analisi statica

2. Analisi di flusso dei dati

Accerta che nessun cammino d'esecuzione del programma acceda a variabili prive di valore

Usa i risultati dell'analisi di flusso di controllo insieme alle informazioni sulle modalità di accesso alle variabili (lettura, scrittura)

Rileva possibili anomalie, p.es. più scritture successive senza letture intermedie

È complicata dalla presenza e dall'uso di dati globali raggiungibili da ogni parte del programma

Corso di Laurea Triennale in Informatica, Università di Padova 11/28

Università degli Studi di Padova

Verifica e validazione: analisi statica

3. Analisi di flusso d'informazione

Determina come l'esecuzione di una unità di codice crei dipendenze (e quali) tra i suoi ingressi e le uscite

Le sole dipendenze consentite sono quelle previste dalla specifica

- Consente l'identificazione di effetti laterali inattesi o indesiderati

Può limitarsi a un singolo modulo oppure estendere a più moduli correlati oppure anche all'intero sistema

Corso di Laurea Triennale in Informatica, Università di Padova 12/28

Università degli Studi di Padova Verifica e validazione: analisi statica

4. Esecuzione simbolica – 1

- Verifica proprietà del programma mediante manipolazione algebrica del codice sorgente senza bisogno di specifica formale
 - Combina tecniche di analisi di flusso di controllo, di flusso di dati e di flusso di informazione
- Si esegue effettuando "sostituzioni inverse"
 - A ogni LHS di un assegnamento sostituisce progressivamente il suo RHS
 - LHS = parte sinistra
 - RHS = parte destra
- Trasforma il flusso sequenziale del programma in un insieme di assegnamenti paralleli nei quali i valori di uscita sono espressi come funzione diretta dei valori di ingresso

Corso di Laurea Triennale in Informatica, Università di Padova 13/28

Università degli Studi di Padova Verifica e validazione: analisi statica

4. Esecuzione simbolica – 2

Assumendo assenza di *aliasing* e di effetti laterali di funzioni, da:

```
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
```

si ottiene:

```
A+B ≤ 0 ⇒
  X == A+B
  Y == D-C
  Z conserva il valore precedente
A+B > 0 ⇒
  X == A+B
  Y == D-C
  Z == D-C+1
```

Corso di Laurea Triennale in Informatica, Università di Padova 14/28

Università degli Studi di Padova Verifica e validazione: analisi statica

5. Verifica formale del codice

- Prova la correttezza del codice sorgente rispetto alla specifica formale dei suoi requisiti
 - Esplora *tutte* le esecuzioni possibili
 - Ciò che non è fattibile mediante prove dinamiche
- Correttezza parziale
 - Elaborazione e prova di teoremi (condizioni di verifica) la cui verità implica che il verificarsi di certe pre-condizioni assicura il verificarsi di determinate post-condizioni
 - Sotto l'ipotesi di terminazione del programma
- Correttezza totale
 - Richiede prova di terminazione

Corso di Laurea Triennale in Informatica, Università di Padova 15/28

Università degli Studi di Padova Verifica e validazione: analisi statica

6. Analisi di limite

- Verifica che i dati del programma restino entro i limiti del loro tipo e della precisione desiderata
 - Analisi di *overflow* nella rappresentazione di grandezze
 - Analisi di errori di arrotondamento
 - Verifica di valori di limite (*range checking*)
 - Analisi di limite di strutture
- Linguaggi evoluti assegnano limiti statici a tipi discreti consentendo verifiche automatiche sulle corrispondenti variabili
- Più problematico con tipi enumerati e reali

Corso di Laurea Triennale in Informatica, Università di Padova 16/28

Università degli Studi di Padova Verifica e validazione: analisi statica

7. Analisi d'uso di *stack*

- Lo *stack* è l'area di memoria che i sottoprogrammi usano per immagazzinare dati locali, temporanei e indirizzi di ritorno generati dal compilatore
 - Concettualmente uno *stack* per ogni flusso di controllo (*thread*)
- Determinare la massima domanda di *stack* richiesta da un'esecuzione e la relazione con la dimensione dell'area di memoria effettivamente assegnata
- Verificare che non vi possa essere collisione tra *stack* e *heap* per qualche esecuzione
 - L'ampiezza dello *stack* cresce con l'annidamento ricorsivo di chiamate
 - L'ampiezza dell'*heap* vuoto è fissata a configurazione e poi consumata con la creazione dinamica di oggetti

Corso di Laurea Triennale in Informatica, Università di Padova 17/28

Università degli Studi di Padova Verifica e validazione: analisi statica

8. Analisi temporale

- Concerne le proprietà temporali richieste ed esibite dalle dipendenze delle uscite dagli ingressi del programma
 - Produrre il valore giusto al momento giusto
- Carenze o eccessi del linguaggio di programmazione e delle tecniche di codifica possono complicare questa analisi
 - Esempi: iterazioni prive di limite statico, ricorso sistematico a strutture dati dinamiche

Corso di Laurea Triennale in Informatica, Università di Padova 18/28

Università degli Studi di Padova Verifica e validazione: analisi statica

9. Analisi d'interferenza

- ❑ Mostra l'assenza di effetti di interferenza tra parti separate ("partizioni") del sistema
 - Non necessariamente limitate ai componenti *software*
- ❑ Veicoli tipici di interferenza
 - Memoria dinamica (*heap*) condivisa, dove parti separate di programma lasciano traccia di dati abbandonati ma non distrutti (*memory leak*)
 - Azzeramento preventivo delle pagine di memoria riutilizzate (p.es. NT v5.x)
 - Zone di I/O
 - Dispositivi condivisi programmabili con effetti a livello sistema (p.es. *watchdog*)

Corso di Laurea Triennale in Informatica, Università di Padova 19/28

Università degli Studi di Padova Verifica e validazione: analisi statica

10. Analisi di codice oggetto

- ❑ Assicura che il codice oggetto da eseguire sia una traduzione corretta del codice sorgente corrispondente e che nessun errore od omissione siano stati introdotti dal compilatore
- ❑ Viene ancora effettuata manualmente
- ❑ Viene facilitata dalle informazioni di corrispondenza prodotte dal compilatore

Corso di Laurea Triennale in Informatica, Università di Padova 20/28

Università degli Studi di Padova Verifica e validazione: analisi statica

Programmi verificabili – 1

- ❑ L'adozione di standard di codifica e di sottoinsiemi del linguaggio appropriati discende dalla scelta dei metodi di verifica richiesti
 - L'uso di costrutti del linguaggio inadatti può compromettere la verificabilità del programma
- ❑ La verifica solo retrospettiva (a valle dello sviluppo) è spesso inadeguata
 - Sviluppo condotto senza tenere conto delle esigenze di verifica è rischioso
 - Il costo di rilevazione e correzione di un errore è tanto maggiore quanto più avanzato è lo stadio di sviluppo

Corso di Laurea Triennale in Informatica, Università di Padova 21/28

Università degli Studi di Padova Verifica e validazione: analisi statica

Costo di correzione di errori

Stadio	Costo
Requirements	1
Design	5
Code	10
Unit Test	20
Acceptance Test	50
Maintenance	200

Corso di Laurea Triennale in Informatica, Università di Padova 22/28

Università degli Studi di Padova Verifica e validazione: analisi statica

Programmi verificabili – 2

- ❑ Eseguire iterazioni di revisione – verifica dopo ogni rilevazione di errore è inefficace e troppo oneroso
 - Approccio retrospettivo
- ❑ Convieni piuttosto effettuare analisi statiche durante la fase di codifica
 - Approccio costruttivo: correttezza per costruzione
 - Produce un miglior rapporto costo/qualità a fronte di un maggior costo concettuale e progettuale

Corso di Laurea Triennale in Informatica, Università di Padova 23/28

Università degli Studi di Padova Verifica e validazione: analisi statica

Programmi verificabili – 3

- ❑ 4 ragioni fondamentali per richiedere o proibire l'uso di particolari costrutti del linguaggio di implementazione
 - Regole d'uso per assicurare comportamento predicibile
 - Regole d'uso per consentire analisi del sistema
 - Regole d'uso per facilitare le prove
 - Considerazioni pragmatiche

Corso di Laurea Triennale in Informatica, Università di Padova 24/28

Verifica e validazione: analisi statica

Università degli Studi di Padova

Verifica e validazione: analisi statica

Comportamento predicibile

- ❑ **Il codice sorgente non deve presentare ambiguità**
 - **Effetti laterali (p.es. di funzioni)**
 - Quando diverse invocazioni della stessa funzione producono risultati diversi
 - **Effetti dell'ordine di elaborazione e inizializzazione**
 - Quando l'esito di un programma dipende dall'ordine di elaborazione entro una singola unità oppure tra unità
 - **Effetti dei meccanismi di passaggio dei parametri**
 - Quando la scelta di una modalità di passaggio (per copia, per riferimento), ove permessa, può influenzare l'esito dell'esecuzione

Corso di Laurea Triennale in Informatica, Università di Padova 25/28

Università degli Studi di Padova

Verifica e validazione: analisi statica

Analizzabilità del sistema

- ❑ **La verifica statica richiede la costruzione di modelli del sistema da analizzare a partire dal codice del programma**
- ❑ **Rappresentano il programma come un grafo diretto e ne studiano i cammini possibili**
 - **Storie di esecuzione**
 - **Le transizioni tra stati (archi) hanno etichette che descrivono proprietà sintattiche o semantiche dell'istruzione corrispondente**
 - La presenza di flussi di eccezione e di risoluzione dinamica di chiamata (*dispatching*) complica notevolmente la struttura del grafo
 - **Ciascun flusso di controllo (*thread*) viene rappresentato ed analizzato separatamente**

Corso di Laurea Triennale in Informatica, Università di Padova 26/28

Università degli Studi di Padova

Verifica e validazione: analisi statica

Facilità di prova

- ❑ **Due forme (o atteggiamenti) di prova**
 - Investigativo: informale (*debugging*)
 - Formale: aderente a norme scelte o imposte
- ❑ **Alcuni costrutti di linguaggio sono di ostacolo**
 - La risoluzione dinamica di chiamata (*dispatching*) complica le prove di copertura
 - Vedi esempio (Per approfondire #17)
 - La conversione forzata tra tipi (*casting*) complica l'analisi dell'identità dei dati
 - Le eccezioni predefinite complicano le prove di copertura presentando cammini di esecuzione difficili da raggiungere senza modificare il codice

Corso di Laurea Triennale in Informatica, Università di Padova 27/28

Università degli Studi di Padova

Verifica e validazione: analisi statica

Considerazioni pragmatiche

- ❑ **L'efficacia dei metodi costruttivi di analisi statica e dinamica è funzione della qualità di strutturazione del codice**
 - Per esempio: ogni modulo abbia un solo punto di ingresso e un solo punto di uscita
- ❑ **La verifica di un programma relaziona frammenti di codice con frammenti di specifica**
 - La verificabilità dipende dalla semplicità dell'informazione di contesto
 - Controllare e limitare gli ambiti (*scope*) e le visibilità
 - **Architettura software**
 - Per esempio: incapsulazione dello stato e controllo di accesso a esso

Corso di Laurea Triennale in Informatica, Università di Padova 28/28