Evoluzione e manutenzione

Una definizione

La manutenzione del sw è la modifica di un prodotto sw dopo la consegna per correggere difetti, migliorare le prestazioni o altri attributi, o per adattare il prodotto a un ambiente modificato (IEEE Std. 610.12 1990, IEEE Std. 1219-1998)

Ma già lo standard IEEE 1219-1998 stabilisce criteri di pianificazione della manutenzione sia per prodotti software esistenti sia per software ancora in sviluppo e asserisce che, idealmente, la pianificazione della manutenzione dovrebbe iniziare insieme alla pianificazione dello sviluppo del software

Lo standard IEEE 14764-2006 (Software Maintenance) prescrive di stabilire una strategia di manutenzione

Leggi dell'evoluzione del sw (Lehman 1980, Lehman e Belady 1985)

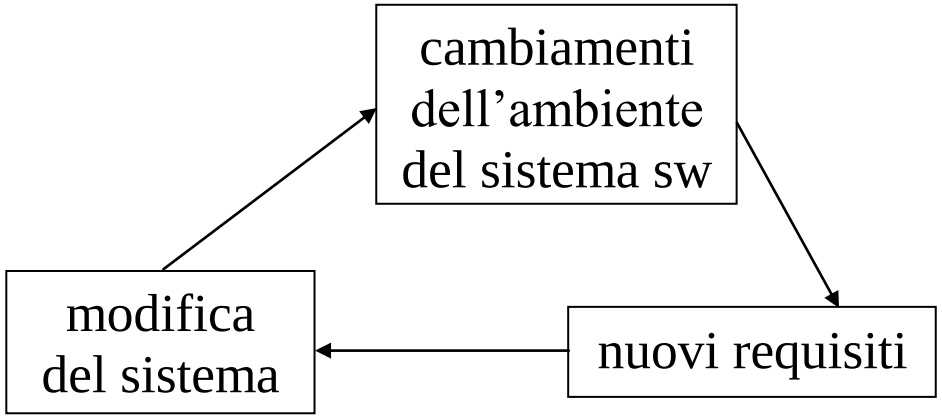
Sono cinque ipotesi derivate dall'analisi (misurata) della crescita di numerosi sistemi sw di grandi dimensioni

- *Cambiamento continuo*: ogni programma in uso o sottostà a cambiamenti continui o diventa progressivamente meno utile; il processo di cambiamento o di decadenza continua finché non diventa più conveniente sostituire il programma

Es. motivi di cambiamento: nuovi vincoli, interazione con altri sistemi, aumento del numero degli utenti, *porting* su altre piattaforme, riscrittura in nuovi linguaggi

Interpretazione:

la manutenzione del sw è inevitabile perché il processo di evoluzione del sw è ciclico

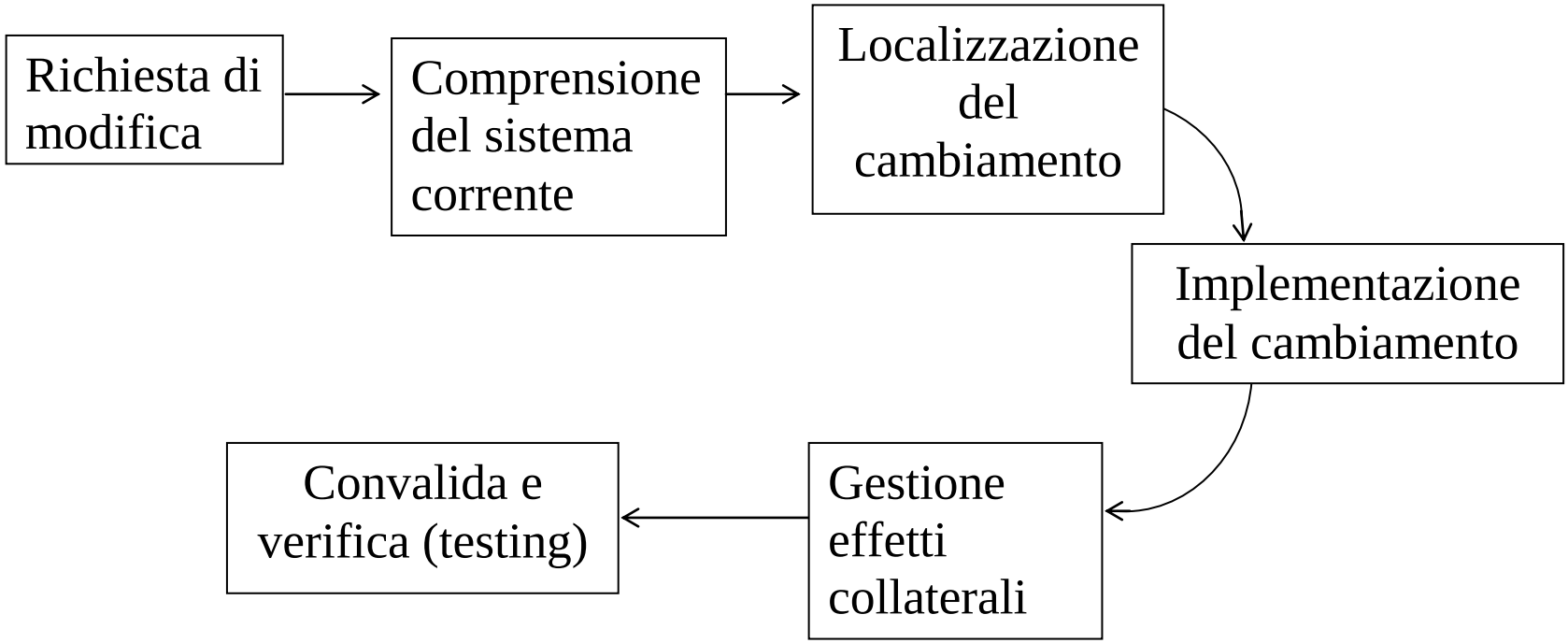


Evoluzione del sw e declino del sistema

Si può decidere di sostituire il sistema anziché manutenerlo se si risponde positivamente ad alcune delle seguenti domande:

- Il costo della manutenzione è troppo elevato?
- L'affidabilità del sistema è inaccettabile?
- È difficile adattare il sistema entro una scadenza ragionevole?
- Le prestazioni del sistema non rispettano i vincoli?
- Le funzionalità del sistema sono di utilità limitata?
- È possibile fare lo stesso lavoro meglio, più rapidamente o spendendo meno usando altri sistemi?
- Il costo di manutenzione dell'hw è tale da giustificare la sostituzione con hw nuovo più economico?

Attività del processo di manutenzione



Attività del processo di manutenzione (cont.)

Comprensione del sistema corrente

- Assorbe dal 50% al 90% del tempo dedicato alla manutenzione
- In termini temporali, è basata 3.5 volte più sul codice che su altri documenti (spesso ciò deriva dalla inattendibilità della documentazione)

Localizzazione del cambiamento

È supportata dall'analisi automatica del codice

Gestione effetti collaterali

- Gli effetti collaterali di una modifica del codice sulle altre parti del programma prendono il nome di *ripple effects*
- La loro gestione è supportata dall'analisi del codice, con generazione automatica delle dipendenze

Manutenzione: una classificazione (ISO/IEC 14764:2006)

- Manutenzione correttiva (21%): correzione dei guasti che quotidianamente vengono scoperti
- Manutenzione adattiva (25%): modifiche secondarie necessarie come conseguenza di modifiche primarie, cambiamenti dell'hw o dell'ambiente
- Manutenzione perfettiva (50%): cambiamenti che migliorano qualche aspetto del sistema (es. chiarificazione della documentazione, aumento della copertura dei test, ecc.)
- Manutenzione preventiva (4%): cambiamento di qualche aspetto del sistema al fine di prevenire i malfunzionamenti (es. miglioramento della gestione degli errori, introduzione di controlli di tipi, ecc.); tipicamente si effettua quando si scopre, e quindi corregge, un guasto effettivo o potenziale che non si è ancora manifestato sotto forma di malfunzionamento

Manutenzione: responsabilità

- La manutenzione assorbe la quantità prevalente delle risorse impiegate nel ciclo di vita del sw
- Essa è condotta da un team, comprendente analisti, progettisti e programmatori (questi ultimi hanno un ruolo più vasto che non nello sviluppo perché è richiesta una conoscenza profonda del codice)
- Il team è composto prevalentemente da persone che non hanno partecipato allo sviluppo ed, eventualmente, da una o più persone che vi hanno partecipato: una squadra nuova è più obiettiva

Sistemi ereditati (legacy systems)

Sistemi per i quali l'attività di manutenzione è diventata prevalente su ogni altra; caratteristiche:

- Sono stati implementati diversi anni fa
- La loro tecnologia (linguaggi di programmazione, stile di codifica, hw) è diventata obsoleta
- Sono stati mantenuti per un lungo periodo
- La loro struttura si è deteriorata e non facilita la comprensione del codice
- La loro documentazione è diventata obsoleta
- Contengono regole di business che non sono documentate altrove
- Non possono essere sostituiti facilmente
- Rappresentano un grosso investimento per l'azienda
- Gli autori originali non sono più disponibili
- La loro evoluzione costituisce una sfida particolare

Manutenzione dei legacy system

Obiettivo: migliorare la qualità del sw contenendo i costi; approcci principali:

- Redocumentation (mediante analisi statica del codice)
- Restructuring / refactoring = trasformazione di codice mal-strutturato in codice ben-strutturato
- Reverse engineering = creazione del design e delle specifiche a partire dal codice
- Re-engineering = reverse engineering + modifica di specifiche e design + forward engineering $\rightarrow$ creazione di un nuovo sistema basato su specifiche e design rivisitati

Strumenti per la ristrutturazione del sw

DMS (Design Maintenance System) Software Reengineering Toolkit
di Semantic Design, Austin, TX, membro OMG
(www.semdesigns.com)
fornisce funzionalità per la ristrutturazione di codice COBOL, C/C++, Java, FORTRAN 90 e VHDL

Function Encapsulation Tool
sviluppato presso la Wayne State University di Detroit, MI
(www.cs.wayne.edu/~vip/RefactoringTools/)
esegue la rifattorizzazione di programmi C in C++

PlusFORT
sviluppato da Polyhedron, Standlake, UK
(www.polyhedron.com)
è un insieme di strumenti FORTRAN che ristrutturano programmi FORTRAN in codice FORTRAN o C

Strumenti per reverse engineering

Imagix 4D
sviluppato da Imagix, San Luis Obispo, CA
(www.imagix.com)
aiuta a comprendere il funzionamento di programmi C e C++, eseguendo sia reverse engineering, sia ridocumentazione del codice sorgente

Understand
sviluppato da Scientific Toolworks, Inc., St. George, UT
(www.scitools.com)
effettua reverse engineering, creazione automatica di documenti, calcolo di metriche; aiuta a comprendere, navigare e manutenere codice sorgente di programmi Ada, FORTRAN, C, C++

Elenco di strumenti di reverse engineering (in fondo alla pagina)
http://scgwiki.iam.unibe.ch:8080/SCG/370

Fattori che influenzano l'entità dello sforzo di manutenzione

Fattori non tecnici

- Disponibilità e avvicendamento del personale
- Durata attesa del sistema (tanto più è lunga, tanta più cura richiede la manutenzione)
- Dipendenza dall'ambiente
- Affidabilità dell'hw
- Novità dell'applicazione

Fattori tecnici

- Qualità del design (modularità ecc.)
- Qualità del codice
- Linguaggio di programmazione
- Qualità della documentazione
- Qualità del testing
- Tipo di applicazione (i sistemi in tempo reale e altamente sincronizzati sono più difficili da modificare)
- Novità dell'implementazione

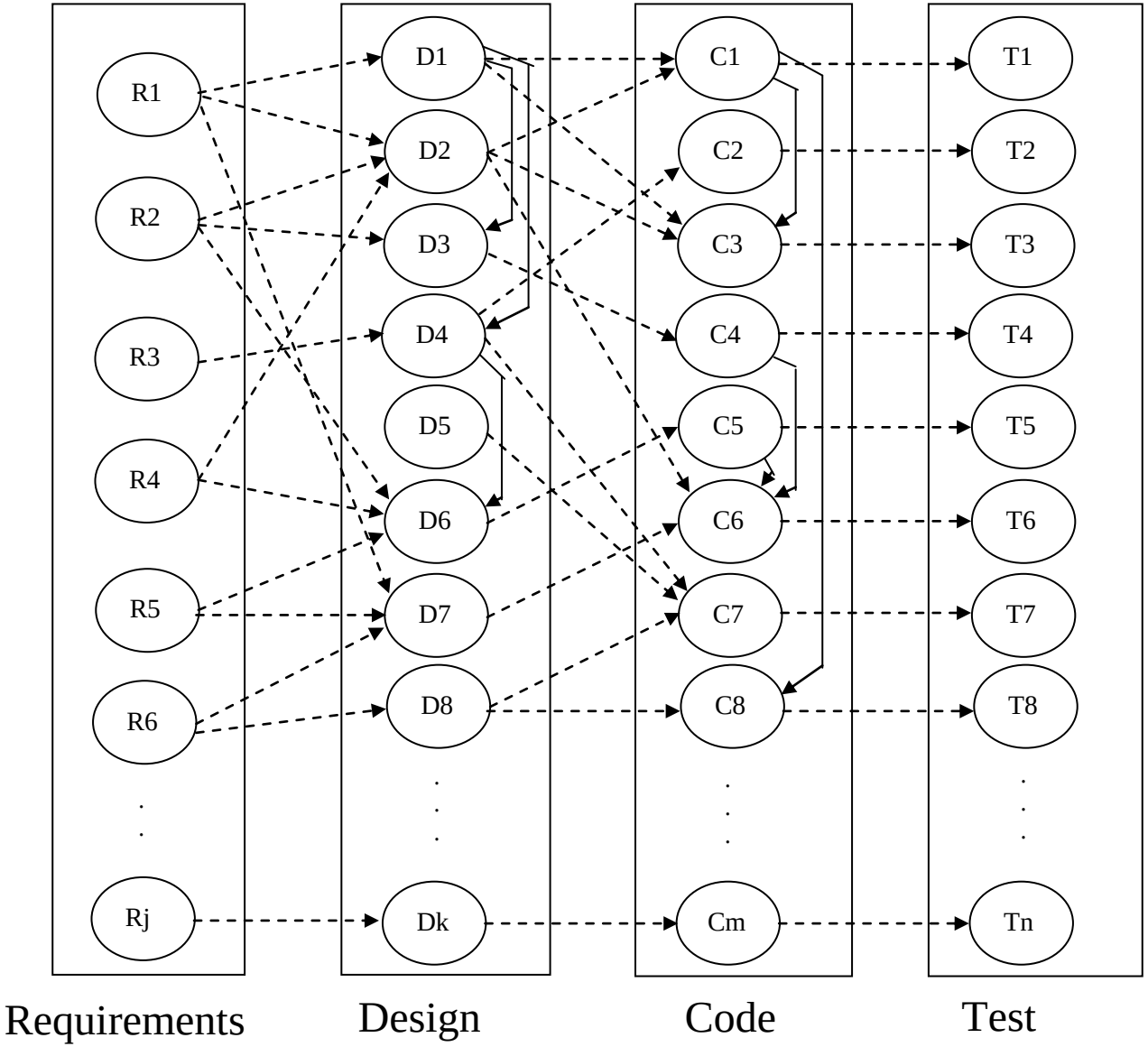
Analisi di impatto

Valutazione dei rischi (effetti su risorse, costi e scheduling) di una modifica proposta per ciascun artefatto coinvolto nella modifica; produce in uscita le seguenti misure:

- *Tracciabilità verticale* = relazioni di dipendenza tra le parti di un artefatto (es. tra i requisiti nel documento di specifica)
- *Tracciabilità orizzontale* = relazioni di dipendenza tra le parti di artefatti prodotti consecutivamente nello sviluppo del sw

Ogni modifica apportata non deve aumentare la complessità del grafo di tracciabilità

Grafo di tracciabilità complessivo



Problemi di sviluppo e manutenzione

- Doppia modifica: l'esistenza di copie/cloni dello stesso frammento di codice comporta che le correzioni vadano replicate
- Modifiche simultanee: programmatori diversi possono modificare concorrentemente librerie di codice condiviso, con possibilità di interferenza tra le modifiche
- Copie locali: programmatori diversi possono lavorare contemporaneamente su copie locali dello stesso modulo
- Tracciabilità: modifiche che non ripercorrono tutte le fasi di sviluppo possono portare ad artefatti inconsistenti tra di loro

↓

necessità di coordinare (identificare, organizzare e controllare) le operazioni di modifica degli artefatti prodotti nelle varie fasi e iterazioni di sviluppo e manutenzione del sw

Famiglie di prodotti

Spesso un'applicazione è in realtà costituita da una famiglia di prodotti, che si differenziano, ad es., per:

- ambiente operativo (hw: processore, sw: sistema operativo)
- ambito legislativo per cui sono concepiti
- funzionalità offerte

Repository condiviso = totalità dei moduli che servono per costruire i prodotti della famiglia + info circa la componibilità degli stessi (sfruttate da strumenti di costruzione)

Configurazione = collezione dei componenti di un sistema che rappresenta un prodotto (cioè una delle varianti) della famiglia

↓

necessità di controllare le differenze fra configurazioni
al fine di minimizzare rischi ed errori di sviluppo e testing

Gestione delle configurazioni

È la risposta ai due tipi di necessità evidenziati sopra; supporta:

- identificazione del sw: ogni modulo è identificato univocamente, assieme alla sua versione
- controllo delle configurazioni: le operazioni di modifica dei moduli di una certa configurazione sono disciplinate
- convalida e verifica delle configurazioni: ogni configurazione corrisponde ai requisiti espressi dall'utente per quella configurazione
- evoluzione delle configurazioni: è possibile avere informazioni sulla storia dei moduli

Repository di gestione delle configurazioni

Contiene info per il controllo dei cambiamenti (problemi riscontrati, organizzazione che li ha scoperti, chi e quando li ha risolti, chi ha autorizzato il cambiamento, priorità del cambiamento, ecc.)

A fronte di ciascun malfunzionamento segnalato dagli utenti, è necessario poter costruire esattamente la stessa versione dell'applicazione usata dal cliente per poter riprodurre la condizione di errore rilevata

Repository di gestione delle configurazioni: accesso esclusivo a un modulo

Operazioni fondamentali

- `checkout [version] [lock] ⟨module⟩` : richiede una copia locale del modulo. Si può specificare quale versione si desidera. Se si intende modificare tale modulo, si dovrà chiedere l'accesso esclusivo (lock), che verrà concesso solo se nessun altro utente ne è già in possesso. Se non è richiesto il lock, il modulo è disponibile in sola lettura
- `checkin ⟨module⟩` : viene inserita nel deposito dei moduli una nuova versione del modulo e viene rilasciato il lock sullo stesso

Strumenti SCM (Software Configuration Management)

- CA Harvest Change Manager
  sistema multipiattaforma sviluppato da Computer Associates di Islandia (NY)
  (www.ca.com/it)

- ClearCase
  sviluppato da Rational (assorbita da IBM)
  (http://www-306.ibm.com/software/awdtools/clearcase)

- PVCS
  distribuito da Serena, San Mateo, CA
  (www.serena.com)
  esiste come Version Manager e come Professional Suite ed è utilizzabile anche per applicazioni web

Strumenti SCM (cont.)

- SourceForge
  distribuito da VA Software Corporation
  (sourceforge.net, sito di software open source)
  fornisce gestione delle versioni, funzionalità di compilazione, monitoraggio dei bug e altro

- SurroundSCM
  sviluppato da Seapine Software, Mason, OH
  (www.seapine.com)

- Vesta
  rilasciato da Compac con licenza GNU
  (www.vestasys.com)

Elenco di strumenti e ambienti SCM commerciali
(www.cmcrossroads.com/component/options,com_directory/)

Sistema CVS (Concurrent Versions System) (http://ximbiot.com/cvs)

- È il sistema open source dominante nel controllo delle versioni trasparente alla rete
- Adotta una architettura client server che consente agli utenti di accedere ai file tramite connessioni Internet
- È disponibile per Windows, Macintosh e Unix
- Tratta solo codice sorgente (e file di testo in generale)
- Crea un semplice archivio
- Gestisce tutte le versioni di un file in un unico file memorizzando solo le differenze fra le versioni progressive
- Protegge il sistema contro modifiche simultanee impiegando direttori diversi per ciascun sviluppatore e unendo le modifiche quando tutti gli sviluppatori hanno completato il proprio lavoro
- Non implementa alcun processo di controllo delle modifiche (ad es. richieste di modifica, rapportistica circa le modifiche, monitoraggio dei bug)
- Non è un sistema di compilazione

Strumenti di costruzione dell'eseguibile

Uniti agli strumenti di configuration management, permettono di controllare le varianti del sistema e l'evoluzione temporale dei moduli componenti.
Consentono di:

- ricompilare solo i moduli modificati o dipendenti da moduli modificati
- specificare diverse varianti dell'applicazione, che vengono costruite usando i moduli necessari a ciascuna variante; a tal fine alcuni sono dotati di Module Interconnection Language (MIL) con cui descrivere la scomposizione logica del sistema, le relazioni tra componenti logiche e componenti fisiche e le dipendenze che vincolano la costruzione dell'eseguibile associato a ciascuna variante
- specificare direttive di compilazione diverse per le diverse varianti del sistema

Lo strumento più semplice di costruzione dell'eseguibile è *make*