Processo – parte VI

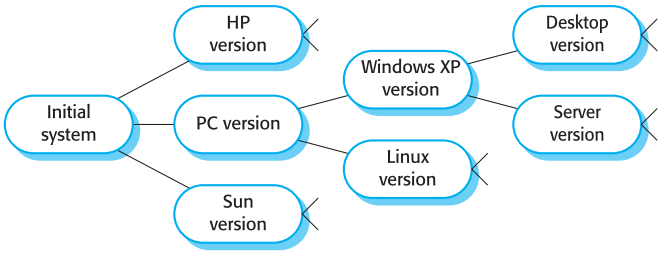
Leggere Cap 29 Sommerville, Sez 7.8 Ghezzi et al.

università di ferrara
DA SEICENTO ANNI GUARDIAMO AVANTI.

Gestione della configurazione (Configuration Management, CM)

- Famiglie di prodotti software:
  - Per diverse piattaforme (hardware, OS)
  - Con funzionalità diverse
  - Versioni personalizzate
- Ogni membro è soggetto a modifiche nel tempo
- La gestione di tutte le versioni è un'attività che richiede sforzo e investimenti
- Sviluppo e applicazione di procedure e standard per gestire una famiglia di prodotti in evoluzione

Famiglia di prodotti



Standard CM

- L'attività di CM dovrebbe essere regolata da standard interni a un'organizzazione, che definiscano:
  - l'identificazione degli elementi
  - il controllo delle modifiche
  - la gestione delle versioni
- Gli standard possono seguire standard esterni
- Standard attuali solitamente adatti al modello a cascata

Sforzo necessario per CM

- Per grandi progetti, è conveniente investire in un piano di CM rigido, che consenta una tracciabilità completa di
  - codice sorgente e dati di configurazione
  - documentazione
  - artefatti
  - test
- Per piccoli progetti (metodi agili) il costo può non essere giustificato; occorre decidere criticamente quali standard e strumenti usare

Pianificazione CM

- Può riguardare tutti gli artefatti del processo:
  - specifiche dei requisiti
  - specifiche di progetto
  - codice sorgente
  - dati di test
  - documentazione
- Un progetto di grandi dimensioni può richiedere la produzione di migliaia di documenti.

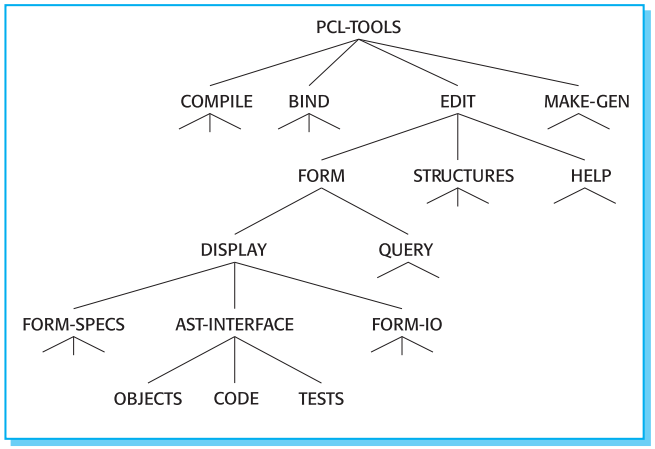
Il piano CM

- Definisce quali tipi di documento gestire e uno schema di identificazione
- Individua i responsabili delle procedure
- Definisce le politiche di CM
- Specifica gli strumenti da usare e le modalità del loro uso
- Descrive quali informazioni devono essere registrate e la struttura del database utilizzato

Identificare gli oggetti

- I documenti (anche migliaia) devono essere identificati univocamente.
- Alcuni dei documenti si evolvono durante l'intero ciclo di vita del software.
- Lo schema di identificazione dovrebbe far sì che documenti correlati abbiano nomi simili.
- Schema gerarchico con nomi multi-livello:
  - PCL-TOOLS/EDIT/FORMS/DISPLAY/AST-INTERFACE/CODE

Esempio: schema gerarchico



Database della configurazione

- Tutte le informazioni di CM dovrebbero essere memorizzate in un DB
- Il DB dovrebbe permettere query come:
  - a quali clienti è stata consegnata una particolare versione?
  - quale piattaforma hw/sw è necessaria per eseguire una determinata versione?
  - quante versioni sono state create e in quali date?
  - su quali versioni ha effetto una modifica di un particolare componente?
  - quante richieste di modifica sono in attesa per una versione?
  - quanti errori sono stati segnalati per una versione?
- Il DB dovrebbe essere collegato ai documenti oggetto di CM, ma spesso si tiene separato per ridurre i costi

Gestione delle modifiche

- I sistemi software sono soggetti a continue richieste di modifiche:
  - Dagli utenti
  - Dagli sviluppatori
  - Dal mercato
- La gestione delle modifiche
  - Tiene traccia delle modifiche
  - Fa sì che siano implementate nel modo più economico

Processo di gestione delle modifiche

```
Request change by completing a change request form
Analyze change request
if change is valid then
    Assess how change might be implemented
    Assess change cost
    Submit request to change control board
    if change is accepted then
        repeat
            make changes to software
            submit changed software for quality approval
        until software quality is adequate
        create new system version
    else
        reject change request
else
    reject change request
```

Modulo di richiesta di una modifica

- La definizione del modulo è parte della pianificazione CM
- Il modulo indica
  - da parte del richiedente:
    - la modifica proposta
    - chi la richiede
    - il motivo della richiesta
    - la sua urgenza
  - da parte dello staff di manutenzione:
    - valutazione
    - analisi di impatto
    - costo
    - raccomandazioni

Commissione di controllo delle modifiche

- Le modifiche valide devono essere valutate da un punto di vista organizzativo e strategico, oltre che tecnico.
- La valutazione dovrebbe essere eseguita da un gruppo esterno rispetto a quello di sviluppo e manutenzione:
  - Per progetti di grandi dimensioni e con funzionalità critiche (es. militari) si ha una commission control board (CCB) strutturata in modo formale
  - Per progetti più piccoli può consistere in un manager di progetto
  - Nei metodi agili il cliente è coinvolto direttamente nelle decisioni sulle modifiche

Esempio

Change Request Form

Project: Proteus/PCL-Tools **Number:** 23/02
Change requester: I. Sommerville **Date:** 1/12/02
Requested change: When a component is selected from the structure, display the name of the file where it is stored.

Change analyser: G. Dean **Analysis date:** 10/12/02
Components affected: Display-Icon.Select, Display-Icon.Display

Associated components: FileTable

Change assessment: Relatively simple to implement as a file name table is available. Requires the design and implementation of a display field. No changes to associated components are required.

Change priority: Low

Change implementation:
Estimated effort: 0.5 days

Date to CCB: 15/12/02 **CCB decision date:** 1/2/03
CCB decision: Accept change. Change to be implemented in Release 2.1.
Change implementor: **Date of change:**
Date submitted to QA: **QA decision:**
Date submitted to CM:
Comments

Storia di derivazione

- La sequenza delle modifiche a un componente software costituisce la sua storia di derivazione
- Si può registrare come commento in formato standard all'inizio del codice sorgente
- In questo modo è analizzabile automaticamente da strumenti.

Esempio

```
// BANKSEC project (IST 6087)
//
// BANKSEC-TOOLS/AUTH/RBAC/USER_ROLE
//
// Object: currentRole
// Author: N. Perwaiz
// Creation date: 10th November 2002
//
// © Lancaster University 2002
//
// Modification history
// Version      ModifierDate    Change        Reason
// 1.0  J. Jones    1/12/2002   Add header    Submitted to CM
// 1.1  N. Perwaiz  9/4/2003    New field     Change req. R07/02
```


Versioni / varianti / rilasci

- Versione
  - Istanza di un sistema funzionalmente distinta da tutte le altre
- Variante
  - Istanza di un sistema funzionalmente identica ad altre, ma non-funzionalmente distinta da esse.
- Rilascio
  - Istanza di un sistema distribuita agli utenti

Gestione delle versioni e dei rilasci

- Identifica e registra le versioni di un sistema, in modo che
  - Possano essere recuperate quando necessario
  - Non vengano modificate accidentalmente
- Pianifica la produzione di nuove versioni e l'eventuale rilascio
- Garantisce l'uso corretto di strumenti e procedure.

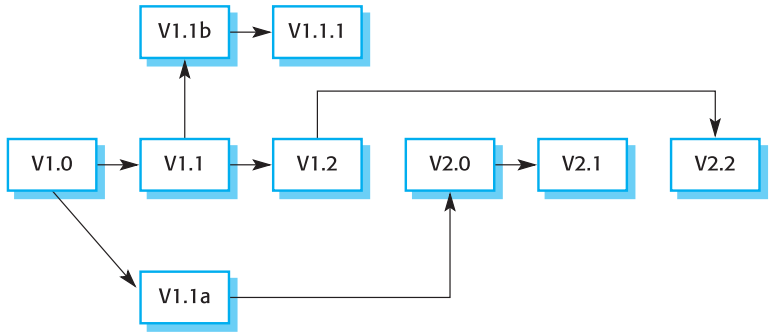
Identificazione delle versioni

- Bisogna definire un modo non ambiguo per identificare la versione di un componente.
- Tecniche:
  - Numerazione
  - Identificazione basata su attributi
  - Identificazione orientata alle modifiche

Numerazione

- Derivazione lineare:
  - V1, V1.1, V1.2, V2.1, V2.2, etc.
- La struttura di derivazione effettiva è un albero o una rete, piuttosto che una sequenza
- I nomi non sono significativi
- Uno schema gerarchico causa meno errori nell'identificazione delle versioni

Struttura di derivazione



Identificazione basata su attributi

- A una versione possono essere associati attributi che la identificano:
  - Data, Creatore, Linguaggio di programmazione, Cliente, Stato, …
- Più flessibile della numerazione
- Unicità non garantita: numerazione comunque necessaria
- Possibilità di eseguire query (es.: la penultima versione personalizzata per un determinato cliente)

Identificazione orientata alle modifiche

- Usata per versioni di un sistema, anziché di un componente
- Versione identificata come insieme di modifiche rispetto a una versione base. Esempio:
  - Sistema per Linux e MySQL
  - Adattamento da Linux a Windows XP
  - Adattamento de MySQL a MS SQL Server
- Poiché alcune modifiche possono essere incompatibili, si possono imporre vincoli sulle combinazioni di modifiche applicabili.

Rilasci

- Non solo programmi eseguibili. Possono comprendere:
  - File di configurazione
  - File di dati
  - Programma di installazione
  - Documentazione
  - Package
- Distribuzione: dischi ottici o Internet

Rilasci autocontenuti

- Un cliente può rifiutare un rilascio:
  - Soddisfatto della versione attuale
  - Nuove funzionalità non necessarie o sgradite
- La gestione dei rilasci non deve supporre che il penultimo rilascio sia stato installato
- I file necessari dovrebbero essere ricreati a ogni aggiornamento

Pianificazione dei rilasci

- Decide quando distribuire una nuova versione
- La preparazione e la distribuzione di un rilascio è un processo costoso.
- Decisione influenzata da vari fattori:
  - Qualità del sistema (errori)
  - Modifiche alla piattaforma
  - Concorrenza
  - Esigenze di marketing
  - Richieste di modifica dagli utenti

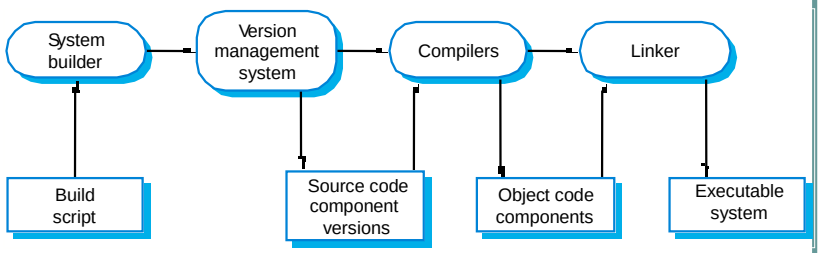
Creazione di rilasci

- Assemblaggio di tutti i componenti necessari (moduli, dati, programmi di installazione, documentazione)
- Versioni diverse (funzionalità, piattaforme)
- Per ogni rilascio è necessario registrare le versioni di tutti i componenti utilizzati, per poterlo ricreare se necessario

Build di sistema

- Compilazione e linking dei moduli che compongono il sistema eseguibile
- Automatizzata tramite “build scripts”
- Fonti di problemi:
  - Mancanza di moduli
  - Versione sbagliata di moduli
  - Mancanza di file di dati
  - Riferimenti a file (es. nomi assoluti)
  - Versione non corretta per una piattaforma
  - Versione non corretta di compilatore, librerie statiche, etc.

Build di sistema



Strumenti per CM

- Necessari per supportare
  - Standard
  - Procedure complesse su grandi quantità di dati
- Workbench
  - Aperti: tool separati per le varie fasi, integrati tramite procedure aziendali. Più flessibili.
  - Integrati: ambienti con tool per tutte le fasi. Più semplici da usare.

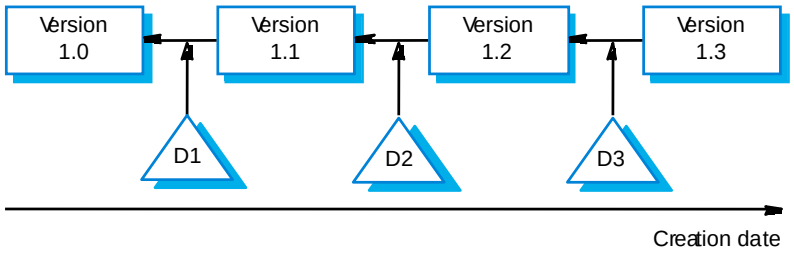
Strumenti per la gestione delle modifiche

- Editor di modulo richiesta modifiche
- Sistema di workflow per definire i compiti di ognuno e per automatizzare i passaggi di informazioni
- Database di modifiche (spesso integrato con sistema di gestione delle versioni): tramite query consente
  - recupero di modifiche
  - report sullo stato attuale o passato di una modifica

Strumenti di gestione delle versioni

- Identificazione delle versioni
- Gestione della memorizzazione (differenziale o delta-based)
- Registrazione dello storico delle modifiche
- Sviluppo indipendente di versioni diverse
- Supporto di progetti

Memorizzazione differenziale



Strumenti per build

- Il build di un sistema di grandi dimensioni è computazionalmente pesante e può richiedere la compilazione di centinaia o migliaia di file
- Funzionalità degli strumenti per build:
  - Linguaggio per dipendenze e interprete
  - Supporto alla selezione degli strumenti (es. determinata versione di un compilatore)
  - Compilazione distribuita
  - Gestione dei file oggetto (ricompilati solo quando cambia il sorgente)

Esempio: make

- Utility nata in ambiente UNIX
- Le dipendenze e le modalità di build per ogni componente sono specificate in *makefile*.
- I makefile possono essere generati automaticamente e adattati alla piattaforma (distribuzione di codice sorgente)