Università degli Studi di Udine

Configuration Management

prof. Maurizio Pighin

Dipartimento di Matematica e Informatica

Configuration management

- Managing the products of a software system
- Involves the development and application of procedures and standards to manage an evolving software product
- May be seen as part of a more general quality management process

CM standards

- CM should always be based on a set of standards which are applied within an organisation
- Standards should define how items are identified, how changes are controlled and how new versions are managed
- Standards may be based on external CM standards (e.g. IEEE 1042 - standard for CM)
- Existing standards are based on a waterfall process model - new standards are needed for evolutionary development

Motivazioni

- Caratteristiche della produzione di software:
  - *Processo svolto da più persone che collaborano per il raggiungimento di un obiettivo comune (importanza della comunicazione e coordinamento)*
  - *Produzione di semilavorati (documenti di specifica e design, casi di test, documentazione)*
  - *Operazioni di manutenzione dopo il rilascio*
  - *"Famiglie" di prodotti:*
    - versioni successive dello stesso prodotto
    - prodotti con caratteristiche simili (customizing e personalizzazioni)
- Come gestire i semilavorati del processo, assicurarne consistenza e integrità, durante tutta la vita del prodotto?

Alcune Situazioni Tipiche

- Accesso coordinato ad un repository/libreria comune:
  - *tutti i moduli sviluppati nell'ambito di un progetto vengono mantenuti all'interno di un'area comune, a cui tutti gli sviluppatori hanno accesso*
  - *uno dei moduli all'interno del repository viene sviluppato da un programmatore ed utilizzato da altri, ad esempio per testare altri moduli*
  - *il programmatore modifica il modulo di cui è responsabile ed introduce un errore*
  - *il team che utilizza il modulo non riceve la notifica delle modifiche (e tanto meno dell'errore introdotto): l'esito di alcuni test può cambiare senza che siano state effettuate modifiche ai moduli sotto esame*

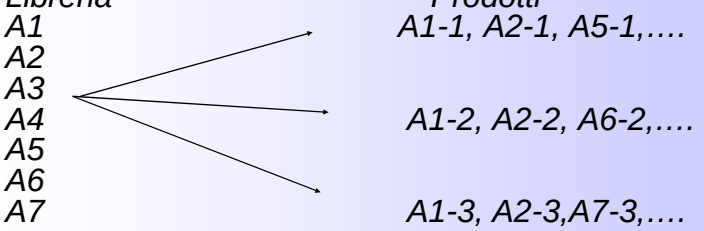
Ingegneria del Software
Progettazione e Laboratorio
Configuration Management
Maurizio Pighin,

Alcune Situazioni Tipiche

- Modifiche simultanee ad un modulo:
  - *due programmatori prelevano una copia lo stesso modulo per apportarvi delle modifiche*
  - *entrambi partono dal medesimo codice e producono due versioni diverse del modulo originario*
  - *uno dei due deposita il modulo modificato nel repository comune*
  - *il secondo deposita il proprio modulo, introducendo le proprie modifiche ed annullando quelle apportate dall'altro programmatore*

Alcune Situazioni Tipiche

- Gestione di famiglie di prodotti:
  - *da un insieme comune di moduli vengono prelevati di volta in volta i moduli necessari a costruire un membro di una famiglia di prodotti*



  - *Problema della doppia manutenzione*
    - A1-1 e A1-2 derivati (copia) dallo stesso modulo A1
    - Modifica alle parti comuni

Problems

- Identification: You should be able to identify the single components and configurations.
  - *This worked yesterday, what has happened?*
  - *Do we have the latest version?*
  - *I have already fixed this problem. Why is it still there?*
- Change tracking:Helps in tracking which changes have been made to which modules and by whom, when and why.
  - *Has this problem been fixed?*
  - *Who is responsible for this change?*
  - *This change looks obvious - has it been tried before?*

Problems

- Software production: Construction of a program involves pre-processing, compilation, linking, etc.
  - *I just corrected this, has something not been compiled?*
  - *How was this binary produced?*
  - *Did we do all the necessary steps in the right order?*
- Concurrent updating: The system should offer possibilities for concurrent changes to components.
  - *Why did my changes disappear?*
  - *How do I get these changes into my version?*
  - *Are our changes in conflict with each other?*

Excuses for not using CM?

- CM only applies to source code
  - *CM is not appropriate during development because we use rapid prototyping*
- It's not that big a project
- You can't stop people from making a quick patch
- We lower our cost by using only minimum-wage persons on our CM staff because CM does not require much skill

Configuration Management

- Obiettivi
  - *massimizzare la produttività riducendo gli errori nel trattamento dei semilavorati*
  - *mantenere l'integrità del prodotto durante tutta la sua vita, dalle specifiche al design, allo sviluppo, all'utilizzo*
  - *supportare le attività di sviluppo e manutenzione*
- The goal is to maximize [programmer] productivity by minimizing [co-ordination] mistakes. (Babich)

Definizione

- CM = insieme di tecniche e metodologie
  - *applicate allo sviluppo di software da parte di team di analisti/progettisti/sviluppatori*
  - *che consentono di:*
    - identificare gli elementi (identificare per poter controllare)
    - controllare le modifiche (controllare l'evoluzione)
    - conoscere lo stato di un sistema (registrare e diffondere informazioni)
- Software Configuration Management is the discipline of organising, controlling and managing the development and evolution of software systems.

Soluzioni

- Shared Data : Problemi connessi ad una semplice condivisione tramite un file-server fra utenti che interagiscono sugli stessi oggetti
- Soluzione 1 - Work-space
  - *Working in isolation:*
    - local dynamicity
    - global stability
    - problems:
      - *double maintenance*

Soluzioni

- Soluzione 2 – Repository comune
  - *Working in group:*
    - global dynamicity
    - problems:
      - *shared data*
      - *simultaneous update*
    - Concurrency strategies:
      - *pessimistic (locking)*
      - *optimistic (copy-merge)*
- Come si pianifica, progetta e gestisce questo/questi repository comune ?

CM planning

- All products of the software process may have to be managed
  - *Specifications*
  - *Designs*
  - *Programs*
  - *Test data*
  - *User manuals*
  - *….*
- Thousands of separate documents are generated for a large software system

The CM plan

- Starts during the early phases of the project
- Defines the types of documents to be managed and their structure
- Defines for each type of document the naming scheme and the positioning scheme
- Defines who takes responsibility for the CM procedures and creation of baselines
- Defines policies for change control and version management
- Defines the CM records which must be maintained

The CM plan

- Describes the tools which should be used to assist the CM process and any limitations on their use
- Defines the process of tool use
- Defines the CM database used to record configuration information

Progettare un sistema di CM

- Analisi del processo di sviluppo
  - *Descrivere fasi, attività, ruoli, responsabilità*
  - *Descrivere le caratteristiche dei prodotti*
- Definizione del processo di configuration management
  - *Definire fasi, attività, ruoli, responsabilità*
  - *Definire tipologie di artefatti trattati*
  - *Definire stati ed evoluzione degli artefatti*
  - *Definire le politiche di configuration management, tenendo conto dell'ambiente, modalità di sviluppo, standard, ...*
- Selezione e customizing degli strumenti utilizzati
- Progetto pilota, training, deployment

Definizione del processo di CM

- Configuration identification
  - *Selezione di ciascun elemento da porre sotto CM*
  - *Identificazione (attribuzione di un identificatore univoco)*
- Change control
  - *Definizione delle politiche secondo cui attuare le modifiche*
  - *Controllo e verifica dell'attuazione delle politiche*
- Configuration status accounting, audit
  - *Raccolta e distribuzione di informazioni sullo stato del CM*
  - *Verifica di conformità (interna ed esterna)*
- Build and release
  - *Costruzione di un semilavorato derivato o del prodotto finale*
  - *Preparazione della distribuzione*

Configuration item identification

- Large projects typically produce thousands of documents which must be uniquely identified
- Some of these documents must be maintained for the lifetime of the software
- Document naming scheme could be defined so that related documents have related names

Configuration item Identification

- Primo passo per la gestione di un insieme di elementi
  - *identificare univocamente ognuno di essi*
    - Selezione dei documenti, moduli, ecc., da sottoporre a CM
    - Identificazione operativa (assegnazione di un identificativo univoco)
- Si possono avere due livelli di identificazione
  - *Logica: Codice del documento, modulo, …*
  - *Fisica: Nome del file contenente il testo del documento, modulo, …*
- Lo schema di identificazione dei documenti, moduli, …, riflette solitamente la struttura gerarchica del prodotto e la gerarchia di responsabilità relative alle diverse parti
- Criticità: la struttura del prodotto potrebbe non essere ancora stata stabilita o fissata

Change management

- Software systems are subject to continual change requests
  - *From users*
  - *From developers*
  - *From market forces*
- Change management is concerned with keeping managing of these changes and ensuring that they are implemented in the most cost-effective way

Change management

- Elemento + modifica = elemento in una nuova versione
- In termini informali, poi definiremo meglio
  - *Versione: stato di un elemento in un istante di tempo*
    - Due configurazioni differiscono se contengono due versioni più o meno diverse dello stesso modulo (contenuto diverso)
  - *Configurazione*
    - Insieme di moduli software, documentazione, ecc., utilizzati per costruire un prodotto (famiglia di prodotti = diverse configurazioni)
    - Lo stesso modulo può far parte di più configurazioni
    - Versioni delle configurazioni
- Tre aspetti da considerare
  - *che cosa si controlla*
  - *chi ha la responsabilità di approvare i cambiamenti*
  - *come operativamente si effettua il controllo*

The change management process


```
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
```


Change request form

- Definition of change request form is part of the CM planning process
- Records change required, suggestor of change, reason why change was suggested and urgency of change(from requestor of the change)
- Records change evaluation, impact analysis, change cost and recommendations (System maintenance staff)

Change request form

Change Request Form

Project: Proteus/PCL-Tools **Number:** 23/94
Change requester: I. Sommerville **Date:** 1/12/98
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:** 10/12/98
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: 15/12/98 **CCB decision date:** 1/2/99
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

Change tracking tools

- A major problem in change management is tracking change status
- Change tracking tools keep track the status of each change request and automatically ensure that change requests are sent to the right people at the right time.
- Often integrated with E-mail systems allowing electronic change request distribution

Change control board

- Changes should be reviewed by an external group who decide whether or not they are cost-effective from a strategic and organizational viewpoint rather than a technical viewpoint
- Should be independent of project responsible for system. The group is sometimes called a change control board
- May include representatives from client and contractor staff

Derivation history

- Record of changes applied to a document or code component
- Should record, in outline, the change made, the rationale for the change, who made the change and when it was implemented
- May be included as a comment in code. If a standard prologue style is used for the derivation history, tools can process this automatically

Component header information


```
// PROTEUS project (ESPRIT 6087)
//
// PCL-TOOLS/EDIT/FORMS/DISPLAY/AST-INTERFACE
//
// Object: PCL-Tool-Desc
// Author: G. Dean
// Creation date: 10th November 1998
//
// © Lancaster University 1998
//
// Modification history
// Version    Modifier  Date        Change        Reason
// 1.0        J. Jones  1/12/1998   Add header    Submitted to CM
// 1.1        G. Dean   9/4/1999    New field     Change req. R07/99
```


Version and release management

- Invent identification scheme for system versions
- Plan when new system version is to be produced
- Ensure that version management procedures and tools are properly applied
- Plan and distribute new system releases

Versions/variants/releases

- *Definition*
  - Version*: an instance of a system which is functionally distinct in some way from other system instances*
  - Variant: *an instance of a system which is functionally identical but non-functionally distinct from other instances of a system*
  - Release*: an instance of a system which is distributed to users outside of the development team*

Concetto di versione e configurazione

- Altre definizioni
- Versione: Stato di un elemento in un istante di tempo
- Configurazione: Insieme di moduli software, documentazione, ecc., utilizzati per costruire un prodotto (famiglia di prodotti = diverse configurazioni)
- Analogie:
  - *Edizioni di un libro (legato al concetto di versione)*
  - *Versione di un'auto*
  - *Configurazioni di un'auto*
  - *Configurazione di un PC: tipo di processore, quantità di memoria, tipo di hard disk, ...*

Concetto di versione e configurazione

- Diverse accezioni del termine "versione":
  - *Revisione: la versione M' di un modulo è una revisione di M se M' sostituisce M in tutte le configurazioni cui M appartiene a partire dal momento in cui M' risulta disponibile (es. correzione di errori)*
  - *Variazione: la versione M' è una variazione di M se M' è un'alternativa ad M in situazioni particolari (es. supporto periferiche diverse)*
- Considerazioni:
  - *Due configurazioni differiscono se contengono elementi diversi*
  - *Due configurazioni differiscono anche se contengono due versioni più o meno diverse dello stesso modulo (contenuto diverso)*
  - *Lo stesso modulo può far parte di più configurazioni*
  - *Si hanno versioni delle configurazioni*

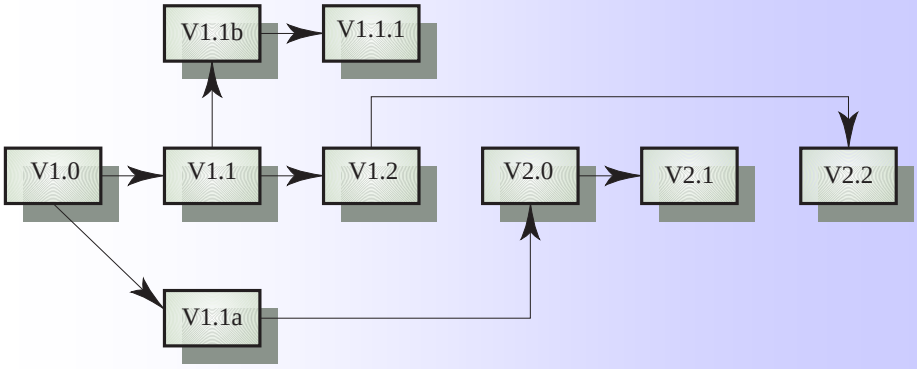
Version identification

- Procedures for version identification should define an unambiguous way of identifying component versions
- Three basic techniques for component identification
  - *Version numbering*
  - *Attribute-based identification*
  - *Change-oriented identification*

Version numbering

- Simple naming scheme uses a linear derivation e.g. V1, V1.1, V1.2, V2.1, V2.2 etc.
- Actual derivation structure is a tree or a network rather than a sequence
- Names are not meaningful.
- Hierarchical naming scheme may be better

Version derivation structure



Attribute-based identification

- Attributes can be associated with a version with the combination of attributes identifying that version
- Examples of attributes are Date, Creator, Programming Language, Customer, Status etc.
- More flexible than an explicit naming scheme for version retrieval; can cause problems with uniqueness
- Needs an associated name for easy reference

Attribute-based queries

- An important advantage of attribute-based identification is that it can support queries so that you can find ‘the most recent version in Java’ etc.
- Example
  - *AC3D (language =Java, platform = NT4, date = Jan 1999)*

Change-oriented identification

- Integrates versions and the changes made to create these versions
- Used for systems rather than components
- Each proposed change has a change set that describes changes made to implement that change
- Change sets are applied in sequence so that, in principle, a version of the system that incorporates an arbitrary set of changes may be created
- Esempio “Windows update”

Memorizzazione delle versioni

- Diversi approcci per la memorizzazione delle versioni:
  - *completa: ogni versione di un modulo in un file separato*
  - *delta: memorizzare in forma completa solo una versione; di tutte le altre versioni si conservano solo le differenze*
    - forward delta: prima versione e differenze rispetto alle successive
    - backward delta: versione più recente e differenze rispetto alle precedenti
  - *si ha compilazione condizionale (diverse versioni nello stesso file)*
- Vantaggi e svantaggi:
  - *spazio di memorizzazione richiesto*
  - *facilità ed efficienza nel reperimento di una specifica versione*
  - *sicurezza (possibilità di perdita dei file memorizzati)*
  - *semplicità nell'apportare modifiche alle parti comuni a più versioni*

Paradigma check-in/check-out

- Check-out
  - *preleva una copia di una versione di un modulo da un repository e la deposita nello spazio di lavoro*
  - *criteri per la selezione della versione (ultima, x.y, data, …)*
  - *possibilità di porre o meno un lock sulla specifica versione (lock = modifica possibile)*
  - *Costruzione del WorkSpace*
    - *How do we "populate" our workspace?*
      - hand picking
      - composition model
      - change set model

Paradigma check-in/check-out

- *Hand picking*
  - Si parte sostanzialmente da una “Lista dei file” (bill of material) da cui si ricavano i file desiderati
- *Composition model*
  - Si parte da un “Modello del sistema”
  - Si definisce un “Criterio di estrazione”
    - *L’ultimo*
    - *L’ultima configurazione “taggata”*
    - *….*
  - Si compone il sistema nell’area di lavoro
- *Change set model*
  - Si parte da una configurazione base (“Base line”)
  - Si scelgono i cambiamenti effettuati da includere (“Change set”)
  - Si ottiene una configurazione nell’area di lavoro
    - *Simile ai meccanismi di aggiornamento di Windows*
    - *Moltissime configurazioni possibili, non necessariamente tutte testate e funzionanti*

Paradigma check-in/check-out

- Modifica della copia nello spazio di lavoro (isolamento)
- Check-in
  - *deposita una nuova versione di un modulo nel repository (se consentito, se presente lock, …)*
  - *criteri per stabilire identificatore nuova versione (vecchia + 1, manualmente, regole, …)*
  - *Permette una gestione “single writer, multiple readers”*
  - *Logica del “locking pessimistico”*

Paradigma check-in/check-out

- *Il paradigma check-in/check-out consente che un solo utente per volta possa modificare un modulo*
- *"Serializzazione" (check-out/modifica/check-in) del lavoro: a volte non è applicabile*
  - più programmatori devono applicare modifiche a parti diverse di un modulo
  - i programmatori lavorano in sedi diversi e devono operare sullo stesso modulo
- *Rilassamento delle politiche di locking:*
  - accesso libero ad un modulo (check-out senza locking)
  - procedure per la verifica di inconsistenze all'atto del check-in
  - "riconciliazione" fra diverse versioni dello stesso modulo semi-automatica (nei casi più semplici) o manuale
  - consente di operare in parallelo sullo stesso componente
- *Logica del "locking ottimistico"*

Paradigma check-in/check-out

- Basta questa metodologia?
- Long Transactions
  - *We do our changes as logical units, we want this logical unit to be kept:*
    - if there is anything that conflicts, we must abort the whole commit (long transactions)
  - *So far we have looked only at one type of conflicts:*
    - people changing the same file in parallel

Paradigma check-in/check-out

- *But we are interested in whole systems/configurations:*
  - can we merge systems/configurations?
  - in theory, yes! – but in practice, NO! (strict LT)
- *La "Strict Long Transiction"*
  - Sostanzialmente racchiude un insieme di file che sono considerati o scartati insieme
  - verifica come processo anche le dipendenze logiche (legami fra sistemi chiamanti/chiamati)

Configuration Status Accounting

- Raccolta e distribuzione delle informazioni relative a:
  - *stato delle richieste di cambiamento proposte*
  - *stato delle richieste di cambiamento approvate ed in corso di implementazione*
- L'Audit implementa il controllo di consistenza del processo
  - *Audit fisico: ci sono tutti i file*
  - *Audit funzionale: tutte le funzioni hanno seguito il processo*

Release management

- Releases must incorporate changes forced on the system by errors discovered by users and by hardware changes
- They must also incorporate new system functionality
- Release planning is concerned with when to issue a system version (configuration) as a release
- Release gives one version (configuration) external visibility

System releases

- Not just a set of executable programs
- May also include
  - *Configuration files defining how the system is configured for a particular installation*
  - *Data files needed for system operation*
  - *An installation program or shell script to install the system on target hardware*
  - *Electronic and paper documentation*
  - *Packaging and associated publicity*
- Systems are now normally released on DVD-ROM or as downloadable installation files from the web

Release problems

- Customer may not want a new release of the system
  - *They may be happy with their current system as the new version may provide unwanted functionality*
- Release management must not assume that all previous releases have been accepted. All files required for a release should be re-created when a new release is installed

Release decision making

- Preparing and distributing a system release is an expensive process
- Factors such as the technical quality of the system, competition, marketing requirements and customer change requests should all influence the decision of when to issue a new system release

System release strategy

Factor	Description
Technical quality of the system	If serious system faults are reported which affect the way in which many customers use the system, it may be necessary to issue a fault repair release. However, minor system faults may be repaired by issuing patches (often distributed over the Internet) that can be applied to the current release of the system.
Lehman's fifth law	This suggests that the increment of functionality which is included in each release is approximately constant. Therefore, if there has been a system release with significant new functionality, then it may have to be followed by a repair release.
Competition	A new system release may be necessary because a competing product is available.
Marketing requirements	The marketing department of an organisation may have made a commitment for releases to be available at a particular date.
Customer change proposals	For customised systems, customers may have made and paid for a specific set of system change proposals and they expect a system release as soon as these have been implemented.

Release creation

- Release creation involves collecting all files and documentation required to create a system release
- Configuration descriptions have to be written for different hardware and installation scripts have to be written
- The specific release must be documented to record exactly what files were used to create it. This allows it to be re-created if necessary

System building

- The process of compiling and linking software components into an executable system
- Different systems are built from different combinations of components
- Invariably supported by automated tools that are driven by ‘build scripts’
- *Quando si crea un sistema di lavoro in un working-space si ha una versione “personale” del System-building, usualmente parziale rispetto al System-building completo che è quello legato ad una release esterna del sistema.*

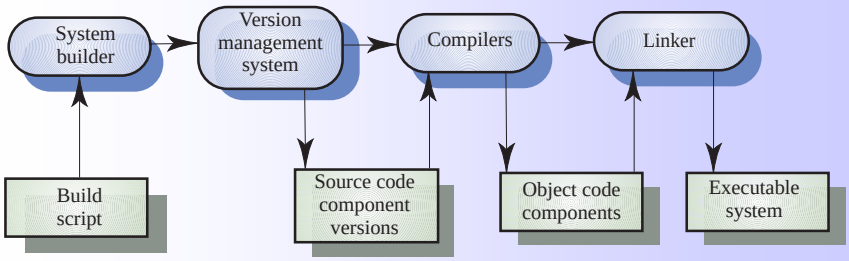
System building problems

- Do the build instructions include all required components?
  - *When there are many hundreds of components making up a system, it is easy to miss one out. This should normally be detected by the linker*
- Is the appropriate component version specified?
  - *A more significant problem. A system built with the wrong version may work initially but fail after delivery*
- Are all data files available?
  - *The build should not rely on 'standard' data files. Standards vary from place to place*

System building problems

- Are data file references within components correct?
  - *Embedding absolute names in code almost always causes problems as naming conventions differ from place to place*
- Is the system being built for the right platform
  - *Sometimes must build for a specific OS version or hardware configuration*
- Is the right version of the compiler and other software tools specified?
  - *Different compiler versions may actually generate different code and the compiled component will exhibit different behaviour*

System building



System representation

- Systems are normally represented for building by specifying the file name to be processed by building tools. Dependencies between these are described to the building tools
- Mistakes can be made as users lose track of which objects are stored in which files
- A system modelling language addresses this problem by using a logical rather than a physical system representation

CASE tools for configuration management

- CM processes are standardised and involve applying pre-defined procedures
- Large amounts of data must be managed
- CASE tool support for CM is therefore essential
- Mature CASE tools to support configuration management are available ranging from stand-alone tools to integrated CM systems

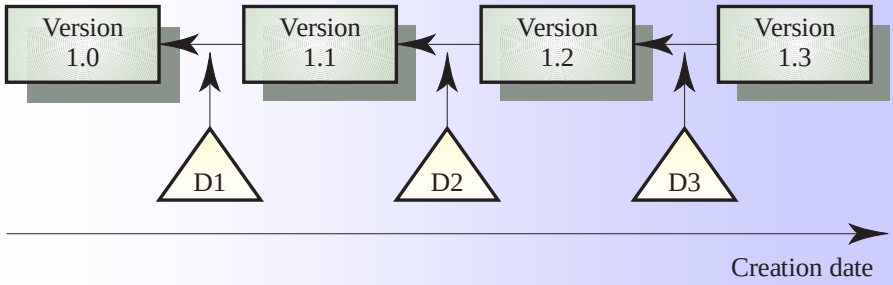
Change management tools

- Change management is a procedural process so it can be modelled and integrated with a version management system
- Change management tools
  - *Form editor to support processing the change request forms*
  - *Workflow system to define who does what and to automate information transfer*
  - *Change database that manages change proposals and is linked to a Version Management System*

Version management tools

- Version and release identification
  - *Systems assign identifiers automatically when a new version is submitted to the system*
- Storage management
  - *System stores the differences between versions rather than all the version code*
- Change history recording
  - *Record reasons for version creation*
- Independent development
  - *Only one version at a time may be checked out for change. Parallel working on different versions*

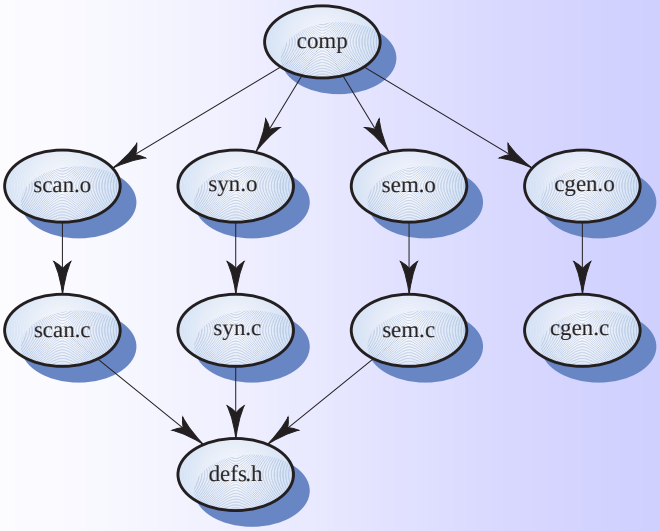
Delta-based versioning tools



System building tools

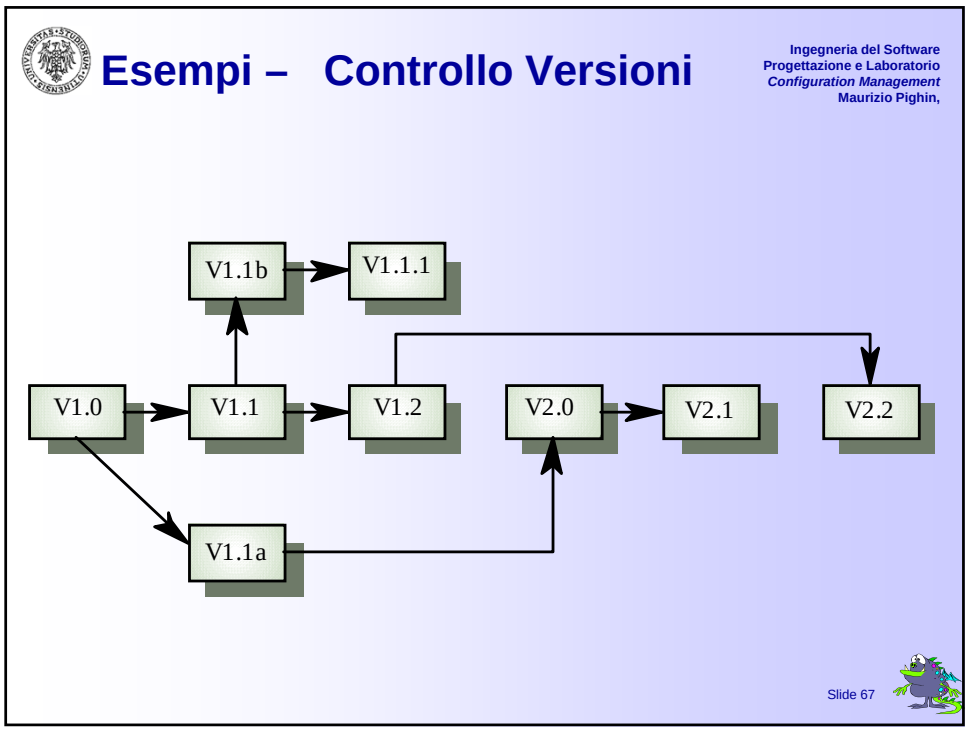
- Building a large system is computationally expensive and may take several hours
- Hundreds of files may be involved
- System building tools may provide
  - *A dependency specification language and interpreter*
  - *Tool selection and instantiation support*
  - *Distributed compilation*
  - *Derived object management*

Component dependencies



Esempi – Controllo Versioni

- RCS: a System for Version Control
- Controllo delle versioni: parte del CM che consente di gestire le versioni degli elementi
- RCS consente di gestire le diverse versioni di un elemento
- Attenzione: RCS non supporta direttamente gli altri processi descritti
- Versioni di un elemento: albero (nel caso più generale un grafo diretto aciclico)
  - *Nuova versione: aggiunta di un nodo e di un arco che lo congiunge con il nodo che rappresenta la versione da cui è derivato*
  - *un nodo può avere più figli nel caso in cui si siano derivate diverse versioni (parallele)*
  - *due rami dell'albero possono ricongiungersi in alcuni casi particolari*



Esempi – Controllo Versioni

Esempi – Controllo Versioni

- Caratteristiche di RCS
  - *Comandi per check-in e check-out dei moduli, comandi per la gestione dello spazio di lavoro*
  - *Check-in: ci*
    - inglobare nello spazio di lavoro la nuova versione di un file o crearne la versione iniziale
    - inserire un commento che descriva i cambiamenti apportati al file
    - assegna automaticamente il numero di versione o consente di specificarlo manualmente (ramificazioni, vedi esempio precedente)
  - *Check-out: co*
    - copia di lavoro di una particolare versione di un file (specificando la versione desiderata o operare selezione su altri attributi)
    - consente di specificare se si intende estrarre il file per modificarlo o solo per poterlo visionare

Esempi – Controllo Versioni

- *Identificazione dei file: ident*
  - associare ad ogni file un identificatore costruito automaticamente a partire da diversi attributi (nome, versione, autore, stato, data di creazione o dell'ultimo check-in)
  - identificare i file che costituiscono un programma eseguibile estraendo tale identificatore (dipende dal linguaggio utilizzato)
- *Rapporti e log: rlog*
  - estrarre da un elemento un insieme di informazioni (es. l'insieme dei commenti che descrivono le modifiche apportate in corrispondenza delle diverse versioni)
- *Altre operazioni*
  - cambiamento dello stato e degli attributi dei file
  - confronto fra le diverse versioni
  - fusione fra diverse versioni

Esempi – Controllo Versioni

- *Memorizzazione versioni: delta*
  - memorizzata solo l'ultima versione del file
  - differenze rispetto alla versione successiva per tutte le versioni precedenti
  - si risparmia spazio a costo di overhead per check-in/check-out
- *Lock delle versioni*
  - meccanismo di lock (default, ma aggirabile) per serializzare modifiche
  - all'atto del check-out è possibile specificare che si intende modificare il file
  - successivi check-out per modifica della stessa versione da parte di altri utenti vengono rifiutati (consentiti check-out per visione)
  - all'atto del check-in si verifica che l'utente abbia acquisito un lock per la versione precedente

Esempi – Controllo Versioni

- CVS: Concurrent Versions System
  - *Logicamente ingloba RCS:*
    - Aggiunge la gestione di alberi di directory (RCS opera su file singoli)
    - Consente di operare in parallelo sullo stesso modulo (rilassa la politica di locking e aggiunge meccanismi di riconciliazione)
    - Opera in modalità client/server (RCS è "monolitico")
  - *Internamente:*
    - E' "come se" il repository fosse gestito attraverso i comandi di RCS (ci, co, …), applicati ripetutamente per più file/directory
    - CVS consente modifiche contemporanee allo stesso modulo - più persone possono fare check-out (no locking)
    - CVS verifica le inconsistenze all'atto del check-in ed impone di risolverle (semi-automaticamente)

Esempi – System Building

- MAKE
  - *Based on a physical rather than a logical model of dependencies*
  - *Dependency specifications (Makefiles) quickly become large, complex, hard to understand and expensive to maintain*
  - *MAKE uses a simple model of change based on file update times. Source code changes NEED not require re-compilation*
  - *MAKE does not (easily) allow versions of tools such as the compiler to be specified*
  - *Not tightly linked to version management tools. Manual intervention is usually needed to check out source code from a version management system for building*