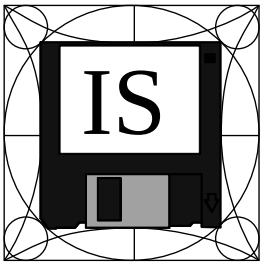
Amministrazione di progetto



Corso di Ingegneria del Software

V. Ambriola, G.A. Cignoni,
C. Montangero, L. Semini

Con aggiornamenti di: T. Vardanega (UniPD)

Contenuti

- Amministrazione di progetto
- Documentazione di progetto
- Ambiente e strumenti
- Norme di progetto e di codifica
- Seminario: leggibilità del codice

Amministrare un progetto

- Amministrare non è dirigere
  - L'amministratore non compie scelte tecnologiche
  - L'amministratore non compie scelte gestionali
- Attività
  - Redazione e manutenzione delle regole
    - La loro approvazione spetta al responsabile di progetto
  - Accertamento di applicazione delle regole
  - Responsabilità su reperimento e disponibilità delle risorse
    - Per tutte le risorse di un progetto tranne che per il personale
    - Ambiente, infrastruttura, strumenti, prodotti, documenti

Documentazione di progetto

- Tutto ciò che documenta le attività
  - Riguardo al prodotto
  - Riguardo al processo
- Documenti di sviluppo
  - Documentazione fornita dal cliente
  - Diagrammi di progettazione
  - Codice
  - Piani di qualifica e risultati delle prove
  - Documentazione di accompagnamento del prodotto
- Documenti di gestione del progetto
  - Documenti contrattuali
  - Piani e consuntivi delle attività
  - Piani di qualità

Disponibilità e diffusione

- I documenti sono utili solo se sono sempre disponibili
  - Chiaramente identificati
  - Corretti nei contenuti
  - Verificati e approvati
  - Aggiornati, datati e dotati di versione
- La loro diffusione deve essere controllata
  - I destinatari devono essere chiaramente identificati
    - Ogni documento ha una sua lista di distribuzione
    - L'amministratore gestisce le liste di distribuzione e ne assicura il rispetto

Ambiente di lavoro

- Quanto serve ai processi di produzione
- La qualità dell'ambiente determina la produttività
  - Influisce sulla qualità del processo
  - Influisce sulla qualità del prodotto
- Caratteristiche di qualità dell'ambiente di lavoro
  - Completo
    - Deve offrire tutto il necessario per svolgere le attività previste
  - Ordinato
    - Deve essere facile trovarvi ciò che si cerca
  - Aggiornato
    - Il materiale obsoleto non deve causare intralcio

Infrastruttura

- **Risorse HW**
  - ***Server***
    - Per ospitare archivi logici centralizzati di prodotti sempre aggiornati e accessibili
  - **Infrastruttura di rete**
    - Sempre operativa, protetta, accedibile anche da remoto
  - **Postazioni di lavoro e dispositivi di utilità**
    - Per assicurare la massima produttività individuale
  - **Archivi fisici**
- **Risorse SW**
  - **Ambienti di sviluppo, prova, studio e documentazione**
    - Strumenti di sviluppo
    - Prodotti del processo di sviluppo (documentazione inclusa)
  - **Archivio logico (*repository)* di prodotti di progetto**

Supporto di processi – 1

- **Gestione di progetto**
  - **Pianificazione, stima e controllo dei costi**
  - **Allocazione e gestione delle risorse**
    - Redazione e consultazione di diagrammi di Gantt e PERT (p.es., http://www.ganttproject.biz/, oppure SpiderPlan @ per-approfondire-12)
  - **Strumenti collaborativi di controllo gestionale e di qualità e di coordinamento attività**
    - Assembla (http://www.assembla.com)
    - Maven (http://maven.apache.org)
    - Jira (http://www.atlassian.com/software/jira)
- **Gestione documentale**
  - **TWiki (http://www.twiki.org)**
  - **Google Docs (http://docs.google.com)**
  - **Versionamento e configurazione**

Supporto di processi – 2

- **Analisi e progettazione**
  - **Analisi, gestione e tracciamento dei requisiti**
    - eRequirements (http://erequirements.com/app)
  - **Supporto alle metodologie**
    - UML (p.es., http://www.eclipse.org/modeling/mdt/papyrus/)
- **Codifica e integrazione**
  - **Ambienti integrati di sviluppo (p.es., http://www.eclipse.org/)**
  - **Strumenti di integrazione continua (*continuous integration*)**
    - Hudson (http://hudson.dev.java.net)
    - CruiseControl (http://cruisecontrol.sourceforge.net/index.html)
    - Merlin ToolChain (http://merlintoolchain.sourceforge.net)
  - **Misurazione e analisi statica del codice prima dell'integrazione**
  - **Generazione ed esecuzione automatica delle prove prima dell'integrazione**

Configurazione

- **Un prodotto SW è l'unione di parti distinte unite insieme seguendo regole rigorose**
  - **Specifiche, progetti, programmi, dati di verifica, manualistica**
- **Le regole di configurazione vanno pianificate**
  - **Le responsabilità di configurazione vanno assegnate**
- **La gestione di configurazione va automatizzata**
  - **Servono strumenti adatti**
    - *Configuration Management*
    - *Build*

Gestione di configurazione – 1

- **Obiettivi**
  - **Mettere in sicurezza la *baseline***
  - **Prevenire sovrascritture accidentali**
  - **Consentire ritorno a versioni precedenti**
  - **Permettere il recupero da perdite accidentali**
- **Attività**
  - **Identificazione di configurazione**
  - **Controllo di *baseline***
  - **Gestione delle modifiche**
  - **Controllo di versione**

Gestione di configurazione – 2

- **Identificazione di configurazione**
  - **Decidere quali sono le parti (*configuration item*, CI) che compongono il prodotto**
  - **Attribuire una identità unica a ciascun CI**
    - ID, nome, data, autore, registro delle modifiche, stato corrente
- **Controllo di *baseline***
  - **Un insieme di CI a una specifica *milestone***
  - **Base verificata, approvata e garantita per la prosecuzione dello sviluppo**
  - **L'esistenza di *baseline* ben identificate permette**
    - Riproducibilità
    - Tracciabilità
    - Analisi e confronto

Gestione delle modifiche – 1

- Le richieste di modifiche hanno origine da
  - Utenti (difetti o mancanze)
  - Sviluppatori (idem)
  - Competizione (valore aggiunto)
- Le richieste di modifica vanno sottoposte a un rigoroso processo di analisi, decisione, realizzazione e verifica
  - Sempre tenendo traccia dello stato precedente

Gestione delle modifiche – 2

- Ogni richiesta/proposta di modifica va inoltrata in modo formale
  - *Change request form*
    - Autore, motivo, urgenza
    - Stima di fattibilità, valutazione di impatto, stima di costo
    - Decisione del responsabile
- Di ogni richiesta di modifica bisogna tenere traccia
  - *Tracking e/o ticketing*
    - Per esempio con Bugzilla
  - Stato corrente ed eventuale esito chiusura

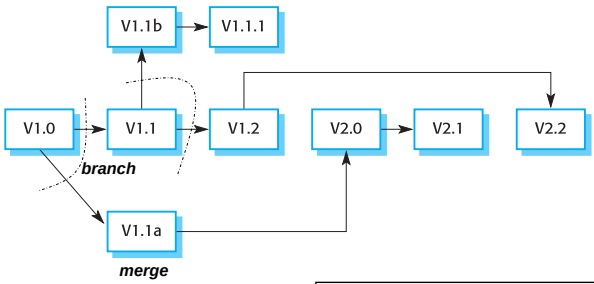
Controllo di versione – 1

- Si appoggia su un *repository*
  - DB centralizzato nel quale risiedono tutti i CI di ogni *baseline* nella loro storia completa
- Permette a ciascuno di lavorare su vecchi e nuovi CI senza rischio di sovrascritture accidentali
  - *Checkout*
- E di condividere il lavorato nello spazio comune
  - *Checkin*
- Verifica la bontà di ogni modifica di *baseline*
  - *Build*

Controllo di versione – 2

- Versione
  - Istanza di prodotto funzionalmente distinta dalle altre
- Variante
  - Istanza di prodotto funzionalmente identica ad altre ma diversa per caratteristiche non funzionali
- Rilascio (*release*)
  - Istanza di prodotto resa disponibile a utenti esterni
- Tutte vanno identificate, pianificate e gestite
  - Identificazione per numero, caratteristiche, modifiche

Controllo di versione – 3



Tratto da: Ian Sommerville, *Software Engineering*, 8th ed.

Strumenti di lavoro

- Non solo compilatore e *debugger* !
- Produrre codice secondo regole
  - Editore integrato con verificatore di regole e stile
- Verificare il codice a partire dalle unità più piccole
  - Strumenti di automazione delle verifiche
- Versionamento per tener traccia della "storia" dei prodotti
  - CVS (*concurrent versions system*, http://www.nongnu.org/cvs/)
  - SVN (*subversion*, http://subversion.tigris.org/)
- Configurazione (*build*) per integrare le parti del prodotto finale
  - Serva più intelligenza di "make"
  - Ant (http://ant.apache.org/), Maven (http://maven.apache.org/)

Norme di progetto

- **Linee guida per le attività di sviluppo**
  - In origine precedevano processi e procedure aziendali
  - Oggi sono uno strumento operativo di complemento alle procedure
- **Contenuti**
  - Organizzazione e uso delle risorse di sviluppo
  - Convenzioni sull'uso degli strumenti di sviluppo
  - Organizzazione della comunicazione e della cooperazione
  - Norme di codifica
  - Gestione dei cambiamenti!

Organizzazione di una norma

- **Regole**
  - Convenzioni di cui si riconosce necessità e convenienza
  - Ne è richiesto e accertato il rispetto
- **Raccomandazioni**
  - Prassi desiderabile
  - Inviti e suggerimenti senza verifica di rispetto
- **Il contesto definisce la portata della norma**
  - Non tutto può essere regolato
  - Troppe regole sono di difficile attuazione e verifica

Obiettivi delle norme di codifica

- **Leggibilità come forma di prevenzione**
  - Verificabilità
  - Manutenibilità
  - Portabilità
- **Come è "scritto" il codice?**
- **È comprensibile a distanza di tempo?**
- **È comprensibile a chi non lo ha prodotto?**

Convenzioni sui nomi

- **Nel codice**
  - Tipi, costanti, variabili, funzioni, ...
    - P.es., le norme **Javadoc** (vedi: http://java.sun.com/j2se/javadoc/)
- **Nel progetto**
  - Strutturazione in moduli, *file*, *directory*, ...
- **Aspetti pratici**
  - Conflitti logici all'interno o all'esterno del codice
  - Abbreviazioni, per comodità o per necessità
  - Limiti intrinseci del linguaggio
    - P.es., identificazione forte o debole dei tipi (Java ↔ C)
  - Limiti degli strumenti
    - P.es., lunghezza massima dei nomi di *file* (p.es.: Windows 95-98)

Indentazione del codice

- **Obiettivi**
  - Programmazione strutturata
  - Evidenziare visivamente la struttura di un programma
- **Aspetti da non sottovalutare**
  - Lunghezza delle linee
  - Indentazione
  - Posizione degli fine linea nei blocchi
  - Posizione degli fine linea nelle espressioni
- **Come evitare guerre ideologiche sugli stili?**

Intestazione del codice

- **Obiettivi**
  - Identificazione e collocamento di una unità (modulo, *file*)
  - Storia e responsabilità delle modifiche
- **Contenuti**
  - Dati dell'unità — tipo, contenuto, posizione
  - Responsabilità — autore, reparto, organizzazione
  - *Copyright* / *copyleft* — licenze, visibilità
  - Avvertenze — limiti di uso e di garanzia
  - Registro modifiche — storia, spiegazione, versione

Intestazione – esempio 1

```
// File:    HAL_kern.H - HAL 9000 KB Data defs -*- C++ -*-
// Module:  HAL 9000 KB kernel
// Created: 1997 January 12
// Author:  Dr. Chandra - 9000 Proj., HAL Inc., Urbana, ILL
// E-Mail:  chandra@p9000.hal.com
//
// Copyright (C) 1996, 1997, Dr. Chandra, HAL Inc.
//               All rights reserved.
//
// This software and related documentation are
// distributed under license. No permission is given
// to use, copy, modify or distribute this software
// without explicit authorization of HAL Inc.
// and its licensors, if any.
//
// Software licensed to:
// NO LICENSE - For HAL internal use only.
//
// This software is provided "as is" WITHOUT ANY WARRANTY
// either expressed or implied, including, but not limited
// to, the implied warranties of MERCHANTABILITY or
// FITNESS FOR A PARTICULAR PURPOSE.
```


Intestazione – esempio 2

```
// 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  Modifier  Date        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
```

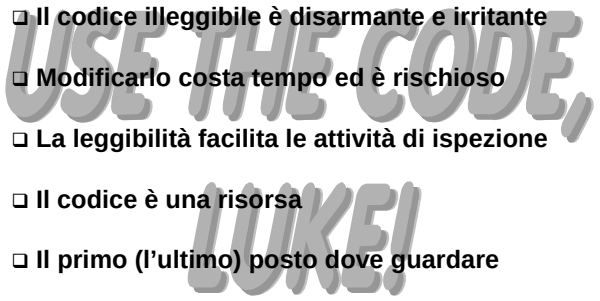
Tratto da: Ian Sommerville, *Software Engineering*, 8th ed.

Uso del linguaggio di programmazione

- Serve una strategia forte per costringere i programmatori a lavorare come si conviene
- Prescrizioni tipiche
  - Compilazione senza errori fatali o potenziali (*warning*)
  - Uso chiaro e coerente dei costrutti del linguaggio
  - Uso di un sottoinsieme appropriato del linguaggio
    - I costrutti di maggiore robustezza, verificabilità, leggibilità
    - Non necessariamente quelli di maggiore potenza espressa e velocità

Leggibilità del codice

- Il codice illeggibile è disarmante e irritante
- Modificarlo costa tempo ed è rischioso
- La leggibilità facilita le attività di ispezione
- Il codice è una risorsa
- Il primo (l'ultimo) posto dove guardare



Norme di codifica

- IOCCC Roemer.c

  International Obfuscated C Code Contest

Notazione ungherese ☺

- `lpfnWndProc` (un campo di struttura)
  - `l` *long* (32-bit integer)
  - `p` *pointer* [to a]
  - `fn` *function* [handling messages directed to]
  - `Wnd` [a] *window*
  - `Proc` *procedure*
  - Un vettore di puntatori a descrittori di finestre, indicizzato sul numero di finestre

- Un'idea di Charles Simonyi @ Microsoft

Riferimenti

- V. Ambriola, G.A. Cignoni, “Laboratorio di progettazione”, Jackson Libri, 1996
- “Programming in C++ – Rules and Recommendations”, Ellemtel TSL (Svezia), 1992
- C. Simonyi, M. Heller, “The Hungarian Revolution”, Byte, agosto 1991
- F. Lanubile et al., “Collaboration Tools for Global Software Engineering”, IEEE Software, 27:2, 2010, 52-55
- J. Portillo Rodriguez et al., “Technologies and Tools for Distributed Teams”, IEEE Software, 27:5, 2010, 10-14