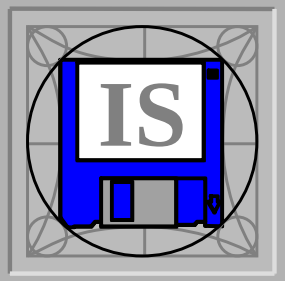
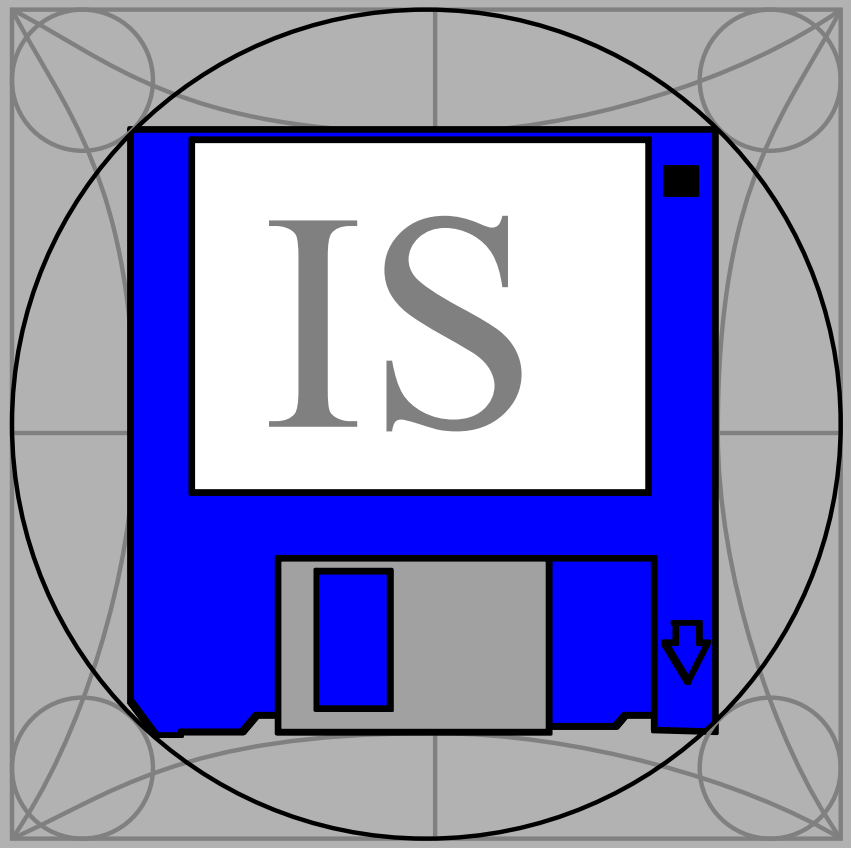


Amministrazione di progetto

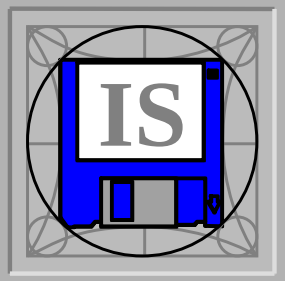
Spunti per *Flipped Classroom*



Ingegneria del Software

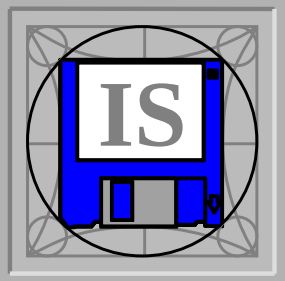
V. Ambriola, G.A. Cignoni,
C. Montangero, L. Semini

Aggiornamenti: T. Vardanega (UniPD)



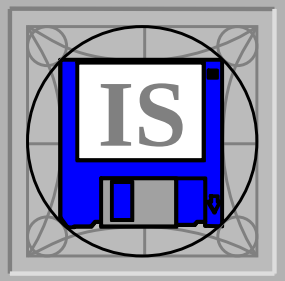
Cosa fa l'amministratore

- **Equipaggia, organizza, documenta e gestisce l'ambiente di lavoro e di produzione del *team* di progetto**
  - **Regole, procedure, strumenti (servizi informatici)**
    - Un servizio standardizzato è tale se fornisce al suo utente valore superiore ai costi (di acquisizione e di uso), riducendo il rischio di improvvisazione e soggettività
  - **A supporto del *way of working* adottato**
- **Attua le scelte procedurali e tecnologiche fissate dal responsabile**
  - **Nel vostro caso, meglio che inizialmente le scelte siano collettive**



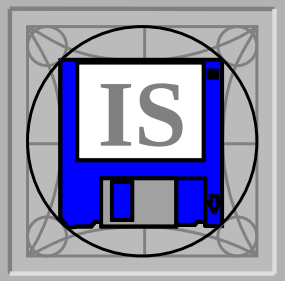
Normare il *way of working*

- Le norme forniscono linee guida per le tutte le attività di progetto
  - Spiegano e abilitano l'attuazione dei processi adottati
  - Organizzate per processi, e le relative procedure, e gli strumenti a supporto
  - Specificano convenzioni sull'uso degli strumenti scelti
  - Organizzano la comunicazione e la cooperazione



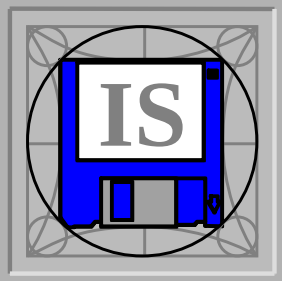
Ambiente di lavoro

- Fornisce quanto serve ai processi di produzione
  - Di sviluppo, di supporto, di organizzazione
  - L'ambiente è fatto da persone, ruoli, procedure, strumenti
- La qualità dell'ambiente determina la produttività
  - Influendo sulla qualità del processo e del prodotto
- Deve essere
  - Completo: tutto il necessario per svolgere le attività previste
  - Ordinato: è facile trovarvi ciò che si cerca
  - Aggiornato: il materiale obsoleto non deve causare intralcio



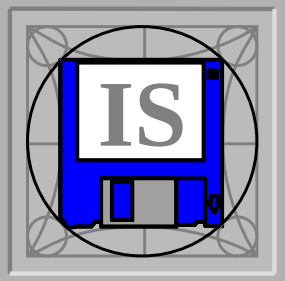
Supporto a gestione di progetto

- **Pianificazione, stima e controllo dei costi**
  - **Allocazione e gestione delle risorse**
    - P.es, InstaGantt https://instagantt.com/
- **Strumenti collaborativi di controllo gestionale, qualità, coordinamento attività**
  - **Come quelli di *issue tracking* / *ticketing***
    - P.es. **Assembla** (http://www.assembla.com), **Jira** (http://www.atlassian.com/software/jira)
- **Gestione documentale**
  - **Controllo di accesso in lettura e scrittura, tracciamento delle modifiche, ripristino**
    - P.es, Google Docs (http://docs.google.com)
  - **Versionamento e configurazione (vedi seguito)**



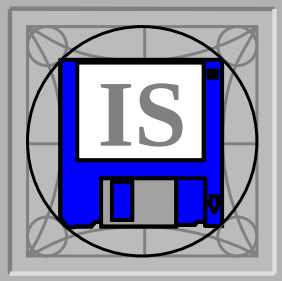
Supporto a sviluppo – 1

- **Analisi dei requisiti**
  - **Raccolta, classificazione, tracciamento**
    - eRequirements (http://erequirements.com/app)
  - **Correlazione con i diagrammi dei casi d'uso**
- **Progettazione**
  - **Ambiente UML per diagrammi, metriche di qualità, generazione di codice**
    - http://www.eclipse.org/papyrus/
    - http://www.modelio.org/



Supporto a sviluppo – 2

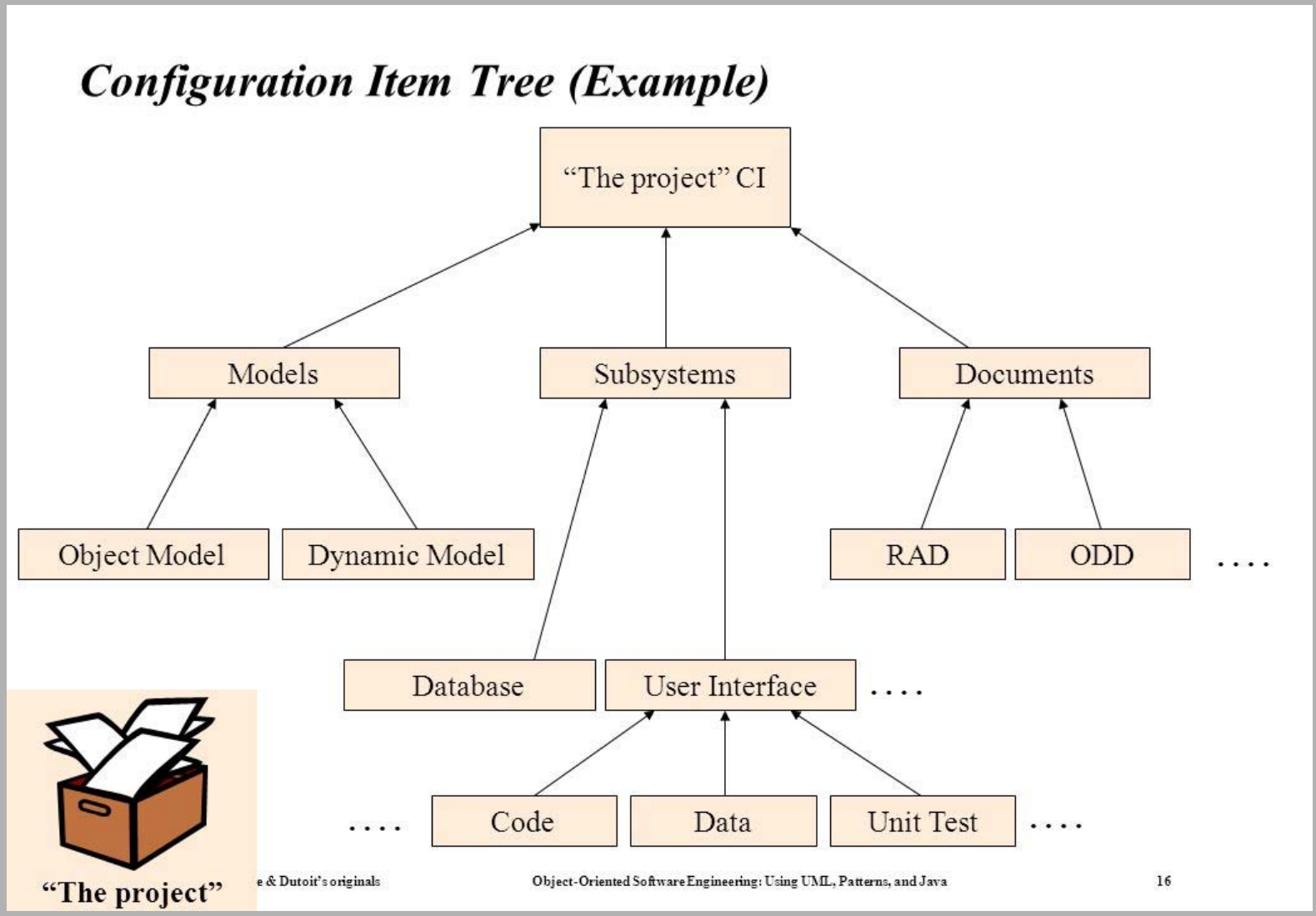
- **Codifica e integrazione**
  - **Ambienti integrati di sviluppo, IDE**
  - **Integrazione continua (*continuous integration*)**
    - **Hudson** (http://hudson-ci.org/)
    - **CruiseControl** (http://cruisecontrol.sourceforge.net)
  - **Misurazione e analisi del codice prima dell'integrazione**
  - **Generazione ed esecuzione automatica delle prove prima dell'integrazione**

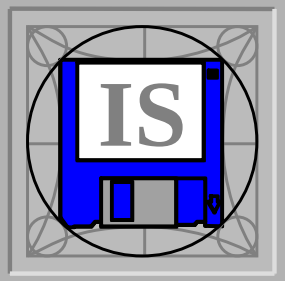


Configurazione – 1

- Un progetto (*project*) ha un singolo prodotto in uscita,
  - Aggregato di distinti sotto-prodotti
    - Specifiche (analisi, *design*), diagrammi, sorgenti, direttive, *test*, manuali, documenti di gestione
- L'aggregazione di parti in un singolo sotto-prodotto è detta ***Build***
  - Nell'integrazione continua, ogni build costituisce una ***baseline***
  - Ogni configurazione di ***baseline*** va messa in sicurezza
- L'insieme delle parti, ordinato secondo regole precise, è detto **configurazione**
  - Le regole di configurazione, ***Configuration Management***, sono parte del ***way of working***
  - La loro attuazione va automatizzata

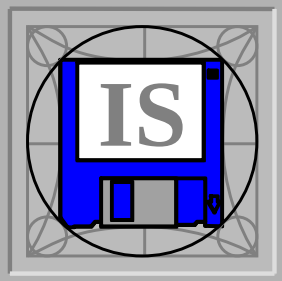
Configurazione – 2





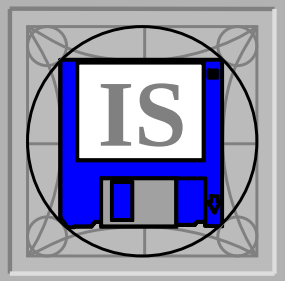
Attività di configurazione – 1

- **Identificazione di configurazione [1]**
  - **Le parti (*configuration item*, CI) che compongono il prodotto**
  - **Ogni CI ha una identità unica**
    - ID, nome, data, autore, registro delle modifiche, stato corrente
- **Controllo di *baseline* [2]**
  - ***Baseline*: insieme di CI consolidato a un dato istante (*milestone*)**
    - Base verificata, approvata e certa per la prosecuzione del progetto



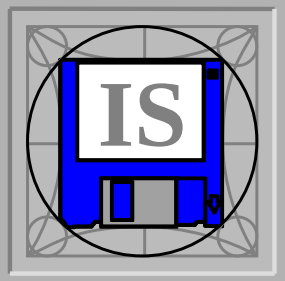
Attività di configurazione – 2

- **Controllo di *baseline* [2]**
  - **Un progetto prevede una successione di *baseline***
    - Che va gestita con processi dedicati
  - **Una *milestone* è una data di calendario che denota un punto di avanzamento atteso, sostanziato da una o più *baseline***
  - **L'esistenza di *baseline* ben identificate garantisce**
    - Riproducibilità
    - Tracciabilità
    - Analisi, valutazione, confronto



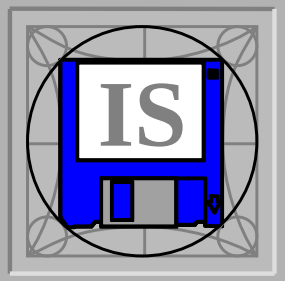
Buone qualità di *milestone*

1. Specifiche per obiettivi di avanzamento, dimostrabili agli *stakeholder*
2. Delimitate per ampiezza e ambizioni
3. Incrementali
4. Coerenti con e rilevanti per la strategia di progetto
5. Misurabili per quantità di impegno necessario
6. Traducibili in compiti assegnabili
7. Raggiungibili
8. Tempestive rispetto alle esigenze di calendario



Attività di configurazione – 3

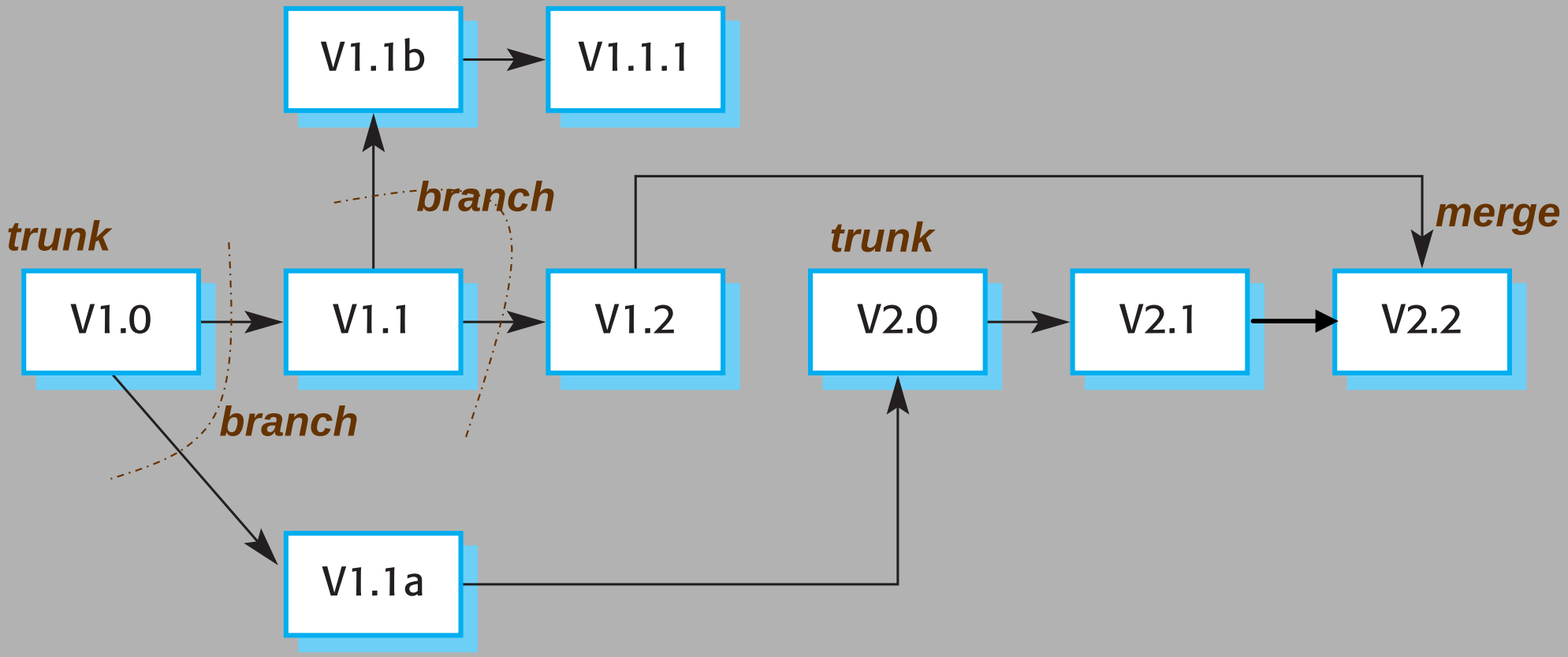
- **Controllo di versione [3]**
  - **Si appoggia su un *repository***
    - DB centralizzato ove risiedono individualmente tutti i CI di ogni *baseline* con la loro storia completa
  - **Permette a ciascuno di lavorare su vecchi e nuovi CI senza rischio di sovrascritture accidentali**
    - *Check-out*
  - **E di condividere il lavorato nello spazio comune**
    - *Check-in → commit*
  - **Verifica la bontà di ogni modifica di baseline**
    - *Build*



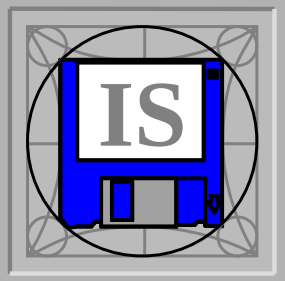
Glossario – 8

- **Versione**
  - Istanza di CI, con identificatore unico, funzionalmente distinta dalle altre
  - Con traccia delle differenze rispetto alla versione precedente
- **Variante**
  - Istanza di CI funzionalmente identica ad altre ma diversa per caratteristiche non funzionali
- **Rilascio (*release*)**
  - Istanza di prodotto resa disponibile a utenti esterni

Esempio di storia di versioni

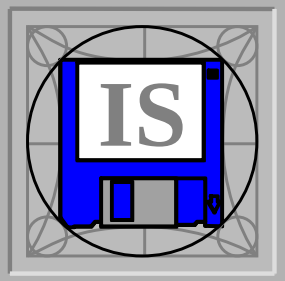


Tratto da: Ian Sommerville, *Software Engineering*, 8th ed.



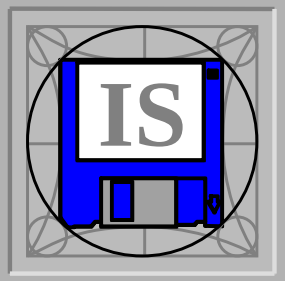
Attività di configurazione – 4

- **Gestione delle modifiche [4]**
  - **Le richieste di modifiche hanno origine da**
    - Utenti (segnalazione di difetti o mancanze)
    - Sviluppatori (idem)
    - Competizione (identificazione di valore aggiunto)
  - **Ogni richiesta di modifica va sottoposta a un rigoroso processo di analisi, decisione, realizzazione e verifica**
    - Chiamato *change management*



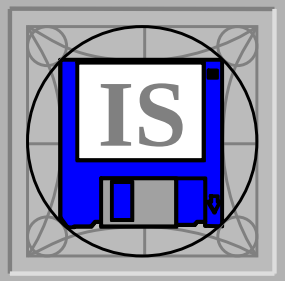
Gestione delle modifiche

- Ogni richiesta di modifica va gestita in modo formale
  - Tramite procedura di *change request*
  - Che identifica autore, motivo, urgenza
  - Con stima di fattibilità, costo e valutazione di impatto
  - Con decisione del responsabile
- Di ogni richiesta di modifica bisogna tenere traccia
  - Tramite *issue tracking / ticketing*
  - Tracciandone lo stato di avanzamento fino a chiusura



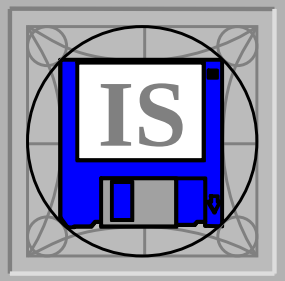
Norme di codifica

- 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?
- Ogni comunità di linguaggio ha sue buone prassi e convenzioni di programmazione



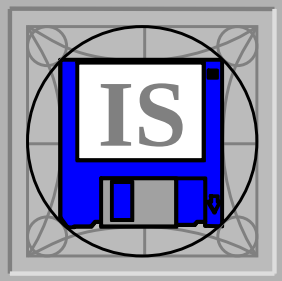
Intestazione del codice

- **Obiettivi**
  - Identificazione e collocamento di 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



Indentazione del codice

- Obiettivi
  - Programmazione strutturata
  - Evidenziare visivamente la struttura di un programma
- Aspetti da non sottovalutare
  - Lunghezza delle linee
  - Ampiezza dell'indentazione
  - Posizione degli fine linea nei blocchi
  - Posizione degli fine linea nelle espressioni
- Evitare guerre ideologiche sugli stili



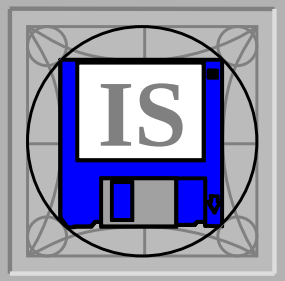
Esempi di intestazione – 1

```
// 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.
```

Esempi di intestazione – 2

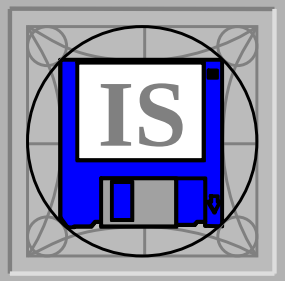
```
// BANKSEC project (IST 6087)
//
// BANKSEC-TOOLS/AUTH/RBAC/USER_ROLE
//
// Object: currentRole
// Author: N. Perwaiz
// Creation date: 10th November 2002
//
// © Lancaster University 2002
//
// Modification history
// Version      Modifier     Date          Change          Reason
// 1.0          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.



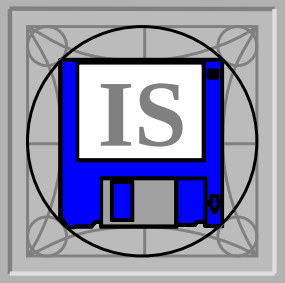
Disciplina di programmazione

- **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

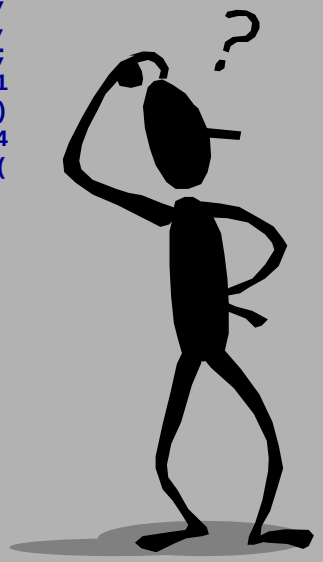


Norme di codifica

- **IOCCC Roemer.c**

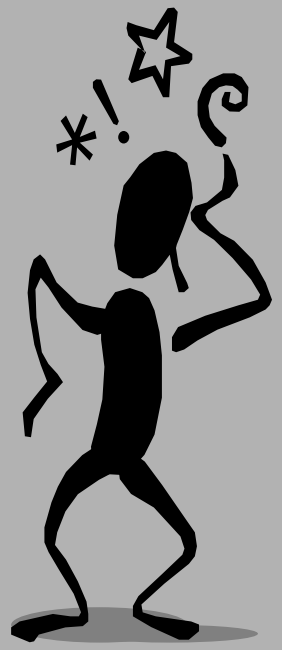
  International Obfuscated C Code Contest

```
                                                                char
                                                        _3141592654[3141
        ],__3141[3141];_314159[31415],_3141[31415];main(){register char*
     _3_141,*_3_1415, *_3__1415; register int _314,_31415,__31415,*_31,
    _3_14159,__3_1415;*_3141592654=__31415=2,_3141592654[0][_3141592654
   -1]=1[__3141]=5;__3_1415=1;do{_3_14159=_314=0,__31415++;for( _31415
  =0;_31415<(3,14-4)*__31415;_31415++)_31415[_3141]=_314159[_31415]= -
1;_3141[*_314159=_3_14159]=_314;_3_141=_3141592654+__3_1415;_3_1415=
__3_1415    +__3141;for                     (_31415 = 3141-
          __3_1415  ;                       _31415;_31415--
          ,_3_141 ++,                       _3_1415++){_314
          +=_314<<2 ;                       _314<<=1;_314+=
         *_3_1415;_31                        =_314159+_314;
         if(!(*_31+1)                        )* _31 =_314 /
         __31415,_314                        [_3141]=_314 %
         __31415 ;* (                        _3__1415=_3_141
        )+= *_3_1415                          = *_31;while(*
        _3__1415 >=                           31415/3141 ) *
        _3__1415+= -                          10,(*--_3__1415
       )++;_314=_314                          [_3141]; if ( !
       _3_14159 && *                          _3_1415)_3_14159
       =1,__3_1415 =                          3141-_31415;}if(
       _314+(__31415                           >>1)>=__31415 )
       while ( ++ *                            _3_141==3141/314
      )*_3_141--=0                             ;}while(_3_14159
      ) ; { char *                             __3_14= "3.1415";
      write((3,1),                             (--*__3_14,__3_14
      ),(_3_14159                               ++,++_3_14159))+
     3.1415926; }                               for ( _31415 = 1;
    _31415<3141-                                1;_31415++)write(
   31415% 314-(                                 3,14),_3141592654[
  _31415    ] +                                "0123456789","314"
  [ 3]+1)-_314;                                 puts((*_3141592654=0
,_3141592654))                                   ;_314= *"3.141592";}
```

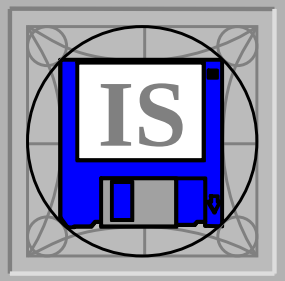


Notazione ungherese ☺

- `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



Buone prassi

- Progettare tracciando scelte a requisiti e valutando qualità
- Produrre codice secondo regole e valutando qualità
- Verificare il codice a partire dalle unità più piccole

- Versionare per tener traccia della “storia” di ciascun CI
- Configurare e integrare sistematicamente (*build*)
- Pianificare, assegnare e gestire compiti

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