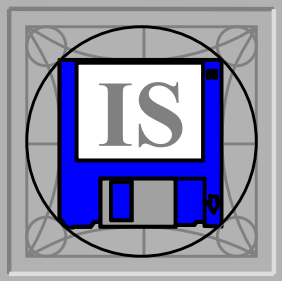
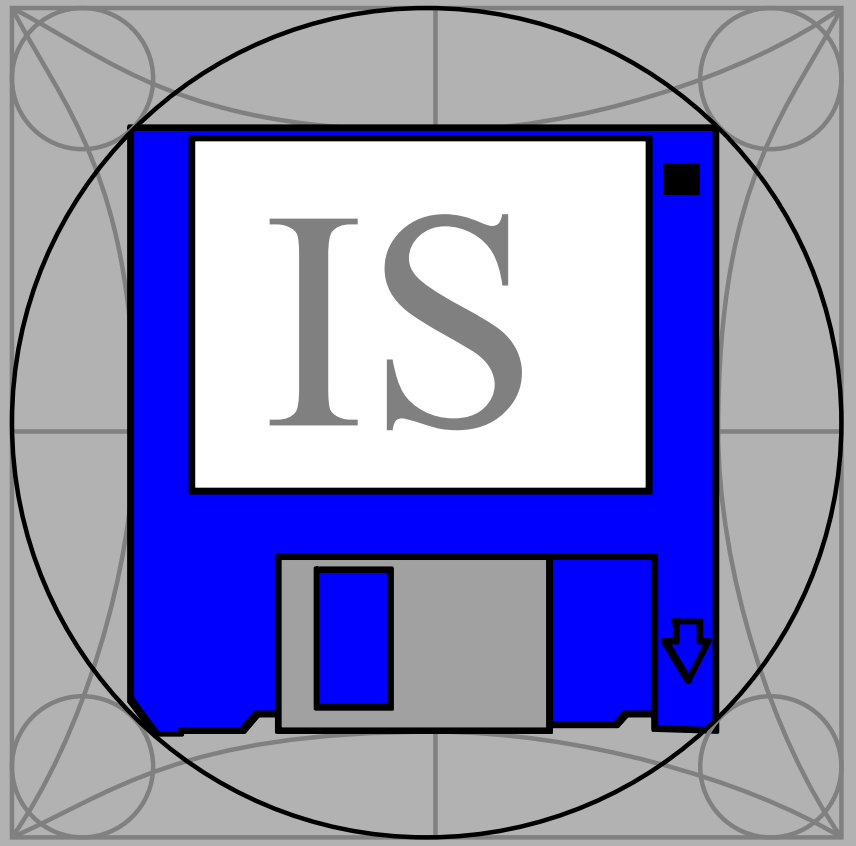


Amministrazione di progetto

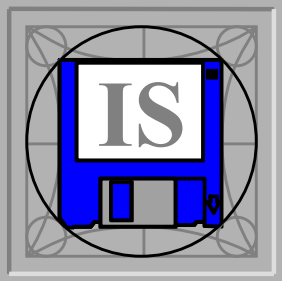
Spunti per *Flipped Classroom*



Ingegneria del Software

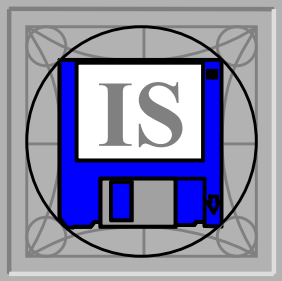
V. Ambriola, G.A. Cignoni,
C. Montangero, L. Semini

Aggiornamenti: T. Vardanega (UniPD)



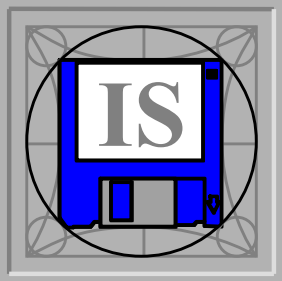
Compiti dell'amministratore

- Equipaggiare, organizzare, e tenere corrente l'ambiente di lavoro di progetto
  - Regole, procedure, strumenti
  - A supporto del *way of working* adottato
- Per attuare scelte procedurali e tecnologiche concordate
  - A servizio, non alla guida



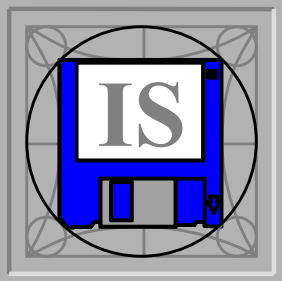
Normare il *way of working*

- Linee guida per le tutte le attività di progetto
  - Spiegano gli obiettivi dei processi (di sviluppo) adottati
  - Ne guidano l'attuazione sistematica e disciplinata
- Le norme sono organizzate gerarchicamente per processi, attività, procedure, e strumenti a supporto
  - Quali attività svolgere per ogni processo
    - Quando, con quali *input*, con quali *output*
  - Quali procedure eseguire per ogni attività
    - Passo-passo, così da non lasciare spazio all'arbitrio
  - Quali strumenti usare per farlo, e come



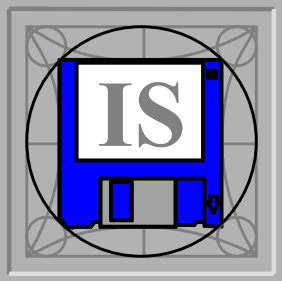
Ambiente di lavoro

- Fornisce tutto ciò che serve ai processi di produzione
  - Primari (sviluppo, fornitura),
  - Di supporto (documentazione, verifica, …)
  - Organizzativi (formazione, …)
- L'ambiente è fatto da persone, ruoli, procedure, strumenti
- La qualità dell'ambiente di lavoro determina la produttività
- Deve essere completo, ordinato, aggiornato
  - Tutto il necessario per svolgere le attività previste
  - È facile trovarvi ciò che si cerca
  - Il materiale divenuto obsoleto non deve causare intralcio



Supporto a gestione di progetto

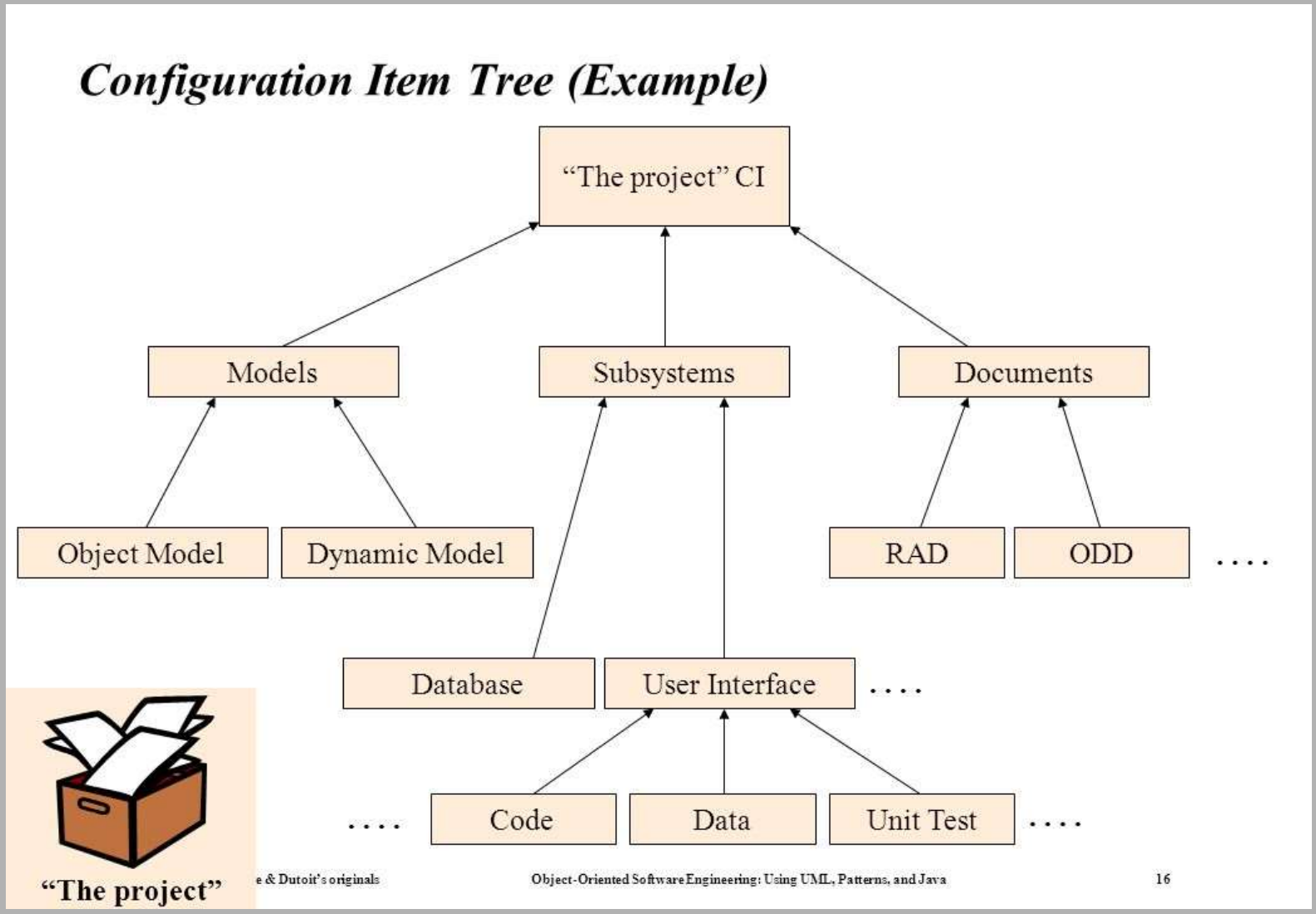
- **Pianificazione, stima e controllo dei costi**
  - **Allocazione e gestione della risorsa «tempo-persona»**
    - P.es, **InstaGantt** (https://instagantt.com/)
- **Strumenti collaborativi di controllo gestionale, qualità, coordinamento attività**
  - **Come quelli di *issue tracking* / *ticketing***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

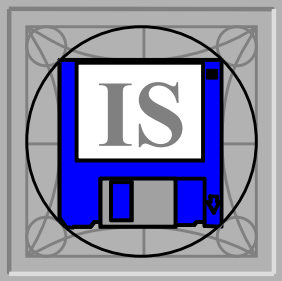


Configurazione – 1

- Un progetto (*project*) ha sempre un singolo prodotto in uscita, che è un aggregato composito di parti distinte
  - Specifiche (analisi, *design*), sorgenti, direttive, *test*, manuali
- Chiamiamo *build* l'aggregazione di parti che compongono un singolo sotto-prodotto (eseguibile o documento)
  - Nell'integrazione continua, ogni *build* costituisce una *baseline*
  - Ogni configurazione di *baseline* va messa in sicurezza
- L'insieme delle parti di una *build*, ordinato secondo regole precise, è detto configurazione
  - Le regole di configurazione, *Configuration Management*, sono parte del *way of working*
  - La loro attuazione va automatizzata

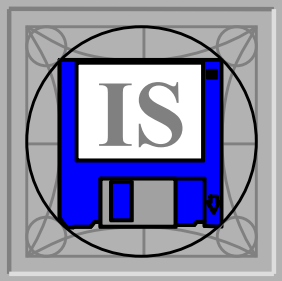
Configurazione – 2





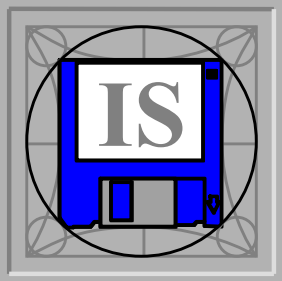
Attività di configurazione – 1

- **Identificazione di configurazione [1]**
  - **Le parti (*configuration item*, CI) che compongono lo specifico (sotto-)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
  - **Un progetto produce una successione di *baseline***
    - Che vanno gestite con processi e strumenti specializzati



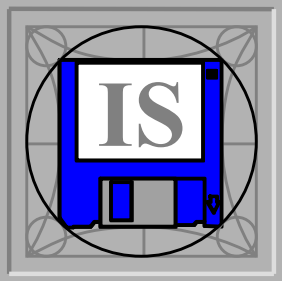
Attività di configurazione – 2

- Una *milestone* è una data di calendario che denota un punto di avanzamento atteso
  - Sostanziato da una o più *baseline*
- L'esistenza di solide *baseline* garantisce
  - Riproducibilità, ripristino
  - Tracciabilità (dove siamo rispetto agli obiettivi)
  - Analisi, valutazione, confronto (rispetto alle attese)



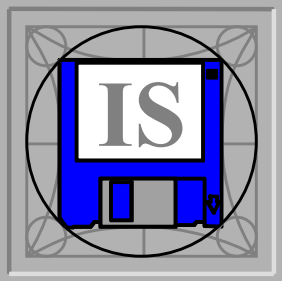
Buone qualità di *milestone*

1. Specifiche per obiettivi di avanzamento, e dimostrabile per risultati attesi
   - In una successione naturalmente incrementale
2. Coerenti con la strategia di progetto
   - Significative per il *team* e per gli *stakeholder*
3. Coerenti con le esigenze di calendario e tempestive
4. Delimitate per ampiezza e ambizioni
   - Realisticamente raggiungibili
5. Misurabili per quantità di impegno necessario
6. Traducibili in compiti assegnabili a singoli individui
   - Corrispondenti a uno *sprint* di metodo agile



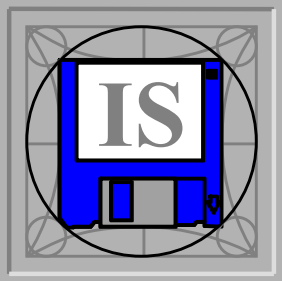
Attività di configurazione – 3

- **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*



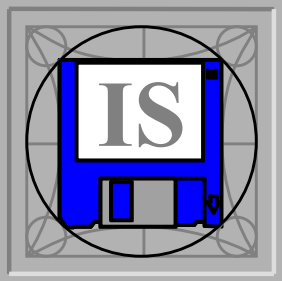
Attività di configurazione – 4

- **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**
    - Quale è il problema, cosa va fatto e come, come verificarlo
  - **Questo intero processo è detto *change management***
  - **Ogni decisione presa in esso va documentata in uno specifico verbale**
    - Ogni azione richiesta è accompagnata da una verifica di esito



Gestione delle modifiche

- Ogni richiesta di modifica va gestita in modo sistematico e disciplinato
  - Tramite procedura di *change request*
  - Che identifica autore, motivo, urgenza
  - Con stima di fattibilità (cosa fare, come, e come verificarne l'esito)
  - Con costo atteso e valutazione di impatto
  - Con decisione di approvazione del responsabile
  - Documentato in verbale dedicato
- Di ogni richiesta di modifica bisogna tenere traccia
  - Tramite *issue tracking* / *ticketing*
  - Tracciandone lo stato di avanzamento fino a chiusura



Riferimenti

- V. Ambriola, G.A. Cignoni, “Laboratorio di progettazione”, Jackson Libri, 1996
- 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