Amministrazione di progetto

Flipped classroom

Ingegneria del Software

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

Aggiornamenti: T. Vardanega (UniPD)

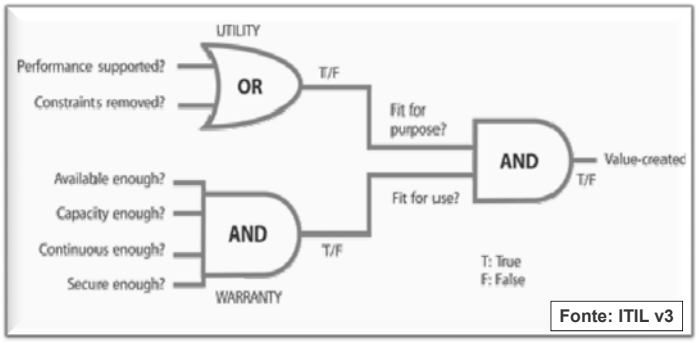
Amministrare un progetto

- Equipaggiare, organizzare e gestire l'ambiente di lavoro e di produzione
  - Tramite regole, procedure, strumenti erogati in forma di servizi
- A supporto dei processi istanziati nel progetto
- L'amministratore non compie scelte gestionali ma attua le scelte tecnologiche concordate con i responsabili aziendali e di progetto

Glossario – 7

- Servizio
  - Mezzo per aiutare l'utente a raggiungere i propri obiettivi riducendo costi e rischi
- Esempio di obiettivo
  - Massima efficacia di prodotto e massima efficienza di lavoro
- Esempio di valore fornito dal servizio a quel fine
  - Abbattimento delle attività improduttive
- Esempio di costi e rischi
  - Ciascuno sceglie i propri strumenti e le proprie regole

Il valore del servizio



Attività di amministrazione

- Redazione e manutenzione di regole e procedure di lavoro (le norme)
  - L' approvazione delle norme spetta al responsabile di progetto
- Reperimento, organizzazione, gestione e manutenzione delle risorse informatiche per l'erogazione dei servizi di supporto
  - Ambiente, infrastruttura, strumenti, prodotti, documenti

Documentazione di progetto

- Tutto ciò che descrive gli ingressi e le uscite delle attività necessarie al progetto
  - Riguardo al prodotto
  - Riguardo al processo
- Documenti di sviluppo
  - Specifiche fornite dal cliente
  - Diagrammi di progettazione
  - Codice commentato
  - Piani di qualifica e risultati delle verifiche
  - Manuali di installazione e uso
- Documenti di gestione del progetto
  - Documenti contrattuali
  - Piani e consuntivi delle attività

Temi che tratteremo nella *flipped classroom* del 10/11/2017

Disponibilità dei documenti

- Per essere utili, i documenti devono essere
  - Sempre raggiungibili
  - Chiaramente identificati (cosa sono, di cosa trattano)
  - Corretti nei contenuti
  - Verificati e approvati
  - Aggiornati, datati e versionati
- La loro diffusione deve essere controllata
  - I destinatari devono essere chiaramente identificati all'origine
    - Ogni documento ha una sua lista di distribuzione
  - L'amministratore gestisce le liste di distribuzione e attua procedure che ne assicurano il rispetto

Ambiente di lavoro

- Quanto serve ai processi di produzione
  - Di sviluppo, di supporto, di organizzazione
  - L'ambiente è fatto da persone, ruoli, procedure, infrastruttura
- La qualità dell'infrastruttura determina la produttività
  - Influendo sulla qualità del processo e del prodotto
- L'ambiente di lavoro 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 – 1

- **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/)
- **Strumenti collaborativi di controllo gestionale, qualità, coordinamento attività**
  - **Con *issue tracking* / *ticketing* tipici del *Service mgmt***
    - **Assembla** (http://www.assembla.com)
    - **Maven** (http://maven.apache.org)
    - **Jira** (http://www.atlassian.com/software/jira)

Supporto a gestione di progetto – 2

- **Gestione documentale**
  - **Condivisione (lettura), collaborazione (scrittura), archiviazione, controllo e supervisione**
    - TWiki (http://www.twiki.org)
    - Google Docs (http://docs.google.com)
    - …
- **Versionamento e configurazione**
  - **Ne riparliamo tra poco ...**

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/

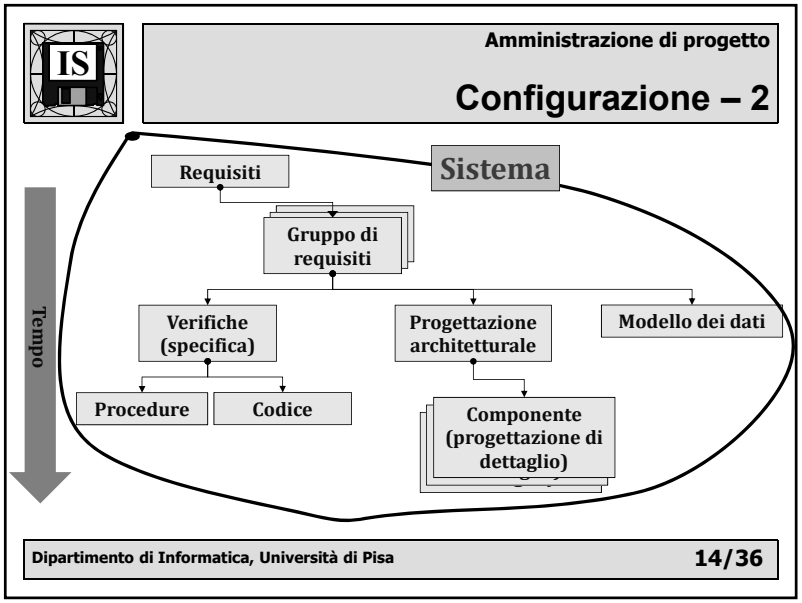
Supporto a sviluppo – 2

- **Codifica e integrazione**
  - **Ambienti integrati di sviluppo, IDE (p.es., http://www.eclipse.org/)**
  - **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**

Ne parleremo in un seminario dedicato

Configurazione – 1

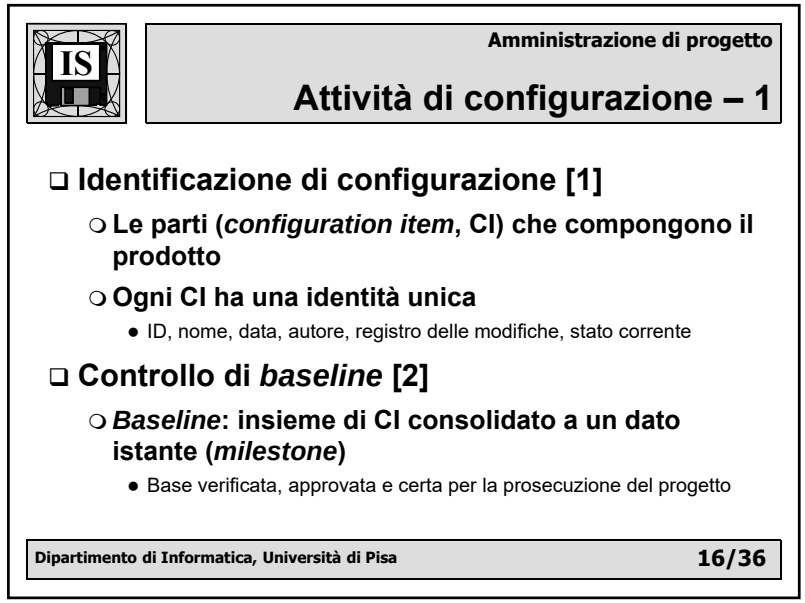
- Un prodotto SW è l'<u>unione ordinata</u> di parti distinte unite insieme secondo regole rigorose
  - Specifiche, progetti, programmi, verifiche, manuali
  - Le regole di configurazione vanno pianificate
  - Le responsabilità di configurazione vanno assegnate
- La gestione di configurazione va automatizzata
  - *Configuration Management*
  - *Build*



Configurazione – 2

Supporto a configurazione

- Obiettivi
  - Mettere in sicurezza le *baseline*
    - Istanze di configurazione che consolidano lo stato di avanzamento del prodotto
  - Prevenire sovrascritture accidentali
  - Consentire ritorno a configurazioni precedenti
  - Permettere recupero da perdite accidentali
- Attività
  1. Identificazione di configurazione
  2. Controllo di *baseline*
  3. Controllo di versione
  4. Gestione delle modifiche



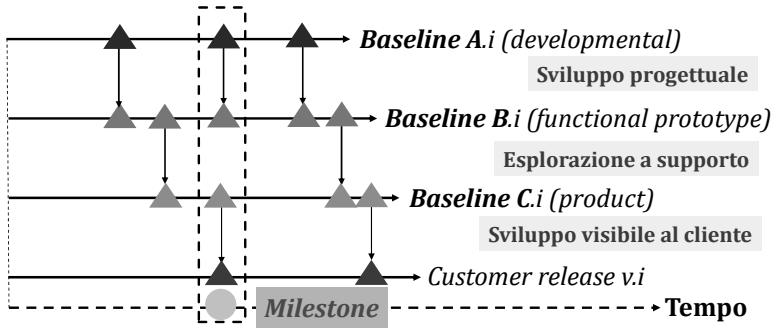
Attività di configurazione – 1

- 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 associata a uno specifico insieme di *baseline* (≥ 1)**
  - **L'esistenza di *baseline* ben identificate garantisce**
    - Riproducibilità
    - Tracciabilità
    - Analisi, valutazione, confronto

Baseline e *milestone*



Buone qualità di *milestone*

1. **Specifiche per obiettivi**
2. **Delimitate per ampiezza e ambizioni**
3. **Incrementali per contenuti**
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. **Puntuali rispetto alle esigenze di calendario**
9. **Dimostrabili agli *stakeholder***

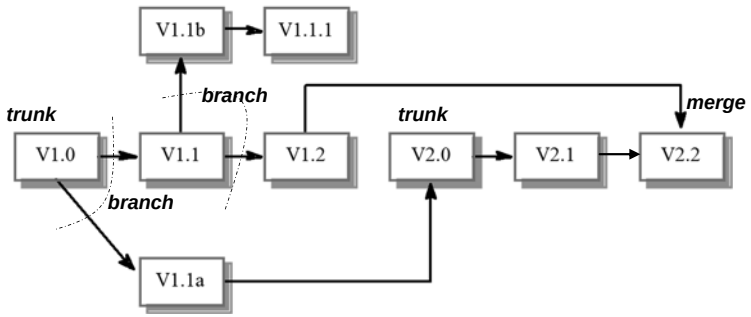
Attività di configurazione – 3

- **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 funzionalmente distinta dalle altre
- **Variante**
  - Istanza di CI 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

Esempio di storia di versioni



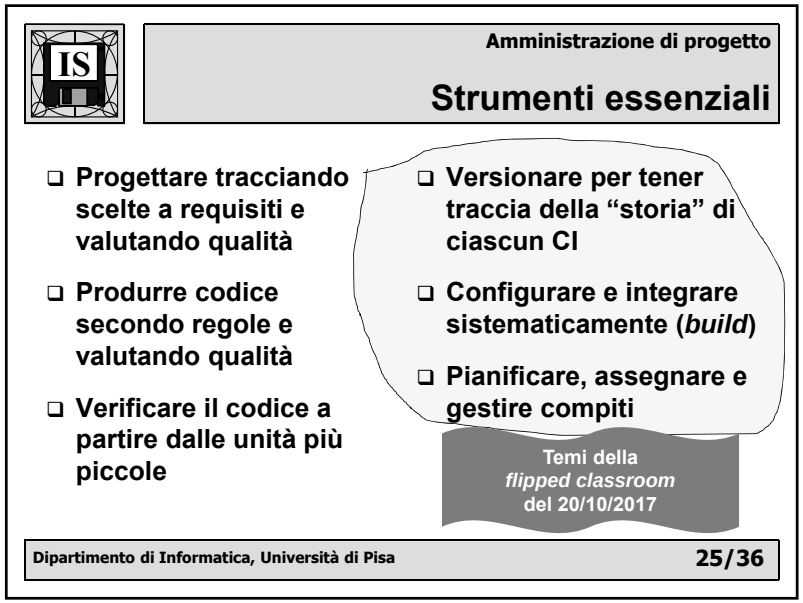
Tratto da: Ian Sommerville, *Software Engineering*, 8th ed.

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**

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



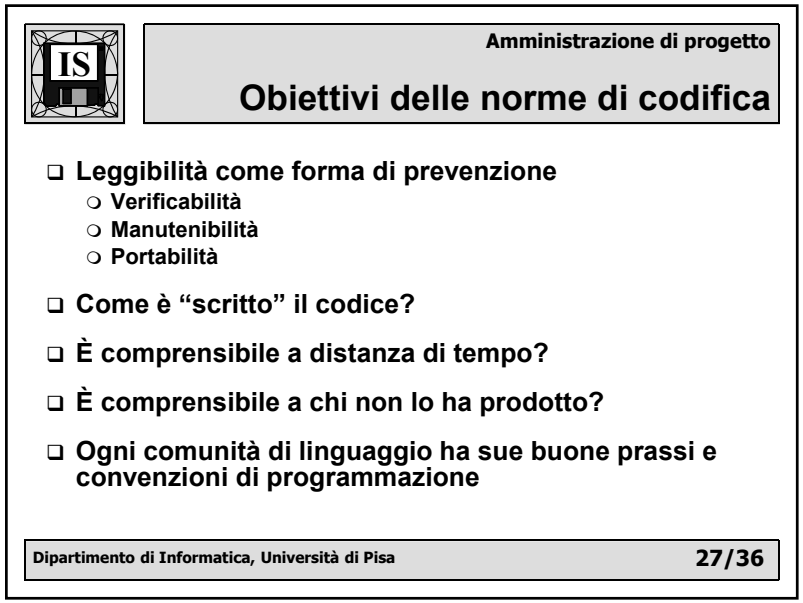
Strumenti essenziali

- 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

Temi della *flipped classroom* del 20/10/2017

Norme di progetto

- 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
    - Processi primari (fornitura, sviluppo) → rapporti con il committente, analisi, progettazione, codifica, integrazione
    - Processi di supporto (documentazione, verifica, validazione, configurazione, gestione modifiche,…)
    - Processi organizzativi (gestione di progetto, formazione, comunicazione)
  - Specificano convenzioni sull'uso degli strumenti scelti
  - Organizzano la comunicazione e la cooperazione



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

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

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

USE THE CODE, LUKE!

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