metodologie

a.a. 2003-2004

metodologia

- una serie di linee guida per raggiungere certi obiettivi

  più formalmente:

  – un processo da seguire
  – documenti o altri elaborati da produrre usando linguaggi più o meno standard

- i nostri obiettivi sono relativi al successo di progetti per lo sviluppo di software

  esempi

  – realizzare software corrispondente alle esigenze dei clienti
  – rispettare i tempi programmati
  – portare a compimento il progetto con successo
  – ecc.

discipline

(disciplines)

(terminologia “unified process”)

- requirements
- business modeling
- design
- implementation
- test
- deployment

- configuration & change management
- project management
- environment

artifacts

(elaborati)

- qualsiasi prodotto del lavoro del team
- esempi
  - codice
  - grafica web
  - documenti di testo
  - diagrammi
  - modelli
  - glossari
  - ecc.
  - anche configurazioni di sistema e installazioni di sw

artifacts

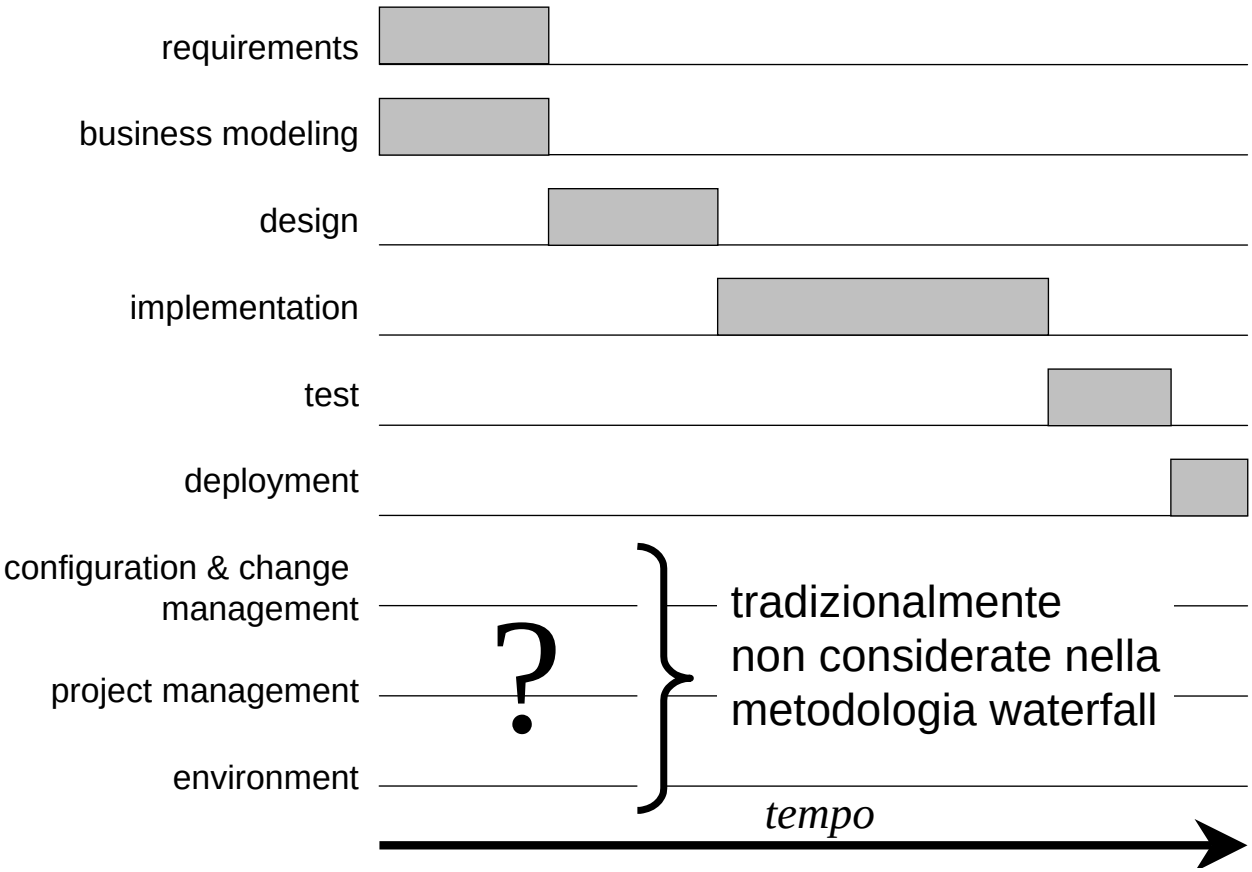
- ciascun artifact è relativo ad una disciplina
- ad esempio:
  - requirements: use cases, glossario, ecc.
  - business modeling: modelli espressi ad esempio da diagrammi ER
  - design: class diagram
  - implementation: codice
  - deployment: sistema up and running
  - environment: configurazione di ambiente per gli sviluppatori
  - ecc.

metodologia waterfall

(cascata)

- analisi dei requisiti
  - si fa di tutto perché i requisiti non cambino nelle successive fasi
- progetto
  - si fa di tutto perché il progetto non cambi nelle successive fasi
- realizzazione
- integrazione
- test

waterfall: impegno nel tempo (ideale)



waterfall: origini

- retaggio di altre aree dell'ingegneria
  - es. ingegneria civile (palazzi, ponti, ecc)
- molto utile quando
  - la realizzazione è molto costosa
    - es. un ponte
  - gli accordi contrattuali sono chiari e non sucettibili di cambiamento
    - es. capitolato tecnico di gara di appalto

waterfall nello sviluppo del software

- il committente non ha chiaro il prodotto che vuole ottenere
  - è spesso compito del team di sviluppo aiutare il committente a chiarirsi le idee
  - i requisiti **sicuramente** cambieranno durante il lavoro
    - a causa di cambiamenti di direttive manageriali
    - a causa di una migliore conoscenza del problema
    - ecc.
- per progetti lunghi cambiano anche...
  - il mercato, la tecnologia, gli utenti, il management, ecc.
- nessuna gestione dei “rischi”...

rischi

(risks)

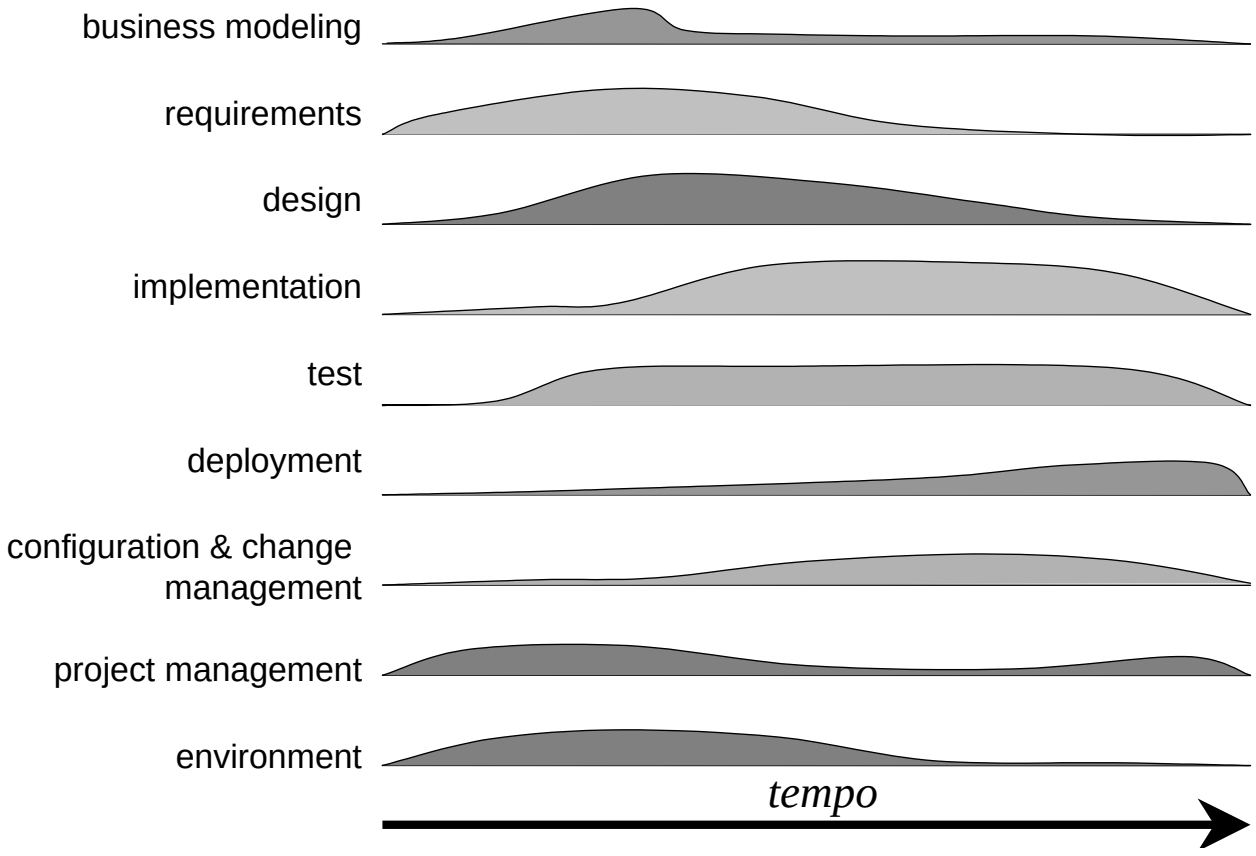
- un rischio è la possibilità di subire un danno
- ad esempio è difficile dire se...
  - i requisiti sono corretti e completi
    - ... e il committente non cambi idea in corso d’opera
  - il modello di dominio è corretto e completo
  - il progetto sia corretto e completo
  - le tecnologie a disposizione siano abbastanza flessibili per realizzare l’architettura progettata
  - le risorse a disposizione saranno sufficienti per portare a termine il progetto
    - in termini di denaro, persone, competenze, attrezzature, ecc.
  - il management manterrà le attuali direttive
    - es. partnership con altre industrie, prodotti da utilizzare, ecc.

unified process

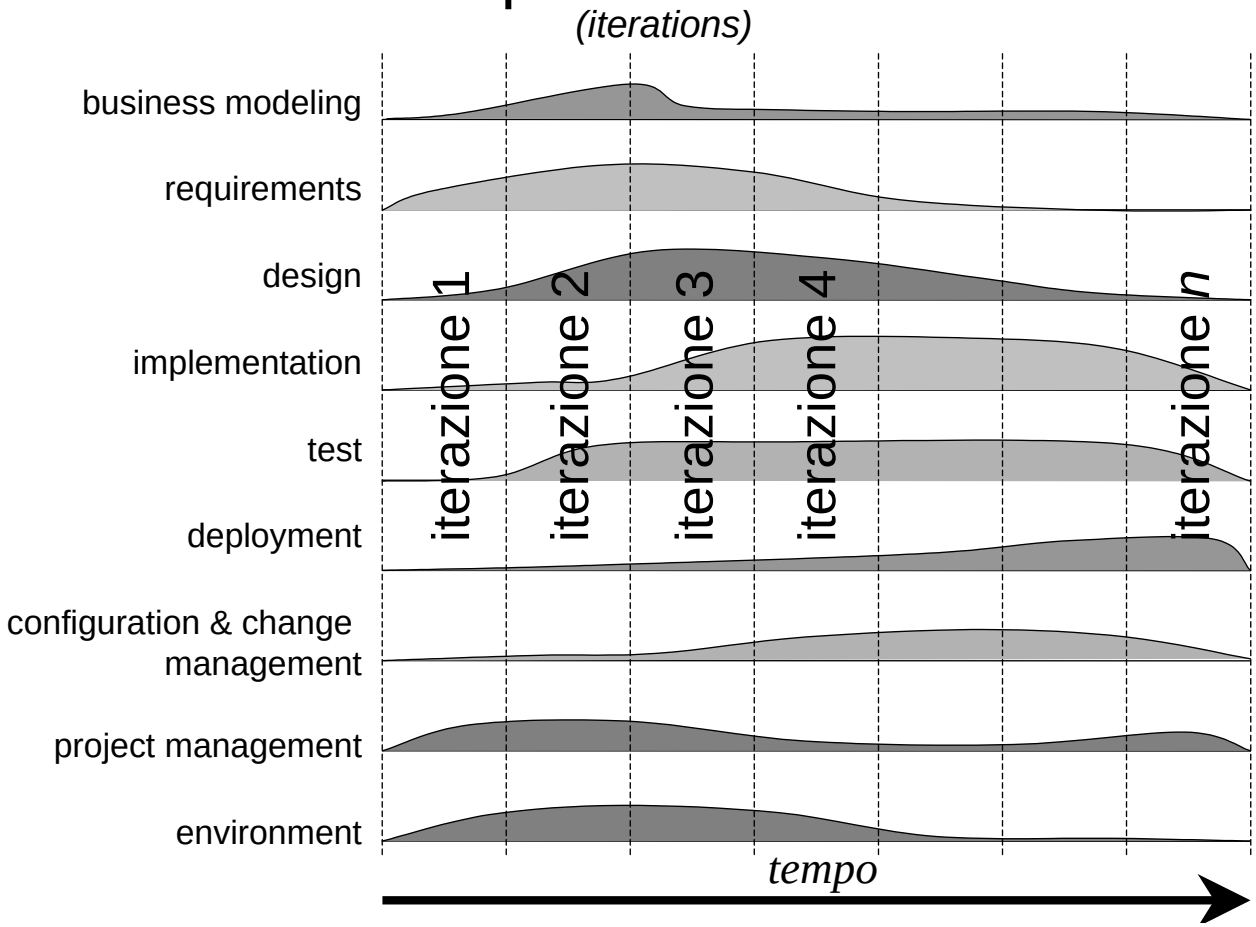
- risk driven
  - tutte le decisioni mirano a “risolvere” i rischi il prima possibile
- iterative development
  - obiettivi piccoli e scadenze chiare e ravvicinate
  - spesso l’iterazione prevede un rilascio del software
- light (o agile)
  - tutti gli elaborati sono opzionali
  - analisi critica della utilità degli elaborati
- adaptive (embrace changes)
  - requisiti e progetto continuamente raffinati

unified process: impegno nel tempo

(questo è solo un esempio, le distribuzioni variano a seconda del progetto)



unified process: iterazioni



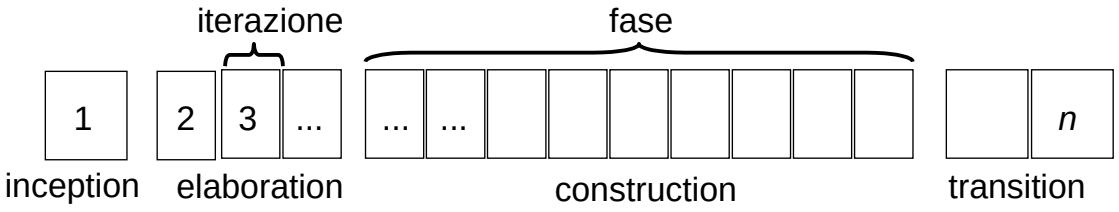
unified process: iterazioni

- ciascuna iterazione ha degli obiettivi e una scadenza
  - la scadenza è inderogabile (timeboxed iterations)
  - è possibile ridimensionare gli obiettivi
- le itearzioni sono brevi
  - es. 2 settimane
- alla fine di ciascuna iterazione
  - si fa il punto della situazione
  - si programma...
    - la durata dell'iterazione successiva
    - i task dell'iterazione successiva

unified process: fasi

(phases)

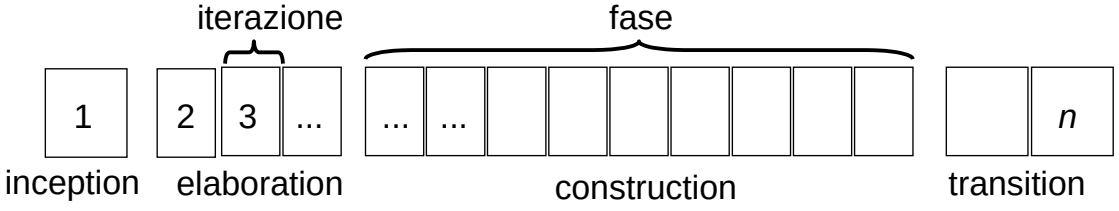
- inception *(ideazione o avviamento)*
  - obiettivi:
    - capire se il progetto è realizzabile con i vincoli di budget e tempo e se esistono soluzioni alternative (“make or buy”)
    - identificare i rischi
- elaboration
  - obiettivi: ridurre i rischi di insuccesso al minimo possibile
  - in concreto
    - requisiti
    - analisi
    - inizio progetto
    - principali aspetti architetturali



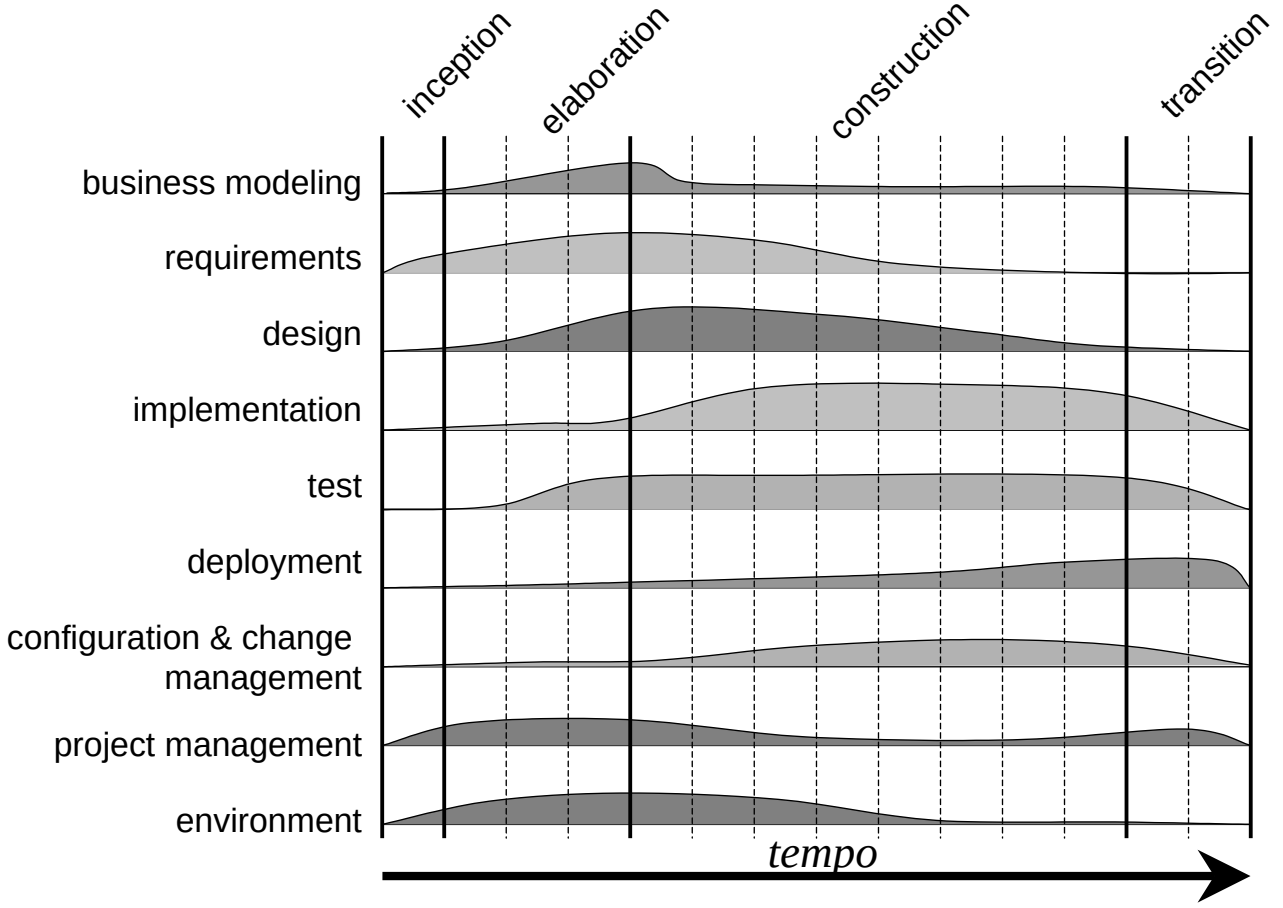
unified process: fasi

(phases)

- construction
  - obiettivi: realizzazione
  - in concreto:
    - stabilizzazione del progetto
    - stabilizzazione del architettura
    - implementazione
    - testing
- transition
  - obiettivi: messa in produzione
  - le attività dipendono dal progetto
    - es. testing, training del personale, migrazione dati da vecchi sistemi, ecc.



unified process



unified process: development case

- è un documento che mostra tutti gli elaborati e la loro evoluzione nel tempo

(questo è solo un esempio, gli elaborati variano a seconda del progetto)

Disciplina	Elaborato	Inc.	Elab.	Constr.	Trans.
business modeling	Domain Model		s	r	
Requirements	Use Cases	s	r		
	Vision	s	r		
	Supplementary Specification	s	r		
	glossary	s	r		
Design	Design Model		s	r	
	SW Architecture doc.		s	r	
	Data Model		s	r	
Implementation	Implementation Model (the code)		s	r	r
Project Management	phase plan (gantt)	s	r	r	r
	risks ranking	s	r		
	requirements ranking	s	r	r	
	iteration plan	for each iteration			
Testing	Test Model		s	r	
	Test code		s	r	r
Environment	Development Case	s	r		

s: start, r: refine

eXtreme Programming (XP)

- metodologia incentrata sullo *sviluppo del codice*
- all'estremo...
  - documenti da produrre: nessuno o pochissimi
  - il codice deve essere scritto (e spesso riscritto) in modo da esprimere
    - i concetti individuati nella fase di analisi
    - le scelte progettuali
- on-line:
  - http://www.extremeprogramming.org/
  - http://www.xprogramming.com/

eXtreme Programming (XP)

- estremamente sensibile agli "aspetti sociali"
  - considera: comunicazione, lavoro di gruppo, soddisfazione personale, relazioni interpersonali, qualità del lavoro, ambiente di lavoro
- risk driven
  - vedi ciò che abbiamo detto di UP
- agile
  - nessun documento da produrre!
- i dogmi
  - 4 valori (*values*)
  - 5 principi (*principles*)
  - 12 pratiche (*practices*)

XP: i valori

- comunicazione
  - discussioni libere tra tutte le persone coinvolte (team di sviluppo, manager, utenti, clienti, ecc.)
- semplicità
  - tutto ciò che non serve *ora* non si fa
- feedback
  - per tutto ciò che si fa ci deve essere un feedback di correttezza e qualità
- coraggio
  - ...di fare la cosa che è chiaramente migliore anche se richiede considerevole fatica

XP: i principi

- feedback rapido (*provide rapid feedback*)
  - se il feedback non è rapido non serve
- soluzioni semplici (*assume simplicity*)
  - inutile perdere tempo per cose che forse non servono
    - es. niente architetture complesse, flessibili, estendibili
- cambiamenti incrementali (*make incremental changes*)
  - niente stravolgimenti nell'architettura e nel codice
  - grandi cambiamenti si fanno a piccoli passi (refactoring)
- adottare il cambiamento (*embrace change*)
  - XP rende i cambiamenti poco onerosi e poco rischiosi quindi non si deve avere paura di cambiare l'architettura e il codice
- lavoro di qualità (*do quality work*)
  - test e molto altro
  - vedi R. M. Pirsig - "Lo Zen e l'arte della manutenzione della motocicletta"

XP: pratiche

all'interno di questo corso:
[applicabile]
[difficilmente applicabile]
[inapplicabile]

- planning game [ok (come in UP)]
  - per ciascuna iterazione (breve) si pianificano i task
- small releases [ok]
  - ciascuna iterazione da luogo ad un rilascio
- simple design [ok]
  - architettura senza inutili fronzoli
  - es. nessun supporto per il riuso del software se non è richiesto
- test-first programming [ok]
  - procedure di testing automatico (parziale e totale)
  - prima si scrive il codice di test e poi si sviluppa
  - per togliere un bug: prima si scrive il test che mostra il bug e poi si fa debugging
- continuous integration [difficile]
  - http://cruisecontrol.sourceforge.net

XP: pratiche

- refactoring [ok]
  - metodologia per cambiare il codice in maniera incrementale
- pair programming (o meglio pair working) [ok]
  - sempre: requisiti, analisi, progetto, sviluppo, debugging, test
- collective ownership [ok]
  - tutti gli sviluppatori possono modificare qualsiasi parte del codice
- 40-hour week [no]
- on-site customer [difficile]
  - l'utente/cliente deve essere consultato di frequente
- methaphor [ok]
  - sviluppare un linguaggio comune con gli utenti/clienti
- coding standard [ok]
  - poiché il codice deve poter essere modificato da tutti deve essere comprensibile a tutti

UP vs. XP

- i documenti in UP servono
  - a formare un “sapere comune”
    - tutti sanno le stesse cose come se il gruppo fosse una sola persona
  - a far ragionare il gruppo in maniera coerente
- in XP il “sapere comune” è formato mediante la comunicazione e il lavoro cooperativo
  - richiede molto tempo al team
    - il “sapere comune” viene aggiornato giornalmente mediante il lavoro cooperativo
  - pensata per gruppi che lavorano full time insieme

perché questo corso non è basato su XP?

- XP richiederebbe, per funzionare, un impegno intenso
  - anche se forse per un tempo più breve
- UP è più flessibile dal punto di vista della programmazione del tempo
  - il vostro impegno è stimato in 75 ore persona diluite in due mesi (studio individuale relativo a 5 crediti formativi)
- in UP è più facile valutare la “cultura di gruppo” poiché ne rimane traccia non solo nel codice