

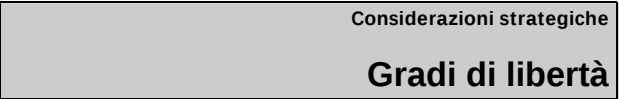


Considerazioni strategiche



Anno accademico 2011/12
Ingegneria del Software mod. A

Tullio Vardanega, tullio.vardanega@math.unipd.it



- ❑ Il segmento di ciclo di vita attivato nel progetto didattico è compreso tra RR e RA
 - Revisione dei requisiti (inizio)
 - Revisione di accettazione (collaudo, fine)
- ❑ Il modello di ciclo di vita interno a esso
 - È scelta autonoma del fornitore
 - Non è specificato nel capitolato di appalto
 - Determina piano e strategia di utilizzo delle risorse disponibili
 - Persone, capacità, strumenti



- ❑ Compiti, risorse e tempo necessario
 - *The Mythical Man-Month*, Frederick P Brooks, Jr (1975)
 - Componenti di impegno non riducibili
 - Compiti non partizionabili
 - Per necessità: esempio: un solo ambiente di prova
 - Per scelta: esempio: per preservare integrità concettuale
 - Compiti che richiedono comunicazione e interazione
 - Coordinarsi troppo frequentemente costa molto sforzo
 - Verifiche a livello sistema
 - Il sistema diventa disponibile solo a fine sviluppo



- ❑ Aggiungere risorse a progetto in corso
 - Aggiunge complessità
 - Nell'esecuzione di tutti i compiti che richiedono comunicazione e coordinamento
 - Aggiunge inefficienza
 - Nell'esecuzione di tutti i compiti non partizionabili
- ❑ Buona pianificazione, buona analisi e buona progettazione architeturale
 - Richiedono investimento a monte
 - Quando ancora c'è tempo
 - Risparmiano costo a valle
 - Quando le risorse cominciano a scarseggiare



Considerazioni strategiche

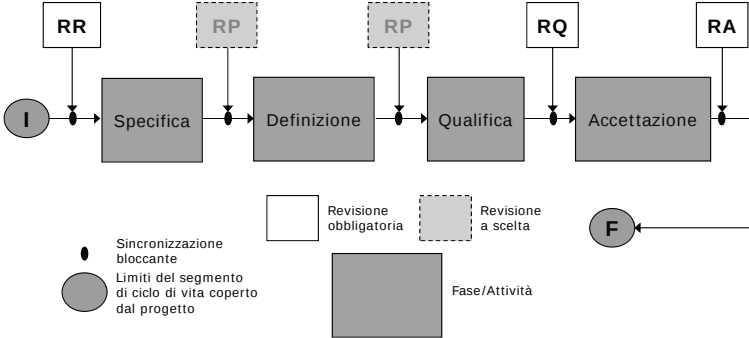
Modello di ciclo di vita interno – 1

- ❑ Non tutti i modelli si adattano allo stesso modo agli adempimenti formali richiesti dal progetto
- ❑ La scelta del modello di ciclo di vita interno è libera
 - Ogni scelta comporta oneri e benefici diversi
- ❑ La scelta è spesso per prodotto (per progetto)
 - Indipendente dall'organizzazione di appartenenza



Considerazioni strategiche

Modello sequenziale – 1



Considerazioni strategiche

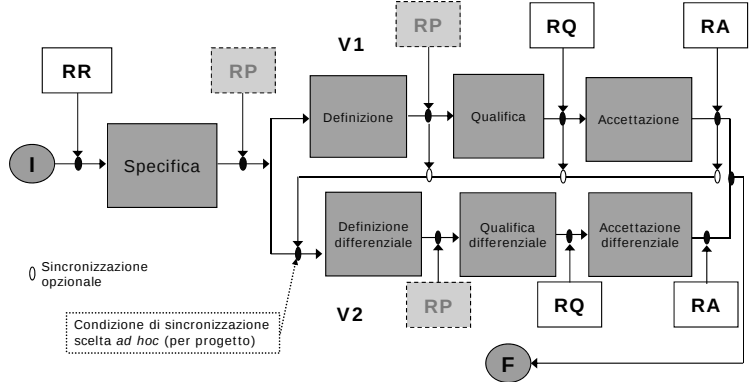
Modello sequenziale – 2

- ❑ Osservazioni
 - Il progetto didattico assume una singola occorrenza di ogni revisione prevista
 - Assunzione vincolante solo per il cliente – committente
 - Il fornitore può liberamente ritenere le revisioni solo come vincolo di calendario
 - Senza obbligo di aderire al modello sequenziale per il proprio sviluppo interno
 - A condizione di sviluppare in tempo tutti i prodotti richiesti in ingresso dalle revisioni sostenute



Considerazioni strategiche

Modello incrementale – 1





Considerazioni strategiche

Modello incrementale – 2

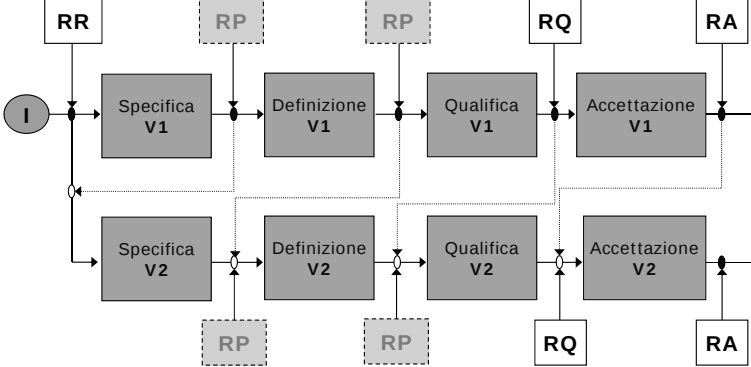
❑ Osservazioni

- Il progetto didattico assume una singola occorrenza di ogni revisione prevista e una sola accettazione
- Il modello incrementale può però necessitare di più istanze di una stessa revisione per incrementi successivi
- Il fornitore deve decidere quale istanza far corrispondere al calendario di revisione fissato dal cliente
 - Realizzando le altre come revisioni interne senza il coinvolgimento del cliente
 - Completando le iterazioni coerentemente con il calendario esterno
 - Le revisioni esterne non possono però essere differenziali!



Considerazioni strategiche

Modello evolutivo – 1



Considerazioni strategiche

Modello evolutivo – 2

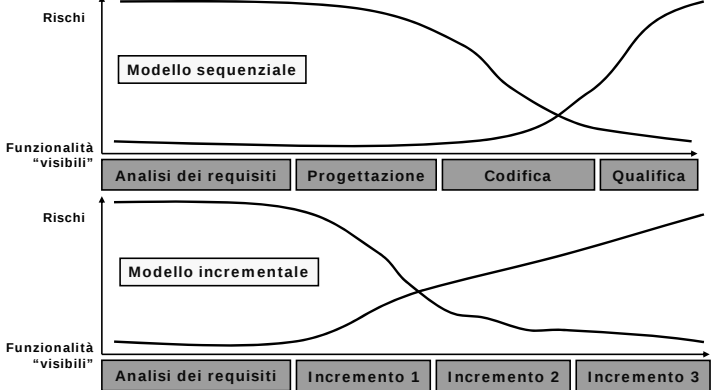
❑ Osservazioni

- Stessi vincoli del modello incrementale
 - Ma le versioni successive possono rilasciare prototipi esterni e anche essere iterative (= distruttive)
- La combinazione di sviluppi evolutivi con il calendario di revisioni imposti dal cliente comporta che
 - Revisioni diverse abbiano come oggetto versioni di prodotto diverse
 - Prodotti presentati in ingresso alla revisione devono essere resi coerenti con il fine della revisione e l'interesse strategico del fornitore



Considerazioni strategiche

Una visione critica





Considerazioni strategiche

Avvertenze

□ Limiti intrinseci della progettazione

- Non tutti i problemi hanno una soluzione
- Fissare con la massima chiarezza possibile
 - Obiettivi
 - Vincoli
 - Alternative
 - Rappresentazioni del problema e delle sue soluzioni

□ Qualità cardine della progettazione

- Fattibilità e verificabilità
 - Tecnica ed economica



Considerazioni strategiche

Tecniche progettuali – 1

□ Decomposizione modulare

- Una buona decomposizione architetutturale identifica componenti tra loro indipendenti
 - A basso o nullo accoppiamento
 - Autosufficienti (funzionalmente coesi)

□ Incapsulazione (*information hiding*)

- Nascondere il dettaglio realizzativo
 - Solo l'interfaccia è pubblica
 - Il dettaglio è noto solo all'interno



Considerazioni strategiche

Tecniche progettuali – 2

□ Controllo di accoppiamento e di coesione

□ L'accoppiamento è misura dell'intensità della relazione tra l'interno delle parti

- La modifica di una comporta la modifica dell'altra
- Forte accoppiamento • scarsa modularità

□ La coesione è misura dell'intensità della relazione all'interno di una singola parte

- Forte coesione • buona caratterizzazione



Considerazioni strategiche

Tecniche progettuali – 3

□ Astrazione

○ “Dimenticare” informazione per applicare operazioni uguali a entità diverse

- La radice di una gerarchia di classi astrae rispetto alle classi più specializzate
 - Ciò che è caratteristico dell'intera gerarchia è fissato in radice
 - Ciò che differenzia si aggiunge per specializzazione
- A ogni astrazione corrisponde una concretizzazione
 - Per parametrizzazione (esempio: *template* in C++)
 - Per specializzazione (esempio: *classe* in Java e C++)



Considerazioni strategiche

Tecniche progettuali – 4

❑ Sufficienza

- La definizione dell'astrazione è sufficiente a caratterizzare l'entità desiderata

❑ Completezza

- L'astrazione fornita esibisce tutte le caratteristiche di interesse del suo utente

❑ Atomicità

- L'utilità dell'astrazione non migliora se ulteriormente decomposta in astrazioni più elementari

Laurea in Informatica, Università di Padova

17 / 25



Considerazioni strategiche

Problematiche critiche – 1

❑ Concorrenza

- Se e come decomporre il sistema in entità attive concorrenti garantendo
 - Efficienza di esecuzione
 - Atomicità di azione
 - Consistenza e integrità di dati condivisi
 - Semantica precisa di comunicazione e sincronizzazione
 - Predicibilità di ordinamento temporale

❑ Distribuzione

- Se e come i componenti sono disseminati su più nodi di elaborazione e come comunicano fra loro

Laurea in Informatica, Università di Padova

18 / 25



Considerazioni strategiche

Problematiche critiche – 2

❑ Controllo e trattamento degli eventi e degli errori

- Relativi al flusso dei dati
 - La disponibilità di un dato (dall'interno o dall'esterno)
- Relativi al flusso di controllo
 - L'ingresso del sistema (o di una sua componente) in un particolare stato
- Relativi al trascorrere del tempo
 - L'attesa di un certo istante temporale

❑ Mai fare assunzioni ottimistiche!

Laurea in Informatica, Università di Padova

19 / 25



Considerazioni strategiche

Integrità concettuale – 1

❑ Facilmente riconoscibile in una architettura fisica (edificio, costruzione, ...)

- Suggerisce uno stile uniforme, coerentemente applicato a tutte le parti del sistema ed alle loro interazioni
- Bilancia capacità funzionale con semplicità d'uso

❑ Desiderabile in ogni architettura di sistema

Laurea in Informatica, Università di Padova

20 / 25



Considerazioni strategiche

Integrità concettuale – 2

❑ **Procede da una definizione unitaria**

- **Ma non unilaterale**
 - Perché passa al vaglio dei membri del progetto
- **Richiede osservanza ai costruttori**
 - E vigilanza all'architetto
- **Nozione aristocratica piuttosto che democratica**

❑ **È distinta dalla realizzazione concreta**

- **Consente più percorsi realizzativi**
- **Facilita parallelismo nella realizzazione**

Laurea in Informatica, Università di Padova

21/25



Considerazioni strategiche

Programmazione difensiva – 1

❑ **Un ottimo modo per minimizzare i difetti del codice è programmare esplicitamente il trattamento dei possibili errori**

- **Errori nei dati in ingresso**
 - Verificare la legalità dei dati prima di usarli
- **Errori logici**
 - Fissare e verificare pre- e post-condizioni

❑ **La strategia di trattamento (*error handling*) va prevista nella progettazione**

Laurea in Informatica, Università di Padova

23/25



Considerazioni strategiche

Enforce intentions

❑ **Precauzione da usare al confine tra progettazione e codifica**

❑ **Rendere chiaro e rigido il confine tra esterno e interno dei moduli**

- **Decidere chiaramente e codificare conseguentemente ciò che può essere specializzato**
 - Rendere il resto imm modificabile (*final*, *const*, ...)
- **Proteggere tutto ciò che non deve essere visto e acceduto dall'esterno**
 - *Private*, *protected*, ...
- **Decidere quali classi possono produrre istanze e quali no**
 - Usare il pattern *Singleton* per le classi a istanza singola

Laurea in Informatica, Università di Padova

22/25



Considerazioni strategiche

Programmazione difensiva – 2

❑ **Trattamento errori nei dati in ingresso**

- **Attendere fino all'arrivo di un valore legale**
- **Assegnare un valore predefinito (*default*)**
- **Usare un valore precedente**
- **Registrare l'errore in un *log* persistente**
- **Sollevare una eccezione**
- **Abbandonare il programma**

Laurea in Informatica, Università di Padova

24/25



Programmazione difensiva – 3

□ Rischi di errore logico

○ Aritmetica in virgola mobile

- Sono intrinsecamente imprecisi
- La loro approssimazione può cumularsi e condurre a errori importanti e a confronti svianti

○ Puntatori e limiti di strutture

- L'accesso erraneo e non controllato a zone di memoria può corrompere gravemente i dati
- L'uso di *aliasing* rende i programmi difficili da comprendere e modificare

○ Allocazione dinamica della memoria

- Può portare a esaurimento della disponibilità e alla sovrapposizione di aree sensibili

○ Ricorsione

- Può portare a esaurimento della memoria oppure alla non terminazione

○ Concorrenza

- Se mal progettata può condurre a situazioni di malfunzionamento difficili da rilevare