Metodologie di progettazione del software

Indice

5. Metodologie di progettazione del software

> *This is the age of Methods; and the university which is to be the exponent of the living condition of the human mind must be the university of methods. [...] Unfortunately practice generally precedes theory, and it is the usual fate of mankind to get things done in some boggling way first, and find out afterwards how they could have done them much more easily and perfectly.*
> Charles S. Peirce , 1882 [1]

Già nella prima lezione si è accennato a diverse metodologie di progettazione, o modelli del processo o ciclo di vita, del software, per introdurre alcune ragioni profonde di analogia fra progettazione di software e progettazione di siti Web. Quei cenni introduttivi vengono approfonditi in questa lezione, con il seguente

Scopo della lezione

acquisire una visione panoramica delle principali metodologie di progettazione del software e delle ragioni che fanno preferire ciascuna di esse alle altre in determinati contesti produttivi e per desiderate caratteristiche di qualità del prodotto nonché del processo produttivo stesso.

Resta intesa l'adozione di un punto di vista in cui si considerano siti Web quali prodotti di nostro interesse specifico, sebbene la panoramica a cui stiamo per accostarci sia in prevalenza risultato di indagini ed esperienze nel campo della progettazione del software. Ciò è giustificato dall'analogia richiamata sopra.
In questa lezione:

- **in Sez. 5.1: il problema del progetto di qualità**
  viene proposto quale punto di partenza del processo produttivo e motivazione per l'adozione di un modello dello stesso, cioè di una definizione del suo ciclo di vita;
- nella fase iniziale di progettazione, la scelta di una metodologia di progetto coincide con l'adozione di un modello di processo o ciclo di vita: viene presentata
  in Sez. 5.2: una panoramica dei modelli di processo più affermati
- infine, come è consuetudine dalle lezioni precedenti, problemi e proposte di lavoro vengono avanzati in Sez. 5.3, e viene formulata
  in Sez. 5.3: una ipotesi di lavoro
  che verrà ripresa in lezioni successive.

5.1 Il problema del progetto di qualità

Il problema può essere enunciato come segue:

Progettare il processo produttivo di un prodotto che abbia desiderate caratteristiche di qualità.

5.1.1 Motivazioni del problema

Il problema trae origine dai seguenti fatti.

- **Non esiste un tipo universale di processo produttivo adatto a tutte le situazioni.**
- **Caratteristiche di qualità del prodotto sono conseguibili solo se il processo produttivo ha caratteristiche di qualità ad esse corrispondenti.**
  Non si possono "aggiungere" qualità al prodotto o migliorarne valori di qualità *dopo* il processo produttivo ma solo *attraverso* un processo produttivo.
- **Mediazione fra obiettivi di qualità del prodotto e vincoli posti dal contesto produttivo.**
  Nella determinazione delle caratteristiche del processo produttivo occorre tener conto sia degli obiettivi di qualità del prodotto sia dei posti dal contesto produttivo. La progettazione del processo produttivo richiede dunque una fase preliminare, che spesso presenta un carattere *negoziale* o *ipotetico* , in cui esplorare e valutare la *fattibilità* di alternative diverse.

5.1.2 Modelli di soluzione

Alla soluzione del problema può essere inizialmente utile la scelta di un *modello del processo* produttivo, altrimenti detto *ciclo di vita* , fra quelli che risultano praticabili, cioè di un'astrazione che raccoglie essenziali caratteristiche comuni di una grande varietà di processi produttivi. Quali sono queste caratteristiche essenziali? Esiste una notevole varietà di punti di vista in proposito, che conduce a diverse caratterizzazioni del

concetto di ciclo di vita:

- **modello di riferimento per il processo**
- **schema di organizzazione delle attività di base (fasi) dello sviluppo produttivo**
- **"filosofia" o modo di vedere e costruire il processo di sviluppo**

In tutte queste definizioni c'è del vero, tuttavia sembra utile cercare di catturare in un solo termine i diversi aspetti che esse mettono in luce. Ora, sembra empiricamente evidente, oltre che coerente con le definizioni richiamate sopra, che i
tratti essenziali del processo di sviluppo
rilevanti al livello di astrazione della sua caratterizzazione iniziale risultano essere:

- **suddivisione in parti (fasi, attività, etc.)**
  utile alla divisione ed organizzazione del lavoro;
- **determinazione di relazioni fra dette parti**
  ad esempio, di precedenza temporale, di causalità, etc.;
- **definizione di modi di interazione fra le parti stesse**
  ad esempio, identificazione di flussi di prodotti parziali, o *semilavorati* , fra le parti, determinazione di punti di sincronizzazione fra parti interdipendenti, etc.

Rivisitando il concetto di *architettura dell'informazione* che abbiamo introdotto in Sez. 4.3.1, non può non colpire la sua forte analogia con l'articolazione appena vista. Coerentemente con la terminologia già introdotta, ci sembra dunque lecito caratterizzare il concetto di modello del processo o ciclo di vita come

architettura del processo produttivo

5.2 Architetture del processo produttivo

Il concetto di metodologia di progetto è stato introdotto nella prima lezione come insieme organizzato e coerente di metodi e tecniche per la realizzazione di soluzioni a problemi di progetto. Il modello di processo, o *ciclo di vita* , adottato è un elemento determinante della metodologia di progetto che caratterizza un processo produttivo. Alla determinazione della metodologia contribuiscono anche altri elementi, come vedremo nella prossima lezione, ma nell'ambito del presente discorso si identifica la scelta di una metodologia di progetto con quella del ciclo di vita.

Si presenta nel seguito una panoramica dei modelli più affermati negli ultimi trent'anni

5.2.1 Modello a cascata

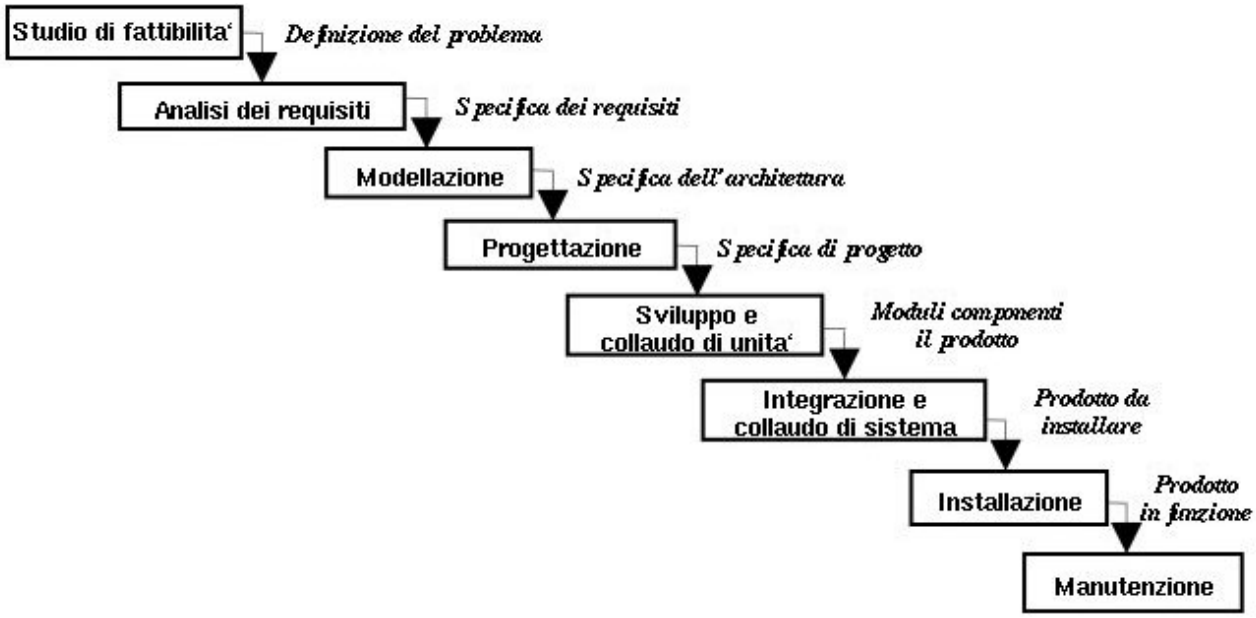

Figura 5.1: Un ciclo di vita a cascata

Sommario:

- **sequenza di fasi che elaborano semilavorati**
- **variabilità delle fasi e dei loro confini**
- **ricicli tra le fasi**
- **limitazioni del modello a cascata**
- **pregi del modello**

Il modello più semplice e tradizionale è detto *a cascata* (ingl. *Waterfall Model*) in quanto consta, come si è già detto, di una sequenza temporale di *fasi* , dove ogni fase intermedia elabora un *semilavorato* prodotto nella fase che la precede per generarne uno per l'elaborazione nella fase successiva, v. Fig. 5.1.

Esistono molte varianti di questo modello, che differiscono per numero di fasi e terminologia, ma che hanno in comune i tratti essenziali appena descritti. Va anche detto che non sempre i confini tra le fasi sono netti, e che la presenza o meno di alcune fasi può dipendere da caratteristiche sia del contesto produttivo che degli utenti. Ad esempio, la fase di installazione ha un ruolo importante ed autonomo nel caso di prodotti destinati al mercato aperto, mentre perde di significatività e viene assorbita nella fase precedente se l'utenza è limitata e ben definita, ad esempio quando è interna all'organizzazione in cui ha luogo lo sviluppo.
La versione originaria del modello (Royce 1970) prevedeva anche flussi di *feedback* , cioè in direzione opposta a quella della sequenza, per tener conto della possibilità di *ricicli* di semilavorati di cui una successiva elaborazione riveli difetti o comunque aspetti suscettibili di miglioria. Nella pratica industriale, tuttavia, questa possibilità è stata eliminata dal modello, per ragioni pratiche di semplicità.
Si è già identificata la ragione fondamentale di inadeguatezza del modello a cascata nella forte evolutività dei tipici sistemi software o siti Web, di complessità non banale. Aggiungiamo qui alcuni tratti particolari di questo argomento, per poi discutere le ragioni per le quali il modello a cascata risulta tuttavia di interesse pratico in molte situazioni, nonostante i suoi limiti.
Il punto di partenza del ciclo di vita a cascata sono i bisogni degli utenti: la loro analisi nella fase di studio di fattibilità conduce alla definizione del problema per la cui soluzione si intraprende il resto del processo produttivo. Il risultato finale del ciclo di vita è il prodotto in funzione per l'uso, dove la possibilità di correzione di difetti o di altri miglioramenti è contemplata nella fase di manutenzione. In queste due fasi, dunque, il modello naturalmente prevede la possibilità di interazione con gli utenti. Un limite del modello a cascata è che queste sono le *sole* fasi in cui questa possibilità sia prevista: una volta avviato il processo oltre lo studio di fattibilità, agli utenti tocca solo attendere il prodotto finale, senza alcuna opportunità di intervento nel corso del processo per eventuali modifiche dei loro requisiti. Inoltre, fin tanto che la manutenzione è meramente *correttiva,* cioè si limita alla correzione di difetti, appare perfettamente ragionevole considerarla come una fase del processo, in cui gli utenti segnalano i difetti e i produttori provvedono alle riparazioni. Accade spesso però che l'esperienza d'uso riveli non solo difetti ma anche necessità di adattamenti a mutamenti ambientali o desiderabilità di nuove funzionalità. Queste forme di "manutenzione", rispettivamente denominate *adattativa* e *perfettiva,* spesso assumono i connotati di un processo produttivo in piena regola, e appare dunque riduttivo confinarle in una fase marginale del processo. Infine, i ricicli sono stati eliminati dal modello per delle (buone, come vedremo fra poco) ragioni pratiche, tuttavia in realtà esistono, giacchè si fa esperienza, e da questa si impara, durante *tutte* le fasi produttive: è questo il punto della citazione che decora l'inizio di questa lezione.
Dati i limiti appena discussi, il modello a cascata risulta tuttavia utile in pratica, e la sua adozione, dove giustificata, presenta vantaggi non indifferenti. Vediamo prima questi, per poi capire in quali casi siano disponibili.
Sarà opportuno innanzitutto ricordare che il modello a cascata è nato da una esigenza di rigore formale, in risposta alla pratica diffusa del *code-and-fix* ("codifica e aggiusta") che vedeva la produzione di software procedere senza alcun piano o architettura di processo, per lasciare poi alle virtù "magiche" del programmatore, unico depositario della conoscenza del prodotto, il compito di risolvere i problemi rivelati dal suo uso. Formalizzare la produzione di semilavorati nelle varie fasi e l'esecuzione di attività di collaudo in corso d'opera è stato un progresso significativo, che naturalmente comporta la disponibilità di *documentazione* , stratificata per livelli di dettaglio, sia sul processo produttivo, sia sul prodotto finale. A questo primo, indiscutibile vantaggio, il modello a cascata ne aggiunge altri, dovuti alla semplicità della sua struttura sequenziale. Questa infatti agevola la *pianificazione* delle attività produttive, permettendo di associare previsioni di tempi, costi e allocazione di risorse alle varie fasi del processo, ed il *controllo* dello svolgimento del processo

con riferimento al piano dello stesso. Ciò spiega la ragione dell'eliminazione dei ricicli dal modello a cascata, in quanto la loro presenza complica sensibilmente la pianificazione.
Se dunque appare chiaro che l'adozione di un modello a cascata comporta vantaggi di notevole interesse pratico, quando risulta realistico il suo uso? Dagli argomenti critici addotti sopra, una prima risposta emerge con naturalezza: in presenza di *stabilità dei requisiti* , e per processi produttivi non troppo complessi. Se questa risposta appare deludente, e certamente insufficiente a spiegare l'interesse industriale verso questo modello, si consideri anche che le due condizioni suddette possono ben essere verificate da *parti* costituenti di altri modelli di processo. Ad esempio, ad un prodotto di grandi dimensioni e ricco di funzionalità sarà meglio adeguato un ciclo di vita incrementale (di cui stiamo per dire di più nel seguito), in cui lo sviluppo di ciascun incremento avvenga secondo un modello a cascata, con i benefici che questo comporta. Anche in un ciclo di vita evolutivo (che esaminiamo più avanti) si presenta una opportunità analoga, in quanto esso prevede lo sviluppo di molteplici versioni del prodotto, dove ciascuna versione può essere sviluppata secondo un modello a cascata.

5.2.2 Modello incrementale

early-subset, early-delivery

- **realizzazione del prodotto per progressivi incrementi di funzionalità**
- **vantaggi: economici + feedback precoce da utenti o committenti**
- **criteri di scelta delle priorità di realizzazione: dipendenze fra le funzionalità, stabilità dei requisiti, etc.**

Quando il problema si presenti complesso e faccia prevedere lo sviluppo di un prodotto ricco di funzionalità relativamente indipendenti, risulta possibile e conveniente pianificare uno sviluppo per sottosistemi separati. Si può in tal modo anticipare la consegna dei primi sottosistemi agli utenti, dunque senza dover attendere il rilascio di tutto il prodotto (tecnica nota con il termine *early-subset, early-delivery*)
Si realizzano dapprima le funzionalità essenziali, specialmente se la loro disponibilità è necessaria alla realizzazione delle altre. La consegna precoce di questo nucleo essenziale (ingl. *core*) di funzionalità agli utenti ne consente la validazione immediata, o altrimenti il rilievo *precoce* della necessità di correzioni. Queste possono cioè essere apportate *prima* che lo sviluppo del resto del prodotto abbia inizio, dunque con notevole riduzione dei costi rispetto al verificarsi di un rilievo analogo in un ciclo di vita a cascata. In quest'ultimo, infatti, si ha il *feedback* degli utenti solo quando *tutto* il prodotto è già stato sviluppato, dunque modifiche al suo nucleo essenziale solitamente comportano modifiche in molte altre parti del prodotto.
Ciascun sottosistema successivo al primo realizza un *incremento* di funzionalità rispetto al nucleo essenziale, da cui il nome del ciclo di vita. Modelli incrementali offrono inoltre un vantaggio rispetto al problema dell'evolutività: in presenza di incertezza o instabilità di una parte dei requisiti, si può iniziare con lo sviluppo dei sottosistemi che realizzano la parte più stabile dei requisiti, e differire il resto dello sviluppo sino a quando, per i requisiti più incerti, non sia disponibile una maggiore stabilità. Ad ottenere questa di solito contribuisce proprio la disponibilità precoce di una parte del prodotto totale, giacché consente agli utenti di sperimentare il soddisfacimento (di una parte) delle loro attese e di precisare aspetti dei requisiti che potevano essere poco chiari a loro stessi in sede di prima definizione del problema.

5.2.3 Prototipazione rapida

- **throwaway prototype (prototipo usa-e-getta)**
- **l'interfaccia di utente come prototipo**
- **prototipazione evolutiva**

Se, in accordo con la citazione iniziale, è fatale che ogni prima realizzazione di un (nuovo) prodotto debba essere insoddisfacente, allora sembra preferibile tenerne conto dal principio. Da qui nasce l'idea di sviluppare un prototipo *usa-e-getta* (ingl. *throw-away prototype*) al fine di acquisire esperienza, prima dello sviluppo vero e proprio. F. Brooks così si esprime in proposito:

> *The management question [...] is not whether to build a pilot system and throw it away. You will do that. The only question is whether to plan in advance to build a throwaway, or to promise to deliver the throwaway to customers ...*
> Frederick P. Brooks, Jr. , 1975 [2] (Brooks 1975)

All'esperienza acquisita è certo utile la validazione da parte degli utenti, e il prototipo è infatti di solito progettato per agevolare questa interazione. Ciò presenta analogie con la situazione nel modello incrementale, eccetto che le funzionalità, anche se essenziali, non sono necessariamente *realizzate* nel prototipo, specie se la validazione è il solo scopo del suo sviluppo. In tal caso basta realizzare solo la *presentazione* delle funzionalità stesse, cioè la loro interfaccia di utente: esistono strumenti di sviluppo che agevolano questo lavoro, e ciò consente di ottenere il prototipo in tempi rapidi ed a costi ridotti.
L'idea della prototipazione può essere combinata con quella dello sviluppo incrementale se, invece di gettar via il prototipo dopo la validazione, si pianifica di incorporarlo nella realizzazione del successivo incremento di prodotto: ne risulta il ciclo di vita *incrementale con prototipazione evolutiva* . Questo propriamente rientra nella categoria di modelli che segue.

5.2.4 Modelli evolutivi

Una visione realistica del ciclo di vita di prodotti così spiccatamente evolutivi quali il software ed i siti Web deve tener conto di almeno due categorie di rilevanti mutamenti ambientali, per le quali indichiamo brevemente caratteristiche tipiche di modelli evolutivi che esse motivano.

- **evoluzione dell'uso del prodotto progettazione partecipativa (v. quarta lezione), sviluppo incrementale con prototipi successivi**
- **evoluzione delle tecnologie usate nel prodotto sviluppo per componenti specificati da interfacce, modularità e stratificazione**

Della prima classe di caratteristiche si è già detto abbastanza, nei limiti di questa trattazione. La seconda classe segue i principi di modularità, astrazione e adattabilità (v. prima lezione). Il concetto di *interfaccia* di un modulo gioca un ruolo analogo a quello della specifica dei requisiti rispetto al processo produttivo: una interfaccia descrive *cosa* il modulo realizza senza alcun

dettaglio su *come* lo realizzi. Se l'interazione con il modulo avviene solo sulla base della sua interfaccia, risulta possibile la sua sostituzione "indolore" con un altro, costruito in tutt'altro modo, purché rispetti l'interfaccia data.

modelli trasformazionali

I *modelli trasformazionali* sono modelli evolutivi in cui lo sviluppo è basato sull'uso di metodi formali per la specifica di requisiti e interfacce, il loro *raffinamento* secondo varie semantiche formali, e la verifica (in parte automatica) di correttezza (Hutter et al. 2002). Il nome è dovuto all'idea che il raffinamento sia una trasformazione di una specifica astratta (requisiti, interfaccia) in una meno astratta, in quanto si aggiunge informazione su un particolare modo di soddisfare i requisiti, che si dimostra matematicamente corretto.
La classe più generale di modelli evolutivi sembra essere la seguente.

5.2.5 Modello a spirale

Originariamente proposto da (Boehm 1988), ne sono state elaborate più varianti (fra cui (Boehm 1998) dallo stesso autore), che hanno i seguenti

Figura 5.2: Un ciclo di vita a spirale

tratti salienti (v. Fig. 5.2):

- **iterazione**
  una visione iterativa, senza fine, del ciclo di vita, dove ogni iterazione attraversa una sequenza di fasi che si ripete all'iterazione successiva;
- **contrazione delle fasi di sviluppo**
  la contrazione di quasi tutte le fasi tipiche del ciclo di vita a cascata, per ciascuna iterazione, in un'unica fase (la IV) di questo modello;
- **interazione con gli utenti**
  l'interazione con gli utenti, nella fase I, o di validazione, ad ogni iterazione: questa fase funge da cerniera fra iterazioni successive;
- **metamodello**
  la possibilità di fungere da meta-modello del processo di sviluppo, in quanto la fase II, o di pianificazione, che ha per oggetto la determinazione di alternative, obiettivi e vincoli per l'iterazione, può includere fra le alternative più modelli di processo per la fase IV, e la fase III scegliere fra questi il modello che risulta più conveniente, a seguito di una analisi dei rischi. Il modello di sviluppo può dunque cambiare da una iterazione all'altra.

Va notato che, mentre il significato di "validazione" è quello usuale nelle iterazioni successive alla prima, in questa esso appare meno ovvio. Precisiamo che in tal caso il termine va inteso come "identificazione di bisogni degli utenti", ovvero validazione non tanto di un prodotto (che non c'è) quanto della aderenza di possibili *intenzioni* progettuali ad effettive esigenze degli utenti. Si rinviano alla settima lezione ulteriori approfondimenti sull'analisi dei rischi.

5.3 Una ipotesi di lavoro

5.3.1 Premessa

Consideriamo come *ipotesi di lavoro* il seguente

Problema

Organizzare (una parte del)la documentazione prodotta dai partecipanti a questo corso nella forma di un sito Web adatto all'uso da parte di docenti e partecipanti a future edizioni del corso stesso.

In questa ipotesi, proponiamo di elaborare le seguenti questioni.

5.3.2 Esercizio 1: caratteristiche di qualità

Determinare caratteristiche di qualità desiderabili di una soluzione al problema.

5.3.3 Esercizio 2: vincoli del processo produttivo

Individuare vincoli di un processo produttivo attraverso cui si voglia generare un prototipo di soluzione al problema in questa edizione del corso.

5.3.4 Esercizio 3: fattibilità

Stabilire quali delle caratteristiche di qualità determinate con l'Esercizio 1 sembrano fattibili in presenza dei vincoli individuati con l'Esercizio 2.

5.3.5 Esercizio 4: ciclo di vita

Soppesata l'applicabilità dei modelli di processo produttivo delineati in Sez. 5.2 al problema in questione, individuare il modello preferibile e motivarne la scelta.

5.4 Note

1. Questa è l'età dei Metodi: e l'università che debba essere l'esponente della condizione di vita della mente umana deve essere l'università dei metodi. [...] Purtroppo la pratica generalmente precede la teoria, ed è il solito destino degli umani di aver le cose fatte dapprima in qualche modo timidamente approssimativo, e di scoprire in seguito come avrebbero potuto farle molto più facilmente e perfettamente.
   Citazione originale tratta da: Eisele, Carolyn (Ed.), *Historical Perspectives on Peirce's Logic of Science: A History of Science,* par. 941. Mouton, Berlin, 1985.
2. Il problema di gestione [...] non è se costruire un sistema pilota e gettarlo via. Lo farai. Il solo problema è se pianificare in anticipo di costruire qualcosa da gettar via, o se promettere di consegnare ai clienti qualcosa da gettar via ...

5.5 Bibliografia

- **Boehm, B.**, 1988. A Spiral Model for Software Development and Enhancement. *Computer*, 21:5, 61-72.
- **Boehm, B.**, 1998. Using the WINWIN Spiral Model: A Case Study. *Computer*, 31:7, 33-44.
- **Brooks, F.P. Jr.**, 1975. *The Mythical Man-Month*. Addison-Wesley.
- **Hutter, D., Basin, D., Lindsay, P., Lüth, C. (Eds.)**, 2002. *First Workshop on Evolutionary Formal Software Development (EFSD 2002)*. Copenhagen, July 21, 2002.
  Web: http://floc02.diku.dk/EFSD
- **Royce, W.W.**, 1970. Managing the Development of Large Software Systems: Concepts and Techniques. *In: Proc. WESCON*. August, 1970.