Ingegneria del software nel progetto di siti Web

Indice

1. Ingegneria del software nel progetto di siti Web

In questa introduzione alla disciplina del *software engineering* (SE) si presentano:

- **un contesto di ingegneria del software al progetto di siti Web** (ingl. *Web design* (WD).
  si mettono in evidenza molteplici analogie fra le attività produttive di software e di siti Web, al punto da includere questi ultimi in una più ampia accezione del termine *software* ;
- **principi e concetti di base di ingegneria del software**
  si passano in rassegna principi ingegneristici tradizionali e concetti di base dell'ingegneria del software, comunemente accettati nella pratica professionale di questa disciplina;
- **il progetto di siti Web quale attività di ingegneria del software**
  si caratterizza l'applicabilità dei principi e concetti già introdotti ad una vasta classe di progetti di siti Web;
- **problemi di Web design, soluzioni di ingegneria del software**
  si propongono alcuni tipici problemi di Web design, e si invita alla riflessione sull'uso dei suddetti principi per l'identificazione e la valutazione di soluzioni a tali problemi.

1.1. Un contesto di ingegneria del software al progetto di siti Web

Principi e pratiche consolidati di ingegneria del software risultano utili nelle attività legate alla progettazione di siti Web

Non è necessario essere ingegneri per apprezzare il vantaggio di sviluppare progetti di siti Web con le metodologie tipiche dell'ingegneria del software.

1.1.1. Tradizionali attività produttive del software

In *prima approssimazione* , si possono distinguere attività di:

- **pianificazione**
- **realizzazione**
- **esercizio o gestione**
- **manutenzione**

In una visione tradizionale del processo produttivo del software, tali categorie di attività costituiscono *fasi* di una sequenza nel tempo, nelle quali si raggruppano specifiche attività produttive. Questa visione non è invero limitata alla produzione di software, in quanto è facile riscontrarla in molti altri processi produttivi:

1. prima si pianifica (lo sviluppo di un prodotto, sia esso un bene o un servizio, che esiste solo nelle intenzioni dei progettisti, e di eventuali committenti);
2. poi lo si realizza (e durante questo sviluppo l'esistenza del prodotto acquista

progressivamente concretezza), e

3. solo quando lo stato di concretezza del prodotto desiderato può dirsi soddisfacente, cioè al termine della sua realizzazione, ha inizio la sua gestione secondo le modalità di uso, o esercizio, pianificate o concordate con i committenti;
4. poiché infine la perfezione non è propria dei risultati di sforzi umani, l'esercizio di un bene o servizio ne rivela sempre limiti, errori o comunque caratteristiche suscettibili di miglioramento: la manutenzione di un prodotto software (o altro, ad esempio un sito Web), mentre ha luogo in concomitanza al suo esercizio, logicamente lo segue, in quanto è l'*uso* del prodotto che rivela la necessità o convenienza di interventi di manutenzione, dei quali determina gli scopi.

1.1.2. Natura non sequenziale del processo produttivo

La stretta sequenzialità temporale implicita in almeno le prime tre delle suddette quattro fasi del processo produttivo del software, nella sua più semplice e tradizionale visione, è ingannevole, e risulta inadeguata a render conto di fenomeni emergenti in misura crescente nelle attuali modalità produttive del software, come stiamo per argomentare più in dettaglio. Prima di addentrarci in dettagli, tuttavia, è utile notare che, *per ragioni del tutto analoghe* , la semplice articolazione in sequenza di fasi produttive risulta inadeguata a render conto di, comprendere e gestire, i fenomeni salienti della progettazione di siti Web. Possiamo dunque già trarre due conclusioni, delle quali la prima riassume l'esito dell'argomentazione esposta nel paragrafo precedente, mentre la seconda ne limita severamente la validità e prospetta altri esiti, tutti da esplorare:

1. la concezione del processo produttivo come sequenza temporale di fasi di pianificazione, realizzazione, esercizio e manutenzione, si applica tanto bene alla progettazione del software quanto alla progettazione di siti Web;

2. la suddetta concezione si applica non solo altrettanto bene ma anche altrettanto **male** ai due tipi di progettazione in considerazione, per ragioni del tutto analoghe.

1.1.3. Doppio contesto evolutivo del processo produttivo

Come già enunciato, l'inadeguatezza dell'articolazione sequenziale del processo produttivo, sia di software che di siti Web, emerge dalla considerazione di fenomeni che si manifestano nel mondo reale della produzione, sia dell'uno che degli altri. Per inquadrare la faccenda in un panorama concettuale sufficientemente ampio, come sembra appropriato al carattere introduttivo di queste osservazioni, è utile annodare il filo del nostro discorso a quello sulla tematica dell'interazione uomo-macchina (ingl. *Human-Computer Interaction* , abbr. HCI) (Hewett 2004).
Si dice che un'immagine valga mille parole; ne proponiamo qui una, che ben corrobora il detto, in Figura 1.1

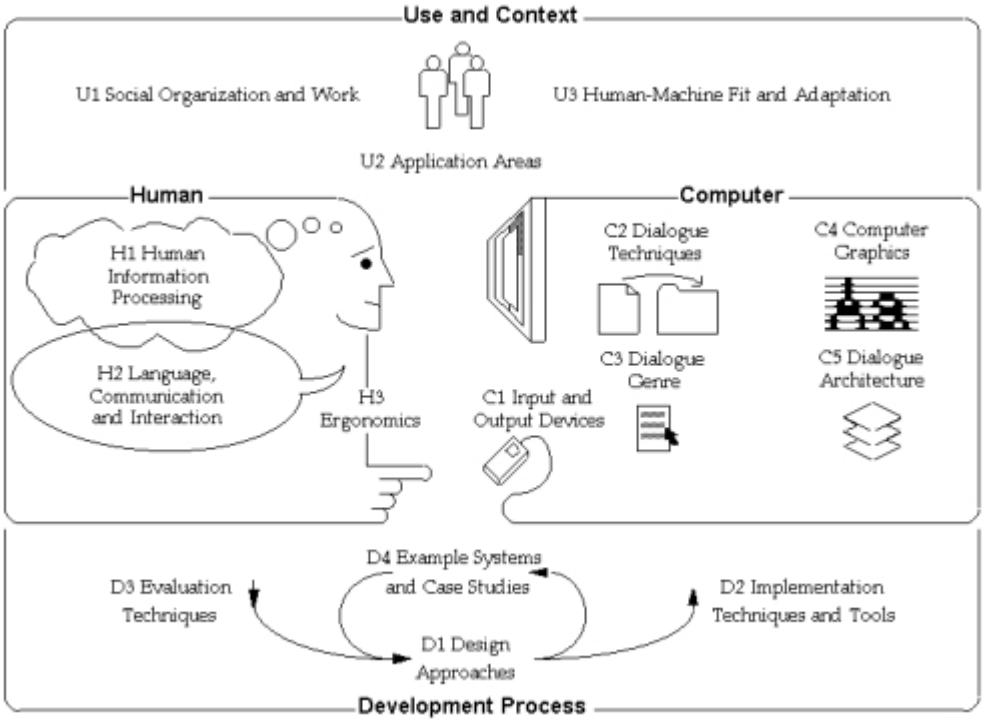

Figura 1.1: HCI fra sviluppo e uso (Hewett 2004)

La figura è tratta [1] da (Hewett 2004, p. 15), Sezione 2.3.
L'immagine raffigura l'interazione uomo-macchina al centro di un doppio contesto: il *contesto di uso* da una parte, quello dei *processi di sviluppo* dall'altra. Quest'ultimo è raffigurato da due percorsi ciclici:

- uno, interno al contesto stesso, rappresenta la mutua interazione fra l'emergere di sempre nuovi approcci alla progettazione e la proliferazione di esempi e casi di studio che li motivano da un canto e ne costituiscono un banco di prova dall'altro;
- l'altro coinvolge, per l'appunto, l'interazione uomo-macchina, attraversando entrambi i lati dell'interazione per diffondersi nella molteplicità dei modi in cui il vivere umano e sociale assorbe i prodotti tecnologici, ne viene modificato, ma allo stesso tempo ne modifica i processi di sviluppo attraverso le varie forme di valutazione che l'uso umano fornisce al processo di sviluppo.

Ora, è proprio la caratteristica *ciclica* di questi percorsi a rivelare l'inadeguatezza della concezione sequenziale dei processi produttivi di nostro interesse. Software e siti Web condividono una forte connotazione *evolutiva* del loro sviluppo, il cui *ciclo di vita* (espressione su cui torneremo più avanti, nel Cap. 5) deve fare i conti con l'evoluzione continua, ed oggi particolarmente rapida, sia nei contesti di uso dei prodotti (software o siti Web) che nel ventaglio delle tecnologie disponibili per la loro produzione (strumenti di sviluppo, normativa tecnica, etc.). Mentre riprenderemo il discorso su modelli del processo produttivo del software nel contesto di una introduzione alle metodologie di progetto tipiche del SE (Cap. 5), qui ci limitiamo a mettere in luce la sostanziale indifferenza, degli argomenti con cui abbiamo effettuato questa prima valutazione di un tale modello, rispetto alla progettazione di software o di siti Web.

1.1.4. Cosa viene appresso

- **fondamenti di ingegneria del software**
- **applicabilità alla progettazione di siti Web**

Ci è sembrato appropriato iniziare l'argomentazione sulle forti analogie fra progettazione di software e di siti Web considerandone una comune caratteristica astratta e di profondo significato, l'evolutività. Esistono molti altri e spesso più concreti aspetti di similarità fra le due. Verranno trattati a tempo debito, per lo più nelle parti più tecnicamente orientate del corso. è bene però aver presente sin dall'inizio le ragioni *profonde* della rilevanza dei contenuti proposti agli obiettivi del corso. In questo primo capitolo esploriamo dunque ulteriormente le più basilari analogie tra SE e WD, procedendo ad un breve esame di alcuni principi e concetti di base comunemente assunti in SE, per stimarne successivamente l'applicabilità al WD.

1.2. Principi comuni di ingegneria del software

Dalla pratica di:

- **decenni di esperienza di costruzione di software**
- **secoli di esperienza di progetti complessi**

Mentre si rinvia al successivo Cap. 2 una presentazione ed argomentazione dettagliata di principi di progetto di qualità, ne introduciamo qui alcuni quali sono comunemente intesi in SE, cioè quali regole di buon senso da seguire nella progettazione del software. Come argomenteremo successivamente, grazie alle forti caratteristiche di evolutività inerenti alla grande maggioranza di casi di costruzione di siti Web, questi principi di buona progettazione di SE si applicano al WD con validità ed utilità inalterate.
Secondo (Ghezzi et al. 2004, p. 45), i fondamenti di SE riposano su alcuni principi ingegneristici, comunemente accettati nella pratica di questa disciplina. Questi principi si presentano come criteri utili ad orientare l'attività del progettista, specie quando ha a che fare con problemi di dimensioni significative ed in cui il ventaglio delle opzioni richiede frequentemente dei canoni orientativi che favoriscano consistenza e coerenza nelle decisioni di progetto. Passiamo brevemente in rassegna i principi in questione, anticipando che alcuni di questi verranno ripresi in considerazione, con ulteriori argomenti ed in una prospettiva più ampia, nel successivo Cap. 2.

1.2.1. Rigore formale

Molteplici aspetti:

- **il software è un prodotto inerentemente formale**
- **il principio vale anche per il processo produttivo del software**
- **rigore non implica rigidità**
- **documentazione del software:** *consistenza notazionale*
- **metodi formali di costruzione e verifica del software: il rigore della matematica**

L'essenza del software è costituita da idee (di soluzioni a problemi) che prendono forma nei termini simbolici di linguaggi formalmente definiti, i cui costrutti sono effettivamente interpretabili da macchine fisiche. Se la forma in cui si esprime l'idea non è rigorosamente corretta, il risultato non è garantito, e può essere (spiacevolmente) sorprendente. Il canone del rigore formale è dunque inerente alla natura stessa del software.
Il principio ha però una portata più ampia in processi produttivi complessi, che coinvolgono più produttori o gruppi di lavoro, e i cui prodotti devono durare nel tempo, possibilmente attraverso evoluzione e adattamenti. In queste condizioni, una rigorosa formalizzazione degli aspetti salienti del processo e del prodotto (processo: ruoli, compiti, responsabilità, meccanismi di controllo e di verifica, etc.; prodotto: contesto operativo, funzionalità, requisiti, architettura, meccanismi di interazione tra le sue parti, etc.), risulta essenziale alla fluidità dello sviluppo. Questo potrà poi meglio adattarsi a mutamenti ambientali se la formalizzazione delle sue regole, pur *rigorosa* , non è *rigida* , bensì prevede meccanismi di adattamento e modifica delle regole stesse.
Va poi aggiunto che a ciascun ambito del processo produttivo può essere appropriato un diverso livello di rigore formale: ad esempio, la descrizione del dominio applicativo in cui si colloca un prodotto software sarà in prima approssimazione sempre descritta in un linguaggio naturale, magari con lessico specifico del dominio stesso, mentre raffinamenti di aspetti della stessa descrizione che caratterizzano l'ambiente operativo del prodotto finale potranno essere formalizzati nei termini del linguaggio di quest'ultimo.
Naturalmente, oltre che al processo produttivo, canoni di rigore formale si applicano anche ai prodotti dello stesso, il software dunque, ma anche annoverando fra essi la *documentazione* - di particolare importanza per prodotti di grandi dimensioni - la cui accuratezza, consistenza notazionale e facilità di consultazione sono determinanti per l'effettiva usabilità del prodotto e per la rapida identificazione delle sue parti interessate da eventuali modifiche.
Quanto al software, la più rigorosa interpretazione che è lecita del principio in questione è quella matematica, disciplina del rigore formale per eccellenza. *Metodi formali* di specifica e costruzione del software, e di verifica della sua correttezza, sono di grande interesse industriale per i molti vantaggi che inducono nella qualità dei processi produttivi e del software prodotto, e ne esiste oggi un'abbondante proliferazione, come testimoniato dai siti (Bowen 2005), (FME 2006). Naturalmente, la loro adozione richiede un certo grado di familiarità dei produttori con i fondamenti matematici degli stessi, condizione che non è sempre agevole soddisfare.
Il principio ingegneristico di rigore formale può essere ricondotto ad un più basilare principio filosofico di *consistenza* , che discuteremo nel Cap. 2.

1.2.2. Separazione di aspetti diversi

Seguire questo principio significa isolare ed affrontare separatamente i vari risvolti di un problema complesso, ad esempio rispetto agli aspetti elencati appresso.

- **caratteristiche degli utenti**
- **diverse caratteristiche dell'ambiente di produzione, ad es. disponibilità di:**
  - **conoscenze tecniche** (ingl. *know-how*)
  - **componenti riusabili per la soluzione**
  - **risorse addizionali su richiesta**
- **diverse proprietà del sistema che si considera come soluzione**
- **tempo necessario alla produzione del sistema.**

L'adesione a questo principio favorisce non solo l'analisi del problema, secondo la classica regola del *divide et impera* , ma anche la successiva applicazione, nel processo di sviluppo della soluzione, del principio di modularità, introdotto appresso.
Notiamo infine che, quando si consideri la separazione di aspetti reciprocamente *indipendenti* , il principio che ne prescrive il mantenimento dell'indipendenza, sia nell'analisi del problema come nello sviluppo della soluzione, prende il nome di *ortogonalità* , e viene più diffusamente trattato nel Cap. 2.

1.2.3. Modularità

Anche questo è un principio di separazione, che differisce dal precedente solo in quanto quello riguarda *il problema* , mentre questo riguarda *la soluzione* .

- **architettura software:** *high cohesion, low coupling*
- **progettazione** *Bottom-Up*
- **progettazione** *Top-Down*

La decomposizione di un sistema in parti, o *moduli* , può in genere essere effettuata in molti modi, spesso equivalenti dal punto di vista dell'utente del sistema, che non necessariamente percepisce le differenze di comportamento fra sistemi che presentano gli stessi servizi ma differiscono nel modo in cui sono internamente organizzati. Il problema di centrale interesse al disegno di ciò che comunemente si intende come *architettura del software* consiste proprio nella determinazione della più conveniente decomposizione del sistema in parti, secondo criteri ingegneristici (Parnas 1972). Ad esempio, un criterio generalmente utile prescrive l'articolazione dell'architettura in moduli caratterizzati da elevata coesione interna (cioè alto grado di interdipendenza fra i componenti interni a ciascun modulo) e bassa interdipendenza fra di essi (ingl. *high cohesion* e *low coupling*).
Mentre dunque la motivazione di questo principio rimane quella della classica regola richiamata sopra, la sua applicazione richiede criteri addizionali, quali quello suddetto e gli altri principi trattati in questa sezione, ed eventuali altri orientamenti metodologici quali, ad esempio, i due classici approcci alla progettazione:

- *Bottom-Up:* sintesi del sistema per successiva composizione di sottosistemi o moduli, a partire dai componenti più semplici, già disponibili;
- *Top-Down:* analisi del problema per successiva decomposizione in sottoproblemi, fino alla identificazione di problemi che non richiedono ulteriore analisi giacché componenti che ne realizzano la soluzione già esistono.

1.2.4. Astrazione

Si applica in tutte le fasi e attività produttive:

- **criterio di eliminazione del rumore**
- **livelli di astrazione: nel processo e nel prodotto**

I già visti principi di separazione rispettivamente isolano aspetti del problema e parti del sistema. Il principio di astrazione, che si applica in tutti i momenti della progettazione, prescrive di escludere

aspetti e parti irrilevanti all'oggetto dell'attività di progettazione in ciascun dato momento; lo si potrebbe altrimenti chiamare un criterio di eliminazione del rumore.
L'astrazione è spesso vista come un ordine in *livelli di astrazione,* ma raramente il significato di questo termine viene precisato con rigore. Un senso di questo termine fa riferimento a un processo produttivo che prende le mosse da una definizione del problema e si conclude con la realizzazione di un sistema che ne costituisce la soluzione: la già accennata visione sequenziale del processo di produzione del software. In tale visione, la progettazione procede per raffinamenti successivi dell'informazione relativa al suo oggetto, la cui descrizione diventa via via più dettagliata. Si procede in tal senso per successivi abbassamenti del livello di astrazione, i livelli più alti essendo quelli iniziali nel processo produttivo, nei quali l'informazione rilevante riguarda prevalentemente il problema e ben poco ha a che fare con caratteristiche della sua soluzione, ancora da progettare.
Ulteriore discussione di questo principio (anche se sotto altro nome: *proprietà*) viene proposta nel Cap. 2.

1.2.5. Anticipazione del cambiamento

Il principio è inerente alla forte caratteristica evolutiva del software.

Si è accennato, in una sezione precedente, alla forte connotazione evolutiva del processo di sviluppo del software, e questo fatto verrà ripreso in considerazione in una prossima sezione, quale ragione principale dell'analogia fra progettazione di software e di siti Web. Il principio qui enunciato è dunque inerente a questa fondamentale caratteristica del software.
Ulteriore discussione di questo principio (sotto il nome di *adattabilità*), a partire dalle sue antiche radici filosofiche, viene proposta nel Cap. 2.

1.2.6. Generalità

Soluzioni a problemi più generali sono più facilmente riusabili, ma spesso più difficili a farsi.

Altra norma di buona progettazione è quella per cui si tenta, dato un problema specifico, di estrapolarne le caratteristiche che lo qualificano come caso particolare di un problema più generale. Se la soluzione a questo è già disponibile, è evidente il vantaggio del suo *riuso* . Altrimenti, ci si può prefiggere lo scopo di risolvere il problema più generale, al fine di ottenere una soluzione riusabile in futuro. Tale tecnica, però, spesso porta ad introdurre un grado di difficoltà maggiore di quello del problema originale; è dunque necessario riuscire a soppesare costo e benefici della generalizzazione del problema, ed a stimare la fattibilità della sua soluzione.
Ulteriore discussione di questo principio viene proposta nel Cap. 2.

1.2.7. Incrementalità

Incrementi di prodotto forniscono le funzionalità individuate da incrementi di analisi del problema.

Connesso al principio di modularità di una buona progettazione è il principio di incrementalità. Questo vale sia per l'analisi del problema come per la costruzione del prodotto. La combinazione dei due principi prevede che per ogni aspetto individuato da un incremento di analisi si realizzi un risultato parziale, cioè un corrispondente incremento di prodotto, che esibisca la prescritta funzionalità senza dover attendere che *tutto* il prodotto sia realizzato. Chiaramente, un certo grado di modularità del prodotto finale è in tal modo automaticamente realizzato.

1.3. Concetti di base di ingegneria del software

Come abbiamo riportato all'inizio della sezione precedente, i principi esaminati sinora sono visti in (Ghezzi et al. 2004, p. 45) come la base su cui riposano i fondamenti di SE. Questi sono concepiti come una struttura complessa, di cui i principi costituiscono solo il nucleo primario. Cos'altro c'è dunque nei fondamenti di SE? La Figura 1.2 dà una risposta a questa domanda, raffigurando la relativa collocazione degli altri *concetti di base* che, assieme ai principi visti, costituiscono la struttura fondamentale della disciplina.

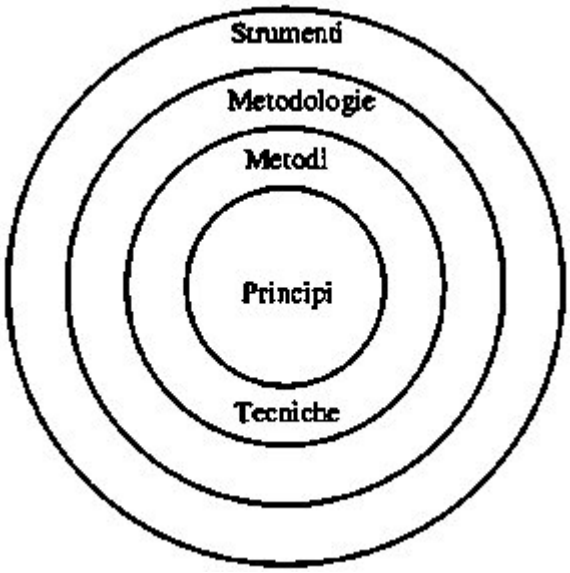

Figura 1.2: Concetti di base di SE (Ghezzi et al. 2004)

.

Sebbene il significato dei concetti riportati in figura sia presumibilmente noto, è preferibile precisare il senso in cui essi sono comunemente intesi nell'ambito di nostro interesse, giacché trattasi di termini di uso frequente ma con diverse sfumature di significato.

1.3.1. Terminologia di base

Metodi, tecniche

linee-guida o regole che governano le attività di SE

Sono applicabili in tutte o in alcune delle fasi della progettazione; possono essere predefinite o, al contrario, definite *ad hoc* , in considerazione di aspetti particolari del problema, dell'ambiente di

sviluppo, di varie caratteristiche del gruppo di lavoro, etc.

Metodologie

> collezioni organizzate di metodi e tecniche, opportunamente selezionate ed integrate fra loro per costituire un insieme coerente, adatto alla realizzazione di soluzioni a problemi di progetto aventi caratteristiche generali comuni

Strumenti

> i supporti software necessari o utili all'attività di progetto e sviluppo, secondo una metodologia data

Frequentemente il successo operativo di metodologie complesse dipende in modo cruciale dalla disponibilità di strumenti che forniscano un aiuto effettivo alla messa in pratica delle prescrizioni metodologiche. Gli strumenti a disposizione dovranno essere sempre di buona praticità e facilmente utilizzabili, o di agevole apprendimento, da parte dei loro destinatari.

1.4. Progetto di siti Web quale attività di ingegneria del software

> caratterizzazione dell'applicabilità di principi e concetti di ingegneria del software ad una vasta classe di progetti di siti Web

1.4.1. Caratteristiche di processi di ingegneria del software

Principi e concetti di SE entrano solitamente in gioco nella costruzione di software quando questa presenti certe caratteristiche, quali quelle elencate appresso.

- **grande dimensione del prodotto**
- **notevole complessità strutturale**
- **lavoro di gruppo e ambiente di sviluppo di tipo collaborativo**
- **generazione di molteplici versioni**
- **lunga durata**
- **cambiamenti frequenti nell'evoluzione del prodotto, e.g. per:**
  - **eliminazione di difetti**
  - **miglioramenti di prestazioni**
  - **arricchimento con nuove funzionalità**
  - **adattamento a mutamenti tecnologici e nel contesto d'uso.**

Naturalmente, è possibile applicare in *tutti* i casi di progetto di software i principi di SE, essendo

questi nient'altro che delle regole di buona progettazione: tutto sommato, delle regole di buon senso corroborate dall'esperienza. Tuttavia, in presenza di una o più delle suddette caratteristiche, l'adesione a tali principi risulta determinante per la qualità sia del processo produttivo che del risultato, ed è dunque imprescindibile.

1.4.2. Caratteristiche della progettazione di siti web

- **caratteristiche simili dei processi di WD, in particolare: evolutività**
- **oggetto della disciplina del WD**
- **caratteristiche dei siti Web dinamici**
- **fonti e importanza della normativa tecnica Web**

Notiamo che, nella grande maggioranza dei casi, anche la costruzione di siti Web presenta una o più delle caratteristiche elencate sopra, ed è facile intuire che questo accade soprattutto per le caratteristiche di *evolutività* , a cui si è già fatto cenno in una sezione precedente, e che troviamo nell'ultima parte dell'elenco qui esposto. A tali casi di WD, come sarà facile intuire, i principi di SE si applicano con altrettanta validità, rispetto alla qualità di processo produttivo e prodotto. Questo fatto corrobora la nostra motivazione a prender le mosse da principi di SE in questo capitolo di introduzione al WD. *Oggetto di studio* di quest'ultima disciplina sono le metodologie che mirano allo sviluppo di piattaforme di comunicazione adeguate ai loro scopi ed accessibili su Web secondo una corretta politica di accesso, pubblico o riservato che sia. Tale può essere, ad esempio, quella che limita l'accesso ad un corso on-line ai suoi partecipanti. Un esempio, purtroppo frequente, di scorretta politica di accesso è quella, discriminatoria, implicitamente messa in opera dalla mancanza di rispetto di norme tecniche che consentono l'accesso all'informazione su Web anche in condizioni di disabilità sensoriale. Si rinvia al Cap. 4 per un'introduzione alla problematica dell'accessibilità di siti Web.
La caratterizzazione data qui dell'oggetto di interesse del WD mostra una seconda ragione per cui questa disciplina naturalmente mutua principi e metodologie dal SE: è molto frequente il caso in cui l'adeguatezza di un sito Web al suo scopo richiede la presenza di *elementi dinamici* nelle sue pagine, parti cioè il cui contenuto muta nel tempo senza per questo richiedere mutamento del *testo sorgente* di cui è fatto il sito. Ciò accade grazie alla possibilità di *incorporare software* nel sorgente stesso. Nei siti più sofisticati sotto questo profilo, il software che ne genera gli elementi dinamici costituisce la parte preponderante del sorgente del sito. Chiaramente, *in questi casi non c'è differenza tra WD e SE* .
Alla natura *multilinguistica e multitecnica* dei siti Web va ricondotto il fatto, fondamentale sin dall'affermazione iniziale della rete, che il loro sviluppo e la loro interconnettività sono sostenuti e regolati da una serie di norme tecniche (ingl. *standard*) in continua e tuttora rapida evoluzione. Il *World Wide Web Consortium* (W3C 2006) è la fonte di queste Raccomandazioni, per le quali rende liberamente disponibili molta documentazione, *forum* di discussione e strumenti di verifica di conformità alle stesse. Avere sempre presente, in ogni fase del processo produttivo di un sito Web, tali linee-guida è importante perché la conformità ad esse garantisce l'accessibilità del sito attraverso tutti i dispositivi che siano essi stessi conformi alle rilevanti norme tecniche. Dalla precedente caratterizzazione dell'oggetto di interesse del WD, consegue immediatamente la rilevanza ad esso delle Raccomandazioni del W3C. La rapidità del ritmo di evoluzione di queste norme pone una sfida non indifferente ai progettisti di siti Web, che devono aggiornare con opportuna frequenza il loro bagaglio tecnico, se vogliono trarre beneficio da rilevanti innovazioni

tecnologiche senza per questo rischiare di compromettere l'accessibilità dei siti stessi. Metodologie di progetto *pensate* per l'evoluzione del prodotto a fronte di mutamenti di requisiti ambientali, includendo in questi i requisiti di conformità a rilevante normativa tecnica, sono una tradizionale area di interesse di SE, e le tecniche ideate a questo scopo costituiscono dunque una potenziale riserva di soluzioni a problemi di WD.

1.5. Problemi ed esercizi

1.5.1. Soluzioni di ingegneria del software a problemi di Web design

Per rendere più palpabili gli argomenti addotti sinora, proponiamo in questa sezione conclusiva del capitolo alcuni esempi di problemi di *Web design* , tutti in qualche modo connessi alla sua caratteristica di rapida evolutività. Sulla scorta di quanto argomentato sinora, asseriamo che l'applicazione dei principi di SE, proposti nella seconda sezione sopra, risulta utile all'individuazione e/o valutazione di soluzioni ai problemi proposti, però ci asteniamo dal mostrare la validità di questa asserzione.

Attenzione!

Non si richiede di proporre soluzioni ai problemi di WD che seguono. Per ciascuno dei problemi presentati, invece, si propone un *doppio esercizio* :

1. identificazione di principi di SE rilevanti al problema
2. argomentazione delle rispettive applicazioni di tali principi alla valutazione di soluzioni al problema

1.5.2. Problema 1: novità del sito

Per siti Web la cui gestione richiede frequenti cambiamenti, è utile all'utente disporre di informazione sugli aggiornamenti del sito, ovvero in risposta alla domanda: *che c'è di nuovo?* Individuare e valutare almeno due soluzioni diverse al problema di come offrire questo servizio.

1.5.3. Problema 2: mappa del sito

La *struttura navigazionale* di un sito Web è rappresentata dalla *mappa del sito* , cioè un grafo i cui nodi rappresentano pagine del sito e gli archi (orientati) rappresentano collegamenti fra le pagine. Per siti di dimensioni molto grandi, la rappresentazione grafica della mappa del sito in una unica immagine può essere problematica. Individuare e valutare rappresentazioni alternative.

1.5.4. Problema 3: modifiche da utenti del sito

Nei siti Web in cui si vuole che l'utente sia non solo destinatario di informazione ma anche *fonte di informazione* , si pone il problema di come offrire questa opportunità evitando però che la struttura del sito cambi continuamente ed in modi imprevedibili. Individuare e valutare soluzioni al problema.

1.6. Ringraziamenti

1. Per il reperimento della Figura 1.1 si ringrazia Anna Valeria Guazzieri, partecipante al Corso Master in Tecnologie e Metodologie della Formazione in Rete dell'Università di Verona, A.A. 2001-02.

1.7. Bibliografia

- **Bowen, J.** (2005). *Formal Methods Virtual Library*. Web: http://vl.fmnet.info.
- **FME** (2006). *Formal Methods Europe*. Web: http://www.fme.org.
- **Fuggetta, A., et al.** (2006). *WebBook di Ingegneria del Software*. Web: http://webbook.cefriel.it.
- **R.S. Pressman & Associates, Inc.** (2006). *Software Engineering Resources*. Web: http://www.rspa.com/spi.
- **W3C** (2006). *World Wide Web Consortium*. Web: http://www.w3.org.
- **Hewett, T. et al.**, 2004. Chapter 2: Human-Computer Interaction. *In: ACM SIGCHI Curricula for Human-Computer Interaction*. ACM SIGCHI.
  Web: http://sigchi.org/cdg/cdg2.html
- **Ghezzi, C., Jazayeri, M. & Mandrioli, D.**, 2004. *Ingegneria del software - Fondamenti e principi*. Addison-Wesley, Pearson Education Italia.
  Web: http://hpe.pearsoned.it
- **Parnas, D.L.**, 1972. On the criteria to be used in decomposing systems into modules. *Comm. of the ACM*, 15, 1053-1058.
- **Binato, A., Fuggetta, A. & Sfardini, L.**, 2006. *Ingegneria del software: creatività e metodo*. Addison-Wesley, Pearson Education Italia.
  Web: http://hpe.pearsoned.it
- **Pressman, R.S.**, 2004. *Principi di Ingegneria del software, 4/Ed.*. McGraw-Hill Italia.
  Web: http://www.mcgraw-hill.it
- **Sommerville, I.**, 2005. *Ingegneria del software, Settima Ed.*. Addison-Wesley, Pearson Education Italia.
  Web: http://hpe.pearsoned.it
- **Van Vliet, H.**, 2000. *Software Engineering - Principles and Practice, 2nd Ed.*. John Wiley & Sons.
  Web: http://www.wiley.co.uk/vanvliet