Dipartimento di Scienze Economiche, Matematiche e Statistiche

Università degli Studi di Foggia

La progettazione di un sistema distribuito orientato alla didattica

Crescenzio Gallo e Michelangelo De Bonis

Quaderno n. 12/2008

"Esemplare fuori commercio per il deposito legale agli effetti della legge 15 aprile 2004 n. 106"

Quaderno riprodotto al
Dipartimento di Scienze Economiche, Matematiche e Statistiche
nel mese di settembre 2008 e
depositato ai sensi di legge

Dipartimento di Scienze Economiche, Matematiche e Statistiche, Largo Papa Giovanni Paolo II, 1,
71100 Foggia (Italy), Phone +39 0881-75.37.30, Fax +39 0881-77.56.16

La progettazione di un sistema distribuito orientato alla didattica

Crescenzio Gallo
c.gallo@unifg.it
Dip.to di Scienze Economiche, Matematiche e Statistiche - Università di Foggia
Largo Papa Giovanni Paolo II, 1 - 71100 Foggia (Italy)

Michelangelo De Bonis
michelangelodebonis@gmail.com

Sommario

Nel corso degli anni l'enorme e sempre crescente disponibilità di informazioni, soprattutto con l'avvento di Internet, e le innumerevoli innovazioni nell'ambiente software hanno portato la produzione di applicazioni multimediali ad essere una strategia educativa efficiente ed efficace.

Le informazioni didattiche accessibili attraverso il Web riguardano esercitazioni, soluzioni, richieste di aiuto basate sostanzialmente su due fattori: riuscire a rendere accattivanti gli obiettivi del corso in modo che l'allievo non si annoi ed abbandoni l'utilizzo del prodotto come mezzo didattico; l'uso del prodotto non deve pesare sul carico di lavoro dello studente e determinare l'abbandono del prodotto stesso.

Con questi presupposti si sono analizzate e definite nel presente lavoro delle linee guida per l'analisi di un prodotto multimediale che consenta all'utente la fruizione in rete di sistemi didattici.

Indice

Introduzione **2**

1 Modelli di progettazione **3**
1.1 Il modello sequenziale lineare 4
1.2 Il modello prototipale 7
1.3 Il modello incrementale 9
1.4 Architettura software: navigation schema 11

2 Progettazione dell'interfaccia uomo-macchina **12**
2.1 Progettazione dell'interfaccia utente 12

3 Analisi dei requisiti di applicazioni multimediali: procedure ed alternative **13**
3.1 Incrementare l'interazione 14
3.1.1 Laboratori Virtuali 14
3.1.2 Strategia di gioco 15
3.1.3 Uso guidato del software di simulazione 15
3.2 Virtual Reality e prime conclusioni 15
3.3 Uso degli strumenti tecnologici di rendimento nell'istruzione . 16
3.4 Analisi tecnica di fattibilità: Internet 16
3.5 Esempio di un sistema didattico e suoi componenti software . 17
3.5.1 Processo "*di accesso*" 17
3.5.2 Processo *browser* 18
3.5.3 Processo *spooler* 19
3.5.4 Processo *server* 20

4 Conclusioni **21**

Bibliografia **22**

Elenco delle figure

1 Schema del modello *sequenziale lineare* 4
2 Curve dei "guasti" per il software 6
3 Schema del modello *prototipale* 7
4 Schema del modello *incrementale* 10
5 Schema della struttura generale del sistema 18

Introduzione

Le applicazioni multimediali devono includere interfacce grafiche, effetti sonori, narrazione, video e devono proporre un alto livello di interazione con l'utente. La multimedialità efficace unisce sia i benefici di uno studio individuale sia i benefici di un lavoro guidato in laboratorio: in questo modo si dà all'utente la possibilità di gestire il flusso delle informazioni. Il software didattico deve presentare attività didattiche strutturate, linee guida di approfondimento e molta flessibilità; in questo modo l'allievo sperimenta, esplora e impara di più.

Con questi presupposti in questo lavoro sono state esaminate le linee guida per l'analisi di un prodotto multimediale che dia modo all'utente di poter effettuare le esercitazioni, ma anche lezioni e laboratori, in rete. Si è anche sviluppata l'analisi di un progetto che raccolga le informazioni relative a una serie di corsi e che in modo multimediale ed interattivo comunichi con l'allievo.

È noto che gli allievi imparano meglio ciò su cui si esercitano. Infatti è in laboratorio che si rispettano i concetti ed i principi insegnati e li si acquisiscono. I laboratori fisici sono spesso molto costosi e richiedono frequente manutenzione ed aggiornamenti per soddisfare le aspettative degli alunni e sfruttare al meglio le innovazioni che l'hardware ed il software propongono. I laboratori virtuali in rete forniscono molti benefici educativi ed economici. Queste tipologie di laboratori virtuali non sono ricostruzione di laboratori reali, ma sono usati o per complementarli, e/o per fornire agli allievi la possibilità di effettuare esperimenti quando i laboratori reali non sono disponibili o non è possibile il loro utilizzo.

Oltre che ad un'analisi dei requisiti, si è dato risalto anche all'analisi della progettazione dell'interfaccia uomo-macchina. Un prodotto software può essere funzionale, innovativo, pieno di contenuti ma se non ha un'interfaccia facile, intuitiva, che invogli all'uso, è destinato a soccombere all concorrenza che esiste in questo settore, sempre in crescita e sempre in continua evoluzione.

1 Modelli di progettazione

In questo paragrafo si analizzano i modelli fondamentali dell'ingegneria del software per la progettazione di un processo software: modello sequenziale lineare, modello prototipale e il modello incrementale, navigation schema.

1.1 Il modello sequenziale lineare

Nella figura 1 è illustrato il modello *sequenziale lineare* per l'ingegneria del software. Tale modello, talvolta denominato *ciclo di vita classico* o *modello a cascata*, suggerisce una strategia di sviluppo del software sistematica e sequenziale, che prende le mosse a livello di sistema e procede attraverso analisi, progettazione, codifica collaudo e la manutenzione. Modellato sulla scorta del tradizionale ciclo ingegneristico, il modello sequenziale lineare comprende le seguenti attività:

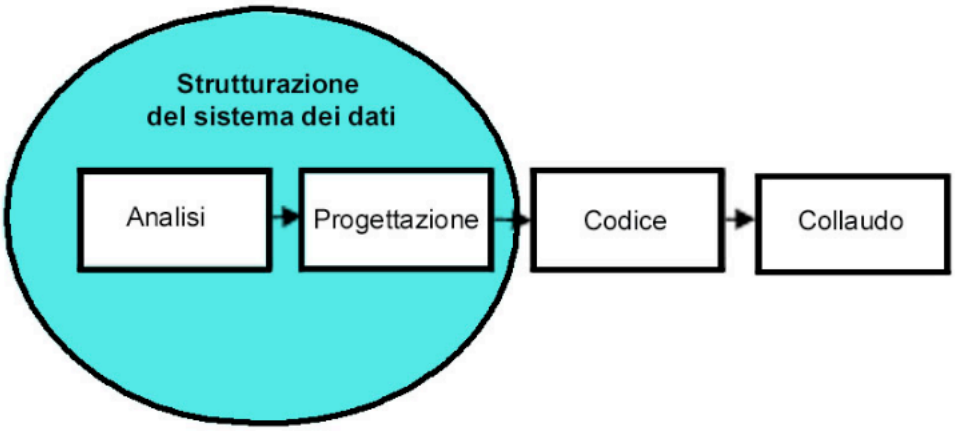

Figura 1: Schema del modello *sequenziale lineare*

Strutturazione e modellazione del sistema e dei dati. Poiché il software fa sempre parte di un sistema più vasto, il lavoro comincia dalla determinazione dei requisiti per tutti gli elementi del sistema e dalla successiva attribuzione al software di una parte di requisiti. Questa considerazione a livello di sistema è essenziale nei casi in cui il software è a contatto con elementi diversi, come hardware, persone e basi dati. La strutturazione e l'analisi del sistema comportano la raccolta dei requisiti a livello sistema, con una parte ridotta di analisi e progettazione ad alto livello.

Analisi dei requisiti del software. La raccolta dei requisiti si intensifica e si concentra sul software. Al fine di comprendere la natura dei programmi da costruire, il progettista deve comprendere il dominio delle informazioni per il software, così come la funzionalità, il comportamento, le prestazioni e le interfacce richieste. I requisiti, sia del sistema sia del software, vanno documentati e riveduti insieme con il cliente.

Progettazione. La progettazione del software è di fatto un processo in più fasi, focalizzato su quattro distinti attributi di un programma: la struttura dei dati, l'architettura software, le interfacce e i dettagli procedurali (gli algoritmi). Il processo di progettazione traduce i requisiti in una rappresentazione del software della quale si possa valutare la qualità prima che cominci la stesura del codice. Come per i requisiti, anche la progettazione è documentata ed entra a far parte della configurazione del software.

Generazione del codice. Il progetto deve essere tradotto in una forma leggibile da una macchina. Questo compito si svolge nella fase di generazione del codice. Se la progettazione è stata condotta fino ai dettagli la stesura del codice può svolgersi in modo più o meno automatico.

Collaudo. Una volta generato il codice, comincia il collaudo dei programmi. Il collaudo si concentra sugli aspetti logici interni del software, al fine di garantire che tutte le istruzioni siano provate, e sulle funzionalità esterne, al fine di scoprire eventuali errori e di accertarsi che, a fronte di dati di input specifici, vengano prodotti i risultati previsti.

Manutenzione. Qualsiasi prodotto software (con l'eccezione del software *embedded*) è soggetto a modifiche dopo la consegna al cliente. Le modifiche si rendono necessarie o perché si sono scoperti errori o per adeguare il software a mutate condizioni esterne (ad esempio un nuovo sistema operativo, una nuova periferica) o perché il cliente richiede migliorie funzionali o prestazionali. Nella manutenzione si applicano tutte le fasi precedenti ad un programma esistente anziché ad uno nuovo. Purtroppo il lavoro di manutenzione tende a deteriorare il software. Infatti a mano a mano che si apportano le modifiche, è probabile che si introducano nuovi difetti, provocando i picchi della frequenza di guasti visibili nella figura 2. Prima che la curva ritorni alla regione stabile, vengono richieste nuove modifiche, che producono un nuovo picco. Così la frequenza minima di guasti tende a salire lentamente: il software si deteriora a causa delle modifiche.

Il modello sequenziale lineare è il più vecchio e il più diffuso tra i paradigmi dell'ingegneria del software. Tuttavia, le critiche di cui è stato fatto oggetto hanno spinto anche i suoi sostenitori a metterne in dubbio l'efficacia [1]. Fra i problemi che si incontrano nella sua applicazione ne citiamo quattro:

1. *Un progetto reale si conforma raramente allo schema sequenziale del modello.* Poiché il modello sequenziale può incorporare solo indiretta-

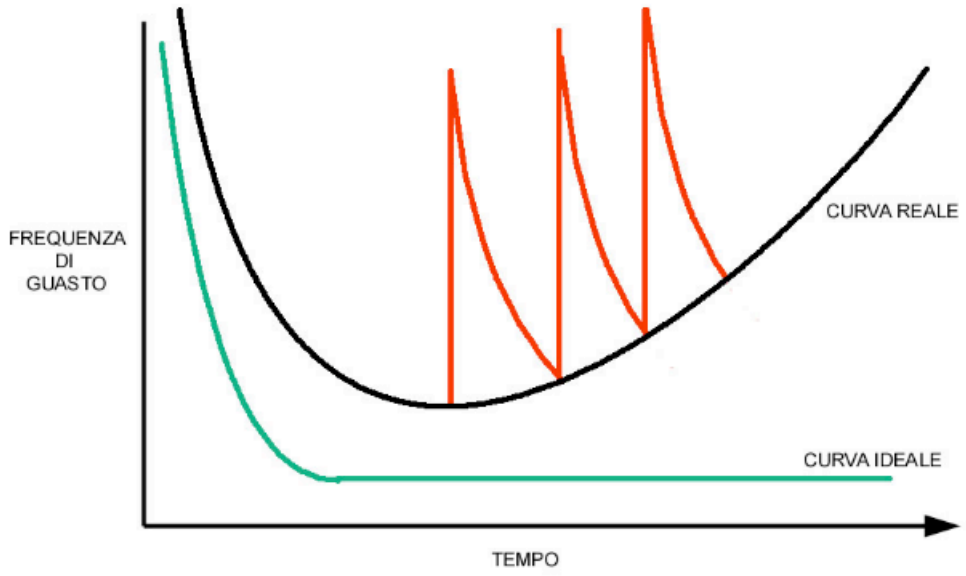

Figura 2: Curve dei "guasti" per il software

mente la nozione di iterazione, ogni modifica è causa di confusione a mano a mano che il progetto avanza.

2. *Spesso è difficile per il cliente enunciare esplicitamente tutti i requisiti.* Il modello sequenziale lo esige e non è in grado di governare in modo adeguato l'incertezza che si accompagna naturalmente al principio di un progetto.

3. *Il cliente deve essere paziente.* Solo verso la fine dell'arco temporale del progetto si genera una versione funzionante dei programmi. Un errore importante può avere conseguenze disastrose se non viene rilevato fino alla revisione del programma in funzione.

4. *Gli sviluppatori restano inoperosi senza motivo.* In una interessante analisi di progetti reali, si scoprì che la natura lineare del ciclo di vita tradizionale conduce a "stati bloccanti" [2], nei quali alcuni membri del team devono attendere che i loro colleghi completino una certa attività. Il tempo passato in attesa può addirittura superare quello passato lavorando effettivamente! Gli stati bloccanti tendono a prodursi soprattutto al principio e alla fine di un processo sequenziale lineare.

Ciascuno di questi problemi si incontra effettivamente nella realtà. Tuttavia il paradigma del ciclo di vita tradizionale occupa un posto ben definito

e importante nella prassi dell'ingegneria del software. Esso fornisce uno schema di riferimento nel quale collocare i metodi per l'analisi, la progettazione, la stesura del codice, il collaudo e la manutenzione. Il ciclo di vita tradizionale è tuttora il modello di processo più diffuso. Con tutti i suoi difetti, è comunque molto meglio di una strategia totalmente casuale per lo sviluppo di software.

1.2 Il modello prototipale

Spesso il cliente stabilisce gli scopi generali da assegnare al software, ma non specifica in modo dettagliato i requisiti relativi all'input, all'elaborazione e all'output. In altri casi, lo sviluppatore può nutrire dei dubbi sull'efficacia di un algoritmo, sull'adeguatezza di un sistema operativo o sulla forma da dare all'interfaccia uomo-macchina. In queste e altre situazioni, la soluzione migliore può venire dal paradigma prototipale.

Il paradigma prototipale, raffigurato nel suo schema di principio nella figura 3, prende le mosse dalla raccolta dei requisiti.

Figura 3: Schema del modello *prototipale*

Sviluppatore e cliente si incontrano per definire gli obiettivi complessivi

del software, individuare i requisiti che si possono già precisare e delineare le aree che richiederanno ulteriori definizioni. Si procede quindi a una progettazione rapida, focalizzata sugli aspetti del software che saranno visibili al cliente-utente (ad esempio l'inserimento di dati e i formati di output). Di qui si giunge alla realizzazione di un prototipo, il quale viene valutato dal cliente-utente e sfruttato per raffinare i requisiti del prodotto definitivo. L'iter si ripete, via via ritoccando il prototipo, in modo da soddisfare le esigenze del cliente e allo stesso tempo dare allo sviluppatore una visione più precisa del problema.

In teoria il prototipo ha lo scopo di individuare i requisiti del software. Nella costruzione di un prototipo funzionante, lo sviluppatore tende a servirsi di frammenti di programmi esistenti e di strumenti (ad esempio generatori di resoconti, gestori di finestre e così via) che gli permettano di generare rapidamente programmi operativi.

Ma che fare del prototipo dopo che il suo scopo è stato raggiunto?

> *Nella maggior parte dei progetti il sistema realizzato per primo è a malapena utilizzabile. Può risultare troppo lento, troppo grande, di uso complicato, o tutte e tre le cose. Non c'è altro da fare che ricominciare da capo, facendo tesoro dell'esperienza, e realizzare una nuova versione, nella quale i problemi siano risolti [...]. Quando si applica una nuova concezione o una nuova tecnologia, è inevitabile costruire un sistema pilota destinato a essere cestinato, perché anche con la migliore pianificazione non si può prevedere tutto e centrare il bersaglio al primo colpo. La questione non è dunque se costruire un sistema pilota e poi gettarlo. Ciò è inevitabile. La questione è se pianificare in anticipo di costruire un oggetto destinato a essere gettato, oppure promettere al cliente di consegnargli quell'oggetto [...].*

Il prototipo può svolgere il ruolo di primo sistema, quello che Brooks [3] raccomanda di gettare. Ma questa è una visione idealizzata. È vero che il paradigma prototipale piace sia agli sviluppatori sia ai clienti: l'utente può farsi un'idea del prodotto finale e lo sviluppatore ha modo di costruire rapidamente un oggetto funzionante. D'altra parte, la prototipazione non manca di problemi: eccone alcuni.

1. Il cliente si trova di fronte ciò che appare come una versione funzionante del software, senza sapere che il prototipo è tenuto insieme "con lo scotch" e che per realizzarlo in fretta si sono trascurate la qualità intrinseca del software e la manutenibilità a lungo termine. Quando

viene informato che il prodotto va costruito ex novo per assicurare una alta qualità, il cliente comincia a strepitare e a esigere che si apportino "pochi ritocchi" al prototipo e lo si trasformi in un prodotto pronto all'uso.

2. Spesso lo sviluppatore, allo scopo di rendere operativo il prototipo in tempi brevi, ricorre a compromessi: a volte si serve di un sistema operativo o di un linguaggio di programmazione inadeguato solo perché è a disposizione ed è noto; a volte implementa un algoritmo inefficiente solo per esibirne le funzionalità. Con il passare del tempo, lo sviluppatore tende a dimenticare i motivi che rendono inadeguate tali scelte, le quali, sebbene tutt'altro che ottimali, diventano parte integrante del sistema.

Sebbene sia soggetta ad alcuni problemi, la prototipazione è un efficace paradigma di ingegneria del software. Il segreto è stabilire le regole del gioco dall'inizio: il cliente e lo sviluppatore devono concordare sulla natura del prototipo come meccanismo per la definizione dei requisiti, destinato ad essere scartato (almeno in parte), mentre il software definitivo va progettato e realizzato secondo criteri di qualità e manutenibilità.

1.3 Il modello incrementale

Prende sempre più corpo la consapevolezza che il software, come ogni sistema complesso, si evolve nel tempo [4]. Spesso i requisiti aziendali e di prodotto mutano nel corso stesso dello sviluppo, rendendo irrealistica l'immagine di un cammino rettilineo verso il prodotto finale: i rigidi vincoli di tempo imposti dal mercato rendono impossibile il completamento di un prodotto software di ampio respiro, e tuttavia è necessario allestirne una versione ridotta per rispondere alla pressione competitiva e aziendale; esiste un nucleo di requisiti di sistema o di prodotto ben precisati, ma i dettagli delle estensioni non sono ancora stati definiti. In situazioni di questo genere, l'ingegnere del software ha bisogno di un modello di processo destinato specificatamente ad un prodotto che si evolve nel tempo.

Il modello sequenziale lineare è rivolto a uno sviluppo rettilineo. Nella sostanza, la strategia a cascata assume che, una volta condotta a termine la sequenza lineare, venga consegnato un sistema completo. Il modello prototipale è rivolto ad assistere il cliente (o lo sviluppatore) nella comprensione dei requisiti. In generale, esso non è destinato allo sviluppo di sistemi commerciali. Entrambi questi paradigmi tradizionali ignorano la natura evolutiva del software.

I modelli evolutivi sono iterativi. Le loro caratteristiche permettono al progettista di sviluppare versioni sempre più complete di un prodotto software.

Il modello incrementale combina aspetti del modello sequenziale lineare (applicato ripetutamente) con la filosofia iterativa della prototipazione. Come illustrato nella figura 4, il modello incrementale consiste nell'applicare più sequenze lineari, scalate nel tempo. Ogni sequenza lineare produce uno "stadio" operativo del software [5].

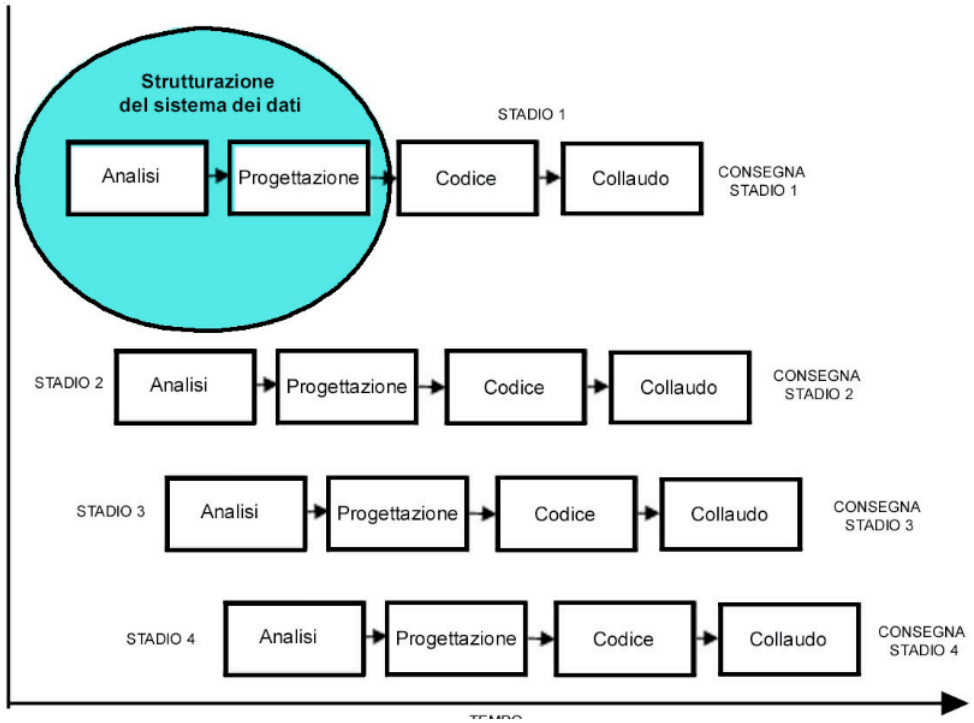

Figura 4: Schema del modello *incrementale*

Nel modello incrementale, il primo stadio consiste in un *prodotto in nuce*, cioè un prodotto che soddisfa i requisiti fondamentali, tralasciando diverse caratteristiche supplementari (alcune note, altre ancora sconosciute). Questo prodotto viene saggiato dal cliente (o sottoposto ad una revisione approfondita). In seguito alla valutazione del cliente, si stende un piano per lo stadio successivo. Tale piano deve curarsi di due aspetti: le modifiche del prodotto tese a soddisfare meglio le esigenze del cliente e l'inserimento di nuove funzioni. Il processo si ripete a mano a mano che si completano i vari stadi, fino a giungere al prodotto finale.

Il modello di processo incrementale, come quello prototipale e altre strategie di tipo evolutivo, è essenzialmente iterativo. Diversamente dal modello prototipale, quello incrementale si caratterizza per la produzione di un programma operativo ad ogni stadio.

I primi stadi sono versioni ridotte del prodotto finale, nondimeno offrono funzionalità utili al cliente e fungono da piattaforma per la valutazione. Lo sviluppo incrementale è particolarmente utile quando la disponibilità di personale è insufficiente a garantire l'implementazione completa nei termini di tempo stabiliti per il progetto. La realizzazione dei primi stadi può essere affidata ad un piccolo team. Se il *prodotto in nuce* riceve un'accoglienza positiva, si può accrescere il team che deve implementare lo stadio successivo.

Si può inoltre pianificare la sequenza degli stadi così da controllare i rischi di natura tecnica. Un sistema complesso, ad esempio, può richiedere la disponibilità di nuovo hardware in corso di sviluppo, la cui data di consegna è incerta. Si può allora pianificare la sequenza degli stadi in modo che i primi non richiedano il nuovo hardware; l'utente finale potrà così disporre di parte della funzionalità, evitando eccessivi ritardi.

1.4 Architettura software: navigation schema

I sistemi e le applicazioni Web sono differenti da molte altre categorie di software. Le principali differenze possono essere riassunte se si considera che i sistemi Web comprendono vari elementi di diversa natura (marketing, informazione, etc.) [7]. Nella maggior parte delle applicazioni Web sono presenti i seguenti attributi:

Uso intensivo della rete. Per sua natura un'applicazione Web sfrutta intensamente la rete.

Attenzione ai contenuti. In molti casi, la funzione principale di un'applicazione Web consiste nell'impiegare dei mezzi ipermediali per presentare testi, grafica, audio e video.

Continua evoluzione. A differenza del software applicativo convenzionale che evolve secondo una serie di versioni ben pianificate e temporalmente spaziate, le applicazioni Web evolvono continuamente.

Immediatezza. Essendo il tempo di realizzazione di applicazioni Web molto breve gli sviluppatori devono usare metodi di analisi, pianificazione ed implementazione adatti a tali applicazioni [8].

Sicurezza. Poiché le applicazioni Web vengono impiegate tramite rete, è molto difficile limitare gli utenti che avranno accesso all'applicazione.

Per proteggere i contenuti più delicati e fornire modalità più sicure occorre implementare forti misure di sicurezza sia nell'infrastruttura che supporta l'applicazione Web sia all'interno dell'applicazione stessa.

Estetica. È innegabile che gran parte della popolarità delle applicazioni Web è dovuta all'aspetto.

Risulta chiaro che intervenendo tutti questi fattori il modo di progettare e di formalizzare il processo software per il Web è cambiato. Infatti i modelli classici non rispondono a quelle che sono le esigenze sopracitate e ancora: velocità di progettazione, facilità d'uso, facilità di comprensione, ridotte dimensioni dei pacchetti software. Oggi il più accreditato metodo di formalizzazione di un prodotto destinato ad interagire con il Web è il *navigation schema*. Esso risulta essere un metodo facile, veloce ed intuitivo. Veloce perché è intrinsicamente nato per offrire la riduzione del tempo di progettazione e di collaudo delle interfacce. Intuitivo perché orientato totalmente alla grafica, alla visione degli oggetti, risulta essere intuitivo e facile seguire la percezione del sistema dell'utente e la relativa immagine del progettista. Tutto questo e molto altro hanno fatto sì che a tutt'oggi il *navigation schema* fosse il miglior metodo di progettazione e di formalizzazione [9] [10].

2 Progettazione dell'interfaccia uomo-macchina

2.1 Progettazione dell'interfaccia utente

Nella prefazione del suo libro sulla progettazione delle interfacce utente, Ben Shneiderman [6] afferma:

> *Frustrazione e tensione sono il pane quotidiano di molti utenti di sistemi informatici, costretti ad apprendere faticosamente i comandi e i menù che in teoria dovrebbero aiutarli. Alcuni sperimentano un tale grado di angoscia, terrore o nevrosi che finiscono per evitare del tutto l'uso del calcolatore.*

I problemi ai quali allude Shneiderman sono reali. Ci siamo imbattuti tutti in interfacce difficili da apprendere e utilizzare, generatrici di confusione, che non perdonano gli errori e, in molti casi, fonte di frustrazione. D'altra parte, qualcuno ha speso tempo ed energia per costruire quelle interfacce ed è probabile che il suo scopo cosciente non fosse quello di suscitare tanti problemi.

La progettazione delle interfacce utente ha molto a che fare con lo studio dell'essere umano, oltre che con questioni tecnologiche. Chi è l'utente? Come

apprende a interagire con un nuovo sistema informatico? Come interpreta le informazioni prodotte dal sistema? Queste sono alcune delle domande che ci si deve porre nel progettare un'interfaccia utente.

Il processo di progettazione dell'interfaccia utente comincia dalla creazione di modelli diversi di funzionalità del sistema (com'è percepita dall'esterno). Si devono quindi delineare le attività necessarie per realizzare tale funzionalità, sia da parte dell'utente sia da parte del calcolatore, si devono considerare questioni comuni alla progettazione di qualsiasi interfaccia, mediante appositi strumenti si crea un prototipo e quindi si implementa il modello, infine si valuta la qualità del risultato.

Ma per progettare correttamente l'interfaccia utente bisogna capire come funziona l'apprendimento dell'uomo; comprendere la struttura principale della memoria e i capisaldi del suo funzionamento; la capacità di percezione ed attenzione dell'uomo; le interfacce grafiche e le metafore costruire per ricondurre l'utente ad affrontare nuove esperienze con vecchi metodi di risoluzione.

3 Analisi dei requisiti di applicazioni multimediali: procedure ed alternative

Recenti innovazioni nel software per la creazione di pacchetti hanno reso la produzione di applicazioni multimediali una strategia educativa efficiente ed efficace. Tuttavia, ci sono delle scelte da effettuare quando si scelgono metodi multimediali, invece che metodi più convenzionali.

I componenti base di applicazioni multimediali includono un'interfaccia grafica, effetti sonori, narrazione e video. Per essere efficace, le applicazioni multimediali devono contenere questi elementi ed effettuare un livello elevato di interazione con l'utente. La multimedialità efficace unisce sia i benefici dello studio indipendente dell'utente, sia i benefici del lavoro guidato nel laboratorio permettendo così che l'utente gestisca il tasso a cui le informazioni devono essere presentate nel corso della lezione. Il programma deve presentare all'utente le attività strutturate come linea guida di approfondimento e deve offrire anche flessibilità permettendo, così, che *l'allievo sperimenti, esplori e impari di più.* I moduli di multimedialità dovrebbero includere la navigazione completa nel progetto multimediale per contribuire al collegamento tra soggetti relativi e poter così seguire il progresso suggerito con la lezione.

La preparazione e la produzione di moduli efficaci di multimedialità interattiva rappresentano i compromessi critici fra i requisiti competenti. Per

aumentare l'interattività si vuole spesso fornire la navigazione completa, lunghi brani di esposizione video e rintracciare le prestazioni dell'allievo. Anche se l'implementazione di queste idee è certamente utile, i requisiti del software e dell'hardware possono essere proibitivi.

Il punto più importante nello sviluppare un'applicazione multimediale è nel progettare il contenuto dei suoi moduli. Esiste una stretta interazione fra la flessibilità e l'interattività d'una lezione e il tempo e le risorse che prende per produrla. *Gli oggetti devono essere considerati con attenzione per evitare di sprecare risorse in concetti che sono appresi facilmente usando i metodi convenzionali.* L'attenzione, preferibilmente, dovrebbe essere posta su una visualizzazione dell'astratto e degli oggetti altamente matematici, sulla partecipazione interattiva agli esperimenti in laboratorio e sulla presentazione di applicazioni pratiche.

Il passo successivo nello sviluppare un'applicazione multimediale è produrre moduli multimediali nei quali includere interfaccia grafica, suoni e video. E' importante, inoltre, considerare l'impatto di un modulo nel contesto della struttura del corso. Le risorse richieste per aggiungere una "novità" ad una lezione è funzione della qualità e della complessità della "novità". Per esempio il fatto che un video di appena 1 minuto richieda alcuni megabyte di spazio suggerisce l'idea che lo stream-video debba essere usato con moderazione. Una volta che la "novità" multimediale è sviluppata, il prossimo passo è prepararla per la piattaforma sulla quale dovrà essere utilizzata. Questo implica una decisione su quale formato sia il migliore basandosi sulle prestazioni, sulla qualità, sul transfer rate dei dati, e sullo spazio disponibile sull'hard disk [11].

3.1 Incrementare l'interazione

Forse il punto cruciale nello sviluppo di moduli di multimedialità nell'educazione è il livello di interazione effettuato nel modulo prodotto. Le risorse sono utilizzate nel migliore dei modi in laboratori virtuali o insieme alle lezioni a distanza. La multimedialità aumenta l'uso del software di simulazione e permette l'uso delle strategie di gioco motivando e stimolando gli allievi che imparano ed aumentano la loro comprensione.

3.1.1 Laboratori Virtuali

È noto che gli allievi imparano meglio ciò in cui si esercitano. L'uso degli esperimenti di laboratorio è essenziale nel fare rispettare i concetti ed i principi insegnati e renderli solidi. L'istituzione dei laboratori fisici è spesso costosa, richiede frequente manutenzione e aggiornamenti per venire a

contatto con le aspettative degli allievi e per approfittare delle ultime possibilità dell'hardware e del software. La costruzione dei laboratori virtuali attraverso la multimedialità fornisce benefici educativi inestimabili. Ciò è particolarmente importante nelle zone dove l'istituzione dei laboratori reali è costosa. Particolare risalto va al fatto che i laboratori virtuali non sono considerati ricostruzione di quelli reali. Sono usati per complementare i laboratori reali e per fornire agli allievi le possibilità di esperimento quando i laboratori reali non sono disponibili oppure non sono possibili.

3.1.2 Strategia di gioco

La strategia di gioco fornisce ancora un'altra procedura utile per l'aumento dell'interattività. I giochi educativi sviluppati con attenzione possono fornire la discussione fra gli allievi e stimolare la revisione dei concetti appresi.

3.1.3 Uso guidato del software di simulazione

L'uso del calcolatore come strumento pedagogico nella scienza, nella matematica e nell'ingegneria è prevalentemente per le simulazioni e per la rappresentazione dei dati. Integrare nel software i tool usati negli studi tecnici è molto importante per la formazione scientifica, poiché gli allievi potranno usare i programmi per risolvere problemi e progettazione ingegneristica che possono incontrare in avvenire, nella loro formazione o nella professione scelta. Ulteriormente, il software di simulazione permette che l'allievo osservi i fenomeni non fisicamente visibili o difficilmente riproducibili.

Lo svantaggio di usare pacchetti di simulazione è che sono abbastanza complessi. *Il tempo di acquisire padronanza dell'interfaccia di simulazione riduce il tempo usato per l'apprendimento.*

3.2 Virtual Reality e prime conclusioni

Lo sviluppo dei moduli di multimedialità è il risultato della necessità di realizzare un compromesso accettabile fra l'interazione, la flessibilità nella navigazione, la quantità di video, la richiesta di memoria e la richiesta minima di velocità del calcolatore. Le procedure per la creazione dell'interazione in moduli multimediali sono state ugualmente discusse. Si è osservato che l'implementazione dei laboratori virtuali, la strategia di gioco e l'uso guidato del software di simulazione forniscono un livello attraente di interattività.

Per concludere, gli sforzi più recenti sono rivolti alla tecnologia della *Virtual Reality* per fornire un'interfaccia umana 3D ed una navigazione in tempo

reale negli ambienti virtuali. La possibilità di usare questa tecnologia per sviluppare moduli educativi è tutta da verificare nel successo e nell'efficacia di quei prodotti multimediali che in futuro la implementeranno.

3.3 Uso degli strumenti tecnologici di rendimento nell'istruzione

Il Prof. Morrison in suo articolo [12] cita la sua esperienza di insegnamento con le nuove tecnologie. Dal suo punto di vista il corso risultava di maggior comprensione, i suoi allievi hanno migliorato le loro competenze ed hanno prodotto lavori molto credibili. Dal punto di vista degli allievi, le valutazioni di corso erano basse. Perché valutazioni di successo hanno corrisposto a scarso interesse? La sua analisi è che era venuto a mancare in due funzioni: (1) non aveva venduto sufficientemente gli obiettivi del corso, quelli che abilità scritte ed orali del professore riescono a trasmettere nel corso delle lezioni; e (2) non aveva compensato il fatto che questi requisiti aggiunti avevano aumentato sostanzialmente il carico di lavoro dello studente. Gli allievi non avevano motivazioni per imparare da soli e spendere così tanto tempo ed energia.

3.4 Analisi tecnica di fattibilità: Internet

Nell'analisi precedente si è mostrato come sia essenziale effettuare una progettazione totale dello sviluppo dei moduli multimediali. Similmente si può effettuare una analisi di fattibilità più incentrata alla progettazione ed ingegnerizzazione di un sito web che abbia gli obiettivi di multimedialità e di didattica. Nello sviluppo dei moduli multimediali possono essere di aiuto sistemi presenti sul mercato che effettuano integrazione ed authoring più o meno automatico. Ma l'ambiente Internet rende insufficienti ed inutilizzabili molti di questi tool perché necessita di moduli più flessibili e di dimensioni più ridotte rispetto a quelli che tipicamente vengono utilizzati nello sviluppo di moduli multimediali adatti, per esempio, nei CD/DVD-ROM multimediali.

Da analisi effettuate e ricerche intraprese in questo senso si è delineato sempre più un taglio netto tra due aree di soluzioni:

- progettare e realizzare un "reader" proprietario;
- utilizzare tool software già esistenti.

La tabella 1 elenca vantaggi e svantaggi di entrambe le soluzioni.

Dall'analisi dei vantaggi e svantaggi elencati nella tabella 1 risulta chiara che la scelta migliore sarebbe una soluzione ibrida (*in medio stat virtus*) che

Tabella 1: Vantaggi e svantaggi delle soluzioni di implementazione

	Realizzazione di un "Reader"		Utilizzo di strumenti del mondo Web
:-)	Definizione di interfaccia grafica univoca per ogni browser	:-(	Interfaccia mutevole, dipendente dalle impostazioni dei browser
:-)	Realizzato in linguaggio multipiattaforma risulta indipendente dalla piattaforma di utenza	:-)	Sviluppo in Java Script, HTML, DHTML
:-(	Complessità notevole nella progettazione e realizzazione	:-)	Più facile nella realizzazione

inglobi i vantaggi delle due opzioni, sopra elencate, in base agli obiettivi che necessita raggiungere. Ricordando l'esperienza citata da Morrison [12] gli obiettivi da raggiungere devono tenere conto anche di due fattori essenziali: (a) riuscire a rendere accattivante gli obiettivi del corso; (b) non pesare sul carico di lavoro dello studente. Quest'ultimo punto implica progettare moduli multimediali flessibili e "facili" in modo che l'allievo non pensi che imparare sia dispendio enorme di tempo ed energia.

3.5 Esempio di un sistema didattico e suoi componenti software

Di seguito sono indicate le linee guida delle caratteristiche che i vari processi software debbano possedere per essere conformi al modello del sistema. La figura 5 mostra le interazioni possibili in un macro-esempio di sistema multimediale tra i vari processi software client/server.

3.5.1 Processo "*di accesso*"

Il processo software denominato di accesso è il ponte che collega le varie applicazioni utente con il server. Questo modo di modellare l'architettura permette facilità di estensione future. Ecco l'analisi di alcuni attributi.

- Dovendo sempre passare per il modulo di accesso il server sa in ogni momento chi è che effettua l'interazione con il server o la base dati.

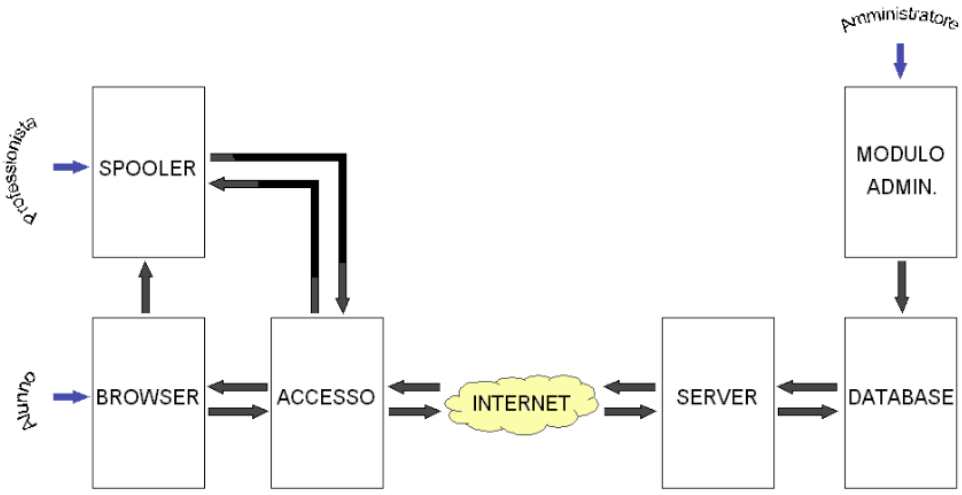

Figura 5: Schema della struttura generale del sistema

- Una delle *feature* possibili che il server può fornire è quella di possedere una coda di job elaborati con il tool richiesto dall'utente e da inviare allo spooler dell'utente. Il dover collegarsi al server e autenticarsi da la possibilità al server di avere l'indirizzo IP della macchina a cui trasparentemente inviare i risultati.

- Se il server non riesce a sopportare il carico di lavoro per le troppe connessioni richieste può disconnettere l'utente che non esegue richieste da molto tempo.

- Si può bloccare l'accesso ad un utente che si identifica in modo errato per un numero di volte consecutivo.

Queste sono solo alcuni dei vantaggi nell'utilizzare il modulo di accesso come "filtro" di accesso alla rete.

3.5.2 Processo *browser*

Se il modulo di accesso è il cuore del client, il browser è l'apparato sensoriale. Esso è un browser con cui è possibile collegarsi, attraverso il modulo di accesso, al server per poter interrogare la base di dati.

Il browser presenterà delle caratteristiche peculiari al suo scopo. Una barra di strumenti, aggiuntiva al programma classico, è lo strumento che metterà l'utente in grado di utilizzare le potenzialità dello strumento software. Questa barra conterrà pulsanti quali: riferimenti teorici, esercizi simili, prossimo

esercizio, richiesta soluzione, tool, aiuto, contatti. Ad ognuno di questi pulsanti si dovrà provvedere ad una analisi ed evoluzione del processo in risposta ai vari input. Attraverso l'utilizzo di un browser già presente in commercio e la progettazione un processo software ex novo che oltre alle classiche caratteristiche dei browser inglobi caratteristiche ad uso del progetto permette di utilizzare per il testo e la risoluzione degli esercizi semplici pagine web, ed inoltre questa soluzione permette di poter sviluppare feature senza doversi preoccupare se l'utente possieda gli strumenti adatti al supporto delle novità introdotte.

3.5.3 Processo *spooler*

Il processo software dello spooler di elaborazione sorge da una duplice esigenza. La prima è la necessità di rendere il più possibile multiprocesso il client. La seconda dalla percezione che l'utente ha di "fare la fila". Questo concetto tende a realizzare metafore del mondo reale per la gestione di interazioni con il computer in modo complesso. Lo spooler prende in consegna i file che si vogliono far processare con il tool presente sul server. Quindi li mette in coda, secondo la visione classica che l'utente percepisce di "fare la fila". In verità lo spooler prende il file e lo comunica al server dove effettivamente i file vengono accodati per l'utilizzo dei tool.

Lo spooler registra lo stato del job in base alle segnalazioni del server. Se il file è stato inoltrato al server ed il server lo ha inserito correttamente in coda, lo stato iniziale è quello di attesa, quando il processo viene consegnato al tool per l'elaborazione lo spooler cambia stato al job dichiarandolo in *elaborazione*. Quando il file è stato elaborato correttamente il server invia, allo spooler, una segnalazione di fine elaborazione e il file elaborato. Lo spooler riceve il file elaborato, lo aggiunge alla lista e definisce lo stato *finito*. Se qualunque operazione dovesse dare un problema nella successione dei vari stati si passerebbe allo stato *fallito*.

Oltre al nome del job ed allo stato in cui si trova lo spooler gestisce altri due importanti attributi. Il primo è quello ovvio del tool da utilizzare per l'elaborazione del file. I tool possono essere di diversa natura; sarà il server, attraverso il modulo d'accesso a decidere se l'utente è abilitato o meno all'utilizzo di quel determinato tool. Se si è in possesso di un tool locale è possibile specificare il tool locale ed utilizzarlo per le proprie elaborazioni. Il secondo attributo è quello indicato nello spooler come "richiesta input". Questo attributo apre al mondo dello spooler condizionale. Infatti se il *tag* della richiesta di input è impostato, cioè si desidera avere richiesta input, quando lo spooler finirà di processare il job, si fermerà ed attenderà che l'utente riavvii il processo di spooling dei file ancora in coda. Questo modo

operativo permette di decidere se processare o meno i job in coda in base alla visione dei risultati del job su cui è impostata la richiesta di input.

3.5.4 Processo *server*

Il processo server è il processo software che utilizza le interazioni del client, gestisce le interrogazioni con la base di dati e gestisce i job da elaborare. Il server oltre a interagire con la base di dati, deve anche preoccuparsi di tenere una coda di job in ingresso da elaborare e una coda di job elaborati in uscita. Ci sono alcune caratteristiche fondamentali:

- La coda di output contiene i job che sono stati elaborati ma non sono ancora stati inviati al loro proprietario, perché – ad esempio – l'utente ha mandato il job da elaborare e poi si è disconnesso dal server. Per inviare il job elaborato si potevano utilizzare due soluzioni. La prima è quella di utilizzare la posta elettronica. Il server finito di elaborare un job invia il file elaborato all'indirizzo mail dell'utente. Il secondo è aspettare che l'utente si ricolleghi al server ed inviare automaticamente il job elaborato. Questa soluzione presenta, però, il problema dell'indirizzo IP dell'utente. Infatti quando un utente si collega ad Internet da casa o dall'ufficio, chiede al provider la connessione ed il provider automaticamente e dinamicamente gli assegna un indirizzo IP tra i suoi disponibili. Cioè l'indirizzo IP dell'utente è dinamico, cambia, non è fisso. Questo problema è risolto dall'utilizzo del modulo di accesso. Infatti un utente per collegarsi al server deve passare attraverso il modulo di accesso. Quest'ultimo nel protocollo di autenticazione e riconoscimento con il server comunicherà anche l'indirizzo IP dinamico che possiede la macchina di lavoro.

- La coda in ingresso sarà gestita secondo i tool, cioè ci saranno tante code quante sono i tool disponibili, così che non ci siano tempi morti di attesa per l'elaborazione di job non relativi a tool di una determinata materia. Nella coda non vi è nessun meccanismo di priorità.

- Il server gestisce l'interfaccia con la base di dati; esso si preoccuperà di effettuare tutte le operazioni di interrogazione e di costruire il pacchetto di risposta al client. L'implementazione di una relazione mutuamente esclusiva comporta in generale la ricerca, nella base di dati, di un record che soddisfa certi criteri. Se nessun record li soddisfa, si passa a un'altra serie di criteri. Il traffico di rete risulta minimo quando l'applicazione che pilota i criteri di ricerca è interamente sul server. La prima trasmissione in rete dal client verso il server contiene in questo

caso i parametri corrispondenti ai criteri primari a quelli alternativi. Il server determina se si deve ricorrere ai criteri alternativi. Il messaggio di risposta al client contiene il record uscito dalla ricerca secondo i criteri primari o secondari. La soluzione alternativa, nella quale il client stabilisce se ricorrere ai criteri alternativi, comporta, nel caso peggiore, un messaggio per la prima ricerca, un messaggio di "record non trovato", un messaggio con i criteri alternativi e la risposta definitiva. Se, in media, si ricorre ai criteri secondari nel 50% dei casi, la soluzione che assegna al server la responsabilità di avviare la seconda ricerca riduce sensibilmente il traffico di rete.

4 Conclusioni

Nell'esaminare i meccanismi usati nello sviluppo dei moduli multimediali si è enfatizzata la necessità di realizzare un compromesso accettabile tra l'interattività e la flessibilità di navigazione, tra la quantità di video e la quantità di memoria del computer richiesta, tra la velocità e la multimedialità dei contenuti.

Le linee guida nella definizione di procedure per creare l'interattività di moduli multimediali sono state accostate a tecniche, in cui la realizzazione di laboratori virtuali, sono oggi il fulcro di sistemi didattici utilizzati in rete. Ma questo non è il punto di arrivo. Infatti oggi esistono già indicazioni di ricerche con lo sviluppo di didattica online e di gestione dei laboratori che usano tecnologia *Virtual Reality* per offrire una interfaccia grafica "umana" in 3D e una navigazione in tempo reale in ambienti virtuali.

La direzione indicata dalla tecnologia del VR è il prossimo step a cui far riferimento negli studi successivi: la sua praticabilità, l'analisi dei compromessi tra nuova tecnologia e velocità di navigazione, efficacia del prodotto multimediale e analisi dei costi per il successo di questa nuova tecnologia nella didattica sono solo alcuni dei punti che devono ancora essere analizzati, vagliati e progettati.

Riferimenti bibliografici

[1] Hanna M., *Farewell to waterfalls*, in Software Magazine, Maggio 1995.

[2] Badrac M., Perry D., Votta L., *Prototyping a process monitoring experiment*, in IEEE Trans. Software Engineering, Ottobre 1994.

[3] Brooks F., The mythical man-month, Addison-Wesley, 1975.

[4] Gilb T., Principles of software engineering management, Addison-Wesley, 1988.

[5] McDermid J., Rook P., *Software development process models*, in Software engineer's reference book, CRC Press, 1993.

[6] Shneiderman Ben, Designing the user interface: strategies for effective human-computer interaction, Addison-Wesley, 1987.

[7] Powell T. A., Web site engineering, Prentice Hall, 1988.

[8] Nrton K., *Applying cross functional evolutionary methodologies to web development*, in Proc. First ICSE Workshop on Web Engineering, ACM, Maggio 1999.

[9] Bruno Giorgio, Model-based software engineering, Chapman and Hall, 1995.

[10] Pressman Roger S., Principi di ingegneria del software, McGraw-Hill, Luglio 2000.

[11] Iskander M. F. et al., *Development of multimedia modules for education*, in Computer applications in engineering education, 1995.

[12] Morrison J. L., *Using technology productivity tools in teaching: One professor's odyssey*, 1997.