"Learning bricks": oggetti riusabili per simulazioni efficaci

Vindice Deplano, consulente di e-learning – v.deplano@tin.it

Abstract

Le simulazioni costituiscono un metodo formativo particolarmente efficace, ma la loro penetrazione è fortemente limitata, oltre che da vincoli culturali, da fattori legati ai costi elevati che il mercato stenta ad accettare.

I *Learning bricks* sono un'architettura che punta a ingegnerizzare lo sviluppo di simulazioni interattive, abbassandone drasticamente costi e tempi di sviluppo senza limitarne la qualità. L'intera interfaccia della simulazione (personaggi, oggetti animati, testi, prove di autovalutazione, dialoghi, ecc.) viene costruita con un'operazione di montaggio non troppo dissimile da quella con cui si usano i mattoncini Lego. L'architettura è composta da un modulo base, che contiene le specifiche del motore della simulazione, e da un certo numero di *bricks*: moduli con funzioni specializzate, completamente parametrizzabili per quanto riguarda il contenuto informativo (testo, immagini, audio, video, animazioni), il collegamento con il motore della simulazione, il comportamento.

1 La simulazione: prospettive e problemi

Oggi quell'insieme di metodi, tecnologie e attività che chiamiamo "e-learning" ha conquistato un posto stabile nel mondo della formazione. Aziende, pubblica amministrazione e scuola hanno capito che è una via per raggiungere esperienze di apprendimento realmente efficaci, che non si potrebbero ottenere in presenza. E non solo per tagliare i costi.

Dove i progetti di e-learning si sono appiattiti imitando vecchi modelli, hanno prodotto risultati deludenti e utenti (e committenti) insoddisfatti. È accaduto fin troppo spesso con i sistemi autodidattici, dove l'utente dovrebbe imparare qualcosa interagendo con un software: la stragrande maggioranza di questi Wbt (Web Based Training) si riduce a "sfogliapagine elettronici" con il solo conforto di un po' di multimedialità (e, quando il budget lo consente, animazioni e filmati). In molti casi, si rimpiange la straordinaria facilità d'uso di un buon libro.

Per realizzare qualcosa di davvero utile bisogna fare di più. È qui che entrano in gioco le simulazioni interattive (o *serious games*), che permettono di integrare l'esperienza diretta nel processo di apprendimento. Vale la pena di ribadire che una simulazione computerizzata non è un pallido surrogato virtuale della realtà "vera", ma produce esperienze di apprendimento per molti versi "migliori", perché aiuta a giocare col mondo, interpretandone le dinamiche, ripetendo più volte gli stessi passaggi, riflettendo per tutto il tempo necessario sulle proprie azioni. E senza correre rischi (meglio, molto meglio, far fallire aziende e distruggere aerei virtuali).

Ci si potrebbe chiedere perché queste meraviglie didattiche sono ancora così poco diffuse nei progetti di e-learning e perché il mercato non le reclama a gran voce invece di ricorrere a modesti sfogliapagine elettronici.

I motivi, a mio avviso, sono due:

- manca una diffusa cultura della simulazione nei clienti (che spesso la vedono come un gioco) e anche nei produttori;
- le simulazioni hanno un costo molto elevato che il mercato, abituato da molti anni a prodotti spesso mediocri ma economici, non riconosce.

Oggi però la domanda di materiali didattici di qualità sta crescendo. Una crescita che va aiutata producendo vere simulazioni interattive a un prezzo non molto diverso da quello dei comuni Wbt sequenziali.

2 Ottimizzare lo sviluppo delle simulazioni

I "*Learning bricks*" vogliono essere un contributo in questa direzione. Costituiscono, nello stesso tempo, un'architettura software e un modo di pensare alle simulazioni che mira a ridurne drasticamente i costi di sviluppo senza porre alcun limite al livello qualitativo.

Dobbiamo chiederci, a questo punto, quali sono gli elementi costitutivi (e, quindi, le voci di costo) di una generica simulazione.

2.1 Struttura di una simulazione

Una simulazione è un sistema software che consente di interagire con un mondo virtuale che astrae alcuni aspetti significativi della realtà. Significa che questo mondo virtuale:

1. comunica con l'utilizzatore attraverso un'interfaccia grafica/sonora e alcune periferiche (mouse, tastiera e altri dispositivi di input);
2. gli consente di effettuare determinate scelte (selezionare opzioni, inserire valori, ecc.);
3. reagisce alle scelte in un modo che è sovrapponibile al mondo reale.

Non è certo la definizione più esauriente di simulazione, ma è utile per metterne in evidenza i due elementi costitutivi, implementati nel software:

1. Il "motore", che comprende:
    - Gli aspetti della realtà che si vogliono prendere in considerazione (trasformati in un insieme di variabili).
    - Le regole di funzionamento del mondo virtuale (in concreto, è il modo con cui le variabili sono messe in relazione tra loro).
    - Il modello di simulazione, ovvero il tipo di logica che sottende alla definizione delle regole di funzionamento. Dipende dalle teorie a cui il progettista fa riferimento e dalle tecnologie disponibili. Da questo punto di vista, distinguiamo alberi decisionali, automi cellulari, reti bayesiane, dinamica dei sistemi, reti neurali, ecc.

2. L'interfaccia, che comprende:
   - Il set di comandi e di dati che il sistema è in grado di riconoscere e interpretare.
   - La metafora comunicativa, quell'insieme di elementi grafici e funzionali che definiscono il modo con cui la realtà "virtuale" della simulazione viene presentata: cruscotto, scrivania, ambiente a due o tre dimensioni, ecc.
   - La forma con cui, all'interno della metafora, viene rappresentato lo stato del sistema: stringhe alfanumeriche, immagini, parole, suoni, filmati, animazioni più o meno complesse.

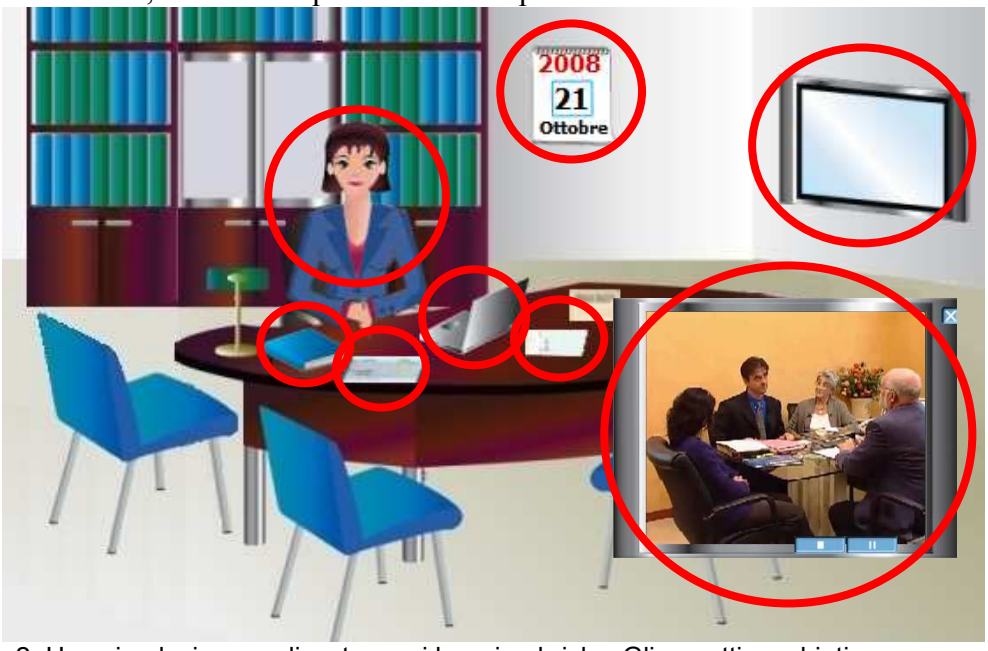

Fig. 2. Una simulazione realizzata con i learning bricks. Gli oggetti cerchiati corrispondono ad altrettanti *bricks*.

Non esiste necessariamente una corrispondenza diretta tra complessità del motore e interfaccia della simulazione. Tipicamente, le simulazioni costruite a scopo di ricerca tendono ad avere motori molto raffinati (perché cercano una corrispondenza molto forte con il segmento di realtà che viene simulato) e interfacce spartane. Al contrario, quelle usate per la formazione possono avere motori semplici, ma l'interfaccia deve essere sofisticata. Magari come un videogioco, risorse permettendo…

2.2 Una questione di costi e tempi

Un corollario di questa situazione è che nelle simulazioni formative il costo di sviluppo di un'interfaccia di qualità (e, in generale, del software) è molto più alto di quello necessario per progettare la logica del motore.

Visto da chi giorno per giorno si scontra con le esigenze di aziende e pubblica amministrazione, il problema ha due facce:

1. Il costo delle simulazioni di buon livello, vale la pena di ripeterlo, è molto più alto di quanto i clienti sono generalmente disposti a spendere.
2. Le risorse impiegate per creare l'interfaccia accattivante richiesta dal cliente non aggiungono nulla alla qualità della simulazione, cioè alla sua capacità di far apprendere con esperienze credibili e coinvolgenti.

I *Learning bricks* puntano a ridurre drasticamente i costi e i tempi di sviluppo dell'interfaccia. Con tre obiettivi:

1. mercato: migliorare la penetrazione delle simulazioni, riducendone i costi in termini assoluti;
2. qualità: liberare risorse per la progettazione del "motore";
3. gestione dei progetti: facilitare la prototipazione e quindi la comunicazione con il cliente.

3 Architettura dei *Learning bricks*

L'idea di base è concettualmente semplice: realizzare simulazioni a partire da oggetti riusabili e parametrizzabili (i *bricks*), facilmente combinabili tra loro, che gestiscano l'interfaccia con grafica, multimedialità, animazioni. Come i mattoncini Lego con cui produciamo velocemente costruzioni complesse montando un numero finito di oggetti semplici.

L'architettura dei *Learning bricks* si compone di due elementi:

1. il modulo base;
2. i *bricks*.

3.1 Modulo base

Il modulo base è il "direttore d'orchestra", in quanto incorpora:

- La **funzione di calcolo** che implementa il motore della simulazione. Il motore può essere di diverso tipo, poiché qualunque modello di simulazione è riducibile a una serie di calcoli che hanno per oggetto variabili e relazioni tra variabili. La funzione è richiamata ogni volta che lo stato del mondo simulato deve essere ricalcolato in seguito a un evento: scorrere del tempo, decisione dell'utente, esito di una prova di verifica, ecc.
- Il **caricamento dei *bricks*** necessari alla gestione dell'interfaccia, con il relativo passaggio di parametri.
- Il **caricamento delle immagini di sfondo** che consentono di realizzare simulazioni con più ambienti (per passare da un ambiente all'altro si usa un particolare *brick* "Porta"- vedi oltre).
- Alcune **funzioni di utilità** generale, richiamabili da tutti i *bricks*.

Le operazioni di sviluppo della simulazione che comportano la scrittura di righe di codice si limitano a inserire nel modulo base:

- le istruzioni contenute nella funzione di calcolo;
- i parametri dei *bricks* utilizzati.

Il modulo base e i *bricks* sono sviluppati in Flash, esclusivamente tramite ActionScript. Chi aprisse i file sorgente troverebbe un solo fotogramma sulla linea temporale e nessun elemento precaricato sullo stage e in libreria.

3.2 I bricks

I *bricks* sono moduli software che hanno:

- una **funzione specifica** nell'interfaccia della simulazione;
- un **collegamento** con una qualunque delle variabili che determina il funzionamento del motore (*variabile di riferimento*);
- un **contenuto comunicativo** (testo, immagini, audio, video, animazioni) personalizzabile tramite *parametri di configurazione*;
- una **comportamento** personalizzabile nei limiti previsti dalla funzione specifica.

Per descrivere meglio lo scopo e il funzionamento di un qualunque *brick*, usiamo come esempio il "Personaggio":

- Funzione.
  Parlare (o cantare, se si vuole), con un duplice movimento animato: occhi aperti e chiusi (sempre), bocca aperta e chiusa (in concomitanza con il parlato).
- Collegamento con la variabile di riferimento.
  In funzione del valore della variabile (o di un intervallo di valori), può pronunciare diversi commenti, assumere una determinata postura o cambiare posizione nello schermo.
- Contenuto comunicativo.
  I parametri di configurazione comprendono un elenco di file sonori e un elenco di file grafici (posture del personaggio).
- Comportamento.
  A seconda del valore della variabile di riferimento, il personaggio può essere attivato o disattivato, comparire o scomparire (anche con effetti di dissolvenza).

Se sono necessari più personaggi, il *brick* "Personaggio" può essere chiamato più volte con parametri di configurazione diversi.

La Tabella 1 presenta un elenco dei *bricks* già realizzati.

4 Uso dei *Learning bricks*

È importante notare che i *bricks* sono moduli totalmente indipendenti l'uno dall'altro, perché si collegano esclusivamente con le variabili che fanno

parte della funzione di calcolo (cioè del motore). Così è possibile inserire un numero illimitato di oggetti animati per immettere dati, visualizzare lo stato delle variabili, effettuare test di autovalutazione, presentare informazioni sotto forma di schede o mini Wbt animati, ecc.

Tabella 1

I *BRICKS*

Nome	Funzione	Particolarità
Personaggio	Parlare o emettere suoni (attivando file mp3) diversificati in funzione del valore della variabile di riferimento.	Possibilità di cambiare postura o posizione nello schermo.
Tutoriale	Presentare schede informative con testo e immagini in movimento e audio sincronizzato.	Si attiva cliccando su un oggetto sensibile.
Tutoriale esterno	Presentare Wbt (con navigazione avanti/indietro) con pagine costituite da animazioni Flash (swf).	Si attiva cliccando su un oggetto sensibile.
Input	Immettere da tastiera valori da assegnare a una variabile.	Si attiva cliccando su un oggetto sensibile.
Test	Somministrare domande di autovalutazione con feedback e assegnazione di punteggi.	Si attiva cliccando su un oggetto sensibile.
Testo	Presentare testi diversificati in funzione del valore della variabile di riferimento.	Si attiva cliccando su un oggetto sensibile.
Siparietto	Presentare brevi testi animati diversificati in funzione del valore della variabile di riferimento.	Si attiva automaticamente.
Video	Presentare filmati (formato flv) diversificati in funzione del valore della variabile di riferimento.	Si attiva cliccando su un oggetto sensibile.
Personaggio video	Presentare oggetti animati 3D o filmati con audio diversificati in funzione del valore della variabile di riferimento.	Utilizzando filmati in chroma key è possibile inserire personaggi umani nell'ambiente.
Dialogo	Presentare un dialogo tra più personaggi, con audio e didascalie.	È' possibile cambiare inquadratura a ogni battuta del dialogo. I personaggi presentano animazioni negli occhi e, quando parlano, nella bocca.
Calendario	Presentare la data (giorno, mese, anno).	
Porta	Consentire il passaggio da un ambiente all'altro cliccando su un'immagine	Consente di realizzare simulazioni con più ambienti.

4.1 Sviluppo di simulazioni

Usando i *Learning bricks*, lo sviluppo di una simulazione, di qualunque complessità, prevede quattro tipi di operazioni (vedi lo schema di figura 2):

1. Definizione di un sistema logico-matematico (variabili e loro relazioni) che ricalchino il funzionamento di alcuni aspetti del mondo reale, con criteri che variano in funzione del modello di simulazione prescelto.

2. Traduzione del sistema in codice ActionScript all'interno della funzione di calcolo.
3. Sviluppo o selezione da una libreria di testi, elementi grafici (sfondi, oggetti, pulsanti) e componenti multimediali (audio, clip video, ecc.).
4. Richiamo dei *bricks* necessari all'interfaccia utente e definizione dei relativi parametri, inserendoli direttamente in un file ActionScript e/o in file XML esterni.

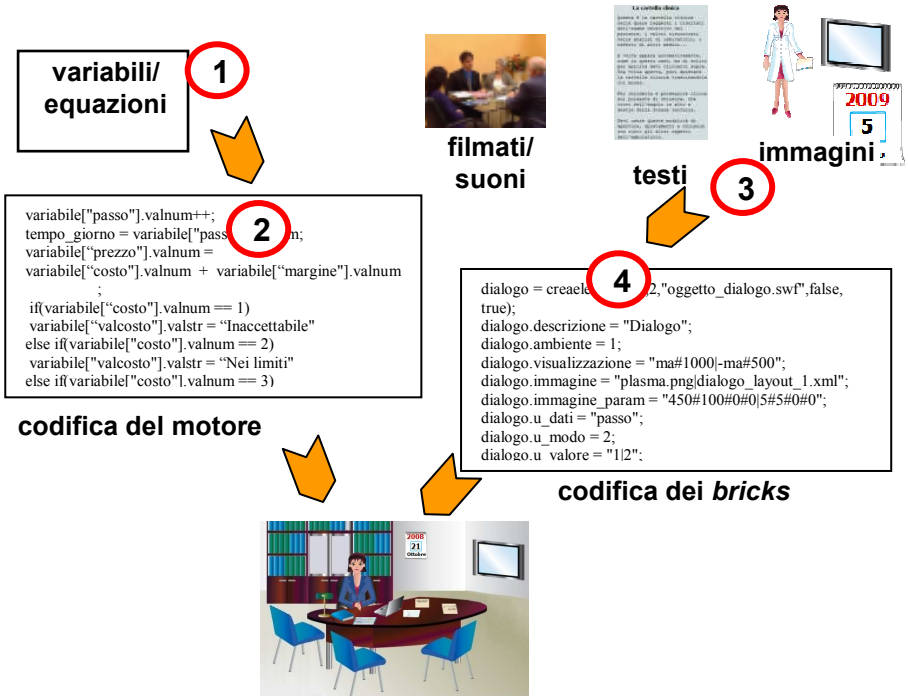

Fig. 2. Schema dello sviluppo di una simulazione

In concreto, le operazioni 1 e 3 si svolgono con modalità tradizionali, perché precedono la fase di implementazione del modello e del montaggio. Il tempo di realizzazione può essere anche molto lungo quando vengono progettate simulazioni complesse con decine o centinaia di variabili.

L'operazione 2 consiste nella traduzione del modello logico matematico in istruzioni ActionScript. Richiede un operatore con conoscenze di base del linguaggio e un tempo variabile da qualche decina di minuti ad alcune ore. Si tratta di un'operazione che necessita di attenzione, ma non risulta particolarmente complessa.

L'operazione 4 è determinante per l'interfaccia, l'interattività e quindi l'esperienza del fruitore. Ciascun *brick* è inserito attraverso una serie di istruzioni del tipo:

oggetto.parametro = valore.

A titolo di esempio, la tabella 2 elenca i parametri necessari per l'oggetto "Video", che può gestire uno o più videoclip.

Tabella 2

PARAMETRI DELL'OGGETTO "VIDEO"

Parametro	Funzione	Tipo di dato
ricalcolo	Richiamo automatico della funzione di calcolo alla chiusura dell'oggetto.	true/false
inibizione oggetti	Inibisce l'attivazione di altri oggetti fino alla chiusura del video.	true/false
descrizione	Testo del tooltip che compare sfiorando l'oggetto col mouse.	testo
ambiente	"Stanza" in cui l'oggetto è presente	codice numerico
visualizzazione	Effetti (dissolvenza, movimento, ecc.) con cui l'oggetto si apre e si chiude.	codici alfanumerici
immagini	File relativi all'oggetto cliccabile e allo sfondo del video aperto.	nomi di file
parametri immagini	Posizione sullo schermo e dimensioni delle due immagini.	codici numerici
pulsanti	File immagine relativi ai pulsanti: chiudi, play, pausa, stop, ecc.	nomi di file
parametri pulsanti	Posizione e dimensioni dei file relativi ai pulsanti.	codici numerici
variabile	Variabile di riferimento che determina il comportamento dell'oggetto	nome di variabile
valore	Valori significativi della variabile di riferimento che determinano la comparsa di un particolare filmato e/o l'attivazione automatica del video	dati alfanumerici
contenuto	Elenco di file relativi ai filmati.	nomi di file
parametri contenuto	Posizione e dimensioni del filmato in relazione allo sfondo del video.	codici numerici
titoli	Elenco dei titoli dei filmati	testo
parametri titolo	Stile del titolo: carattere, dimensioni, grassetto, ecc.	codici alfanumerici
comandi	Azioni associate a valori della variabile di riferimento: attivazione del filmato, scomparsa o disabilitazione dell'oggetto cliccabile.	codici alfanumerici
suoni	Effetti sonori associati all'avvio del film, alla sua chiusura, ecc.	codici numerici

La maggior parte dei parametri, quelli che descrivono gli aspetti estetici e i comportamenti di base valgono per tutti gli oggetti "Video" della simulazione (o di una collana di simulazioni con interfaccia grafica simile).

Gli oggetti più complessi presentano meno parametri, perché gli aspetti dinamici del comportamento sono descritti da file XML esterni. Per esempio, l'oggetto "Test", che presenta una batteria di domande, richiede per ciascuna di esse un file con:

- testo della domanda;
- opzioni di risposta;
- risposta esatta;
- testo della soluzione;
- testo del feedback per risposta esatta;
- testo dei feedback per le diverse risposte errate.

Un programmatore è in grado di costruire uno scenario nuovo con una decina di *bricks* in un tempo non superiore alle due ore. Il tempo si riduce notevolmente (anche della metà) se lo scenario presenta oggetti già usati in precedenza.

4.2 Competenza

Al momento l'uso dei *Learning bricks* richiede la competenza di un programmatore junior o comunque un addestramento sufficiente a lavorare direttamente nel codice.

Per i non programmatori (e per gli stessi progettisti delle simulazioni) sarebbe utile un editor con un interfaccia amichevole che permetta di implementare il motore e richiamare i *bricks* senza programmare.

Ovviamente, se una particolare simulazione richiede funzioni non presenti in nessuno dei *bricks* disponibili, è necessario un programmatore senior per svilupparne uno *ex novo*. *Brick* che da quel momento sarà disponibile per i progetti futuri, facilitando il recupero dell'investimento. La libreria di "mattoni" disponibili è, quindi, destinata a crescere nel tempo.

4.3 Simulazioni e altri scenari

I *Learning bricks* sono stati pensati per simulazioni il cui motore è costituito da equazioni sul modello della dinamica dei sistemi (Senge, 1990). Un esempio di questo genere di simulazioni, realizzato con metodi tradizionali, è *Impresa oltre i confini* (Deplano, 2004).

Tuttavia, questa architettura si può applicare anche a:

- Simulazioni con motori basati su alberi decisionali o di tipo probabilistico (al momento non sembra adattabile facilmente ad automi cellulari e reti neurali).
- Learning objects più semplici, costruiti attorno alla narrazione di una storia (qui la variabile guida del comportamento degli oggetti è il tempo).
- Learning objects dalla classica struttura sequenziale. Qui, ovviamente, i *Learning bricks* non aggiungono nulla a una metodologia didattica non particolarmente evoluta, ma permettono di ottenere prodotti più articolati e gradevoli per l'utente.

5 Learning bricks e sistemi autore

Escludendo i moltissimi editor di Wbt e/o di test di valutazione, è possibile creare velocemente simulazioni usando alcune categorie di strumenti disponibili sul mercato:

1. Registratori di schermo, che memorizzano schermate, movimenti di cursore, digitazioni, ecc. con l'aggiunta di audio, testi, prove di valutazione, ecc. Sono utilissimi per simulare i software applicativi. Il più famoso è *Captivate* di Adobe.
2. Strumenti destinati a simulazioni settoriali: geometria, circuiti elettronici, ecc. Un esempio è l'ottimo *Geogebra*.
3. Editor di simulazioni all'interno di schemi predefiniti (per esempio dialoghi con un interlocutore virtuale su determinati argomenti). Un esempio è stato realizzato all'interno del progetto europeo *SiSiNe*.

I *Learning bricks*, tuttavia, non sono nati da un modello già noto. Per questo non è possibile fare confronti con questi sistemi autore.

6 Conclusioni

I *Learning bricks* hanno l'obiettivo immediato di abbassare i costi delle simulazioni interattive. Alcuni progetti in corso stanno dimostrando la possibilità di ridurre drasticamente la complessità del lavoro. Ma il fine ultimo è un altro: facilitare quel salto metodologico che porti a fare delle simulazioni interattive una componente essenziale di qualunque sistema autodidattico realmente efficace.

Per questo, sarà necessario che il sistema evolva continuamente:

1. realizzando nuovi *bricks* e ottimizzando di quelli esistenti con l'aggiunta di funzioni evolute;
2. sviluppando un editor per non programmatori;
3. creando di simulazioni accessibili, secondo quanto richiesto dalla legge 4/2004.

Questa è probabilmente la sfida più complessa, che però è un passaggio obbligato per poter impiegare vere simulazioni nella pubblica amministrazione e nella scuola.

Bibliografia

Deplano V. (2003), *Come valutare i materiali didattici nei progetti di e-learning?* in: Isfol, *La qualità dell'e-learning nella formazione continua*.

Deplano V. (2004), *Come con Lara Croft*, E-learning & Knowledge management, 2.

Geogebra. URL: http:// www.geogebra.org (verificato il 10 agosto 2009).

Impresa oltre i confini, URL http://www.vindice.it/demo/impresa_oltre_confini/main.htm (verificato il 10 agosto 2009).

Parisi D. (2001), *Simulazioni*, Il Mulino.

Senge P. (1990), *The Fifth Discipline*, Doubleday.

Sisine Project, URL: http://www.nac.unina.it/sisine/ (verificato il 10 agosto 2009).