E-learning by doing, un approccio basato su tecniche di ragionamento su azioni

M. Baldoni, C. Baroglio, B. Demo, V. Patti, L. Torasso(*)
{baldoni,barbara,baroglio,patti}@di.unito.it
(*)laura.torasso@realemutua.it

Dipartimento di Informatica — Università degli Studi di Torino
C.so Svizzera, 185 — I-10149 Torino (Italy)

(*)Società Reale Mutua di Assicurazioni
Servizio Informatico-S.I.Ge.A.
Via Corte d'Appello 11, Torino

Abstract. In questo articolo gli autori propongono un approccio basato su tecniche AI di ragionamento su azioni per la realizzazione di sistemi in autoistruzione per l'apprendimento dell'uso di software applicativi, rivolgendo particolare attenzione alle suite da ufficio. In particolare verrà mostrato come tali tecniche consentono di implementare in modo naturale l'approccio noto come *e-learning by doing*, nel quale lo studente impara esercitandosi e sfruttando i feedback che gli giungono dal sistema.

1 Introduzione, motivazioni e contesto

Lo sviluppo delle tecnologie informatiche, trasformando il computer da strumento per iniziati a presenza familiare, economicamente alla portata di tutti, ha risvegliato un crescente interesse da parte di nuovi utilizzatori non solo in ambito aziendale ma anche domestico. Contabilità e bilancio di casa, o la semplice corrispondenza, vengono sempre più frequentemente gestiti tramite programmi quali Office, StarOffice o OpenOffice. L'utilizzo quotidiano e assiduo del calcolatore ha fatto crescere la richiesta di *corsi* finalizzati all'apprendimento dei pacchetti software di uso più frequente, causando la pubblicazione di manuali (cartacei e multimediali) ricchi di documentazione ed esempi.

Sistemi computer-based più sofisticati, che consentono ad uno studente di imparare ad utilizzare uno strumento in modo *interattivo*, esercitandosi sull'elaboratore stesso, sono invece ancora poco diffusi, soprattutto a causa dei costi di realizzazione. Questi sistemi, che implementano un paradigma noto come "e-learning by-doing", vengono adottati quasi esclusivamente per applicazioni complesse e con un elevato rischio di pericolosità: un esempio classico è il software di controllo semiautomatico della guida di aerei. In questi casi l'approccio utilizzato prevede non tanto l'uso diretto dell'applicativo, quanto la ricostruzione dell'interazione con il software da imparare e del contesto d'uso in un ambiente detto di simulazione. Un simulatore contiene un *modello* in grado di riprodurre gli

effetti delle possibili scelte dell'utente mostrandone i risultati. In contesti meno specifici della guida di veicoli (quali gli aerei) e più legati all'*uso* di software, spesso i simulatori si poggiano su ricostruzioni grafiche finalizzate a mostrare all'utente cosa accade premendo questo o quel bottone. Naturalmente affinché questa soluzione sia efficace l'ambiente di simulazione deve essere il più possibile allineato con l'applicativo reale: ogni nuova modifica di quest'ultimo deve essere aggiornata nel corrispettivo simulato. Come è intuibile, ciò comporta notevoli spese di mantenimento del software, per questo motivo l'approccio descritto non viene utilizzato per favorire l'apprendimento di software considerati "poveri" come le suite di ufficio.

Una soluzione più economica consisterebbe nell'utilizzare l'*applicativo stesso* del quale si desidera imparare l'uso in modo guidato e interattivo, secondo il paradigma dell'*e-learning by doing*. In altri termini, se l'utente si esercitasse per *obiettivi* e se fosse possibile *intercettare* le azioni che esegue mentre si esercita con l'applicativo di interesse, sarebbe possibile fornirgli dei feedback. Un esempio di soluzione sviluppata in questa direzione è il sistema di autoistruzione ELVIRA [4, 5]. Scopo di questo sistema è supportare gli utenti durante le esercitazioni di Excel; più in particolare, lo studente sfrutta un sistema computer-based, che gli consente di sperimentare le funzionalità dell'applicativo, viste a livello teorico, eseguendole in Excel stesso, senza l'ausilio di alcuna simulazione. In questo modo lo studente apprende sia durante la fase di illustrazione teorica da parte di un docente sia durate la fase di ripetizione individuale al calcolatore, nella quale familiarizza con l'ambiente reale del sistema. Nonostante queste premesse, l'obiettivo principale degli sviluppatori del sistema ELVIRA è stato la realizzazione di API per l'intercettazione delle operazioni eseguite dall'utente e per la registrazione di lezioni, quindi questo applicativo *non* realizza il paradigma di e-learning by doing nella sua accezione più ampia. Infatti nella fase di esercitazione lo studente dapprima osserva una sequenza di azioni compiute in forma dimostrativa dal sistema e quindi le ripete nello *stesso ordine* con cui sono state presentate. Lo studente non è libero nè di esplorare Excel (per esempio cliccando un bottone del cui significato non è completamente certo anche se ciò non compromette il risultato finale) nè di adottare soluzioni alternative, comunque valide ai fini del risultato dell'esercitazione, perché il sistema non è in grado di valutare la correttezza di una generica sequenza di azioni. Per questo motivo potremmo affermare che ELVIRA si basa su di un paradigma che è un caso particolare di quello che ci interessa e che potremmo chiamare "e-learning by repeating". Senza dubbio l'e-learning by repeating è di più facile implementazione ma nello stesso tempo nei sistemi che se ne avvalgono l'esercitazione risulta meno personalizzata e l'attività mnemonica che lo studente deve svolgere non facilita il suo apprendimento, soprattutto in un ambito in cui l'oggetto da conoscere è ricco di elementi grafici e intuitivi, spesso utilizzati come scorciatoia per eseguire comandi equivalenti nascosti nei menú.

Per una completa realizzazione del paradigma di e-learning by doing bisognerebbe tener conto delle esperienze dell'utente durante la prova ed essere in grado di valutarle; in tal senso sarebbe necesssario aggiungere un componente *es-*

perto al sistema, in grado di valutare il libero comportamento tenuto dall'utente durante l'esercitazione e confrontarlo con il dominio dell'applicazione. In questo lavoro proponiamo un approccio al problema nel quale si sfruttano tecniche di Intelligenza Artificiale (AI) per ragionare su azioni. Il connubbio tra e-learning e AI si è già realizzato con successo in diversi settori applicativi e, in particolare, nel contesto dei siti web adattativi applicati all'istruzione, dove nella realizzazione di Tutoring System sono state messe in luce le potenzialità di agenti in grado di adeguare il proprio comportamento all'ambiente e di comunicare con altri agenti artificiali e con agenti naturali -gli utenti umani- [3]. L'intento di questo articolo è di analizzare come *meccanismi di ragionamento* affiancati a tecniche di ricerca in uno *spazio degli stati* possano essere utili alla realizzazione di sistemi di monitoraggio e di tutoring dello studente in un contesto applicativo di computer-based training, realizzando le specifiche del paradigma *e-learning by doing*. In particolare il nostro interesse è rivolto all'applicazione di DyLOG, un linguaggio logico ad azioni già utilizzato in applicazioni Web-based educational [1, 2]. In questi lavori le capacità di ragionamento di agenti razionali sono state sfruttate per realizzare un sistema Web-based adattivo, che aiuta degli studenti universitari nella costruzione di piani di studi personalizzati per il corso di laurea triennale in informatica e effettua la verifica della correttezza di piani di studi proposti dallo studente. Per piano di studi personalizzato si intende una sequenza di corsi che consente di conseguire una specifica *competenza* di interesse; un piano di studi è corretto quando l'insieme dei prerequisiti di ciascun corso è fornito dai corsi che lo precedono o è una competenza già propria dello studente prima della formulazione del piano. Le capacità di adattamento dell'agente razionale rendono questo sistema un utile assistente software in grado di aiutare sia lo studente sia il professore nelle diverse fasi della costruzione di un piano di studi da seguire o da proporre. Questo approccio rientra anche nella filosofia dell'e-learning by doing almeno per quanto riguarda la validazione di un piano di studi, infatti ciascun utente impara a costruirsi un piano di studi personalizzato effettuando delle prove (esercitandosi) e lasciando al sistema il problema di verificare la correttezza dell'ordinamento e le competenze offerte dai corsi che vorrebbe sostenere.

2 E-learning come ragionamento su azioni

Un sistema intelligente ragiona sul proprio comportamento e adotta una certa strategia di comportamento sulla base del proprio stato mentale interno. DyLOG, come linguaggio di programmazione per la definizione di agenti razionali, consente di modellare la conoscenza sul mondo e l'evoluzione dinamica del sistema stesso tramite meccanismi di ragionamento su azioni. Nel seguito mostriamo brevemente come il dominio di conoscenza che ci interessa può essere descritto in termini di azioni che modificano la conoscenza, allo scopo di spiegare come è possibile interpretare l'e-learning by doing come ragionamento su azioni in DyLOG.

Le azioni hanno precondizioni alla loro applicazione, che possono dipendere dal mondo esterno o dallo stato interno dell'agente, e producono degli effetti. Supponiamo per esempio di volere descrivere l'operazione di "creazione di un

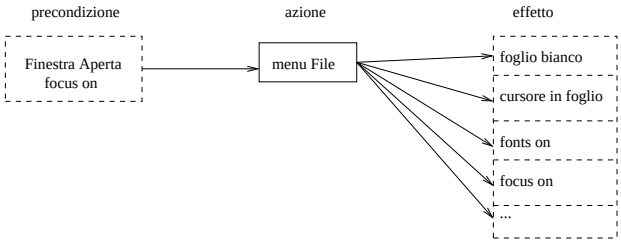

Fig. 1. Esempio di azione con precondizioni ed effetti.

nuovo documento da menú" come un'azione elementare con precondizioni e effetti che chiamiamo *menuFile*, Figura 1. Le precondizioni per creare un documento da menu sono a) che la finestra dell'applicativo sia aperta e b) che il fuoco sia su quella finestra; gli effetti sono diversi: viene creato un foglio bianco, il fuoco passa all'interno del foglio, etc. Concentriamoci ora sull'operazione più generale

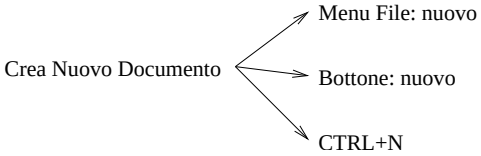

Fig. 2. L'operazione di creazione di un documento può essere effettuata tramite azioni alternative.

di "creazione di un nuovo documento" (Figura 2). Questa operazione puó essere eseguita, utilizzando un comando contenuto in uno dei menú (quindi eseguendo l'azione elementare appena descritta), ma anche in modi diversi. Ad esempio le suite per ufficio, come Office, forniscono normalmente in una delle barre degli strumenti un pulsante ad hoc che permette di portare a termine l'operazione. Naturalmente è possibile cliccare su tale bottone solo se la barra degli strumenti è attiva (precondizione); l'effetto sarà l'apertura di un foglio bianco pronto per esser riempito.

Di norma i pulsanti e le icone nelle suite da ufficio rappresentano i comandi principali per l'utilizzo delle funzionalità dell'applicativo (sono detti pulsanti ad accesso veloce); per questo motivo sono resi visibili e di facile riconoscimento, tanto che i simboli usati rimandano spesso alla funzione che l'elemento svolge (per esempio la figura del dischetto è associata alla funzione di salvataggio del file). L'utilizzo di differenti elementi grafici di accesso a uno stesso servizio è

motivata dall'esigenza di soddisfare una maggiore varietà di utenti del software. Tuttavia l'insieme dei pulsanti ad accesso veloce può essere spesso ridefinito dall'utente, che in questo modo ha occasione di *personalizzare* le diverse barre degli strumenti adattandole alle proprie esigenze lavorative.

In molte situazioni che l'utente utilizzi l'una o l'altra soluzione (pulsante o menú) è *indifferente*, però affinché la sua scelta sia indifferente anche al sistema è necessario che quest'ultimo abbia a disposizione una descrizione *astratta* delle operazioni supportate dall'applicativo. Questa descrizione manca nei sistemi che sfruttano un apprendimento "by repeating", che infatti non tengono in considerazione la pluralità di soluzioni consentita dai prodotti né le preferenze dei singoli utenti, limitandone la creatività e talvolta la comprensione dello strumento. Tale descrizione astratta dovrebbe essere basata su di un insieme di elementi di conoscenza, *atomici* oppure *composti* a partire da elementi di conoscenza più semplici in relazione gli uni con gli altri. Ad esempio l'operazione generica di creazione di un documento potrebbe essere descritta come un'operazione complessa di scelta non deterministica fra diverse operazioni elementari, fra cui la creazione del documento via menu e la creazione di documento via pulsante sulla barra degli strumenti. L'insieme di tutti questi elementi e delle loro relazioni definisce un'*ontologia* che è il *modello della conoscenza* del dominio da apprendere. L'ontologia definisce anche il vocabolario dei termini da usare per descrivere le operazioni che l'utente può compiere e il suo *learning goal*. Un

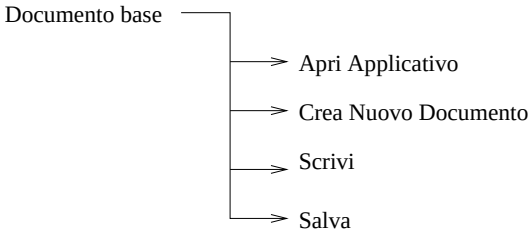

Fig. 3. Un goal può essere conseguito tramite l'esecuzine di sequenze di azioni, atomiche oppure complesse.

modello di conoscenza può essere descritto tramite le relazioni sussistenti tra i suoi elementi; nel linguaggio DyLOG è possibile farlo considerando relazioni di *causalità*. Nella figura 3 è riportato un esempio: supponiamo che lo scopo dell'utente sia di imparare i passi principali necessari a comporre un documento (learning goal), questo scopo può essere raggiunto sfruttando competenze più specifiche, come "apri applicativo", "crea nuovo documento", "scrivi" e "salva". La sequenza di queste operazioni causa la creazione di un documento, tuttavia i modi in cui tali operazioni possono essere effettuate sono i più vari. La figura riporta le alternative possibili per l'operazione di creazione di un documento nuovo però non viene posto nessun vincolo neppure su cosa l'utente deve scri-

vere all'interno del documento per completare l'esercizio o come deve operare per salvarlo.

3 Ragionamento e agenti

Sulla base della definizione di un modello di conoscenza specificato come suggerito nella sezione precedente, il learning by doing può essere realizzato sotto forma di un *tutor virtuale* (per esempio un agente razionale), che applica meccanismi di ragionamento su azioni per facilitare il processo di apprendimento dell'utente. La figura 4 mostra un esempio di azioni rappresentate in uno pseudo-codice relativo ad un linguaggio di programmazione di agenti, sulla falsariga di quanto si fa in DyLOG.

```
action(apri_applicativo) causes view(finestra_applicativo)
        & state(focus_in) & state(cursor_on).
action(bottone_nuovo) causes view(foglio_bianco)
        & view(document) & state(fonts_on) & ...
        if state(focus_in) & view(finestra_applicativo).
action(click_in) causes state(focus_in).
action(scrivi(X)) causes view(X)
        if state (focus_in) & state(cursor_on).
action(bottone_salva) causes state(salva)
        if state (focus_in) & view(document).
```

Fig. 4. Esempio di uso di DyLOG per rappresentare azioni.

I meccanismi di ragionamento applicati saranno finalizzati a) a suggerire sequenze di azioni utili a raggiungere il learning goal dell'utente, b) a monitorare il comportamento dell'utente che deve raggiungere un learning goal o c) a verificare se una sequenza di azioni-utente (click del mouse e input da tastiera) ha come risultato il learning goal atteso. Più specificamente possiamo identificare quattro problemi che possono essere affrontati ulilizzando forme di ragionamento su azioni:

1. dato un *learning goal*, il sistema costruisce una soluzione, che tiene in considerazione la personalizzazione dell'applicativo fatta dall'utente, e la mostra all'utente stesso (*planning* di una soluzione sua *dimostrazione*);
2. dato un *learning goal*, il sistema verifica *se* l'utente riesce a conseguirlo (se la verifica è fatta passo passo si parla di *monitoraggio*);
3. dato un *learning goal*, il sistema verifica *come* l'utente riesce a conseguirlo (valutazione della soluzione a posteriori o monitoraggio);
4. appoggiandosi ad una descrizione della conoscenza gerarchica, del tipo visto nella precedente sezione, diventa possibile definire nuovi learning goal come composizione di learning goal predefiniti e il sistema manterrà la propria capacità di tutoraggio.

Abbiamo visto come uno stesso goal possa essere raggiunto seguendo percorsi differenti. Un esempio di possibili alternative per il caso di creazione di un nuovo documento è riportato nella figura 5. Cominciamo con il considerare il problema (1), planning e dimostrazione. Dato un learning goal, il tutor virtuale può, in generale, costruire molte soluzioni alternative, sfruttando la propria conoscenza in merito alle precondizioni ed agli effetti delle azioni, delle quali si è parlato nella sezione precedente. Qualora si desideri che il tutor insegni all'utente a eseguire un compito mostrandogli in pratica una soluzione, è necessario che la tecnica di ragionamento adottata restituisca una soluzione semplice ed intuitiva, possibilmente minima rispetto al numero di azioni, e quindi facilmente comprensibile all'utente che la deve ripetere. Sequenze quali quella condificata dal ramo di sinistra dovrebbero quindi essere evitate. Per risolvere questo tipo di problema potrebbero quindi rivelarsi particolarmente utili algoritmi classici di AI e di pianificazione, come per esempio A* [6].

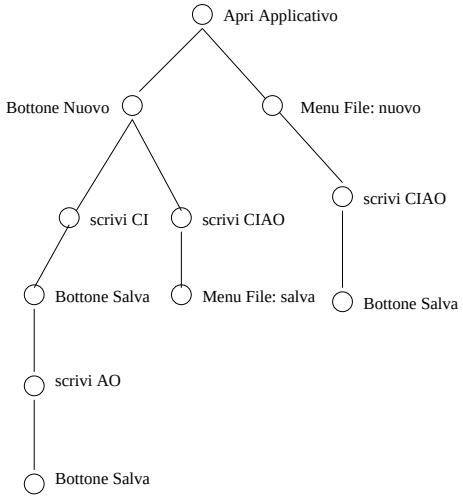

Fig. 5. Un goal può essere conseguito tramite sequenze di azioni differenti.

Nel caso dei problemi di verifica, valutazione e monitoraggio, casi (2) e (3), invece riteniamo che possano essere particolarmente adeguati i meccanismi di ragionamento logico incorporati in linguaggi per la programmazione di agenti, come ad esempio il linguaggio DyLOG, congiunti ad opportuni indici di progresso. Meccanismi di temporal projection (per la verifica) e temporal explanation (per la spiegazione delle cause di fallimento) sono già stati utilizzati nell'applicazione relativa ai piani di studi [1, 2]. In quel contesto il sistema doveva verificare delle proposte di piani di studi costruiti dagli studenti; volendo fare un parallelo con il problema attuale potremmo immaginare un utente che, definito un learning goal, esegue una sequenza di azioni e poi, magari tramite un opportuno strumento dell'interfaccia grafica, richiede l'intervento del tutor per avere un feedback sulla

soluzione applicata (e registrata dal sistema in modo trasparente all'utente). Il sistema controllerà se la condizione goal è stata raggiunta e restituirà opportuni feedback.

Un secondo meccanismo di verifica dell'operato di uno studente consiste nel monitorare costantemente la sua attività, valutando via via le sue azioni. La differenza è quindi che mentre nel primo caso l'agente verificava la correttezza dell'*intera sequenza* prima (caso dei piani di studi) o dopo (caso della verifica del comportamento) la sua applicazione, nei problemi di monitoraggio la verifica delle azioni dell'utente deve essere eseguita *passo passo*, appena *dopo* che ogni azione è stata eseguita. La realizzazione di questa modalità di interazione presenta difficoltà non banali però, a nostro avviso, è anche una delle più interessanti sia perché consente di realizzare potenti strumenti di e-learning sia come caso di studio in un contesto di AI. Per approfondire questo discorso, immaginiamo che i percorsi di azione alternativi riportati nella figura 5 siano questa volta differenti corsi d'azione scelti da alcuni utenti, che affrontano lo stesso learning goal. Si tratta di percorsi corretti, nel senso che consentono tutti di conseguire il goal però se vogliamo che il tutor virtuale restituisca un feedback significativo all'utente occorre che questo compia delle valutazioni relative all'ottimalità della soluzione seguita. Questo tipo di valutazione è estremamente complesso in quanto le azioni possono avere diversi effetti sullo stato del sistema. Infatti un'azione può causare il progresso verso il goal, oppure lo stato può rimanere invariato (apro e chiudo un menú senza cliccare su alcun bottone), si può determinare un regresso ad uno stato precedente (per esempio se chiudo l'applicativo prima di scrivere nel file dovrò riaprirlo per poter conseguire il goal) oppure ancora si può raggiungere uno stato diverso ma non migliore (intendiamo il termine "migliore" in termini di numero di stati che separano dal goal). In generale è molto difficile dire che un utente esegua un'azione errata. Se un utente che deve creare un nuovo file chiude l'applicativo senza salvare, questo potrebbe secondo un'interpretazione severa essere considerato un errore, però cosa dire di quell'utente che, dovendo modificare un dato apre l'applicativo, poi lo chiude, poi lo riapre e modifica il dato? La procedura è stata senza dubbio contorta però lo scopo finale è stato raggiunto. Altro problema: come interpretare il tempo che trascorre fra un'azione e l'altra dell'utente? L'intuito suggerisce che ha acquisito maggiore dimestichezza con l'applicativo quell'utente che esegue una sequenza di azioni corrette rapidamente. Come tenere conte della dimenzione tempo? Una scappatoia a questi problemi è offerta da una caratteristica comune degli esercizi che uno studente affronta quando si esercita con un applicativo: normalmente le esercitazioni sono mirate e le soluzioni non richiedono lunghe sequenze di azioni. Possiamo quindi pensare di ovviare al problema introducendo degli indici di progresso fra i quali una *funzione di costo*. Un utente con un comportamento esplorativo (caratteristico delle fasi iniziali dell'apprendimento) consegue il goal in un numero di passi maggiore; l'esplorazione dovrebbe diminuire quando si richiede all'utente di svolgere operazioni delle quali ha già esperienza.

Una volta definiti indicatori di progresso opportuni, è possibile personalizzare l'interazione fra il tutor virtuale e l'utente (intesa come feedback che il

sistema fornisce) rispetto al comportamento di quest'ultimo. Per esempio, nel caso in cui l'utente si perda in esplorazioni troppo lunghe, il sistema può dedurre che non sappia che cosa fare e quindi costruire e proporre una soluzione all'utente, mescolando in questo modo diversi tipi di ragionamento (verifica e pianificazione).

4 Conclusioni

In questo articolo abbiamo mostrato un possibile uso di tecniche di AI e, in particolar modo, di linguaggi di programmazione di agenti, che supportano il ragionamento su azioni, nel campo dei sistemi di autoistruzione per l'apprendimento di applicativi software. A nostro avviso il dominio dei sistemi di insegnamento interattivi offre ampi spazi d'applicazione, tutt'ora inesplorati, e fornisce un campo sperimentale estremamente interessante per chi si occupa di ragionamento, di agenti e di comunicazione. In particolare ci siamo soffermati sull'analisi dei problemi inerenti la realizzazione di un servizio di monitoraggio delle azioni di un utente, che impara ad utilizza un applicativo compiendo esercitazioni direttamente sull'applicativo stesso e non su di un simulatore. Abbiamo visto che l'uso di agenti razionali per il monitoraggio del comportamento dell'utente offre la possibilitá di realizzare il paradigma di e-learning by doing, paradigma di apprendimento naturale ed efficace e ben piú sofisticato rispetto all'appredimento by repeating, già adottato da alcuni sistemi. L'e-learning by doing è un paradigma potenzialmente assai utile per la realizzare sistemi di appredimento on-line, come i corsi di alfabetizzazione informatica o per il conseguimento della patente europea, piú flessibili degli strumenti attualmente disponibili, che spesso prevedono come unica forma di esercitazione con verifica il quiz. L'applicativo descritto è in corso di progetto e, nelle nostre intenzioni, sarà il punto di partenza per la realizzazione di sistemi ad agenti che coniugano diverse forme di ragionamento, nonché il banco di prova di nuovi meccanismi di ragionamento.

References

1. M. Baldoni, C. Baroglio, and V. Patti. Applying logic inference techniques for gaining flexibility and adaptivity in tutoring systems. In C. Stephanidis, editor, *Proceedings of the 10th International Conference on Human-Computer Interaction (HCII 2003)*, Crete, Grece, 2003. Lawrence Erlbaum Associates. To appear.
2. M. Baldoni, C. Baroglio, V. Patti, and L. Torasso. Using a rational agent in an adaptive web-based tutoring system. In *Proc. of the Workshop on Adaptive Systems for Web-Based Education, AH2002, Selected Papers*, pages 43–55, Malaga, Spain, 2002.
3. P. Brusilovsky. Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11:87–110, 2001.
4. F. Casellato, E. Chiocchi, B. Demo, and M. Lucenteforte. Una forma logica per specificare gli oggetti di windows98. Technical report, Dipartimento di Informatica, Universitá di Torino, November 2002.

5. F.Casellato E. Chiocchi. Elvira: E-learning very innovative research application. Master's thesis, Dipartimento di Informatica, Universitá di Torino, a.a. 2001/2002.
6. P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimal cost paths. *IEEE Trans. Systemns Science and Cybernetics*, 4(2):100–107, 1968.