

Università degli Studi Roma Tre

Dipartimento di Ingegneria



Corso di Laurea Triennale in Ingegneria Informatica

TESI DI LAUREA

Adattamento della piattaforma

Moodle per la personalizzazione di corsi e-learning

Laureando

PIERLUIGI SCIMIA

428020

Relatrice

Prof.ssa CARLA LIMONGELLI

Correlatore

Dott. CARLO DE MEDIO

ANNO ACCADEMICO 2019/2020

Indice

Introduzione.....	2
Capitolo 1 Un sistema di insegnamento adattivo: LS-PLAN.....	4
1.1 Concetti e modellazione della realta di interesse.....	4
1.1.1 Modello dello studente.....	5
1.1.2 Modello della conoscenza.....	6
1.2 Struttura di LS-Plan.....	8
1.2.1 Il Motore Adattivo.....	9
1.2.1.1 Creazione del modello dello studente.....	9
1.2.1.2 Aggiornamento del modello dello studente.....	9
1.2.1.3 Scelta adattiva del nuovo percorso.....	11
1.2.2 Algoritmo Pianificatore.....	12
1.2.3 Teacher Assistant.....	14
Capitolo 2 Un'applicazione di LS-Plan: Moodle_LS.....	16
2.1 La piattaforma Moodle.....	16
2.1.1 Servizi offerti.....	17
2.1.2 Creazione di un nuovo modulo.....	18
2.1.3 Creazione di un nuovo formato di corso.....	19
2.2 LS-Plan in Moodle: Moodle_LS.....	20
2.2.1 Modulo TestLS.....	20
2.2.2 Modulo TeacherAssistant.....	22
2.2.3 Modulo Acourse.....	31
2.2.4 Formato di un corso Adaptive.....	33
2.2.5 LS-Plan.....	46

Capitolo 3 Raffinamento del sistema Moodle_LS.....	54
3.1 Introduzione al lavoro svolto.....	54
3.2 Spazi nei nomi delle conoscenze.....	55
3.3 Eliminazione delle conoscenze.....	58
3.4 Controllo su prerequisiti non definiti.....	59
3.5 Visualizzazione condizionale della sezione iniziale del corso.....	60
3.6 Evitare la ripetizione dei quiz.....	62
Capitolo 4 Approfondimenti sul TeacherAssistant.....	65
4.1 L'architettura a tre livelli	65
4.2 Le progettazioni MVC e modulare	65
4.3 Sicurezza e prestazioni.....	66
4.4 Gestione delle eccezioni.....	67
4.5 Migliorie del plugin.....	67
Capitolo 5 Controllo della consistenza di un corso.....	73
5.1 Introduzione al problema.....	73
5.2 Comando acyclic di GraphViz.....	73
5.3 Algoritmo di Hawick – James.....	74
5.4 Integrazione nel sistema Moodle_LS.....	76
Capitolo 6 Learning Path interattivo.....	79
6.1 Proposta di modifica.....	79
6.2 Funzione Cmap di GraphViz.....	79
Capitolo 7 Conclusione e sviluppi futuri	82
7.1 Conclusioni dopo l'intervento e sviluppi futuri.....	82
Bibliografia.....	84

Introduzione

Nell'ambito della didattica, negli ultimi anni si è cercato di innovare il metodo di fruizione di corsi educativi, non rimanendo ancorati a metodi canonici (lezioni frontali o da remoto tramite computer) che risultano essere statici e non specifici da persona a persona, ma puntando molto su sistemi adattivi, ossia che possano adattarsi dinamicamente alle capacità dello studente, in base a dei parametri che definiscono lo stile di apprendimento e alle conoscenze pregresse. Alcuni studi (si vedano i lavori di Brusilovsky [Br11], Chin [Ch01] e Gena [Ge05]) hanno dato le linee guida per la valutazione di un sistema adattivo, concetto che in ambito didattico spesso viene associato al course sequencing [BV03], ossia alla creazione di una sequenza di attività personalizzata per lo studente, costruita scegliendo materiali considerati dal sistema i più adatti per facilitare il suo percorso formativo.

In questo contesto si colloca LS-PLAN [LSV08], un sistema web che permette sia di avere un corso interattivo, personalizzato e adattivo, sia di non essere legato a particolari tecnologie o piattaforme, quindi utilizzabile in qualsiasi sistema educativo ipermediale.

Oggetto del lavoro di tirocinio è stata un'applicazione di LS-Plan integrata nell'ambiente web Moodle, dal nome Moodle_LS: esso propone le caratteristiche di LS-Plan unite alla facilità di gestione ed utilizzo di Moodle, uno dei più famosi applicativi web per la fruizione di corsi didattici online.

Nel corso del suo sviluppo, Moodle_LS ha subito alcune evoluzioni asincrone, che hanno portato ad una frammentazione della piattaforma, con diverse istanze contenenti diverse caratteristiche: il primo obiettivo è stato unire tutte le migliori precedenti in un'unica versione finale di Moodle_LS. Ciò ha portato ad uno studio preliminare approfondito della

piattaforma, delle modifiche apportate in passato e di eventuali conflitti tra loro, basandosi anche sulla documentazione creata a riguardo.

Successivamente, si sono individuati una serie di interventi mirati per migliorare la fruibilità del sistema, sia eliminando delle pecche funzionali che minavano l'esperienza dello studente all'interno di Moodle_LS, sia inserendo nuove funzioni al servizio del docente per un maggior controllo nella definizione della struttura di un corso.

Nei capitoli che seguiranno verranno presentati inizialmente il sistema adattivo LS_PLAN e le sue caratteristiche teoriche, per poi passare al sistema Moodle_LS nel suo insieme; successivamente verranno descritti tutti gli interventi effettuati nel corso del lavoro di tirocinio, con un'analisi dettagliata dei problemi incontrati, del metodo utilizzato per la loro risoluzione ed un esempio di come l'intervento abbia avuto esito positivo. Nel dettaglio nel capitolo 3 verranno descritte le modifiche fatte al sistema per eliminare difetti (come l'impossibilità di usare il carattere spazio nella definizione di nuove conoscenze, il non poter eliminare delle conoscenze definite ma non più utilizzate in un corso) ed imperfezioni (una visualizzazione più logica e meno ambigua della sezione iniziale di un corso o non riproporre quiz già superati dallo studente); nel capitolo 4 viene introdotto il controllo di consistenza del corso, uno strumento a disposizione di un docente per tenere sotto controllo la struttura del suo corso ed evidenziare eventuali cicli, nel capitolo 5 viene descritta la possibilità di interagire col grafico della struttura del corso, finora proposta in maniera totalmente statica e priva di possibili informazioni aggiuntive.

Nel capitolo finale verranno proposti degli spunti su come potrà evolversi in futuro la piattaforma Moodle_LS, con alcuni appunti su come migliorarne la stabilità.

Capitolo 1

Un sistema di insegnamento adattivo: LS-PLAN

1.1 Concetti e modellazione della realtà di interesse

Il sistema LS-Plan svolge due azioni principali in sinergia tra loro: una modellazione degli stili di apprendimento (Learning Styles) di uno studente, basandosi sulla sua autovalutazione e sul suo comportamento nella navigazione dei materiali didattici; guidare lo studente passo dopo passo, tramite il proprio algoritmo adattivo, imitando il comportamento di un vero docente, intervenendo in maniera mirata specialmente in caso di difficoltà nell'apprendimento.

La modellazione dei Learning Styles si basa sull'idea che essi non rimangono fissi nel tempo, ma possono variare in base alle proprie esperienze, alla sperimentazione di nuovi metodi di studio, come esposto nell'opera di Felder e Spurlin [FSp04]: per tanto, il sistema tiene conto delle azioni che compie lo studente, in modo da valutare in ogni istante se l'attuale strategia didattica offerta sia efficace o bisogna approntare dei cambiamenti.

LS-Plan si basa su tre moduli principali: il motore adattivo, l'algoritmo pianificatore e il Teacher Assistant. Il motore adattivo si occupa della modellazione dello studente; il pianificatore (nella versione precedente del sistema si utilizzava PDK [CMLOP07], ora si è sviluppato un algoritmo di ricerca topologica sulla base del sistema IWT [SGCM08]) produce una sequenza di materiali (Learning Objects Sequence – LOS), sulla base del modello corrente dello studente e delle strategie didattiche indicate dal docente; il Teacher Assistant, infine, si occupa di fornire al docente i mezzi necessari per definire le strategie di insegnamento per ogni materiale didattico e gli obiettivi del corso.

Nelle righe precedenti sono stati nominati concetti come Learning Styles e Learning Object: prima di analizzare in dettaglio il sistema, è bene soffermarsi sui concetti e le teorie che sono dietro LS-Plan.

1.1.1 Modello dello studente

Analizziamo per prima cosa il bisogno di modellare lo studente e la sua conoscenza: l'overlay model [BM07] è il più comune e può usare diversi tipi di parametri per indicare se e quanto una porzione del dominio della conoscenza sia una conoscenza pregressa da parte dello studente. Esso può modellare una conoscenza concettuale o procedurale e può essere esteso tramite un bug model, per tenere in considerazione anche delle conoscenze errate da parte dello studente.

LS-Plan adotta un overlay model, sfruttando la tassonomia di Bloom [Bl56]: nota anche come tassonomia degli obiettivi educativi, è una classificazione gerarchica di passi fondamentali nel processo di apprendimento. Esso consente di creare un sistema che permetta ai docenti di classificare l'apprendimento, così da aiutare gli studenti a migliorarsi a seconda delle loro caratteristiche. Si compone di tre aree: affettiva, cognitiva, e psicomotoria. Ogni area ha dei livelli interni classificati gerarchicamente, in modo da sottolineare l'idea che uno studente debba cercare di sviluppare una solida base in ogni area del dominio prima di poter passare alla successiva.

La tassonomia di Bloom è un sistema utile per aiutare lo sviluppo educativo di uno studente: capire in quale livello è carente può permettere al docente di ideare un percorso formativo ad hoc e aiutarlo a superare le sue difficoltà.

In LS-Plan si è deciso di usare tre dei sei livelli della area cognitiva, per la precisione il livello della Conoscenza, dell'Applicazione e della Valutazione.

Il discente viene rappresentato dal suo stato cognitivo (Cognitive State – CS) e dal suo stile di apprendimento (Learning Style). Il Cognitive State è l'insieme di tutte le conoscenze, rappresentate dai Knowledge Items (KI), che lo studente possiede relativamente ad un dato

argomento. Un Knowledge Item è un elemento atomico di conoscenza relativo ad un dato argomento. Formalmente esso è un insieme così definito:

$$KI = \{KI_k, KI_a, KI_e\} = KI_l$$

dove i valori $l = \{k, a, e\}$ rappresentano un livello cognitivo della tassonomia di Bloom.

Nel modello del discente, oltre al Cognitive State, consideriamo il suo Learning Style; formalmente, è un insieme così definito:

$$LS = \langle D1, D2, D3, D4 \rangle \text{ con } D_i \in [-11, 11]$$

dove ogni elemento rappresenta una dimensione dello stile di apprendimento del modello Felder-Silverman [FS88].

Pubblicato nel 1988, è uno dei modelli più famosi ed utilizzati nella cosiddetta ingegneria dell'educazione. Esso prende ad ampie mani da modelli preesistenti come quello di David Kolb [Kolb84] e dal modello MBTI (Myers-Briggs Type Indicator [MB8085]), specialmente nella categorizzazione degli studenti in base a delle coppie dicotomiche: Attivo/Riflessivo, Sensoriale/Intuitivo, Visuale/Verbale e Sequenziale/Globale.

Nel sistema LS-Plan questi parametri hanno un range che vanno da -11 a +11, secondo la scala ILS (Index of Learning Styles) [FSol] formulata da Felder e Soloman, in modo da definire lo stile di apprendimento preferito dallo studente e renderlo facilmente valutabile da un sistema automatizzato.

1.1.2 Modello della conoscenza

Passiamo ad analizzare come in LS-Plan viene rappresentato il mondo della conoscenza: abbiamo già parlato dei Knowledge Item, unità atomiche di conoscenza di un dato argomento; il loro insieme forma il Dominio della Conoscenza (Domain Knowledge – DK). L'elemento centrale rimane il Learning Node: esso è definito formalmente da una quintupla

$$LN = \langle LM, AK, RK, LS, T \rangle$$

dove rispettivamente:

- LM (Learning Material) indica il materiale didattico usato;
- AK (Acquired Knowledge) è un Knowledge Item KI_i che rappresenta la conoscenza appresa di un dato livello cognitivo, avendo passato il test di valutazione, se presente;
- RK (Required Knowledge) è l'insieme dei KI_i necessari per studiare le risorse di un determinato nodo;
- LS è il Learning Style associato ad un nodo, che il docente ha settato;
- T è una coppia di valori reali $T = (T_{\min}, T_{\max})$ che indicano il tempo stimato per lo studio di quel nodo, valutato dal docente; il sistema valuta automaticamente eventuali pause prese nel corso dello studio (*coffee break effect*) se il tempo di fruizione T_f è maggiore di T_{\max} , ma valuta anche un'eventuale visualizzazione casuale da parte dello studente se T_f è minore di T_{\min} .

Prima abbiamo citato i test di valutazione: essi sono un insieme di k elementi (domande, per esempio) con $k \in \mathbb{N}$; ad ogni elemento è associato un peso $Q_j \in \mathbb{R}$; ogni elemento ha m risposte, con $m \in \mathbb{N} - \{0,1\}$ e ad ogni risposta è associato un peso $w_i \in \mathbb{R}$. S_{KII} è il risultato associato al test e stima la conoscenza dello studente per un dato KI_i .

Nella valutazione dei test si tiene conto anche di un valore soglia, indicato come:

$$\sigma_{KII} = \frac{S_t}{S_{\max}} \quad \text{con } 0 < \sigma_{KII} < 1$$

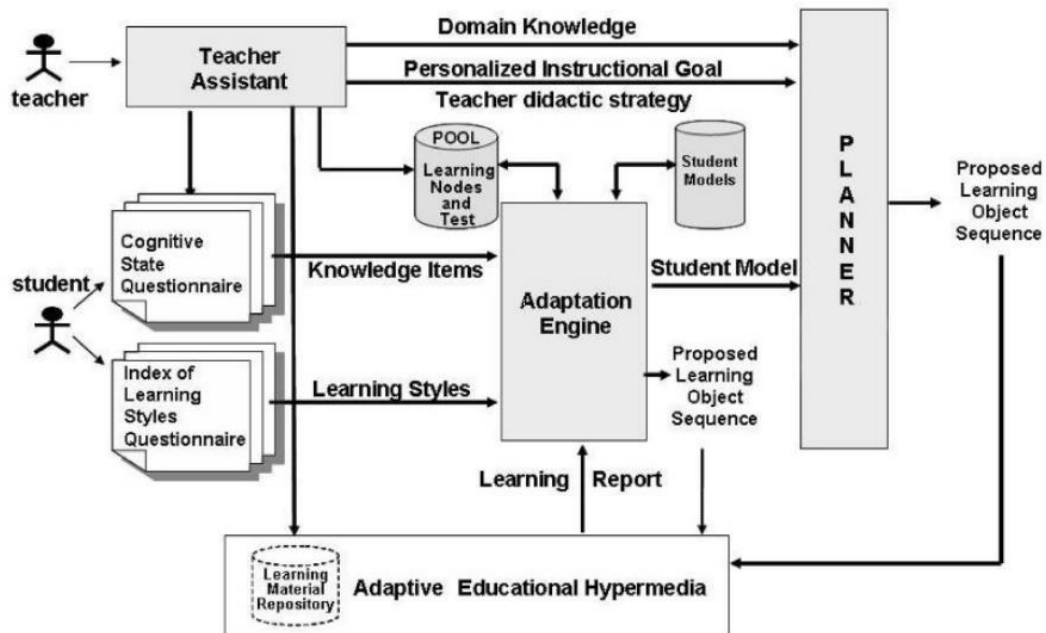
dove S_t è il risultato minimo da prendere in un test per considerare il KI_i come appreso; S_{\max} è il risultato massimo ottenibile (considerando sempre valori positivi).

Da notare come in LS-Plan le domande di un quiz sono in relazione con l'apprendimento di un dato Knowledge Item: includerle nei nodi che trattano certi argomenti permette al sistema di contestualizzarle.

Alcune di queste caratteristiche erano già presenti in altri sistemi: per esempio, il sistema Dynamic Course Generation [BV03] implementa la creazione di una Learning Objects Sequence per mezzo di tecniche di pianificazione, seguendo le orme dei sistemi ELM-ART [WB01] e AHA! [DbSS06]. Ma mentre quest'ultimo non si serve della valutazione per l'adattività, ELM-ART e DCG non implementano gli stili di apprendimento. LS-Plan, invece, prevede entrambe le funzionalità, permettendo al docente di definire uno specifico Learning Style per ogni materiale didattico, potendo così implementare delle strategie didattiche per tipologie di studenti diversi.

1.2 Struttura di LS-Plan

LS-Plan, come descritto rapidamente nella sezione precedente, è composto da tre elementi principali: il Motore Adattivo, l'algoritmo pianificatore e il Teacher Assistant.



Schema del sistema LS-Plan e della sua interazione con un Learning Management System

1.2.1 Il Motore Adattivo

E' il tassello principale del sistema e si occupa di seguire passo dopo passo il percorso dello studente, prendendo in esame ogni sua azione, i suoi risultati nelle prove intermedie, il tempo speso su ogni materiale didattico, al fine di aggiornare il suo modello e nel formulare un cammino didattico ideale per le sue caratteristiche.

1.2.1.1 Creazione del modello dello studente

Non appena lo studente si iscrive al corso, deve affrontare un test per la valutazione delle sue conoscenze pregresse (Cognitive State Questionnaire): ogni domanda è associata ad un Knowledge Item del Dominio della Conoscenza; l'insieme dei KI che risulteranno come appresi (potrebbero anche non essercene, se lo studente non ha conoscenze relative a questo argomento) comporranno il Cognitive State iniziale dello studente. Ma l'inizializzazione del modello dello studente ha bisogno anche di definire una prima versione del suo stile di apprendimento: a questo scopo viene proposto un questionario particolare, l'ILS Questionnaire [FSol], redatto da Felder e Soloman proprio con lo scopo di mappare le sue preferenze secondo le quattro dimensioni del modello Felder-Silverman (modello F-S in seguito), di cui si è parlato in precedenza.

A fine di questo ciclo iniziale di operazioni, la prima modellazione dello studente è conclusa e tutte le sue informazioni vengono salvate in un database apposito.

1.2.1.2 Aggiornamento del modello dello studente

Ogni volta che uno studente consulta un Learning Node, il sistema si attiva per aggiornare il suo modello tramite la funzione UpdateSM, che considera il nodo in questione e la versione attuale dello Student Model. Prima di iniziare, ha bisogno di alcuni dati aggiuntivi: la funzione TimeSpentOnTheNode calcola il tempo speso nella consultazione del LN; la funzione ComputeScorePostTest calcola il voto ottenuto dallo studente nel test relativo al KI_i del nodo, che indica l'Acquired Knowledge che si ottiene dal LN. Se non vi era alcun test, si considera come appreso quel KI_i del nodo.

Successivamente, si considera l'impatto che può aver avuto lo studio dell'argomento sul Learning Style dello studente, in base all'esito dell'eventuale test: se negativo, questo può essere causato o da una definizione del suo stile di apprendimento non fedele alle sue reali caratteristiche o ad una carenza di conoscenze dei prerequisiti richiesti da quella sezione. Il sistema interviene, cercando di emulare il comportamento di un docente: per prima cosa cerca di riproporre l'argomento tramite un Learning Node con un LS differente; se l'esito del test è ancora negativo, la fonte del problema è da ricercarsi nei prerequisiti di quell'argomento, probabilmente non appresi pienamente, che vengono dunque riproposti. Per valutare questo impatto nel Learning Style dello studente, si usano due funzioni alfa e beta per normalizzare il tempo speso sul nodo ed il voto del test, così definite:

$$\alpha(S_{KII}) = \frac{S_{KII} - S_{\min}}{S_{\max} - S_{\min}} \quad \beta(T_f) = \frac{T_f - T_{\min}}{T_{\max} - T_{\min}}$$

con $S_{\min} \leq S_{KII} \leq S_{\max}$ e $T_{\min} \leq T_f \leq T_{\max}$.

Se il voto ottenuto è maggiore o uguale della soglia per quel KI_1 , si considera l'argomento come appreso: in questo caso verrà aggiornato il LS dello studente con una correzione che tende verso i valori del LS del nodo; l'impatto del test sullo studente sarà maggiore se maggiore sarà il voto e minore il tempo speso nello studio. Verrà usata dunque la seguente funzione:

$$\eta_1(T_f, S_{KII}) = \frac{\alpha(S_{KII}) + (1 - \beta(T_f))}{2}$$

Se il voto è minore della soglia, vuol dire che il KI_1 non è stato appreso, quindi il sistema aggiornerà il LS dello studente discostandosi dai valori del LS del nodo: daranno un impatto maggiore un voto particolarmente basso nel test e un tempo di studio elevato. Verrà usata la funzione:

$$\eta_2(T_f, S_{KII}) = \frac{(1-\alpha(S_{KII})) + \beta(T_f)}{2}$$

Le funzioni η_1 e η_2 restituiscono entrambe un numero reale compreso nell'intervallo $[0,1]$; l'effettivo incremento o decremento dei valori dei Learning Styles è dato dal segno della differenza tra i valori D_i dello studente e quelli del nodo:

$$\Delta D_i = D_{iLN} - D_{iold}$$

L'aggiornamento effettivo dei LS dello studente avverrà attraverso la seguente formula, che varia a seconda se il KI_1 è stato appreso o meno:

$$d_{inew} = d_{iold} + \eta_1(T_f, S_{KII}) \cdot \text{sign}(\Delta D_i)$$

$$d_{iold} = \eta_2(T_f, S_{KII}) \cdot \text{sign}(\Delta D_i)$$

Verrà infine aggiornato lo Student Model: se il KI_1 è stato ottenuto, viene aggiornato anche il cognitive state dello studente.

$$SM_{new} \leftarrow (CS_{old} \cup AK, LS_{new})$$

1.2.1.3 Scelta adattiva del nuovo percorso

Ora il motore adattivo ha tutti i dati necessari per poter scegliere il percorso didattico personalizzato per lo studente. La prima operazione da fare è la scelta del prossimo nodo da consigliare: la funzione *NextNode* si occupa di tutto il processo, prendendo come input il modello dello studente, appena aggiornato. Come prima cosa viene fatto un check sui valori D_i del LS: se vi sono state variazioni di segno è sintomo che i LS sono cambiati in maniera molto decisa, quindi si considera necessaria una ripianificazione della Learning Objects Sequence da parte dell'algoritmo pianificatore. Il nodo consigliato diventa dunque il primo della nuova sequenza ottenuta.

Se invece lo studente non ha superato il test, viene valutato il tempo di studio T_f tramite la funzione *Time-Out*, verificando se T_f non è compreso nell'intervallo di tempo assegnato dal docente nella definizione del Learning Node: in tal caso verrà semplicemente riproposto lo stesso nodo appena studiato. Se T_f è congruo al tempo indicato dal docente per quel Learning

Node o lo studente è al secondo tentativo nello studio di quel nodo, verrà usata la funzione CheckClosestNode: cercherà tra i vari Learning Node un nodo alternativo LN_{alt} che abbia gli stessi prerequisiti RK e le stesse conoscenze acquisite AK, ma che sia il più vicino possibile ai Learning Styles dello studente. Tale distanza euclidea è calcolata tramite:

$$d(LS_{LNalt}, LS_{stud}) = \sqrt{(\sum_{i=1}^4 (D_i^{LNalt} - D_i^{LNstud})^2)}$$

Se la ricerca non va a buon fine, viene attivata la funzione OrderedPredecessorsList, che calcola la lista L dei nodi prerequisiti RK, ordinati secondo alcune proprietà:

- i nodi predecessori che non sono stati mai visitati dallo studente;
- i nodi che non hanno un test collegato sono ordinati secondo i livelli di difficoltà K, A, E;
- i nodi che hanno un test collegato, i cui valori LS sono vicini a quelli dello studente.

Le conoscenze acquisite AK dei nodi prerequisiti appena calcolati, se esistono, sono rimossi dallo stato cognitivo CS dello studente, perché evidente segno di una carenza di quegli argomenti. La lista L viene inserita in testa alla LOS e viene consigliato il suo primo elemento come prossimo nodo.

Se l'utilizzo di nodi alternativi o la riproposizione dei prerequisiti non porta al superamento delle difficoltà, il sistema pianifica una nuova LOS da proporre allo studente: se tutti i possibili percorsi portano ad un fallimento, il sistema consiglia un confronto diretto col docente, per trovare una soluzione più adeguata.

1.2.2 Algoritmo Pianificatore

La creazione della Learning Objects Sequence è affidata all'algoritmo pianificatore, che ha il compito di generare il percorso formativo da consigliare allo studente.

Il processo viene gestito come un problema di pianificazione, dove lo studente è l'attore principale, lo stato iniziale è il suo Student Model e le azioni sono l'insieme di tutti i Learning Node.

L'azione principale è lo studio di un materiale didattico da parte dello studente, che può necessitare di conoscenze già acquisite (precondizione) e che, dopo aver studiato e passato l'eventuale test, può dire di aver acquisito una nuova conoscenza (effetto).

L'obiettivo della pianificazione è il target del corso: il sequenziamento del corso equivale quindi ad una sequenza di azioni che portano all'obiettivo.

In LS-Plan, in precedenza, l'algoritmo pianificatore era PDK (Planning with Domain Knowledge, [CMLOP07]), ora sostituito da un algoritmo di ricerca topologica: i nodi di un grafo si definiscono ordinati topologicamente se i nodi sono disposti in modo tale che ogni nodo viene prima di tutti i nodi collegati ai suoi archi uscenti. L'ordinamento topologico non è un ordinamento totale, poiché la soluzione può non essere unica. Nel caso peggiore infatti si possono avere $n!$ ordinamenti topologici diversi, che corrispondono a tutte le possibili permutazioni degli n nodi. E' possibile ordinare topologicamente un grafo se e solo se non ha circuiti al suo interno. La sua applicazione tipica risiede nel problema di pianificare l'esecuzione di una sequenza di attività in base alle loro dipendenze; le attività sono rappresentate dai nodi di un grafo: vi è un arco tra x ed y se l'attività x deve essere completata prima che possa iniziare l'esecuzione dell'attività y . Un ordinamento topologico del grafo così ottenuto fornisce un ordine in cui eseguire le attività.

Trattando i Learning Node ed i loro prerequisiti come un grafo orientato aciclico, possiamo sfruttare l'ordinamento topologico per ottenere la LOS.

L'algoritmo usato in LS-Plan sfrutta la ricerca in profondità, utilizzando tre livelli di marcatura: nulla se il nodo non è stato ancora visitato, grigia se il nodo è attualmente bloccato nell'iterazione ricorsiva corrente, nera se il nodo è stato già visitato. Vengono passati come input l'attuale modello dello studente ed il grafo che rappresenta il corso: per ogni nodo presente nel target formativo dello studente, viene avviato l'ordinamento topologico.

Come prima cosa si controlla se il nodo non sia già stato marcato come nero: in tal caso, il nodo viene bloccato e marcato come grigio. Successivamente si avvia un controllo di consistenza sul grafo, controllando tutti suoi nodi genitori (i prerequisiti): se uno di questi è

stato già marcato come grigio, è stato trovato un ciclo; se non è stato marcato come nero (nodo già visitato), viene rilanciato ricorsivamente l'algoritmo principale.

Alla fine della ricorsione, si completa il processo di creazione della Learning Objects Sequence: di ogni nodo analizzato si verifica se questo non faccia già parte del Cognitive State dello studente, così da aggiungerlo alla sequenza.

Il risultato viene poi consegnato al motore adattivo di LS-Plan, che si occupa di integrarla con gli eventuali Learning Node alternativi, da consigliare in caso lo studente trovi difficoltà nell'affrontare il corso.

1.2.3 Teacher Assistant

Il Teacher Assistant è il terzo componente principale di LS-Plan: si occupa di dare il supporto necessario al docente per creare e gestire l'insieme dei Learning Node, aiutarlo nella gestione della struttura del corso, nell'introduzione di strategie didattiche e l'obiettivo di apprendimento.

Ad ogni materiale didattico introdotto dal docente, deve corrispondere un Learning Node: il docente può esprimere così l'argomento appreso dallo studente e descriverlo nei parametri indicati come nella definizione al paragrafo 1.1.2, ossia definendo una conoscenza acquisita (Acquired Knowledge), le eventuali conoscenze richieste (Required Knowledge), il minimo e massimo tempo stimato per la fruizione di quel materiale didattico, la soglia minima per considerare superato il test associato a quel nodo (se presente) ed i parametri per i Learning Styles. Quest'ultimo passaggio può risultare non troppo chiaro per il docente, che potrebbe definire in maniera errata i parametri delle quattro dimensioni di Felder-Silverman: il sistema, comunque, è in grado di valutare se la definizione dello stile di apprendimento associato è corretto o meno, valutando l'andamento degli studenti. Se coloro che hanno LS coerenti con quello definito per il LN dal docente trovano difficoltà nell'apprendere l'argomento, mentre studenti con LS differenti riescono con facilità nello studio, la definizione data dal docente è da aggiornare.

Tramite il Teacher Assistant, inoltre, il docente può preparare un test per valutare lo stato cognitivo iniziale degli studenti, scegliendo le domande tra quelle utilizzate nella definizione del corso (così da sfruttare l'associazione tra domanda e Knowledge Item, come indicato nel paragrafo 1.1.2) e scegliendo una soglia per considerare tale test come superato.

Capitolo 2

Un'applicazione di LS-Plan: Moodle_LS

2.1 La piattaforma Moodle

Moodle è una piattaforma web open-source per offrire corsi didattici online (elearning), definibile anche come Learning Management System (applicativo software per l'amministrazione, gestione, controllo e fruizione di corsi didattici). Il suo nome è l'acronimo di Modular Object-Oriented Dynamic Learning Environment, ed è frutto del lavoro svolto da Martin Dougiamas, suo ideatore, che ne ha rilasciato la prima versione stabile nell'Agosto del 2002. Ad oggi, Moodle è disponibile nella sua versione più recente, la 3.10.1, ed è uno dei principali LMS al mondo, usato da oltre settanta mila siti nel web.

Essendo scritto in PHP, linguaggio molto usato nel web, e permettendo l'utilizzo di un qualsiasi tipo di database di tipo SQL, Moodle ha trovato anche una grande accoglienza dal lato degli sviluppatori, potendo sfruttare tecnologie ampiamente conosciute.

Considerando la sua natura open-source, l'ampia documentazione messa a disposizione dagli sviluppatori di Moodle e la vasta comunità di supporto e sviluppo che si è venuta a creare col passare degli anni, Moodle rimane una scelta vincente per chi ha necessità di erogare un corso tramite e-learning.

E' da sempre distribuito sotto la licenza GNU General Public License, che permette di utilizzare, copiare e modificare il codice di Moodle, ma obbliga chi ne fa uso di applicare questa stessa licenza ad ogni opera derivata.

2.1.1 Servizi offerti

Moodle offre una vasta gamma di servizi, non solo quelli canonici di un sistema e-learning, per permettere ad amministratori, docenti e studenti una piena e produttiva esperienza del corso.

Lato amministratore permette:

- gestione degli utenti;
- installazione di nuovi plugin;
- gestione della sicurezza (connessioni sicure tramite protocollo HTTPS, gestione dei cookie, antivirus);
- gestione dell'aspetto del sito, con l'installazione di nuovi temi;
- gestione del server, con strumenti di debug per gli sviluppatori.

Lato docente permette:

- creazione e gestione di nuovi corsi;
- creazione di materiali didattici (pagine testuali, pagine web);
- utilizzo di file multimediali da caricare sul server;
- creazione e gestione di quiz, con un resoconto dettagliato delle valutazioni ottenute dagli studenti;
- la creazione di pagine wiki;
- creazione di sondaggi;
- un calendario dove annotare eventi o scadenze;
- gestione di un forum di discussione con gli studenti.

Lato studente permette:

- visualizzazione dei materiali didattici;
- un feedback della propria preparazione con i test messi a disposizione, con possibilità di commenti e di interagire col docente;
- un forum di discussione dove parlare con gli altri studenti e confrontarsi col docente.

Indubbiamente, ciò che ha permesso un così vasto utilizzo di Moodle è stata la possibilità di creare ed installare nuovi plugin: la “M” nel nome Moodle sta proprio ad indicare la modularità di cui gode questa piattaforma.

E' possibile installare o creare plugin di svariato genere: nuove tipologie di corso, nuove attività (o moduli), nuove tipologie di compiti per gli studenti, nuove modalità di esportare o importare dati, nuove tipologie di domande, nuovi motori di ricerca interni, nuovi temi grafici. Questa componente modulare è stata fondamentale nella decisione di implementare LS-Plan per Moodle.

La versione di Moodle che prenderemo in considerazione d'ora in poi sarà la 3.10.1, ossia quella attualmente utilizzata in Moodle_LS.

2.1.2 Creazione di un nuovo modulo

Per implementare il sistema LS-Plan in Moodle è stato necessario creare dei moduli adatti. Un modulo in Moodle richiede dei file necessari per il corretto funzionamento:

view.php, serve per la corretta visualizzazione del modulo da parte degli utenti e viene richiamato dal file view.php del corso al momento della selezione di quel particolare modulo all'interno del corso;

lib.php, dove sono contenute tutte le funzioni standard che regolano il corretto funzionamento del modulo; Moodle si attende alcune funzioni fondamentali che regolano il comportamento del modulo in questione e che vengono richiamate dal sistema, come modulename_add_instance, modulename_update_instance o modulename_delete_instance; mod_form.php, dove vi è definita la form da utilizzare per raccogliere tutti i dati per la creazione del nuovo modulo ed eventualmente per la sua modifica; contiene una classe di nome mod_modulename_mod_form che estende la classe moodleform_mod; questa classe verrà istanziata da modedit.php del corso, che gestisce la modifica e l'aggiunta dei moduli in Moodle relativamente a uno specifico corso; estendendo la classe moodleform_mod si eredita un oggetto \$mform che richiamando una sua funzione chiamata

addElement(tipoElemento, nomeCampo, testoDelCampo) permette di strutturare molto facilmente una form;

index.php, usata da Moodle per elencare tutte le istanze del modulo che vi sono in un dato corso;

version.php, che tiene traccia della versione del modulo e di altri attributi, come la versione di Moodle richiesta per il suo corretto funzionamento;

una cartella db, contenente il file install.xml che indica la struttura delle tabelle da creare quando viene installato il modulo; è obbligatoria almeno una tabella che abbia lo stesso nome del modulo da installare e che possenga i campi:

int id;

char course;

char name;

una cartella lang, con una cartella per ogni lingua supportata dal modulo e all'interno di esse un file modulename.php, che contiene le informazioni per la traduzione del modulo. L'installazione di un modulo avviene caricando sul server tutti i suoi file nella directory /var/www/moodle/mod, dove /var/www è la locazione in cui si trova l'istanza di Moodle.

2.1.3 Creazione di un nuovo formato di corso

In Moodle per definire un nuovo formato di corso si devono rispettare alcune convenzioni per il suo corretto funzionamento, ossia devono essere presenti:

una cartella col nome del nuovo formato, che conterrà tutti i suoi file di configurazione e gestione, da mettere poi all'interno del percorso moodle/course/format;

una cartella db, con la medesima struttura di quella dei moduli, anche se per i formati dei corsi è opzionale;

una cartella lang, anche questa con la stessa struttura e significato della cartella lang dei moduli, con l'eccezione che delle informazioni sulla lingua dovranno trovarsi nel file moodle.php contenuto in moodle/lang/linguaDiRiferimento;

tre file, precisamente ajax.php, config.php e format.php, con quest'ultimo che si occupa della gestione e della visualizzazione del nuovo formato di corso.

2.2 LS-Plan in Moodle: Moodle_LS

L'integrazione di LS-Plan in Moodle è del tutto trasparente per l'utente che sta utilizzando il sistema, sia questo docente o studente: le attività ed i quiz usati sono quelli che offre Moodle, mentre la logica di personalizzazione del corso è presentata come un nuovo formato di corso. Analizzando la struttura del sito, invece, per completare l'integrazione si sono resi necessari la creazione di tre nuovi moduli (TestLS, TeacherAssistant, Acourse), alcune modifiche nel database di Moodle, un nuovo database dedicato a LS-Plan e la definizione di un nuovo formato di corso, Adaptive.

2.2.1 *Modulo TestLS*

Questo modulo, che poi verrà inserito in automatico nella sezione zero del corso, permette l'inserimento dei Learning Styles da parte dello studente, nel caso in cui ancora non siano stati inseriti.

Come ogni modulo in Moodle, contiene i file .php canonici previsti dal sistema (index, lib, mod_form, view e version): analizziamoli da vicino.

- index.php

Il file index.php di questo modulo non ha particolari funzioni aggiuntive, per questo contiene solo il codice per la visualizzazione degli elementi standard della pagina di Moodle.

- mod_form.php

Il file mod_form.php serve per la visualizzazione della form per l'inserimento dei dati del modulo; è costituito da una condizione if centrale, che differenzia i due casi in cui si può

accedere al file, controllando se è presente il parametro id nell'URL con cui si arriva a tale file, il quale indica se si tratta di una creazione o di una modifica (visualizzazione) di un'istanza del modulo.

Nel caso si tratti di una nuova istanza, l'utente sarà per forza un docente: in questo caso non sarà richiesto nessun dato, ma solo la conferma della creazione di tale modulo.

Nel caso di visualizzazione o modifica di un'istanza già presente, il sistema controlla il ruolo dell'utente tramite la funzione *isTeacher(\$COURSE->id, \$USER->id)*: se è uno studente, verranno mostrati i campi per settare i valori tramite delle choice, valori che verranno inviati alla funzione *test_update_instance* di *lib.php*, che elaborerà i dati ottenuti ed aggiornerà il record nella tabella *testls_users* del database; in caso di utente docente, invece, verrà mostrato solo lo schema della pagina di visualizzazione, con un link per tornare alla pagina del corso.

- lib.php

Contiene solo le funzioni standard previste per i moduli in Moodle, l'unica modifica apportata riguarda la funzione *testls_update_instance(\$testls)*: essa come parametro riceve un oggetto che ha come campi i valori inseriti dallo studente nella form visualizzata da *mod_form.php*. Questa funzione controlla, tramite una serie di condizioni if-else, i valori delle quattro dimensioni del modello F-S inseriti: se la componente sinistra della dimensione è diversa da zero, allora la dimensione avrà il valore di questa componente cambiato di segno, altrimenti la dimensione assumerà direttamente il valore della componente destra. I valori di queste dimensioni vengono memorizzate all'interno dei campi (quattro campi, uno per ogni dimensione), aggiunti all'oggetto *testls* ed inseriti nel database.

- view.php

Rispetto al template tipico dei file *view.php* che si trovano nei moduli di Moodle, in questa occasione si è pesantemente intervenuti nel codice, affinché potesse svolgere le stesse funzioni del file *modedit.php* presente nella cartella *course*, appoggiandosi sempre al file

mod_form.php di testls per la visualizzazione della form necessaria a richiedere i LS allo studente. Questo perché quando l'utente clicca su un'istanza di un modulo all'interno di un corso, viene reindirizzato al file view.php del modulo, passando sull'URL l'id dell'istanza che si vuole visualizzare: nel caso in questione, la visualizzazione dell'istanza sarà la form di mod_form.php, quindi risultano indispensabili tutte le funzioni di supporto alla sua gestione.

- cartella db

Il file install.xml, necessario per la creazione delle tabelle nel database che riguardano il modulo, contiene sia la definizione della tabella testls (richiesta da Moodle), ma anche la tabella testls_users, che serve per rendere persistenti i LS di uno studente. Essa ha i campi id (auto-incrementante), userid per l'id dello studente, più quattro campi che rappresentano le quattro dimensioni del modello F-S (active-reflective, sensing-intuitive, visual-verbal, sequential-global).

2.2.2 Modulo TeacherAssistant

Il modulo TeacherAssistant implementa le funzioni dell'omonimo componente di LS-Plan, quindi permette al docente di associare i Learning Object con le informazioni richieste dal motore adattivo. Alla creazione di una nuova istanza di TeacherAssistant viene richiamata, da modedit.php, la pagina mod_form.php; alla conferma ed invio dei dati inseriti al server, la richiesta viene presa in consegna dalla funzione teacherassistant_add_instance di lib.php: si occuperà di associare i Knowledge Item alle domande dell'eventuale quiz, salvando il tutto nella tabella adaptive_questions_ki; successivamente le risorse della sezione vengono registrate nella tabella teacherassistant_attivita_corso, mentre il Learning Node sarà salvato in teacherassistant_domain. Come ultima azione verrà preparata una stringa da inviare al sistema LS-Plan, per salvare le informazioni sui metadati nella tabella learningnodes.

- view.php

Viene usato per reindirizzare l'utente al file `modedit.php` di `course`, indicando nell'URL alcuni parametri:

- `update`, con il valore `id` del record della tabella `course_modules` relativo all'istanza di `TeacherAssistant` che si vuole visualizzare;
- `course`, con il valore `id` del record della tabella `course` relativo al corso che contiene l'istanza di `TeacherAssistant` che si vuole visualizzare;
- `section`, con il valore della sezione del corso in cui è presente il modulo di `TeacherAssistant` che si vuole visualizzare;
- `return` indica se è richiesto un qualche dato di ritorno, nel caso in questione è settato a zero.

E' così possibile sfruttare la stessa form utilizzata nel momento dell'aggiunta di un'istanza di tale modulo, con `mod_form` che avrà i suoi campi opportunamente compilati con i dati precedentemente inseriti tramite specifiche funzioni contenute in `lib.php`.

- `lib.php`

Il suo compito è di tenere sincronizzate le informazioni che sono memorizzate nelle tabelle del modulo `TeacherAssistant` nel database di Moodle e le informazioni riportate nel database di LS-Plan. Queste operazioni devono avvenire quando vengono chiamate le funzioni standard dei moduli, quali:

- `teacherassistant_add_instance($teacherassistant)`: oltre a svolgere le istruzioni di base previste per ogni modulo, come l'inserimento di un nuovo record nella tabella `teacherassistant`, esegue la creazione di una nuova istanza del modulo e l'aggiunta di un `Learning Node` al dominio, con le informazioni ottenute tramite la form `mod_form.php` e contenuti nell'oggetto `$teacherassistant` passato come parametro.

Tra le altre mansioni la funzione si occupa di:

- associare le domande ai Knowledge Item richiamando la funzione *associa_KI_Questions(\$teacherassistant)*;
- registrare le risorse e le attività del LN come componenti del corso adattivo, tramite la funzione *attivitacorso(\$teacherassistant)*;
- aggiungere il LN alla tabella *teacherassistant_domain*, controllando se non sia già stato definito quel concetto come RequiredKnowledge di un altro nodo (in tal caso si aggiorna il record già presente); per questo si controlla l'esistenza di un record nella tabella *teacherassistant_domain* con AK uguale a quello del nodo che si vuole aggiungere: in presenza di un risultato, si controlla se il campo *section* sia maggiore di zero (ossia si tratta di un LN distinto con lo stesso AK) e in tal caso si aggiunge al dominio tramite la funzione *addLnToDomain(\$teacherassistant)*, altrimenti si richiama la funzione *updateDomain(\$teacherassistant)* aggiungendo a *\$teacherassistant* il campo *esisteLN* settato a -1; se non dovesse esserci nessun record nel database, significa che non esiste nessun LN con quell'AK e non è mai stato usato come RK, quindi è sufficiente richiamare la funzione *addLnToDomain(\$teacherassistant)*;
- prepara una stringa chiamata *\$metadati* contenente tutti i metadati del nodo tramite la funzione *elaboraMetadati(\$teacherassistant)*;
- richiama *Accesso.php* per poter salvare la stringa *\$metadati* all'interno del database di LS-Plan, utilizzando come carattere di controllo il numero 1, che verrà interpretato come operazione di inserimento di un LN;
- richiama la funzione *aggiungiNodoAlGrafo(\$teacherassistant->name, \$teacherassistant->rk, \$teacherassistant->course, \$COURSE->fullname)*, per inserire tale nodo nel file *.dot* per la costruzione del grafo.

- *teacherassistant_update_instance(\$teacherassistant)*: si occupa di aggiornare i metadati di un LN ottenuti tramite la form con l'oggetto \$teacherassistant, andando ad aggiornare le tabelle teacherassistant e teacherassistant_domain, elaborare i metadati salvandoli in una stringa tramite la funzione *elaboraMetadati(\$teacherassistant)* e aggiornare il record nel database di LS-Plan richiamando il file Accesso.php, utilizzando come carattere di controllo il numero 2, interpretato come operazione di aggiornamento del LN.
- *teacherassistant_delete_instance(\$id)*: si occupa di cancellare l'istanza \$id di TeacherAssistant, occupandosi di eliminare anche il LN a cui fa riferimento quell'istanza; prima si occupa di cancellare ogni record relativo a quel nodo presente nelle tabelle teacherassistant_domain, adaptive_question_ki, teacherassistant_attivita_corso che ha i campi section e id_course coincidenti con quelli del LN da cancellare; successivamente richiama Accesso.php e passa come parametro l'id del LN ed il corso a cui appartiene, con codice operazione 3 (cancellazione del nodo).

Sono state aggiunte anche delle funzioni di supporto, alcune delle quali sono state già citate nelle righe precedenti:

- *elaboraMetadati(\$form)*: elabora tutti i valori inseriti nella form e riportati nell'oggetto \$form, costruendo così una stringa composta da tutti i metadati inseriti dall'utente, separati da “;”;
- *attivitacorso(\$form)*: inserisce tutte le risorse ed attività che sono contenute all'interno della sezione del modulo TeacherAssistant all'interno della tabella teacherassistant_attivita_corso, questo per sapere quali sono le risorse/attività che compongono il corso adattivo; per ottenere la sequenza degli elementi in una sezione questa funzione usa *getSequence(\$section, \$course)* che ritorna una stringa come sequenza di id degli elementi della sezione;

- *hasQuiz (\$section, \$course)*: individua e restituisce, se presente, l'id della tabella *course_modules* dell'istanza del quiz intermedio presente nella sezione, altrimenti ritorna -1; anche questa funzione si appoggia a *getSequence(\$section, \$course)* cercando semplicemente, all'interno della sezione, l'elemento che sia un'istanza di "quiz";
- *getSequence (\$section, \$course)*: ritorna una stringa che rappresenta la sequenza di id dei record della tabella *course_modules*, che indicano le istanze degli elementi che sono presenti nella sezione;
- *associa_KI_Questions(\$form)*: si occupa di legare ogni domanda contenuta nel quiz della sezione a tutte le risorse ed attività presenti nella sezione; per fare questo sfrutta la funzione *hasQuiz (\$section, \$course)* per ottenere l'id dell'istanza del quiz, così da poter ottenere la sequenza delle domande del quiz; tramite un ciclo *for* lega ogni domanda alla sezione in cui è contenuta, tramite la funzione *KI_Question (\$form, \$idQuestion)*;
- *KI_Question (\$form, \$idQuestion)*: si occupa di raccogliere i dati necessari all'inserimento di un nuovo record nella tabella *teacherassistant_questions_ki* creando un nuovo oggetto con i campi richiesti;
- *addLnToDomain(\$form)*: si occupa di raccogliere tutti i metadati del nuovo LN da aggiungere al dominio, nella tabella *teacherassistant_domain*;
- *updateDomain(\$form)*: si occupa dell'aggiornamento dei valori nella tabella *teacherassistant_domain*, sfruttando la funzione *aggiornaRecords (\$table, \$object, \$where)* con la clausola *\$where* differente a seconda se si tratta dell'aggiornamento di un LN con i metadati già definiti o se si vuole utilizzare il record di un RK non ancora legato a nessun LN, ma che ha lo stesso AK del nodo che si sta aggiungendo;
- *aggiornaRecords (\$table, \$object, \$where)*: prende tutti i record della tabella *\$table* che rispettano le condizioni riportate in *\$where* e li aggiorna con i valori riportati nei campi contenuti nell'oggetto *\$obj*;

- *getKi(\$course)*: restituisce un array con tutti gli AK e i livelli degli AK della tabella `teacherassistant_domain`, riportati nel campo `ak_levelak`;
- *getMetadatiLN (\$idLN, \$idCourse)*: ritorna i metadati del LN con id uguale a "id" concatenato con \$idLN del corso \$idCourse; a questo scopo richiama il file `Accesso.php` con codice operazione 4, che permette di leggere dalla tabella `learningnodes` di LS-Plan e legge da file i metadati restituiti da tale classe, restituendo la prima riga di tale file in un array contenente i vari campi dei metadati;
- *checkMultipli (\$RKs, \$newRK)*: controlla se c'è già l'elemento \$newRK all'interno della stringa \$RKs costituita da un insieme di stringhe separate da “,”; ritorna true se è presente, altrimenti false;
- *restoreConnectionDB ()*: ripristina la connessione al database di Moodle, a seguito delle operazioni eseguite sul database di LS-Plan, precedentemente effettuate tramite il file `Accesso.php`.

- locallib.php

E' una seconda libreria a disposizione del modulo, ideata per isolare altre funzioni per compiti specifici (come la gestione dei punteggi nei quiz) e per non appesantire troppo il file `lib.php`.

- *normalizzaPunteggio (\$score, \$section, \$course, \$maxScore, \$votoMax)*: ritorna il valore normalizzato di \$score, leggendo la scala di normalizzazione dal campo `scale` della tabella `mdl_acourse_quiz_course`;
- *getScores(\$form)*: ritorna un oggetto in cui sono riportati l'id del quiz, il punteggio massimo ottenibile ed il voto massimo del test intermedio riportato nella sezione, mentre ritorna -1 se non è stato trovato nessun quiz nella sezione;

- *getMinScore(\$idQuiz)*: calcola e ritorna il punteggio minimo del quiz con id pari a \$idQuiz, sommando il punteggio minimo ottenibile di tutte le domande che compongono il quiz, appoggiandosi alla funzione *get_min_fraction(\$answers)*;
- *get_min_fraction(\$question)*: ritorna il valore più penalizzante della domanda;
- *get_scala(\$course)*: ritorna la scala di normalizzazione impostata dal docente per il corso con id \$course;
- *aggiungiNodoAlGrafo (\$ak, \$rk, \$corso, \$nomeCorso)*: richiama la funzione *inserisciCoppiaNelGrafo (\$ak, \$rk, \$corso, \$nomeCorso)* del file grafo.php in adaptive, per inserire tale nodo nel file .dot per la definizione del grafo;
- *eliminaNodo (\$ak, \$course)*: richiama la funzione *cancellaNodo (\$ak, \$course)* del file grafo.php in adaptive, per eliminare tale nodo dal file .dot per la definizione del grafo;
- *modificaNodoDelGrafo (\$vecchioAK, \$nuovoAK, \$rk, \$corso)*: richiama la funzione *modificaGrafo (\$vecchioAK, \$nuovoAK, \$rk, \$corso)* del file grafo.php in adaptive, per aggiornare un nodo già esistente nel file .dot.

- mod_form.php

Si occupa di visualizzare e gestire la form di TeacherAssistant; se si tratta di un aggiornamento compilerà tutti i campi con i dati salvati in precedenza. Inoltre, si occupa di controllare se esiste un quiz intermedio per il LN a cui fa riferimento l'istanza specifica di TeacherAssistant, mostrando eventualmente i campi relativi.

Per capire se si tratta di una nuova creazione di un'istanza o di una sua modifica, viene controllato l'URL con cui è stata chiamata la pagina modedit.php e si ricerca la presenza del parametro \$update: se vale zero (ossia è assente), si è nel caso di una nuova istanza di TeacherAssistant, quindi tutti i campi della form avranno i valori di default; per capire se

esiste anche un quiz intermedio, si richiama la funzione `getScores($form)` di `locallib.php` ed eventualmente se ne normalizza il punteggio; se `$update` è diverso da zero, esso contiene l'id della tabella `course_modules` relativo all'istanza di `TeacherAssistant` che si vuole modificare, quindi si riempirà la form con tutti i dati ottenuti dalla funzione `getMetadatiLN($idLN, $idCourse)` di `lib.php` e per il quiz si controllerà se il valore di `$maxScore` del quiz è maggiore di zero.

In entrambi i casi, i dati inviati dalla form saranno elaborati dalla corrispondente funzione di `lib.php`: `teacherassistant_add_instance($teacherassistant)` nel caso della creazione, `teacherassistant_update_instance($teacherassistant)` nel caso di aggiornamento.

- `index.php`

Non svolge particolari funzioni aggiuntive, ma solo quelle di default previste da Moodle.

- cartella db

Vengono create, al momento dell'installazione del modulo, le seguenti tabelle: `teacherassistant`, con tutti i campi standard previsti da Moodle ed il campo `section`, che indica l'id del LN a cui fa riferimento; `teacherassistant_domain`, contenente tutte le informazioni su `Acquired Knowledge` e `Required Knowledge`, utili poi per generare il grafo; `teacherassistant_attivita_corso`, che riporta tutte le attività o risorse che fanno parte del corso.

2.2.3 Modulo *Acourse*

Questo modulo, inserito in automatico nella sezione zero del corso, permette al docente l'inserimento di informazioni generali quali l'obiettivo formativo, la soglia di acquisizione dei Knowledge Item per il test iniziale, la scala di valutazione dei punteggi dei quiz.

Dei classici file richiesti da Moodle, il file `view.php` ricopre una funzione particolare.

- `index.php`

Contiene il codice per la sola visualizzazione degli elementi standard della pagina di Moodle.

- lib.php

Contiene, oltre alle funzioni standard previste in Moodle, alcune funzioni necessarie per il recupero dei dati precedentemente inseriti; inoltre sono state apportate delle modifiche alla funzione di base per l'aggiornamento dell'istanza.

- *acourse_update_instance (\$acourse)*: recupera dalla form di Acourse i dati da inserire nel database, ossia tutti i KI che faranno parte dell'obiettivo formativo del corso, il punteggio minimo da ottenere tra tutte le domande del test iniziale relative ad un KI per considerarlo già appreso e la scala di valutazione dei quiz;
- *get_Ki_Course (\$course)*: ritorna un array che ha come chiave e valore l'AK di tutti i KI che compongono corso con id pari a \$course;
- *getStudentGoalRecord (\$courseid, \$studentid)*: ritorna il record *acourse_goal_course* che rappresenta il goal dello studente *\$studentid* per il corso *\$courseid*;
- *get_quiz_course_record (\$courseid)*: ritorna il record dalla tabella *acourse_quiz_course* che rappresenta le informazioni sui quiz per il corso *\$courseid*;
- *get_students_course (\$courseid)*: ritorna un array con tutti gli id degli studenti iscritti al corso *\$courseid*.

- mod_form.php

Permette la visualizzazione della form per l'inserimento di un obiettivo formativo per gli studenti; inoltre permette l'inserimento di una soglia per considerare già posseduto un KI e la scelta di una scala di valutazione dei punteggi.

Una volta inviati al server i dati della form, si viene reindirizzati verso view.php di Acourse, che eseguirà le stesse operazioni di modedit.php.

- view.php

Come anticipato, il file view.php di acourse è stato pesantemente modificato rispetto alle funzioni classiche previste da Moodle, affinché svolgesse le stesse funzioni del file modedit.php di course, appoggiandosi sempre a mod_form.php di Acourse per la visualizzazione della form necessaria per richiedere le informazioni del corso.

Questo perché quando l'utente seleziona un'istanza di un modulo all'interno di un corso, viene reindirizzato alla pagina view.php del modulo, con l'id esplicitato come argomento nell'URL; nel nostro caso, la visualizzazione dell'istanza sarà proprio la form di mod_form.php, quindi sono necessarie tutte le funzioni di supporto per la sua gestione.

- cartella db

Nel file install.xml sono specificate come necessarie per il modulo acourse le seguenti tabelle: acourse, come tabella indispensabile per Moodle, con i campi per indicare l'id della tabella mdl_course che contiene l'istanza di Acourse ed il suo nome; acourse_goal_course, che contiene l'elenco degli obiettivi formativi per ogni studente per un dato corso (se l'id dello studente è 0, l'obiettivo si estende a tutti gli studenti); acourse_quiz_course, per registrare la soglia da ottenere tra tutte le domande del test iniziale relative ad un KI per considerarlo già appreso e la scala di valutazione dei quiz intermedi.

2.2.4 Formato di un corso Adaptive

Per la creazione del corso Adaptive si è sfruttato il file `format.php` del formato di corso Topics, dato che l'idea è quella di sfruttare la stessa struttura, in cui la sequenza degli argomenti corrisponde al corso personalizzato ottenuto tramite il sistema LS-Plan.

Per far questo, si è individuata la variabile `$sequence`, di tipo array, in cui viene memorizzato l'ordine di presentazione delle sezioni all'interno del corso: modificando la sequenza di sezioni da visualizzare, si può presentare all'utente un corso personalizzato.

Per supportare questa caratteristica, è necessario che il sistema conosca i Learning Styles ed il Cognitive State dello studente; per questo nella sezione iniziale viene imposto al docente di inserire un'istanza del modulo Testls ed il questionario iniziale.

La visualizzazione del corso cambia in base al ruolo dell'utente, controllo che si effettua tramite la funzione di Moodle, `isteacher($course->id, $USER->id)`: in caso di docente, verrà visualizzata l'intera struttura del corso, nell'ordine in cui i Learning Node sono stati inseriti, comprese le varie istanze del modulo TeacherAssistant; se l'utente è un docente, per prima cosa viene effettuato un controllo sulla presenza di un record nella tabella `testls_users`, contenente i LS dello studente, per verificare se il sistema conosce già il suo stile di apprendimento. In caso positivo, non verrà mostrata l'istanza del modulo TestLS, ma solo il corso personalizzato.

- `format.php`

Le funzioni principali che gestiscono la personalizzazione del corso risiedono in `format.php`. Come prima cosa vengono svolti sempre dei controlli di routine: l'esistenza o meno di un'istanza di TestLS e di una risorsa di link al Learning Path nella sezione iniziale del corso. In caso contrario, si provvederà ad aggiungere tali moduli in maniera automatica tramite le rispettive funzioni `insert_testLS($courseid)` e `add_link_to_graph($courseid, $coursename)`. Successivamente si presenta una prima importante diramazione nel flusso di esecuzione del codice, basato sul ruolo dell'utente.

Se l'utente è un docente (o un utente di ruolo superiore), si controlla se l'ultima operazione svolta sia una modifica ad un quiz intermedio: in tal caso si effettuerà l'associazione delle domande del quiz modificato con la sezione a cui appartiene tramite la funzione *associaKIQuestions(\$section, \$courseid, \$quizid)*; successivamente elimina dalla sezione iniziale le voci che riguardano lo studente (istanza del modulo TestLS e link al suo Learning Path), stampando successivamente l'intero corso, senza approntare alcuna personalizzazione.

Se l'utente è uno studente, il sistema passerà a gestire la personalizzazione del corso. Prima di tutto si controlla se ha già affrontato il questionario sugli stili di apprendimento, andando a ricercare un record valido nella tabella *testls_users*: in caso positivo, non verrà mostrata l'istanza di TestLS presente nella sezione iniziale.

Successivamente, inizia la personalizzazione del corso: il flusso di esecuzione dipende dalla situazione in cui si trova lo studente, che dipende dal fatto se sia stato o meno avviato LS-Plan per quello studente in quel corso, controllo che si effettua tramite una richiesta al database, alla tabella *adaptive_log_lsplan*:

- se non è presente alcun risultato, si controlla se è stato già fatto il test iniziale, tramite la funzione *get_record_initial_test(\$course->id, \$USER->id)*;
 - in caso negativo, viene mostrata solo la sezione iniziale del corso, contenente le risorse per effettuare il test iniziale;
 - se il test è già stato effettuato, si avvia per la prima volta il motore adattivo, utilizzando la funzione contenuta in *locallib.php* *first_call_lsplan(\$USER, \$cs, \$ls, \$goal, \$course->id)*, che prende in input anche il Cognitive State dello studente, i suoi Learning Styles e gli obiettivi formativi del corso. Il suo risultato sarà poi inserito nella tabella *adaptive_course_pers*;
- se vi è un risultato, vuol dire che il test iniziale è stato compilato ed è stato avviato il motore adattivo almeno una volta; si va a controllare, dunque, l'ultima azione svolta dallo studente tramite una richiesta alle tabelle *log* e *teacherassistant_attivita_corso*;

- se si tratta di un quiz, si verifica se è stato portato a termine o meno;
 - in caso di sola visualizzazione, viene caricata la Learning Objects Sequence ottenuta precedentemente dall'ultima chiamata di LS-Plan;
 - se il quiz è concluso, si verifica che LS-Plan non sia stato già chiamato per quell'azione, confrontando l'orario della sua ultima chiamata con quello di conclusione del quiz;
 - in caso di orari diversi, si prepara la chiamata al motore adattivo, preparando i giusti dati in input:
 - si preleva il punteggio già normalizzato ottenuto dallo studente, tramite la funzione *get_sumgrades_last_attempt(\$userid, \$idQuiz, \$courseid)*;
 - si calcola il tempo di fruizione delle risorse didattiche tramite la funzione *get_fruition_time(\$ln, \$userid, \$lastCallTime)*;
 - si chiama LS-Plan tramite la funzione *call_lsplan(\$USER-id, \$section, \$course->id, \$score, \$fruitionTime)*;
 - si tiene conto della chiamata effettuata, inserendo un nuovo record nella tabella *adaptive_log_lsplan*;
 - si legge la LOS ottenuta dal motore adattivo e la si salva nella tabella *adaptive_course_pers*;
 - in conclusione, nell'array *\$sequence* viene messa la sequenza personalizzata e nella sezione iniziale verrà oscurata la risorsa link al test iniziale;

- se non riguarda un quiz, ma una risorsa, si controlla se a questa è associata un test intermedio, tramite la funzione *exist_test(\$course_section->section, \$course_section->course);*
 - se è presente un quiz, viene visualizzata la LOS precedentemente ottenuta;
 - altrimenti, si verifica che LS-Plan non sia stato già chiamato per quell'azione;
 - in caso positivo, viene visualizzata la LOS precedentemente ottenuta;
 - in caso negativo, si effettua la chiamata al motore adattivo tramite la funzione *call_lsplan(\$USER->id, \$section, \$course->id, 0, \$fruitionTime)*, dove il valore 0 indica che non è presente il quiz;
 - si tiene conto della chiamata effettuata, inserendo un nuovo record nella tabella *adaptive_log_lsplan*;
 - si legge la LOS ottenuta dal motore adattivo e la si salva nella tabella *adaptive_course_pers*;
 - in conclusione, nell'array *\$sequence* viene messa la sequenza personalizzata e nella sezione iniziale verrà oscurata la risorsa link al test iniziale.

- *locallib.php*

Si tratta di una libreria secondaria, non richiesta dal sistema Moodle, ma dove sono state implementate molte funzioni di vitale importanza per il corretto funzionamento del sistema LS-Plan.

- *KIToQuestion(\$section, \$idQuestion, \$course)*: raccoglie i dati necessari all'inserimento di un nuovo record nella tabella `adaptive_questions_ki`, creando un nuovo oggetto con i campi richiesti per legare le domande del quiz alla sezione in cui è contenuto;
- *acquisitoKI(\$qTestArray, \$userId, \$ki, \$courseId, \$idTestIniziale, \$answers)*: controlla se lo studente ha ottenuto un punteggio (tra tutte le domande associate al KI e contenute nel test iniziale) tale da considerare il KI già posseduto;
- *add_link_to_graph(\$courseid, \$coursename)*: aggiunge nella sezione iniziale la risorsa per il link al Learning Path del corso, visibile solo al docente;
- *associaKIQuestions(\$section, \$course, \$cmidQuiz)*: si occupa di legare ogni domanda contenuta nel quiz della sezione a tutte le risorse/attività presenti nella sezione stessa. Per fare questo sfrutta la funzione `hasQuiz` per ottenere l'id dell'istanza del quiz, così da poter ottenere la sequenza delle domande del test. Tramite un ciclo `for`, lega ogni domanda alla sezione in cui è contenuta, ossia l'id del LN, tramite la funzione *KIToQuestion(\$form, \$idQuestion)*;
- *call_lsplan(\$userid, \$section, \$courseid, \$score, \$fruitionTime)*: effettua la chiamata intermedia al sistema LS-Plan, ossia a seguito della pianificazione iniziale;
- *correction_initial_test(\$idTestIniziale, \$userId, \$courseId)*: effettua la correzione del test iniziale, ritornando l'insieme dei KI acquisiti (Cognitive State dello studente) sotto forma di stringa;
- *exist_test(\$section, \$course)*: ritorna un valore bool a seconda dell'esistenza o meno di un test per una risorsa, controllando se all'interno della sezione che contiene il materiale didattico è presente o meno un'istanza di quiz;

- *getKiCourse(\$course)*: ritorna un array che ha come chiave l'AK e come valori un oggetto che contiene l'AK e la sezione dei KI del corso, indicato da \$course;
- *getMinScore(\$idQuiz)*: ritorna il punteggio minimo ottenibile nelle domande contenute nel quiz con id \$idQuiz;
- *getQuestions(\$ki, \$qTestArray, \$courseId)*: ritorna un array di id della tabella question relativo alle domande inserite nel test iniziale e relative al KnowledgeItem \$ki;
- *getScoreMax(\$questions)*: restituisce il punteggio massimo ottenibile da tutte le domande di un KI contenute nel test iniziale e riportate nell'array \$questions;
- *getScoreStudent(\$questions, \$answers)*: restituisce il punteggio ottenuto dallo studente, relativamente alle domande di un KI contenute nel test iniziale;
- *getScores(\$form)*: restituisce il punteggio massimo e il voto massimo ottenibile dal quiz presente nella sezione, racchiusi all'interno dell'oggetto \$punteggiQuiz; -1 se nella sezione non c'è quiz. La sezione e l'id del corso sono contenuti nell'oggetto \$form;
- *get_Metadati_LN(\$idLN, \$idCourse)*: restituisce i metadati del LN con id uguale a \$idLN del corso \$idCourse. Per fare questo richiama il file Accesso.php con codice operazione 4, che permette di leggere dalla tabella learningnodes di LS-Plan, e legge da file i metadati, ossia una stringa in cui i metadati sono separati da “;”;
- *get_fruition_time(\$ln, \$userid, \$lastCallTime)*: ritorna il tempo di fruizione, in secondi, del LN con id della tabella mdl_course_modules \$ln; il tempo di fruizione è calcolato dal timestamp relativo al click sul link della risorsa fino alla visualizzazione dell'ultimo test consegnato ad essa associato (se presente);

- *get_last_call(\$user, \$course, \$cmid)*: restituisce l'istante di tempo dell'ultima chiamata a LS-Plan dovuta ad una risorsa con id pari a *\$cmid*, visitata da un utente con id *\$user* del corso con id *\$course*;
- *get_last_course_action(\$course, \$user)*: tramite un controllo incrociato tra log e *teacherassistant_attivita_corso*, ritorna l'ultima azione svolta dall'utente con id *\$user* nel corso con id *\$course*, al di fuori della sezione iniziale;
- *get_last_course_view(\$courseid, \$userid)*: restituisce il record della tabella *mdl_log* relativo all'ultima visualizzazione del corso;
- *get_min_fraction(\$question)*: ritorna il valore più penalizzante della domanda;
- *get_module_id(\$moduleName)*: restituisce l'id della tabella *mdl_modules* del modulo con nome *\$moduleName*;
- *get_record_section(\$courseid, \$section)*: ritorna il record della tabella *course_sections* della sezione *\$section* del corso *\$courseid*;
- *get_secondlast_course_view(\$courseid, \$userid, \$lastId)*: ritorna il record della tabella *mdl_log* relativo all'ultima attività del corso visualizzata;
- *get_scala(\$course)*: ritorna la scala di normalizzazione impostata dal docente per il corso con id *\$course*;
- *get_soglia(\$courseid)*: ritorna la soglia minima, compresa tra 0 e 1, per impostare il minimo punteggio da ottenere tra le domande del test iniziale relative a un KI per considerarlo già posseduto dallo studente prima di iniziare il corso indicato da *\$courseid*;

- *get_sumgrades_last_attempt(\$userid, \$idQuiz, \$courseid)*: restituisce il punteggio già normalizzato ottenuto dallo studente *\$userid* nell'ultimo tentativo del quiz *\$idQuiz*, nel corso *\$courseid*;
- *hasQuiz(\$section, \$course)*: individua e restituisce, se presente, l'id della tabella *course_modules* dell'istanza del quiz intermedio presente nella sezione, altrimenti ritorna -1;
- *insert_in_cm(\$cm, \$visible)*: completa l'oggetto *\$cm* per l'inserimento di una nuova istanza di un modulo nella tabella *mdl_course_modules*; tramite la variabile *\$visible* è possibile stabilire se tale istanza potrà essere visibile allo studente o no;
- *insert_testLS(\$courseid)*: aggiunge nella sezione iniziale un'istanza del modulo *testls*, visibile solo allo studente;
- *insert_acourse(\$courseid)*: aggiunge nella sezione iniziale un'istanza del modulo *acourse*, visibile solo al docente;
- *isLN(\$cmid)*: controlla se il modulo *\$cmid* passato come parametro è di un LN (una risorsa o qualsiasi attività non quiz contenuta all'interno di una sezione), in tal caso ritorna lo stesso *\$cmid*, altrimenti ritorna -1;
- *is_edit_questions(\$courseid, \$userid)*: controlla se sono state aggiunte o eliminate delle domande all'interno di un quiz intermedio;
- *normalizzaPunteggio(\$score, \$course, \$maxScore, \$votoMax)*: normalizza il punteggio che gli viene passato come parametro; se la scala è il voto, allora *\$score* viene normalizzato, altrimenti è già nella scala giusta;
- *first_call_lsplan(\$user, \$cs, \$recordLS, \$goal, \$course)*: effettua la prima chiamata al motore adattivo;

- *get_sequence(\$section, \$course)*: ritorna una stringa che rappresenta la sequenza di id dei record nella tabella *course_modules* delle istanze degli elementi che sono nella sezione;
- *getCmidTestIniziale(\$course)*: ritorna l'id del record della tabella *course_modules* relativo al test iniziale del corso, verificando il tipo di attività di ogni elemento della sezione 0, ritornando l'id dell'istanza che è di tipo "quiz";
- *get_goal(\$course)*: restituisce l'obiettivo formativo relativo al corso dalla tabella *acourse_goal_course*; se non presente si considera come obiettivo tutti i KI del dominio;
- *get_record_initial_test(\$course, \$userid)*: ritorna il primo record della tabella log relativo ad un quiz per un determinato corso, esso sarà sempre riferito al test iniziale;
- *read_output_lsplan(\$course, \$userid)*: ritorna un array contenente le tre righe del file di output di LS-Plan, ossia la prima riga riporta l'intera sequenza dei LN non personalizzati sullo studente, la seconda riga è la sequenza personalizzata e la terza riga riporta un eventuale messaggio di output di LS-Plan;
- *is_modified_section(\$courseid)*: verifica se è stato modificato il materiale didattico di una sezione del corso, eventualmente aggiorna la tabella *teacherassistant_attivita_corso*;
- *add_activity_course(\$cmid)*: aggiunge nella tabella *teacherassistant_attivita_corso* l'ultima attività creata nella sezione, in modo che venga considerata durante la personalizzazione per il calcolo del tempo di fruizione; questo avviene registrando *\$cmid* come materiale didattico del corso personalizzato;

- *check_ta(\$sequence)*: controlla se è presente un'istanza di TeacherAssistant nella sequenza di attività di una sezione.

- cartella db

Per il formato di corso Adaptive sono necessarie le tabelle: *adaptive_course_pers* per salvare l'ultima LOS generata da LS-Plan per ogni studente; *adaptive_log_lpslan* per tenere traccia delle chiamate giunte al motore adattivo; *adaptive_questions_ki* per associare le domande ai vari Knowledge Items, per definire poi il Cognitive State in base ai risultati del test iniziale.

- grafo.php

Un'altra libreria molto importante è il file *grafo.php*, che si occupa della gestione della visualizzazione grafica del Learning Path, sia del corso per quanto riguarda il docente sia quello relativo ad uno studente. Per la sua rappresentazione viene sfruttato un componente esterno, GraphViz [GrVi], un software open-source per rappresentazione di grafi. Sfruttando le definizioni contenute in un file *.dot* opportunamente formattato, GraphViz genera come output un'immagine (nel nostro caso si utilizza un file *.gif*) che descrive il grafo in maniera visuale.

Per generare e gestire tali file, sono state create delle funzioni apposite:

- *init(\$corso, \$nomeCorso, \$studentID)*: inizializza il file *.dot* relativo al corso di id *\$corso* e nome *\$nomeCorso*, sulla base di *\$studentID* che indica se ci si sta riferendo al Learning Path di uno studente o del corso;
- *addInit2dot(\$toAdd, \$filename)*: funzione di supporto ad *init*, serve per aggiungere le righe di intestazione del file *.dot* al momento della sua creazione;
- *add2dot(\$toAdd, \$filename, \$add)*: si occupa di scrivere fisicamente nel file *.dot*, indicato tramite *\$filename*, la riga *\$toAdd*;

- *inserisciCoppiaNelGrafo(\$ak, \$rk, \$corso, \$nomeCorso)*: inserisce nel file .dot del corso la riga di definizione del nodo *\$ak*, aggiungendo anche le eventuali righe relative agli archi con cui viene collegato ai suoi prerequisiti *\$rk*;
- *visualizzaGrafo(\$corsoId, \$nomeCorso, \$student)*: richiamato nella pagina web *visualizza_grafo.php*, recupera il file .gif relativo al corso o allo studente sulla base di *\$student* e la stampa nella pagina;
- *modificaGrafo(\$vecchioAK, \$nuovoAK, \$rk, \$corso)*: richiamata in caso di modifica ad un Learning Node, prima cancella la definizione di tale nodo tramite la funzione *cancellaNodo*, successivamente ne inserisce la versione aggiornata tramite *inserisciCoppiaNelGrafo*;
- *cancellaNodo(\$ak, \$corso)*: identifica le righe in cui è definito il nodo *\$ak* e tutti gli archi entranti nel nodo, cancellando tali righe dal file .dot; se vi sono più definizioni (ossia più materiali didattici riguardanti quel nodo), prima ne conta le occorrenze e richiama la funzione *get_colore*, che restituisce il colore del nodo che va eliminato;
- *get_colore(\$numero)*: restituisce un colore sulla base della legenda che assegna ad un determinato numero di materiali didattici relativi ad un Learning Node un colore diverso;
- *creaGrafoStudente(\$userid, \$corsoid, \$nomeCorso, \$los)*: gestisce l'inizializzazione del grafo per lo studente, inserendo tutti i nodi validi per uno studente tramite la funzione *rigaValidaStudente* e poi colorandoli in maniera opportuna in base alla sequenza *\$los* tramite la funzione *addColorNode*;
- *rigaValidaStudente(\$riga)*: verifica se la riga ricevuta in input sia valida per uno studente, ossia se rappresenti la definizione di un Learning Node che sia presente in almeno una sezione del corso;
- *addColorNode(\$file, \$los, \$color)*: richiama la funzione *add2dot*, passandogli una definizione di un nodo basata sui dati *\$los* e *\$color*;
- *aggiornaGrafoStudente(\$userid, \$corsoid, \$los)*: funzione per aggiornare l'immagine del Learning Path dello studente, che si appoggia alla funzione

`getGreenNodes` e alla variabile `$los` per modificare i nodi da segnare come appresi (verdi);

- `getGreenNodes($sorgente)`: ritorna la lista di tutti i nodi già segnati come appresi;
- `rigaGreen($riga)`: verifica se nella riga ricevuta in input vi sia la definizione di un nodo già appreso (controlla la presenza della stringa `color=green`).

2.2.5 LS-Plan

Il sistema LS-Plan è posto nella directory `/var/www/moodle/course/format/adaptive`, dove `/var/www` è la locazione in cui si trova l'istanza di Moodle: qui risiede sia il codice del motore adattivo e le classi che gestiscono il modello, sia il codice dell'algoritmo pianificatore.

L'attuale struttura del sistema ha creato un'astrazione del cuore del motore adattivo dal pianificatore, tramite una classe cuscinetto (`Utils.php`) che intercetta le chiamate provenienti da LS-Plan e si occupa personalmente di girare l'interrogazione per ottenere la Learning Objects Sequence: questo permette un basso accoppiamento tra i due sistemi.

Inseguendo questo obiettivo, si è cercato di creare dei livelli di astrazione tra le chiamate provenienti da Moodle e LS-Plan: il file `Lsplan.php` si occupa di raccogliere tutte le richieste esterne e processarle (sfruttando i metodi contenuti in `AdaptationEngine.php`), rendendo pubblico solo il suo metodo principale.

Stessa cosa si può dire lato database: la gestione di richieste di inserimento, aggiornamento e cancellazione dei Learning Node dall'esterno sono intercettate dal file `Accesso.php` che, in base ad un codice operativo, richiama la corrispondente funzione di `GestoreDB.php`, che è il solo in grado di accedere direttamente al database dedicato di LS-Plan.

- gestione del modello

In LS-Plan sono presenti alcune classi atte solo a fornire una rappresentazione di concetti utili ai fini del sistema, con una serie di funzioni per accedervi. Questi file sono:

- KnowledgeItem.php
- LearningObjectsSequence.php
- LearningNode.php
- StudentModel.php

Essi offrono principalmente funzioni per ottenere o impostare valori dei vari campi che ogni oggetto possiede.

- gestione del database di LS-Plan

Come anticipato, la gestione del database è unicamente affidata al file GestoreDB.php: è l'unico attore che ha le credenziali per accedere direttamente alle tabelle di riferimento di LS-Plan. Esso permette la creazione, la modifica e la cancellazione di elementi presenti nel database, quali Learning Node, Student Model (con metodi dedicati per i Learning Styles ed il Cognitive State), Learning Objects Sequence (da cui estrarre anche i nodi alternativi). Per le richieste dall'esterno di LS-Plan, il file Accesso.php offre una funzione unica, main(\$arg), che a seconda del codice operativo contenuto in \$arg la funzione per inserire, aggiornare, cancellare o aggiornare i metadati di un Learning Node.

- Lsplan.php

E' l'interfacciamento verso l'esterno del sistema LS-Plan: offre anch'esso un'unica funzione pubblica, main(\$values), che effettua un parsing della stringa ricevuta in input spezzandola in un array grazie al carattere spazio (questo sarà oggetto di modifica, come descritto nel capitolo 3); questo permette di ottenere i dati d'interesse (nome, cognome e matricola dello studente, Cognitive State, i valori dei Learning Styles, obiettivo formativo del corso) ed un codice operativo: 1 indica la creazione del modello dello studente e chiamare per la prima volta il motore adattivo; 2 indica un aggiornamento del modello dello studente e della Learning Objects Sequence.

Gli altri metodi presenti sono:

- *createStudentModel(\$matricola,\$startingknowledge,\$startingknowledgelevels,\$ls1, \$ls2, \$ls3, \$ls4, \$nome, \$cognome, \$targetknowledge, \$targetknowledgelevels, \$corso)*: richiamata dalla funzione main di Lsplan.php, serve a creare il modello dello studente, chiamando il motore adattivo AdaptationEngine.php e passandogli i parametri ricevuti in input, trasformandoli da stringhe in opportuni array;
- *createFirstLearningObjectsSequence(\$sm, \$corso)*: si occupa di chiamare due funzioni di AdaptationEngine, createFirstLearningObjectsSequence e getLOScompleta, per ottenere e scrivere su file la Learning Objects Sequence prodotta dal motore adattivo;
- *update(\$idStudente, \$fruitiontime, \$score, \$corso, \$idnode)*: si occupa di aggiornare lo stato dello studente dopo ogni sua azione all'interno del corso; se si notano cambiamenti di segno nelle dimensioni dei LS, si ricalcola la LOS; se il LN è stato acquisito, viene salvato su un file un messaggio da stampare per lo studente, che verrà poi recuperato da format.php di Adaptive.

- AdaptationEngine.php

Il cuore pulsante di LS-Plan, il motore adattivo, ha sede nel file AdaptationEngine.php: qui vi sono tutte le funzioni principali che implementano le teorie descritte nel capitolo precedente. Analizziamo il suo contenuto:

- *createStudentModel(\$matricola, \$startingKnowledge, \$levels, \$learningStyles, \$nome, \$cognome, \$goal, \$goalLevels, \$corso)*: si occupa dell'inizializzazione del modello dello studente; esso crea un oggetto StudentModel e lo popola con i dati di interesse ottenuti in input; successivamente passa tutti i dati al GestoreDB, affinché possa salvarli nel database;

- *createFirstLearningObjectsSequence(\$sm, \$corso, \$firstTime)*: gestisce la creazione della prima Learning Objects Sequence, richiamando l'algoritmo pianificatore tramite il file Utils.php e la sua funzione *callAlg(\$studentModel, \$corso)*; genera sia la LOS completa del corso, considerando un Cognitive State iniziale vuoto, corredata di Learning Node alternativi, sia la LOS dello specifico studente tramite la funzione *checkClosestNode*; la variabile *\$firstTime* è impostata ad 1 per un nuovo studente, 2 se è richiesta una ripianificazione del percorso didattico;
- *createStudiedLearningNode(\$idLN, \$fruitionTime, \$obtainedScore, \$corso)*: richiama la funzione di GestoreDB per ottenere e restituire il Learning Node con id pari a *\$idLN*, a cui aggiunge i dati ottenuti in input;
- *updateStudentModel(\$currentSM, \$studiedLN, \$los)*: aggiorna il modello dello studente; per farlo prima controlla se il LN passato come parametro sia stato appreso: in tal caso si aggiorna lo StudentModel nel database e si modifica la LOS, non consigliando più il nodo appreso ed i suoi nodi alternativi; se ci sono test collegati al nodo, si calcola l'aggiornamento dei LS come spiegato ampiamente nel capitolo precedente; successivamente, si controlla se ci sono cambiamenti di segno nei parametri dei LS ed eventualmente lo si notifica nello Student Model corrente; infine, il nuovo Student Model viene aggiornato anche nel database tramite l'apposita funzione di GestoreDB;

- *updateLearningObjectsSequence(\$currentSM, \$studiedLN, \$currentLOS, \$corso)*: si occupa di aggiornare la sequenza di Learning Object dello studente; come primo passo controlla se non ci sono state variazioni di segno nei LS, specialmente lo studente è passato da sequenziale a globale (o viceversa), si necessita una ripianificazione della LOS; altrimenti, modifico la LOS attuale, cercando di consigliare dei Learning Node più vicini allo stile di apprendimento dello studente; se invece non ho avuto cambiamenti significativi nei LS, controllo prima se il tempo di studio speso dallo studente è congruo a quello indicato dal docente per quel LN: in caso negativo, ripropongo la stessa LOS; se invece il tempo dedicato allo studio rientra nei parametri, il sistema cerca un LN equivalente tramite la funzione *checkClosestNode* e lo propone nella LOS; se questa ricerca non va a buon fine, sfrutta la funzione *orderedPredecessorsList* per trovare i nodi predecessori da riproporre allo studente e da eliminare dal Cognitive State, aggiornando così la LOS nel database; la funzione restituisce un messaggio per lo studente sull'operazione intrapresa dal sistema;
- *checkClosestNode(\$currentSM, \$currentLN, \$firstTime)*: verifica l'esistenza di un LN equivalente a *\$currentLN*, ossia con gli stessi RK ed AK, sceglie quello più vicino ai LS dello studente (tramite la funzione *nearestLearningNode*) e non visitato in precedenza;
- *nearestLearningNode(lnVicini, \$learningStylesSM)*: utilizzando la definizione di distanza euclidea tra nodi, di cui si è discusso nel capitolo 1, cerca il nodo più vicino ai LS dello studente tra quelli passati come input;
- *orderedPredecessorsList(\$currentLN, \$currentSM, \$corso)*: recupera dal database i Learning Node prerequisiti di *\$currentLN*, tali dunque che i loro AK siano RK di *\$currentLN*, dando priorità ai nodi non visitati e poi a quelli che hanno dei test e sono vicini ai LS dello studente;

- *getStudentModel(\$idstudente, \$corso)*: recupera dal database le informazioni per ritornare lo StudentModel dello studente con id pari ad *\$idstudente*;
- *getLOS(\$idstudente, \$complete, \$corso)*: ritorna la Learning Objects Sequence dello studente; se *\$complete* vale 1, restituisce la LOS raccomandata, se vale 0 restituisce la LOS completa.

- AlberoXML.php

Questo file serve per prelevare i metadati dei nodi dal database di LS-Plan e poterli poi usare per il parser dell'algoritmo pianificatore. Un oggetto AlberoXML ha tra i suoi campi una lista contenente tutti i Learning Node e le loro informazioni.

- Utils.php

Offre una sola funzione, *callAlg(\$sm, \$corso)*, che si occupa di richiamare l'esecutore dell'algoritmo pianificatore e di restituire la Learning Objects Sequence in una stringa, dove i vari id dei nodi sono separati da una virgola.

- Algoritmo Pianificatore

Esso viene implementato tramite tre file .php: la classe principale, il parser e l'esecutore. La classe principale si occupa dell'algoritmo principale di ricerca topologica, come ampiamente descritto nel paragrafo 1.2.2. Un suo oggetto ha come campi un parser, un \$target che rappresenta l'obiettivo formativo del corso, un campo \$presentation che rappresenta la LOS da riportare poi a LS-Plan. L'algoritmo viene avviato non appena creato l'oggetto, tramite la funzione *start()*: essa richiama la funzione *visitaPerR(\$n, \$cs)*, che rappresenta il vero algoritmo, per ogni nodo appartenente al \$target. Essa lavora a stretto contatto con la funzione di controllo *check(\$n, \$cs)*, che va alla ricerca di possibili inconsistenze tra i nodi del corso. Elementi di supporto sono la classe parser, che aiuta l'algoritmo nella visita dei vari nodi tramite la funzione *searchNodeForAK(\$ak)*, che a sua volta sfrutta un oggetto di

tipo `AlberoXML`; per finire, l'utilizzo di una classe per l'esecuzione dell'algoritmo aiuta a ridurre l'accoppiamento del pianificatore col resto del sistema, poiché istanziando un suo oggetto si ottiene come risultato la LOS stessa, facendo risultare il processo di creazione sequenza del tutto trasparente dall'esterno.

Capitolo 3

Raffinamento del sistema Moodle_LS

3.1 Introduzione al lavoro svolto

La prima parte del tirocinio consiste nell'implementazione di modifiche già svolte in passato, ma che avevano portato ad una forte frammentazione del sistema, con diverse versioni di Moodle_LS, ognuna con caratteristiche diverse. Si necessitava di un accorpamento delle novità create finora in un'unica istanza.

Le modifiche principali che sono state implementate sono:

- la possibilità di inserire degli spazi nei nomi dei Learning Node, azione inizialmente non permessa visto che il sistema utilizzava proprio il carattere spazio nel parsing delle stringhe di metadati o nei comandi che giungevano ad LS-Plan;
- l'eliminazione dal dominio della conoscenza di LN non più utilizzati da alcuna sezione e che non siano prerequisito di nessun altro nodo;
- un controllo utile per il docente per trovare eventuali prerequisiti ancora non definiti all'interno del corso.

Sono state, inoltre, aggiunte delle migliorie nella gestione del grafo e nella visualizzazione della sezione iniziale del corso, frutto di problematiche considerate nel corso del tirocinio: nel resto del capitolo verranno descritti in dettaglio tutti gli interventi.

3.2 Spazi nei nomi delle conoscenze

Come descritto nel capitolo 2, il sistema LS-Plan interagisce con Moodle attraverso dei parametri di tipo stringa, che seguono delle regole di formattazione ben precise. Il carattere spazio delimita ogni campo passato come input, la virgola separa gli id dei vari Learning Node nella LOS, il punto e virgola indica eventuali nodi alternativi da consigliare. Questo comporta che l'utente debba inserire nei campi delle form che gli vengono proposte dei dati che non interferiscano con queste regole: a tal proposito, sono state introdotte delle regole di validazione delle form, prima del tutto assenti.

In Moodle si possono aggiungere delle regole di validazione di un form in due modi:

- definizione di una regola all'interno della pagina, tramite la funzione *addRule(\$element, \$message, \$type, \$format, \$validation, \$reset, \$force)*, che si applica ad un elemento *\$element* della form; il campo *\$message* riporta il messaggio da mostrare all'utente in caso di violazione della regola, *\$type* è il tipo di regola (nel nostro caso sarà un'espressione regolare), *\$format* è la nostra espressione regolare di verifica, *\$validation* indica se sarà il server o il client a doversi occupare della validazione, *\$reset* indica la possibilità di resettare l'elemento al suo valore di default in caso di errore e di validazione lato client, *\$force* forza l'applicazione della regola in tutti i casi;
- il secondo sistema prevede di inserire dei controlli personalizzati all'interno della funzione *validation()*, che viene eseguita ogni volta che si clicca sul tasto di invio della form al server.

Si è scelto la prima opzione, quindi sono state inserite delle regole nei vari campi a rischio, per evitare che l'utente possa utilizzare caratteri dal significato particolare per il sistema.

La modifica è stata applicata nel file *mod_form.php* di TeacherAssistant: nella funzione *definition()* sono state aggiunte le regole per i campi in cui vengono definiti nuove Required Knowledge, nuove AcquiredKnowledge e nei campi dedicati al tempo di fruizione minimo

e massimo. Un esempio di applicazione della verifica su un campo, in questo caso si richiede che il tempo minimo di fruizione del LN sia definito e non nullo:

```
$mform->addRule('min_time', null, 'required', null, 'client');
```

Successivamente si è considerata la possibilità di abilitare gli spazi nei nomi delle conoscenze, che era una forte limitazione per l'utente, sostituendo il carattere spazio con il carattere ^; per questo si sono dovute apportare modifiche a molte funzioni del sistema che sfruttavano il carattere spazio come elemento di parsing. I file che hanno subito delle modifiche sono:

- /mod/acourse/lib.php
- /mod/teacherassistant/mod_form.php
- /course/format/adaptive/grafo.php
- /course/format/adaptive/locallib.php
- /course/format/adaptive/lsplan/src/Lsplan.php
- /course/format/adaptive/lsplan/src/Accesso.php

Nel file mod_form.php di TeacherAssistant è stata aggiunta la seguente regola al campo di definizione di una nuova conoscenza acquisita:

```
$mform->addRule('newAK', 'I caratteri: ^ - , - ; non sono concessi nella definizione di conoscenze', 'regex', '/^[^\^,;]+$', 'client', true, false);
```

Ora il sistema, nel caso di inserimento di uno dei caratteri “^” o “,” o “;”, cattura la violazione della regola e mostra un avviso all'utente.

Violazione della regola sull'inserimento di caratteri speciali di sistema

Particolare attenzione ha richiesto il file grafo.php, che si occupa della creazione e della gestione dell'illustrazione visiva del Learning Path, sia del corso che dello studente.

Uno dei problemi che si sono venuti a creare dalla possibilità di inserire degli spazi nei nomi delle conoscenze, ma che non era stato precedentemente corretto, riguarda proprio la generazione e modifica dei file .dot. Per capire meglio il problema, analizziamo un esempio di come si definisce un nodo in un file .dot:

```
"nome_nodo" [color=black,style=continue] ;
```

Mentre un arco viene definito:

"nodo_padre" → "nodo_figlio" [label="nodo_padre"] ;

Visto il precedente divieto di usare spazi nella definizione dei nodi e visto che nei file .dot vi è uno spazio tra il nodo e le sue caratteristiche o la → che indica l'arco, si era deciso di utilizzare il carattere spazio come elemento per il parsing del file.

L'introduzione degli spazi nei nomi delle conoscenze ha richiesto una modifica importante del file grafo.php: per l'occasione sono state introdotte delle funzioni ad hoc per il parsing delle righe dei file .dot, per migliorare la coesione e togliere codice ripetuto in più parti del file, che rendevano decisamente difficoltosa la manutenibilità dello stesso.

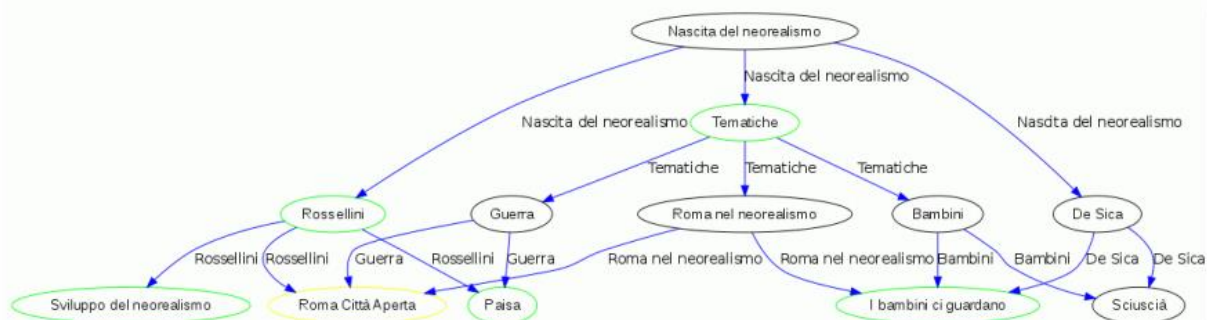
Ecco la lista delle nuove funzioni, inerenti il parsing del file.dot:

- *isArc(\$riga)*: verifica se la riga ricevuta come input sia un arco, provando a cercare al suo interno la stringa “ → “;
- *isNode(\$riga)*: verifica se la riga ricevuta come input sia un nodo, provando a cercare una caratteristica comune a tutti i nodi del grafo;
- *getNodeFromLine(\$riga)*: ricava dalla riga data in input il nome del nodo, andando a tagliare la riga non appena trova una definizione (indicata tra parentesi quadre);
- *getDefFromLine(\$riga)*: ricava dalla riga ricevuta come input la definizione del nodo, andando ad estrapolarla dalle parentesi quadre;
- *getParentFromLine(\$riga)*: verifica che la riga in input si riferisca ad un arco e ne ricava il nodo genitore;
- *getSonFromLine(\$riga)*: verifica che la riga in input si riferisca ad un arco e ne ricava il nodo figlio;
- *existsNode(\$riga, \$ak)*: verifica che la riga in input si riferisca alla definizione del nodo dal nome *\$ak*;
- *testToDelete(\$riga, \$ak)*: verifica che la riga sia un arco e abbia come nodo figlio *\$ak* (serve in fase di cancellazione di un nodo).

Queste funzioni risolvono il problema che si presentava nel parsing dei file .dot: come si può notare, ora i nodi possono avere nomi con spazi.

This is the learning path for the course Cinema Neorealista Italiano; to view the learning materials about a topic, click the relative node.

Graph colors:
 No didactic material available
 1 didactic material
 2 didactic materials
 3 didactic materials
 4 didactic materials



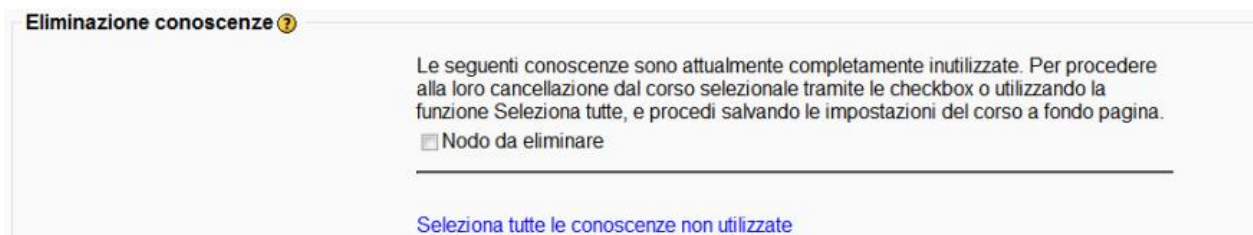
I nodi nel grafo ora possono contenere spazi

3.3 Eliminazione delle conoscenze

Negli stadi precedenti dello sviluppo di Moodle_LS ogni Learning Node introdotto nel dominio della conoscenza era destinato a rimanere presente, anche se fosse del tutto inutilizzato, quindi sia non presente in alcuna sezione sia non richiesto da alcun nodo come suo prerequisito.

Il concetto dietro questa scelta è lasciare la possibilità al docente di recuperare un Learning Node precedentemente definito, ma non più utilizzato, e poterlo riproporre nel suo corso: i nodi vengono tutti assegnati alla sezione 0. Ciò, comunque, può portare ad un accumulo di nodi non più utilizzati, che comunque venivano mostrati nel grafo con un bordo rosso tratteggiato.

Per stabilire se una conoscenza non è più utilizzata all'interno del corso, come prima cosa si ricercano tutte le conoscenze associate alla sezione 0, successivamente si verifica tra queste quali non sono prerequisito di nessuna conoscenza appartenente al corso ma con sezione diversa da 0; è stata aggiunta una sezione nel modulo Acourse che permetta, con delle checkbox, la selezione di quali conoscenze eventualmente cancellare definitivamente.



Sezione nell'istanza di Acourse dove è possibile selezionare le conoscenze da eliminare

L'intervento ha richiesto la definizione di alcune funzioni di supporto e la modifica del file `/mod/acourse/mod_form.php` per inserire la sezione mostrata nell'immagine. Ecco una rapida lista:

- modificata la funzione `acourse_update_instance($acourse)` per gestire l'eliminazione delle conoscenze;
- `get_unused_ki_course($courseID)`: ritorna tutte le conoscenze risultate non utilizzate attualmente e non definite come prerequisito di altre conoscenze;
- `get_ki_dependent_upon($courseID, $ki)`: ritorna un array contenente i Knowledge Item utilizzati nel corso che sono dipendenti da `$ki`;
- `delete_ki_course($courseID, $ki)`: elimina definitivamente il nodo dal corso e dal suo file `.dot`.

3.4 Controllo su prerequisiti non definiti

Col crescere della dimensione del corso, il docente potrebbe avere qualche difficoltà nel tenere sotto controllo tutti i legami tra le conoscenze, quindi esiste la possibilità di aver specificato un determinato prerequisito per un nodo, senza che sia effettivamente presente (o mai definito o eliminato e non inserito in una nuova sezione).

E' stata ideata, per tanto, una sezione nella pagina delle impostazioni del corso adattivo (`/mod/acourse/mod_form.php`) che viene mostrata se tale controllo restituisce delle

conoscenze al momento non definite. Il sistema segnala all'utente docente il nome della conoscenza richiesta e la sezione in cui tale conoscenza è segnata come prerequisito.

Per il controllo sono state definite alcune funzioni di supporto, qui elencate:

- *get_unconsistent_kis(\$courseID)*: ritorna un array contenente i KI inconsistenti, che sono quindi indicati come prerequisiti di altri KI ma mai indicati come conoscenze acquisite; come prima cosa cerca tutti KI che hanno dei prerequisiti definiti, successivamente per ogni KI con RK verifica che ogni RK che gli appartiene sia definito come AK di una sezione diversa da 0, altrimenti lo classifica come inconsistente;
- *get_rk(\$courseID, \$ki)*: ritorna un array contenente tutti gli RK del KI passato nel corso indicato;
- *get_section_dependent_upon(\$courseID, \$ki)*: ritorna un array contenente le sezioni del corso che hanno il KI passato come prerequisito.

Ecco un esempio di come appare la pagina `mod_form.php` di Acourse se la verifica trova delle incongruenze.

Controllo di consistenza ?

Attenzione! Le seguenti conoscenze sono indicate come prerequisiti di altre conoscenze ma non sono definite come conoscenza acquisita di nessuna sezione. Questo potrebbe pregiudicare la capacità di fruire di alcune sezioni del corso. Si consiglia di modificare i metadati relativi alle conoscenze indicate o di reintrodurre una risorsa che abbia come conoscenza acquisita quella mancante.

Conoscenza non definita richiesto da: Sezione 1

Sezione sul controllo di prerequisiti non definiti nel corso

3.5 Visualizzazione condizionale della sezione iniziale del corso

Prima dell'intervento effettuato in questo tirocinio, la sezione iniziale di un corso, per uno studente che si era appena iscritto, poteva risultare ambigua: venivano mostrate insieme le istanze del test iniziale, del suo Learning Path (nonostante non fosse ancora generato), del forum del corso e, se non l'avesse già compilato, il questionario sui Learning Styles. Per chi non avesse conosciuto a fondo il sistema o se non ci fosse stato scritto esplicitamente nella sezione, lo studente non avrebbe mai saputo la corretta sequenza con cui consultare tali istanze, generando così un giudizio personale negativo nei confronti del sistema. Ciò è fortemente sconsigliato, quindi si è cercato di rendere intuitivo il percorso da intraprendere, mostrando passo passo solo le risorse che sono necessarie per l'avvio della personalizzazione del corso e della modellazione dello studente.

Dopo l'intervento eseguito, uno studente che si è appena registrato nel sistema ed ha appena confermato l'intenzione di voler seguire un corso, si trova di fronte solo il link al test per i Learning Styles, come si vede nell'esempio.


Ti invitiamo, prima di iniziare il corso, a darci alcune informazioni su come ti piacerebbe imparare, cioè sul tipo di materiale didattico che preferiresti avere a disposizione. Inoltre devi eseguire il test d'ingresso per verificare le conoscenze che già hai sull'argomento.

 [How I'd like to learn](#)

Uno studente appena iscritto deve prima compilare il test per i LS.

Una volta compilati, il sistema nasconde il test per i LS e mostra la fase successiva: il quiz iniziale del corso.

Ti invitiamo, prima di iniziare il corso, a darci alcune informazioni su come ti piacerebbe imparare, cioè sul tipo di materiale didattico che preferiresti avere a disposizione. Inoltre devi eseguire il test d'ingresso per verificare le conoscenze che già hai sull'argomento.

 [Test d'ingresso](#)

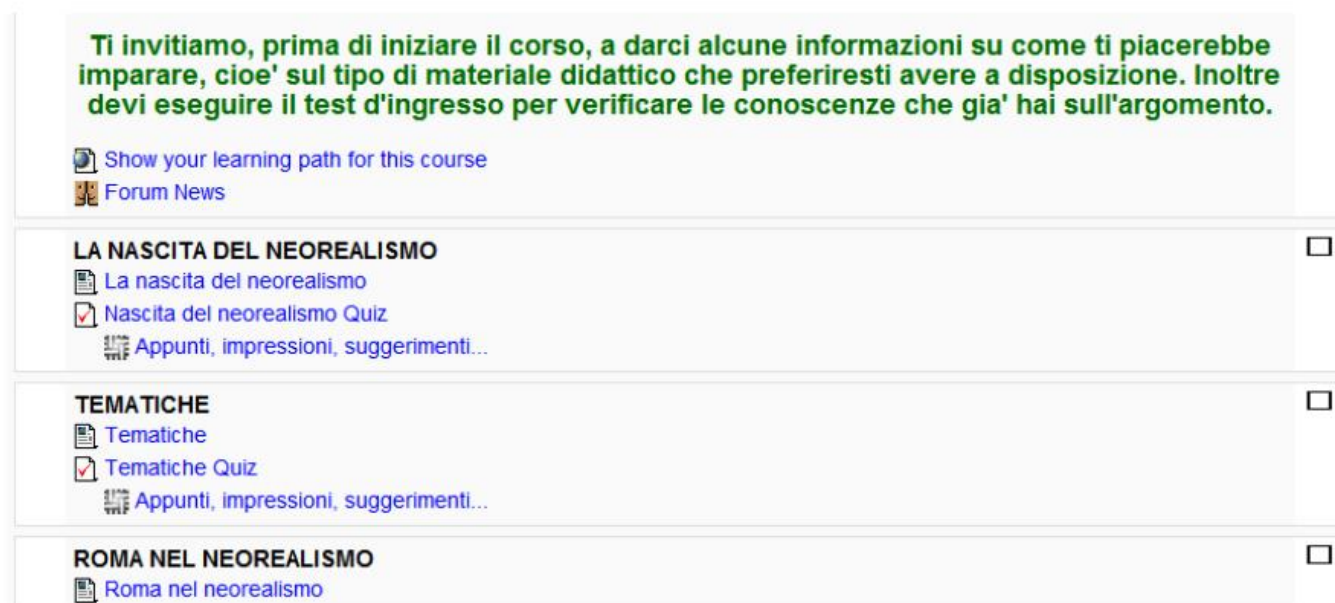
Un corso non può iniziare se prima non viene effettuato il quiz iniziale.

Ora il sistema ha tutti i dati per la personalizzazione del corso: viene quindi nascosto anche il link al quiz iniziale, viene mostrato il resto della sezione iniziale (link al forum, al grafo del proprio Learning Path e le altre eventuali risorse) ed il corso, risultato del processo di personalizzazione di LS-Plan.


Per ottenere questo risultato si sono dovute effettuare alcune modifiche: è il file `format.php` di `/course/format/adaptive` che si occupa degli elementi da mostrare nella pagina principale del corso ed erano stati già implementati dei controlli sulla sezione 0, in modo tale che l'istanza di TestLS e del quiz iniziale fossero nascoste non appena completate.


L'intervento si è concentrato nel nascondere le istanze presenti nella sezione quando non necessarie: questo risultato è facilmente ottenibile, visto che Moodle offre una variabile di sistema, l'array `$mods`, che contiene le istanze dei moduli che compongono la sezione.

Selezionando l'elemento nell'array con chiave pari all'id dell'istanza del modulo e ponendo il suo contenuto pari a " (stringa vuota), si può nascondere tale istanza nella visualizzazione finale della pagina; nel codice avremo dunque `$mods[$cm->id]="`.





Ti invitiamo, prima di iniziare il corso, a darci alcune informazioni su come ti piacerebbe imparare, cioè' sul tipo di materiale didattico che preferiresti avere a disposizione. Inoltre devi eseguire il test d'ingresso per verificare le conoscenze che già' hai sull'argomento.


 [Show your learning path for this course](#)

 [Forum News](#)


LA NASCITA DEL NEOREALISMO ☐


 [La nascita del neorealismo](#)


 [Nascita del neorealismo Quiz](#)

 [Appunti, impressioni, suggerimenti...](#)


TEMATICHE ☐

 [Tematiche](#)

 [Tematiche Quiz](#)

 [Appunti, impressioni, suggerimenti...](#)

ROMA NEL NEOREALISMO ☐

 [Roma nel neorealismo](#)

Il corso viene visualizzato e nella sezione iniziale vi sono solo i link al forum, al grafo del proprio Learning Path e le altre eventuali risorse.

3.6 Evitare la ripetizione dei quiz

Una delle peculiarità di Moodle_LS, la netta divisione tra il sistema LS-Plan ed il Learning Management System, portava ad un'incoerenza nella gestione dei quiz.

Visto che la soglia per considerare come appreso un Learning Node è salvata nel database di LS-Plan, mentre per i test si sfrutta il modulo Quiz di Moodle, il sistema non si accorgeva che lo studente aveva appena superato una prova intermedia finché non fosse stata ricaricata la pagina principale del corso, nella quale sono inseriti i controlli e le chiamate al sistema adattivo. Quindi, a fine di un test veniva comunque proposta la possibilità di ritentarlo immediatamente (se vi erano ancora dei tentativi disponibili), anche se si era ottenuto il massimo del punteggio.

Per risolvere il problema, si è dovuto modificare il modulo Quiz, contenuto nella directory /mod/quiz, intervenendo sui file locallib.php e view.php, ma anche modificando il file locallib.php di /course/format/adaptive e GestoreDB.php di /adaptive/lsplan/src:

- in /mod/quiz/locallib.php è stata creata la funzione *check_threshold(\$idQuiz, \$course)*, che si occupa di recuperare l'id della sezione di cui fa parte il quiz e chiamare la funzione *getThreshold* per ottenere e poi restituire al sistema la soglia richiesta per quel LN;
- nella libreria /course/format/adaptive/locallib.php è stata creata la funzione *getThreshold(\$idLn, \$corso)*, che interroga il database di LS-Plan attraverso GestoreDB per ottenere la soglia del nodo *\$idLn*;
- in /adaptive/lsplan/src/GestoreDB.php è stata creata la funzione *getSoglia(\$idLn, \$corso)* che recupera dal database il Learning Node *\$idLn* e ne restituisce la soglia;
- in /mod/quiz/view.php è stata introdotta la chiamata alla funzione *check_threshold* e viene fatto il controllo tra il miglior voto ottenuto finora (salvato nella variabile già presente *\$mygrade*) e la soglia del nodo; se il voto è maggiore o uguale della soglia, non viene stampato il bottone per riprovare il corso (*\$buttontext = ""*), ma

viene stampato il bottone per tornare alla pagina principale del corso, tramite la funzione `print_continue($CFG->wwwroot . '/course/view.php?id=' . $course->id)`.

Nascita del neorealismo Quiz
Metodo di valutazione: Voto più alto
Riepilogo dei tuoi tentativi

Tentativo	Completato	Punti / 3	Valutazione / 10
1	mercoledì, 13 febbraio 2013, 01:25	3	10

Voto più alto: 10 / 10.

Continua

Non viene più data la possibilità di ritentare il quiz se il voto è sufficiente per conseguire il Learning Node.

Capitolo 4

Approfondimenti sul TeacherAssistant

4.1 L'architettura a tre livelli

Questo progetto si basa sul libro di Steve Prettyman [Pret16]. In questo progetto si sono utilizzati i linguaggi PHP, HTML5, CSS3, JavaScript.

Si suddivide in 3 livelli, il livello delle regole operative che elabora tutte le informazioni e i dati ricevuti, il livello dell'interfaccia e il livello dei dati. Questo livello restituisce le informazioni richieste dal livello dell'interfaccia e le invia al livello dei dati.

Il livello del codice operativo risiede sul server ed è nascosto all'utente.

Il livello dei dati, che utilizza istruzioni SQL, memorizza i dati in un database oppure restituisce informazioni al livello delle regole operative. Il livello dell'interfaccia comprende il programma `e65metadato_interface.php`. Il livello delle regole operative comprende la classe `e65metadato.php`. La classe `metadato_data` fa parte del livello dei dati. Ogni richiesta di comunicazione del livello delle regole operative proveniente dall'interfaccia deve passare attraverso `metadato_container`, ed ogni richiesta di informazioni di dati da parte del livello delle regole operative deve passare da `metadato_container`. I livelli sono indipendenti e le modifiche apportate ad un livello non influenzano sugli altri livelli.

4.2 Le progettazioni MVC e modulare

La progettazione MVC è quella con un model, una vista ed un controller: interagiscono tra loro e la vista riassume delle informazioni controllate dal controller e che riguardano un modello.

La programmazione modulare permette il riuso di parti di codice che sappiamo che funzionano bene, questo permette un risparmio di tempo e di errori.

4.3 Sicurezza e prestazioni

Le tecniche di programmazione utilizzate sono all'avanguardia nel campo della programmazione a oggetti per la sicurezza e le prestazioni; per quanto riguarda la sicurezza in particolare possiamo solo cercare di farla sicura il più possibile. Per verificare e filtrare i dati inseriti dall'utente si sono inserite le funzioni set a oggetti. Abbiamo anche la creazione dei file di log (registri) degli errori.

PHP è un linguaggio procedurale, ad oggetti e open-source; offre la possibilità di utilizzare molte librerie per l'utilizzo di funzioni che agevolano lo sviluppatore; il sito ufficiale di php è php.net . PHP è un linguaggio per script, cioè non usa un compilatore, il codice viene interpretato riga per riga dal calcolatore, ma resta in memoria sul server-web, e quello molto utilizzato è Xampp.

Per quanto riguarda il linguaggio SQL, usiamo il Database Management System (DBMS MySql) che elabora le istruzioni. Per connettermi al DBMS ho utilizzato sia Git CMD da riga di comando, sia un plugin di Moodle, Adminer, che offre una GUI molto semplice da usare.

Come editor ho usato sia Visual Studio Code, sia Notepad++.

PHP ed AJAX (Asynchronous JavaScript and XML) collaborano tra loro. Ajax permette di eseguire del codice senza dover ricaricare la pagina che è costituita da diverse parti statiche, per esempio quando si clicca su un pulsante, questo evento non fa cambiare il resto della pagina.

L'applicazione è stata suddivisa in moduli: come l'automobile è costituita da molte parti che sono state poi assemblate tutte insieme, così anche questa grossa applicazione è costituita da diverse parti, e se si danneggia una parte, come per l'automobile, si sostituisce solo quel pezzo, nel nostro caso dei blocchi di codice.

Gli oggetti sono blocchi di codice già compilati per una applicazione che si possono riutilizzare, inserendoli in un programma. Questo programma riduce al minimo le informazioni scambiate tra utente e server ed inoltre filtra i dati.

Il programma client utilizza la Dependency injection, dove un blocco di codice per lavorare non deve conoscere l'implementazione di determinate classi, non ne conosce neanche il nome.

4.4 Gestione delle eccezioni

Vengono introdotti i metodi try e catch per controllare eventuali errori o eccezioni. Le eccezioni vengono gestite dal file interface, e se il programma non trova la clausola catch, sale di livello e va a cercare il blocco catch nel programma chiamante e se non lo trova lì allora decide che è l'ambiente stesso che deve gestire l'eccezione. Così è possibile lanciare eccezioni nei moduli metadato_container e metadato senza utilizzare catch. Nella interface c'è un blocco che avvolge le chiamate a questi file. Il livello delle regole operative e il livello dei dati passano i messaggi di errore all'interfaccia, che di volta in volta li gestisce a seconda dell'importanza.

4.5 Migliorie del plugin

Il plugin è costituito anche dai seguenti file che ho aggiunto: header.php, modulo_dati.php, footer.php, e65metadato_interface.php, main_interface.php, e65metadato_container.php, e61metadato_applications.xml, e65metadato.php, e63metadato_data.php, emetadato_data.xml, 02252021105833change.log, Errors.log.

Ora ne faccio una breve spiegazione:

-header.php

Ha le seguenti funzioni: mysqli_connect per connettersi al database usa la posizione del server, le credenziali di accesso al db, la password per il db e il nome del db; crea la tabella mdl_teacherassistant_tempo con tutti i dati che l'utente introduce nel form; mdl_teacherassistant_metadato mi serve per avere valori non ripetuti nel form alla voce prerequisiti e conoscenze acquisite, ed inoltre se non inserisco nella casella di testo del form una nuova conoscenza acquisita non devo aggiornare la tabella; mdl_teacherassistant_concatena e' una tabella di appoggio che mi serve per non avere due o piu' righe uguali nella tabella mdl_teacherassistant_modulo.

-modulo_dati.php

Il livello dell'interfaccia che si ha con questo file visualizza il modulo da compilare da parte dell'utente con una interfaccia grafica per l'utente (GUI); nella progettazione MVC rappresenta la parte vista; qui c'e' un controllo css per vedere se javascript e' abilitato, quindi i moduli sono due: se javascript non e' abilitato, il primo modulo con funzioni javascript resta nascosto e si vede solo il secondo che ne e' privo.

Le verifiche dei dati vengono eseguite nel browser dell'utente e non sul server, questo per migliorare le prestazioni infatti cosi' e' piu' veloce e se si facessero sul server, ogni volta si dovrebbe ricaricare la pagina perdendo le informazioni già inserite.

Solo nel campo `metadato_name` si può inserire del codice nascosto, ma le funzioni di clean (pulizia) toglierebbero tutto le parti che non sono caratteri alfabetici, come ad esempio le parentesi acute.

Nel pulsante submit si ha che ogni volta che devo chiamare l'applicazione Metadato, devo ricorrere all'interfaccia. Si comunica con tutte le classi attraverso l'interfaccia.

-footer.php

In questa pagina le informazioni ricevute dal livello dei dati vengono prima manipolate e poi visualizzate in una tabella.

Come formato di file per lo scambio di informazioni utilizziamo JSON ed XML.

Il progetto è stato sottoposto a diverse revisioni volte sempre a migliorarlo nei bisogni, nei problemi di sicurezza, nei problemi logici.

-e65metadato_interface.php

Visualizzerà un riepilogo dei dati inseriti, con un link per tornare alla pagina del form, ed è un controller in quanto ha delle funzioni che controllano i dati inseriti; si utilizzano i metodi get definiti nella classe Metadato. Fornisce l'interfaccia per tutte le parti dell'applicazione metadato.

Metodi presenti nel file: `mysqli_connect` apre una nuova connessione al server MySQL; die che se e' presente un errore nella connessione, il processo si deve fermare e mostrare l'errore; `clean_input` rende il codice pulito; `is_array` trova se una variabile è un array; `implode` unisce gli elementi dell'array con una stringa separandoli da un virgola; `htmlentities` rimuove qualsiasi codice HTML dalla stringa e lo trasforma nel formato '<'; `strip_tags` elimina i tag HTML e PHP da una stringa; `str_ireplace` sostituisce tutte le occorrenze dei 'caratteri non voluti(\$bad_chars)' con la stringa nulla; `setException` imposta le eccezioni, cioè `setException` e' una classe figlia che estende la classe `Exception`: con la parola chiave `extends` eredita tutte le funzionalità della classe genitore, quindi le proprietà e i metodi; `errorMessage` mostra i messaggi di errore dei blocchi `Exception` ed `Error`; se si forniscono

all'utente troppe informazioni sull'errore, questo costituisce una violazione sulla sicurezza, mentre vanno bene se arrivano solo all'amministratore del sito; `list` assegna le variabili come se fossero un array; `explode` divide una stringa per una stringa: restituisce un array di stringhe, ognuna delle quali è una sottostringa di stringa formata dividendola sui limiti formati dal separatore di stringhe; `get_metadato_app_properties` ottiene le proprietà metadato dalla variabile `$lab`.

- `main_interface.php`

Metodi presenti nel file: `file_exists` controlla se esiste un file o una directory; `require_once("e65metadato_container.php")` con questa espressione `require_once` si ha che è identica a `require` eccetto che PHP controllerà se il file è già stato incluso e, in tal caso, non lo includerà (richiederà) di nuovo; `if (isset($_POST['metadato_app']))` dove `if` determina se le proprietà siano tutte passate al programma attraverso il metodo POST, `isset` determina se una variabile è dichiarata ed è diversa da null, `$_POST` è utilizzato per passare le variabili; `isset($_POST['metadato_prerequisiti'])` è un array che precedentemente trasformato in una stringa con il metodo `implode`, però se non è settato, cioè non viene scelto alcun prerequisito, allora lo devo settare ad una stringa o mi restituirebbe un errore. I programmi sono dotati di funzioni che controllano le eccezioni mediante costrutti `try/catch` in modo che non si blocca se c'è qualcosa che non va, ma manda messaggi di errore. Inoltre nel blocco `try` si possono mettere anche ulteriori funzioni, ad esempio le condizioni `if`.

Inoltre crea ed utilizza l'oggetto `metadato_container` per contenere, creare e passare ogni altro oggetto richiesto (senza conoscere il nome di tale oggetto). Utilizza un file xml per individuare la posizione e il nome dei file contenenti le classi che verranno create (per garantire la Dependency Injection). L'applicazione utilizza il programma `main_interface` per accedere alle altre classi. Il programma individuerà quali classi servono per eseguire un determinato compito, e utilizza `metadato_container` per cercarle (tramite il file xml). Tutti gli oggetti `metadato_container` vengono poi distrutti quando si fa clic sul pulsante `submit`

del form. Si usano gli oggetti creati da `metadato_container`. Alla fine dell'utilizzo, il garbage collector elimina `$metadato_object` dalla memoria.

- `e65metadato_container.php`

Contiene due metodi : `get_metadato_application` che è utilizzato per leggere il nome e la posizione di uno dei file elencati nel file xml; `create_object` che crea una istanza della classe `metadato`; per la Dependency Injection non solo c'è il controllo sul nome e la posizione del file, ma occorre anche sapere il nome della classe. Contiene due variabili private cioè il valore è noto solo all'interno della classe; il costruttore accetta come valore (`$value`) il nome di un tipo di applicazione che si vuole trovare nel file xml; controlla se esiste la funzione `clean_input`: questo è un controllo perchè se si tentasse di creare un programma da mandare in esecuzione e che non abbia al suo interno la funzione `clean_input`, si lancerebbe una eccezione. Il ciclo `foreach` controlla che la posizione del file sia quella del nostro file altrimenti lancerebbe una eccezione.

- `e61metadato_applications.xml`

Un file xml per individuare la posizione e il nome dei file contenenti le classi che verranno create (per garantire la Dependency Injection). Utilizzando il file xml è possibile eseguire modifiche senza intervenire sul codice dell'applicazione. Si tiene traccia delle varie versioni dei file.

- `e65metadato.php`

Contiene la class `Metadato`, dove si inizializzano le variabili, mentre nel costruttore si assegnano i valori iniziali.

In questo file è rappresentato il modello di un metadato della MVC; questa progettazione MVC inoltre permette la Dependency Injection, cioè un programma(client) può usare una classe, in questo caso la classe `Metadato` di questo file, senza doverne conoscere

necessariamente l'implementazione. Questo permette la libertà di modificare un blocco di codice indipendentemente dagli altri blocchi di codice che restano invariati.

Sono presenti i metodi setter e getter; i metodi setter sono utilizzati per fare una verifica sui dati inseriti: se i dati non passano la verifica, questi possono essere non accettati e si lancia una eccezione. Viene utilizzato un operatore ternario per vedere se ci sono errori. La stringa dei messaggi d'errore viene costruita con un artificio, in quanto l'errore restituirebbe un intero (0 o 1) invece a noi serve una stringa e quindi utilizziamo un operatore ternario che restituisce appunto valori stringa. Si controlla ogni singolo aggiornamento, per sicurezza. Questa classe riceve i dati dal modulo e deve cercare di filtrarli ripulendo il codice errato. Questo in due modi: uno è che devono essere inserite tutte le informazioni. Il secondo controllo è che vengono filtrate tutte le istruzioni HTML, JavaScript e PHP che possono nascondere codice pericoloso, con la funzione `clean_input`.

L'applicazione metadato utilizza classi e funzioni contenute in diversi file, questo significa che ci sono diversi 'require'. Questa classe utilizza la classe `metadato_data` per memorizzare le informazioni sui metadati in un file XML.

- `e63metadato_data.php`

Nel costruttore c'è la funzione `libxml_use_internal_errors` che disabilita gli errori libxml e consente all'utente di recuperare le informazioni sull'errore secondo necessità; `simplexml_load_string` che interpreta una stringa di XML in un oggetto; `libxml_get_errors` che recupera array di errori; `json_encode` che restituisce la rappresentazione JSON di un array associativo; `json_decode` che decodifica una stringa JSON, significa che prende una stringa codificata JSON e la converte in una variabile PHP, in questo caso un array associativo; `__destruct()` che è un distruttore, viene chiamato quando l'oggetto viene distrutto o lo script viene interrotto o chiuso; se crei una funzione `__destruct()`, PHP chiamerà automaticamente questa funzione alla fine dello script; `preg_replace` che esegue una ricerca e sostituzione di espressioni regolari; `file_put_contents` scrive i dati in un file;

unset annulla l'impostazione di una data variabile; serialize genera una rappresentazione memorizzabile di un valore.

- emetadato_data.xml

E' un file xml creato dinamicamente dal sistema che contiene i metadato che si salvano nel file xml:

- 02252021105833change.log

Questo è un file di log creato dal programma, ed è in un formato particolare che fornisce tutte le informazioni necessarie per il processo di ripristino.

- 02252021105834emetadato_data.xml

E' un file di backup da utilizzare per il ripristino nel caso in cui l'inserimento dei dati da parte dell'utente provochi qualche danneggiamento.

- Errors.log

Il file di registro degli errori si crea automaticamente quando si installa il plugin nella posizione indicata nel file metadato_interface. Il path di questo file si setta all'inizio del file metadato_interface per trovarlo facilmente da parte dell'analista.

E' in una posizione al di fuori della cartella dove risiede il plugin e deve consentire all'applicazione un accesso in scrittura; contiene la data e l'ora del messaggio; solo il personale autorizzato ha accesso a questi log, mentre ad un utente generico arriva solo una frase generica sull'errore.

Capitolo 5

Controllo della consistenza di un corso

5.1 Introduzione al problema

Oltre al controllo della presenza all'interno di un corso di prerequisiti non definiti in alcuna sezione, è d'interesse verificare la presenza di cicli che si possono instaurare tra i Learning Node. Un controllo in tal proposito lo effettua successivamente l'algoritmo pianificatore, ma questo avviene solo lato studente: si vuole offrire questa possibilità anche lato docente, andando oltre. L'algoritmo pianificatore segnala solo la presenza di un ciclo, ma non lo evidenzia in maniera intuitiva e non mostra tutti i possibili cicli: l'idea, quindi, è di evidenziare sul grafo del Learning Path del corso tutti i possibili cicli presenti.

L'intervento si è strutturato in due parti: offrire un rapido mezzo per capire se un corso contiene dei percorsi ciclici, successivamente fornire uno strumento per elencare tali cicli e mostrarli in maniera visuale.

Si sono individuate le soluzioni nel comando `acyclic` di GraphViz e nell'implementazione in PHP dell'algoritmo di Hawick-James.

5.2 Comando `acyclic` di GraphViz

Invece di accoppiare le funzionalità del grafo con l'algoritmo pianificatore, operazione che avrebbe richiesto anche di ottenere ulteriori dati necessari per il suo corretto funzionamento (modello dello studente, obiettivo formativo del corso, rappresentazione in un oggetto `AlberoXML` del corso), si è voluto sfruttare il software esterno GraphViz, già utilizzato per generare il grafo del Learning Path.

GraphViz ha un tool interno che può esser usato per verificare la presenza di cicli all'interno di un grafo: *acyclic*. Il suo scopo primario è di restituire, dato come input un file *.dot* che rappresenta un grafo orientato con dei cicli, un nuovo grafo con alcuni archi invertiti, in modo da eliminare i cicli. Risulta più interessante ai fini del nostro intervento una sua funzione secondaria, che permette di controllare solo se il grafo contenga dei cicli o meno. *Acyclic* restituisce a schermo il risultato di tale verifica, dichiarando il grafo come aciclico o contenente dei cicli; direzionando il suo output dal terminale ad un file, si rende tale risultato fruibile dal sistema. Il tutto si ottiene facendo eseguire al server il comando:

acyclic -nv \$sourcefile 2> \$dest

dove *\$sourcefile* indica il file *.dot* dato in input, l'operatore *>* serve a direzionare l'output del comando che lo precede verso *\$dest*, che è un file temporaneo in cui poi andare a leggere tale risultato; il flag *-n* serve per non far generare il nuovo grafo, ma effettuare solo la verifica di ciclicità, mentre il flag *-v* fa stampare il risultato sul flusso di output *stderr* (standard error); l'operatore *2>* serve per redirigere l'output di *stderr* verso un file.

5.3 Algoritmo di Hawick – James

Il problema di trovare eventuali cicli in un grafo orientato è stato sempre oggetto di studio da anni, vista la loro importanza applicativa. Da questo ci si è anche chiesti di trovare non solo un eventuale ciclo in un grafo orientato, ma anche un elenco completo di questi cicli: sono stati presentati alcuni algoritmi che svolgono questo compito, come gli algoritmi di Tiernan [Tier70] o Tarjan [Tarj73], ma le loro implementazioni sono piuttosto dispendiose in termini di memoria di calcolo richiesta e tempo limitati nell'ordine $O(N \cdot e \cdot (c+1))$, dove *N* indica il numero di nodi, *e* il numero di archi e *c* il numero di cicli. Un algoritmo migliore in prestazioni è Johnson [John75], che riesce ad elencare tutti i circuiti elementari di un grafo diretto in tempo limitato da $O((n+e) \cdot (c+1))$ e spazio limitato

$O(N + e)$, questo anche grazie a dei controlli che eliminano la riproposizione di circuiti già considerati in precedenza sotto forma di una permutazione dei loro nodi.

Si è considerato per il nostro intervento l'algoritmo di Hawick e James [HJ08], un'evoluzione dell'algoritmo di Johnson, ma che considera anche grafi con componenti isolate e dà alcune informazioni strutturali come la lunghezza media dei cicli, i nodi che hanno più partecipazione ai cicli e il ciclo con lunghezza maggiore.

L'algoritmo descrive il grafo tramite una lista di adiacenza contenute in un array bidimensionale $A_k[i][j]$, dove l' i -esimo vertice ha j archi in uscita (con $j \in [0, \dots, m]$, m è il numero totale di archi presenti nel grafo), indicati tramite un array in cui il valore è il nodo di arrivo. Prendendo in analisi tutti i nodi del grafo, l'algoritmo inserisce in uno stack i nodi dai quali ha tentato una visita ricorsiva e li blocca, per evitare che vengano ripresi in considerazione; la funzione principale `circuit($v)` viene richiamata ricorsivamente per cercare un ciclo: se il nodo v è diverso da quello con cui è iniziata l'iterazione dell'algoritmo e non è stato preso in considerazione finora, viene rilanciata ricorsivamente `circuit` su di esso; se il nodo che ha ricevuto in input è uguale al nodo iniziale, è stato trovato un ciclo, salvo l'attuale stack, imposto un flag per indicare la fine della ricerca e richiamo la funzione `unblock($u)` per sbloccare i vari nodi. L'algoritmo elenca i circuiti elementari presenti nel grafo, secondo l'ordine con cui sono stati passati i nodi.

L'implementazione in PHP è contenuta nel file `hawickJames.php`, inserito nella directory `/course/format/adaptive`, ha le funzioni richieste dall'algoritmo ed alcune funzioni di supporto, per gestire al meglio le strutture dati utilizzate. Ecco l'elenco nel dettaglio:

- `countArcs()`: ritorna il numero di archi presenti nel grafo;
- `removeFromList($list, $arc)`: elimina l'elemento dall'array e ne ricompatta i valori;
- `stackInit()`: inizializza lo stack ed azzerla la variabile `$stackTop`, che punta alla testa dello stesso;
- `stackPush($val)`: immette il valore `$val` in testa allo stack;
- `stackPop()`: ritorna l'elemento in testa allo stack e decrementa `$stackTop`;
- `stackClear()`: svuota lo stack;

- `unblock($u)`: funzione che imposta a false il nodo nell'array `blocked`, che indica se un nodo è bloccato o meno; successivamente, per ogni nodo `$w` figlio di `$u`, lo rimuove dall'array `$B[$u]`, che rappresenta il grafo degli elementi bloccati, e sblocca ricorsivamente `$w`;
- `circuit($v)`: la funzione principale dell'algoritmo, immette i nodi nello stack, blocca i nodi considerati lungo il cammino e verifica se è stato trovato un ciclo, come descritto precedentemente;
- `setupGlobals($nVert, $ak)`: si occupa di inizializzare le variabili globali necessarie ai fini dell'algoritmo e imposta il grafo sulla base di `$ak`;
- `launchHawickJames($a, $nv)`: è la funzione richiamata dall'esterno per lanciare l'algoritmo di Hawick-James, si occupa di lanciare la funzione `circuit($v)` su ogni nodo del grafo e di riportare le variabili di supporto al loro stato originale dopo ogni iterazione; ritorna l'array bidimensionale contenente tutti i cicli trovati.

5.4 Integrazione nel sistema Moodle_LS

Per integrare l'algoritmo nel sistema Moodle_LS, si sono apportate due modifiche principali:

- l'aggiunta di una risorsa nella sezione iniziale del corso, che permetta di visitare la pagina dedicata per il controllo sulla consistenza del corso, come indicato nel capitolo 3;
- la creazione di alcune funzioni di supporto nel file `grafo.php`.

Concentriamo l'attenzione sul secondo punto: sono state create delle funzioni per generare un input adatto all'algoritmo e per tradurre il suo risultato in un file `.dot`; inoltre è stata creata una funzione che gestisce il processo e stampa il risultato del controllo nella pagina web `check_consistenza.php`. Ecco nel dettaglio le funzioni:

- *node2index(\$corso)*: si occupa di recuperare l'elenco dei nodi del grafo dal file .dot del corso ricevuto in input;
- *graph2array(\$nodi,\$corso)*: restituisce un array bidimensionale $\$graph2array[\$i][\$j]$, dove $\$i$ è l'indice del nodo di partenza dell'arco, nell'ordine creato da *node2index*, mentre $\$j$ indica il numero di archi tramite un array in cui il valore è l'indice del nodo di arrivo; per esempio se il nodo con indice 2 ha due archi in uscita verso il nodo 4 ed il nodo 6, lo si indica con $\$graph2array[2] = \{0 \rightarrow 4; 1 \rightarrow 6\}$;
- *array2graph(\$mappa, \$matrice, \$corso)*: si occupa di generare il file .dot dedicato al controllo sui cicli, prendendo in input l'elenco dei nodi del corso $\$mappa$, la matrice con tutti i cicli del corso e l'id del corso stesso; sulla base del file .dot del corso, ne crea una copia e va a modificare gli archi che formano gli eventuali cicli, colorandoli di rosso; $\$mappa$ serve per far corrispondere l'id del nodo con il suo nome, necessari per la ricerca della riga da modificare nel file .dot;
- *checkConsistenza(\$corsoId,\$nomeCorso)*: richiamata dall'utente docente visualizzando la pagina *check_consistenza.php*, verifica prima che il grafo del corso sia aciclico o meno, lanciando il comando *acyclic* di GraphViz e leggendone il risultato da un file temporaneo; in caso di presenza di cicli, avvia il processo per ottenere un input valido per l'algoritmo di Hawick-James e lo richiama tramite la funzione *launchHawickJames*; il risultato viene poi gestito tramite la funzione *array2graph* e viene mostrato nella pagina il grafo contenente i cicli; se invece il grafo risulta aciclico, viene semplicemente richiamata la funzione *visualizzaGrafo*, che stampa il grafo del corso.

Ecco un esempio di come viene visualizzato il grafo del Learning Path del corso in presenza di cicli.

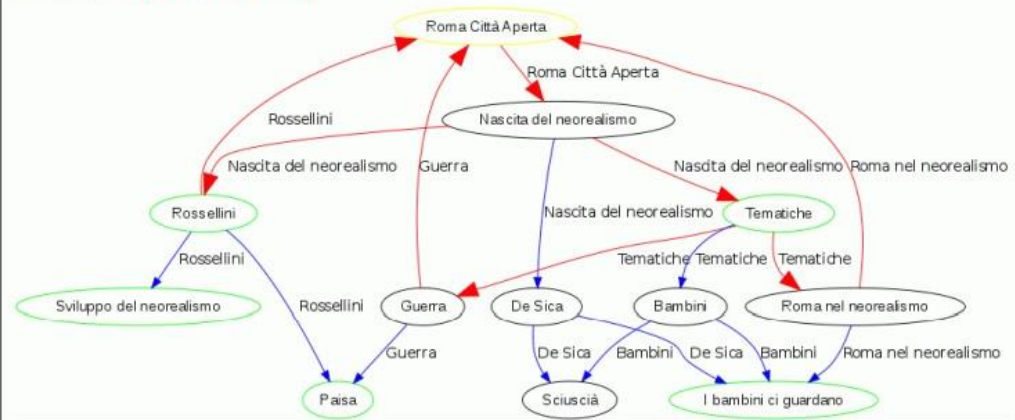
Cinema Neorealista Italiano -> Consistency check

Go back

Close window

Consistency check for the course Cinema Neorealista Italiano; any cycle is marked in red.

Consistency check failed! Cycle(s) detected!



Il sistema evidenzia in rosso gli archi che formano i cicli.

Capitolo 6

Learning Path interattivo

6.1 Proposta di modifica

Finora la visualizzazione del Learning Path consisteva nel mostrare in una pagina un'immagine statica, che mostra il dominio della conoscenza di un corso se l'utente è un docente, o il proprio percorso formativo in caso di uno studente.

Si vuole fornire agli utenti, invece, un sistema che possa sfruttare il grafo in questione ma che possa visualizzare anche delle informazioni più dettagliate per un nodo specifico, come ad esempio tutte le risorse didattiche a disposizione, mantenendo comunque la più completa compatibilità con i browser più diffusi.

Inizialmente si era pensato di sfruttare il formato SVG, che è tra i formati disponibili in output da GraphViz: esso permette di associare degli eventi dinamici all'immagine, come poteva essere il click o il passaggio del mouse su un nodo. Purtroppo, non tutti i browser supportano questo formato (specialmente le versioni più vecchie di Microsoft Internet Explorer), quindi si è scartata l'opzione.

6.2 Funzione Cmap di GraphViz

Lo stesso GraphViz ha una funzione aggiuntiva per il comando dot che è stata scelta come soluzione per questo intervento: Cmap.

Tramite Cmap, GraphViz genera un file .map, dove salva le associazioni tra una particolare area dell'immagine ed un link, che trova associato al nodo nel file .dot. Per aggiungere

questo campo, si è dovuto intervenire nelle funzioni che gestiscono il file .dot, affinché fosse generato nella descrizione del nodo anche il link alla risorsa. Successivamente, si è aggiunta in locallib.php di Adaptive la funzione *risorseNodo(\$corsoId, \$nodoNome)*, che recupera dal database le risorse richieste. Nel dettaglio, ecco l'elenco delle modifiche apportate, tutte contenute nella directory /course/format/adaptive:

- in grafo.php la funzione *inserisciCoppiaNelGrafo(\$ak,\$rk,\$corso,\$nomeCorso)* è stata modificata, aggiungendo nella descrizione del nodo da scrivere il campo `URL="".$CFG->wwwroot.'/course/format/adaptive/dettagli_nodo.php?courseid='.`
`$corso.'&node='.$ak.'`, prendendo i dati necessari da quelli ottenuti in input alla funzione;
- in grafo.php la funzione *cancellaNodo(\$ak,\$corso)* è stata modificata in maniera simile al punto precedente;
- in locallib.php, è stata aggiunta la funzione *risorseNodo(\$corsoId, \$nodoNome)*, che interroga il database prima per ottenere la sezione in cui risiede il Learning Node, poi per ottenere la sequenza di risorse collegate e si scelgono solo quelle che sono etichettate come risorse (quindi si scartano eventuali quiz); successivamente si recuperano i nomi di tali risorse e si generano i link; la funzione ritorna un array in cui nella posizione 0 si trova l'array dei link alle risorse, mentre in posizione 1 vi sono i loro nomi;
- è stata creata la pagina web `dettagli_nodo.php`, che mostra i dettagli di un nodo; la pagina effettua anche un controllo sul ruolo dell'utente, per richiamare la funzione *visualizzaGrafo* col parametro corretto (0 se è un docente, l'id dell'utente se è uno studente);
- in grafo.php la funzione *visualizzaGrafo(\$corsoId, \$nomeCorso, \$student)* è stata modificata, utilizzando un nuovo comando per la chiamata a GraphViz: `dot -Tcmap -o $map -Tgif -o $destfile $sourcefile`, dove il flag `-Tcmap` indica a GraphViz di generare anche una mappatura dell'immagine e di salvare l'output (primo flag `-o`) nel file precedentemente creato `$map`; è stata modificata anche la stampa delle linee di

codice per la visualizzazione dell'immagine, con l'aggiunta del tag HTML<map>, al cui interno viene stampato il contenuto del file \$map.

Ecco un esempio di un dettaglio di un nodo, che ha alcune risorse collegate.

Cinema Neorealista Italiano -> Info learning node: Tematiche

[Go back](#)

[Close window](#)

These are the learning materials for node Tematiche :

[Tematiche](#) [Tematiche](#)

To view the learning materials about a topic, click the relative node.

Graph colors:
No didactic material available
1 didactic material
2 didactic materials
3 didactic materials
4 didactic materials

```
graph TD; A([Nascita del neorealismo]) -- "Nascita del neorealismo" --> B([Tematiche]); B --- C[Tematiche]; B --- D[Tematiche];
```

In alto vengono mostrati i link alle risorse collegate al nodo scelto.

Capitolo 7

Conclusioni e sviluppi futuri

7.1 Conclusioni dopo l'intervento e sviluppi futuri

Da come si è potuto evincere dagli esempi forniti, l'intervento ha centrato tutti gli obiettivi prefissati, migliorando l'esperienza d'uso del sistema e fornendo nuove funzioni a servizio degli utenti. La scelta effettuata in passato di utilizzare il tool GraphViz ha portato all'utilizzo di sue ulteriori funzioni, come l'introduzione di un primo livello di interattività col grafo, un più facile controllo riguardante la presenza di cicli nella struttura di un corso e la loro segnalazione. A questo si aggiunge l'implementazione dell'algoritmo di Hawick – James che ha reso possibile la ricerca di tutti i cicli presenti, in maniera efficiente.

Le modifiche effettuate nel codice di Moodle hanno portato all'eliminazione di fastidiose incongruenze, come la riproposizione di un quiz che era da considerarsi come superato. A tal proposito, uno dei prossimi più importanti interventi da realizzare è la migrazione della piattaforma ad una più recente versione di Moodle: finora si è utilizzata la 1.9.8, il cui supporto tecnico da parte dello staff di Moodle è stato oramai interrotto. Con le versioni più recenti (la 2.4.1 al momento è l'ultima uscita) si godrebbe di molti nuovi servizi introdotti e diversi bug di sistema corretti, ma ad un prezzo: con la versione 2.0 vi è stato un forte cambiamento nel codice di Moodle, nel suo database e nel modo in cui interfacciarsi con esso. Nella documentazione ufficiale è esplicitamente scritto che qualsiasi plugin o tema personalizzato che si basi su versioni di Moodle precedenti alla 2 non potranno funzionare: ciò comporta un profondo lavoro di aggiornamento di 64 Moodle_LS che riguarda ogni sua singola funzione, affinché possa essere usata in Moodle 2.0 o in successive versioni.

Riguardo l'attuale istanza di Moodle_LS, si è notato nel corso del lavoro di tirocinio come sia necessario, per un corretto funzionamento del processo di personalizzazione del percorso didattico, che il docente crei il corso seguendo sempre un determinato ordine e non apporti modifiche che riguardino i quiz, altrimenti si possono presentare comportamenti anomali: è bene correggere tale problema, visto che mina la stabilità del sistema.

Per concludere, si è notata la mancanza di strumenti di testing a servizio dell'amministratore: si potrebbe creare uno studente di prova, utile nell'eseguire dei test all'interno dei corsi adattivi, che possa essere riportato al suo stato iniziale di studente appena iscritto al sistema, visto che attualmente si è obbligati a cancellare manualmente l'utente e ripetere una nuova iscrizione ogni volta.

Bibliografia

- [Bl56] B.S. Bloom, "Taxonomy of educational objectives: the classification of educational goals; Handbook 1: Cognitive Domain" New York, Longmans, Green, 1956.
- [BM07] P. Brusilovsky, E. Millan, "User Models for Adaptive Hypermedia and Adaptive Educational Systems", "The Adaptive Web: Methods and Strategies of Web Personalization", P. Brusilovsky, A. Kobsa, W. Nejdl, eds., Springer-Verlag, 2007.
- [Br11] P. Brusilovsky, "Adaptive Hypermedia", User Modeling and User-Adapted Interaction, vol. 11, pp. 87-110, 2001.
- [BV03] P. Brusilovsky, J. Vassileva, "Course Sequencing Techniques for Large-Scale Web-Based Education", Intl. J. Continuing Eng. Education and Life-Long Learning, vol. 13, pp. 75-94, 2003.
- [Ch01] D.N. Chin, "Empirical Evaluation of User Models and User-Adapted Systems", User Modeling and User-Adapted Interaction, vol. 11, n. 1, pp. 181-194, Mar 2001.
- [CMLOP07] M. Cialdea Mayer, C. Limongelli, A. Orlandini, V. Poggioni, "Linear Temporal Logics and Executable Semantics for Planning Languages", J. Logic, Language, and Information, vol. 1, n. 16, pp. 63-89, Gen. 2007.
- [DbSS06] P. De Bra, D. Smits, N. Stash, "Creating and Delivering Adaptive Courses with AHA!" Proc. European Conf. Technology Enhanced Learning (EC-TEL '06), pp. 21-33, 2006.
- [FS88] R.M. Felder, L.K. Silverman, "Learning and Teaching Styles in Engineering Education" Eng. Education, vol. 78, n. 7, pp. 674-681, 1988.

- [FSol] R.M. Felder, B.A. Soloman, “Index of Learning Styles”, informazioni sulla scala usata e sulle teorie sono disponibili al sito <http://www4.ncsu.edu/unity/lockers/users/f/felder/public/ILSpage.html>
- [FSp04] R.M. Felder, J.E. Spurlin, “Applications, Reliability, and Validity of the Index of Learning Styles. “Intl. Journal of Engineering Education, 21, pp. 103-112, 2005.
- [Ge05] C. Gena, “Methods and Techniques for the Evaluation of User-Adaptive Systems”, Knowledge Eng. Rev., vol. 20, n. 1, pp. 1-37, 2005.
- [GrVi] Documentazione e caratteristiche disponibili al sito web <http://www.graphviz.org/>
- [HJ08] K.A.Hawick, H.A.James, "Enumerating Circuits and Loops in Graphs with Self-Arcs and Multiple-Arcs", Proc. 2008, International Conference on Foundations of Computer Science (FCS'08), pp. 14- 20, Las Vegas, USA, 14-17 Luglio 2008.
- [John75] D.B. Johnson, “Finding all the elementary circuits of a directed graph”, SIAM Journal on Computing 4, pp. 77–84, 1975
- [LSV08] C. Limongelli, F. Sciarrone, G. Vaste, “LS-Plan: An Effective Combination of Dynamic Courseware Generation and Learning Styles in Web-Based Education”, Adaptive Hypermedia and Adaptive Web-based Systems – 5th International Conference, AH 2008, Hannover, Germana, 29 Luglio – 1 Agosto, Proceedings, pp. 133-142, 2008.
- [Kolb84] D.A. Kolb, “Experiential Learning: Experience As the Source of Learning and Development”, Prentice-Hall, 1984.
- [MB8085] I. Briggs Myers, P. B. Myers, “Gifts Differing: Understanding Personality Type”. Mountain View, CA, Davies-Black Publishing, 1980, 1995.

- [Tier70] J. C. Tiernan, “An efficient search algorithm to find the elementary circuits of a graph”, *Communications of the ACM* 13, pp. 722–726, 1970.
- [Tarj73] R. Tarjan, “Enumeration of the elementary circuits of a directed graph”, *SIAM Journal on Computing* 2, pp. 211–216, 1973.
- [WB01] G. Weber, P. Brusilovsky, “Elm-Art: An Adaptive Versatile System for Web-Based Instruction” *Intl J. Artificial Intelligence in Education*, vol. 12, pp. 351-384, 2001.