tesi di laurea

Impiego della concept analisys nella classificazione di pagine web ai fini del reverse engineering

Anno Accademico 2005/2006

relatore
Ch.ma prof. Anna Rita Fasolino

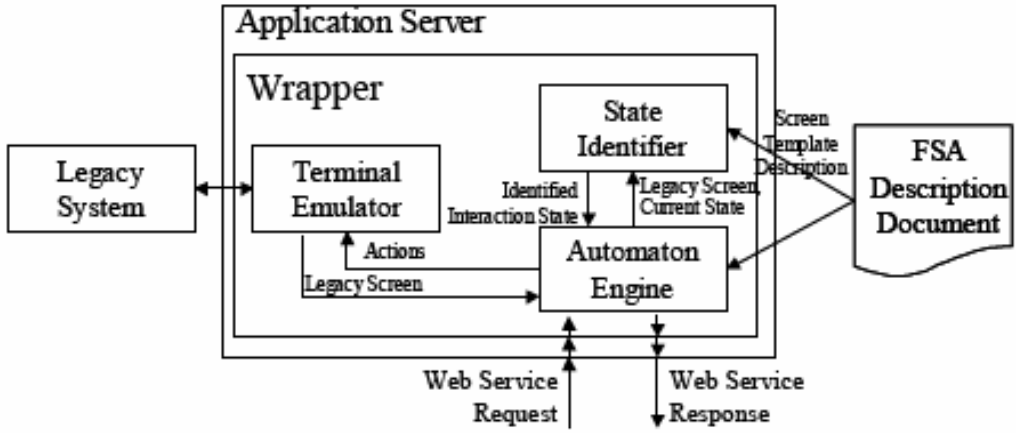
correlatore
Ch.mo prof. Porfirio Tramontana

candidato
Angelo Di Maria
Matr. 534/761

Il contesto:

“Migrazione da Web Application a Web Service”

- Esportare i servizi di una Web Application per poterli accedere come un Web Service ➔ si passa dal paradigma interattivo al paradigma richiesta/risposta.
- Le tecniche di ammodernamento esistenti sono di tipo **black box reverse engineering basata sul Wrapping**.
- La metodologia del Wrapping si basa sull’utilizzo di un ***Wrapper*** rappresentante uno strato software che incapsula il sistema originale; il Wrapper fornisce una nuova interfaccia di interazione nascondendo l’originale.
- Il Wrapper rappresenta l’interprete dell’automa (FSA); l’automa rappresenta il modello di interazione tra l’utente e la Web Application. Ad ogni stato dell’automa è associato un set di azioni che il Wrapper dovrà eseguire.



Il problema:

- Da una stessa Server Page possono essere generate dinamicamente un insieme molto ampio di ***B***uilt ***C***lient ***P***age (BCP o screen).
- Il Wrapper deve saper riconoscere dinamicamente, dallo screen ritornato dalla Web Application, lo stato di interazione (o Interaction State) correntemente in esecuzione in modo da eseguire le azioni associate allo stato.
- Per riuscire ad identificare un Interaction State bisogna prima di tutto avere una descrizione degli screen ritornati dalla Web Application.

La soluzione proposta:

- Classificare le BCP, ritornate dalla Web Application, in classi di equivalenza (o Equivalent Built ***C***lient ***P***age).
- Una classe di equivalenza racchiude un set di BCP aventi caratteristiche comuni e per le quali si vuole che il Wrapper si comporti nello stesso modo.

Contributo apportato

- Implementazione di un Framework di *supporto* alla ricerca di un set minimale di features in grado di discriminare le BCP, ritornate dalla Web Application, tra differenti EquivalentBCP (o classi di equivalenza) afferenti ad una stessa Server Page.

1. Utilizzo di *PageClassfier*, tool ereditato scritto in VisualBasic, per l'archiviazione di classificazioni di pagine web.

2. Conversione di pagine web in formato HTML nel formato XHTML (XML-compatibile); ausilio della libreria JTidy.

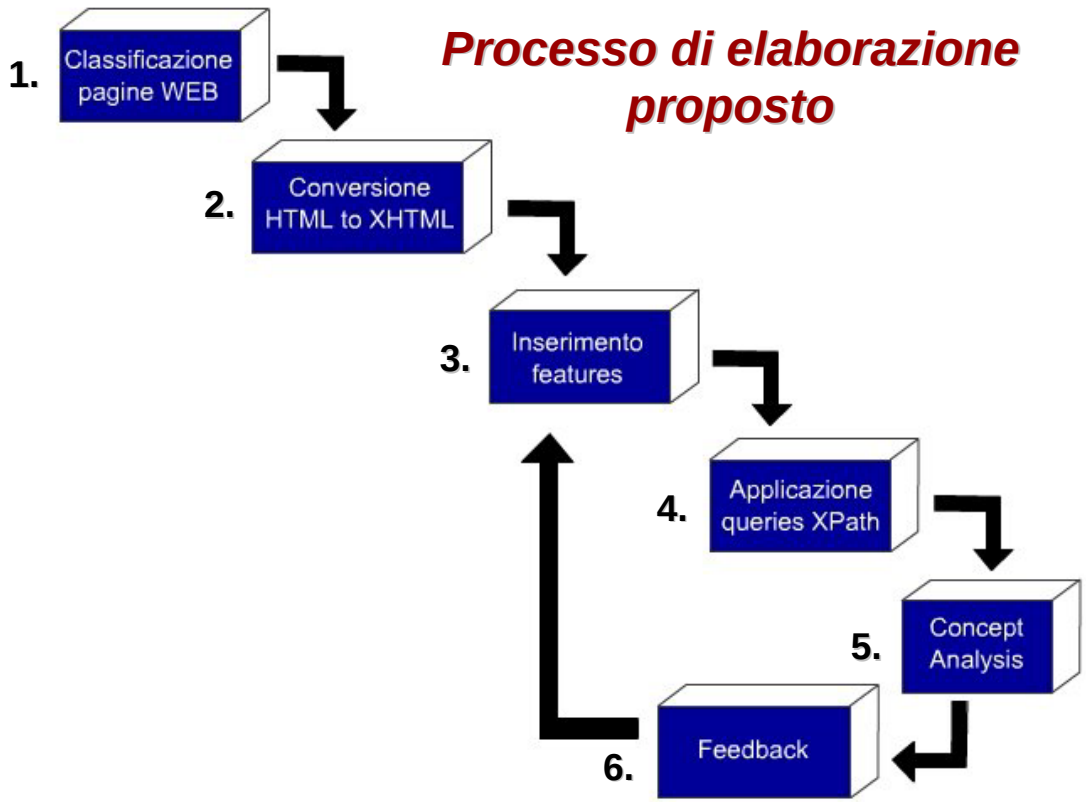
3. Inserimento di features (generate da un tool esterno o proposte da un esperto) caratterizzante una BCP.

4. Calcolo della presenza delle features modellate sotto forma di queries XPath.

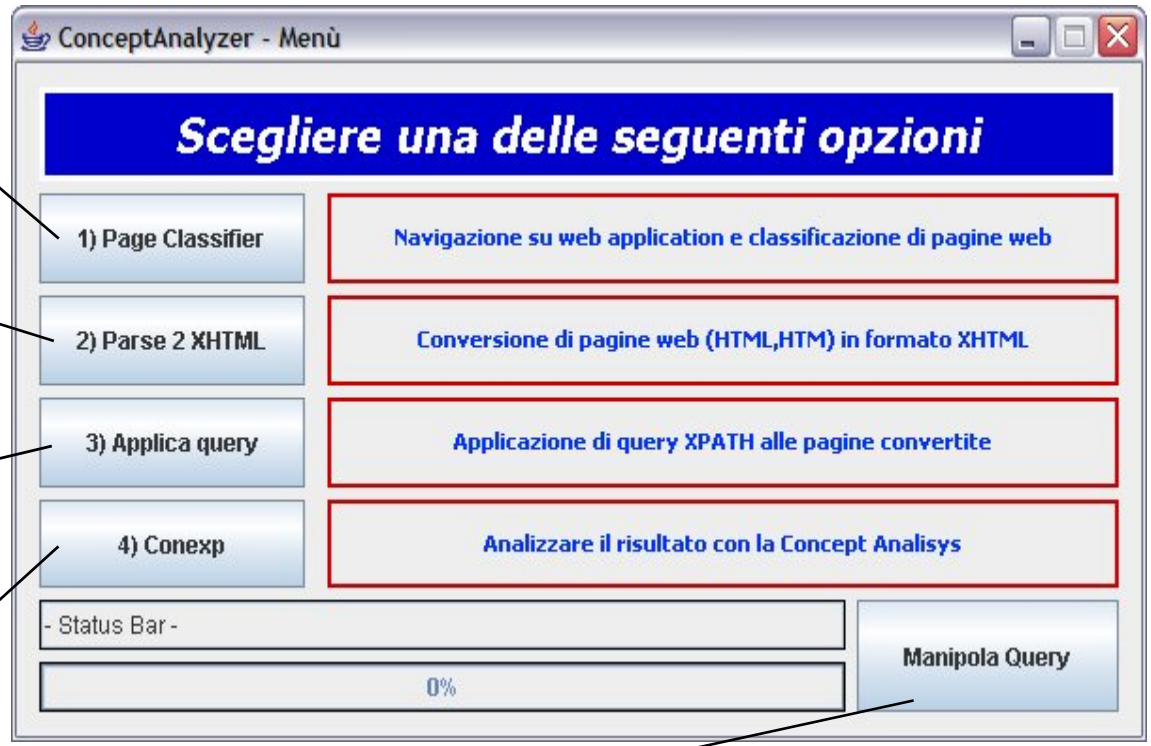
5. Analisi dei risultati ottenuti con *Conexp*; tool ereditato scritto in Java che sfrutta i concetti della Concept Analysis.

6. Qualora i risultati forniscono valori degl'indici di prestazione (Recall, Precision) non soddisfacenti si riparte dal punto 3.

Processo di elaborazione proposto



Il Tool: *"ConceptAnalyzer" (1/2)*



- Esegue PageClassifier per la navigazione della Web Application. Input richiesti: directory di salvataggio delle BCPs classificate; directory dove è contenuto il database; indirizzo web della Web Application.
- Esegue la conversione delle BCPs dal formato HTML nel formato XHTML. Input richiesti: files da convertire; directory di salvataggio.

```
tidy.setXHTML(true);
tidy.parse(is,os);
```

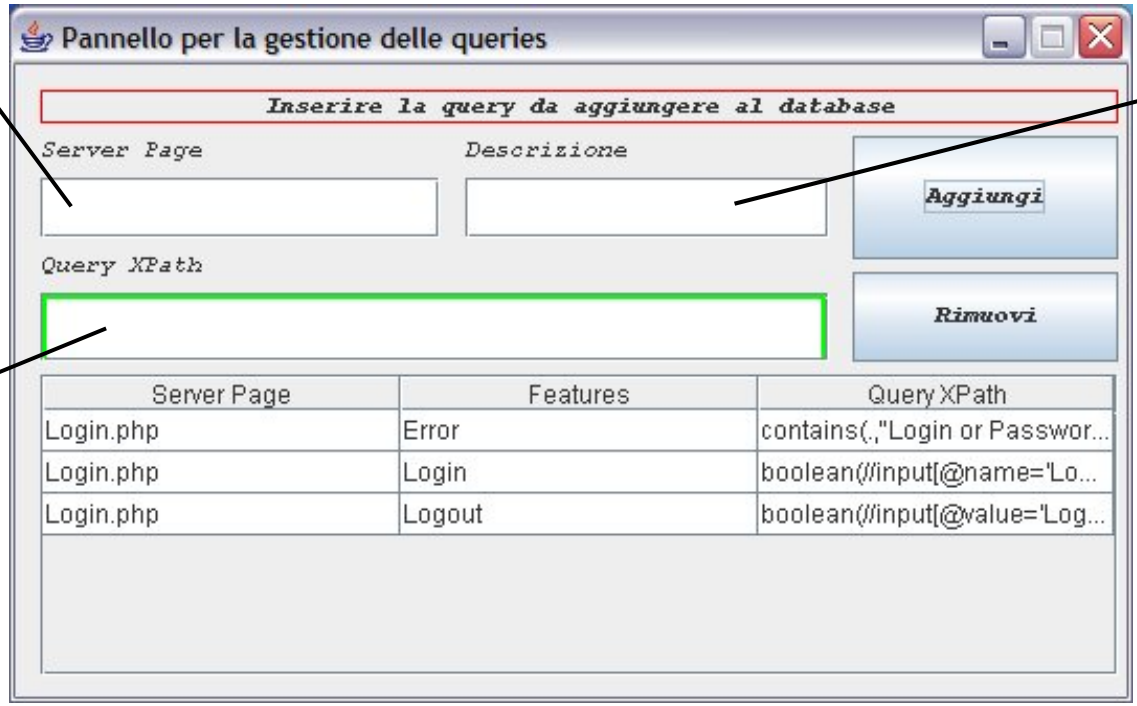
- Preleva le features inserite nel database ed esegue le queries memorizzando i risultati in un file testuale compatibile con il formato letto da Conexp. Input richiesti: files convertiti in formato XHTML; nome della server page.
- Esegue Conexp che permette di analizzare i risultati ottenuti con la teoria della Concept Analysis. Input richiesti: file.cxt.
- Lancia un form grafico per la gestione delle features presenti nel database.

Il Tool: *"ConceptAnalyzer" (2/2)*

- ConceptAnalyzer fornisce una finestra grafica per l'inserimento e l'eliminazione di una feature all'interno del database di supporto al processo.
- Accesso immediato alle queries inserite in precedenti sessioni di lavoro.
- Richiesta obbligatoria di tre campi caratterizzanti una feature: Server Page, Descrizione, Query XPath

Server Page: campo obbligatorio; informazione necessaria al fine di poter selezionare, durante il processo, soltanto le features associate alla serve page selezionata.

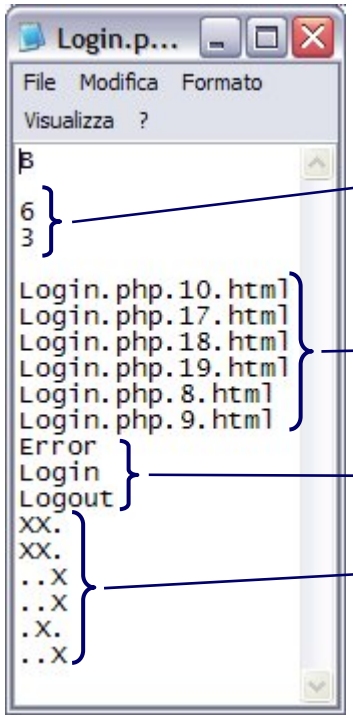
Query XPath: campo obbligatorio; rappresenta la feature sotto forma di espressione XPath.



Descrizione: campo obbligatorio; rappresenta una breve descrizione della query XPath. Fondamentale per le creazione dell'output del processo.

Output prodotto

- Il punto chiave del processo è l'applicazione delle features sotto forma di queries XPath
- Il risultato di tale fase è rappresentato da un file testuale con estensione ***".cxt"***
- Il file potrà essere letto dal tool ***Conexp*** per l'analisi delle features adottate



- Queste due righe in sequenza indicano rispettivamente il numero di BCPs navigate e il numero di features candidate; queste due informazioni indicano la dimensione del ***contesto***.
- Riportano il nome delle BCPs navigate.
- Riportano una breve descrizione delle features candidate.
- Rappresentazione matriciale del contesto di dimensione pari a ***#BCPs x #features***; ogni elemento *(i,j)* rappresenta una relazione binaria tra l'oggetto *i* e la feature *j*; la coppia *(i,j)* è rappresentata da una "**X**" se l'attributo *j* è presente nell'oggetto *i* altrimenti da un punto "**.**" se l'attributo j non è presente nell'oggetto i.

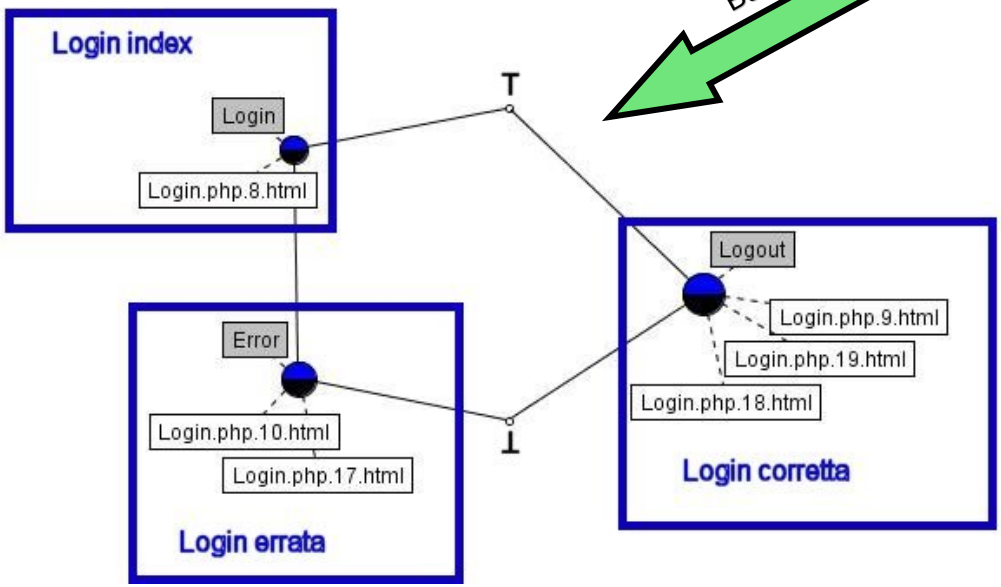
Interpretazione dell'Output prodotto...

- Il file.cxt prodotto, dato in ingresso a Conexp, restituisce una rappresentazione tabellare delle informazioni prodotte in fase di elaborazione ➔ ***CONTESTO***

Build Context

A	B	C	D
	Error	Login	Logout
Login.php.10.html	X	X	
Login.php.17.html	X	X	
Login.php.18.html			X
Login.php.19.html			X
Login.php.8.html		X	
Login.php.9.html			X

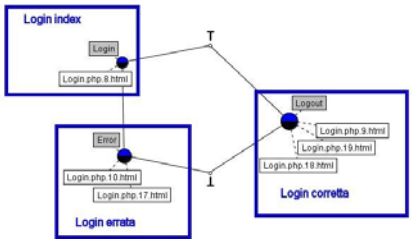
Build Lattice



- Dal contesto in forma tabellare si può passare ad un'analisi in veste grafica ➔ ***LATTICE***
- Tale reticolo è composto da un insieme di di concetti (o *concept*), ciascuno dei quali copre un sottoinsieme di **oggetti** (*estensione* del concetto) ed è descritto da un sottoinsieme di **attributi** (*intensione* del concetto).
- In un reticolo il nodo top ovvero quello più in alto rappresenta il concetto più generale ovvero con estensione massima e intensione minima. Viceversa il nodo down che rappresenta un concetto con estensione minima e intensione massima.
- Ogni concetto è rappresentato da un nodo; esso è diviso in due semicerchi, quello superiore con colorazione blue se al concetto è associato almeno un attributo, bianco altrimenti; un semicerchio inferiore con colorazione black se al concetto è associato almeno un oggetto, bianco altrimenti.

… Discriminant Expression

- Dal Lattice prodotto è possibile calcolare una ***D***iscriminant ***E***xpression (o predicato booleano) per ogni classe di equivalenza; tale operazione viene eseguita manualmente a valle della classificazione delle features per ogni EquivalentBCP.



Classificazione features → Specific, Relevant, Conditionally Specific → **Discriminant Expression minima** → Calcolo ***Recall*** e ***Precision*** su un ***Training Set*** → Se non è soddisfacente → Proporre nuove features

Se soddisfacente → Calcolo ***Recall*** e ***Precision*** su un ***Test Set*** → Se non è soddisfacente → Proporre nuove features

Se soddisfacente → ***STOP***

- Le operazioni sono da intendersi per ogni EBCP⊆Lattice
- Proporre nuove features: in questo caso bisognerà ripartire da punto 3 del processo di elaborazione (inserimento di features…) e proseguire con lo studio di un nuovo Lattice.
- Recall e Precision sono indici di prestazione; utili per studiare il comportamento di una Discriminant Expression al variare dell'insieme di BCPs considerato.
- Obbiettivo: trovare una Discriminant Expression che restituisca una valore di Recall e Precision massimo.

Conclusioni e sviluppi futuri

- E' stato realizzato un tool che permette di effettuare studi sperimentali sulla ricerca di criteri generali per la ricerca di espressioni minime che siano in grado di identificare le Equivalent Build Client Page (o classi di equivalenza) a cui appartengono le Build Client Page (o Screen Template) ritornate dalla Web Application.

- I limiti a tale approccio sono rappresentati da: classificazione manuale delle features per ogni EBCP (in base a definizioni teoriche); descrizione manuale delle BCPs; eventuale complessità del Lattice da analizzare; minimizzazione manuale delle Discriminant Expression per ogni EBCP.

- Uno sviluppo futuro, dunque, può essere rappresentato dalla creazione di algoritmi in grado di classificare automaticamente le features per ogni EBCP e generare una Discriminant Expression minima per ogni classe di equivalenza.