UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

tesi di laurea

Progetto e sviluppo di un prototipo per la migrazione di applicazioni Web di tipo Legacy

Anno Accademico 2005/2006

relatore
Ch.mo prof.ssa Valentina Casola

relatore
Ch.mo prof. Porfirio Tramontana

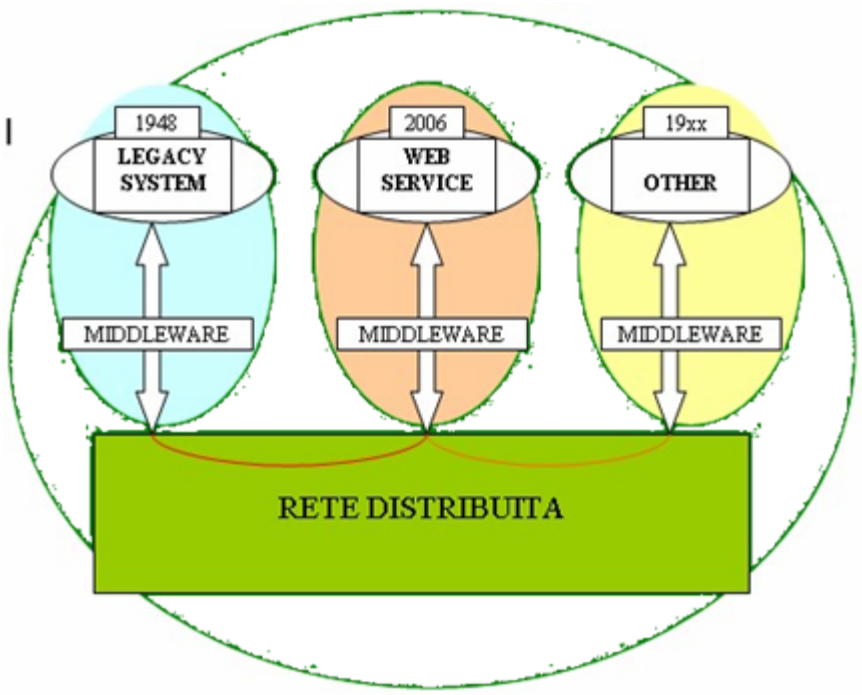
candidato
Antonino Esposito
Matr. 831/160

Panoramica

Migrazione da Legacy Application a Web Service

Obiettivi:

- Esportare i servizi di un Legacy System verso un paradigma WS al fine di integrare.
- Interoperabilità, eterogeneità, modularità e integrabilità
- Cooperazione applicativa
- Trasformare il paradigma interattivo in quello request/response tipico dei Web Services (automatizzazione).



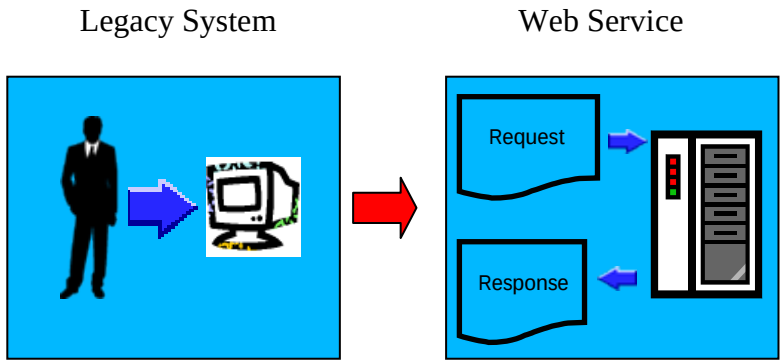
Soluzione Adottata:

- tecnica *black box* di reverse engineering basata su *wrapping*. Ovvero incapsulando il sistema originale con uno strato software che mostra una nuova interfaccia, nascondendo quella originale, ed interagisce con il Legacy System.

Contesto e Motivazioni

Contesto:

- Obsolescenza dei sistemi legacy rispetto alle nuove tecnologie e necessità di integrazione con esse.
- Integrazione sistemi legacy e web service



Motivazioni:

Vantaggi LS:

- presenti nelle attività produttive fondamentali;
- Costi di riprogettazione elevati e documentazione inadeguata;
- Funzionamento soddisfacente;

- **Vantaggi WS:**
- Indipendenti da applicazioni e piattaforme;
- Protocolli e dati standard.
- Ricerca automatica di servizi
- Esecuzione automatica
- Composizione automatica di servizi
- *interoperabilità* e *cooperazione applicativa*;

Architettura per l'interoperabilità: SOA

- logica Service-oriented orientata al riutilizzo e all'integrazione;
- consente di riutilizzare le funzionalità legacy tramite l'uso di WS e standard per protocolli e dati (es. XML);
- WS rendono standard il processo di *wrapping*

Il wrapper

Ruolo: software intermediario che rende un caso d'uso LA accessibile come un WS.

Obiettivo: guidare il legacy system durante l'esecuzione di ogni possibile scenario d'interazione del caso d'uso da migrare con il flusso di dati e comandi necessario.

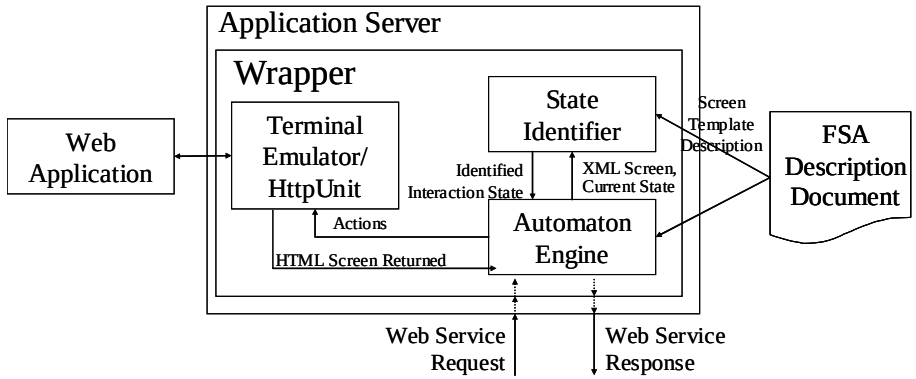
Terminal Emulator:

- Gestisce la comunicazione con la WA;
- Esegue azioni sulla WA (es.settaggio parametri di input e comandi di submit);

State Identifier:

- Identificazione Interaction States;
- Utilizzo di Screen Template per riconoscere lo stato raggiunto.

- **FSA Description**: contiene la descrizione e le variabili dell'automa.



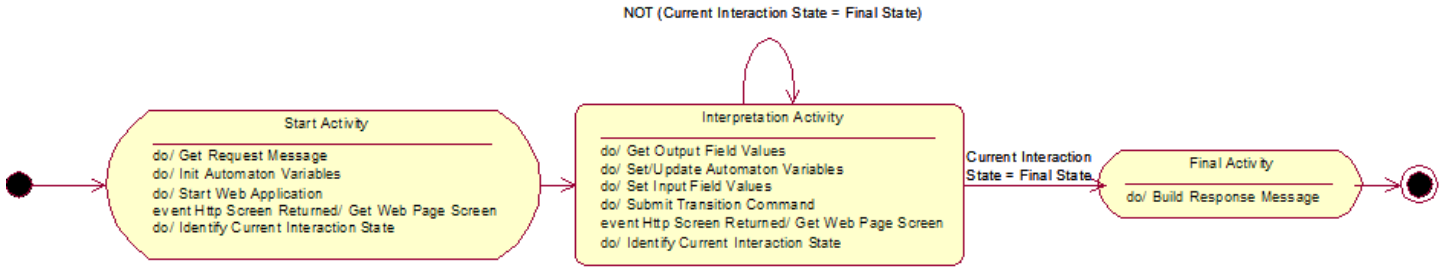
Automaton Engine:

- E' il motore dell'Automa (nucleo);
- interpreta l'Automa descritto nell'FSA repository;
- Interagisce con il Terminal Emulator: invia richieste HTTP e riceve pagine HTML;
- Interagisce con lo Screen Identifier: invia pagine HTML e riceve indicazioni su IS;
- Mantiene lo stato dell'automa facendo uso di apposite variabili.

Automaton Description e Automaton Engine

Automaton Description:

- Legge il file XML (SAX)
- memorizza le informazioni lette nelle classi *bean* dell'automa;
- nuovo container Variable per le variabili d'Automa per la compatibilità con una generica WA;
- implementazione funzionalità di popolamento del container Variable.



Automaton Engine:

- metodo *Get Output Field Value* per gli outpu delle schermate (Rhino);
- metodo *Set Update Automaton Variable* per le variabili d'automa (Rhino);
- metodo *Submit Transition Command* per le azioni che causano le transizioni (Httpunit).

Mozilla Rhino:

- Implementazione Java dell'interprete Javascript;
- Consente l'uso di script agli utenti finali;
- Crea un Contesto che memorizza informazioni circa l'ambiente d'esecuzione di uno script.
- Utile per la creazione e l'uso di variabili di variabili di contesto.

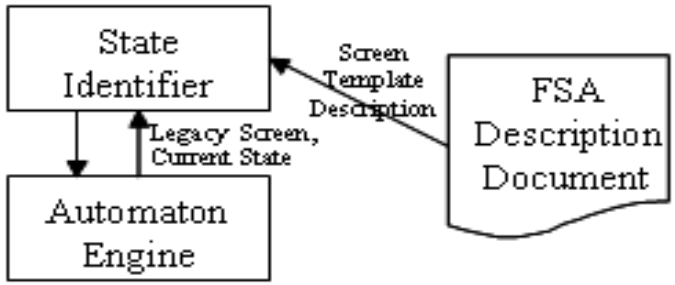
State Identifier

State Identifier:

- Le pagine HTML restituite dalla WA vengono convertite in XHTML (JTidy);
- Identifica l'IS con query XPath nel file XHTML;
- Le query XPath sono contenute nelle descrizioni degli Screen Template;
- lettura/scrittura su XML (JDom).

XPath:

- Basato su rappresentazione logica del documento;
- Individua elementi e attributi in un file XML tramite path;
- Fornisce funzionalità di manipolazione stringhe, numeri, …;



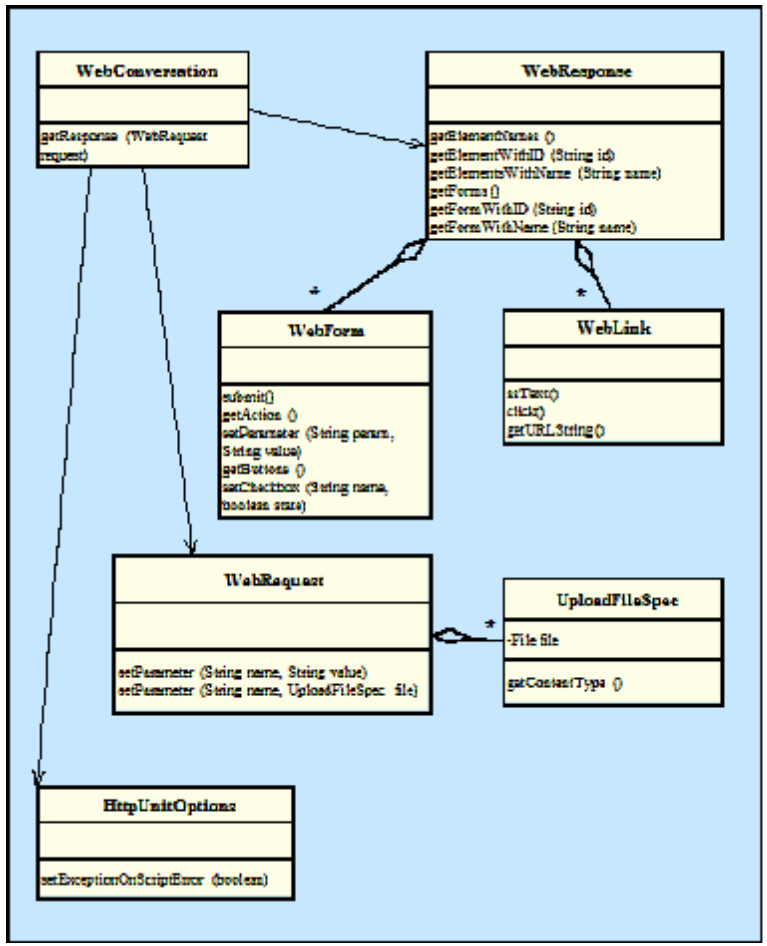
JTidy:

- Ispirata a Tidy;
- Può essere utilizzata all'interno di un programma Java;
- Trasforma un file HTML in XHTML per l'analisi mediante parser XML ;

JDom:

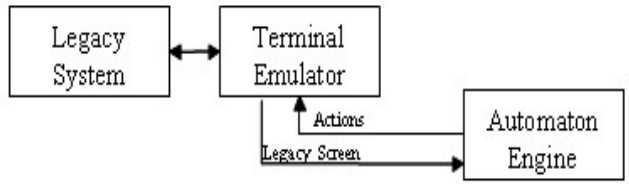
- Gestione file XML con meccanismi Java;
- Interagisce con SAX e DOM;
- Fornisce builder per il parsing dei documenti XML;

Terminal Emulator



Terminal Emulator:

- instaura la comunicazione con la WA;
- invia richieste HTTP (Httpunit), riceve pagine HTML e le trasforma in XML (JTidy).



Httpunit:

- simula tutti gli aspetti (es. GET e POST) di un browser che interagisce con un server web;
- permette di esaminare pagine restituite, forms, tabelle,links,... ;
- implementa funzionalità per la comunicazione.

Caso di studio: Show Cart

È stato eseguito un caso d'uso in accordo al processo di migrazione.

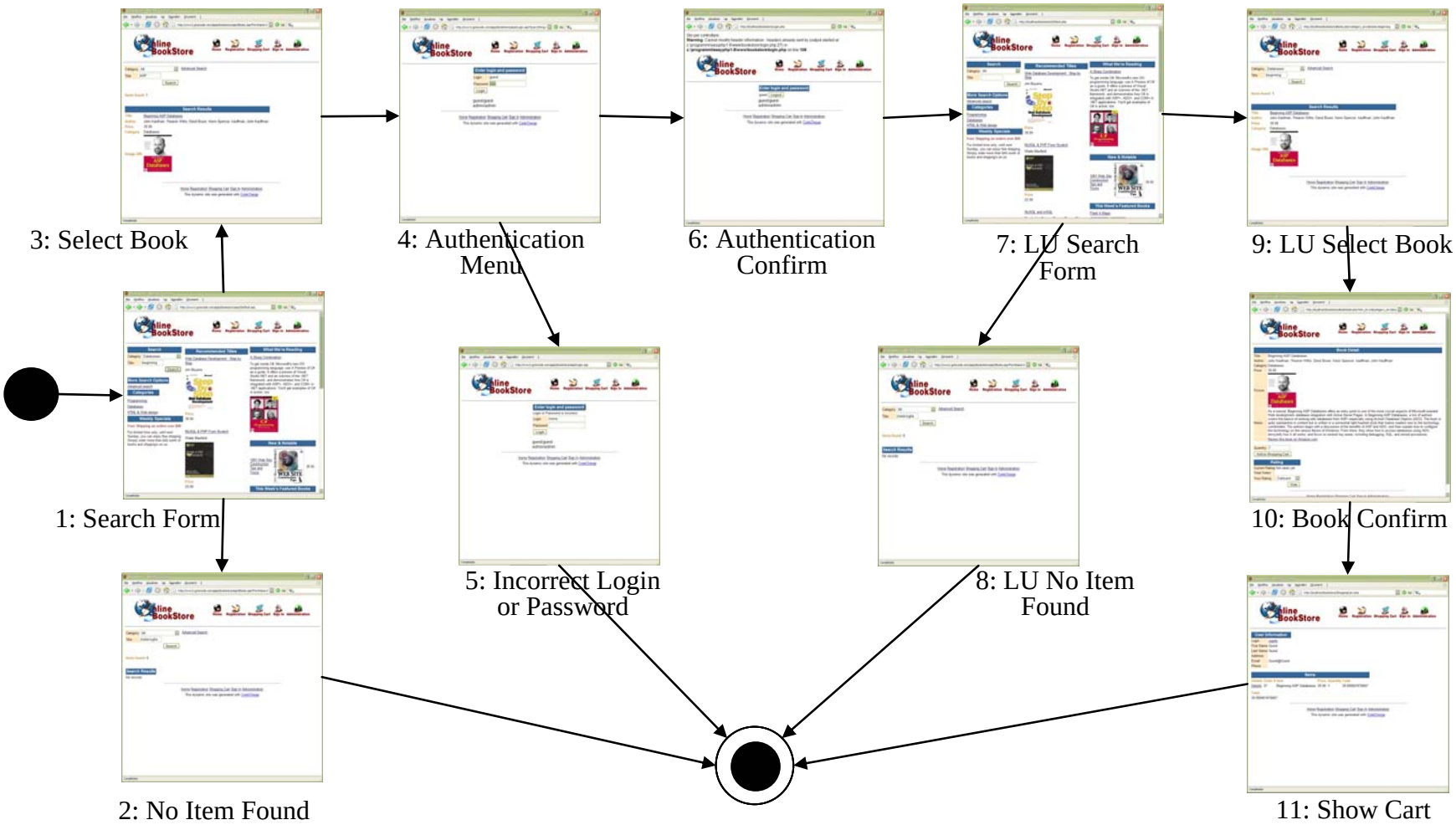
```
<lState number="11" name="Show Cart">
  <st>
    <field type="label">
      <name>Total</name>
      <type></type>
      <position>
        <absolute>not(contains(//table[position()=3]//table//tr[position()=1]//td//text(),'records'))</absolute>
      </position>
      <formcontainer></formcontainer>
    </field>
    <field type="output">
      <name>Total</name>
      <type></type>
      <position>
        <absolute>//table[position()=3]//table//tr[position()=2]//td//text()</absolute>
      </position>
      <formcontainer></formcontainer>
    </field>
  </st>
  <NextStates>
    <NState number="12"/>
  </NextStates>
<SUAV>Total=output_Total</SUAV>
</lState>
```

Caso d'uso:	Add to Shopping Cart
Precondizioni:	None
Input:	Address, name, category_id, Login, Password, quantity
Output:	Total, Exception
Postcondizioni:	None

StateID	11
ISDescription	Show Cart
Screen Template	
Actions	SUAV:Total=output_Total GOFV:Total
Submit command	nessuno
Next State	END

Caso di studio: Activity Diagram

La funzionalità *Add to Shopping Cart* di una Web Application Bookstore



Conclusioni

- È stato esteso un tool che implementa i metodi per l'accesso alle funzionalità di un sistema Legacy secondo un paradigma request/ response che rende possibile l'accesso tramite Web Services.

Sviluppi futuri

- Il tool potrà essere utilizzato come Web Service per svolgere la funzionalità offerta.
- Automatizzazione del processo di scrittura del FSA Description Document tramite l'identificazione dei campi discriminanti.
- Adeguamento a nuove tecnologie e paradigmi,