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

tesi di laurea

Un'architettura per la cooperazione di applicazioni: un approccio basato sulla migrazione di applicazioni Legacy

Anno Accademico 2005/2006

relatore
Ch.mo prof.ssa Valentina Casola

relatore
Ch.mo prof. Porfirio Tramontana

candidato
Rosa Guerra
Matr. 831/182

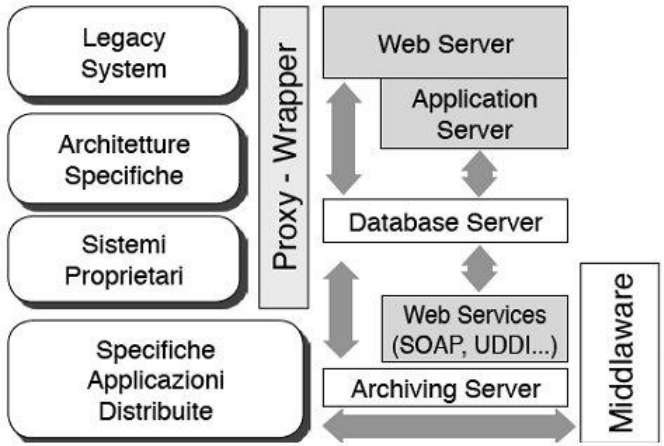
Panoramica

Migrazione di applicazioni Legacy

Contesto:
evoluzione dell'ambiente informatico verso un sistema altamente distribuito caratterizzato da innovazioni tecnologiche come i web services e coesistente necessità di sopravvivenza dei sistemi legacy.

Obiettivi:

- Riuso dei servizi di un Legacy System
- Passaggio da un paradigma interattivo ad un paradigma request/response;
- Interoperabilità e cooperazione applicativa;
- Rendere automatizzato il processo di intera-zione utente-applicazione.



Soluzione Adottata:
Progettazione e sviluppo di un componente software mediante la tecnica reverse engineering di tipo *black box*.

Il contesto

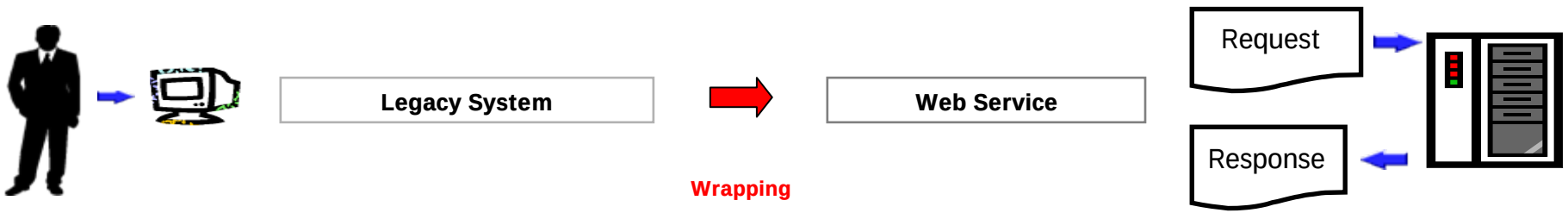
Riutilizzare e Migrare funzionalità di Web Application esistenti (legacy) verso il paradigma request/response

Legacy Applications:
- obsoleti;
- poco competitivi;
- documentazione inadeguata.

Validi motivi per il riuso:
- costi di progettazione proibitivi;
- disponibilità quasi pari al 100% (Pubblica Amministrazione, banche,…);
- funzionamento ancora soddisfacente;
- affidabilità.

Cooperazione applicativa è un paradigma per:
- l'erogazione e la fruizione di servizi;
- lo scambio di dati tra due o più Enti;
- la creazione di un sistema di cooperazione applicativa

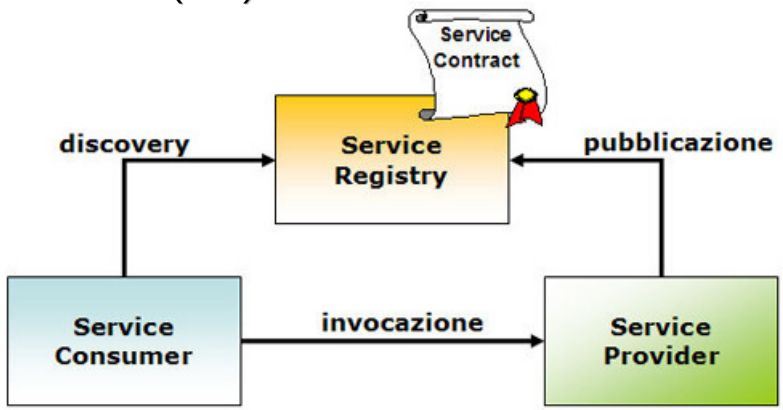
Inoltre, deve rispettare i requisiti di:
- **interoperabilità** tra i sistemi;
- **eterogeneità** delle piattaforme tecnologiche;
- **modularità** dei componenti;
- **integrabilità** con i sistemi preesistenti.



Service-Oriented Architecture

Una ***SOA*** è:

- un'architettura per l'interoperabilità di sistemi distribuiti (WS)
- orientata al riutilizzo e all'integrazione
- indipendente dalla tecnologia.



SOA mediante l'utilizzo di *Web Services* e di *protocolli standard* permette di effettuare in maniera più semplice ed efficace il processo di migrazione.

I Web Services:

- utilizzano standard e protocolli "open";
- possono essere facilmente utilizzati per formare servizi "integrati" e complessi;
- sono indipendenti dalle Web Applications da migrare;

I protocolli standard possono essere:

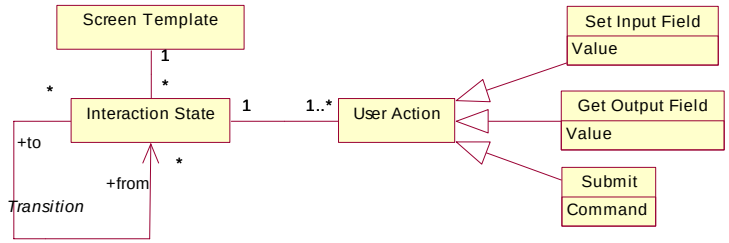
- per il trasporto dei messaggi: *HTTP*;
- per la formattazione dei dati scambiati: *XML*
- per la descrizione dell'interfaccia: *WSDL*
- per la descrizione e la localizzazione veloce: *UDDI*.

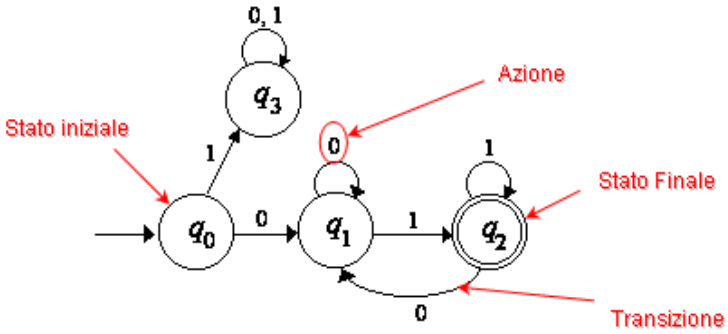
Dal modello d'interazione al Wrapper

È necessario ottenere un modello completo dell'interazione LS-utente per ogni caso d'uso (reverse engineering).

Viene creato un modello di interazione tra l'utente e la Web Application:

- Si analizza il comportamento dell'utente nell'esecuzione degli scenari del caso d'uso da migrare e lo si descrive con un FSA.
- vengono descritti tutti i possibili percorsi, in corrispondenza di uno o più casi d'uso (*use cases*);





Un **Finite State Automaton** è costituito da:

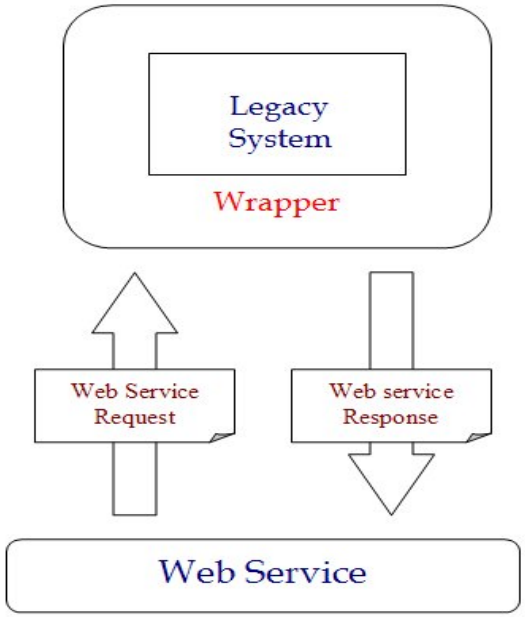
- *Stati di Interazione;*
- *Azioni;*
- *Transizioni tra gli stati;*
- *Stato iniziale e stato finale*.

Il Wrapper

Componente software che si pone da intermediario tra sistemi esistenti (LS) e le attuali tecnologie del Web (WS).

Requisiti:

- riusabilità;
- conoscenza possibili Next States (FSA non deterministico);
- capacità di identificare lo stato corrente tramite ST;
- il suo comportamento dipende dal caso d'uso.



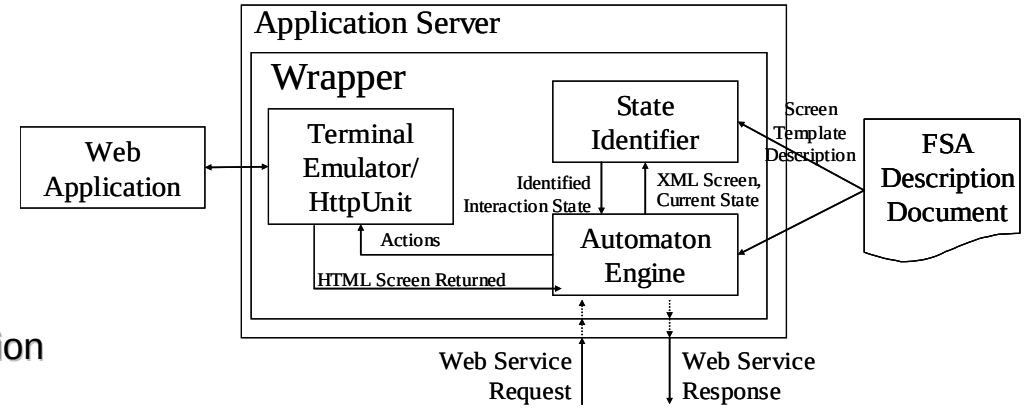
Il **Wrapper** deve:

- sostituire l'utente nell'interazione con la Web Application;
- leggere i dati di input contenuti in un messaggio di richiesta;
- guidare la WA nell'esecuzione di ogni possibile scenario di interazione;
- restituire un messaggio di output contenente i valori ricavati;
- fornisce servizi già attivi e funzionanti;

Architettura del Wrapper

Automaton Engine:

- responsabile dell'interpretazione del FSA.
- legge la richiesta e, per ogni stato, il comportamento da tenere su FSA Description.

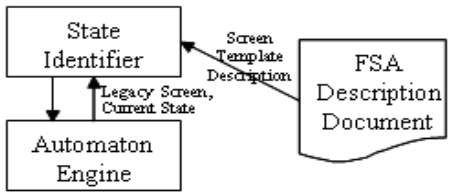


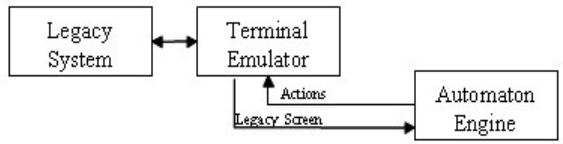
State Identifier:

- responsabile dell'identificazione dell'Interaction State.
- riceve una pagina XML da identificare
- conoscendo l'IS, legge sul FSA Description i possibili NS e i corrispondenti Screen Templates.
- confronta ed identifica la schermata corrente.
- riconosce gli output e comunica lo stato e gli output all'Automaton Engine (lettura/scrittura su XML: JDOM).

Terminal Emulator:

- responsabile della comunicazione tra wrapper e LS.
- ricevere comandi dall'Automaton Engine da eseguire;
- eseguirli sulla Web Application
- catturare la schermata HTML in risposta;
- restituirla all'Automaton Engine.





FSA Description Document

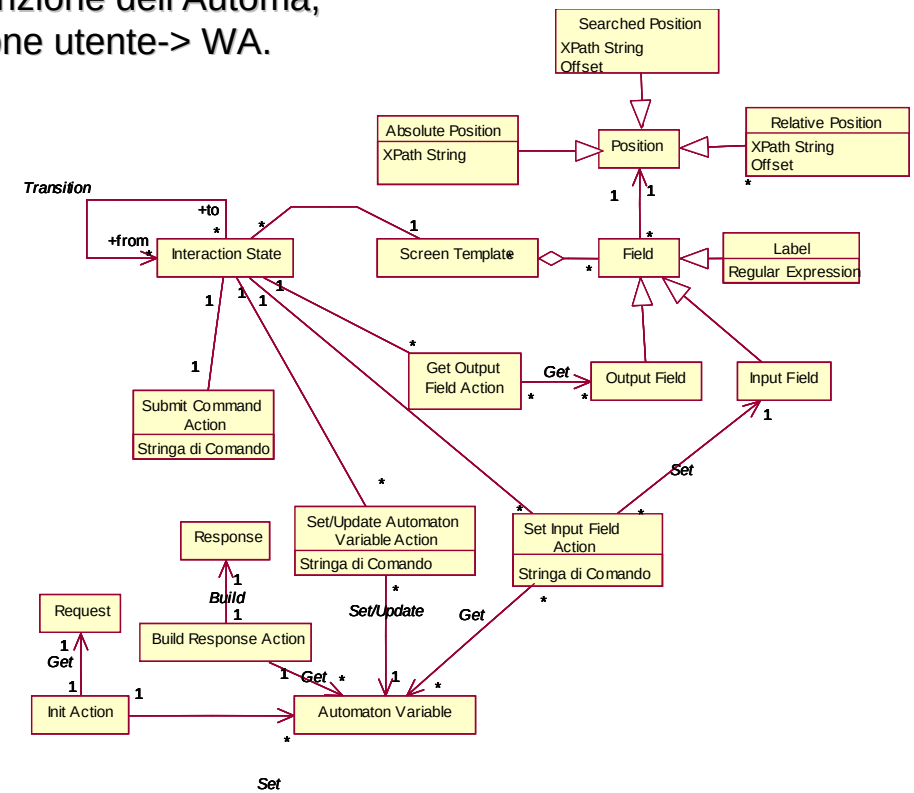
Il documento di descrizione dell'Automa

- Dipende dal caso d'uso da migrare e dal modello dell'Automa;
- E' composto da uno o più file XML;
- Memorizza i dati seguendo il modello di descrizione dell'Automa;
- È Scritto manualmente osservando l'interazione utente-> WA.

Il modello dell'automa è costituito da:

- Stati, a ciascuno è associato uno Screen Template;
- Transizioni tra stati;
- Azioni necessarie alla transizione;
- Stato Iniziale (legge e salva gli input);
- Stato Finale (genera il messaggio di output);

- Le azioni considerate sono: *Set Input Field Actions*,*Get Output Field Actions* e *Submit Command Actions*;
- Uno Screen Template è necessario per l'identificazione degli stati.

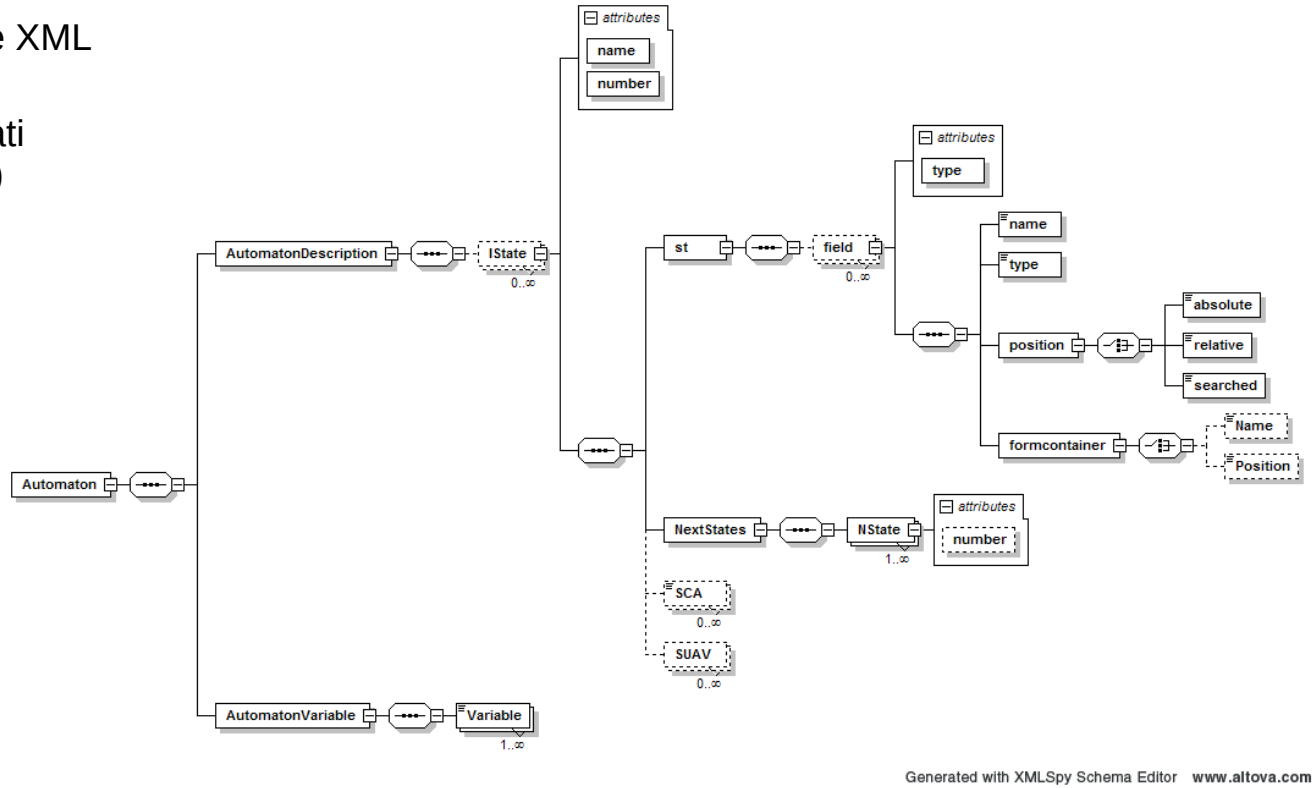


FSA: struttura XML

Sono state effettuate diverse modifiche rispetto al progetto di partenza.

FSA Description:un unico file XML per:

- la descrizione degli stati (Automaton Description)
- variabili d'automa (Automaton Variable).



- I tag *gofa* e *sifa* sono stati eliminati.
- I tag *sca* e *suav* sono stati resi opzionali
- È stata introdotta la possibilità di identificare gli eventuali form di una pagina HTML per *nome* o per *posizione*.

```
<formcontainer>
    <Position>0</Position>
</formcontainer>
```

```
<formcontainer>
<Name>Search</Name>
</formcontainer>
```

Conclusioni

- È stato presentato un metodo che consenta il riuso di funzionalità offerte da sistemi Legacy secondo un paradigma request/response, che consente l'accesso tramite Web Services.

Sviluppi futuri

- Esplorare la scalabilità dell'approccio per migrare funzionalità molto più complesse.
- Realizzazione di un tool che potrà essere utilizzato in applicazioni di più grandi dimensioni per svolgere la funzionalità offerta come un WS.