Turning Web Applications into Web Services by Wrapping Techniques

Giusy Di Lorenzo

Anna Rita Fasolino

Lorenzo Melcarne

Porfirio Tramontana

Valeria Vittorini

Dipartimento di Informatica e Sistemistica

University of Naples Federico II, Italy

Motivation

- With the diffusion of new paradigms and technological solutions for the Web (RIA and Ajax, SOA, ...), existing Web Applications are rapidly becoming legacy

- A strategic objective: to integrate existing Web applications with the new platforms

- An Open Issue:
  - *Turning Web applications functionalities into Web Services using systematic migration approaches* based on Wrapping techniques

Comparing Interaction paradigms...

Web Applications

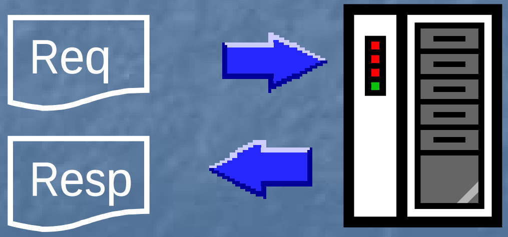
Users seeing an HTML page, fill input fields in a form and submit them, or click on a link in the page.

Web Server answers by producing a HTML page containing output values and new input fields and command buttons that is rendered by the browser

Web Services

A Client party invokes a service implemented by a provider party, using a request message.

The provider processes the request and sends a response message with the obtained results.



Turning WA to WS

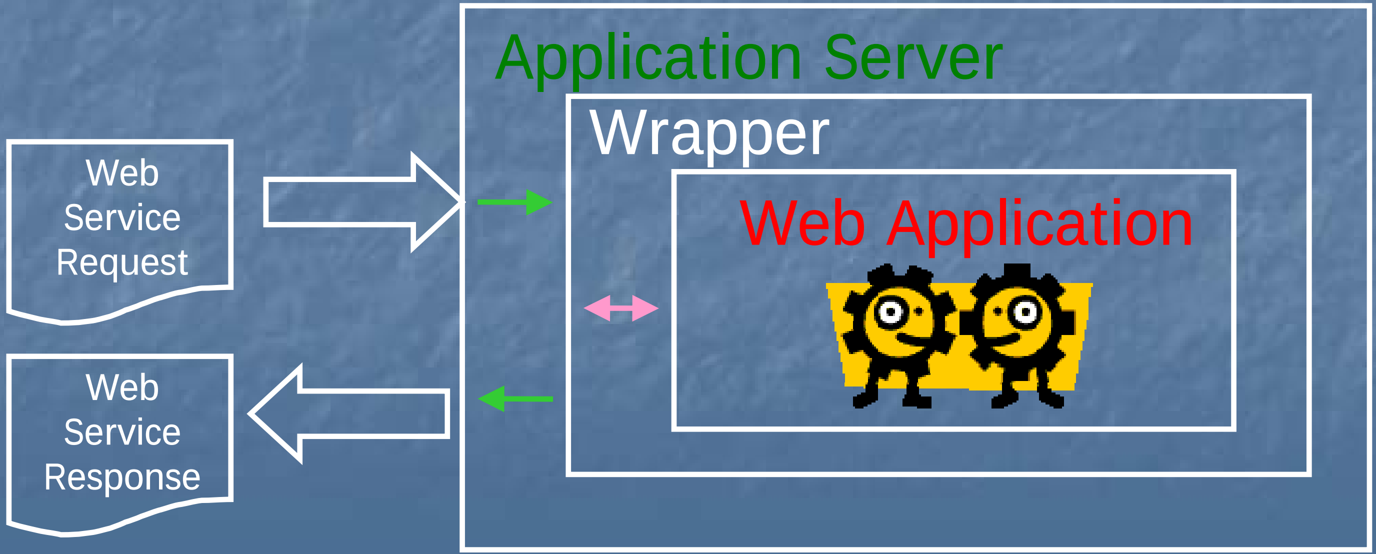
- Turning Web applications functionalities into Web services
  - Transforming the original (non programmatic) user-oriented interface of the Web application into a programmatic interface that exposes the full functionality and data of the application and that can be suitable for:
    - Web Services
    - APIs (e.g. accessible via Ajax requests)

Migration solutions

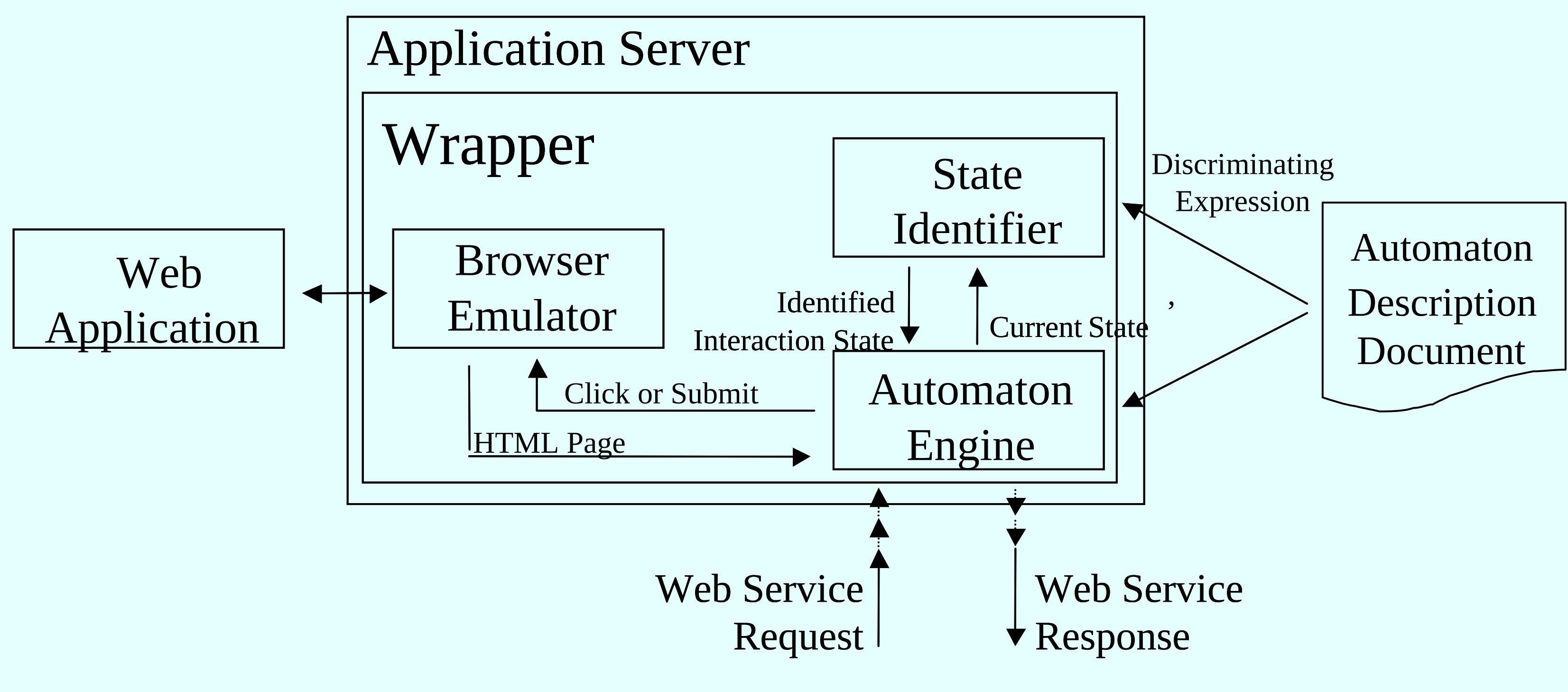
- White Box
  - Invasive; they depend on the specific languages and technologies adopted on the server side of the Web Application
- Black Box
  - Web Wrappers solutions
  - Baumgartner et al., 2004
    - Web Services returning output data of Web pages obtained via visual selection
  - Jiang and Stroulia 2004
    - Web Services returning output data of Web pages obtained via http request/response analysis
  - Canfora et al. 2006
    - Web Services executing functionalities of form based legacy system via wrappers

Our Wrapper solution

- The goal of our wrapper is to drive the Web Application during the execution of each possible interaction scenario associated with the use case to migrate, by providing it with the needed flow of http requests
- The Wrapper behaviour must replicate all the possible behaviour of the user of the Web application in the execution of the functionality to migrate



Wrapper Architecture

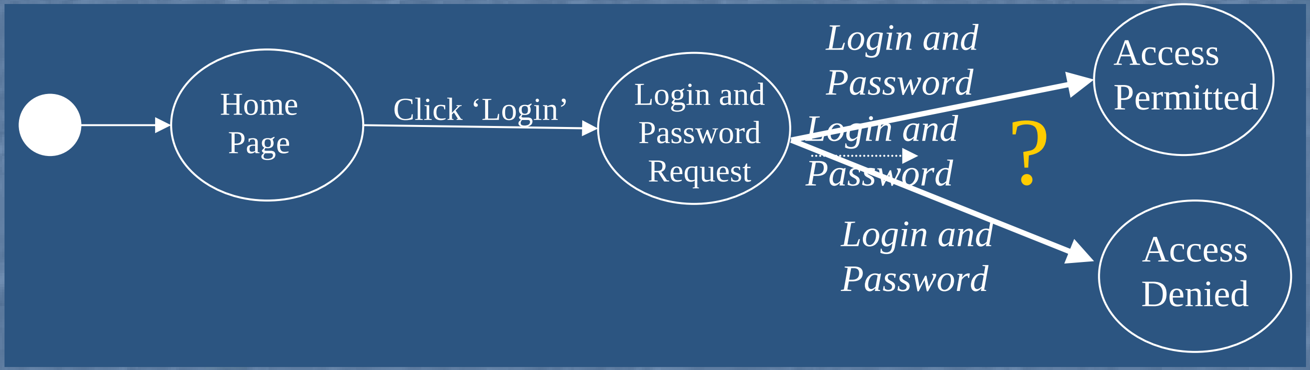


- Http Unit acts as a Browser Emulator
- A XPath Library (such as Jaxp) supports the Class Identifier
- The behaviour of the wrapper for a given functionality to migrate is completely described in the corresponding Automaton Description Document

The Model of the Interaction

- A Finite State Automaton FSA= (S, T, A, Sin, Sfin) where:
  - S is the set of Interaction States; they may be:
    - Input States
    - Output States
    - I/O States
    - Exception States
  - A is the set of Actions performed by the user when an Interaction State occurs,
  - T is the set of Transitions between states,
  - Sin and Sfin are the Initial and Final states of the interaction.

Non Deterministic finite State Automata



What is the next State?

- ➢ The next state depends on the internal logic or on the internal state of the legacy system.
- ➢ Non Deterministic Finite State Automaton (NFA) is a finite state machine where for each pair of state and input symbol there may be several possible next states
- ➢ The wrapper must know the list of the possible Next States of a given State
  - Possible successors of Password Request State are Access Permitted and Access Denied states
- ➢ The wrapper must be able to identify the current state on the basis of the returned screen
  - Wrapper must discriminate among Access Permitted screen and Access Denied screen

The proposed Migration process

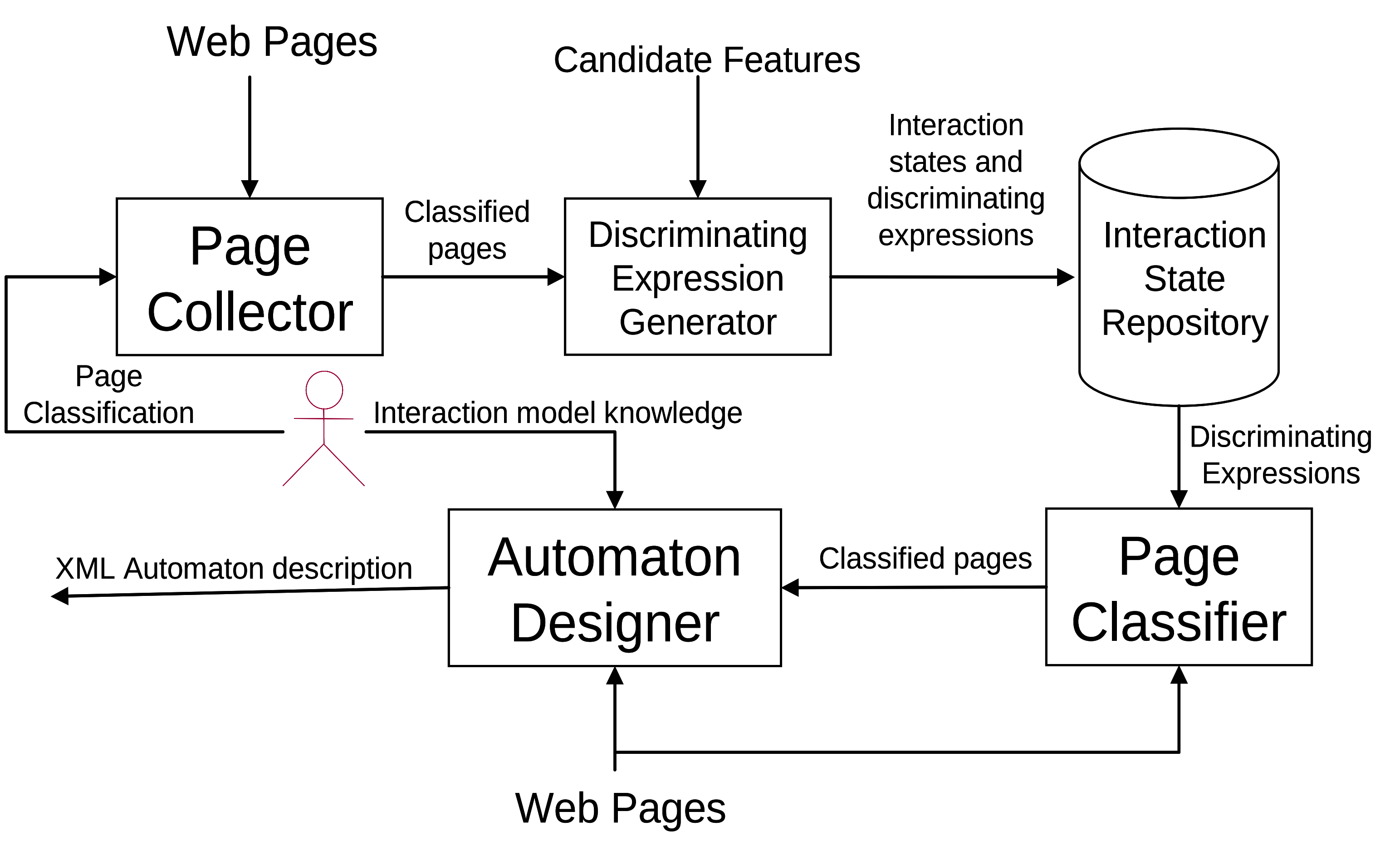
1. *Selection* of the Web application functionality to be turned into a Web service, based on:
   - *Business Value*
   - *Potential reusability*
   - *Cohesion and State Independence*

2. *Reverse Engineering* of the Web application User Interface
   1. Identification of execution scenarios
   2. Characterisation of execution scenario steps

3. *Interaction Model design*
   1. Evaluation of alternative modelling solutions
   2. XML-based design of the model

4. *Wrapper Deploy and Validation*

The Migration Toolkit


Web Pages
Candidate Features
Page Collector
Classified pages
Discriminating Expression Generator
Interaction states and discriminating expressions
Interaction State Repository
Page Classification
Interaction model knowledge
Discriminating Expressions
XML Automaton description
Automaton Designer
Classified pages
Page Classifier
Web Pages

This toolkit supports the migration process steps needed to produce the Automaton description needed by the Wrapper

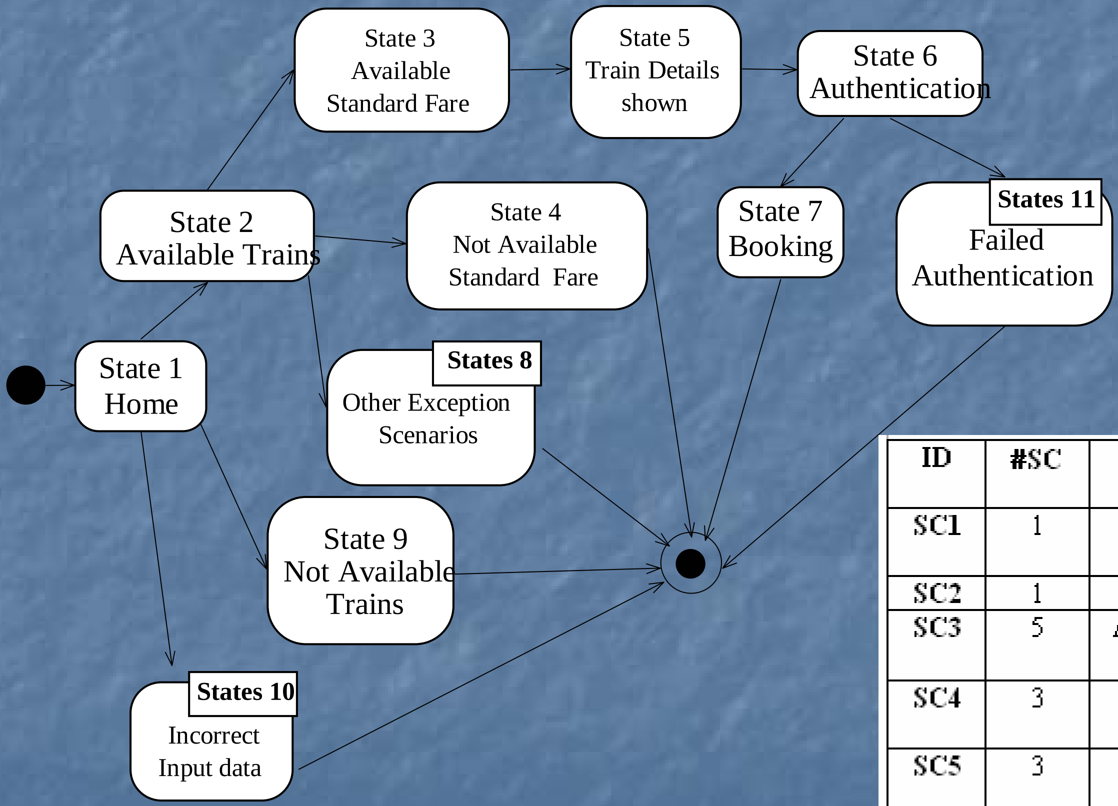
Case studies

- Based on a real italian Web Application for train journeys planning and booking
- 3 case studies have been carried out n order to:
  - Explore the feasibility of the wrapping approach
  - Explore the different migration alternatives
    - Service BS1 realise a simple train booking service
    - Services BS2 and BS2* realise a more complex service in two different manners

Booking Service #1

- Input:
  - Departure Station – Arrive Station – Date and Starting Time – Login – Password
- Behaviour:
  - The system books one seat on the first available train solution listed in the timetable iff it can be purchased on-line and the Standard fare can be applied
- Output:
  - Train number – Departure Time – Coach number – seat number – Exception description

BS1 Automaton



- No preconditions
- 19 Interaction States
- 14 scenarios
  - 1 successful
  - 13 exceptions

ID	#SC	Description	Interaction States
SC1	1	Seat booked with success	S-1-2-3-5-6-7-E
SC2	1	Not Available Trains	S-1-9-E
SC3	5	Available Trains but other journey search criteria cannot be satisfied	S-1-2-8-E
SC4	3	Failed User Authentication	S-1-2-3-5-6-11-E
SC5	3	Wrong or Incomplete journey specification	S-1-10-E
SC6	1	Not Available trains with Standard Fare	S-1-2-4-E

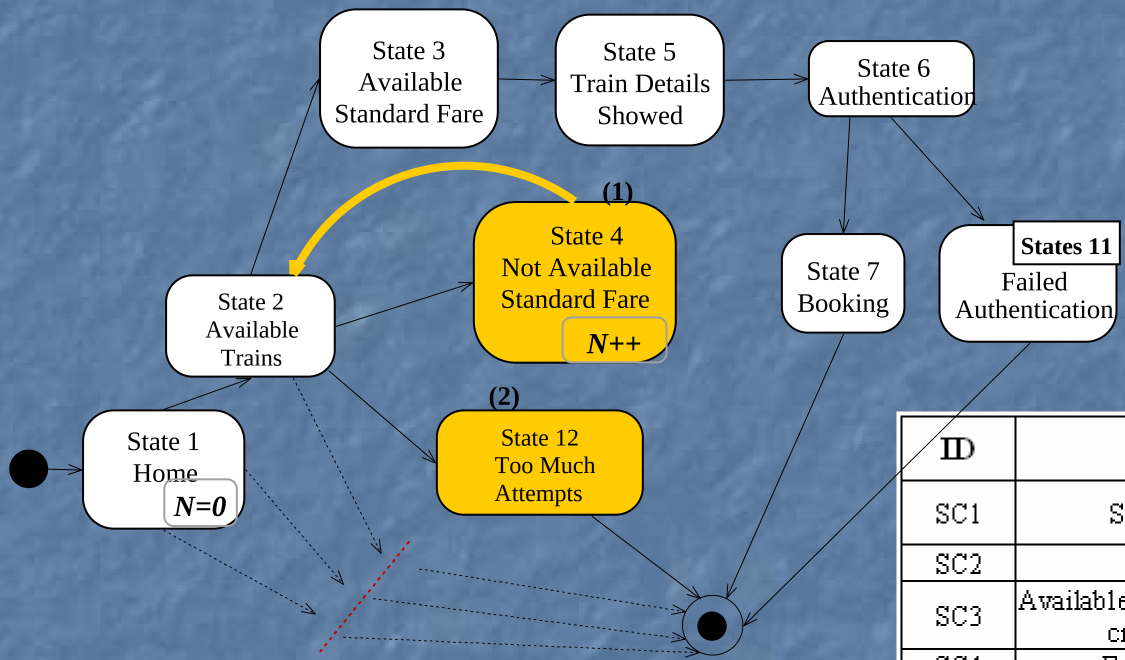
Booking Service #2

- Input:
  - Departure Station – Arrive Station – Date and Starting Time – Login – Password - NMax
- Behaviour:
  - The system books one seat on the first available train solution listed in the timetable iff it can be purchased on-line, the Standard fare can be applied and it is in the first NMax listed solutions
- Output:
  - Train number – Departure Time – Coach number – seat number – Exception description

BS2 realisation

- Problem:
  - The interaction model need to maintain some information about the current state of the interaction
    - Number of done searching attempts
  - Some pieces of logic must be added to the automaton
    - If the number of attempts N is equal to the number of maximum allowed attempts NMax then go to an exception state else decrease the departure time and retry
  - 2 possible solution:
    - Put the logic in the automaton
    - Put the logic in a Workflow

BS2 – Automaton solution



State 1 Home
N=0

State 2 Available Trains

State 3 Available Standard Fare

State 4 Not Available Standard Fare
N++

(1)

State 5 Train Details Showed

State 6 Authentication

State 7 Booking

States 11 Failed Authentication

(2)

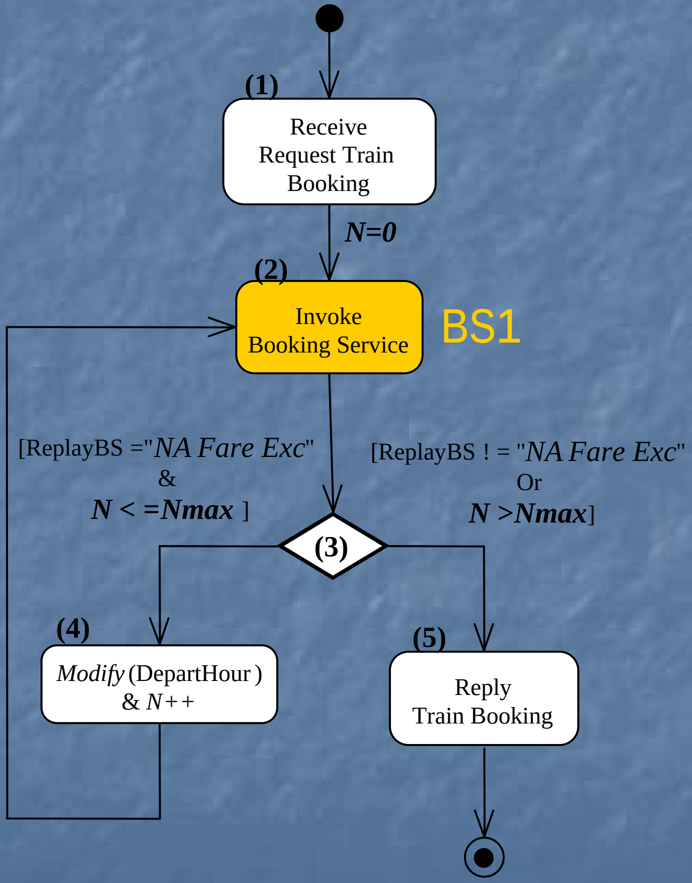
State 12 Too Much Attempts

- An Automaton Variable N is managed, representing the actual attempt number.
- The value of Departure Time is decreased in State 4

ID	Description	Interaction States
SC1	Seat booked with success	S-1-2-3-5-6-7-E
SC2	Not Available Trains	S-1-9-E
SC3	Available Trains but other journey search criteria cannot be satisfied	S-1-2-8-E
SC4	Failed User Authentication	S-1-2-3-5-6-11-E
SC5	Wrong or Incomplete journey specification	S-1-10-E
SC6	Not Available trains with Standard Fare	S-1-2-12-E
SC7	Booking of the second available train in timetable	S-1-2-4-2-3-5-6-7-E
SC8	Booking of the fourth available train in timetable	S-1-2-4-2-4-2-4-2-4-3-5-6-7-E

- States 4 and 12 hthe State Identifier has the same interface: the State Identifier can discriminate between them by evaluating the value of the Automaton Variable N:
    - N=NMax ➔ State 12
    - N<NMax ➔ State 4

BS2* – Workflow Solution


(1)
Receive Request Train Booking
N=0
(2)
Invoke Booking Service
BS1
[ReplayBS ="NA Fare Exc" & N < =Nmax]
[ReplayBS ! = "NA Fare Exc" Or N >Nmax]
(3)
(4)
Modify (DepartHour) & N++
(5)
Reply Train Booking

- Implemented with BPEL
- Reuses the basic service BS1
- The other activities model the business logic of BS2
  - Realised as BPEL directives or other Web Services

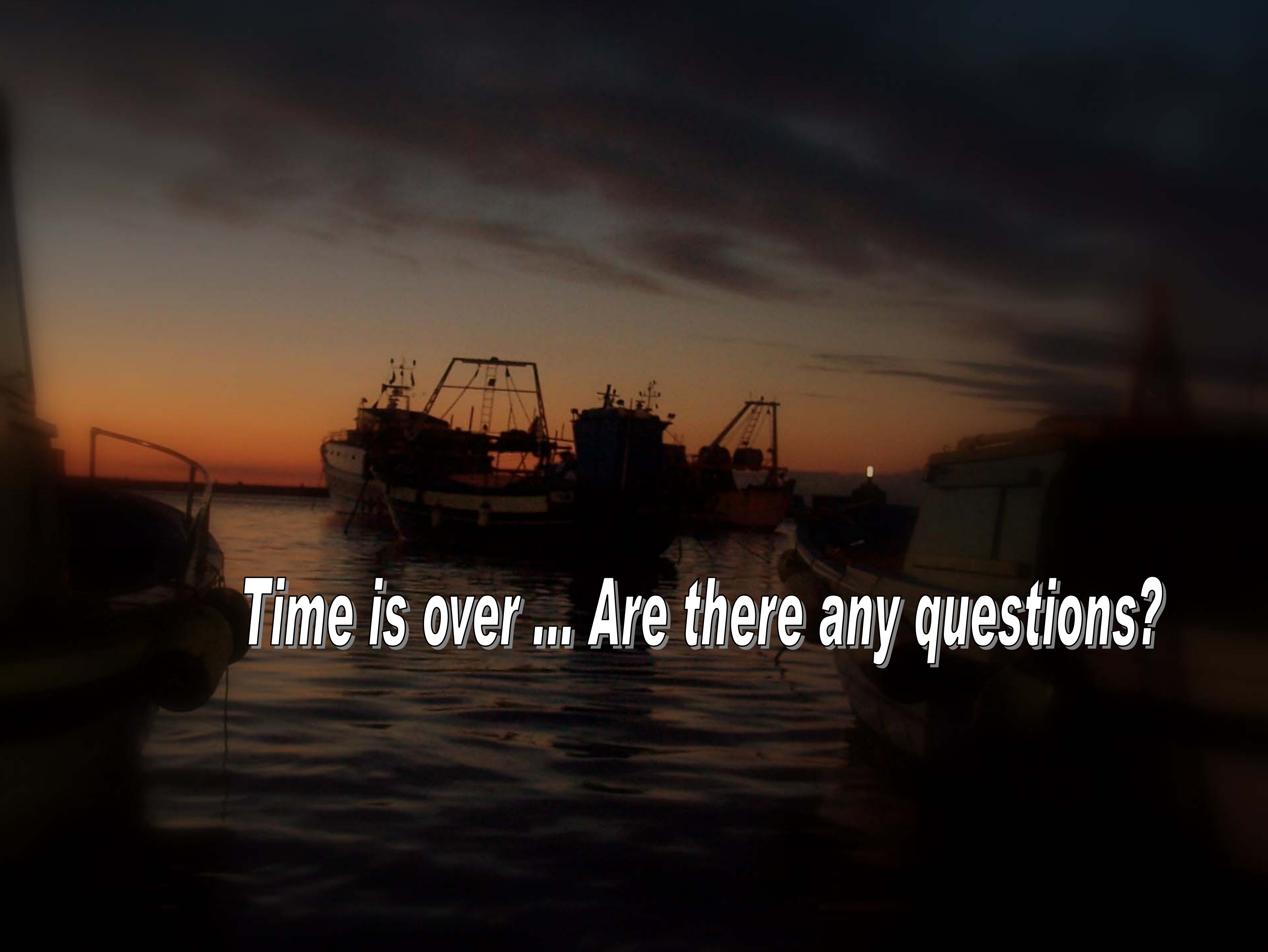
Discussion

- BS2, Automaton Solution
  - ☺Better performances
  - ☺Can reuse transition isolation mechanisms that could be implemented in the existing WA
  - ☹Larger Automaton
  - ☹Much effort in Automaton design and testing

- BS2*, Workflow Solution
  - ☺Easier to implement and test (for peoples that are skilled in workflow design!)
  - ☺Possibility to reuse existing services
  - ☹Transition isolation mechanisms should be implemented in the workflow
  - ☹Worst performances

Conclusions

- A tool supported process for wrapping based migration of functionalities of Web Application to Web Services has been proposed
- Reported case studies show the feasibility of the wrapper solution in real problems
- Comparisons between different possible solutions have been proposed
- Future Works:
  - Support to existing RIAs
  - Improvements of process steps in order to reduce effort
  - Training of peoples in order to realise empirical studies for evaluation and comparison of the needed effort



Time is over ... Are there any questions?