Sample Project:
Travel Agency System (TAS)

Behrouz Far
Schulich School of Engineering, University of Calgary
far@ucalgary.ca
http://www.enel.ucalgary.ca/People/far/

Contents

- Sample Project
  - System Specifications
  - Design Document
  - Detailed Design Document
  - Data Dictionary
  - Inter-Agents Messages
Travel Agency System (TAS)

System Specifications

1. Business Case

- The travel industry is increasingly becoming internationalized. Airlines, accommodators and tour operators control the market, but these retailers face high costs in communication and personnel.
- Communication is one of the main problems in the travel industry.
- Many travel agents retain a great deal of information about their clients on individual preferences and do not share it with other agencies. Because this information translates into business, it is usually accessible only to individual travel companies, and not the industry as a whole. For example, airlines have their own portal where they offer travel packages to their customers on their fleet.
1. Business Case (cont’d)

- The customers adopt the Internet travel sites model as a medium to purchase a travel package. Customers, usually looking for holiday packages, depend on their travel agent to show them what is available according to their taste and budget, or visit specific content providers, such as airlines, accommodators and car rental agencies.

- The existing travel reservation systems only show the customer the different travel packages. The customer has to compare the different travel packages on his/her own and make his/her decision.

- In this environment, intelligent agents have a great potential in helping the customer get the best ‘deal’ on a travel package and help him/her book the selected travel product.

2. System Description

- The proposed Travel Agency System (TAS) is a multi-agent system designed to obtain travel packages for user depending on their preferences.

- The travel package is composed of a flight ticket, a hotel accommodation, and a car rental.

- The TAS application deals with multiple airline companies, hotels, and car rentals to find the optimum package for its user.

- The application has to schedule the travel and proceed with the transactions related to the purchase of the travel package.
2. System Description (Cont’d)

- USER
  - GUI
  - Travel Agency Agents

- INTERNET
  - Flight Web Services
  - Hotel Web Services
  - Car Rental Web Services

- User Travel Agency Database

3. Assumptions

- The system maintains a user profile where each client has his or her own individual preferences, such as:
  - Hotel preferences (i.e., rating)
  - Car rental preferences (i.e., car model)
  - Period when the system shall search for a flight deal (e.g., every Sunday)
  - Period when the system shall remind the user of a flight departure (e.g., two hours before a flight departure).

- The user requests a travel package through a user interface. The user interface captures the flight information. Hotel and car inputs are retrieved from the user profile, accordingly.

- The user must book a complete package (flight, hotel and car rental).

- There are already predefined web services on the Internet which the Travel Agency makes use of. These web services have the same WSDL description meaning their input and output messages have a standard syntax.
4. Requirements

- The TAS shall provide flight, hotel accommodation and car rental enquiries.
- The TAS shall deal with different airline, hotel and car rental companies.
- The TAS shall verify the user schedule to schedule a travel.
- The TAS shall book the flight, the hotel accommodation and the rental car upon the approval by the user.
- The TAS shall send a memo to the user about the travel information and alert user before the flight. The period before which the system shall remind the user is specified in the user profile.
- The TAS shall automatically enquire for flights depending on the user destination preferences and his/her flight booking history. The period when the system shall initiate an enquiry to get a flight ‘deal’ is specified in the user profile.

5. Wish List (Not Implemented)

- The Travel Agency System shall allow a user book a flight, hotel, and a car rental separately.
- Facilitates the setting of Insurance, Administration Charges, Confirmation and Cancellation policy, Vacation packages, international taxes or other related supplements.
- During any booking any flight, car or accommodation unit is held off from the main inventory the moment it is selected, removing the possibility of overbooking from a multi-user system.
- The System shall check the flight Web Service for any delay in flight departure and automatically alert user if any.
- The Agents should use lookup services to locate Web services, and their selection of Web services should be based on trusted rating services.
- The Travel Agency System shall allow the user choose a specific carrier and the connectivity cities.
How to Proceed?

- Follow the agent-based development methodology and create the analysis and design documents specified by the methodology.
- Typical tasks include:
  - Role identification
  - Agent system architecture
  - Agent description
  - Agent internal architecture
System Architecture

- The way this system is envisioned to work is that there is a layer of multi-agent system in between the client (browser) and already existing Web Services.
- The Web Services may come from one or more providers and they return the respective quotations of their service area to the agents based on client input and user profile.
- The agents then try to sort the results in a way that the client gets the best price for the travel package and matches his/her time constraints.

Role Identification

- What roles are required?
  - Personal assistance role
  - Scheduling role
  - Web-services handling roles: hotel, flight, car

- Assign roles to agents
In this system we have the client submitting his/her request on an Internet browser interface. The main input screen is focused on the flight and retrieves only flight information from the user. Hotel and Car information are pre-stored in the TAS database as user preferences.

Any user that wants to use the Travel Agency System services has to create his/her own profile before hand.

The multi-agent system has to interact with Web Services to get the user a best ‘deal’ travel package, and with the user’s scheduler tool to schedule a travel and display the booked travel package information.
Agent Description

Travel Agent

- The Travel Agent intercepts the request sent by the user through the Internet browser. In our architecture the Travel Agent handles the correspondence with all other agents and in this way we have a central point of delegation.
- The Travel Agent requests flight, hotel rooms, and car rentals listings from the adequate agents and displays the different travel packages on the user’s Internet browser. Moreover, the Travel Agent requests from the Flight, Hotel, and Car Rental agents to book the respective travel package selected by the user (i.e. flight, hotel room, and car rental).
- The Travel Agent generates a Memo that contains the information on the travel package, and sends it to the Scheduler Agent.
- Finally, the Travel Agent has to trigger a search for an automatic enquiry for a flight ‘deal’ based on the user flight history and period scheduled by the user stored in the TAS database.

Scheduler Agent

- The Scheduler Agent communicates with the Travel Agent. Once the user has specified his/her flight date and time preferences the Travel Agent talks to the Scheduler Agent in order to check if the client schedule is free on that date and time. Instead of the customer checking his/her own schedule before making flight decisions the Scheduler Agent does that on behalf of the user.
- The Scheduler Agent also handles conflicts in user appointments. Flight booking takes precedence over any low priority events. The user low priority events are rescheduled and the user is notified. If the user selects a flight offered by the TAS and a conflict is caused with a high priority event then the Scheduler Agent prompts the user to reschedule himself/herself his/her appointments.
- The Scheduler Agent makes sure that the date and time is blocked in the user’s scheduler once the travel package is booked. Instead of the user blocking his/her calendar and scribbling the details of his/her travel, the Scheduler Agent takes on that responsibility and blocks the time and fills in all the details of travelling (flight, hotel and car).
- The Scheduler Agent also triggers a reminder to alert the user about his/her departure date and airport arrival time based on the time the user wants to be notified (i.e. two hours before flight departure).
- Finally, the Scheduler Agent displays the Memo sent by the Travel Agent on the user’s scheduler tool browser.
Agent Description (cont’d)

**Flight Agent**

As the name suggests this agent take the responsibility of making flight arrangements for the user based on the information he/she submitted through the Internet browser interface. It takes the inquiry request from the Travel Agent and then communicates with the predefined Flight Web Services on the Internet. The Flight Web Services offered by various providers return their quotations on the flight. The Flight Agent retrieves these responses, parses them, sorts them, and passes them to the Travel Agent, which then presents it to the user.

The Flight Agent is also responsible of booking the flight selected by the user. This is accomplished by getting all payment information from TAS database and then passes it to the Flight Web Service, which in turn returns the confirmation for the flight. The Flight Agent returns the flight confirmation number to the Travel Agent.

**Hotel Agent**

The Hotel Agent makes hotel room arrangements for the user based on his/her hotel preferences that are stored in the TAS database. It takes the inquiry request from the Travel Agent and then communicates with the predefined Hotel Web Services on the Internet. The Hotel Web Services offered by various providers return their quotations on hotel rooms. The Hotel Agent retrieves these responses, parses them, sorts them, and passes them to the Travel Agent, which then presents it to the user.

The Hotel Agent is also responsible of booking the hotel room selected by the user. This is accomplished by getting all payment information from TAS database and then passes it to the Hotel Web Service, which in turn returns the confirmation for the booked hotel room. The Hotel Agent returns the hotel room confirmation number to the Travel Agent.
Agent Description (cont’d)

**Car Rental Agent**

- The Car Rental Agent makes car rental arrangements for the user based on his/her car rental preferences that are stored in the TAS database. It takes the inquiry request from the Travel Agent and then communicates with the predefined Car Rental Web Services on the Internet. The Car Rental Web Services offered by various providers return their quotations on car rentals. The Car Rental Agent retrieves these responses, parses them, sorts them, and passes them to the Travel Agent, which then presents it to the user.

- The Car Rental Agent is also responsible of booking the car rental selected by the user. This is accomplished by getting all payment information from TAS database and then passes it to the Car Rental Web Service, which in turn returns the confirmation for the booked car rental. The Car Rental Agent returns the car rental confirmation number to the Travel Agent.

Agent Internal Architecture

[Diagram showing the internal architecture of the agent system with nodes for Listener, Interpreter, Processor, Discovery Agent, Rendering Agent, Travel Agency System user, and Web Services connected by arrows indicating flow and interaction.]
Agent Internal Architecture

**Listener:**
- The *Listener* component listens to a port for any incoming requests from the TAS application.

**Interpreter**
- The *Interpreter* parses and interprets the XML messages. We assume that all agents have agreed on a Document Type Definition (DTD).

**Processor:**
- The *Processor* receives an XML document as an input. It uses the Interpreter to parse the document, and calls the appropriate function to run a process.

**Discovery Agent:**
- The *Discovery Agent* provides the service discovery base-service (a superset of UDDI). *May be implemented as an external service.*

**Rendering Agent:**
- The Rendering Agent is optional; it can be used by the Processor to render data before sending it back to the calling function. *May be implemented as an external service.*

Technology Overview

**Approaches to Service Discovery**
- Two categories of information must be discovered before a service is used:
  - Logical information
  - Technical information
- The logical information provides a description of the nature of the service, such as business information.
- The technical information describes the interface parameters of the logic that drives the service. Technical information, such as the exact protocol that the service uses, the inputs and outputs, and the encoding of messages, form a crucial part in machine-to-machine conversation.
Technology Overview

Approaches to Service Discovery

- A service must be visible to be discovered. In much the same way businesses list themselves in directories, such as the yellow pages, and mount signs on their entrances describing the nature of their business, Web services must provide short descriptions (on the Web) of themselves.

- Roughly, there are three major ways by which Web services can be described and, consequently, discovered:
  - Universal Description, Discovery, and Integration (UDDI) and Web Service Description Language (WSDL), the yellow pages approach
  - World Wide Consortium’s (W3C) Resource Description Framework (RDF), the sign mounting approach
  - JINI discovery, the Java code discovery approach

1. UDDI and WSDL

- Universal Description, Discovery, and Integration (UDDI) with Web Service Description Language (WSDL) are analogous to the yellow pages approach. UDDI is a set of specifications for describing services and registries that maintain these descriptions. WSDL is where the technical information of a service is described. This approach depends on worldwide standardization between businesses and the limitations of directories. Not all businesses are listed in the yellow pages and depending on such semi-centralized approach provides the potential for monopolizing the directories resulting in charges for businesses and clients. However, the yellow pages approach has proved to be a success and, we believe, will continue to be so in the future.

- For the scope of this project, we consider that Web Services are predefined and they all use a standard WSDL description document. UDDI approach is in our Wish List and we assume that each TAS agent will use a Discovery Agent to locate Web Services.
2. RDF

- The World Wide Web Consortium’s (W3C) Resource Description Framework (RDF) is analogous to the sign mounting approach.
- RDF provides the specifications and mechanisms to provide meta-data (data describing data), and thus the ability to describe both logical and technical information of services.
- Although RDF can be used as the framework for describing services within UDDI registries, RDF does not require such registries. RDF meta-data describing a given service will be stored within the service itself.
- The next-generation search engines will be used to walk through these descriptions and discover ample information describing different aspects of the service. Such an approach will be another huge success.

3. JINI

- In the JINI approach to Web service description and discovery, each JINI service has a proxy object. These are objects that the client uses in order to interact with the service. These proxies are maintained by a special service, called the lookup service. A JINI service advertises itself to other services by publishing its proxy with the lookup service. Since lookup is another service, services need to find and download its proxy so that they can use it. JINI solves this problem by building communities of JINI services. One way to assemble communities is by having these services (including lookup) periodically send messages over the network allowing them to be aware of each other. When a JINI service is discovered, its proxy is downloaded to the client, which need only know how to interact with the proxy object.
- JINI proxies are Java objects and, therefore, require a Java Virtual Machine (JVM) running on the client for desired service to be usable by the client. In fact, the JINI clients must be written in Java and, even though a service may be written in any other language, a Java Surrogate is needed for JINI clients to be able to talk to these services.
Communication protocol: SOAP

- Simple Object Access Protocol (SOAP) is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts:
  - An envelope that defines a framework for describing what is in a message and how to process it
  - A set of encoding rules for expressing instances of application-defined data types
  - A convention for representing remote procedure calls and responses.
- SOAP is proposed to encode an HTTP header and an XML message so that the TAS agents can call and pass information to each other and communicate with external Web Services.
Detailed Design

- Use cases and use case definition for all the participating agents should be documented.

- This includes
  - Use Case for Travel Agent
  - Use Case for Scheduler Agent
  - Use Case for Flight Agent
  - Use Case for Hotel Agent
  - Use Case for Booking Car Agent

Use Case: Travel Agent

<<Enquires about flight, hotel & car>>

<<Displays travel packages to user>>

<<Books flight, hotel & car>>

<<Remind user of his/her flight schedule>>

<<Triggers a request for a flight deal>>
**Use Case Def. : Travel Agent**

**Brief Description:**
The Actor uses this use case to request and book a travel package.

**Precondition(s):**
User Profile is created before providing any service.

**Post condition(s):**
If all the business rules are successfully met, than actor will be able to avail the facilities provided by the travel agency.

**Process Steps**
1. Actor makes a request for travelling by providing date for departure, departure location, arrival location, one way or two way, date of return, business class or economy class through a browser interface.
2. Travel Agent collects the user preferences for Hotel and Car Rental from user profile.
3. Travel Agent requests from Scheduler Agent to check schedule availability and manage conflicts if any (described in U002).
4. Travel Agent requests from Flight Agent to get a list of proposed flights from Flight Web Services.
5. Travel Agent requests from Hotel Agent to get a list of proposed hotels Hotel Web Services.
6. Travel Agent request from Car Rental Agent to get a list of proposed car rentals from Car Rental Web Services.
7. Travel Agent displays list of proposed flights, hotels, and car rentals on user’s browser.
8. Actor selects a flight, hotel, and a car rental from the proposed list on his/her browser.
9. Travel Agent requests from Flight, Hotel, and Car Rental Agents to respectively book flight, hotel and car rental.
   - Travel Agent generates a Memo composed of travel package information and confirmation numbers and sends it to Scheduler Agent.

**Exceptions**
1a. System is not accessible at this point. Error message is generated stating that System is not accessible at this point. Use Case Terminated.
4a. System can not find the flight other than waiting list ones. Message is generated, Use Case Terminated. In this phase of design we are only considering the flights that are fully open.

**Relationships:**
- **Initiating**: Actor
- **Collaborating**: Scheduler Agent, Flight Agent, Hotel Agent, Car Rental Agent

**Other Diagrams:**

**Data Requirements**
- **Data required for Travel Agent**:
  - Flight information
  - Hotel and Car rental preferences
  - Memo that contains information and confirmation number of flight, hotel and car rental booked
Use Case: Scheduler Agent

<<Handle Conflicts >>

Scheduler Agent

<<Reminds user of his/her flight schedules and displays Travel Package information Memos >>

Use Case: Flight Agent

<<Inquires about flight availability >>

Flight Agent

Flight Web Services

<<Presents the flights sorted by price >>

<<Books Flight >>

Travel Agent
Use Case: Hotel Agent

<<Inquires about Hotel availability >>

Travel Agent

<<Present the Hotel List >>

Hotel Agent

<<Book Hotel >>

Hotel Web Services

Use Case: Car Booking Agent

<<Inquires about Car Rental availability

Travel Agent

<<Present the Car Rental List >>

Car Rental Agent

<<Book Car Rental >>

Car Rental Web Services
Detailed Design

- Detail design (class diagram and message sequence chart) for each agent participant should follow.
- Modularization of the internal architecture will help and save a lot of time!

Travel Agency System (TAS)

Data Specification
TAS: E-R Diagram

ER Travel Agency System

Typical Data Definition

Flight:
- The information of a flight booked by a user is stored in the Flight table. This table is also accessed by the TAS to get the user’s flight history.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flight_ID</td>
<td>Unique ID</td>
<td>Long</td>
</tr>
<tr>
<td>Flight_No</td>
<td>Flight number</td>
<td>Long</td>
</tr>
<tr>
<td>Departure_Calendar_ID</td>
<td>Departure date and time from origin</td>
<td>Calendar</td>
</tr>
<tr>
<td>Arrival_Calendar_ID</td>
<td>Arrival date and time to destination</td>
<td>Calendar</td>
</tr>
<tr>
<td>Departure_Location</td>
<td>Departure location</td>
<td>Varchar (50)</td>
</tr>
<tr>
<td>Arrival_Location</td>
<td>Arrival location</td>
<td>Varchar (50)</td>
</tr>
<tr>
<td>Trip_Type</td>
<td>1=One way / 2= Round trip</td>
<td>Integer</td>
</tr>
<tr>
<td>Class_Type</td>
<td>1=First class / 2=Business/ 3=Economy</td>
<td>Integer</td>
</tr>
<tr>
<td>Airline</td>
<td>Airline company name</td>
<td>Varchar (50)</td>
</tr>
<tr>
<td>Rate</td>
<td>Flight fee</td>
<td>Float</td>
</tr>
</tbody>
</table>
Inter-Agents Messages

- As discussed in the Design document, SOAP will be used as a protocol of communication between agents and between agents and Web services.
- The input and output parameters of each function introduced below has an XML format.
- These XML documents map the data structure defined in the Data Dictionary document.
Get Flight

- Input parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;flight&gt;</td>
<td>User’s Flight Related Information</td>
</tr>
<tr>
<td>&lt;departureLocation&gt;</td>
<td>String</td>
</tr>
<tr>
<td>&lt;departureDate&gt;</td>
<td>String</td>
</tr>
<tr>
<td>&lt;arrivalLocation&gt;</td>
<td>String</td>
</tr>
<tr>
<td>&lt;arrivalDate&gt;</td>
<td>String</td>
</tr>
<tr>
<td>&lt;tripType&gt;</td>
<td>String</td>
</tr>
<tr>
<td>&lt;classType&gt;</td>
<td>String</td>
</tr>
</tbody>
</table>

- Output parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;flights&gt;</td>
<td>List of all available flights</td>
</tr>
<tr>
<td>&lt;flight&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;flightID&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;flightNo&gt;</td>
<td>String</td>
</tr>
<tr>
<td>&lt;departureLocation&gt;</td>
<td>String</td>
</tr>
<tr>
<td>&lt;departureDate&gt;</td>
<td>Calendar</td>
</tr>
<tr>
<td>&lt;arrivalLocation&gt;</td>
<td>String</td>
</tr>
<tr>
<td>&lt;arrivalDate&gt;</td>
<td>Calendar</td>
</tr>
<tr>
<td>&lt;tripType&gt;</td>
<td>String</td>
</tr>
<tr>
<td>&lt;classType&gt;</td>
<td>String</td>
</tr>
<tr>
<td>&lt;airline&gt;</td>
<td>String</td>
</tr>
<tr>
<td>&lt;rate&gt;</td>
<td>Float</td>
</tr>
</tbody>
</table>

...
Inter-Agents Messages

- Similar descriptions for
  - Book Flight; Get Hotel Room; Book Hotel Room; Get Car Rental; Book Car Rental; and Send Memo

should be generated.

Documents Required

1. System overall specification
2. Design documents for the agency (system architecture, explain why it is necessary to use each agent in your design)
3. Design documents for each participating agent, their basic functions and/or services
4. Data and/or knowledge sharing specification
5. Communication specification (message format, conforming to the communication protocol)
6. Journal of important design decisions
Design Journal

- We need to keep track of the design decisions that the developers make; e.g., why a role was discarded? why several roles were assigned to a single agent? why a use-case was split into two? why a communication protocol was chosen? etc.

- Create a separate document and keep the record the followings:
  - What was the input before refinement?
  - What was the decision made by the designer?
  - What was the outcome of refinement (output)?
  - Why such decision was made?

Questions?