Chapter 2.28

A Framework for Ontology-Based Tourism Application Generator

Roopa Jakkilinki
Victoria University, Australia

Nalin Sharda
Victoria University, Australia

ABSTRACT

This chapter provides an overview of tourism ontology and how it can be used for developing e-tourism applications. The Semantic Web is the next generation Web; it uses background knowledge captured as an ontology and stored in machine-processable and interpretable form. Ontologies form the core of the Semantic Web and can be used to develop intelligent applications. However, generating applications based on ontology still remains a challenging task. This chapter presents a framework that provides a systematic process for developing intelligent e-tourism applications by using a tourism ontology.

INTRODUCTION

Tourism is one of the most successful and dynamic industries in the world, and it is constantly evolving because of technological advancements. Information technology is being used to enhance tourism services such as travel bookings, itinerary planning, destination marketing, and information sharing. These services use dynamic Web applications.

The current tourism applications rely on static information sources such as Web sites to create tourism products and services. These applications lack intelligence; for example, an itinerary planner in the current scenario will allow the tourist to make bookings, but it cannot suggest an itinerary based on the travellers preferences. A Semantic Web application using an ontology, generic profiling, and semi-structured query tools can overcome the technical limitations of the current systems, and help build intelligent e-tourism tools, or applications.

This chapter discusses the purpose of developing a tourism ontology and proposes a model to develop intelligent tourism applications based on the same. The second section presents the background knowledge, followed by a proposed model for developing e-tourism applications, the following section demonstrates the working of an itinerary planner, and we finish with the conclusions.

The main objective of this chapter is to present a framework for developing ontology based...
e-tourism applications. The specific foci of the chapter are:

- To provide an understanding of the Semantic Web and ontologies
- To introduce various existing travel ontologies and applications based on the same
- To describe a process model for developing e-tourism applications
- To present a case study using an intelligent itinerary planner

**BACKGROUND**

**Semantic Web**

The Semantic Web was thought up by Tim Berners-Lee as a mesh of information linked up in such a way so as to be easily processable by machines. It is not intended to be read by people, as it describes relationships between data that software will interpret (Palmer, 2001). Figure 1 represents the Semantic Web stack which has a layered architecture, it is based on a hierarchy of languages, each language both exploiting the features, and extending the capabilities of the layers below (Butler, 2003). A brief introduction to the Semantic Web layers is presented in the following:

- **Uniform resource identifier (URI):** The Web naming and addressing convention, like the strings starting with “http” or “ftp”; they are short strings used to identify resources on the Web. Anyone can create new URIs. Example: http://melba.vu.edu.au/roopa.txt.
- **Unicode:** A replacement for the older ASCII code and can cope with multiple languages. It is a 16-bit code that can be used to represent the characters in most of the world’s scripts.
- **Extensible Markup Language (XML):** A standard format for serializing data using tags; XML file can contain data like a database, it is derived from Standard Generalized Markup Language (SGML) and is somewhat similar to Hypertext Markup Language (HTML). XML schema is a schema language used for describing XML data as well-defined schemas or data models. XML namespaces (NS) is an extension to XML for managing a collection of names identified by URIs.

*Figure 1. The Semantic Web stack, and its layers covered in this chapter*
• **Resource description framework (RDF):**
  This allows users to add metadata to describe the core data; RDF Schema is a language for describing RDF vocabularies (Bray, n.d.); in other words, RDF schema provides a way of organizing a large set of RDF vocabulary.

• **Ontology vocabulary:** A data model that represents the terminology used in a domain; it also is used to reason about the objects in that domain and the relations between them. Web Ontology Language (OWL) and Darpa Agent Markup Language + Ontology Interchange Language (DAML+OIL) are some of the languages used to describe ontologies.

• **Logic:** The Logic layer allows carrying out reasoning on a set of data, based on predefined rules, in order to draw conclusions. Inference engines or reasoners (Inference Engine, n.d.), such as, Racer, Fact, and Pellet work at this layer.

• **Proof and trust:** The proof and trust layers are still nascent. In most applications construction of proof is done by using some rules, the other party can use these rules to see whether or not a statement is true. Trust layer allows the creation of digital signatures for authentication and encryption.

### What are Ontologies?

An ontology is a data model that represents a domain; it can be used to reason about the objects in that domain and the relations between them. Ontologies represent knowledge about the world or some part of it, they consist of: classes, collection of objects; attributes, properties an object can have and share; relations, represent the way the objects are related; and individuals, which are instances of the class (Chandrasekaran, Josephson, & Benjamins, 1999).

An ontology can be a domain ontology and theory ontology (Swarzout, Patil, Knight, & Russ, 1997). A domain ontology models a specific domain; it represents the particular meanings of terms as they apply to that domain, for example, tourism. A theory ontology provides a set of concepts for representing some aspect of the world, such as time and space.

### Need for Tourism Ontology

Ontologies are especially useful where multiple entities such as researchers and organisations are active in the same domain, but each entity uses their own data model for that domain. For example, in the tourism domain, different entities such as travel agents, hotel chains, national, and regional tourism organisations have their own way of representing their services to the consumer (accommodation, events, attractions, services, etc.) using different data models. Furthermore, these data models maybe represented using different software technologies. This leads to interoperability problems, that is, software developed for one system cannot access data on another.

If tourism entities need to communicate with one another, a common data representation is needed. This common representation needs to represent both the concepts in the domain, and the relationships between these concepts. In addition, it should be possible for each tourism entity to map its data models to that used in a common ontology.

Having an ontology is very useful in this situation; as it models the domain in a structured manner, all entities will be able to use the ontology to communicate with all other entities, by mapping their source data model to the common ontology and then using the existing mappings between the ontology and the other destination data models (Clissmann & Höpken, n.d.).

Another benefit of ontologies is that they make it possible to carry out reasoning on the domain, they also act as back ends for intelligent applications, that is, they provide the ability to derive domain knowledge for developing intelligent applications. These applications, with the help
of a reasoner, can infer facts from the domain ontology. Creating an ontology for tourism will allow knowledge sharing between different tourism organizations, and also will allow for the creation of intelligent e-tourism tools such as search engines and tour planners.

Travel Ontologies

A variety of tourism ontologies have been developed to date. In this section we give an overview of a number of tourism related ontologies. The Harmonise ontology is not only a minimum standard ontology, but also a means of reconciling various ontologies.

OTA Specification. The Open Travel Alliance specifications have been designed to serve two purposes, namely to act as a common language for travel related services, and to provide a mechanism for information exchange between travel industry members (The Open Travel Alliance, n.d.). It is possible to view the OTA specifications as a comprehensive ontology, defining concepts such as AirSchedule, GolfCourseReservation, HotelContentDescription, HotelPreferences, and so on. The OTA specification has already been utilised in a travel related project called Agentcities (Gordon, Kowalski, Paprzycki, Pelech, Szymczak, & Wasowicz, 2005).

MONDECA. MONDECA’s tourism ontology defines tourism concepts based on the World Tourism Organization (WTO) thesaurus (MONDECA, n.d.). These include among others, terms for tourism object profiling, tourism and cultural objects, tourism packages, and tourism multimedia content. MONDECA has created a proprietary system called the Intelligent Topic Manager (ITM) that is used to manage its travel ontology.

TAGA Ontology. The Travel Agent Game in Agentcities (TAGA) is an agent framework for simulating the global travel market on the Web. Its purpose is to demonstrate the capabilities of Agentcities (Agentcities, n.d.) and Semantic Web technologies. TAGA works on the Foundation for Intelligent Physical Agents (FIPA) compliant platforms within the Agentcities environment (The Foundation for Intelligent physical agents, n.d.). In addition to the FIPA content language ontology, TAGA defines two domain ontologies to be used in simulations. The first ontology covers basic travel concepts such as itineraries, customers, travel services, and service reservations. The second ontology is devoted to auctions and defines different type of auctions, roles the participants play in them, and the protocols used. TAGA ontologies are limited in their usability, and are rather unrealistic due to the nature of the TAGA simulations.

Harmonize Ontology. Harmonize is an attempt at ontology-mediated integration of tourism systems following different standards (E-Tourism, n.d.). Its goal is to allow organizations to exchange information without changing their data models. The Harmonize project also involves sub-domains that are partially related to the world of travel: geographical and geo-spatial concepts, means of transportation, political, temporal, and gastronomy, and so forth. These sub-domain concepts can be used within the travel system.

Numerous ontologies have been developed for the domain of tourism. Defining and agreeing on the right ontology is a difficult task. One could argue that the choice of the right ontology is purely subjective, because the meaning of various terms differs across domains, users, and situations. Each of these ontologies have been developed with a specific task in mind and specialises in a particular aspect of the tourism domain; for example, a tourism ontology that specialises in tourism events can be used to develop an event planner. Most of the ontologies have been developed with a tool in mind, and their scope is limited to that tool. We have developed an abstract ontology called the Australian sustainable tourism ontology (AuSTO) which covers all the general concepts used in tourism, both from the customer perspective and from the enterprise perspective;
subsequently several intelligent tools such as a tour planner, search engines and travel recommender systems are planned for development based on the AuSTO ontology.

Applications Based on Travel Ontologies

A number of intelligent applications such as search engines, tour planners, and location-based tour guides have been developed using ontologies. These applications help the traveller as well as tour operator to plan trips and find information about destinations. In this section we describe two such applications and their usage.

On Tour. On Tour can be considered as an intelligent search engine. The main objective of the On Tour system is to connect isolated pieces of information, that is, to assist the user in finding information from a variety of sources, and to allow individualized use of the same (Daniel, 2005). As a search engine, On Tour allows for the querying of distributed data as well as considering the semantics of discovered concepts and instances. It allows the user to specify preferences like maximum budget and minimum comfort, and define further constraints such as personal schedule. This system helps the user to plan a vacation from the beginning to the end. In later phases of development On Tour will act as a recommender system by giving advice on best restaurants, venue for musicals, and so on. It also will provide support for mobile devices. On Tour approach is to extract pieces of information from structured Web pages and conduct constraint based reasoning for the integration of multiple information sources.

Talea. Talea is a platform aimed at supporting the development of Web-based tourism applications (Levi, Vagliengo, & Goy, 2005). This software was designed and developed within the Diadi 2000 (Dissemination of Innovation in Industrial Decline areas) project. The Diadi 2000 project aims at applying ICT technologies to small and medium enterprises (SMEs) to increase the value of their businesses. Talea provides for multi device access, where customers and suppliers can use PDAs or smart phones to buy and offer tourism services. This software acts as a matchmaker by matching service provision with request; tourism suppliers can advertise their services such as room availability, car rentals, and so on, and customers can perform a search for a particular service.

Dynamic Packaging System. An important type of e-tourism application that has evolved in recent years is a dynamic packaging system (Cardaso, 2005). It is used by airlines, hotels, tour operators, and travel agencies to create customised packages for individual consumers. Dynamic packaging can be defined as the combining of different travel components, bundled and priced in real-time, in response to the request from a consumer or a booking agent. They have created an e-tourism ontology that allows interoperability through the use of shared vocabulary and meanings for terms. Semantic mediators are used to support a virtual view that integrates semantically annotated e-tourism information sources. Final dynamic package processes are created using conditional planning ranking and selection. Once the dynamic package processes are evaluated they are presented to the tourist and the tourist can select the package that he finds most appealing or suitable according to his preferences.

TOURISM APPLICATION GENERATOR ARCHITECTURE

Figure 2 presents the underlying model for generating ontology based e-tourism applications; called the e-tourism application generator architecture (e-TAGA). The e-TAGA model consists of
three layers: the ontology layer (OL), the business logic layer (BLL) and the graphical user interface layer (GUIL). The OL provides persistence for the tourism ontology; the business logic layer includes two common components and parts of the tourism applications themselves. The two common components in the BL are the inference engine (IE) and the custom logic (CL). The GUI layer includes the graphical user interface (GUI) components of individual tourism applications and some common GUI elements.

**Ontology Layer (OL)**

The ontology layer consists of the ontology which embodies knowledge about the domain and some additional specifications. This layer is the core of all Semantic Web systems. We will use the AuSTO ontology to exemplify the operation of the various layers.

**Tools for Ontology Development**

In any ontology development project one needs to begin by selecting an ontology development tool. In the AuSTO project three different tools were compared in order to decide which one of these would be most suitable for our tourism ontology development. The three tools considered were Protégé 2000, Ontolingua, and OntoEdit free. This comparison was based on ontological aspects and usability aspects. Our study indicated that Protégé 2000 is far superior in usability and ontology aspects as compared to Ontolingua and OntoEdit free (Jakkilinki, Sharda, & Georgievski, 2005), hence Protégé2000 was selected. Protégé2000 (Protégé, n.d.) comprises an open architecture that allows programmers to insert arbitrary components into the tool. This feature can be exploited during the development of Semantic Web applications based on the ontology. An additional benefit of Protégé 2000 toolkit is
developers can package the implementation of the application as a Protégé plug-in and test how the system behaves in response to any changes in the ontology.

Methodology Followed to Develop AuSTO Ontology

Ontology development methodology includes tools, techniques and process followed in order to develop the ontology. The methodology followed to develop the AuSTO ontology is as follows (Jakkilinki et al., 2005).

1. **Identify the purpose behind ontology development**: The pertinent questions are listed here, along with their answer for AuSTO development.
   - Why is the ontology being built? In the case of AuSTO, the ontology is being built to describe the tourism domain.
   - What is its intended use? AuSTO ontology will be used as a knowledge base to develop intelligent tools such as an itinerary planner.
   - Who are the users? AuSTO will be used by operators in tourism domain, such as the tourist operators, tourism vendors.

2. **Ontology capture mechanism** consists of three different stages:
   - **Determining the scope of the ontology**: This involves identifying all the key concepts and relationships in the domain.
   - **Selecting a method to develop the ontology**: The method we followed to develop AuSTO is the top-down approach.
   - **Defining the concepts in the ontology**: This involves taking closely related terms and grouping them as classes.

3. **Coding the ontology**: Coding refers to representing the ontology in some formal language. A suitable ontology editor has to be selected, in the case of AuSTO the ontology editor used is Protégé. Once the ontology editor is selected the classes have to be entered as concepts and their attributes are entered as slots.

4. **Refinement**: This consists of two phases, namely intra-coding refinement and extra-coding refinement. Intra-coding refinement refers to the refinement done during the coding phase, whereas extra-coding refinement refers to the changes made to overcome the errors uncovered during the testing and maintenance stages.

5. **Testing**: The testing process uncovers any defects in functional logic and implementation. Testing should be carried out during all stages of development.

6. **Maintenance**: This can be corrective, adaptive or perfective. Corrective maintenance involves correcting the ontology to overcome the errors discovered by users while querying the ontology. Adaptive maintenance involves modifying the ontology to fulfil new requirements. Perfective maintenance involves improving the ontology by further refining it, in order to enhance its functionality (Pressman, 1997).

**Brief Description of Classes in AuSTO**

Creating an ontology involves delineating concepts into a class hierarchy. Three important approaches to develop class hierarchies are top-down, bottom-up and a combination approach (Uschold & Gruninger, 1996). In the top-down approach the development process starts with the definition of the most general concepts in the domain, followed by specialised concepts. In the bottom-up approach the development process starts with the definition of the most specific classes, which form the leaves of the class hierarchy tree, with subsequent grouping of these classes into more general concepts. The combined approach
A Framework for Ontology-Based Tourism Application Generator

Figure 3. A screen shot of the AuSTO ontology

uses a combination of top-down and bottom-up processes. The approach followed for AuSTO is the bottom-up approach. This approach is usually driven by the need for having a workable vocabulary quickly and then enhancing it as the project progresses. AuSTO is written in OWL (Web Ontology Language). Figure 3 is a screen shot of AuSTO ontology in Protégé. The AuSTO ontology consists of a class hierarchy shown on the left, each class has properties and one can create individuals, or instances of a class, using the instance tab in the Protégé interface.

AuSTO being tourism ontology it contains classes from the tourism domain. Following list gives some of the important classes in AuSTO:

- **Involved party** can be traveller, vendor, operator, and so forth.
- **Requirement** refers to travel requirements.
- **Offering** includes travel products and services.
- **Solution** refers to systems outputs such as itineraries.
- **Resource** can be reserved or rented items.
- **Specification** allows for both offering specifications and requirement specifications
- **Preference** includes traveller’s preferences such as date, time, location, or price range.

Each of these classes can represent a plethora of tourism information. For example, requirement represents diverse travel requirements such as accommodation, entertainment, transport, and offering represents the wide range of travel products and services that vendors make available to the traveller, often as part of a packaged solution.

Business Layer

The business logic layer (BLL) uses an inference engine and custom logic, and is responsible for generating outcomes; that is, it returns results based on user interactions. Figure 4 describes the BLL for the applications based on AuSTO. A user specifies his travel requirements in the ontology and the tourism vendor advertises his offerings which are tourism services in the ontology. The offerings and requirements are loaded into the ontology model of the Jena subsystem. Pellet reasoner matches the travel requirements to the vendor offerings and sends it to the travel application manager. Travel tools such as tour
planners can query the travel application manager and produce the travel solution, which in this case is an itinerary.

**What are Reasoners?**

The reasoner is a software that applies logic to the knowledge embodied in the ontology for arriving at some conclusions (*Inference Engine*, n.d.). We generally recognize two types of inferencing, namely forward chaining, and backward chaining. In forward chaining one proceeds from a given situation towards a desired goal by adding new assertions along the way; whereas, in backward chaining one starts with the desired goal and then attempts to find the method for arriving at the goal. A number of reasoners are available, such as Racer, FaCT, Pellet, and F-OWL. Pellet is the reasoner used in the AuSTO project, it is an open-source Java-based OWL DL reasoner developed by the Maryland Information and Network Dynamics Lab Semantic Web Agents Project (Mindswap, n.d.).

**Role Played by Jena**

Jena is a Java framework for building Semantic Web applications, it is open source and has been
A Framework for Ontology-Based Tourism Application Generator

developed by HP (Hewlett Packard) Labs. Jena acts as a middleware which connects the ontology, reasoner and the user interface. In Jena all operations are done by manipulating the Jena model. Therefore, to manipulate an ontology it needs to be loaded into the Jena model first. Jena has four subsystems: query engine, database interface, reasoning engine, and ontology management. Jena’s architecture allows external reasoners to be plugged into the Jena models.

User Interface Layer

One of the most difficult aspects of building an application is designing the user interface. User Interface design is the design of the graphical elements on the computer screen with which a user interacts to conduct application tasks. The user interface is as important as the functionality of the application, and plays an important role in the success of any product. User interfaces accomplish two fundamental tasks: communicating information from the computer to the user and communicating information from the user to the computer.

The benefits of a good user interface design include: lower training costs, less user stress, consistency in application usage, increased ability to recover from errors, better user control, less clicks to find information, ability to store more information per screen, easier to use the software, selection amongst many choices using limited space, see all selections at all times, better understanding of the software, save screen space, and higher data entry speed (Miller, n.d.).

Task analysis and modelling techniques are increasingly being used in designing user interfaces, they form an important part of user interface design process and help design more intuitive interfaces.

What are Task Analysis and Task Modelling?

Task analysis involves the study of a system functionality as a collection of tasks. Generally the systems function is divided into a set of top-level tasks, and each one of these is further divided into sub-tasks, and so forth to develop a task-tree. This process can be used to guide the design of new systems beginning with user requirement capture. One of the most important applications of task analysis is designing user interfaces, in which menus are based on the task trees. The top level menus can be labelled after the top level tasks and the sub menus after the next level tasks (Dix, Finlay, Abowd, & Beale, 1998).

After an informal task analysis where the main tasks and their attributes have been identified, task modelling is used to understand the relationships among the various tasks in order to better address the design of interactive applications.

As task modelling is used to model the behaviour of a system from user’s perspective, it captures the system requirements and actions defined as a set of tasks, and models the behaviour of the system as a scenario of tasks. This allows the designers to improve the human computer interaction aspects when designing a system’s operation (Georgievski & Sharda, 2003). Although task models have long been considered in human-computer interaction, only recently have user interface developers and designers realized their importance to obtain more effective and consistent solutions (Giulio, Paterno, & Santaro, 2002). Task models play an important role because they represent the logical activities that should support users in reaching their goals, and knowing the tasks necessary to attain a goal is fundamental to any good design (Paterno, 2002).

There are two types of task models: user task model and system task model. A user task model
states the problems to be solved by the system, and thus consists of overlapping user scenarios (Georgievski & Sharda, 2003). Actors involved in a user task model are generally human; however, it may include external systems and the environment. A system task model forms the basis for specifying a solution in the form of system requirements. Actors involved in a system task model are generally subsystems, interfaces, and, at times humans.

Tools for Task Modelling

One of the main problems in task modelling is that it is a time-consuming and sometimes tedious process. To overcome this problem interest has been increasing in tools that support task analysis and modelling. However, current tools are outcomes of research projects, and are used mainly by groups that have developed them.

The concur task tree environment (CTTE) is a Java Applet based tool developed by Human Computer Interaction Group – ISTI (Pisa). CTTE provides the ability to build task models from a visual perspective where the user can define and structure the tasks in a logical fashion using the graphical editor provided in the tool. CTTE enables the user to focus on the activities of their model and thus allowing the user to identify the requirements of the model and organize them into a logical hierarchy of task and subtasks (Georgievski & Sharda, 2003).

The main features of the CTTE tool are (Giulio et al., 2002):

- **Focuses on activities**: Allows designers to concentrate on the activities that a user has to perform, rather than programming details.
- **Hierarchical structure**: Provides a wide range of granularity allowing large and small task tree structures to be developed and reused.
- **Graphical syntax**: Facilitates easy interpretation of the logical task structure using graphical representation.
- **Concurrent notation**: Provides rich set of possible temporal relationships that can be used to specify the relationship between the tasks.
- **Distinct task representations**: Uses distinct icons to represent user task, application task, interaction task, and abstract task.

CTTE provides the ability to build two types of task models: single user task models and cooperative task models. Single user task models are used to represent systems that a single user controls. A cooperative task model is similar to a single user task model; however it includes tasks executed by two or more users.

Other useful features of CTTE tool are model comparison, reachability analysis, and interactive task model simulator (Giulio et al., 2002).

**Itinerary Planner Case Study**

In this section we describe the task model created to represent the user interface for the itinerary planner. This task model guide the development of the user interface by focusing on the various functions the user interface needs to perform.

CTTE allows the following types of tasks to build the entire task model:

- **Abstract tasks** define a set of subtasks to be performed at a conceptual level
- **User tasks** denoted the operation/tasks executed by the user
- **Interaction tasks** represent tasks that carry out communication between entities within the task model.
- **Application tasks** are of tasks executed by the system or application entities in the process model.
A Framework for Ontology-Based Tourism Application Generator

Table 1. Temporal operators used in CTTE

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1 [] T2</td>
<td>Choice</td>
<td>A choice between two or more tasks</td>
</tr>
<tr>
<td>T1&gt;&gt;T2</td>
<td>Enabling</td>
<td>T1 enables T2 when T1 is terminated</td>
</tr>
<tr>
<td>T1[]&gt;&gt;T2</td>
<td>Enabling with Information Exchange</td>
<td>T1 provides some information to T2 besides enabling it</td>
</tr>
<tr>
<td>T1</td>
<td>&gt; T2</td>
<td>Suspend/Resume</td>
</tr>
<tr>
<td>T*</td>
<td>Iteration</td>
<td>Tasks performed repetitively</td>
</tr>
</tbody>
</table>

CTTE uses transition notations to describe the temporal relationships between tasks and the execution sequence for the task model. The temporal operators used in CTTE are described in Table 1.

We have implemented the task model for itinerary planner as a single user task model that represents the overall function of the user interface from a user perspective. We represent these tasks as tree diagrams in Figures 5 to 11. In the task tree diagram we define the execution sequence for each task using the temporal operators described in Table 1.

**Figure 5. Itinerary planner abstract model**

CTTE uses transition notations to describe the temporal relationships between tasks and the execution sequence for the task model. The temporal operators used in CTTE are described in Table 1.

We have implemented the task model for itinerary planner as a single user task model that represents the overall function of the user interface from a user perspective. We represent these tasks as tree diagrams in Figures 5 to 11. In the task tree diagram we define the execution sequence for each task using the temporal operators described in Table 1.

**Figure 5:** This shows the itinerary planner abstract task model. It illustrates the tasks the user can perform on connecting to the itinerary planner Web site. Task 5 consists of connecting to the Web site and viewing the home page. Task 6 involves logging into the Web site and then carrying out the data maintenance activities. Figures 8 to 13 expand on these activities.

**Figure 6:** This shows the task model that expands the Login Activity task. Login Activities describe the tasks to be performed for the user to login, it allows for an existing user, as well as new user.

**Figure 7:** Data Maintenance Activities task tree, which provides a choice between four abstract tasks: Preference Maintenance, Requirement Maintenance, Itinerary Maintenance, and Offering Maintenance.

**Figure 8:** Preference Maintenance Activities task tree, which consists of the tasks involved in maintaining the preferences of the traveler. Here preferences refer to what of the traveler likes, with regard to various facilities such as accommodation, transport, and so forth.
A Framework for Ontology-Based Tourism Application Generator

Figure 6. Login activities

![Login Activities Diagram]

Figure 7. Data maintenance activities

![Data Maintenance Activities Diagram]

Figure 8. Preference maintenance activities

![Preference Maintenance Activities Diagram]
A Framework for Ontology-Based Tourism Application Generator

Figure 9. Requirement maintenance activities

Figure 10. Itinerary maintenance activities

Figure 11. Offering maintenance activities
Figure 9: Requirement Maintenance Activities task tree, which describes the tasks involved in maintaining the traveler requirements. Requirements refer to the travelers demands for the tour, for example the traveler may want 5-star accommodation and a business class flight.

Figure 10: Itinerary Maintenance Activities task tree, which describes the tasks involved in creating an itinerary for the traveler. An itinerary is generated by matching the traveler’s requirements with the offerings available.

Figure 11: Offering Maintenance Activities task tree, which describes the tasks involved in maintaining the offerings being provided by various travel vendors. Tourism vendors can advertise their services such as room availability, tickets availability through this option.

Working of an Itinerary Planner

In this section we describe the operation of a travel itinerary planner; this itinerary planner has been developed based on the application generator framework. It consists of AuSTO ontology, a user interface created using ASP.net, and a business logic layer, which acts as a connector between the ontology and the user interface. The AuSTO ontology is populated by a tourism domain expert, the application allows tourism operators to advertise their offerings in the offerings page, and the offerings are stored in the ontology. The end user or the tourist can specify his requirements in the requirements page; these requirements are stored in the ontology, and the tourist also can specify his preferences which also are be stored in the ontology. The itinerary planner matches the requirements with the offerings, and produces an itinerary; the end user can either reject the itinerary offerings or accept these and confirm bookings in the itinerary page. The user interface for the itinerary planner has been developed based on the task model described in the previous section. This application is explained in some more detail with the help of screen shots in the following.

Figure 12: This shows the new user screen, which allows the creation of a new user. It is necessary to have an account in order to use the application.

Figure 13: This shows the user login screen where an existing user can login into the application. Once the user logs in, he has access to facilities such as storing preferences or specifying requirements.

Figure 14: This shows the requirements screen where a tourist can enter his requirements for a trip. Once the requirements are entered the tourist clicks the Add Trip button and a new leg can be added.

Figure 15: This shows the itinerary screen, the tourists’ requirements are matched with the vendor offerings and an itinerary is produced. The tourist can accept offerings in the itinerary with the help of the checkboxes, and make booking by clicking on the Make Bookings button and then confirm bookings.

Figure 16: This shows the preferences screen, which allows the user to store his preferences for accommodation, transport facilities and other such services. Different preferences for different kind of trips can be stored, such as family trip and business trip.

Figure 17: This shows the offerings maintenance screen, tourism vendors such as hotels or transport providers can advertise their offerings on this page. For example, hotels can advertise their room availability, and transport providers can advertise their vehicle availability.

CONCLUSION

There is a need for standardisation of definitions and concepts in the field of tourism; the solution is to develop travel ontologies. Number of travel ontologies have been developed in recent times, each with an application in mind; we have developed an Australian sustainable tourism ontology (AuSTO), specifically for the Australian
A Framework for Ontology-Based Tourism Application Generator

Figure 12. New user screen

Figure 13. Login screen

Figure 14. Requirements screen

Figure 15. Itinerary screen
tourism sector. This AuSTO ontology reuses the knowledge from some of the existing ontologies. Ontologies enable the development of Semantic Web applications, but ontology driven application development is still a nascent field. We are developing an intelligent travel application generator based on the AuSTO ontology. The application generator framework enables the production of different intelligent travel tools such as itinerary planners, and recommender systems. We are building an itinerary planner by using this framework, and this intelligent itinerary planner can match the user requirements specified in the ontology with vendor offerings specified in the ontology and produce an itinerary as a solution. This chapter presented an overview of the Semantic Web, introduced different tourism ontologies and some applications based on tourism ontologies, and describes in detail a framework for developing e-tourism applications based on ontologies.

ACKNOWLEDGMENT

We would like to thank Sustainable Tourism Cooperative Research Center (STCRC), Australia, for the funding this research. We also would like to thank Henk Meijerink and Paul Mohinyan for their help in this research.

REFERENCES


This work was previously published in Information and Communication Technologies in Support of the Tourism Industry, edited by W. Pease; M. Rowe, and M. Cooper, pp. 26-49, copyright 2007 by IGI Publishing, formerly known as Idea Group Publishing (an imprint of IGI Global).