A Multi-Agent Based Solution for Dynamic Tour Recommendation

M.K.M Kalpanie

129104M

Dissertation submitted to the Faculty of Information Technology, University of Moratuwa, Sri Lanka for the partial fulfillment of the requirements of the Degree of MSc in Artificial Intelligence

February 2015

Declaration

I declare that this dissertation does not incorporate, without acknowledgment, any material previously submitted for a Degree or a Diploma in any University and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, to be made available for photocopying and for interlibrary loans, and for the title and summary to be made available to outside organization.

Name of Student

M.K Mahesha Kalpanie Signature of Student

Date:

Supervised by

Name of Supervisor

Prof. Asoka Karunananda Signature of Supervisor(s)

Date:

Acknowledgements

My research experience has been fruitful with the motivation and guidance from many people. Firstly, I would like to extend my heartiest gratitude to my supervisor, Prof. Asoka Karunananda for his invaluable insights and continuous support throughout the Artificial Intelligence Master program and the research. Also my special thanks go to all the lecturers and the staff at University of Mortuwa for giving me their support in various manners for the study.

I would like to take this opportunity to thank the Chief Technical Officer of Araaya Business Solutions (Pvt.) Ltd. Mr. Ashoka Ekanayaka for guiding me and allowing me to do the research while working for his company. My immense gratitude also go to the staff of Tourist Board of Sri Lanka and the staff of Sky Management Systems (Pvt) Ltd. for sharing their knowledge and the ideas on tour planning and Tourism domain. Lastly, I would like to thank my mother and all my friends who have helped me a lot.

Abstract

Tour recommendation and planning has become a complex and challenging task in the modern world. Even though there is wide-variety of existing sources that provide tour recommendations for tourists based on their preferences, it has become a difficult task for tourists to pick up the right and up to date touristic information from those sources. The solutions provided by the existing application and travel companies are not sufficient for a tourist to plan their holidays in a satisfactory and efficient manner. Even when the existing touristic information sources are up to-date, there is no proper way to recommend best touristic activities based on specific preferences of a tourist using those up to date information. This thesis discuss about the software that has been developed to tackle the problem of handling the complexity of tour recommendation in dynamic manner based on up to date information. The main purpose of this application is to help tourists by providing them with up-to date information about touristic information and help them to choose the right destination and tour activities. In this application tour recommendation will be created based on users' dynamic changes on preferences and the dynamic changes of the other factors such as changes of the schedules of events or tour activities, bad weather condition, heavy traffic etc.

With the approval of an administrator, any user who registered in this system can input information about new tour locations or touristic activities to the systems or update changes of existing information to the system. A tourist can access the systems using their hand held devices and specify their preferences as the inputs. The preferences of the tourist mainly include arrival date, departure date and the interests of the tourist. A tourist can have one or more interests such as wildlife, scenery, hiking, cultural discovery, beach & water related activities, meditation, romantic getaway, shopping and etc. based on up-to date information and tour preference the system will recommend using multi-agent technology, a customized set of touristic activities to the tourist in adaptable and efficient manner. According to the evaluation performed using real data captured via questionnaire, the best results were obtained and it is concluded that a tourist can easily calculate their own tour activities in dynamic manner.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Prolegomena

This chapter describes regarding the proposed solution for the problem of tour recommendation in dynamic manner and also what inspired to start this project. While discussing the existing similar solutions and their limitations, an attempt is made to provide a solution to the identified problem.

1.2 Background & Motivation

Tourism has become highly popular and competitive industry in the world. Also it is high stimulant for business, trade, investment and other economic activities. With the advances and the innovations in the information technology and mobile technologies, touristic information has become readily available. [1] The travel and tourism industry is one of the most important and dynamic sectors in Business-to-consumer (B2C) e-Commerce. In this context, recommender applications can be valuable tools supporting, for example, information search, decision making, and package assembly. Considering lifestyle of the population, for the majority of the population, time factor is one of the most main reasons to restrain the practice of different activities. Not only for the time that is necessary to accomplish the activity but also for the time spent on planning and organizing it. [2] Motivation is also an important variable on the consumer and it is stimulated by a complex mix of economic, social, psychological, cultural and political influences related to industry and the wider environment in which the tourists are involved. [3] Motivation is a state of necessity, a condition that puts pressure on the individuals to certain types of activities that bring satisfaction. Thus, only realizing the true motivations of tourists we can understand what they really want, in order to present the best solutions that even they would never have taken into account. Recommendation System will be an extremely useful tool being able to help tourist in decisions, as well as in activities planning.

The Recommendation System has information about the tourists and its possibilities to analyze it in order to understand the best way to satisfy the interests of tourists. Time and the motivation factors are the most important factors which should be analyzed in tour recommendation.

Many studies have introduced a variety of tourist information systems which provide travel information and recommended tourist attractions according to profiles and the preferences of a tourists. In addition, there are many studies on the ontological approach in modeling the tourist information in order to make the tour planning in intelligent manner. Ontologies are used in knowledge-based systems as conceptual frameworks for providing, accessing, and structuring information in a comprehensive manner [4]. The locations recommended by most of the recommendation systems are not providing up-to date information. The time necessary to visit a number of places at a destination can be more than a traveler's available time. Automated trip planners and recommendation systems should consider the preferences and requirements of tourists for activities, accommodations, route planning and etc. Currently, travel agents and travelers must visit multiple independent web-sites for various information such as accommodation and activity facilities to plan a trip tailored to given preferences. When a tour plan is created in order to satisfy the interests of a tourist, by travel agents companies or by using most of existing tour applications, a huge manual task involved in searching in different activities in different domains. This is hard and time consuming task. The lack of standards in the tourism domain brings up the necessity of integration of heterogeneous information sources. The main focus of this thesis, the tour recommendation, according to user preferences and constraints, aids in the selection and scheduling of various aspects of a tour such as events, attractions, and routes. This system can also function independently as a location centric recommender or day planner of the aforementioned tourist entities.

1.3 Aim and Objectives

The aim of this research is to design, develop and evaluate a system, addressing the complexity of tour recommendation in a dynamic manner. This is a mult-agent based solution which can adapt in a dynamic environment based on changes of user preferences and the touristic information. The system is evaluated against its objectives using the results obtained by a survey conducted using a questionnaire. As touristic information is highly dynamic, those changes happening in the touristic information sources should be reflected in tour recommendations. In order to achieve this objective, an adaptable solution has to be designed. The multi-agent technology is a good methodology to implement an adaptable solution to this kind of problems. The complexity of the touristic attractions and events can be addressed using conceptual design of ontologies. The main objective the system is to provide an adaptable solution on the changes of the user preferences or the other changes in the environment. The other objectives of this system are achieved as follows.

i. To achieve the adaptability of the system, the multi-agent technology is used to cater the changes in user preferences and tour activities, in dynamic manner in tour recommendation calculation.
ii. To provide the optimum recommendations for the specified user preferences, the attributes associated with both tour activities and user preferences are analyzed and compared.
iii. To achieve relevancy of recommendations provided, only the tour activities which suits for the profile of particular user based on his preferences are calculated.
iv. The system is capable enough to process data and generate the tour recommendations in efficient manner.
v. Cost effective tour recommendations are provided by the system.
vi. Accurate information is provided in the computed tour activities and recommendations.

1.4 Proposed Solution

This research is focused on dynamic tour recommendation as the tour plan can adapt on the dynamic changes of the environment and the preferences of tourist at any time. This is a multi-agent based system which starts with the preference of a tourist, searches for up-to date information in dynamic manner, ignores irrelevant information, combines several data resources of the tourism sub-domains using agents, and generate a coherent results of touristic activities. This research will also consider psychological aspect of the interests. Tour agent companies won't usually consider those things when creating tour plans and recommendations for tourists. This is fully dynamic because as the main concern of this system is the nature of user interest that he has specified, it will dynamically inferred a profile for a particular user based on his interest and will calculate the recommendations for him by matching that profile with inferred tour activities for that profile. A tourist can access the systems using their hand held devices and specify their preferences as the inputs. The preferences of the tourist mainly include arrival date, departure date and the interests of the tourist. User can input information about new tour locations or touristic activities to the systems or update changes of existing information to the system. A tourist can have one or more interests such as wildlife, scenery, hiking, cultural discovery, beach and water related activities, meditation, romantic getaway, shopping and etc. based on up-to date information and tour preference the system will recommend using multi-agent technology, a customized set of touristic activities to the tourist in adaptable and efficient manner. The agents will coordinate and process the user requests to provide the customized tour plan for particular tourist based on the user preferences and the knowledge base which is associated with them and can be changed dynamically. Those tour plans suggested by the system will include recommended tourist locations, hotels, activities and all the other necessities of which a tourist can be interested in. Use the real data for evaluation using a questionnaire according to the evaluation the best results obtained and concluded that a tourist can easily calculate their own tour plan.

It is used to deploy the tour planner, Tomcat Servlet container and MySQL database by providing web access to anyone who use personal computers or hand held devices.

The tourists are preferable to use hand held devices during the tour in order to access their geographical locations using GPS facilities of their hand held device. Hand held devices with Android and GPS facilities will enable more features for the users. Having an internet connection is a must until this system is improved to manage the cache. The system will deal with cache of the touristic information during the course of the tour whenever user does not have an internet connection. In the development work, mainly Jade and Jena libraries are used with J2EE. As this tour planner is developed as a platform for all touristic parties, the Java Enterprise Edition will be used by providing scalability and security. Jade libraries are used to develop agent related implementations while Jena libraries are used for ontological manipulations of the touristic information. The mobile application which is provided for tourists is developed using Android which is the most popular mobile operating system.

1.5 Thesis Organization

A background study of recommendation systems in tourism, along with brief introductions to other basic concepts, is presented in Chapter 2. Chapter 3 presents an overview of technologies used in implementing the tour recommendation application. Then in Chapter 4 describes the multi-agent approach used in this application which is built on top of tourism ontology. In Chapter 5, the design of the modules and sub modules of this application are described. An overview of the implementation of the system is included in Chapter 6. Finally, in Chapter 7, the evaluation of the system is explained while Chapter 8 concludes the thesis with a summary of my contributions and a discussion of possible future work.

1.6 Summary

This chapter explained the overview, objectives and resource requirements of the tour planning system. Organization of the thesis is presented under the thesis organization section. The next chapter presents an overview of technologies used in implementing the tour recommendation application.

Chapter 2

Current Issues in Tour Recommendation

2.1 Introduction

This chapter describes the findings of the review performed on the past research effort in the context of tour recommendation based on multi-agent technology. Also the general concepts of modeling the complexity in the tourism domain using ontology concepts are also discussed. The purpose of this chapter is to identify the strengths and the weaknesses of existing tour recommendation applications.

2.2 Ontology Based Modeling in Tour Recommendation

In the recent past, researchers have explored a wide array of technologies in modeling the complexity in tour planning using ontologies. In addition, many ontologies have been introduced in tourist domain in order to provide various intelligent tourist information services. Ontology, in the present context, was originally suggested in 1992 by Tom Gruber who defined it as "a specification of a conceptualization" [5]. The word seems to generate a lot of controversy in discussions about Artificial Intelligence. It has a long history in philosophy, in which it refers to the subject of existence in the sense that it describes the concepts and relationships that can exist for an agent or a community of agents.

As a part of a running project SAMAP, Susana Fernandez and Laura Sebastia [6] introduced a solution on planning tourist visits adapted to user preferences. This is an attempt to overcome the difficulty in solving a real world problem which is having multiple goals with Artificial Intelligence planning. Through a preliminary experiment, they have demonstrated that solving the tour planning problem is hard because the time and space complexity is high when the number of goal increases.

There are many sources and systems that the static information is provided with the tour locations, activities, restaurants and hotels etc. However the quantity of that information is large as the all users are presented this information in the same way without considering the interests of the user. As those systems are not intelligent enough to provide only the relevant information to support users, the users should put a huge effort in tour planning. The most important feature of SAMAP project is that the user preferences have been taken into account when computing tour plan of a particular user, presenting only the information that the user interested in. This application computes a tour which contains the elements; a selection of the most interesting places for this user according to his model, indications about which transport he/she should take to move between different places, recommendations about where to have lunch or dinner (restaurants, bars, etc.), and proposals of places of leisure such as cinemas or theaters.

SAMAP application processes a tour plan under three steps. Under the first step it builds the user model which requires the user to introduce his personal details including his/her preferences and interest. The second step consists of the analysis of the user model in order to obtain a list of places it may like to visit according to its preferences. Each place in this list is associated with a number which indicates the utility that visiting this place might have for the user. This module uses both the user model and a repository which stores previous visits of other users. Therefore, the system can compare the previous visits to the same city of other users with a similar profile to the current user. The last step is the computation of the tourist plan by taking considering only the visits that may be interesting for the user (the output of the previous step). This planning task has several features that make it hard for current planners. According to this literature some of the main issues in the existing planners in the real world domain that they have been addressed are; multiple criteria satisfaction such as time and money, selecting the appropriate visits while maximizing the utility, incorporating the transport sub plans, solving partially instantiated goals.

With the recent advances in Internet, the use of intelligent tourist information services via the mobile systems has become very popular. Another good feature is this tour planning system can run in any device with Internet connection, such as PDA (Personal Digital Assistants) or third generation mobile phone. User Preferences can be gathered by using any device with Internet connection. Once the application has computed a plan, it must be showed to the user through its device.

They have developed ontology to store all the information introduced by the user and the output generated by each module. In order to obtain more interesting data about the user, the system has been used past information about the same user with the learning techniques, mainly classification techniques. However it doesn't consider the dynamic changes in the environment even though the user preferences have been taken into account.

Heum Park, Aesun Yoon and Hyuk-Chul Kwon [7] presented an intelligent tourist information service on top of a task ontology together with a task model based on the tasks of tourist. Most of the existing solutions provided on tourist information services were domain-oriented using domain ontology, not task-oriented, and they did not considered user needs and activities. A task-oriented menu provided with this approach, enables users to search for services based on "what they want to do" rather than by "name of category". Construction of such a task-oriented menu is based on a task ontology which supports the description of user activity such as task execution and the solving of problem encountered during the task. In this paper, a design for an Intelligent Tourist Information System using Task Ontology (ITISTO) based on various perspectives of tourist has been introduced.

They have considered generic tasks and task ontology based on the perspectives of tourist, and intelligent tourist information services using them. Therefore, they have proposed 1) a task model of travelers' perspective based on their needs and activities, 2) a task ontology using the generic tasks, their activities, relations, and properties, and 3) an intelligent tourist information system using task ontology based on various tasks and activities of travelers.

Most ontological studies based on tourism have focused on domain ontologies for tourist attractions, hotel services, tour agents, package tours, user profiles, etc. and they have constructed systems for tourist information, tour recommendation or tour agents by using those ontologies and various databases providing tourist content. However, in this literature, they have attempted to build a task ontology based on the tasks of tourists as there have only been a few task ontology studies found related to the aspects of travelers' tasks. [7] Task ontology describes the reasoning concepts and their relationships within given tasks for a specific domain, for example, diagnosis, monitoring, scheduling, and designing. The Task ontology describes a generic task or activity, such as the process of booking a package tour, including the flight and, perhaps, a rental car. They have introduced a task model based on travelers' perspectives using traveler's needs and activities, and a task ontology by using the task model for intelligent tourist information Services. Thus, the ITISTO can provide a task-oriented menu for intelligent tourist information services using the concepts, instances, properties and relations of task ontology, and those of domain ontology with semantic links, for various needs of tourists and activities. Those are then compared with domain-oriented menus by domain ontology based on system's perspective, not the perspectives of the users, and showed more intelligent than those of the existing tourist information services. Therefore, ITISTO is an intelligent tourist information service model centered on travelers' needs and adopting an ontological approach.

As per this literature, a domain ontology has been modeled which contain concepts and relations for tourist content. Before designing a task model, his needs and activities for various cases using his viewpoints both before and during trip has been investigated. Generally, travelers before a trip want the tourist information about the attraction nearby, the weather condition, travelling expenses, etc. During the trip, tourists might want to know real time information and they might want to do actions on site tour as follows: the most famous restaurant nearby, the price of a ticket to the event (a cultural show), move to the hotel, confirm the reservation, etc. With the above information, by obtaining their all possible specific actions, defined generic tasks and rules, and a task ontology has been constructed based on the perspectives of tourist for the tourist information service domain.

In this literature, the user perspectives of user needs and activities are similar to the user preferences in first paper which has been reviewed and discussed. In here the ontology is well constructed by providing efficient and intelligent tourist information services for various needs of users based on perspectives of them. Perhaps the complex mappings can be found in some cases depending on the tourist information services that are integrated to the system. In addition to the above, the various kinds of data stores (task ontology, domain ontology and records from various databases) will also make it complex. Even though they have a concern on real time information, their approach doesn't explain any solution which enables calculating any dynamic tour planning solution to the tourist. Only the user preferences have been considered in this application in tour planning. The dynamic changes of the environment such as changes of the events, weather etc. is not considered.

[8] João Laranjeira, João Carneiro, Goreti Marreiros, Ricardo Santos and Carlos Ramos introduced, an Agent Based Recommendation System for Tourism. The aim of their research is to show how a Recommendation System can assist tourists in their choices, not only the ones based on the usual variables such as time, money, among others, but also on their interests. To be able to simulate the tourists in a virtual world, allowing them to communicate with each other, exchanging information and evolving its profile and knowledge, the presented recommendation system was based on an architecture based on agents. In that multi-agent system, there exist two main types of agents: Agent Adviser, who will be responsible for the Recommendation Module and the Agent Tourist: an agent as this will represent the tourist in this virtual world.

The system presented by the techniques described may have a huge importance in the process of decision making for tourists in tourism domain. Assuming that nowadays a tourist is a person with a very busy life and expecting to make the process of choosing a destination to visit spending the shortest time possible, a recommendation system that focuses on these aspects will certainly be a great aid for the tourist. Their objective is not to set a new system but to create features and techniques that are still underdeveloped in this area.

Their main goal is that by using this system, tourists who use it, find and are presented with the best suggestions, the ones that really suit their interests. They expected that their recommendation system allows suggesting to tourists ideal recommendations. But they haven't either provide a complete tour planning solution or a recommendation which caters user needs for dynamic data.

2.3 Multi-agent approach in dynamic tour recommendation

Juan Pavón [9] in his paper "Mobile Tourist Guide Services with Software Agents" presents a mobile tourist guide service, which has been built as a multi-agent system. There are several types of agents in the system. Some of them working in the mobile devices and they interact with the user, and other agents distributed in servers. The agents who reside in the servers combine the Beliefs-Desires-Intentions (BDI) approach with learning capabilities of Case Base Reasoning (CBR) techniques. These techniques facilitate context-aware behavior which makes use of past experiences to find the best plans to achieve goals.

This system is able to calculate a tourist route, and modify it according to the conditions of the places to visit and the available time for the tourist. The main advantage of this design is the ability to extend the services of the tourist guide agent very easily. The development of mobile services has the benefit of distributing capabilities of agents among servers in a network. One important feature of mobile services is that they can be context-aware. As the user moves from one location to another, new information should be taken into account and some adaptation mechanisms are required. Because of its limited computing capabilities, adaptation mechanisms are normally difficult to implement in a mobile. In this case, the availability of collaboration with agents in powerful servers is a solution. If the user is new to the system, there is no past experience from him. The collective experience from the users who have the past experiences in the system can be combined to get this knowledge. Although distribution capabilities of agents are helpful, when dealing with mobile devices there is a need to consider failures in the connectivity. This requires caching of application data in the mobile device, which is complex task as the amount of data can be huge and vary dynamically.

This application consisted with two groups of agents. One is the group of agents which are deployed on the mobile devices and the other consists of those agents that run on servers as they have to process a considerable amount of information, both descriptive data and cases from past experiences. There is one assistant agent who resides in mobiles for each user of the system. Each tourist willing to use the system has to register and request one of these agents. The tourist has to provide his preferences; visiting period of time, cost, tourist interest in order to start the system. Once the tourist has configured the preferences for a tour, the associated agent will contact with, which assesses tourists and help them to identify tourist routes in the city taking into account their preferences. There is another type of agents who resides on network and will maintain updated information in information sources. In this way the set of agents will assist tourists in planning their tourist routes and enable them to modify their schedules on the move using wireless communication systems.

There is only one type of deliberative agent in this architecture, called planner agent who has been built as a BDI agent that makes use of a CBR engine. Only this agent can adapt to changing conditions by using past experience to decide which plans to propose. Even though this application provides the adaptability feature, it is not fully supported in a changing environment.

Houssein Ben-Ameur in his paper [10] presented A Multi-agent Platform for Travel Services Aggregation called NADIM-Travel. This is an agent-based architecture for trip planning that uses information gathered from the Web to build personalized travel plans. Some of the researchers have investigated the use of agent-based architectures to aggregate distributed travel and tourism information. Tourists need more intelligent tools to help them surf through the huge amount of information and to choose the best fares and travel packages. Although information overload has long been an issue in tourism, new technologies and business procedures are changing the way this information is accessible.

This platform involves a planner-coordinator agent that receives a travel-related request from a user and is responsible for successfully responding to the request. In addition, service agents act as gateways to the various external travel services providers. These agents use a distributed space they call "infospace" to share information. In its current form, this infospace is a shared virtual marketplace that is used by the agents to post requests and responses. In this platform, a planner-coordinator agent manages users' requests as well as the response aggregation process. Service agents, on their part, act as gateways to external service providers, and are utilized to retrieve the responses.

The autonomy of agents in this architecture makes the platform robust, scalable and dynamic. But this application has couple of issues that they have identified. Only limited service agents have been implemented at this stage. Also this architecture can be extended by adding more agents to connect to other services such as hotel and entertainment etc. The dynamic packaging of on-line travel services are not supported with sophisticated algorithms. Though this application is not directly related to the tour planning problem, it is studied to find some important aspects aspect of usage in multi-agent technology in planning on similar context.

For assisting roaming tourists, G.M.P. O'Hare and M.J. O'Grady [11] designed a context-aware tourist guide called Gulliver's Genie. The intelligent agents will collectively determine the user context in terms of position, orientation and user preferences. System agents are strong intentional agents based on a mental state comprising of Beliefs, Desires and Intentions (BDI) model. While visiting a city a tourist would have access to a wide variety of information related to the various attractions and other location aware services. Gulliver's Genie is a distribution of context sensitive information to tourists which provides personalized, up to date and multi-media enhanced presentations about such attractions. As tourists make their way towards some tourist attraction, the Genie displays an electronic map with their position and orientation highlighted upon it. As they approach a particular tourist attraction, the Genie will start pre-caching a presentation. Upon completion of pre-caching, and, provided the tourist is within a critical distance of the attraction, a presentation will be displayed. The presentation structure is simple and consistent.

A small image is displayed showing the attraction in question. A sound script and the initial image provide a brief introduction to the attraction. After this has been played, the tourist may select other topics of interest. Each topic consists of an image and sound script or a short video clip. As the tourist moves away, the Genie returns to its default state and restores the map and position indicator. Adaptive presentations are provided reflecting the individual's interests. Thus the follow up links will vary as will their associated content. Furthermore individuals can annotate the information space by depositing and attaching individual insights by way of hotspot creation. This process enables authors to create mixed media content, associate this with proscribed activation points and to designate an audience. Subsequently, this content acts as an experiential overlay available to designated groups or constituencies.

They have made maximum use of the existing technologies such as GPS and wireless data communications in a ubiquitous environment. A GPS receiver, laptop and sound recorder have been used to develop initial prototypes for demonstrating the possibilities of the Genie. One of their objectives has become to run this system on wide variety of devices such as laptops and PDAs. For this reason they have selected Java programming language to implement the system. Implicitly the tourists interact with their environment by moving within it. Inferences can be made from their movement and the system should plan accordingly. Explicit user interaction must be supported, whereby users can prompt the system. For example the system should be able to recommend a tourist, suitable restaurants within the immediate vicinity upon the request.

The architecture of this system is based on the classic client-server model: Genie Client. The client is essentially a hand held computer or PDA that is capable of displaying multimedia presentations and support for GPS. Genie Server consists with the components of multimedia database and software for assembling personalized presentations. Two of the agents reside on the hand held device and the remainder on the server side. The core component on the server is a Multi-Agent System (MAS) which instantiates and manages all the agents required. All agents are implemented using the Agent Factory (AF) system which was designed for the development of agents that paid little credence to computational constraints.

An alternate approach was adopted delivering a second implementation of the agent run time environment or Agent Virtual Machine (AVM) using Java. This system is known as Agent Factory Lite. Thus server hosted agents within the Genie are Agent Factory (Smalltalk) agents, whilst client hosted agents are Agent Factory Lite (Java) entities. Gulliver's Genie is hosted the Jakarta Tomcat Servlet engine for routing requests to the server. DB2 has been used for database purposes.

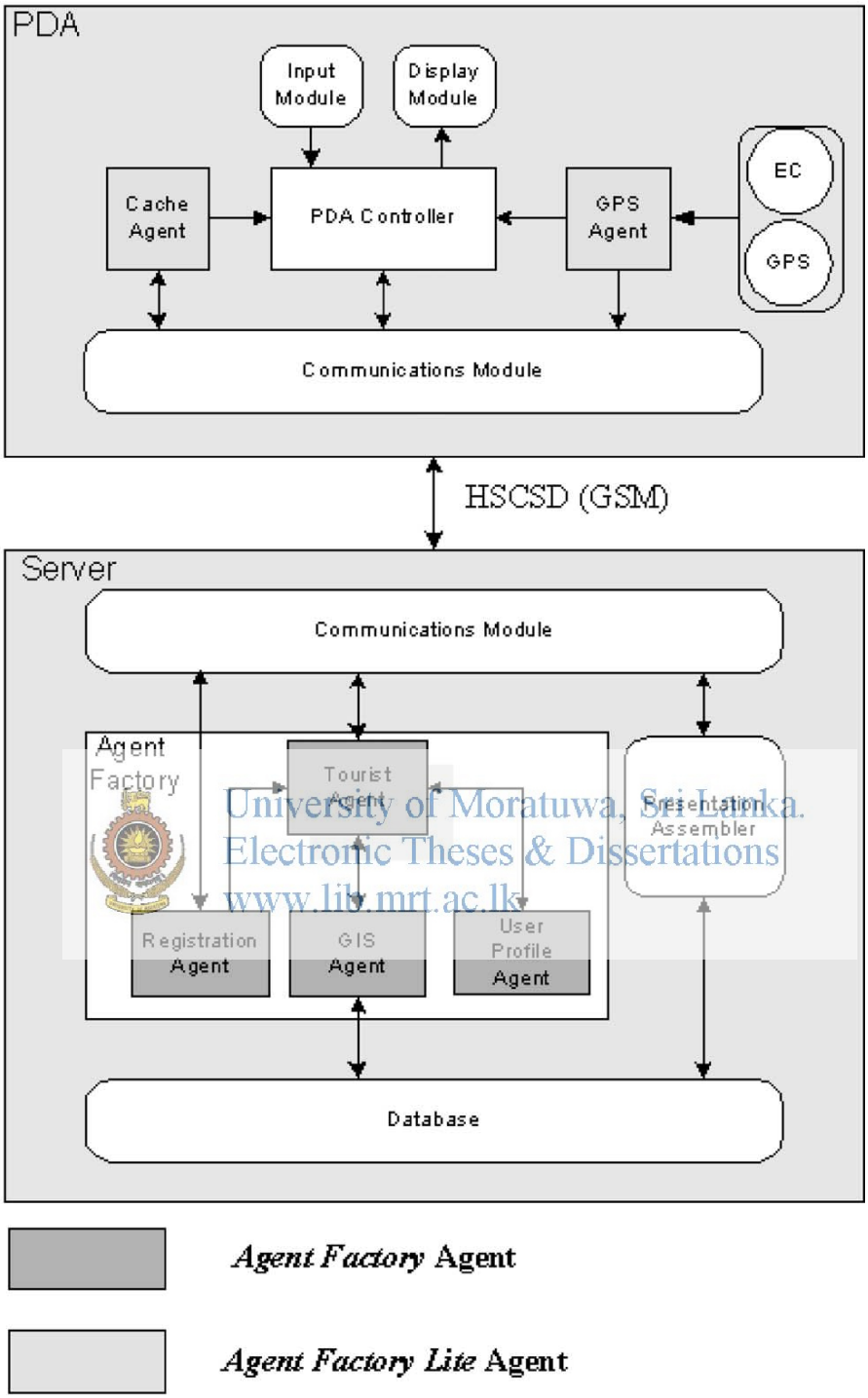


Figure 2.1: Architecture of Gulliver's Genie System

The architecture of the Genie is shown in Figure 2.1. Client components PDA Controller is responsible for the initial configuration of the client. Following initialization, it accepts the various inputs to the system and responds accordingly. Input Module monitors any user interaction and routes requests to the PDA controller. Display Module displays a map with the user's position and orientation highlighted. Generally, tourists are familiar with the map imagery. The map is thus used as the default display metaphor. As the user moves, the map is scrolled automatically. It also responds to explicit user scroll requests. When a presentation is available and the Cache Agent recommends that it should be shown, the Display Module displays it on screen and initiates the presentation.

The GPS Agent is responsible both for managing and interpreting the sensor data. It monitors both the GPS receiver and the EC. If the GPS Agent is satisfied that the position reading is valid, it then informs the display module which proceeds to update the user's position on the screen. Subsequently it compares this position with previous positions and decides if the user's activity has changed from, for example, walking to standing. If this is the case, it immediately informs the tourist's own agent (Tourist Agent) on the server. If the user's activity has not changed, the GPS Agent deliberates on whether the user has made what it considers to be a meaningful movement, i.e. moved more that 20 m from the last known position which was registered with the tourist's agent on the server. If this should prove to be the case, then a message is dispatched to the tourist's agent requesting that it updates its last known position for the tourist.

To overcome the slowness of the data transfer rates on wireless networks a Cache Agent has been introduced in this system. A pre-caching mechanism has been implemented via an agent to improve the performance. Once the server has been updated with a new position reading, it generates a list of nearby attractions and passes it on to the Cache Agent. This then monitors GPS positions from the GPS Agent and adjusts its list accordingly. Once it has identified an attraction that it predicted the tourist will visit, it asks the Tourist Agent on the server for a presentation about that particular exhibit.

The anticipation process involves consultation with the User Profile Agent, which uses an individual's profile as a predictor of future interests and activities. The Cache Agent notices the PDA Controller that a presentation is available. The controller then proceeds to arrange for the presentation to be displayed. Both the client and the server have communications modules that essentially perform similar functions of routing all requests and responses to the appropriate modules.

Agent Factory system provides the scaffolding for the operation of the four server side agents. The BDI deductive machinery is delivered enabling agents to reason based on a belief set and the application of commitment rules resulting in an adoption of commitments that will be honored at some future time point. Agents interact by way of an Agent Communication Language (ACL) called Teanga. This Multi-Agent System forms the core component on the server side and consists of the following agents:

Registration Agent: After verifying and authenticating that a tourist is eligible to use the Genie, the Registration Agent precedes to instantiate an agent from the Tourist Agent template creating a personal tourist agent for each individual. Tourist Agent; once tourists connect to the Genie server, an individual agent from the Tourist Agent template is instantiated for them. In consultation with the GIS Agent, it maintains a list of exhibits that the tourist is currently near and is responsible for ensuring that the Cache Agent is always furnished with an up-to-date list of exhibits, which it can subsequently monitor. The Tourist Agent frequently apprises the User Profile Agent of salient and observed user behavior, which it subsequently uses to dynamically update the user profile. GIS Agent. It provides GIS related services to the Tourist Agents and interfaces with the GIS component in the database. In particular, the GIS Agent handles cache update requests and recommends a list of exhibits for the Cache Agent to monitor. Ultimately, it is the Tourist Agent that decides whether the Cache Agent should have its list updated or otherwise. User Profile Agent dynamically updates the individual user profile reflecting user activities and inferred preferences. For example, once a tourist has listened to a given presentation, a record of what has been actively selected (and implicitly ignored) by the tourist is returned to the corresponding Tourist Agent.

This then asks the User Profile Agent to update the relevant user model accordingly. Presentation Assembler, as the name suggests, builds presentations for users. On receiving a request to build a presentation, it liaises with the User Profile Agent for information concerning the individual user and the attraction in question. This information acts as a filter of admissibility and it proceeds to dynamically build a presentation that matches the user's interests. As part of the pre-caching strategy employed by the Genie, the Presentation Assembler actually pre-caches presentations on the advice of the Tourist Agent. When a request comes in from the Cache Agent, this cache is then searched for the appropriate presentation. It then arranges for the presentation (including images, sound files and text) to be dispatched to the client.

The entire Gulliver's Genie system is a sophisticated database. In terms of multimedia, Genie presentations consist of a rich combination of sound, images, text and video clips. These media elements and the meta-data associated with them are stored in the multimedia section of the database. Managing electronic GEO coded maps is of fundamental importance of Geo-spatial. In addition, geo-spatial data allows the association of a set of GPS readings with a particular tourist attraction. All users of the Genie must first register in User Model. User profiles such as language, nationality, age, gender user interests such as art, literature, and etc. are developed and dynamically maintained for each user. Even though these attributed are maintained dynamically, this calculations in recommendations are not performed in adaptable manner as the proposed system does.

2.4 Summary

In this chapter we discussed about various studies related to the Ontology Based Modeling and Multi-agent approach on dynamic tour recommendation. More emphasis given on multi-agent based existing solutions proposed to address the tour recommendation problem. The next chapter presents the technology adapted to implement the proposed solution.

Chapter 3

Multi-Agent Systems and Tourism

3.1 Introduction

Previous chapter discussed about the findings of the review performed on the past research effort in the context of tour planning based on multi-agent technology. This chapter presents an overview of technologies used in implementing the tour planning application. The multi-agent technology is used in implementing the core modules while using the ontology concept in modeling tourism knowledge base in this research.

3.2 Multi-Agent Systems

The multi-agent technology is used in implementing the core modules while using the ontology concept in modeling tourism knowledge base in this research. Agent technology has been applied in recent years to solve various problems in tourism domain, such as dynamic service discovery, personalization of touristic information or planning of touristic activities etc. This research covers the planning of complex touristic activities in dynamic manner, to support tourism industry. Multi-agent technology is adapted to implement the solution since it has lot of powerful features which are capable to address this tour planning problem.

Certain types of software system are inherently more difficult to correctly design and implement than others. The simplest general classes of software systems are functional. Such systems work by taking some input, computing a function of it, and giving this result as output. Compilers are obvious examples of functional systems. In contrast, reactive systems, which maintain an ongoing interaction with some environment, are inherently much more difficult to design and correctly implement. Process control systems, computer operating systems, and computer network management systems are all well-known examples of reactive systems. In all of these

examples, a computer system is required that can operate independently, typically over long periods of time. However, for certain types of reactive system, even specialized software engineering techniques and tools fail. Hence new techniques are required. We can broadly subdivide these systems into three classes: open systems, complex systems, and ubiquitous computing systems.

3.3 Agents and the environment

[13] A multi agent system consists of a group of agents in a shared environment. Each agent has an internal state consisting of goals, beliefs, knowledge, plans and other such information. The environment also has state, referred to as external state, which agents can change by performing actions through the execution of their plans. Agents only have direct access to information in their own internal states. External state information must be obtained by sensing the environment. Other agents' internal state must either be obtained

3.4 Open Systems

An open system is one in which the structure of the system itself is capable of dynamically changing. The characteristics of such a system are that its components are not known in advance, can change over time, and may be highly heterogeneous (in that they are implemented by different people, at different times, using different software tools and techniques). Computing applications are increasingly demanded by users to operate in such domains. Perhaps the best-known example of a highly open software environment is the Internet.

3.5 Complex Systems

The most powerful tools for handling complexity in software development are modularity and abstraction. Agents represent a powerful tool for making systems modular. If a problem domain is particularly complex, large, or unpredictable, then it may be that the only way it can reasonably be addressed is to develop a number of

(nearly) modular components that are specialized (in terms of their representation and problem solving paradigm) at solving a particular aspect of it. In such cases, when inter dependent problems arise; the agents in the systems must cooperate with one another to ensure that inter dependencies are properly managed. In such domains, an agent-based approach means that the overall problem can be partitioned into a number of smaller and simpler components, which are easier to develop and maintain, and which are specialized at solving the constituent sub problems. This decomposition allows each agent to employ the most appropriate paradigm for solving its particular problem, rather than being forced to adopt a common uniform approach that represents a compromise for the entire system, but which is not optimal for any of its sub parts. The notion of an autonomous agent also provides a useful abstraction in just the same way that procedures, abstract data types, and, most recently, objects provide abstractions. They allow a software developer to conceptualize a complex software system as a society of cooperating autonomous problem solvers. For many applications, this high-level view is simply more appropriate than the alternatives.

3.6 Ubiquity

Despite the many innovations in human-computer interface design over the past two decades, and the wide availability of powerful window-based user interfaces, computer-naïve users still find most software difficult to use. One reason for this is that the user of a software product typically has to describe each and every step that needs to be performed to solve a problem, down to the smallest level of detail. If the power of current software applications is ever to be fully utilized by such users, then a fundamental rethink is needed about the nature of the interaction between computer and user. It must become an equal partnership.

To deliver such functionality, software applications must be: autonomous: given a vague and imprecise specification, it must determine how the problem is best solved and then solve it, without constant guidance from the user, proactive: it should not wait to be told what to do next, rather it should make suggestions to the user, responsive: it should take account of changing user needs and changes in the task environment, and adaptive: it should come to know user's preferences and tailor

interactions to reflect these. In other words, it needs to behave as an intelligent agent. These considerations give rise to the idea of an agent acting as an 'expert assistant' with respect to some application, knowledgeable about both the application itself and the user, and capable of acting with the user in order to achieve the user's goals.

3.7 Summary

This chapter described about the multi-agent technology which is used in implementing the tour planning application. Open, ubiquitous and complex nature of tour planning systems was addressed using the multi-agent systems. The next chapter describes the approach used in this system.

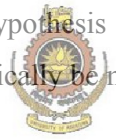
Chapter 4

Multi-agent based approach in tourism

4.1 Introduction

This chapter describes the multi-agent approach used in this tour recommendation application which is built on top of tourism ontology. The chapter explains the approach to achieve the aims and objectives in terms of the hypothesis, inputs, process, outputs, features and users of the system. By reading this chapter, the reader can understand the process of the research by knowing how the technology is adopted to solve the problems with the features of the system.

4.2 Hypothesis



The hypothesis of the research is "The complexity of the tourism domain can dynamically be modeled using ontology based multi-agent technology".

4.3 Input

The tourist has to provide as inputs, a set of parameters, such as interests, visiting period of time, restrictions related to cost etc. Visiting period is very important because the tour activities recommended by the system depend on the number of days that the tourist can spend. The interests specified by the user are the main concern in calculation the touristic activities. Interests of the tourist includes wildlife, scenery, hiking & nature trails, cultural discovery, beach and water related activities, meditation, spa & ayurvedic wellness, shopping and activities such as elephant rides, horse rides, boat rides, cart rides will be considered as the preferences. In addition to those, the preferences include the arrival date and departure date, maximum budget, type of accommodation (single, double, triple) hotel category: (five star hotel, four star hotel etc) meal plan: (bed & breakfast, half board, full board).

In addition to the above, the details such as number of adults, number of children will be useful in recommending suitable and affordable accommodation.

4.4 Process

According to the proposed solution, the multi-agent technology has been adapted to solve the tour recommendation problem on top of tourism ontology. This application provides a set of agents to create and modify tour recommendations based on the preferences of the tourists which will be taken as inputs to the system. Each tourist who wants to use the system has to register and solicit one of the Tourist Agents. This agent will always listen to the dynamic changes of the user preferences and suggest the changes to the existing tour recommendations accordingly. Once the tourist has configured the preferences for a tour, the associated Tourist Agent, based on those preferences, will contact the Planner Agent in order to calculate the tour activities in collaboration with service agents such as Religious Agents, Historical Agents, Leisure Agents, Relaxation Agent etc. These agents are dealing with the knowledge inferred from the tourism ontology in order to create the optimum tour recommendations for the tourist. Planner Agent will evaluate the user interest and will find the attraction attributes which are associated with those interests. A set of agents will then be created for those agents. For example a Beach Agent will be created if the user has an interest of the beach. Planner agent will delegate those agents the tasks of finding the relevant touristic activities. Those attraction agents will find the best locations, events and activities which has the attraction attributes that best matches with the attribute attached to the corresponding agent. All these agents will coordinate with planner agent and create tour recommendations for the specified duration by considering the time specified for those touristic activities. The tour recommendations provided as the output of the system consists with the tour locations, hotels and activities recommended for the specified days. Handling the dynamic aspect in calculating tour recommendations in the touristic environment is the main concern in this research. Dynamic changes of the touristic environment can happen due to various reasons. Due to bad weather an event can be canceled or the roads may be closed. A new location or event can be added to the system.

In such situation the corresponding attraction agents get activated and inform planner agent regarding the changes or newly added touristic activities. Based on the changes the planner agent will adjust the active tour plans by providing alternatives. The proposed system will assist tourists to create their own tour activities and find the information about the tourist activities, events and the locations of interest according to their preferences using a desktop computer or hand held device such as smart phone or PDA device.

4.5 Output

Touristic activities recommended for the specified preferences of the tourist are given as the output of the system. Touristic activities include the tour locations, hotels and the activities which are suggested for that duration depending on the user preferences. Interests of the tourist are mainly considered by the system in recommending the tour activities for a specific tourist.

4.6 Features

1. The system provides the optimum tour recommendation for the specified user preferences.
2. Only the relevant recommendations and results is given to particular user based on user preferences.
3. The system is capable enough to process data and generate the tour recommendations in efficient manner.
4. The system provides adaptable solutions on the changes of the user preferences or the other changes in the environment.
5. Cost effective tour recommendations are provided by the system.
6. Accurate information is provided in the computed tour recommendations.

4.7 Summary

This chapter explained about the approach used in this system in terms of input, hypothesis, output, process, features. The next chapter discusses the design of the system based on multi-agent technology and ontological modeling.

Chapter 5

Analysis & Design

5.1 Introduction

In this section an explanation will be given on multi-agent design of this system together with how ontologies are used to structure the knowledge on tourism domain with respect to tour recommendation. Further this chapter describes the architecture of the system with main modules and their functionality. The Design of the system mainly elaborates WHAT each module does in each stage to produce the final outcome of the research.

5.2 Ontology Based Complexity Modeling

The complex relationships which have been identified in the tourism are mapped in a way similar to the following using the ontology concepts.

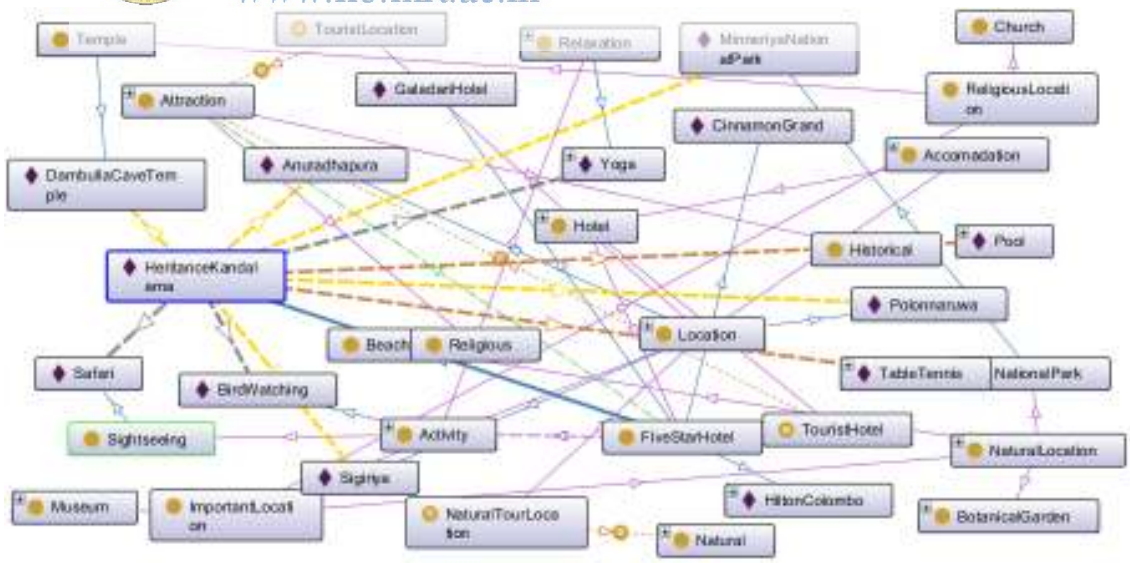


Figure 5.1: Ontology of the Proposed System

Ontology concept is used to model the complex real world scenarios in tourism domain in a systematically structured way. The concepts or 'classes' of the tourism sub domains are represented in RDF Schema which provides modeling primitives for organizing objects into hierarchies. It is viewed as a basic language for writing light-weight ontologies that are subClassOf taxonomy. In RDFS, objects sharing similar characteristics can be typed with classes. Examples of classes in this problem are events, attractions, accommodations, etc. Individuals belonging to a class are often referred to as instances of that class. For example, the "2014 new year festival" can be an instance of the events class. Classes can be grouped into hierarchies through the subClassOf relationship. For example, every instance of a cultural festival is an event instance since cultural festival is a subclass of the event class. The sub-domains which are covered by this ontology are as follows:

- Activity
- Location
- Attractions
- Events

These ontologies are designed using Protege, which is an ontology-development tool used in such a way that it can provide the type definitions. The ontology is kept classes (RDFS light-weight ontologies) provide the type definitions for each of the sub domains, and the instances of tourist entities for these classes are maintained in a database.

5.2.1 Attraction

Attractions are the variables those describe in terms of locations, activities and events, the nature of the interests of the tourists. For example spiritual attraction is the variable which describes the interest of a tourist who is having desires such as meditation. The meditation locations, or the other locations where we can find the meditation related activities or the locations where we can find events with meditation activities can be counted as spiritual locations. The location can be a temple or any other place.

5.2.2 Activity

Activity consists with the activities that tourists may wish to perform. Those activities can have one or more attractions. For example a Sightseeing activity can have an attraction of romantic, spiritual, religious or natural. These attractions will be matched with the nature of the interests specified by the tourists in order to make recommendations in calculating the touristic activities.

5.2.3 Events

The Events sub domain provides a model of a real-world event, characterized in terms of locations, activities, timescale, theme, and other properties. These locations and activities can have attractions. For example if the event is a beach party, a beach location itself can have the attraction Leisure. Further more if the party is consisted with the activity of dance, it will have the attraction of relaxation. Hence this event can be recommended to the tourist who has the interest on relaxation or leisure.

5.2.4 Location

A location can be a hotel, accommodation, tour location etc and a location can have attractions, activities or events. The attractions of events and activities are also inherently become the attractions of the tour location.

5.3 Multi-Agent Based Dynamic Recommendation

The following figure depicts the high level agent based design of the Dynamic Tour Recommendation system.

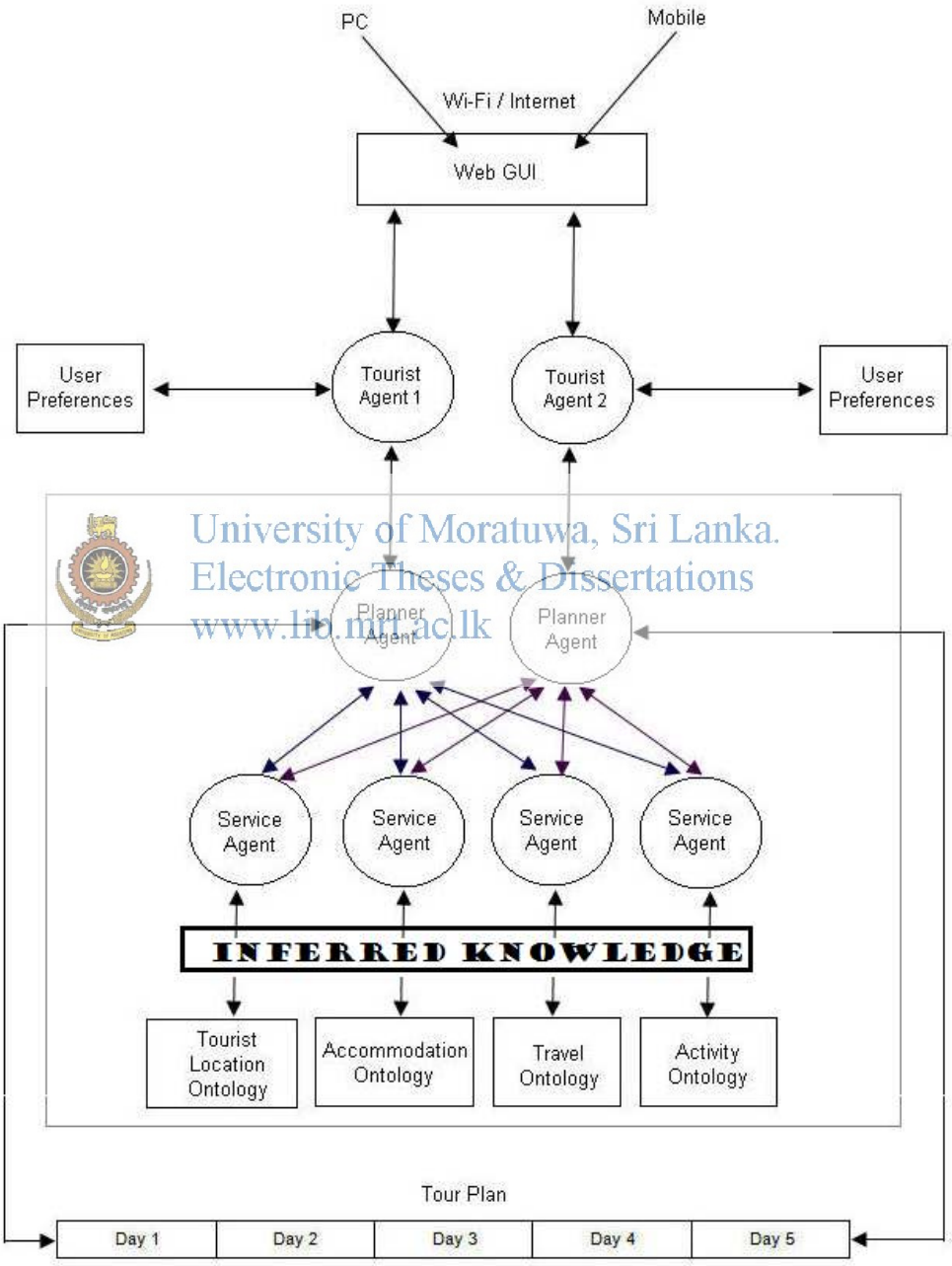


Figure 5.2: Overall System Design

5.3.1 Web GUI (Graphical User Interface)

Web GUI will be provided where the users can access to this application via desktop computers or a hand help device (HHD). As this application has been designed as a web application, having an internet connection is a must at this stage. A cache mechanism will be provided at a later stage where the tourists want to access to the system in an area which doesn't have internet access.

5.3.2 Mobile Application

The tourist agents are running in mobiles and only one tourist agent can be executed in one mobile device. When user open the mobile application a tourist agent will be created if an agent doesn't exists already. If a tourist agent is running already, that agent will get activated and work for tour planning. These tourist agents will consider the preferences of the user and delegate the required tasks to the planner agents in order to create tour recommendations. On the other hand, the output provided by the planner agents will be shown in the tour map by tourist agents.

5.3.3 Search

As we have a tourist information database, a comprehensive search is provided for all users of the system. The users can find any kind of information related to tourism. For example tour locations can be searched by attractions, foods, activities and etc. This module is very important for the tourists who already have an idea about his tour. Then they can find more information related to what they already have planned.

5.3.4 User Management

There are many types of entities involved in tourism: Tourists, Tour Location Providers, Event Organizers, Travel Agents, general public, social media, Hotels, Restaurants, Accommodation Providers, Food Sources, Tour Information Sources like wiki pages, Google information sources (Google maps, weather information) and

other tourism related organizations and people. Role base security mechanism is provided in such a way that the uses can be created and assign them the required modules and the level of privileges.

5.3.5 MAS - Multi-Agent System

MAS consist with another two sub modules in addition to the complete tour recommendation system. They are Location-centric Activity Planner (LAP) and the Day Planner (DP). LAP will capture the geographical location of a particular user at the current location and plan a tour around that location. DP will create a plan based on a given date by the user totally based on the dynamic information.

The following agents contains in the MAS will coordinate and communicate in order to create tour recommendations for a particular tourist.

Tourist Agents

Tourist Agents are the agents who take the input of user preferences from a tourist via the GUI and communicate with the planner agent and make the tour plan. These agents are also listening to the changes of the user preferences. Hence these agents will dynamically keep on informing the planner agents on sudden changes of the preferences of the tourist. However there will only be one tourist agent per a tourist always. The profiles and the preferences of the tourists will securely be maintained in a private location which is only accessible to that particular user account associated with the tourist. This agent displays the created tour recommendations to the tourist via GUI.

Planner Agents

Once the tourist agent observed the user preferences it will call a planner agent to perform the planning task by coordinating with the other required service agents. Upon creation the planner agent will send the calculated tour activities back to the tourist agent. Planner agent will have to communicate and coordinate with multiple

service agents in order to find recommendations for tourists based on their preferences which were taken as inputs. Planner agents should keep on listening to the tourist agents because planner agents should also consider the dynamic changes of user preferences in tour recommendation. Based on the nature of user interests inferred from user preferences, the tour activities will be calculated as it can fit in to the specified time duration.

Service Agents

Service Agents are the most important agents who perform the most difficult task in this application. Service agents include the Religious Agents, Historical Agents, Leisure Agents, and Relaxation Agent etc. Calculating tour recommendations for a particular tourist with set of specific desires is a collaborative task as it involves in dealing with a complex ontology. Service agents use the knowledge inferred from the tourism ontology. For an example, the historical agent will find all the locations, activities and events associated with historical attraction if user has specified historical interests in his preferences.

These agents are fully dynamic and specific to the tourist. For example an agent called spiritual agent will be created if the preferences of that particular tourist consists with some spatiality. If we introduce totally new attraction to the ontology such as Romantic, an agent called romantic agent will be created dynamically to work for the tourists who have some romantic desires. Service agents always aware of the changes of that ontology associated with them and the knowledge will be inferred. Hence they will keep informed to the planner agents about the changes of the ontology related to existing tour plans. Then the tourist may have the option on re-creating their tour plans as they aware of the changes which can affect already planned tour activities. The users with hand held devices (with GPS) may receive touristic information at any point of the city dynamically. All the instances of locations and events contain their GIS information in the ontology. Therefore the service agents are also aware of its nearby tour locations and events. Therefore they can make recommendations on tour activities in vicinity.

Tracker Agent

Tracker agent will perform search for the changes of that particular attraction and notify the changes to the particular service agents where needed. This agent also plays a major role, since it maintains all up-to date information for making the system more dynamic. For example if new location instance which is having a historical attraction is added to the store, the historical agent will be notified by the tracker agent and then the historical agent will search the store for historical related tourist activities.

5.4 Summary

In this section an explanation will be given on various agents used in this system and how the modules of the systems are designed. Also the design of the ontotologies is explained. The next chapter presents the implementation of the modules described in this chapter.

Chapter 6

Implementation

6.1 Introduction

The previous chapter described the design of the modules of this application. This chapter presents an overview of the implementation of tour planning application. This section further elaborates the implementation details of each module along with the tool and technology, methodology, algorithms and techniques. Mainly implementation chapter provides a clear idea about HOW to develop each module to produce the final outcome of the research.

6.2 Overview

The two main characteristics of the tourism domain that make it suitable for semantic web technologies are the heterogeneity of the market and the distributed nature of the high volume of information on the web. In this thesis, we aim to bring semantics and structure to bear on tourist information on the web together with the multi-agent technology to cater the needs in a dynamic environment.

A multi-agent based system should start with the preference of tourist, search for up-to date information in dynamic manner, ignore irrelevant information, combine several web resources of the tourism sub-domains, and generate tour recommendations. Ontologies are used in knowledge-based systems as conceptual frameworks for providing, accessing, and structuring information in a comprehensive manner. This system is an integration of search, recommendation and planning sub-systems. The main focus of this thesis is tour recommendation according to user preferences and constraints, aids in the selection and scheduling of various aspects of a tour such as events, attractions, and routes. This system can also function independently as a location centric recommender or day planner of the aforementioned tourist entities.

As touristic information is highly dynamic, those changes happening in the touristic information sources should be reflected in tour plans and tour recommendations. In order to achieve this objective, an adaptable solution has to be designed. The multi-agent technology is a good methodology to implement an adaptable solution to this kind of problems. The complexity of the touristic attractions and events can be addressed using conceptual design of ontologies.

6.3 Sub Modules

6.3.1 Web GUI (Graphical User Interface)

Users will connect to the MAS (Multi Agent System) via Web GUI which is implemented using JSP (Java Server Pager) and struts framework for the web users. Refer Appendix A for the implementation of the prototype pages of the GUI.

6.3.2 Mobile Application

In order to handle the dynamic information, this agent frequently send requests to the planner agents and find whatever the changes occurred in the touristic information sources. Then filter that information and present only the relevant information to the user. As the users are moving location to location, at a certain time this tourist agent can capture the location details of a particular user and pass it to the planner agents in order to calculate location-centric tour plans. Nearby locations which is considered in calculation of tour recommendation are also calculated in this way by using the GPS technology of the mobile device.

6.3.4 User Management

In order to maintain up-to date information, most of the above entities are provided an interface to access to the system. As those different entities deal in different ways in the system, role base authorization mechanism is implemented. Those activities performed by a particular user is defined in a flexible way and stored in a MySQL database.

6.3.5 MAS (Multi-Agent System)

MAS is the heart of this tour recommendation system which consists with Tourist Agents, Planner Agents and the Other Service Agents. Accessing those agents using a web client was a challenging task. To achieve this, JADE library with spring framework is used as the main technology. While the agents are maintained in the agent container of JADE, a separate Spring container is used to maintain the other objects such as User, Preferences and etc.

JADE environment

JADE environment is injected to the UserPreferenceAction.java as shown in the source below. This class will initiate the container if it is not started already. Once the user specified his preferences, the mobile agents are activated based on the preferences.


```
public UserPreferenceAction(UserManager userManager, Environment environment)
{
    this.userManager = userManager;
    this.environment = environment;
}
```

After the user manager did its task by adding the preferences to the database, the required agents will be creates using environment object which is injected.

```
userManager.addUserPreference(preference);
environment.createAgents(attractionList);
```

The required agents are created in the createAgents() method. The desired no of planner agents will be created in this way.

```
AgentController plannerAgent = mc.createNewAgent("Planner1",
PlannerAgent.class.getName(), arguments);
```

Planner Agent class itself contains the planner logic. Based on the already defined attraction agents (Service Agents) which are retrieve from the database, the desired attraction agents will be created in the container at deployment time. Attraction Agents will provide different services such as Wild Life service, Beach service and etc.

```
Object[] arguments = new Object[2];
        arguments[0] = locationService;
        arguments[1] = attractionList;

for (Attraction a : attractionList) {
      (mc.createNewAgent(a.getName(),AttractionAgent.class.getName(),
arguments)).start();
}
```


LocationService object is provided by the Spring Container as it is injected to the Environment. This LocationService object also pass as an argument to the agent.

Tour Recommendation

The core of this system is tour recommendation which will be carried out by the Planner Agents. Yet the Planner Agent cannot perform his task without coordinating with Service Agents. Both kinds of agents reside in the agent container provided by JADE. Refer Appendix A for the complete source code of Planner Agent.

At the initialization of Planner Agent, it will retrieve the LocationService and the attractionList came as an arguments.

```
LocationService ls = (LocationService) args[0];
List<Attraction> attractionList = (ArrayList<Attraction>) args[1];
targetAttraction = attractionList.get(0).getName();
```

Now add a TickerBehaviour that schedules a request to seller agents every minute and find required service agents from DFService.

```
DFAgentDescription[] result = DFService.search(myAgent, template);

attractionAgents = new AID[result.length];

for (int i = 0; i < result.length; ++i) {
        attractionAgents[i] = result[i].getName();
        System.out.println(attractionAgents[i].getName());
}
```

It will then perform a request to the Service Agents.


```
myAgent.addBehaviour(new RequestPerformer());
```

Relevant Service Agents will receive the request and will send List of proposed touristic activities to the planner agent. Then planner agent will look in to all the activities and compute the plan and sent it back to the tourist agent in order to show it in tour map of user.

6.4 Ontology - Inferred Class Hierarchy

Class Hierarchy and Inferred class hierarchy which is derived from Ontoloogy is as follows. Refer Appendix C for full hierarchy.

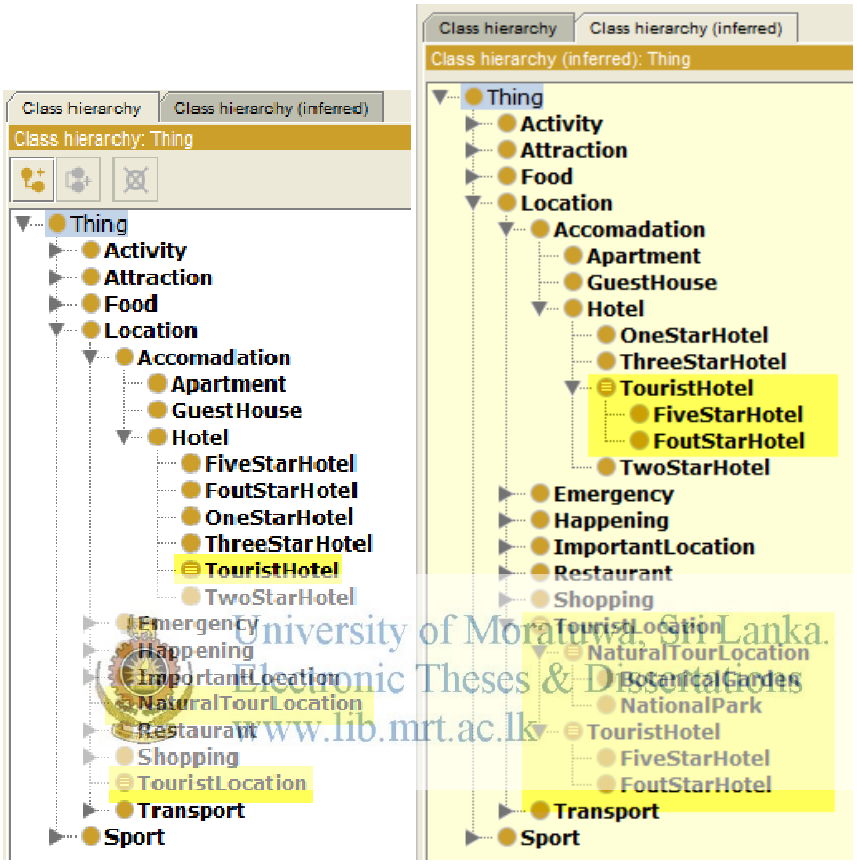


Figure 6.1: Ontology - Inferred Class Hierarchy

For example the Hotels which are having some attraction of any kind have been defined as TouristHotels. Refer Appendix D for full OWL ontology.

```
<!-- http://www.location.lk/ontologies/location.owl#TouristHotel -->
  <owl:Class rdf:about="&location;TouristHotel">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <rdf:Description rdf:about="&location;Hotel"/>
```

```
            <owl:Restriction>
              <owl:onProperty rdf:resource="&location;hasAttraction"/>
              <owl:someValuesFrom rdf:resource="&location;Attraction"/>
            </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
      </owl:equivalentClass>
    </owl:Class>
```

6.5 Hardware & Software

Mainly a Tomcat Servlet container and MySQL database will be used to deploy the system by providing web access to anyone who use personal computers or hand held devices. The tourists are preferable to use hand held devices during the tour in order to access their geographical locations using GPS facilities of their hand held device. Hand held devices with Android and GPS facilities will enable more features for the users. Having an internet connection is a must until this system is improved to manage the cache. The system will deal with cache of the touristic information during the course of the tour whenever user does not have an internet connection. In the development work, mainly Jade and Jena libraries will be used with J2EE. Since this system is developed as a platform for all touristic parties, the Java Enterprise Edition will be used by providing scalability and security. Jade libraries are used to develop agent related implementations while Jena is used for ontological manipulations of the touristic information. The mobile application is developed using Android.

6.6 Summary

This chapter described the implementation of this application and its sub modules. Implementation of Mobile Application, user management, the core module of tour recommendation and the hardware and the software requirements are explained. The next chapter presents the evaluation of the system.

Chapter 7

Evaluation

7.1 Introduction

The previous chapter described the implementation of the modules of this application. This chapter discussed on how the tour recommendation system is tested under typical operations using various types of different queries. Each of the tourism sub-domains are tested with different sample data sets and questionnaires. Mainly two aspects of the system are thoroughly tested: computation of tour recommendations and the dynamic aspect of tour recommendation.

7.2 Evaluation Strategy

Two strategies were used to evaluate the system. The system is tested and gathered the results using a sample test data set which is selected based on the preferences of tourists. Also another set of results is obtained through survey conducted among 30 participants who has an interest on travel and tourism. A questionnaire is used to get the feedback of the proposed system opposed to existing systems. The results gathered from both of those strategies are analyzed and performed a statistical evaluation in order to conclude that the system has met its objectives.

7.3 Evaluation questionnaire

A questionnaire is prepared and distributed to the group of participants who have been used similar system to plan their travels. Thus the user accounts were created and access is given to them in order to use the system. The answers for those questions provided by those users are used to evaluate the system. The questionnaire consists of two sections 10 questions per each. One section is related to existing systems and the other system is based on proposed system. 30 participants were selected for this survey.

7.3.1 Adaptability of proposed system

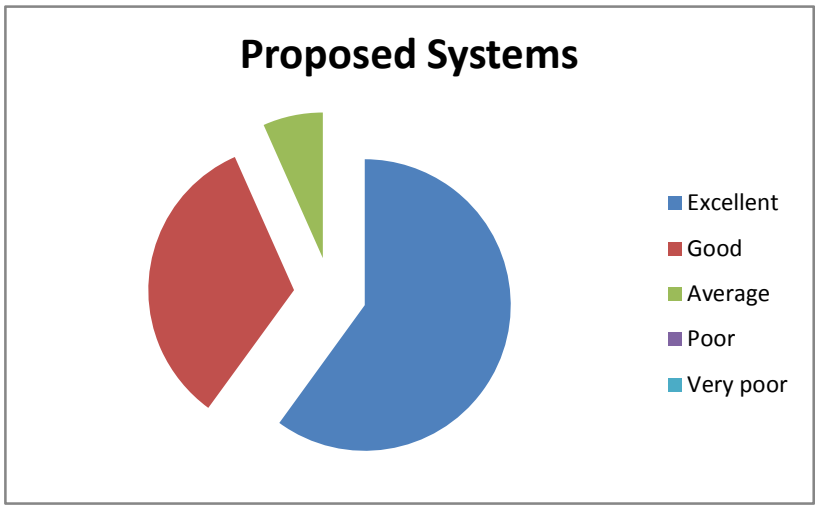


Figure 7.1: Adaptability of proposed system

The adaptability is ability to adapt to the changes of the user preferences and the environment factors. 18 participants reported the adaptability as excellent while 10 participants reported the adaptability as good. Only two participants mentioned that the adaptability is at average level. Based on this information the system has achieved its main objective of adaption to the changes in a dynamic environment.

7.3.2 Accuracy

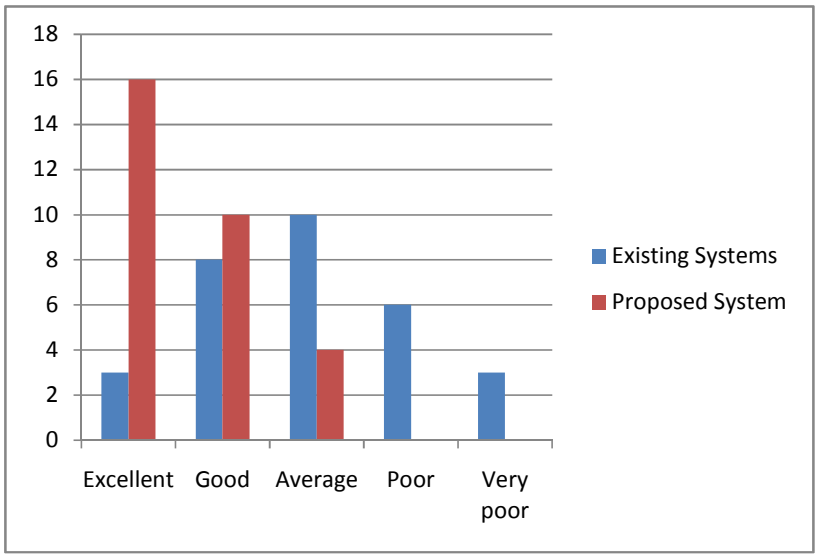


Figure 7.2: Accuracy

The accuracy of the output provided by the system is very important in tour recommendation problem. Hence the accuracy of the proposed system is measures against the accuracy of the existing systems. The proposed system gave a higher accuracy than the existing systems. The reason is that the touristic activities recommended by the proposed system were much more accurate since the proposed system is always based on up-to date information as it can dynamically adapt to the changes.

7.3.3 Relevancy of Recommendations

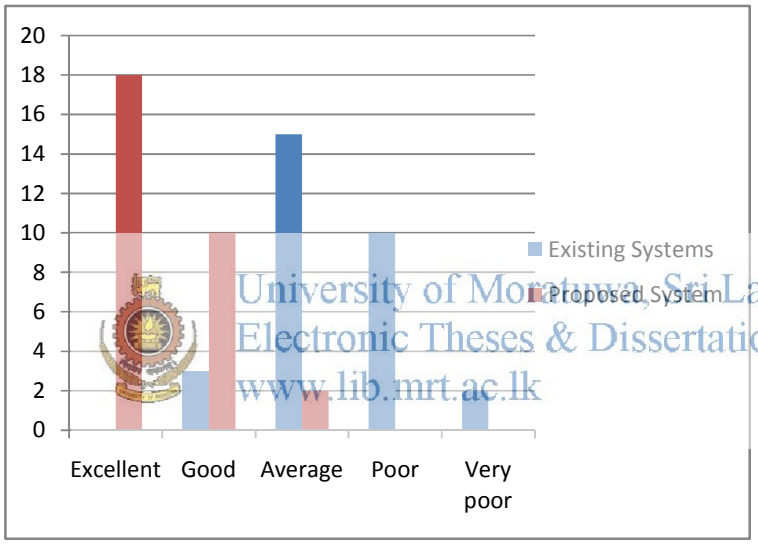


Figure 7.3: Relevancy of Recommendations

Relevancy of recommendations of the proposed systems is measured against the existing systems. 18 out of 30 participants rated the relevancy as excellent. Comparing to the feedback received for existing systems, the relevancy of recommendations given by proposed system was high.

7.3.4 Response Time

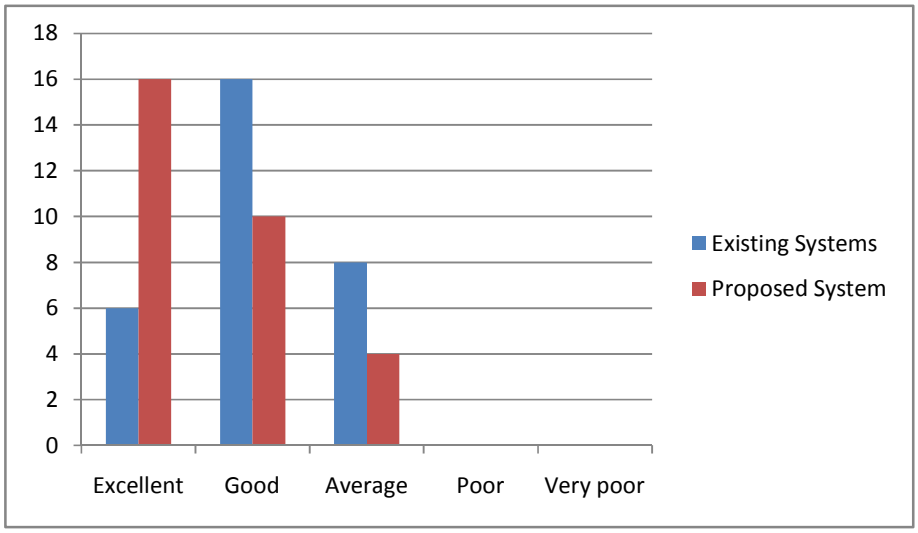


Figure 7.4: Response Time

Response time is the time taken to calculate the tour plan and present it to the user. Comparing to the existing systems the proposed system has given quick response to the users according to the feedback received. Less number of participants reported as response time is good because majority has reported as excellent.

7.4 Test Data

The test data samples used for the entire evaluation is as follows. Yet the same location can have multiple attractions. Some of the results generated by the system for the specified user preferences are attached in Appendix B.

Touristic Information	Type of Location	No of Locations
Activities	Relaxation	10
	Sightseeing	20
	Cycling	5
	Surfing	10
	Diving	10
	Hiking	5
	Night Life	5
	Rafting	2
Attractions	Cultural	8
	Historical	10
	Nature	20
	Spiritual	25
Other	Happenings	15
	Accommodation	20
	Emergency	5
	Restaurants	10
	Shopping	10

Table 7.1: Test Data Summery

7.5 Computation of Tour Recommendations

Accuracy of the computed touristic activities based on the user preferences and the environment changes. The computation of recommendations was performed based on the above data set which is consisted of 100 locations and attractions of real data. A sample of 20 tourist's profiles is used and created tour recommendations based on their preferences. According to the results 80% accuracy is obtained based on the user satisfaction about the tour recommendation calculated by the system.

The results given by the survey which was based on the feedback of real users, was much more accurate than the testing performed.

7.6 Dynamic Aspect of Tour Recommendation

The adaptability of the system is evaluated by creating dynamic changes on touristic information simultaneously based on 100 locations of test data. The agents were able to adapt with 95% accuracy according to the evaluation results obtained by testing using test data and survey conducted through a questionnaire. 100% accuracy can be achieved if all the touristic information is up-to date.

7.7 Summary

This chapter explained the evaluation of the system. Mainly two aspects of the system are thoroughly tested: computation of tour recommendations and dynamic aspect of tour recommendations. This chapter also explained the way that the results gathered from both of the evaluation strategies used and the statistical evaluation performed in order to conclude that the system has met its objectives. The next chapter gives the conclusion of the system.

Chapter 8

Conclusion and Future Work

8.1 Introduction

In the previous chapter, we discussed about the evaluation. In this chapter, we discuss conclusions, problems encountered during the development of the project and the developments that need to be done in the near future. This chapter contains two main sections and first section describes the achieved objectives of the research while second section provides directions to improvements and future works of the system.

8.2 Conclusion

With the development and evolve of technology we live in a world where it is necessary to have tools that help us to find solutions to solve problems such as tour planning and recommendation. Agent technology has shown the capabilities to be a reliable tool to find solutions to the problems in different scenarios like medicine, information retrieval, education and tourism. Tourism industry is an area with an economic potential and social benefit for the future and there are range of users according to age, interest and purchasing power. Tourism suggestions and services must include a huge level of additional facilities to meet the requirements of its end users, the tourists.

Agent technology can be defined as a supportive technology to be applied to domains in which information is physically distributed and a set of autonomous entities have to join their efforts and coordinate their activities to solve a complex task. Tourism is a domain is which is collaborated with numerous factors and a tourist needs to search for information related to touristic activities available in a given city, filter those that fit with the personal interests of tourists, and try to build a plan in which the selected activities may be performed within a given time span.

A planning system that focuses on these aspects will certainly be a great aid for the tourist with a very busy life and expecting to make the process of choosing a destination he preferred to visit spending the shortest time possible. Using this system users are allowed to suggest to tourists, the ideal recommendations to his profile including the recommendations that tourists could not imagine at a first glance. This research was mainly focused to design, implement, and evaluate a tour planner based on Tourism in Sri Lanka. I have designed a multi-agent system together with light-weight ontologies using RDFS to capture the tourism sub domains such as Attractions, Events, Hotels and etc. Then the ontologies are built based on Sri Lanka tourist information as they can maintain by the tourist information providers. The ontology-structured knowledgebase is used by the multiple types of agents in order to generate tour recommendations. The system presented by the techniques described has a huge importance in the process of decision making for the tourists.

The proposed system is evaluated by conducting a survey. According to the results of the testing performed by using sample data and the feedback received on the survey questionnaire, the following objectives of this system could be achieved.

i. To achieve the adaptability of the system, the multi-agent technology is used to cater the changes in user preferences and tour activities, in dynamic manner in tour recommendation calculation.
ii. To provide the optimum recommendations for the specified user preferences, the attributes associated with both tour activities are user preferences are analyzed and compared.
iii. Only the relevant recommendations and results will be given to particular user based on user preferences.
iv. The system is capable enough to process data and generate the tour recommendations in efficient manner.
v. Cost effective tour recommendations should be provided by the system.
vi. Accurate information is provided in the computed tour activities and recommendations.

According to the results of the evaluation, the system can provide adaptable solutions on the changes of the preferences or the other changes in the environment of a tourist. In addition to that the system can provide the optimum tour plan for the specified user preferences. This system makes the lie of tourist easy by providing only the relevant recommendations and results to a particular user based on user preferences. The most important thing is the system is capable enough to process data and generate the tour plan with higher accuracy rate in efficient and cost effective manner.

Tourism planning can be defined as a complex activity requiring an integrated approach, private and public sector partnership, and inter-agency and inter-sectoral coordination, as well as community involvement. In order to achieve the challenges of the tourism enterprise the tourism planning should be more dynamic by considering social, cultural, climatic changes, etc. The main challenge of the research was the coordination and integration of disparate information and service resources together with the provision of personalized assistance to tourists during their trip.

8.3 Future Work

Future work includes implementing a cache manager in order to enable the touristic information even when the user doesn't have an internet connection. Furthermore this system can be improved as a tourist platform as a place where everyone who has an interest to tourism get together and share the knowledge.

8.4 Summary

This chapter presents the conclusion of the research based on the evaluation performed. Further it explains my contributions towards the research together with the challenges faced, limitations and future work. I have achieved the main objective of the project by providing a system which computes adaptable tour plans and recommendations based on the changes of the preferences or the other changes in the environment of a tourist.

References

[1] Ana Casali1, Llu´ Godo and Carles Sierra. A Tourism Recommender Agent: from theory to practice; Depto. de Sistemas e Inform´tica - Depto. de Cs. de la Computaci´n, Facultad de Cs. Exactas, Ingenier´ y Agrimensura, Universidad Nacional de Rosari, Av Pellegrini 250, 2000 Rosario, Argentina. Institute d'Investigaci´ en Intel·lig`ncia Artificial (IIIA), CSIC .Campus de la Universidad Aut´noma de Barcelona.

[2] Stephen Page, Connell J, Tourism: a modern synthesis; London,Cengage Learning EMEA, 2006.

[3] Moutinho, L. - Consumer behavior in tourism, European Journal of Marketing

[4] Alexander Maedche, Steffen Staab, Applying Semantic Web Technologies for Tourism Information Systems; Research Center for Information Technologies at the University of Karlsruhe, Research Group Knowledge Management(WIM), January 2002.

[5] Heum Park, Task Model and Task Ontology based on Mobile Users' Generic Activities for Task-Oriented Tourist Information Service ; Department of Transportation Engineering, Youngsan University, Yangsan-si, Kyungnam-do, Republic of Korea, hmpark@ysu.ac.kr

[6] Susana Fern´ ndez, Laura Sebastia, Juan Fdez-Olivares, Planning tourist visits adapted to user Preferences; Univ. Politecnica de Valencia

[7] Heum Park, Aesun Yoon and Hyuk-Chul Kwon, Task Model and Task Ontology for Intelligent Tourist Information Service; Center for U-Port IT Research and Education, Department of French, and School of Computer Science and Engineering, Pusan National University.

[8] João Laranjeira, João Carneiro, Goreti Marreiros, Ricardo Santos, Carlos Ramos, An Agent Based Recommendation System for Tourism.

[9] Juan Pavón, Juan M. Corchado, Jorge J. Gómez-Sanz and Luis F. Castillo Ossa. Mobile Tourist Guide Services with Software Agents; Dep. Sistemas Informáticos y Programación Universidad Complutense Madrid, Ciudad Universitaria, 28040, Madrid, Spain. Dep. Informática y Automática , Universidad de Salamanca , Plaza de la Merced s/n , 37008, Salamanca, Spain

[10] Houssein Ben-Ameur, François Bédard, Stéphane Vaucher, Peter Kropf, Brahim Chaib-draa, Robert Gérin-Lajoie, NADIM-Travel : A Multiagent Platform for Travel Services Aggregation, Center for Interuniversitary Research and Analysis on Organizations Canada, Département d'études urbaines et touristiques, Université du Québec à Montréal, Canada. Département d'informatique et de recherche opérationnelle, Université de Montréal, Canada.

[11] G.M.P. O'Hare, M.J. O'Grady, Gulliver's Genie: a multi-agent system for ubiquitous and intelligent content delivery, Practice and Research in Intelligent Systems and Media (PRISM) Laboratory, Department of Computer Science, University College Dublin (UCD), Dublin, Ireland.

[12] N. R. Jennings and M. Wooldridge, Applications of Intelligent Agents, Queen Mary & Westfield College, University of London.

[13] Dave Gurnell, A Comparative Study of Approaches to Multi Agent Planning, Doctor of Philosophy, School of Computer Science, The University of Birmingham, Birmingham B15 2TT, United Kingdom, September 2005

[14] Olawande Daramolaa, Mathew Adigunb, Charles Ayo, Building an Ontology-Based Framework for Tourism, Recommendation Services, Department of Computer and Information Sciences, Covenant University, Nigeria, Centre for Mobile e-Services for Development, University of Zululand, South Africa.

Appendix A

Screen Prototypes

1. Tour map

   This is how the tour map is shown in the map as the output of the system. The shortest path has been calculated and shown using polylines of google map.

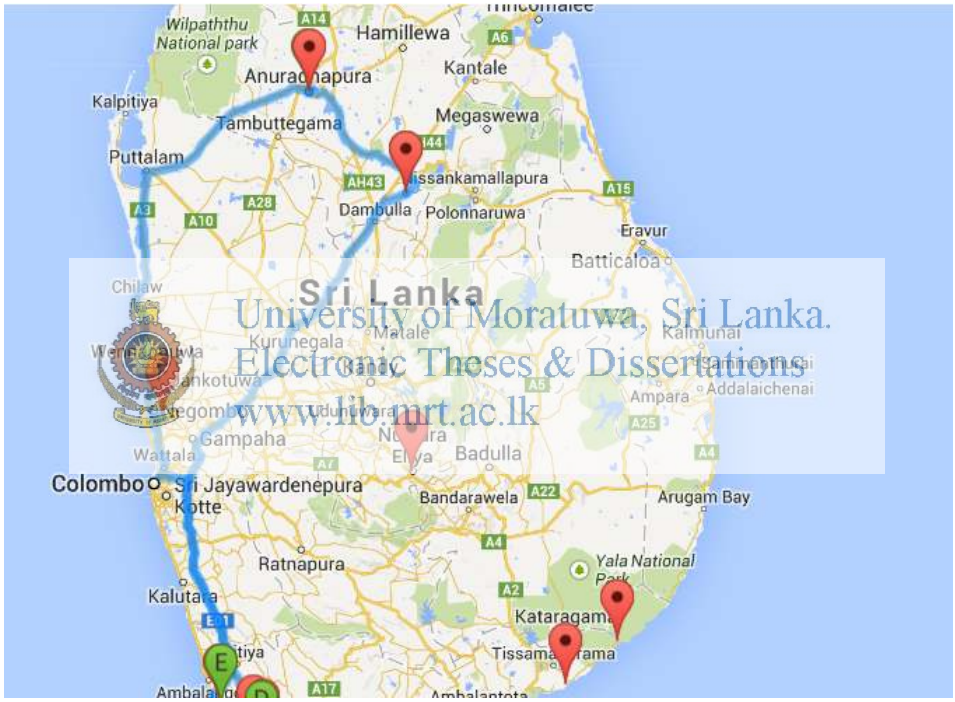


Figure A.1 : Tour Map

2. Login Screen

   This screen will be used to create new user accounts for the tourists and the administrators of the system. The same screen will be used to login to the system.

Figure A.2 : Login Screen

2. Home page

   Once login to the system, the recommended touristic activities of the logged in user will be shown in the home page.

Figure A.3 : Home Page

3. User Preferences

   This screen will be used to capture the preference of the user in order to calculate the tour plan.

Figure A.4 : User Preferences

4. Touristic Activities

   Tour Location Providers will use this screen to input touristic activities, tour locations and events.

Figure A.5 : Touristic Activities

Appendix B

Query Results

1. What are the five star hotels available around the natural tour locations in Sri Lanka?

```
PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
PREFIX owl: http://www.w3.org/2002/07/owl#
PREFIX xsd: http://www.w3.org/2001/XMLSchema#
PREFIX rdfs: http://www.w3.org/2000/01/rdf-schema#
PREFIX tp: http://www.location.lk/ontologies/location.owl#
                    SELECT ?Hotel ?NaturalLocation
                    WHERE { ?Hotel rdf:type tp:FiveStarHotel .
                    ?Hotel tp:hasCloseLocation ?NaturalLocation .
                    ?NaturalLocation rdf:type tp:NaturalTourLocation . }
```



```
==========
Result
==========================================
| Hotel                | NaturalLocation |
==========================================
| tp:KirindaBeachHotel | tp:Udawalawe    |
------------------------------------------
==========================================
Inferred Result
==========================================
| Hotel                | NaturalLocation         |
==========================================
| tp:KirindaBeachHotel  | tp:YalaNationalPark      |
| tp:KirindaBeachHotel  | tp:Udawalawe             |
| tp:HeritanceKandalama | tp:DambullaCaveTemple    |
| tp:HeritanceKandalama | tp:MinneriyaNationalPark |
```

1. Find the other activities of the tourist Locations which has the activity Bird Watching

```
PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
PREFIX owl: http://www.w3.org/2002/07/owl#
PREFIX xsd: http://www.w3.org/2001/XMLSchema#
PREFIX rdfs: http://www.w3.org/2000/01/rdf-schema#
PREFIX tp: http://www.location.lk/ontologies/location.owl#
                    SELECT ?TourLocation ?Activity
                    WHERE { ?TourLocation rdf:type tp:Location .
                    ?TourLocation tp:hasActivity tp:BirdWatching .
                    ?TourLocation tp:hasActivity ?Activity . }
```

==========

Result

==========

```
-------------------------------------
| TourLocation      | Activity      |
=====================================
| tp:AmayaLeasurePark | tp:TableTennis  |
| tp:AmayaLeasurePark | tp:Safari       |
| tp:AmayaLeasurePark | tp:BirdWatching |
-------------------------------------
```

==========

Infered Result

==========

| TourLocation | Activity |

==========

| tp:AmayaLeasurePark | tp:TableTennis |

| tp:AmayaLeasurePark | tp:Safari |

| tp:AmayaLeasurePark | tp:BirdWatching |

| tp:KirindaBeachHotel | tp:SunBathing |

| tp:KirindaBeachHotel | tp:BirdWatching |

| tp:HeritanceKandalama | tp:Yoga |

| tp:HeritanceKandalama | tp:BirdWatching |

| tp:HeritanceKandalama | tp:Safari |

| tp:HeritageDambulla | tp:TableTennis |

| tp:HeritageDambulla | tp:BirdWatching |

| tp:AraliyaHotel | tp:TableTennis |

| tp:AraliyaHotel | tp:Pool |

| tp:AraliyaHotel | tp:BirdWatching |

| tp:YalaNationalPark | tp:Safari |

| tp:YalaNationalPark | tp:BirdWatching |

Appendix C

Inferred Class Hierarchy of Ontology

Class Hierarchy and Inferred class hierarchy is as shown below. Refer owl in Appendix E for full class hierarchy.

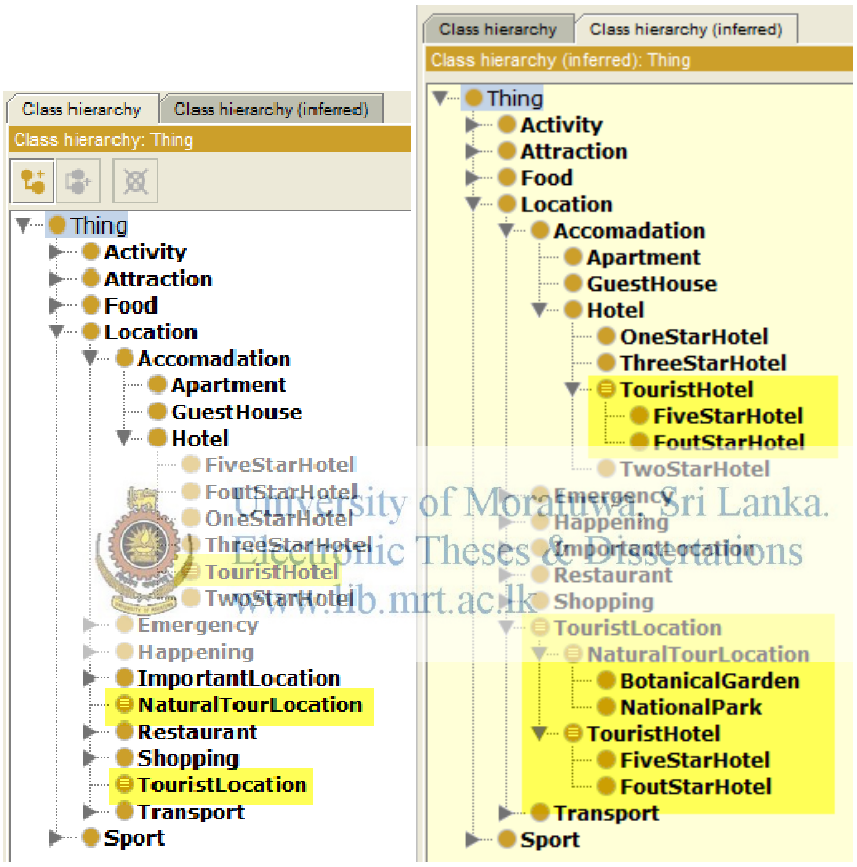


Figure C.1 : Inferred Class Hierarchy of Ontology

For example the Hotels which are having some attraction of any kind have been defined as TouristHotels.

Appendix D

Ontology OWL

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY location "http://www.location.lk/ontologies/location.owl#" >
]>

<rdf:RDF xmlns="http://www.location.lk/ontologies/location#"
     xml:base="http://www.location.lk/ontologies/location"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:location="http://www.location.lk/ontologies/location.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about="http://www.location.lk/ontologies/location.owl"/>

    <!-- http://www.location.lk/ontologies/location.owl#hasActivity -->

    <owl:ObjectProperty rdf:about="&location;hasActivity">
       <owl:inverseOf rdf:resource="&location;isActivityOf"/>
    </owl:ObjectProperty>

    <!-- http://www.location.lk/ontologies/location.owl#hasAttraction -->

    <owl:ObjectProperty rdf:about="&location;hasAttraction"/>
```

```
<!-- http://www.location.lk/ontologies/location.owl#hasCloseLocation -->

<owl:ObjectProperty rdf:about="&location;hasCloseLocation">
    <rdf:type rdf:resource="&owl;SymmetricProperty"/>
    <rdf:type rdf:resource="&owl;TransitiveProperty"/>
</owl:ObjectProperty>

<!-- http://www.location.lk/ontologies/location.owl#hasFood -->

<owl:ObjectProperty rdf:about="&location;hasFood"/>

<!-- http://www.location.lk/ontologies/location.owl#hasLocation -->

<owl:ObjectProperty rdf:about="&location;hasLocation"/>

<!-- http://www.location.lk/ontologies/location.owl#hasSport -->
```



```
<owl:ObjectProperty rdf:about="&location;hasSport">
    <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
</owl:ObjectProperty>

<!-- http://www.location.lk/ontologies/location.owl#isActivityOf -->

<owl:ObjectProperty rdf:about="&location;isActivityOf"/>
    <!--
///////////////////////////////////////////////////////////////////////////////////
//
// Data properties
//
///////////////////////////////////////////////////////////////////////////////////
 -->
<!-- http://www.location.lk/ontologies/location.owl#hasCoordinate -->
```

```
    <owl:DatatypeProperty rdf:about="&location;hasCoordinate">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    </owl:DatatypeProperty>

    <!-- http://www.location.lk/ontologies/location.owl#hasName -->

    <owl:DatatypeProperty rdf:about="&location;hasName"/>

    <!--
    ///////////////////////////////////////////////////////////////////////////////////
    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////////////////
     -->

    <!-- http://www.location.lk/ontologies/location.owl#Accomadation -->

    <owl:Class rdf:about="&location;Accomadation">
        <rdfs:subClassOf rdf:resource="&location;Location"/>
    </owl:Class>

    <!-- http://www.location.lk/ontologies/location.owl#Activity -->

    <owl:Class rdf:about="&location;Activity">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
        <rdfs:comment>An activities that tourists can carry out. These activites can be
available in tourist locations</rdfs:comment>
    </owl:Class>

    <!-- http://www.location.lk/ontologies/location.owl#Adventure -->
    <owl:Class rdf:about="&location;Adventure">
        <rdfs:subClassOf rdf:resource="&location;Sport"/>
```

```
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#AirSport -->

<owl:Class rdf:about="&location;AirSport">
   <rdfs:subClassOf rdf:resource="&location;SportActivity"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Airline -->

<owl:Class rdf:about="&location;Airline">
   <rdfs:subClassOf rdf:resource="&location;Transport"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Apartment -->
```



```
<owl:Class rdf:about="&location;Apartment">
   <rdfs:subClassOf rdf:resource="&location;Accomadation"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Aquatic -->

<owl:Class rdf:about="&location;Aquatic">
   <rdfs:subClassOf rdf:resource="&location;Sport"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Archeology -->

<owl:Class rdf:about="&location;Archeology">
   <rdfs:subClassOf rdf:resource="&location;Museum"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Architectural -->
```

```
    <owl:Class rdf:about="&location;Architectural">
        <rdfs:subClassOf rdf:resource="&location;Museum"/>
    </owl:Class>

    <!-- http://www.location.lk/ontologies/location.owl#Art -->

    <owl:Class rdf:about="&location;Art">
        <rdfs:subClassOf rdf:resource="&location;Traditional"/>
    </owl:Class>

    <!-- http://www.location.lk/ontologies/location.owl#ArtPerformance -->

    <owl:Class rdf:about="&location;ArtPerformance">
        <rdfs:subClassOf rdf:resource="&location;Happening"/>
    </owl:Class>

    <!-- http://www.location.lk/ontologies/location.owl#Attraction -->
```



```
    <owl:Class rdf:about="&location;Attraction">
        <rdfs:comment>Diffrent kind of attractions refers to tourist
locations</rdfs:comment>
    </owl:Class>

    <!-- http://www.location.lk/ontologies/location.owl#Beach -->

    <owl:Class rdf:about="&location;Beach">
        <rdfs:subClassOf rdf:resource="&location;NaturalLocation"/>
    </owl:Class>

    <!-- http://www.location.lk/ontologies/location.owl#Beauty -->

    <owl:Class rdf:about="&location;Beauty">
        <rdfs:subClassOf rdf:resource="&location;Attraction"/>
```

```
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#BookStore -->

<owl:Class rdf:about="&location;BookStore">
    <rdfs:subClassOf rdf:resource="&location;Shopping"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#BotanicalGarden -->

<owl:Class rdf:about="&location;BotanicalGarden">
    <rdfs:subClassOf rdf:resource="&location;NaturalLocation"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&location;hasAttraction"/>
            <owl:someValuesFrom rdf:resource="&location;Natural"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Bowling -->

<owl:Class rdf:about="&location;Bowling">
    <rdfs:subClassOf rdf:resource="&location;SportActivity"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#BusService -->

<owl:Class rdf:about="&location;BusService">
    <rdfs:subClassOf rdf:resource="&location;Transport"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Cafe -->
```

```
<owl:Class rdf:about="&location;Cafe">
   <rdfs:subClassOf rdf:resource="&location;Restaurant"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Casino -->

<owl:Class rdf:about="&location;Casino">
   <rdfs:subClassOf rdf:resource="&location;SportActivity"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Celebrations -->

<owl:Class rdf:about="&location;Celebrations">
   <rdfs:subClassOf rdf:resource="&location;Traditional"/>
</owl:Class>
```



```
<!-- http://www.location.lk/ontologies/location.owl#Chinese -->

<owl:Class rdf:about="&location;Chinese">
   <rdfs:subClassOf rdf:resource="&location;Food"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Church -->

<owl:Class rdf:about="&location;Church">
   <rdfs:subClassOf rdf:resource="&location;ReligiousLocation"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#CofeeShop -->

<owl:Class rdf:about="&location;CofeeShop">
   <rdfs:subClassOf rdf:resource="&location;Food"/>
```

```
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Concert -->

<owl:Class rdf:about="&location;Concert">
   <rdfs:subClassOf rdf:resource="&location;ArtPerformance"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Cultural -->

<owl:Class rdf:about="&location;Cultural">
   <rdfs:subClassOf rdf:resource="&location;Attraction"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Cycling -->
```



```
<owl:Class rdf:about="&location;Cycling">
   <rdfs:subClassOf rdf:resource="&location;SportActivity"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#DanceFestival -->

<owl:Class rdf:about="&location;DanceFestival">
   <rdfs:subClassOf rdf:resource="&location;Happening"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#DutyFreeShop -->

<owl:Class rdf:about="&location;DutyFreeShop">
   <rdfs:subClassOf rdf:resource="&location;Shopping"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Emergency -->
```

```
<owl:Class rdf:about="&location;Emergency">
    <rdfs:subClassOf rdf:resource="&location;Location"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Estate -->

<owl:Class rdf:about="&location;Estate">
    <rdfs:subClassOf rdf:resource="&location;NaturalLocation"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Exhibition -->

<owl:Class rdf:about="&location;Exhibition">
    <rdfs:subClassOf rdf:resource="&location;Happening"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#FiveStarHotel -->
```



```
<owl:Class rdf:about="&location;FiveStarHotel">
    <rdfs:subClassOf rdf:resource="&location;Hotel"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&location;hasAttraction"/>
            <owl:someValuesFrom rdf:resource="&location;Attraction"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&location;hasActivity"/>
            <owl:someValuesFrom rdf:resource="&location;Activity"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

```
<!-- http://www.location.lk/ontologies/location.owl#Food -->

<owl:Class rdf:about="&location;Food">
   <rdfs:subClassOf rdf:resource="&owl;Thing"/>
   <rdfs:comment>Food categories expected by tourists</rdfs:comment>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Forest -->

<owl:Class rdf:about="&location;Forest">
   <rdfs:subClassOf rdf:resource="&location;NaturalLocation"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#FoutStarHotel -->

<owl:Class rdf:about="&location;FoutStarHotel">
   <rdfs:subClassOf rdf:resource="&location;Hotel"/>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="&location;hasAttraction"/>
         <owl:someValuesFrom rdf:resource="&location;Attraction"/>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Golf -->

<owl:Class rdf:about="&location;Golf">
   <rdfs:subClassOf rdf:resource="&location;SportActivity"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#GuestHouse -->
```

```
<owl:Class rdf:about="&location;GuestHouse">
   <rdfs:subClassOf rdf:resource="&location;Accomadation"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Happening -->

<owl:Class rdf:about="&location;Happening">
   <rdfs:subClassOf rdf:resource="&location;Location"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Hiking -->

<owl:Class rdf:about="&location;Hiking">
   <rdfs:subClassOf rdf:resource="&location;SportActivity"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#HistorialMuseum -->

<owl:Class rdf:about="&location;HistorialMuseum">
   <rdfs:subClassOf rdf:resource="&location;Museum"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Historical -->

<owl:Class rdf:about="&location;Historical">
   <rdfs:subClassOf rdf:resource="&location;Attraction"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Hospital -->

<owl:Class rdf:about="&location;Hospital">
   <rdfs:subClassOf rdf:resource="&location;Emergency"/>
   <owl:disjointWith rdf:resource="&location;Police"/>
```

```
</owl:Class>
<!-- http://www.location.lk/ontologies/location.owl#Hotel -->

<owl:Class rdf:about="&location;Hotel">
   <rdfs:subClassOf rdf:resource="&location;Accomadation"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#ImportantLocation -->

<owl:Class rdf:about="&location;ImportantLocation">
   <rdfs:subClassOf rdf:resource="&location;Location"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Indoor -->

<owl:Class rdf:about="&location;Indoor">
   <rdfs:subClassOf rdf:resource="&location;Sport"/>
   <owl:disjointWith rdf:resource="&location;Outdoor"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#InternationalRestaurant -->

<owl:Class rdf:about="&location;InternationalRestaurant">
   <rdfs:subClassOf rdf:resource="&location;Restaurant"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Island -->

<owl:Class rdf:about="&location;Island">
   <rdfs:subClassOf rdf:resource="&location;NaturalLocation"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Japanese -->
```

```
    <owl:Class rdf:about="&location;Japanese">
        <rdfs:subClassOf rdf:resource="&location;Food"/>
    </owl:Class>

    <!-- http://www.location.lk/ontologies/location.owl#Korean -->

    <owl:Class rdf:about="&location;Korean">
        <rdfs:subClassOf rdf:resource="&location;Food"/>
    </owl:Class>

    <!-- http://www.location.lk/ontologies/location.owl#Location -->

    <owl:Class rdf:about="&location;Location">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
        <rdfs:comment>All the important locations that tourists can be
interested</rdfs:comment>
    </owl:Class>
```



```
    <!-- http://www.location.lk/ontologies/location.owl#MotorSport -->

    <owl:Class rdf:about="&location;MotorSport">
        <rdfs:subClassOf rdf:resource="&location;SportActivity"/>
    </owl:Class>

    <!-- http://www.location.lk/ontologies/location.owl#Mountain -->

    <owl:Class rdf:about="&location;Mountain">
        <rdfs:subClassOf rdf:resource="&location;NaturalLocation"/>
    </owl:Class>

    <!-- http://www.location.lk/ontologies/location.owl#Museum -->

    <owl:Class rdf:about="&location;Museum">
```

```
    <rdfs:subClassOf rdf:resource="&location;ImportantLocation"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#MusicFestival -->

<owl:Class rdf:about="&location;MusicFestival">
    <rdfs:subClassOf rdf:resource="&location;Happening"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#NationalPark -->

<owl:Class rdf:about="&location;NationalPark">
    <rdfs:subClassOf rdf:resource="&location;NaturalLocation"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&location;hasAttraction"/>
            <owl:someValuesFrom rdf:resource="&location;Natural"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Natural -->

<owl:Class rdf:about="&location;Natural">
    <rdfs:subClassOf rdf:resource="&location;Attraction"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#NaturalLocation -->

<owl:Class rdf:about="&location;NaturalLocation">
    <rdfs:subClassOf rdf:resource="&location;ImportantLocation"/>
</owl:Class>
```

```
<!-- http://www.location.lk/ontologies/location.owl#NaturalTourLocation -->

<owl:Class rdf:about="&location;NaturalTourLocation">
   <owl:equivalentClass>
      <owl:Class>
         <owl:intersectionOf rdf:parseType="Collection">
            <rdf:Description rdf:about="&location;Location"/>
            <owl:Restriction>
               <owl:onProperty rdf:resource="&location;hasAttraction"/>
               <owl:someValuesFrom rdf:resource="&location;Natural"/>
            </owl:Restriction>
         </owl:intersectionOf>
      </owl:Class>
   </owl:equivalentClass>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#NatureMuseum -->
```



```
<owl:Class rdf:about="&location;NatureMuseum">
   <rdfs:subClassOf rdf:resource="&location;Museum"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#NightLife -->

<owl:Class rdf:about="&location;NightLife">
   <rdfs:subClassOf rdf:resource="&location;Activity"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#NonAquatic -->

<owl:Class rdf:about="&location;NonAquatic">
   <rdfs:subClassOf rdf:resource="&location;Sport"/>
</owl:Class>
```

```
<!-- http://www.location.lk/ontologies/location.owl#OlympicGames -->

<owl:Class rdf:about="&location;OlympicGames">
   <rdfs:subClassOf rdf:resource="&location;SportEvents"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#OneStarHotel -->

<owl:Class rdf:about="&location;OneStarHotel">
   <rdfs:subClassOf rdf:resource="&location;Hotel"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Opera -->

<owl:Class rdf:about="&location;Opera">
   <rdfs:subClassOf rdf:resource="&location;ArtPerformance"/>
</owl:Class>
```



```
<!-- http://www.location.lk/ontologies/location.owl#Outdoor -->

<owl:Class rdf:about="&location;Outdoor">
   <rdfs:subClassOf rdf:resource="&location;Sport"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Phamacy -->

<owl:Class rdf:about="&location;Phamacy">
   <rdfs:subClassOf rdf:resource="&location;Emergency"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Police -->

<owl:Class rdf:about="&location;Police">
```

```
    <rdfs:subClassOf rdf:resource="&location;Emergency"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Rafting -->

<owl:Class rdf:about="&location;Rafting">
    <rdfs:subClassOf rdf:resource="&location;SportActivity"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#RawFish -->

<owl:Class rdf:about="&location;RawFish">
    <rdfs:subClassOf rdf:resource="&location;Food"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Relaxation -->
```



```
<owl:Class rdf:about="&location;Relaxation">
    <rdfs:subClassOf rdf:resource="&location;Activity"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Religious -->

<owl:Class rdf:about="&location;Religious">
    <rdfs:subClassOf rdf:resource="&location;Attraction"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#ReligiousLocation -->

<owl:Class rdf:about="&location;ReligiousLocation">
    <rdfs:subClassOf rdf:resource="&location;ImportantLocation"/>
</owl:Class>
```

<!-- http://www.location.lk/ontologies/location.owl#Restaurant -->

<owl:Class rdf:about="&location;Restaurant">
  <rdfs:subClassOf rdf:resource="&location;Location"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#River -->

<owl:Class rdf:about="&location;River">
  <rdfs:subClassOf rdf:resource="&location;NaturalLocation"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Sailing -->

<owl:Class rdf:about="&location;Sailing">
  <rdfs:subClassOf rdf:resource="&location;SportActivity"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Shopping -->

<owl:Class rdf:about="&location;Shopping">
  <rdfs:subClassOf rdf:resource="&location;Location"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#ShoppingComplex -->

<owl:Class rdf:about="&location;ShoppingComplex">
  <rdfs:subClassOf rdf:resource="&location;Shopping"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Sightseeing -->

<owl:Class rdf:about="&location;Sightseeing">

```
    <rdfs:subClassOf rdf:resource="&location;Activity"/>
  </owl:Class>

  <!-- http://www.location.lk/ontologies/location.owl#SouvenirShop -->

  <owl:Class rdf:about="&location;SouvenirShop">
    <rdfs:subClassOf rdf:resource="&location;Shopping"/>
  </owl:Class>
  <!-- http://www.location.lk/ontologies/location.owl#Sport -->

  <owl:Class rdf:about="&location;Sport">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:comment>Different categories of sport a tourist can play in tour
locations</rdfs:comment>
  </owl:Class>

  <!-- http://www.location.lk/ontologies/location.owl#SportActivity -->
```

```
  <owl:Class rdf:about="&location;SportActivity">
    <rdfs:subClassOf rdf:resource="&location;Activity"/>
  </owl:Class>

  <!-- http://www.location.lk/ontologies/location.owl#SportEvents -->

  <owl:Class rdf:about="&location;SportEvents">
    <rdfs:subClassOf rdf:resource="&location;Happening"/>
  </owl:Class>

  <!-- http://www.location.lk/ontologies/location.owl#SriLankan -->

  <owl:Class rdf:about="&location;SriLankan">
    <rdfs:subClassOf rdf:resource="&location;Food"/>
  </owl:Class>
```

<!-- http://www.location.lk/ontologies/location.owl#Surfing -->

<owl:Class rdf:about="&location;Surfing">
  <rdfs:subClassOf rdf:resource="&location;SportActivity"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#TaxiService -->

<owl:Class rdf:about="&location;TaxiService">
  <rdfs:subClassOf rdf:resource="&location;Transport"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Temple -->

<owl:Class rdf:about="&location;Temple">
  <rdfs:subClassOf rdf:resource="&location;ReligiousLocation"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#TheatrePerformance -->

<owl:Class rdf:about="&location;TheatrePerformance">
  <rdfs:subClassOf rdf:resource="&location;ArtPerformance"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#ThreeStarHotel -->

<owl:Class rdf:about="&location;ThreeStarHotel">
  <rdfs:subClassOf rdf:resource="&location;Hotel"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#TouristHotel -->

<owl:Class rdf:about="&location;TouristHotel">

```
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <rdf:Description rdf:about="&location;Hotel"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="&location;hasAttraction"/>
            <owl:someValuesFrom rdf:resource="&location;Attraction"/>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#TouristLocation -->

<owl:Class rdf:about="&location;TouristLocation">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <rdf:Description rdf:about="&location;Location"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="&location;hasAttraction"/>
            <owl:someValuesFrom rdf:resource="&location;Attraction"/>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Tournament -->

<owl:Class rdf:about="&location;Tournament">
    <rdfs:subClassOf rdf:resource="&location;SportEvents"/>
```

```
</owl:Class>
<!-- http://www.location.lk/ontologies/location.owl#Traditional -->

<owl:Class rdf:about="&location;Traditional">
    <rdfs:subClassOf rdf:resource="&location;Cultural"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#TraditionalCelebration -->

<owl:Class rdf:about="&location;TraditionalCelebration">
    <rdfs:subClassOf rdf:resource="&location;Happening"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#TraditionalMarket -->

<owl:Class rdf:about="&location;TraditionalMarket">
    <rdfs:subClassOf rdf:resource="&location;Shopping"/>
</owl:Class>
```



```
<!-- http://www.location.lk/ontologies/location.owl#TraditionalRestaurant -->

<owl:Class rdf:about="&location;TraditionalRestaurant">
    <rdfs:subClassOf rdf:resource="&location;Restaurant"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Train -->

<owl:Class rdf:about="&location;Train">
    <rdfs:subClassOf rdf:resource="&location;Transport"/>
</owl:Class>
```

<!-- http://www.location.lk/ontologies/location.owl#Transport -->

<owl:Class rdf:about="&location;Transport">
  <rdfs:subClassOf rdf:resource="&location;Location"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#TwoStarHotel -->

<owl:Class rdf:about="&location;TwoStarHotel">
  <rdfs:subClassOf rdf:resource="&location;Hotel"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#UnderWater -->

<owl:Class rdf:about="&location;UnderWater">
  <rdfs:subClassOf rdf:resource="&location;SportActivity"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Waterfall -->

<owl:Class rdf:about="&location;Waterfall">
  <rdfs:subClassOf rdf:resource="&location;NaturalLocation"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#Waterskiing -->

<owl:Class rdf:about="&location;Waterskiing">
  <rdfs:subClassOf rdf:resource="&location;SportActivity"/>
</owl:Class>

<!-- http://www.location.lk/ontologies/location.owl#ZooPark -->

<owl:Class rdf:about="&location;ZooPark">

```
    <rdfs:subClassOf rdf:resource="&location;NaturalLocation"/>
</owl:Class>

<!-- http://www.w3.org/2002/07/owl#Thing -->

<rdf:Description rdf:about="&owl;Thing">
    <rdfs:comment>Describes a tourism ontology.</rdfs:comment>
</rdf:Description>

<!--
///////////////////////////////////////////////////////////////////////////////////
//
// Individuals
//
///////////////////////////////////////////////////////////////////////////////////
 -->
```



```
<!-- http://www.location.lk/ontologies/location.owl#AmayaLakeDambulla -->

<owl:NamedIndividual rdf:about="&location;AmayaLakeDambulla">
    <rdf:type rdf:resource="&location;FoutStarHotel"/>
    <location:hasCloseLocation rdf:resource="&location;Anuradhapura"/>
    <location:hasCloseLocation rdf:resource="&location;DambullaCaveTemple"/>
    <location:hasCloseLocation rdf:resource="&location;Polonnaruwa"/>
    <location:hasCloseLocation rdf:resource="&location;Sigiriya"/>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#AmayaLeasurePark -->

<owl:NamedIndividual rdf:about="&location;AmayaLeasurePark">
    <rdf:type rdf:resource="&location;Location"/>
    <location:hasActivity rdf:resource="&location;BirdWatching"/>
    <location:hasActivity rdf:resource="&location;Safari"/>
```

```
    <location:hasActivity rdf:resource="&location;TableTennis"/>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#Anuradhapura -->

<owl:NamedIndividual rdf:about="&location;Anuradhapura">
    <rdf:type rdf:resource="&location;Location"/>
    <rdf:type>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&location;hasAttraction"/>
            <owl:someValuesFrom rdf:resource="&location;Religious"/>
        </owl:Restriction>
    </rdf:type>
    <rdf:type>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&location;hasAttraction"/>
            <owl:someValuesFrom rdf:resource="&location;Historical"/>
        </owl:Restriction>
    </rdf:type>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#AraliyaHotel -->

<owl:NamedIndividual rdf:about="&location;AraliyaHotel">
    <rdf:type rdf:resource="&location;Hotel"/>
    <location:hasActivity rdf:resource="&location;BirdWatching"/>
    <location:hasActivity rdf:resource="&location;Pool"/>
    <location:hasActivity rdf:resource="&location;TableTennis"/>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#BirdWatching -->

<owl:NamedIndividual rdf:about="&location;BirdWatching">
```

```
    <rdf:type rdf:resource="&location;Activity"/>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#CinnamonGrand -->

<owl:NamedIndividual rdf:about="&location;CinnamonGrand">
    <rdf:type rdf:resource="&location;FiveStarHotel"/>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#Clubbing -->

<owl:NamedIndividual rdf:about="&location;Clubbing">
    <rdf:type rdf:resource="&location;NightLife"/>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#DambullaCaveTemple -->
```



```
<owl:NamedIndividual rdf:about="&location;DambullaCaveTemple">
    <rdf:type rdf:resource="&location;Temple"/>
    <rdf:type>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&location;hasAttraction"/>
            <owl:someValuesFrom rdf:resource="&location;Historical"/>
        </owl:Restriction>
    </rdf:type>
    <rdf:type>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&location;hasAttraction"/>
            <owl:someValuesFrom rdf:resource="&location;Natural"/>
        </owl:Restriction>
    </rdf:type>
    <rdf:type>
        <owl:Restriction>
```

```
            <owl:onProperty rdf:resource="&location;hasAttraction"/>
            <owl:someValuesFrom rdf:resource="&location;Religious"/>
        </owl:Restriction>
    </rdf:type>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#ElephantWatching -->

<owl:NamedIndividual rdf:about="&location;ElephantWatching">
    <rdf:type rdf:resource="&location;Activity"/>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#GaladariHotel -->

<owl:NamedIndividual rdf:about="&location;GaladariHotel">
    <rdf:type rdf:resource="&location;FiveStarHotel"/>
</owl:NamedIndividual>
```



```
<!-- http://www.location.lk/ontologies/location.owl#HeritageDambulla -->

<owl:NamedIndividual rdf:about="&location;HeritageDambulla">
    <rdf:type rdf:resource="&location;ThreeStarHotel"/>
    <location:hasActivity rdf:resource="&location;BirdWatching"/>
    <location:hasActivity rdf:resource="&location;TableTennis"/>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#HeritanceKandalama -->

<owl:NamedIndividual rdf:about="&location;HeritanceKandalama">
    <rdf:type rdf:resource="&location;FiveStarHotel"/>
    <location:hasCoordinate rdf:datatype="&xsd;string">
7.86667,80.7167</location:hasCoordinate>
    <location:hasCloseLocation rdf:resource="&location;Anuradhapura"/>
```

```
    <location:hasActivity rdf:resource="&location;BirdWatching"/>
    <location:hasCloseLocation rdf:resource="&location;DambullaCaveTemple"/>
    <location:hasCloseLocation rdf:resource="&location;MinneriyaNationalPark"/>
    <location:hasCloseLocation rdf:resource="&location;Polonnaruwa"/>
    <location:hasSport rdf:resource="&location;Pool"/>
    <location:hasActivity rdf:resource="&location;Safari"/>
    <location:hasCloseLocation rdf:resource="&location;Sigiriya"/>
    <location:hasSport rdf:resource="&location;TableTennis"/>
    <location:hasActivity rdf:resource="&location;Yoga"/>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#HiltonColombo -->

<owl:NamedIndividual rdf:about="&location;HiltonColombo">
    <rdf:type rdf:resource="&location;FiveStarHotel"/>
    <location:hasActivity rdf:resource="&location;Yoga"/>
</owl:NamedIndividual>
```



```
<!-- http://www.location.lk/ontologies/location.owl#KirindaBeach -->

<owl:NamedIndividual rdf:about="&location;KirindaBeach">
    <rdf:type rdf:resource="&location;Beach"/>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#KirindaBeachHotel -->

<owl:NamedIndividual rdf:about="&location;KirindaBeachHotel">
    <rdf:type rdf:resource="&location;FiveStarHotel"/>
    <location:hasActivity rdf:resource="&location;BirdWatching"/>
    <location:hasCloseLocation rdf:resource="&location;KirindaBeach"/>
    <location:hasActivity rdf:resource="&location;SunBathing"/>
    <location:hasCloseLocation rdf:resource="&location;Udawalawe"/>
    <location:hasCloseLocation rdf:resource="&location;YalaNationalPark"/>
```

```
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#LakeLodge -->

<owl:NamedIndividual rdf:about="&location;LakeLodge">
    <rdf:type rdf:resource="&location;FoutStarHotel"/>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#MinneriyaNationalPark -->

<owl:NamedIndividual rdf:about="&location;MinneriyaNationalPark">
    <rdf:type rdf:resource="&location;NationalPark"/>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#Polonnaruwa -->
```



```
<owl:NamedIndividual rdf:about="&location;Polonnaruwa">
    <rdf:type rdf:resource="&location;Location"/>
    <rdf:type>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&location;hasAttraction"/>
            <owl:someValuesFrom rdf:resource="&location;Historical"/>
        </owl:Restriction>
    </rdf:type>
    <rdf:type>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&location;hasAttraction"/>
            <owl:someValuesFrom rdf:resource="&location;Religious"/>
        </owl:Restriction>
    </rdf:type>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#Pool -->
```

```
<owl:NamedIndividual rdf:about="&location;Pool">
   <rdf:type rdf:resource="&location;Indoor"/>
   <rdf:type rdf:resource="&location;NonAquatic"/>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#Safari -->

<owl:NamedIndividual rdf:about="&location;Safari">
   <rdf:type rdf:resource="&location;Sightseeing"/>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#Sigiriya -->

<owl:NamedIndividual rdf:about="&location;Sigiriya">
   <rdf:type rdf:resource="&location;Location"/>
   <rdf:type>
      <owl:Restriction>
         <owl:onProperty rdf:resource="&location;hasAttraction"/>
         <owl:someValuesFrom rdf:resource="&location;Historical"/>
      </owl:Restriction>
   </rdf:type>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#SunBathing -->

<owl:NamedIndividual rdf:about="&location;SunBathing">
   <rdf:type rdf:resource="&location;Relaxation"/>
</owl:NamedIndividual>

<!-- http://www.location.lk/ontologies/location.owl#TableTennis -->

<owl:NamedIndividual rdf:about="&location;TableTennis">
   <rdf:type rdf:resource="&location;Indoor"/>
```

```
        <rdf:type rdf:resource="&location;NonAquatic"/>
    </owl:NamedIndividual>

    <!-- http://www.location.lk/ontologies/location.owl#Udawalawe -->

    <owl:NamedIndividual rdf:about="&location;Udawalawe">
        <rdf:type rdf:resource="&location;NaturalTourLocation"/>
    </owl:NamedIndividual>

    <!-- http://www.location.lk/ontologies/location.owl#YalaNationalPark -->

    <owl:NamedIndividual rdf:about="&location;YalaNationalPark">
        <rdf:type rdf:resource="&location;NationalPark"/>
        <location:hasActivity rdf:resource="&location;BirdWatching"/>
        <location:hasActivity rdf:resource="&location;Safari"/>
    </owl:NamedIndividual>
```

```
    <!-- http://www.location.lk/ontologies/location.owl#Yoga -->

    <owl:NamedIndividual rdf:about="&location;Yoga">
        <rdf:type rdf:resource="&location;Relaxation"/>
    </owl:NamedIndividual>
</rdf:RDF>
```