Arab American University – Jenin

Faculty of Graduate Studies

Exploring Heritage Locations Based on Multi-Agent Systems

By

Shafaq Jaber Mahmoud Abulhouf

Supervisor

Dr. Amjad RATTROUT

This thesis was submitted in partial fulfillment of the requirements for the Master`s degree in Computer Science

August, 2016

2

Exploring Heritage Locations Based on Multi-agent Systems

By

Shafaq Jaber Mahmoud Abulhouf

This thesis was defended successfully on 21/9/2016 and approved by:

Committee members	Signature
1. Supervisor Name	.....
2. Co- Supervisor Name	.....
3. Internal Examiner Name Dr. Muath Sabha	..... muath Sabha
4. External Examiner Name Pr. Khaldoun ZREIK	.....

Acknowledgements

First of all I thank God the Almighty for sustaining me until the completion of this work.

I would like to take this opportunity to express my gratitude to the following people:

My academic supervisor Dr. Amjad Rattrout who has provided valuable advice, guidance and feedback during my studies.

The Citu Lab head Khaldoon Zrik, Dr. Nasreddine Bouhai, and all the team members there who welcomed me during two months training at Paris 8 university, and provided valuable support, and shared their knowledge with me.

My Family for their continuous support, encouragement and patience through the years of my studies.

My friends: Safa, Zaynab, Shorouq, Kefaya, Wahiba, Abeer, Rudina and all my colleagues who have been an endless source of help and encouragement.

Without the support and assistance of these people I would not have been able to complete this thesis.

Abstract

This thesis presents a forward step in modeling digital heritage system from Multi-Agent System (MAS) perspectives in the heritage locations. Furthermore, it presents the relations and interactions between the main actors; visitors, location and content in the digital heritage domain. It focuses on delivering the content to the visitors at the right time, appropriate method and the changing behavior of the visitor.

Our approach considers the cultural heritage as a field of application, where visitors of heritage locations can explore information in real-time and contribute in providing content to enhance the available information. In this work we deal with heritage locations environment as a complex adaptive system (CAS). The complexity of such systems raises from the diversity of the visitor's communities and their dynamic behavior inside the location. In addition, in this environment the set of visitors changes dynamically over time and many users may be viewing the same Point of Interest (PoI) at the same time. Finally the complexity expands due to the process of delivering the right content to these visitors according to their individual characteristics and preferences, in addition to the changes in their behaviors.

We use Multi-Agent System (MAS) as a tool to reduce the complexity in the environment, and we propose a model which is composed of a set of agents that interact and collaborate to 1) Create visitor's profile and track his dynamic behavior in the physical space of the heritage location. 2) Plan a personalized tour according to the visitor's profile. 3) Detect and solve conflicts in the planned tours in real time. 4) Simplify the visitor's interaction between the physical environment (heritage location) and the virtual system (application).

We build a prototype based on JDAE platform which demonstrates our model with proposed agents and the interactions between them. The process of choosing the tours, adapt the changes, and deliver information has also been considered.

Table of Content

List of Tables

List of Figures

List of Abbreviations

Abbreviations	Explanations
MAS	Multi-Agent System
CAS	Complex Adaptive System
JADE	Java Agent Development Framework
PoI	Point of Interest
BDI	Belief–Desire–Intention Model
FIPA	Foundation for Intelligent Physical Agents
ACL	Agent Communication Language

1. Chapter One: Introduction

This chapter consists of two sections. The first section describes the motivation, problem statement, and the contribution of our thesis. The second section presents the structure of the thesis.

1.1. Motivation

Personalizing is highly demanded in different domains to deliver the right information at the right time [1]. In heritage locations the creation of a personalized experience is affected by several aspects such as the diversity of visitors' communities and the diversity of the available content.

Nowadays, many available technologies support the creation of personalized visits. The localization techniques, smartphones, and the development of planning algorithms could be combined together to help the visitors to get the best visit.

Our thesis primarily aims to view the heritage locations as a Complex Adaptive System (CAS) and to model the relation between the main actors; location, visitors, and content using Multi-Agent System (MAS). We propose a model to enrich the visitor's experience, and improve the management inside the locations. This is achieved through the creation of a mobile application that can track the visitors inside the building, and provide a step-by-step information.

In addition to the main goal, there are different objectives to achieve:

Objective 1: create a user profile that contains the visitor's basic characteristics and the interactions of the visitor inside the location.

Objective 2: create a user interface for the application that matches the visitor's profile.

Objective 3: determine the visitor's location in real time in order to retrieve information according to his location.

Objective 4: provide the visitor with a set of tours; free, recommended and planned.

Objective 5: adapt real time changes in the proposed tour path such as changes in the visitor's time and the location constraints.

1.2. Research Problem

Cultural heritage plays an important role in regional development. It provides economical support to communities and contributes in social development through educating about the past and enhancing social cohesion and regional identity [2]. According to UNESCO heritage is defined as "our legacy from the past, what we live with today and what we pass on to future generations" [3]. It includes tangible heritage (artifacts, monuments, historical buildings), and intangible heritage (folklore, events, traditions).

Palestine has long history and rich cultural heritage, but much of this heritage endangered due to environmental degradation, political and socio-economic pressures [4]. Consequently, the documentation, preservation and digitization of cultural heritage have to be recognized as a vital strategy for its sustainability.

Currently, there are several solutions for publishing and managing heritage content. These solutions aim to deliver textual and multimedia content for the public and to deliver contextual, real time information to the end users. Developing such applications face multiple problems such as the lack of digitalized heritage content, the low degree of interactivity and the static delivery of content.

As mentioned earlier in this thesis we study the relations and interactions between the main actors; visitors, location and content in the digital heritage domain.

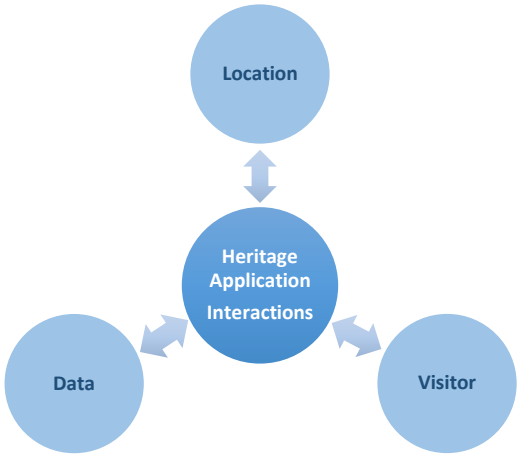

Figure 1 Heritage Application Main Actors

When modeling the relation between these actors we consider the following challenges: First, visitors come from different cultural backgrounds and that implies that they differ in their language, interests and behavior inside the location.

Second, planning the tour path is affected by several factors: I) the visit duration for the location which limits the number of selected objects in the tour. II) the visitor's interest that defines the type of the selected objects. III) the system constraints such as the closing time, the closed spaces, and the maximum visitor capacity for each PoI. IV) real time changes that happen during the planned tour such as changing the plan according to the visitor's time and the location constraints.

Third, for the free visits, we consider the dynamic behavior of the visitors as they do not follow a constant plan inside the location and move randomly from one place to another, requesting information about different PoIs.

1.3. Contributions

In this thesis we achieved several key contributions:

We primarily propose a multi-agent framework that is considered the base to model the interaction between the location, data and visitors. This framework is composed of a set of agents: i) a visitor agent which represents the visitor and is responsible for initiating and updating his profile. ii) a position agent to track the visitor's position inside the location. iii) a wiki agent which is responsible for searching the database and giving back the results. iv) a planner agent to select a personalized path that matches the visitor's time and interest. v) a route agent that presents the content and cooperates with all the other agents to deliver the final results. vi) an avatar agent to present the content through an animated character. vii) a capacity agent to monitor the location capacity conditions.

We use agent technology as a tool to:

- Reduce the complexity in the heritage location's environment.
- Support the design of real-time systems.
- Improve the system flexibility. The system is able to run on multiple platforms within different environments.
- Personalize the user's experience by adapting services that meet their knowledge and preferences.

In addition, we present a system model that is capable of adapting visitors' behaviors, time and location constraints to provide dynamic propositions.

While visiting the CiTu Lab in Paris 8 university, I enriched my experience, knowledge and gained in-depth understanding of planning the visitor's routes in heritage locations. To

model these routes, we used the dijkstra algorithm to find the shortest path between the PoIs that matches the visitor interests, objects and time. We also took into consideration different attributes such as the time per PoI, the speed of the visitor, and the location constraints to track the visitor in the location.

Finally, we have published our proposed framework "Modeling Interactive Multi-Agent System in Cultural Heritage Locations" in the third Hyperheritage International Seminar (HIS.3), 4-7 May 2016, Tunis. International Conference. [5]

1.4. Methodology

In this thesis we follow four main phases: in the first phase, the research problem is specified and a study of the existing literature is performed on the use of Multi-Agent System (MAS) in the cultural heritage domain and on the planning of personalized visits to heritage locations. We also studied the cultural locations in Palestine and we considered "St. George's" Church which is located in Burqin (South West of Jenin-Palestine) as a case study. In the second phase, a study of the interactions between the main actors in the system is completed. Then the main drivers of the complexity are defined and the proposed model is presented. In the third phase, the main building blocks of the Multi-Agent System (MAS); agents, interactions, roles and organization are demonstrated and the agent's main algorithms are also introduced. Furthermore, a prototype which demonstrates our model with the proposed agents and the interactions between them is developed based on JADE platform and using netbeans IDE 8.1. Finally, the behavior of the individual agents, the diversity between them and their scalability are simulated using Netlogo.

1.5. Document Structure

This thesis is represented in six chapters and a conclusion. In the first chapter we list the motivation, problem statement and the main contributions.

In chapter two we describe the state of art where the cultural heritage is identified, the development of the digital heritage is reviewed, and the most related work is listed. It also details the Multi-Agent System (MAS); its definition, characteristics, types and agent frameworks. Finally it presents the most used indoor positioning techniques.

The third chapter is composed of two sections: the first section illustrates the model architecture and defines the main agents and their main objectives. The second views cultural heritage locations environment as a Complex Adaptive System and models it using Multi-Agent System.

The fourth chapter describes the agents from four main perspectives: agents' organization, interactions, definitions and roles.

The fifth chapter presents the scenarios that describe the flow of the events in the system and illustrates the main UML diagrams of the application. It finally describes the prototype of the model.

The sixth chapter illustrates the prototype implementation parts; tools, codes and results. Additionally it presents the agent simulation phase using Netlogo simulator to illustrate visitors' behaviors in the environment.

Finally, the conclusion and future work chapter.

2. Chapter Two: State of Art

This chapter introduces the field of cultural heritage and details some of the techniques and developments that have been made since its creation. In addition, it lists the most related work in the field of personalizing the interactions with cultural heritage locations. We also describe the Multi-Agent System (MAS); its definition, characteristics and agent frameworks. We finally describe the most used indoor positioning techniques.

2.1. Cultural Heritage

Cultural heritage plays a significant role in regional development. It provides economical support to communities, and contributes to social development through education of the past, and it also enhances social cohesion and regional identity [2]. According to UNESCO [3] heritage is defined as "our legacy from the past, what we live with today and what we pass on to future generations", and it includes tangible heritage (artifacts, monuments, historical buildings), and intangible heritage (folklore, events, traditions) [6].

In Palestine much of our cultural heritage is threatened due to environmental, political and socio-economic pressures [4]. The documentation and preservation of cultural heritage has to be recognized as a vital strategy for its sustainability. In addition to traditional protection of cultural heritage, digitization of cultural heritage is a new form of preservation which is suitable to the digital society we live in today.

Recently, many significant research tend to adopt technology trends to cultural heritage domain. According to Radu Comes et al [7], the creation and preservation of digital cultural

heritage includes three main perspectives: the three dimensional digitization, data storage and data management.

The first effort in digital cultural heritage domain started in the 90s, and was limited to one way content delivery in which applications provide text, multimedia presentation and audio tours for museums [8].

Following the development of interactive computer technologies such as virtual and augmented reality, cultural heritage applications started to adapt these changes to provide virtual tourism and virtual museum exhibits. Virtual reality provides visual representation of buildings, artifacts to be available for the public [9]. Many applications and research follow this technology. Foni et al [10] proposed a system to establish a complete real-time interactive digital narrative visualization of the Hagia Sophia in Istanbul.

The augmenting reality system is a combination of both the real and virtual worlds. It creates another dimension in which physical and digital objects interact together. The emergence of this technology gives the chance to enrich the presentation of cultural heritage content inside the heritage site [11]. Aguilo et al [12] proposed the augmented reality system that allows visitors to see real objects and reconstructions of the past at the same time. Other efforts such as PRISMA project [13], ARCO [14] follow the same technology.

At the level of data management, semantic technologies offer great potential for dealing with cultural heritage complex domain. Semantic technologies help on data acquisition, collaboration and data consumption to provide more interesting and personalized content and context for end users [15]. Other efforts propose interactive wikis that combine

ontology concepts with wiki templates, and these systems allow the participation of users on providing content.

The use of the Multi-Agent System (MAS) is also addressed in the cultural heritage domain to support the visitors' interactions in both physical and virtual environment. We shed more light on these efforts in the following section.

2.2. Related Work

In cultural heritage domain there are several solutions for publishing and managing heritage content. The available applications aim to deliver textual, multimedia, contextual and real time content to the end users. This section lists the most related work.

Alba Amato et al. in [16] present a framework to support cultural heritage experts to augment the archaeological sites. They provide a set of applications and contents to be delivered to the visitors to provide tour guidance and to reinforce their knowledge of the environment.

They also propose an agent framework to model the dynamic environment in which the visitor is moving, and which is rich of artworks, and monuments. Authors introduced two agents based on Belief–Desire–Intention (BDI) agent model. The first is a student agent which is running on the mobile device and it discovers the surrounding objects and uses them to provide digital information that are relevant to the context. The second is a teacher agent which returns the selection of contents that best fit the user's needs and preferences.

An agent-based tourist guiding system "iJADE FreeWalker" proposed by Toby Lam in [17] is developed based on JADE platform and integrates Ontology, FOM (Fuzzy Ontology

Map), agent technology and GPS technology. A GPS system is used to provide position and geographic information and intelligent agents are used to enhance the functionality of the guidance system. Additionally, fuzzy ontology is adapted to provide recommendations to users according to their interest.

Inmaculada Ayala et al. [18] proposed a mobile-based system "iMuseumA" which is mainly composed of three applications; the first provides visitors with customized visits and guide them in the museum. The second targets the museum guides to help create and control routes, and communicate with both visitors and museum staff. The third application is for the museum staff to allow them interact with users and monitor the museum's rooms. iMuseumA follows an agent-based approach which introduces intelligence to the system as agents are used to sense real time data from the environment and use them in decision making and negotiation process between visitor agents and guided agents.

Nazaraf Shah, et al in [19] presents a multi agent system to manage a virtual educational framework that supports the access to a variety of cultural heritage resources. This addition came as an enhancement to (MOSAICA) project, which is a collaborative web portal for sharing and managing cultural resources in a globally distributed environment.
Authors build their work on the use of intelligent agents to provide an intelligent virtual learning environment. The proposed model follows the Belief–Desire–Intention (BDI) agent model and consists of two agents; user agent and virtual expedition agent (VEA). The user agent is used to manage user's interactions with VEA during a virtual expedition by

obtaining the user's input and sending the requests to the VEA. The VEA in turn is responsible for managing the interactions between the users and the web resources.

Tsvi Kuflik, et al in [20] propose an infrastructure for visitor's guides in museums. They introduce PEACH project and suggest a generic architecture of Active Museums.

PEACH (Personal experience with Active Cultural Heritage) project provides the visitor of the museum with presentations and information relevant to the objects with respect to the visited physical location.

PEACH architecture is mainly composed of three main components; the first is the user assistant entity which represents the visitor and interacts with the system on his behalf, the second is the presentation composer which builds the appropriate presentations to the user, finally the Information mediator that provides the presentation composer with the needed information to generate the presentations.

It developed as a Multi-Agent System to model the dynamic environment in which users interacts dynamically the services availability cannot be guaranteed. Furthermore to provide flexibility by separating the application from the agent's communication infrastructure, facilitate the process of adapting new types of services.

They developed two prototypes based on PEACH architecture, the first one was developed for the Buonconsiglio Castle in Trento and the second for Hecht museum. These prototypes extended to provide more services, the spatial information broker and the position agent were added to manage the user position information and send it to the presentation composer. In addition the user modeling agent was added to support the personalized presentation.

The authors proposed a general scalable generic agent architecture for active museums visitor's guides. In this architecture each service is provided by implicit organization of agents, these organizations works according to different polices to reach their goals, they either negotiate, compete, or collaborate. The main idea behind this proposition is to provide flexibility to add new services by adding relevant agents and roles.

The previously mentioned works focus on delivering content to the visitors of the heritage locations based on the Multi-Agent System (MAS) which helps modeling such dynamic environments.

The following related works are concerned with the planning of visits. We noticed that the efforts in the field of trip planning concentrate on the planning of tours in regions which contain multiple heritage locations. These plans do not take into consideration that inside each location the visitor has to spend a specific time, and may not be interested in all the objects and regions.

Some efforts targeted the planning of visits inside single locations. X. Claude et al in [21] proposed a plan to enrich the experience of the museum visitors through guiding them to exhibits that match their interest and thus minimize the visitor's walking time.

They model the physical location, as well as the visitors' characteristics and behaviors. The physical space "museum" is modeled as a set of exhibits that has a maximum crowd capacity, position and distance between exhibits. When modeling the visitors they consider different characteristics and visiting styles. They categorize the visitors into four different

visiting styles according to their behavior patterns inside the museum, these styles are represented using animal metaphors: ant, butterfly, fish and grasshopper.

Ivo Roes in his master thesis [22] proposed a mobile guide that rely on three main factors to generate recommended tours inside museums. These factors are 1) the visitor's interests 2) real time location of the visitor 3) the semantic relation that exists between the artworks. The main contribution in this work is the real time adaptation of the tour according to the available time, user preference, and spatial information. To generate the recommendations, the user can set the time of the visit, choose the number of the artworks and the system keeps tracking the visitor’s location during the tour.

We share characteristics with some of the previous systems and applications such as the use of mobile devices to provide visitors with tours in museums and cultural locations. On the other hand there are some differences as follows: First, iMuseum [18] focuses on group guidance and the negotiation between agents to select the best tour from a set of predefined tours while we focus on creating a tour plan for each individual visitor according to his profile, visit duration and the location constraints. Some projects are targeting the multimedia content such as [15]. Second, in [22] the process of choosing artworks involves visitor intervention and focuses on adapting the route according to the change in their interest through rating the artworks. In contrast, we track the visitor during the tour to adapt changes in time and location conditions to provide an adaptable and real time proposition.

2.3. Multi-Agent Systems

An agent is a computer program that exists and operates in an environment, and can be considered to act autonomously on behalf of an individual or organization. An agent will receive data from its environment and react in an appropriate action [23].

A Multi-Agent system (MAS) consists of a number of agents, which interact with one another to accumulate specific actions. These Agents can work and act on behalf of its users, so it's capable of intelligent behavior [24].

2.3.1. Agent characteristics

Franklin and graesser in [25] summarize the characteristics of agents as follow:

- Autonomy: Agents control their own actions, and can operate without the direct control or involvement of humans or other agents.
- Social Ability: Agents can communicate with both agents and humans through a standard agent communication language.
- Reactivity: Agents should respond to real time changes that occur in the environment.
- Mobility: Agents are able to move between machines and different platforms.
- Pro-activeness: the role of the agents is not limited to reacting to the environment rather they direct their behavior to achieve some goals.
- Learning: Agents change their behavior based on their previous experience, and they increase their efficiency through time.

2.3.2. Agent Types

Hyacinth Nwana in [26] classified the different types of agents along several attributes including autonomy, learning and cooperation. Autonomy means that agents can operate on behalf of the system users. Cooperation property indicates that agents need to communicate with both agents and humans to fulfill shared goals. Finally agents learn as they interact with their environment by adjusting their behaviors.

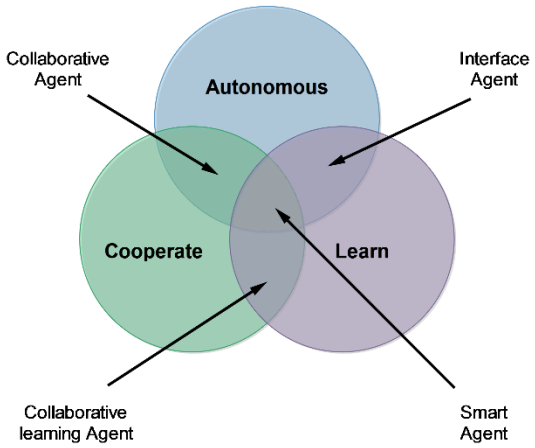

Figure 2 Agent Types [26]

1. Collaborative Agents: agents of this type have two characteristics; the autonomy and the cooperation, they cooperate with other agents to perform goals on behalf of their owners. Generally this type of agents used to solve large problems that cannot performed by one single agent. Additionally they provide solutions to distributed problems as they enhance modularity, speed, reliability, and flexibility.
2. Interface Agents: agents of this type have two characteristics; the autonomy and the learning, they support and assist users autonomously when interact with the application. Mainly these agents assist the end user when interacting with the system. They also observe and monitor the actions taken by the user in the interface

so that they can learn from the user's behavior therefor they adapt to its user's preferences.

3. Mobile Agents: are agents that have the capability to move from one machine to another, interact with foreign hosts, gather information from different sources and perform tasks on other machines and return results to the original machine. These agents are used first, to reduce communication costs. Second, to run programs on external machine when the local machine has limited processing power and storage. Third, to simplify the coordination between a number of remote and independent requests. Finally to adapt parallel computing.
4. Information Agents: this type of agents is responsible for gathering information from distributed sources and reporting what they retrieve to the user. These agents could be mobile or local, they can cooperate and learn.
5. Reactive Agents: this type of agents precepts the surrounding environment and take actions according to the real time parameters. They take simple actions depending on the current state and without previous plans. Generally reactive agents offer fault tolerance, flexibility and adaptability of the system.
6. Hybrid Agents: this type of agents presents a combination of characteristics of two or more types of agents such as the mobile, collaboration and information agents. Primarily the idea behind the use of hybrid agents is to combine the advantages of each type of agents in one agent.
7. Heterogeneous Agents Systems: these systems composed of a set of agents belong to different types, the reason behind developing such systems is to integrate several

services with each other to provide enhanced service, in addition to make use of previously developed software instead of go for writing software from scratch.

2.3.3. Agent Frameworks

Different frameworks and programming languages are appropriate for the development of Multi-agent System (MAS) such as JADE, JATLite, SoFAR, DESIRE and others. These frameworks mainly aim to enable the creation of agents, and the agent communication and coordination.

We chose to work on JADE (Java Agent Development Framework) which is a software framework to develop distributed agent-based applications which facilitate the construction of the Multi-Agent System following the Foundation for Intelligent Physical Agents (FIPA) standard [27] . The platform JADE can work on PCs, servers, and there are add-ons to work on smartphones, tablets such as Jade-leap.

JADE offers a graphical user interface to facilitate managing several agents and multi-agents platforms it also provides other features like agent mobility, ontologies, and fault tolerance.

The communication between agents is done through exchanging messages which follow the Agent Communication Language (ACL) format defined by the FIPA international standard.

2.4. Localization technologies

In our work we aim to localize the visitor in order to deliver contextual information about the surrounding environment. Different technologies are used for indoor positioning like

the RFID, Wi-Fi, Bluetooth and others. Each of these technologies has different capability, accuracy and limit.

1-GPS: set of satellites that send precise details of their positions in space back to earth. It uses trilateration algorithm to find the exact position which calculates the time the signal takes to travel from the satellite to the GPS receiver. By calculating the travel time of the signal from three different satellites and considering their position, we can get the three dimensions; east, north, and altitude [28]. The GPS is suitable for the outdoor, but does not work in indoor because the signal strength inside is affected by the walls of the building [29].

2-Bluetooth: localization can be done using beacons, which are small devices placed in the location and send out continuous signals via Bluetooth. The sent signals can be received by any nearby Bluetooth enabled device such as smart phones. It uses the Received Signal Strength (RSS) to detect the distance between the device antenna and the beacon [29]. This method of localization covers small range up to 30 meters with accuracy up to one meter [30].

3-RFID: Radio Frequency Identifier which is a technology used to identify objects by reading the tags associated to these objects. RFID uses the electromagnetic fields to store and retrieve data. The RFID system is composed of the tags and the reader. Tags are categorized into Passive and active. Active tags are attached with a battery to provide the power to perform the communications. In contrast, passive tags are not attached with battery, instead they use the power of the RFID reader to perform the communications [29].

4- Wi-Fi: Wi-Fi technology is widely used for indoor location identification. The availability of several Wi-Fi access points facilitates the process of indoor

positioning wherever a Wi-Fi is enabled [30]. Identifying a smartphone location is done either by using received signal strength indication (RSSI) method or by wireless fingerprinting method. RSSI is the most common method that depends on the signal detected by the phone from a nearby access point. It uses a database that identifies the position of each known access point and to get the location of the device by calculating how far is a device from the access point. Wireless fingerprinting identifies a phone location by recording the detected signal strength from several access points and storing the information of the place in profiles within the database [31].

Conclusion

In this chapter we have discussed the term of cultural heritage, and the development phases in the field of the digital heritage. We have also presented the Multi-Agent System; its definition, characteristics and agent frameworks.

Furthermore, we have listed the most related work in the field of personalizing the interaction with cultural heritage locations.

3. Chapter Three: Our Approach

This chapter presents our approach to create an intelligent application for cultural heritage locations. In the first section we illustrate the proposed model architecture, its functions, components and technologies and justify our methodologies. In the second section we discuss the environment complexity and its characteristics in addition to the properties and mechanisms of complex adaptive system according to Holland [32].

3.1. Model Architecture

In this thesis we achieved our goal in enriching the visitor's experience inside the heritage locations by modeling the relations and interactions between the main actors; visitors, location and content. We introduce an architecture illustrated in Figure 3. This architecture has been published in [5], and it inherits a Multi-Agent System (MAS) features to deliver the content to the visitors at the right time and appropriate method and it adapts visitors' behaviors and system constraints.

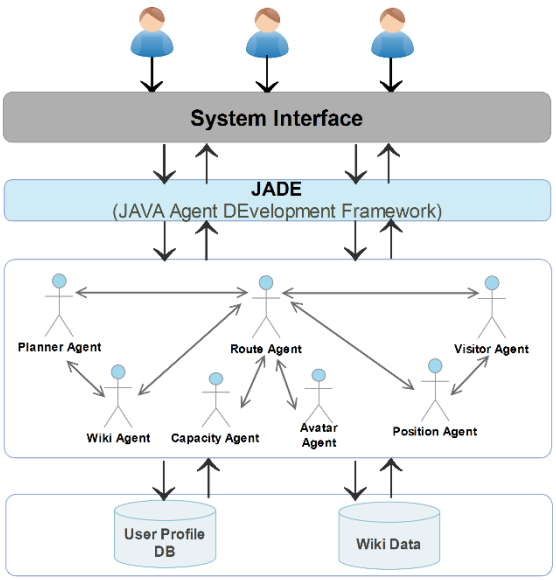

Figure 3 Model Architecture

Our proposed architecture presents the interaction between the visitors and the system in one hand and the relation between the system and the data model on the other hand. System interface enables the interactions between the users and system services. It responds to the user's requests, obtains his information, and displays the content.

Multi-Agent System (MAS) is composed of agents, an environment and the interactions between them which form the organizations. Agents are computer programs that exist in an environment, and can act autonomously on behalf of an individual or an organization [23].

We use a MAS to reduce the complexity as an approved tool in this domain. It can also handle real time changes and it is capable of decision making on behalf of the system users. It also distributes the control of the system among the agents which leads to speed up the system's performance.

The core of the system is composed of a set of agents that are developed using JADE (Java Agent Development Framework).

Our application complements and provides a way of interaction between the visitors and the wiki model implemented in [33].Two agents are responsible for the interactions with the data model which is composed of a wiki database to store all types of content about the location, and a user profile database that contains their basic information, interests and interactions.

The system follows the client server architecture in which the client side is an application on the visitor's mobile device and works as input/output terminal, to present the content, issue requests, and provide visitor's position. The server side contains the main system components that provide the main system functions. It is composed of the agents and the databases. This approach is suitable for our architecture since we need the application to be

light to meet the computing power of any mobile device. Additionally, in typical mobile applications, the user should update the application constantly to receive any new updates. Our system's environment is composed of two main parts; physical which represents the physical location and the virtual which is the application. In general it is a dynamic and real time environment, which has the following characteristics:

- System content is continuously growing and changing.
- Many users may be viewing the same PoI at the same time.
- The set of users changes dynamically over time.
- Users can interact with the application to get the main services.
- Services are provided by a set of agents that can join and leave the environment, and can operate anywhere.
- In this environment the users move continuously, so agents should dynamically locate the users in order to know what they are looking at.
- In the physical space each PoI has a limited capacity and this defines the number of visitors viewing it.
- A set of constraints in the physical location controls and affects the system behavior such as the closing time and the closed spaces of the location.

3.2. Environment Complexity

The originality of this work comes from dealing with heritage locations environment as a Complex Adaptive System (CAS) and modeling it using Multi-Agent System (MAS). The complexity of such systems raises from different factors: First, the diversity of the visitors' communities as they differ in their language, age, and interest. Second, in this environment the set of visitors changes dynamically over time and multiple users may be in a single place viewing the same item simultaneously.

Finally, the complexity expands due to the process of delivering the right content to these visitors according to their dynamic behavior inside the location. Visitors may follow a plan or move randomly from one place to another, requesting information about different PoIs.

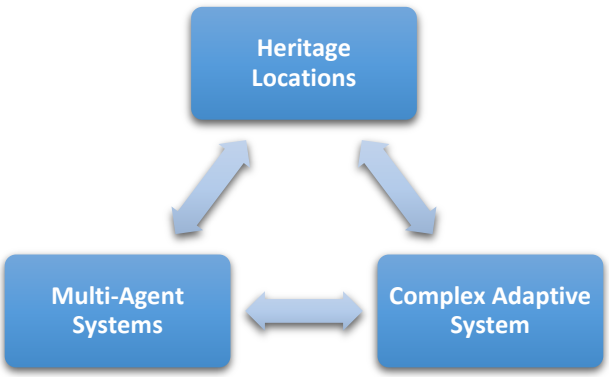

Figure 4 Complexity Model

The French Roadmap for Complex Systems in [34] defined the complex system as "any system comprised of a great number of heterogeneous entities, among which local interactions create multiple levels of collective structure and organization." The word complex implies that the system's inter-related parts are in a continual change due to the

dynamic processes and interactions inside it [35]. We call it adaptive if the system copes with the changes to fit the environment.

Based on the complex system presented in [36], we present the main characteristics that define the complexity in our system:

1-Emergent behavior: the behavior of the whole system is formed of the behavior of the single components and the interaction between them. In the heritage system the interaction between the main components (visitors, data, and location) shape the system's behavior, and any change in any single component affects and reflects on the general behavior of the system.

2-Self-organization: there is no centralized entity to control the system's behavior. Each agent behaves according to its own rules, and the perception from the environment. The agents act independently to select the objects that both keep the tour in time and match the interest of each visitor.

3-Adaptation: agents adapt the changing environment, by changing their behavior. In our system, adaptability includes, i) the increase and decrease in the number of agents according to the number of visitors and the amount of available content. ii) the change in agent's behaviors according to parameters like the position of each single visitor. iii) the cope with the dynamic behavior of the visitor. For example, if the visitor skips one of the objects in the path, the system will adjust and recalculate the path according to the new parameters. iv) the adjustment of the path according to the physical capacity of the PoI and the closing time of the location... etc.

4-Co-evolution: the system evolves as a result of the adaptation property.

John Holland [32] identified seven basic properties and mechanisms of a CAS as shown in figure 5.

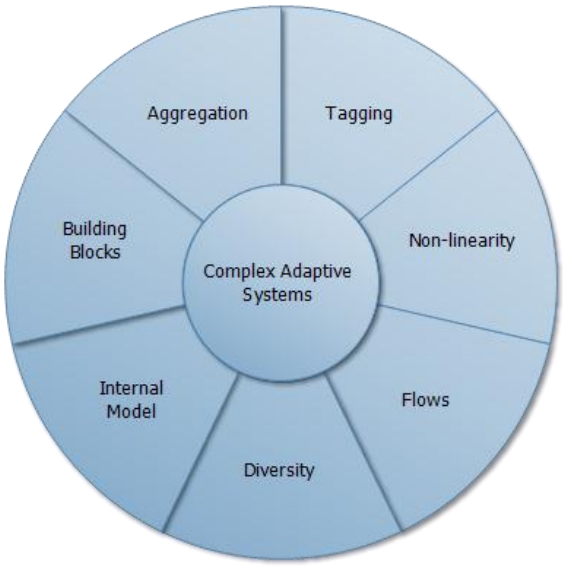

Figure 5 CAS Elements

We use the previous properties and mechanisms to describe the complexity in our system:

1-Aggregation: the property in which agents are organized into categories and then each category is treated in the same way. This formulates higher-level groups and leads to more complexity. In the heritage system we group visitors who have the same age, language, interest and position and aggregate data related to specific topic and PoIs.

2-Tagging: is the mechanism in which each agent is identified with specific attribute to facilitate the grouping and categorizing of agents and smooth the interactions between agents. In our case, the tag refers to the type of the agent. This simplifies the process of grouping agents from the same type, and the interactions between different types of agents.

3-Non-linearity: the behavior of the whole system resulted from the interaction between the aggregated agents is much more than the behavior resulted from the sum of all the simple

agents. In the case of heritage system, the non-linearity results from the dynamic interaction between the system parts.

4-Flows: are the physical resources that circulate through a network of agents. In this system, agents exchange information about the PoIs and the visitors to form the paths of the visit.

5-Diversity: is the property which states that the agents' population in CAS diverse in their rules, skills, and strategies. This diversity is due to their different functions inside the environment and the diversity of each single entity they are dealing with (visitor's communities, data and locations)

6-Internal Model: represents the rules and regulations that each agent uses to interact with other agents and with his environment. In this work we define the roles for each agent in chapter 4.

7-Building blocks: the main building blocks of the system are the agents. We propose seven types of agent populations (visitor, route, position, wiki, planner, avatar and capacity). These populations deal mainly with three entities (visitor communities, the physical location of each visitor, and the available data about that location).

Conclusion

In this chapter we have presented our approach to model the interactions of the main actors in the cultural heritage domain. In the second section we have discussed the environment complexity and its characteristics in addition to the properties and mechanisms of complex adaptive system according to Holland.

4. Chapter Four: Approach Modeling using Multi-Agent System

In this chapter we state the main agents in our system. In the first section, we describe the agent's organization and the main characteristics that form the agent's environment in our system. In the second section, we define each proposed agent according to a set of parameters: space, operation, tag, life cycle and the number of agents. We also explain the role of agents which includes their skills, responsibilities and permissions. Finally, we illustrate the main interactions between the agents following the agent's behaviors.

4.1. Agent organization

An agent organization is a group of agents that accomplish several distinct tasks or functions, and have their own roles and capabilities [37]. Organizations arise in order to achieve specific objectives, and these objectives are achieved via the interactions between the real environment and the virtual (agent) environment in addition to the interactions between agents themselves.

The main characteristics of the agent organizations in our model are:

1- Agents interact with the surrounding environment and adapt the changes.

2- Agents have a set of tasks to do.

3- Agents are capable of interacting with each other in order to provide the system services.

4- The creation and termination of agents is done dynamically and automatically depending on the number of users and the required services.

Figure 6 MAS Organization

4.2. Agents Definitions

In this section we describe the definitions of agents based on [38], and in order to identify each agent we need to specify the following parameters:

Agent :< Space(S), Operation (O), Tag (T), Life (L), Number (N)>/<Type>

- Space: is the environment in which the agents operate and perform their tasks.
- Operation (O): represents the agents' basic task or function within the group. Each agent in the group handles a specific role.
- Life (L): defines the lifetime of the agent identified by millisecond, <time (t), energy (e)>/<Page> it takes into consideration the linear time and the energy for each agent which depends on the value of the information the agent carries from the resource.
- Tag (T): <Agent (A), Value (V), Page (P)> / </Type: word> identifies the agent by one kind of information. Tags are used in the aggregation process.
- Page (P): is the set of wiki pages the agent searches to get the needed information.

- Number (N): indicates the number of agents, which depends on the number of users, and the number of available resources.

Our proposed architecture includes seven populations of agents. Each agent is defined as the following:

A. Visitor Agent:

Visitor agent can be defined as *the agent that is responsible for creating the user profile. It stores visitor's basic information such as: name, age, language and interest. This agent also tracks visitor's interactions and any interaction with the user profile database is done through this agent.*

Let the visitor agent space (S_v) is the space for all the visitor's agents access the system. Suppose $A_v^i$is the visitor agent i, as $A_v^i G_v^i$ where G_v^i is the set of visitor's agents in a specific group works in the same spaceS_v. This agent is tagged with type (T_v) with a specific Operation (O_v) to handle visitor's request and initiate user's profile for (t) time for his Life (L_v).

Agent: < Space (S_v), Operation (O_v), Tag (T_v), Life (L_v), Number (N_v)>/<Type>

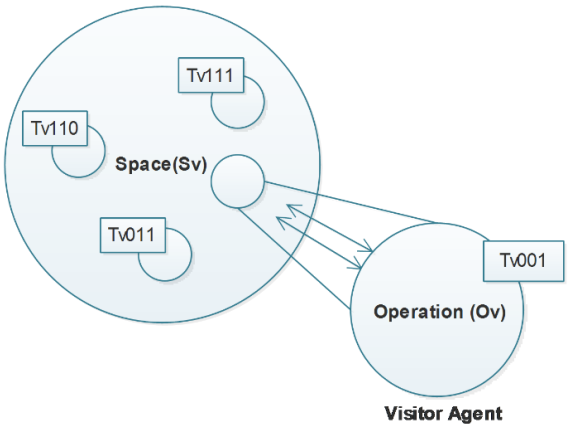

Figure 7 Visitor Agent Definition

B. Route Agent:

Route agent is responsible for the actual tour. It tracks the visitor during the tour to insure that the visitor follows the plan and interacts with other agents to accomplish the goal.

Let the Route agent space (S_r) is the space for all the route's agents initiated in the system. Suppose $A_r^i$is the route agent i, as $A_r^i G_r^i$ where G_r^i is the set of route's agents in a specific group works in the same spaceS_r. This agent is tagged with type (T_r) with a specific Operation (O_r) to guide the user for (t) time for his Life (L_r).

Agent :< Space (S_r), Operation (O_r), Tag (T_r), Life (L_r), Number (N_r)>/<Type>

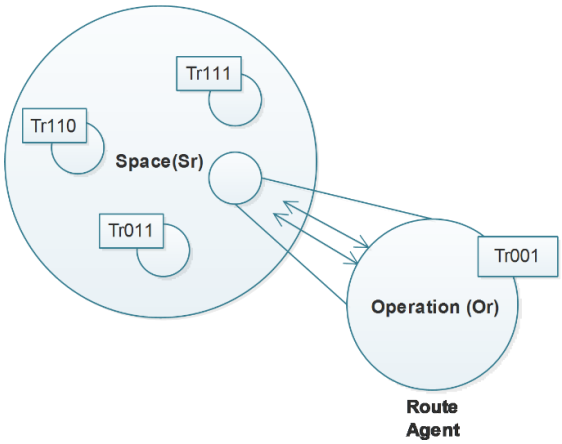

Figure 8 Route Agent Definition

C. Wiki Agent:

Wiki agent provides the search results obtained from searching the wiki with the position value received from the route agent. Other agents cannot issue queries to the database without contacting this agent.

Let the wiki agent space (S_w) is the space for all the wiki's agents initiated by the system. Suppose $A_w^i$is the wiki agent i, as $A_w^i G_w^i$ where G_w^i is the set of wiki's agents in a specific group works in the same space S_w. This agent is tagged with type (T_w) with a specific Operation (O_w) to handle the search request and return best results for (t) time for his Life (L_w).

Agent :< Space (S_w), Operation (O_w), Tag (T_w), Life (L_w), Number (N_w)>/<Type>

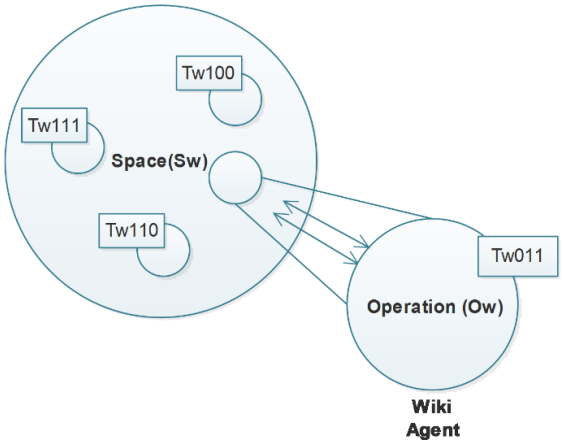

Figure 9 Wiki Agent Definition

D. Position Agent:

Position agent is responsible for detecting the visitor location using the available technologies such as the GPS, Wi-Fi and Near Field Communication (NFC) technology.

Let the position agent space (S_p) is the space for all the position's agents initiated by the system. Suppose $A_p^i$is the position agent i, as $A_p^i G_p^i$ where G_p^i is the set of position's agents in a specific group works in the same spaceS_p. This agent is tagged

with type (T_p) with a specific Operation (O_p) to identify the visitor position for (t) time of his Life (L_p).

Agent :< Space (S_p), Operation (O_p), Tag (T_p), Life (L_p), Number (N_p)>/<Type>

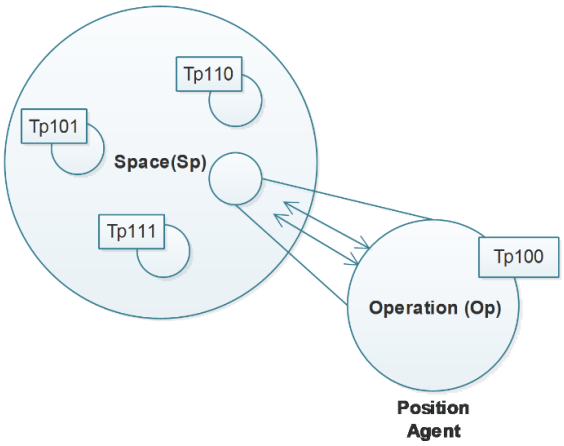

Figure 10 Position Agent Definition

E. Planner Agent:

The agent is responsible for creating a suitable tour path that matches the visitor's interest and the available time. It considers the shortest path between the points of interest to effectively use the available time.

Let the planner agent space (S_{pl}) is the space for all the planner's agents initiated in the system. Suppose $A_{pl}^{i}$is the planner agent i, as $A_{pl}^{i}G_{pl}^{i}$ where G_{pl}^{i} is the set of planner's agents in a specific group works in the same spaceS_{pl}. This agent is tagged with type (T_{pl}) with a specific Operation (O_{pl}) to plan the visit path for (t) time of his Life (L_{pl}).

Agent :< Space (S_{pl}), Operation (O_{pl}), Tag (T_{pl}), Life (L_{pl}), Number (N_{pl})>/<Type>

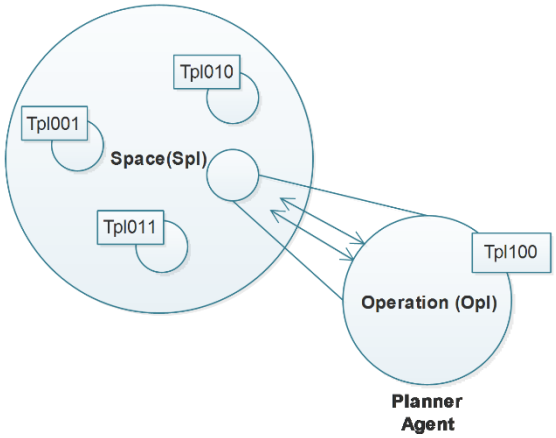

Figure 11 Planner Agent Definition

F. Capacity Agent:

Capacity agent handles the task of monitoring the maximum visitor capacity of each point of interest and reports the maximum capacity to the route agent.

Let the capacity agent space (S_C) is the space for all the capacity agents initiated in the system. Suppose $A_C^i$is the capacity agent i, as $A_C^i G_C^i$where G_C^i is the set of capacity agents in a specific group works in the same spaceS_C. This agent is tagged with type (T_C) with a specific Operation (O_C) to monitor the maximum space capacity for (t) time of his Life (L_C).

Agent :< Space (S_C), Operation (O_C), Tag (T_C), Life (L_C), Number (N_C)>/<Type>

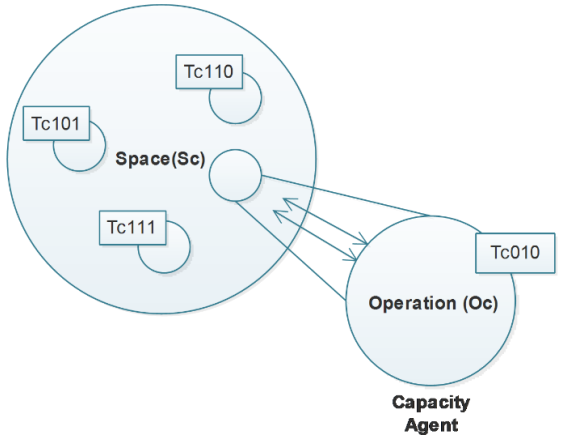

Figure 12 Capacity Agent Definition

G. Avatar Agent:

The agent that is responsible for presenting the results and information to the visitor, this agent takes into consideration the mobile device capabilities, and present the content as a text, image or video, or it use the avatar "character".

Let the capacity agent space (S_{AV}) is the space for all the avatar's agents initiated in the system. Suppose $A_{AV}^{i}$is the avatar agent i, as $A_{AV}^{i}G_{AV}^{i}$where G_{AV}^{i} is the set of avatar's agents in a specific group works in the same spaceS_{AV}. This agent is tagged with type (T_{AV}) with a specific Operation (O_{AV}) to monitor space capacity for (t) time of his Life (L_{AV}).

Agent :< Space (S_{AV}), Operation (O_{AV}), Tag (T_{AV}), Life (L_{AV}), Number (N_{AV})>/<Type>

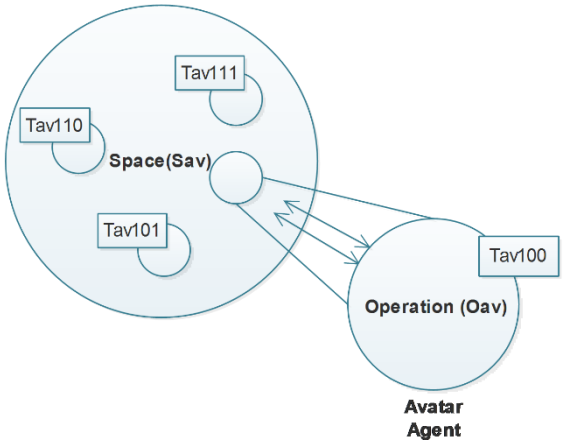

Figure 13Avatar Agent Definition

4.3. The Agent's Roles

The role model identifies the basic skills that are required by the system. It is mainly defined by responsibilities, permissions, activities and protocols [39]. The System Agents have the following major roles:

A. Visitor Agent:

Table 1 Visitor agent role

Role:	**VisitorAgent (VA)**
Description:	This agent issues queries on the wiki database to retrieve data about the location and the PoIs. It also inserts new data proposed by the visitors.
Protocols and Activities:	Initialize user profile, update user profile, store user interactions
Permissions:	Write to user database, read user data, send/receive messages to/from position and route agents.

Responsibilitie:	**Liveness:** VisitorAgent= (InitializeUserProfile, UpdateUserProfile, RespondToRequest, SendData) **Safety:** Successfully interact with both route agent and planner agent.

B. Route Agent:

Table 2 Route agent role

Role:	**RouteAgent (AV)**
Description:	This agent tracks the visitor during the route to insure that the visitor follows the plan. It receives the location from the position agent and requests the content from the wiki agent. It also requests the planner agent to generate plans.
Protocols and Activities:	Receive position report and send request to wiki agent to get information. Send updates to the visitor profile. Receive capacity report and handle it. Monitor the tour plan.
Permissions:	Send/receive messages to/from visitor, position, planner, capacity and wiki agents.
Responsibilitie:	**Liveness:** AvatarAgent= (ReceivePosition, ReceiveCapacityReport , SendSearchRequest, AlterVisitorAgent, FollowPlan) **Safety:** Retrieve the right information in the right time.

C. Wiki Agent:

Table 3 Wiki agent role

Role:	**WikiSAgent(WSA)**
Description:	It provides the search results obtained from searching the wiki according to the position value received from the route agent. Other agents cannot issue queries to the database without contacting this agent. It also inserts new data proposed by the visitors.
Protocols and Activities:	Receive search request from the route agent. Respond to the request.
Permissions:	Reads wiki content, send/receive messages to/from avatar agent.
Responsibilities:	**Liveness:** WikiSAgent= (ReceiveRequest, SearchWiki, GetResults, SendResults) **Safety:** A successful connection to the database.

D. Position Agent

Table 4 Position agent role

Role:	**PositionAgent(PA)**
Description:	This agent reports the visitor location to the route agent to be used to retrieve the content according to the visitor's current location. It uses the available technologies such as the GPS, Wi-Fi and Near Field Communication (NFC) technology.

Protocols and Activities:	Report position periodically
Permissions:	Report Position, send/receive messages to/from route agent.
Responsibilitie:	**Liveness:** PositionAgent = (ReceiveRequest, Report Position) **Safety:** Successfully get accurate position.

E. Planner Agent:

Table 5 Planner agent role

Role:	**PlannerAgent(PlA)**
Description:	This agent is responsible for creating a suitable tour path that matches the visitor's interest and the available time. It considers the shortest path between the PoIs to effectively use the available time.
Protocols and Activities:	Request visitor interest. Request location information (tags, positions) Calculate weights of interest Choose suitable path that matches available time.
Permissions:	Send/receive messages to/from visitor, route and wiki gents.
Responsibilitie:	**Liveness:** PlannerAgent = (ReceiveRequest, CreatePlan) **Safety:** Successfully propose a plan that matches visitor interest and time.

F. Capacity Agent:

Table 6 Capacity agent role

Role:	**CapacityAgent(CA)**
Description:	This agent is responsible for monitoring the physical capacity of each PoI, and when the PoI reaches its maximum capacity, this agent informs all the route agents to take this into consideration.
Protocols and Activities:	Report Capacity status to route agent.
Permissions:	Send/receive messages to/from route agent.
Responsibilitie:	**Liveness:** Capacity Agent = (Report Capacity) **Safety:** Successfully report accurate data.

G. Avatar Agent:

Table 7 Avatar agent Role

Role:	**AvatarAgent(AVA)**
Description:	The agent that is responsible for presenting the results and information to the visitor. This agent takes into consideration the device capabilities, and present the content as a text, image or video, or it users the avatar "character".
Protocols and Activities:	Present Content.
Permissions:	Send/receive messages to/from route agent.

Responsibilities :	**Liveness:** Avatar Agent = (Get_Content) **Safety:** Successfully present data.

4.4. Agent Interactions

Agents are known as social entities that can interact with each other in order to fulfill common objectives. Agents should have the ability to coordinate, cooperate and negotiate in order to interact successfully [40].

Agents cooperate when their goals are compatible, and compete when their goals are incompatible, so agents compete to deliver services. They attempt to collaborate when they have a shared objective to reach and their individual capabilities cannot fulfill the whole objective.

Interactions of agents enable the establishment of virtual organizations, in which groups of agents work together to achieve overall objectives [41]. The main aim of the interactions in our model is collaboration where each agent has a specific role, and interacts with others to reach a shared objective. Agents exchange information via messages and these messages have to follow a specific format.

In this section we illustrate the interactions between the agents and the flow of the interactions will be presented in the next chapter.

1- Position-Route

Purpose: report visitor location periodically

Initiator: position agent

Response: route agent

Input: send visitor location

Output: acknowledgment

Processing: provide visitor location

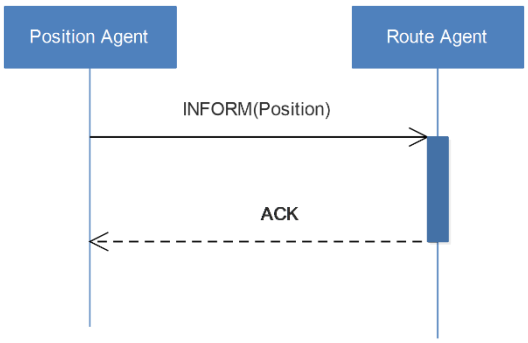

Figure 14 Position- Route Interaction

2- Capacity-Route

Purpose: report space capacity conflict.

Initiator: capacity agent

Response: route agent

Input: send the PoI id.

Output: acknowledgment

Processing: check space capacity.

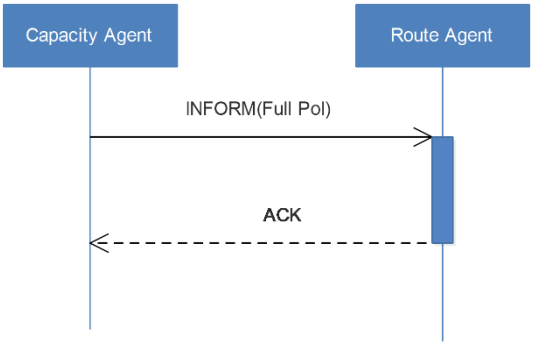

Figure 15 Capacity-Route Interaction

3- Route-Wiki

Purpose: request information about the PoI.

Initiator: route agent

Response: wiki agent

Input: PoI location

Output: content related to the requested PoI.

Processing: get visitor position and request content according to this value.

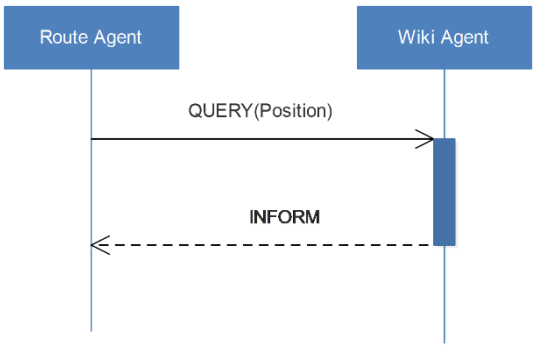

Figure 16 Route-Wiki Interaction

4- Route-Planner

Purpose: call for a new plan to adapt changes happened during the route activity.

Initiator: route agent.

Response: planner agent.

Input: time, remaining PoI.

Output: new plan.

Processing: calculate the remaining time, and the unvisited PoI.

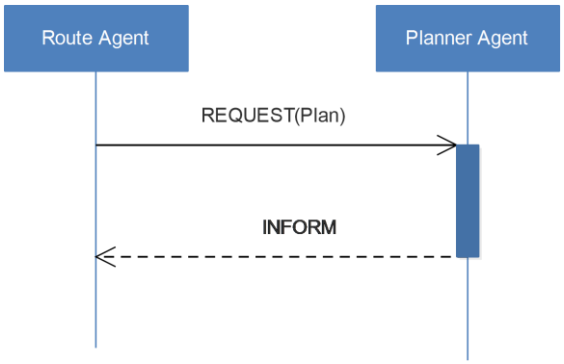

Figure 17 Route-Planner Interaction

5- Planner-Wiki

Purpose: request information about the location and the PoIs positions.

Initiator: planner agent

Response: wiki agent

Input: -

Output: PoIs, tags related to each PoI.

Processing: match PoI tags with visitor interest. Calculate shortest path

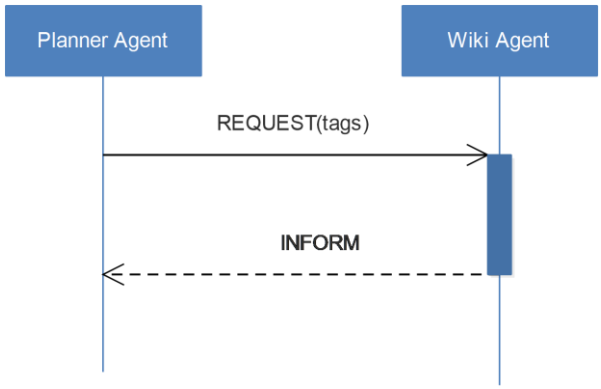

Figure 18 Planner-Wiki Interaction

6- Planner-Visitor

Purpose: create plans according to visitor interest and time

Initiator: planner agent

Response: visitor agent

Input: visitor id.

Output: set of visitor interests

Processing: match PoI tags with visitor interest.

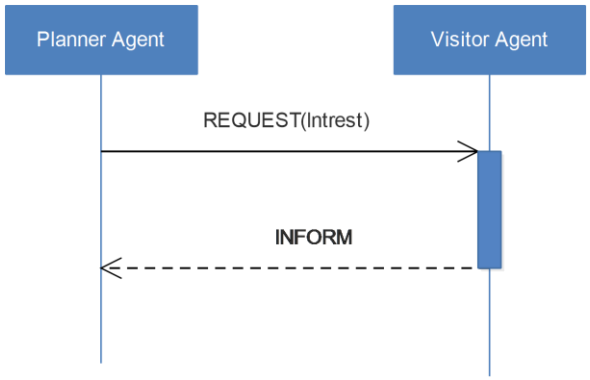

Figure 19 Planner-Visitor Interaction

7- Route-visitor

Purpose: update visitor profile according to visitor interaction

Initiator: route agent

Response: visitor agent

Input: interaction parameters

Output: acknowledgement

Processing: analyze interaction parameters and send it to visitor agent.

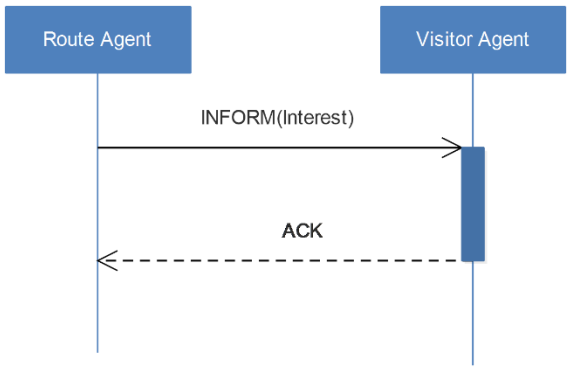

Figure 20 Route-Visitor Interaction

Conclusion

In this chapter we have presented the main building blocks of the MAS which are the organization, agents, roles, and interactions. We have also defined these blocks in terms of our model by illustrating the main agents and their roles which form the general behavior and the organization of the system. The definition of the agents, roles and interactions are modeled in the next chapter.

5. Chapter Five: Software Modeling

In this chapter we introduce the scenarios that describe the flow of the events in the proposed system and we illustrate the main UML diagrams of the application including the use case, class and sequence diagrams. We finally present our model prototype that demonstrates the proposed agents, their behaviors and the interactions between them.

5.1. Goals and Objectives

The main goal is to create a mobile application that is capable of combining the available information about the location with the visitor's preferences, position, and time to provide the visitor with the best tour plan.

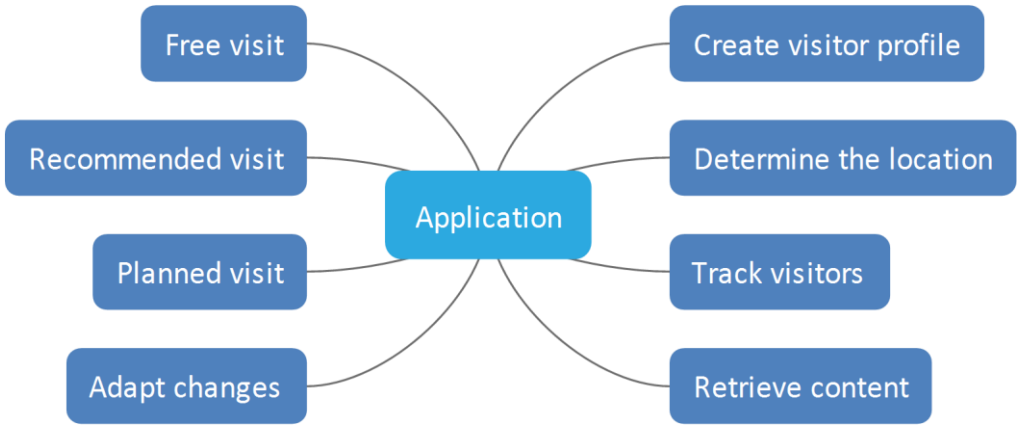

Figure 21 System Objectives

In addition to the main goal, there are different objectives to achieve:

Objective 1: initiate/update the visitor's profile and store the basic information about the visitor including age, language, interest… etc.

Objective 2: determine the visitor's location in real time in order to retrieve information according to his location.

Objective 3: track and store the visitors' interactions with the system to be able to follow their behavior and thus enforce rearrangements in the physical location based on the visitor's interactions.

Objective 4: provide the visitor with a set of tours; free/ recommended/ planned tours.

Objective 5: adapt real time changes that happen during the planned tour such as changing the plan according to the space capacity for the PoI, the visitor's time and the location constraints.

5.2. Usage Scenarios

The availability of mobile devices support the delivery of information to the visitors of cultural locations. In our system we use the visitors' mobile devices to provide contextual and personalized information. To do so, we have three scenarios; one is for the planned visit, the second is for the normal visit and the third is for the recommended visit.

Scenario 1: Normal visit

Sami is interested in visiting old churches around the world. In each visit he tries to get the maximum knowledge about the place he is visiting. To achieve this, he installs our application to use as a guide. He fills his basic information such as the name, age, and interests. Now he has to choose whether to follow a recommended plan, a tailored plan or to be free in the location. He decides to choose the normal/free visit because he has enough

time and wants to get more information. Once he presses the normal visit, a map of the location will be presented to him providing the main PoIs.

When he makes his choice, the system will search for the available information about it and retrieve the most relevant information to the visitor's interest.

- Agents interaction in the normal visit:

Once Sami logs into the system, an avatar agent and visitor agent are created. The avatar agent is responsible for presenting information to the visitor. First, it shows a form to get the visitor's information, and when the visitor fills his basic information the visitor agent will initiate a record in the user's profile database. The avatar agent will then ask the visitor to choose a planned, recommended or normal visit. Suppose he chooses the normal visit, a position agent will be created to get the location of the visitor and report it to the route agent. Once Sami is close to a PoI, the route agent will send a request to the wiki agent to get the PoI information. Finally the avatar agent will present the information.

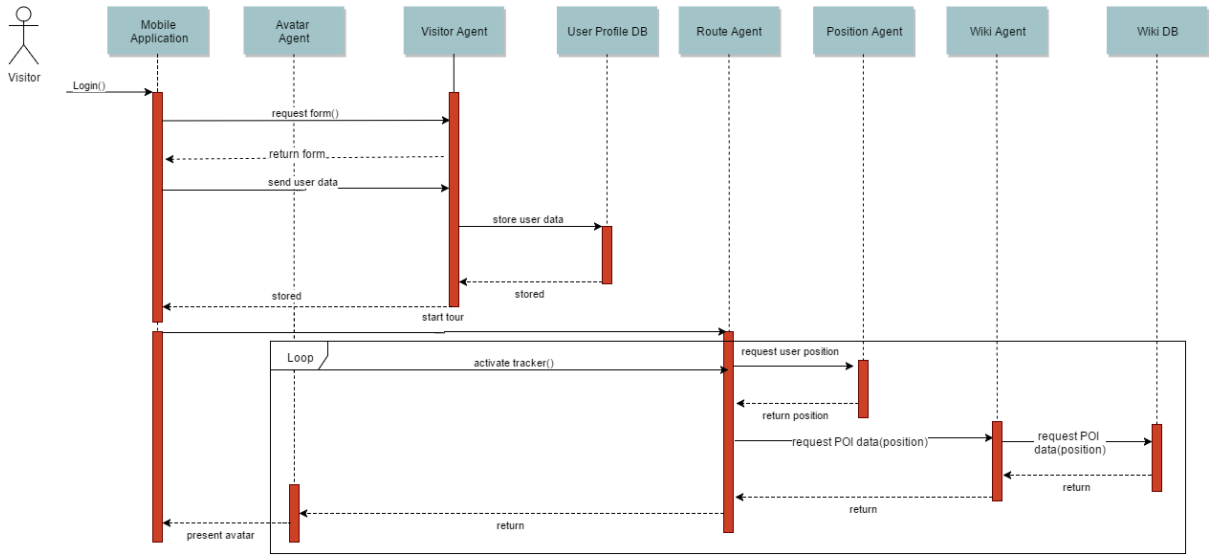

Figure 22 Normal Visit Sequence Diagram

Scenario 2: Planned visit

Sarah is visiting the church for the first time, and she only has 30 minutes to view as many PoIs as possible. She chooses a planned visit. The system will ask her to set the visit duration since there are too many PoIs inside the location and she has limited time. The system will call the planner agent to plan a visit according to her time and interests.

- Agents interaction in the planned visit:

First, the avatar agent will send a request to the planner agent with the time of the visit. The planner agent will then follow several steps: First, it will get Sarah's information from the visitor agent. Then it will evaluate each PoI according to her interests after that it will calculate the optimal path between the PoIs and compare it with the requested time. The PoIs will be presented on the location map as nodes. When Sarah presses the GO button the tour will start and the route agent will get activated and the position agent will be created to get the real time location of the visitor. Once she reaches each PoI the route agent will send a request to the wiki agent to get the PoI information, and the avatar agent will present it.

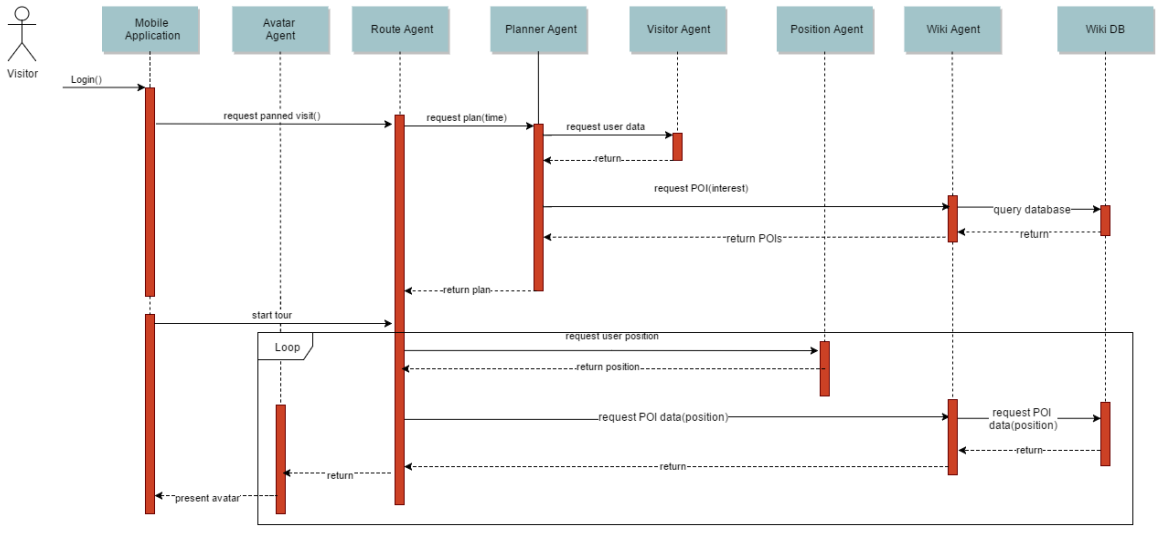

Figure 23 Planned Visit Sequence Diagram

Scenario 3: Recommended Visit

Ahmad prefers to choose a recommended visit to the church. In this case, the system will suggest to him a set of static visits in which the PoIs are ordered in sequence by the system. As this type of visits do not have limited time, Ahmad is free in the time he spends at each PoI. The visit starts when he presses the start button and once he reaches each PoI in the plan, the system will retrieve the related information about it. The visit will end when he visits all the PoIs in the plan.

- Agents interactions in recommended visit:

A set of agents will collaborate to provide this type of visit. Once Ahmad logs into the system, an avatar agent and a visitor agent are created. The avatar agent is responsible for presenting information and receiving requests from the visitor. First, it shows a form to get the visitor's information, and when he fills his basic information the visitor agent will initiate a record in the user's profile database. The avatar agent will then ask Ahmad to choose a recommended, planned or normal visit. Suppose he chooses the recommended visit, the avatar agent will retrieve a map with a set of PoIs. When he presses the start button the position agent will be created to get the location of the visitor. A route agent will also be initiated to track the tour. Once Ahmad is close to a PoI, the route agent will send a request to the wiki agent to get the PoI information, and the avatar agent will present the PoI information.

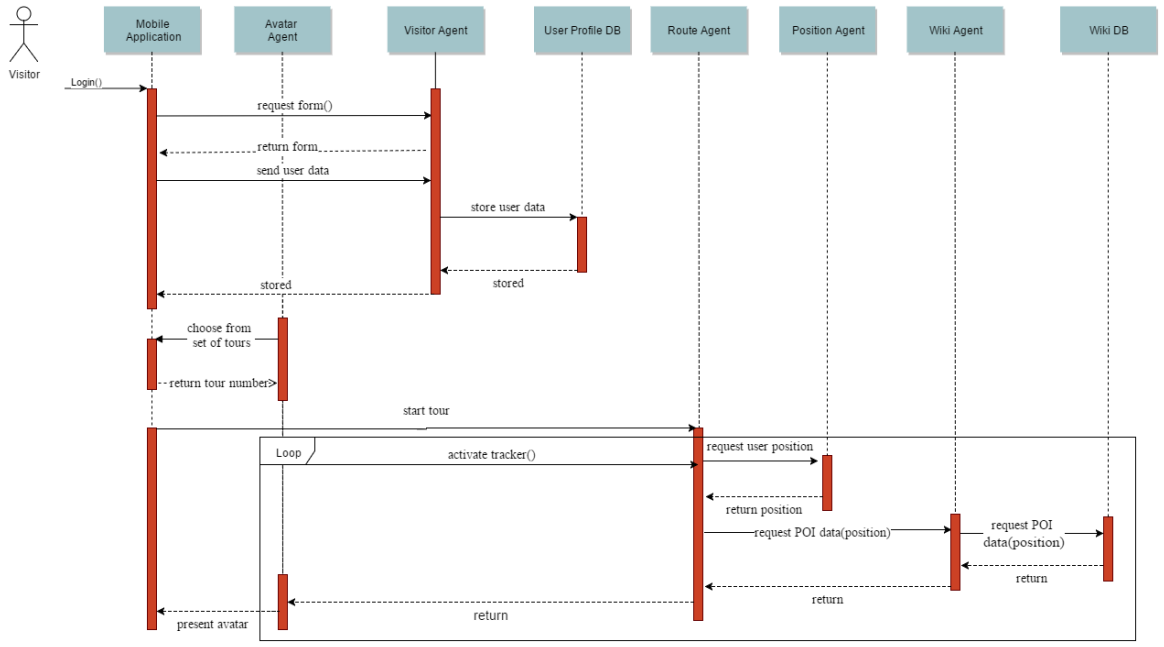

Figure 24 Recommended Visit Sequence Diagram

5.3. Class Diagram

Class diagram describes the static structure of a system. The classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes.

This class diagram represents main classes that form the core of the system as follow:

- **User class** represents the visitor entity.
- **User Profile** class represents the visitor profile which contains all information about the visitor such as id, name, language, interest and behavior.
- **Route Agent** class represents the route agent; its attributes, and the operations performed by this agent. (Monitors visit tours)

- **Visitor Agent** class represents the visitor agent; its attributes, and the operations performed by this agent. (Get visitor id, age, language, interest, interactions)
- **Position Agent** class represents the position agent; its attributes and operations. (Report visitor's position)
- **Location Provider** class represents the location of the visitor.
- **Planner Agent** class represents the planner agent; its attributes and operations. (Create plan)
- **Wiki Agent** class represents the wiki agent; its attributes, and operations (search the database, retrieve data).
- **DB Adapter** class represents the entity which deals with internal and external databases.
- **Avatar Agent** class represents the avatar agent; its attributes and operations. (Represent the content to the visitors).
- **Capacity Agent** class represents the avatar agent; its attributes and operations. (Report maximum visitor capacity of each PoI).

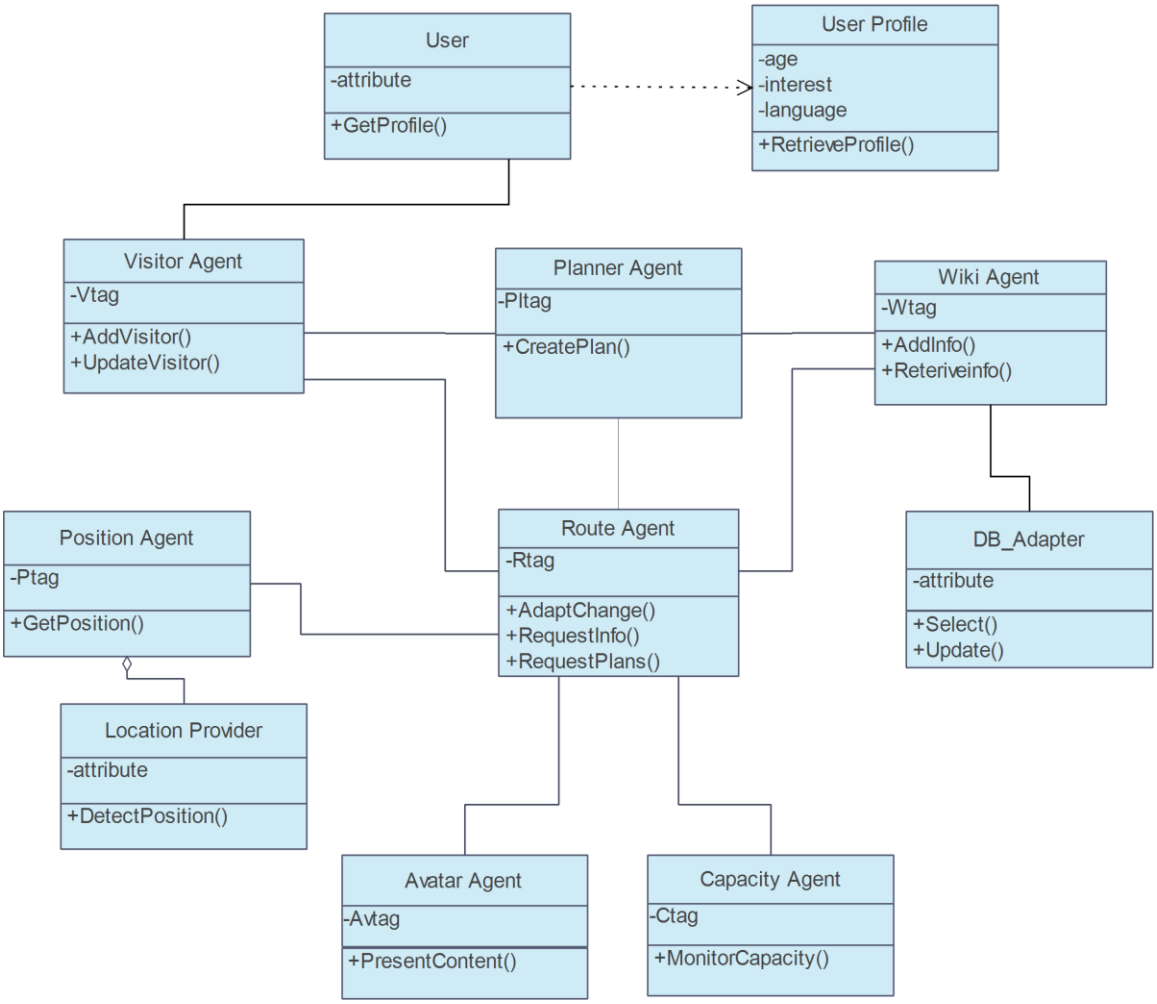

Figure 25 Class Diagram

5.4. Use Case Diagram

Class diagram shows the main actors with the main functions provided by the system. In this system we have seven main actors which are: i) visitor agent which is responsible for gathering the user's basic information. ii) avatar agent which mainly displays data to the visitors. iii) wiki agent that retrieves the requested data. iv) planner agent which generates plans. v) capacity agent which monitors the maximum number of visitors allowed for each

PoI. vi) position agent which detects the visitor position vii) route agent which monitors the tours to keep the visitor in time and to request data and position.

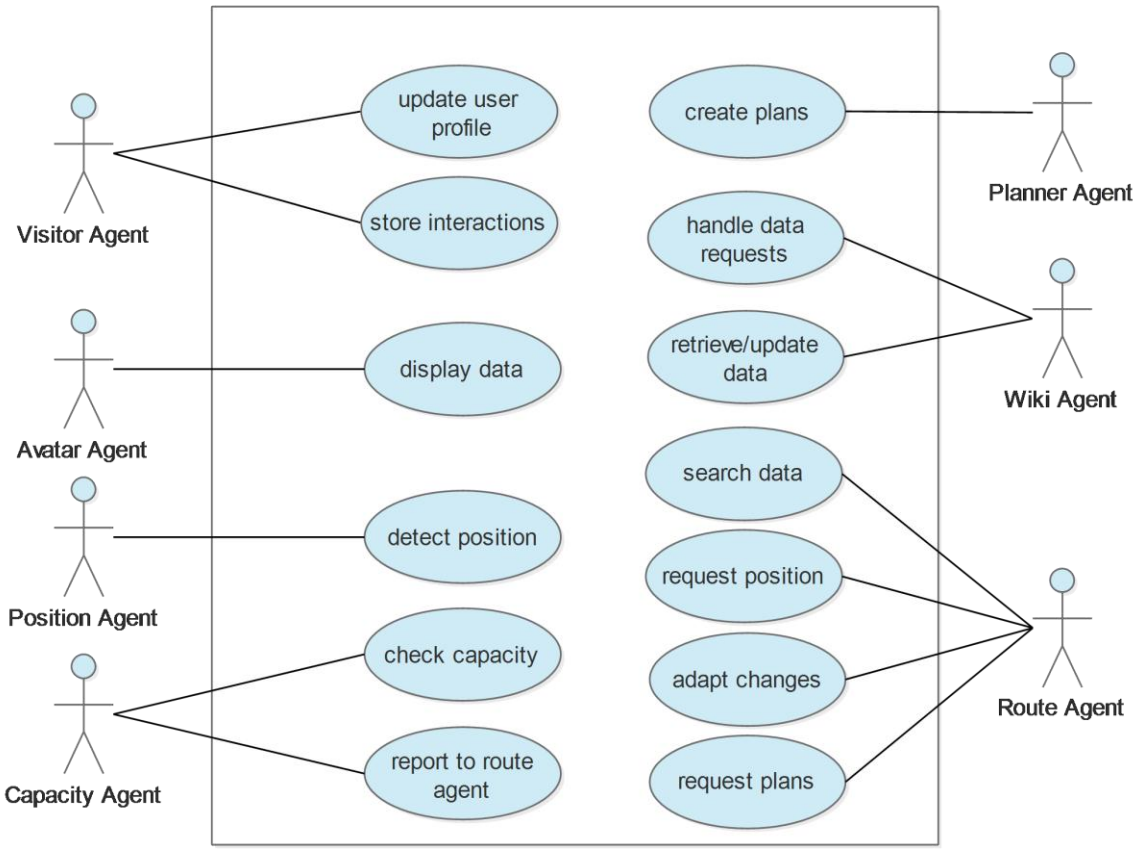

Figure 26 Use Case Diagram

5.5. Agents Algorithms

We consider the following parameters to implement our model:

- Visitor Interest (v_interest): real number refers to interest id that represents the type of the visitor interest.It is used to select the PoIs having the same type.
- Available time (total_v_time): time unit refers to the visit duration that each visitor wants to spend inside the location.
- Time per PoI (p_time): refers to the time the visitor spends on each PoIs.

- Number of PoI (p_number): refers to the number of PoIs involved in the path.
- Distance between PoIs (p_distance): simulates the distance between the PoIs. It is used with the visitor speed to calculate the cost of the selected path.
- Speed of the visitor (v_speed): simulates the speed of the visitor inside the location. It is used to calculate the cost of the path.
- Capacity of PoIs (p_capacity): refers to the maximum visitor capacity of each PoI.
- Related PoIs (PoI[x]): refers to the set of the point of interest that matches the visitor interest v_interest.

For the adapting real time changes we consider additional set of parameters which are:

- Total PoIs (p_total): refers to all the PoIs composing the visit path.
- Visited PoIs (p_visited): refers to the PoIs that are already visited by the visitor.
- Remain PoIs (p_remain): refers to the PoIs remained not visited by the visitor.
- Number of remains (pr_num): refers to the number of PoIs that remain in the path.

1- Avatar Agent: is responsible for presenting the results and information to the visitor. This agent takes into consideration the device capabilities. It presents the content as a text, image or video, or uses the avatar "character".

```
private class PresentContent extends CyclicBehaviour
RECEIVE visitor arguments FROM RouteAgent
RECEIVE Message FROM RouteAgent
IF Age between 5-10
        Presentation_style 1
ELSE Age between 10-18
```

```
        Presentation_style 2
ELSE
        Presentation_style 3
END IF
```

Algorithm 1: Avatar Agent "Present Content"

2- Visitor Agent: this agent is responsible for creating and updating the visitor's profile and saving his interactions. Any interaction with the user profile database is done through this agent.

```
private class AddVisitor extends OneShotBehaviour
        GET name, age,language
        GET v_interest
        INSERT INTO visitor_porfile VALUES (name, age,language, interest)

private class UpdateVisitor extends OneShotBehaviour
        GET visitor_id
        GET v_interest
        UPDATE visitor_porfile SET interest= v_interest WHERE id=visitor_id
```

Algorithm 2: Visitor Agent "Create/Update Profile" algorithm

3- Position Agent: this agent reports the visitor's location to the route agent to be used to retrieve the content according to the visitor's current location.

```
private class GetPosition extends TickerBehaviour

        GET visitorposition
        ACLMessagemsgp = new ACLMessage()
        SET content to visitorposition
        SET Receiver id to route agent id
        SET Address to route agent address
        SET Receiver(routeagent)
        SEND msgp
```

Algorithm 3: Position Agent "Get Position" algorithm

4- Wiki Agent: this agent issues queries on the wiki database to retrieve data about the location and the PoIs. It also inserts new data proposed by the visitors.

```
private class RetreiveInfo extends OneShotBehaviour
        Receive msg FROM route agent
        GET position
        SELECT item_name, item_description from ITEMS
        SEND result To route agent

private class AddInfo extends OneShotBehaviour
        Receive msg FROM route agent
        GET item_name, item_decription, item_position
        INSERT INTO items values item_name, item_description, item_position
```

Algorithm 4: Wiki agent "retrieve/ add information" algorithm

5- Route Agent: this agent tracks the visitor during the route to insure that the visitor follows the plan. It receives the location from the position agent and requests the content from the wiki agent. In addition, it requests the planner agent to generate plans. Once the visitor chooses between Planned/Normal/ Recommended visits, the route agent will act according to this choice.

A. Planned Visit:

```
SEND request to PlannerAgent with arguments [visitor ID]
Receive PoI list from Planner Agent
GET Item-Positions
GET calculated time for the path
GET Visitor-Position
While (total-v-time!=0)
 IF (Visitor-Position INRANGE Item-Position)
        SEND  Request To WikiAgent
        AdaptChange()
 END IF
END WHILE
```

```
AdaptChange()
GET calculated time for the path (p_distance, v_speed, p_number, p_time)
GET actual time to reach the object
FOR all elements in the visitor path
IF (actual time > calculated time && position of the visitor= PoI position)
        p_remain= p_total-p_visited
        remaining time= total_v_time- current time
        Call planner agent (p_remain, remaining time)
 END IF
END FOR
```

Algorithm 5: Route agent for planned visit algorithm

B. Normal Visit:

```
GET visitor_position From PositionAgent
IF visitor_position != Previous Position
        SEND Message(visitor_position) to WikiAgent
        RECEIVE results from WikiAgent
        SEND results to AvatarAgent with visitor_arguments
END IF
```

Algorithm 6: Route agent for normal visit algorithm

C. Recommended Visit:

```
FOR all PoIs in Recommended_List
GET PoIs_positions, visitor_position
   IF  visitor_position INRANGE PoIs_position[i]
            SEDND request To WikiAgent
             RECEIVE result
             SEND result To AvatarAgentwith visitor_arguments
  END IF
END FOR
```

Algorithm 7: Route agent for recommended visit algorithm

Furthermore route agent receives an inform message from capacity agent when one of the POIs is locked and reached the maximum visitor capacity.

```
Receive msg FROM capacity agent
GET PoI id
IF visit_path contains(PoI id) && Next node = PoI id
        p_remain= p_total - p_visited
        remaining time= total_v_time - current time
Call planner agent (p_remain, remaining time)
END IF
```

Algorithm 8: Route agent adapt capacity

6- Planner Agent: this agent generates plans according to i) visitor's interest and time ii) shortest path with the best weight. To choose the best path we perform several steps:

- Get the visitor's interest by matching his interest with the tags of each PoI and return the top matches.
- Calculate the shortest path between the PoIs using the dijkistra algorithm which finds the shortest path from the source to the destination. All the nodes have a value or weight represent the distance. The algorithm starts with the source node and visits the connected nodes by selecting the node with the lowest value. The process is repeated until reaching the destination [42].
- Finally calculate the time needed to visit the PoIs according to the requested visit duration.

```
private class CreatePlan extends OneShotBehaviour
    GET v_duration
    IF closing_time-v_visit !=0
            GET v_intrest
            GET p_distance
            SET result=bestTour(p-distance, v_interst, v_time);
    ACLMessagemsg_path = new ACLMessage();
    msg_path.setContent(result);
```

```
            SET AID receiver to route agent id
            SET address to route agent address
            SEND msg_path
        ELSE
        NOTIFY visitor
```

Algorithm 9: Planner agent "create plan" algorithm

7- Capacity Agent: this agent is responsible for monitoring the physical capacity of each PoI, and when the PoI reaches its maximum visitor capacity, this agent informs all the route agents to take this into consideration.

```
private class MonitorCapacity extends TickerBehaviour
FOR all registered visitors
  Get visitor_path;
  Get capacity;
  FOR all point in visitor path
        Get visitor position
        IF number_of_visitors =capacity[i+1]  THEN
        INFORM route agent
        END IF
   END FOR
END FOR
```

Algorithm 10: Capacity Agent "monitoring Capacity" algorithm

Conclusion

In this Chapter we have presented our application software modeling. First, we have presented the main objectives and the flow of the events which is described in the scenarios. Furthermore, we have presented the model prototype which is built based on JDAE platform and demonstrates our model with proposed agents, their behaviors and the interactions between them.

6. Chapter Six: Implementation and Simulation:

In this chapter, we illustrate the implementation and testing phases. The first section is composed of the prototype implementation part; tools, code and results. In the second section we present agent simulation using netlogo to illustrate visitors' behaviors in the environment.

6.1. System implementation

We build a prototype based on JADE platform which demonstrates our model with proposed agents and the interactions between them. The process of choosing the routes, adapt the changes, and deliver information has also been considered. As mentioned previously the application is based on the client server architecture. The client acts as an input output terminal and the server performs all the processes. We implemented this prototype as a desktop application to facilitate the process of creating and managing the agents.

At this stage we use netbeans IDE 8.1 and JADE platform. We convert each service provided in the model to a separated class. Agents' actions are specified by the behavior class with two main methods which are the action method to perform the actual task and the done method to verify if the task has been completed [43].

- Main Class: This class is responsible for creating and starting all the agents

```
public static void main(String[] args){
      Runtime rt=Runtime.instance();
      Profile p=new ProfileImpl();
p.setParameter(Profile.MAIN_HOST, "localhost");
p.setParameter(Profile.GUI, "true");
ContainerController cc=rt.createMainContainer(p);
AgentController ac;
      try{
      ac=cc.createNewAgent("AvatarA", "prototype.AvatarAgent", null);
      ac=cc.createNewAgent("PositionA", "prototype.PositionAgent", null);
      ac=cc.createNewAgent("VisitorA", "prototype.VisitorAgent", null);
      ac=cc.createNewAgent("WikiA", "prototype.WikiAgent", null);
      ac=cc.createNewAgent("CapacityA", "prototype.CapacityAgent", null);
      ac=cc.createNewAgent("PlannerA", "prototype.PlannerAgent", null);
      ac=cc.createNewAgent("RouteA", "prototype.RouteAgent", null);
ac.start();
      }
      catch(StaleProxyException e) {e.printStackTrace();}
   }
```

Class 1 Main Class

- Visitor Agent class: gets the visitor's information from the form and store it in the user profile database.

```
try {
   Connection con= DriverManager.getConnection(host, uName, uPass);
   Statement stmt = con.createStatement();
   String sql = "insert into SHAFAQ.USER_PROFILE (USERNAME,LANG,AGE)
values ('"+username+"','"+Language+"','"+Age+"')";
stmt.executeUpdate(sql);

     }catch ( SQLException err ) {
System.out.println( err.getMessage( ) );
   }
```

Class 2 Visitor Agent Class

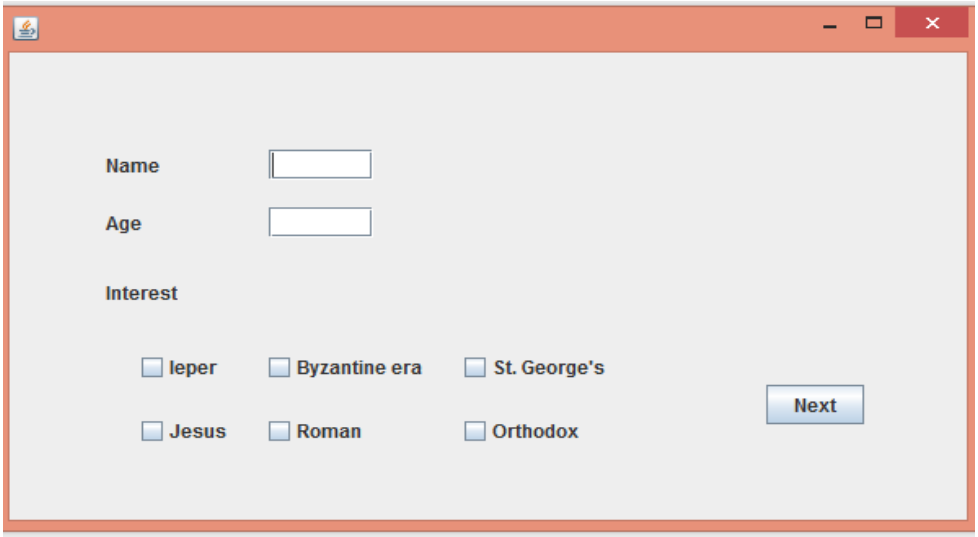

Figure 27 Visitor agent GUI

- Position Agent class: position agent reports the visitor's location to the route agent to be used to retrieve the content according to the visitor's current location.

```
public class PositionAgent extends Agent{
        @Override
            protected void setup() {
addBehaviour(new OneShotBehaviour() {
        @Override
        public void action() {
int position= findposition();
            String msg = String.valueOf(position);

ACLMessagemsgp = new ACLMessage(ACLMessage.INFORM);
msgp.setContent(msg);
            AID remoteAMSf = new AID("route@192.168.1.6:1099/JADE",
AID.ISGUID);
remoteAMSf.addAddresses("http://192.168.1.6:1099/JADE");
msgp.addReceiver(remoteAMSf);
            send(msgp);
          }

      });
   }
 }
```

Class 3 Position Agent Class

- Wiki Agent Class: it is composed of two main behaviors; one is for searching the wiki database and the second is to insert/update data about the location. Here we implement the first behavior. This class receives a request message from the route and returns the search results from the wiki database.

```
       public void setup(){
addBehaviour(new CyclicBehaviour() {
              @Override
        public void action()
              {
                 String host = "jdbc:derby://localhost:1527/church";
                 String uName = "shafaq";
                 String uPass = "shafaq";
ACLMessagemsg = receive();
               if (msg!=null) {
intItemPos = Integer.parseInt(msg.getContent());
                  try {
                  Connection con = DriverManager.getConnection(host, uName, uPass);
                  Statement stmt = con.createStatement();
                  String sql = "SELECT * FROM SHAFAQ.CITEMS WHERE
Position=" + ItemPos + "";
ResultSetrs = stmt.executeQuery(sql);
                  while (rs.next()) {
System.out.println("conn");
                     result=rs.getString("itemdesc");
                  }
rs.close();
               } catch (SQLException err) {
System.out.println(err.getMessage());
               }

ACLMessage reply = msg.createReply();
reply.setPerformative( ACLMessage.INFORM );
reply.addReceiver(msg.getSender());
reply.setContent(result);
                  send(reply);
                }
               block();
             }
});  }
```

Class 4 Wiki Agent Class

- Route Agent class: this class represents the route agent which is responsible for tracking the visitor and retrieving the content according to his location. Route agent will act according to the visit type.

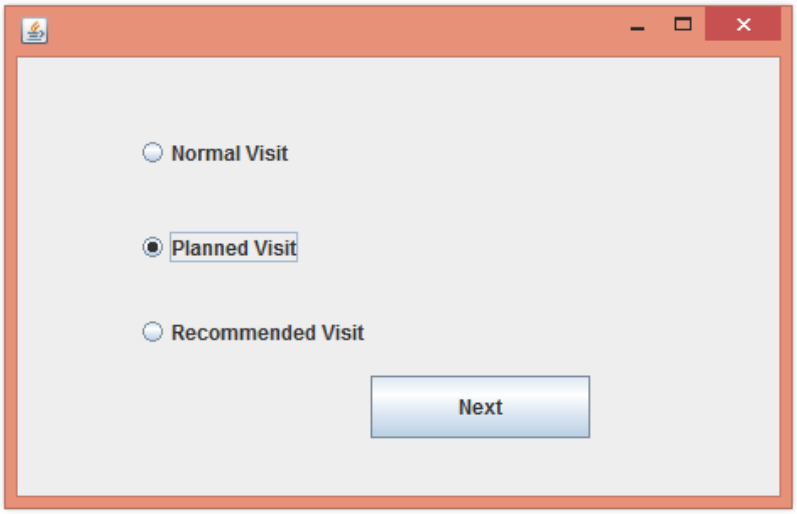

Figure 28 Visit types

In normal and recommended visit, it will get the position, send a message to the wiki agent to get the information about the PoI. Then when it receives a replay, route agent will pass the result to the avatar agent.

```
// Receive Position from PositionAgent

addBehaviour(new CyclicBehaviour() {
        @Override
        public void action() {
MessageTemplate mt1 =
        MessageTemplate.and(
        MessageTemplate.MatchPerformative( ACLMessage.INFORM ),
        MessageTemplate.MatchSender( new AID( "PositionA",
                                        AID.ISLOCALNAME))) ;
ACLMessagemsg = receive(mt1);
            if (msg != null) {
                Position= Integer.parseInt(msg.getContent());
        }else block();
        } });
```

```
// Send Position to the wiki and wait for reply

addBehaviour(new CyclicBehaviour() {
        @Override
        public void action() {
ACLMessagemsgpt = new ACLMessage(ACLMessage.REQUEST);
            String vPosition= String.valueOf(Position);
msgpt.setContent(vPosition);
            AID remoteAMSf = new AID("WikiA@192.168.1.6:1099/JADE",
AID.ISGUID);
remoteAMSf.addAddresses("http://192.168.1.6:1099/JADE");
msgpt.addReceiver(remoteAMSf);
            send(msgpt);

ACLMessage reply = receive();
                                    if (reply != null) {
                                results=reply.getContent();
                            }

        }
      });
```

Class 5 Route Agent Class "Normal"

In a planned visit, route agent will send a request message to the planner agent and send the visitor's id with the request. The planner agent will replay with the proposed path.

```
// request plan from Planner Agent

addBehaviour(new CyclicBehaviour() {
        @Override
        public void action() {
ACLMessagemsgpt = new ACLMessage(ACLMessage.REQUEST);
msgpt.setContent("Plan");
AID remoteAMSf = new AID("PlannerA@192.168.1.6:1099/JADE", AID.ISGUID);
remoteAMSf.addAddresses("http://192.168.1.6:1099/JADE");
msgpt.addReceiver(remoteAMSf);
send(msgpt);
ACLMessage reply = receive();
        if (reply != null) { bestPOI=reply.getContent();}}  });
```

Class 6 Route Agent Class "Planned"

- Planner Agent class:

In order to reach the best plan that matches the visitor's interest and time, planner agent performs several sequential behaviors:

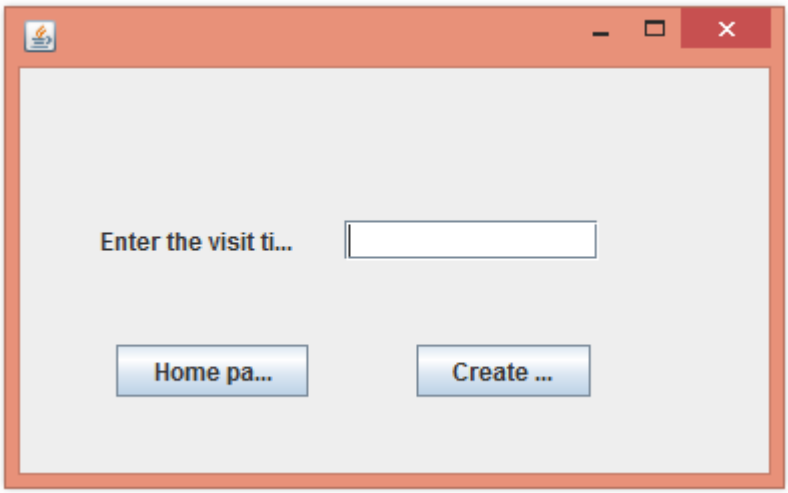

Figure 29Visit Duration

1-Evaluate the PoIs according to the visitor interest;

```
   public static Integer[] weights() {

          String host = "jdbc:derby://localhost:1527/church";
          String uName = "shafaq";
          String uPass = "shafaq";
          ArrayList<Integer> list= new ArrayList<Integer>();

          Integer[] result={};
             try {
                    Connection con = DriverManager.getConnection(host, uName, uPass);
                    Statement stmt = con.createStatement();
        String sql ="Select ItemID,Count(TagID) as Weight from ITag where TagID in (
SELECT TagID FROM SHAFAQ.USER_TAG Where UserTag = 1) Group by
ITEMID Union Select ID,0 as Weight from CItems where ID not in ( Select ItemID
from ITag where TagID in ( SELECT TagID FROM SHAFAQ.USER_TAG Where
UserTag = 1))";
                 ResultSet rs = stmt.executeQuery(sql);
                    while (rs.next()) {
                       list.add(Integer.parseInt(rs.getString("weight")));
```

```
                }
                result = list.toArray(result);
                rs.close();   }
            catch (SQLException err) {
                System.out.println(err.getMessage()); }
            return result;
    }
```

Class 7 Planner Agent "get weights"

2- Initiate the graph: the graph represents the PoIs in the church, and the distance between each them as illustrated in figure 30.

```
private static final Graph.Edge[] GRAPH = {
    new Graph.Edge("2", "3", 3),
    new Graph.Edge("2", "4", 7),
    new Graph.Edge("2", "5", 10),
    new Graph.Edge("3", "4", 5),
    new Graph.Edge("3", "4", 5),
    new Graph.Edge("4", "5", 2),
    new Graph.Edge("5", "6", 20),
    new Graph.Edge("5", "2", 10),
    new Graph.Edge("2", "6", 20),
    new Graph.Edge("6", "7", 5),
    new Graph.Edge("6", "8", 6),
    new Graph.Edge("7", "8", 7),
    new Graph.Edge("2", "8", 30),
    new Graph.Edge("1", "2", 15),
    new Graph.Edge("1", "3", 16),
    new Graph.Edge("1", "4", 14),
    new Graph.Edge("1", "5", 14),
    new Graph.Edge("1", "6", 16),
    new Graph.Edge("1", "7", 20),
    new Graph.Edge("1", "8", 20),
    new Graph.Edge("2", "1", 15),
    new Graph.Edge("3", "1", 16),
    new Graph.Edge("4", "1", 14),
    new Graph.Edge("5", "1", 14),
    new Graph.Edge("6", "1", 16),
    new Graph.Edge("7", "1", 20),
    new Graph.Edge("8", "1", 20),
  };
```

Class 8 Planner Agent "GRAPH"

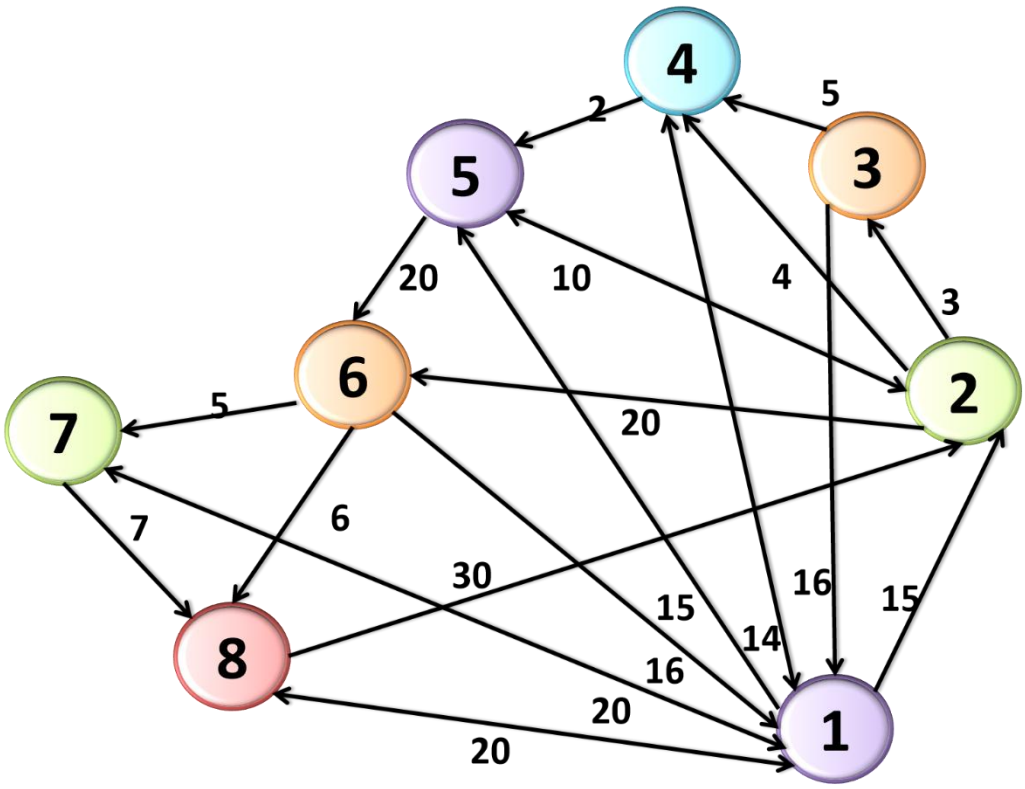

Figure 30 Distance between PoIs

3- Finds the best combination of PoI that satisfies the visitor time and interest.

```
Graph g = new Graph(GRAPH);
// get visitor interest
Interest visitor_int=new Interest();
Integer[] nodesV=visitor_int.weights();
String a = Arrays.toString(nodesV);
String nodes[]=a.substring(1,a.length()-1).split(", ");
//declear needed variables:
int j=0;
String result;
String start;
String end;
```

```
        public static String bestTour() {
String[] visited= new String[nodes.length];
    int k=0;
    double time=0;
    double totaltime=1800;
    String finalPath="";
    // visitor interest
    System.out.println("According to your interst you can visit: "+
Arrays.toString(nodes));
    // creating the path and calculating the time
    while(j<nodes.length-1)
    {//set start** end nodes
    start=nodes[j];
    end=nodes[j+1];
    // get shortest path using dijkstra algorithm
    g.dijkstra(start);
    result=g.getPath(end);
    String finaldist=g.getDis(end);
    String[] arraypath = result.split(";");
    String[] arraydis = finaldist.split(";");
    System.out.println("The Shortest path between each two elements "+
Arrays.toString(arraypath) + "with total distance "+arraydis[arraydis.length-1]);
    int lastdis=Integer.parseInt(arraydis[arraydis.length-1]);
    time+=lastdis/1.4+300; // dist for path+ time view
    if (time< totaltime)
    {
   // check if the shotest distance between each 2 elements includes nodes from the
path
    for (int t = 0; t < arraypath.length; t++) {
       if(Arrays.toString(nodes).contains(arraypath[t])==true)
             {
              if(Arrays.toString(visited).contains(arraypath[t])==false)
              {
                 if (time+300<=totaltime) // if we have only 5 mint or less
                 { // in case there is an interested node in the path
                 visited[k]=arraypath[t];
                  k++;
                  time+=300;
                }
                 finalPath+=arraypath[t]+" ->";
              }
          }
       }
    } else { System.out.println("Time out");
```

```
        break;
    }
        j++;
    }
    System.out.println("your final path is " + finalPath);
  }//end while
}// end setup
```

Class 9 Planner Agent "bestTour"

The results are represented in figure 31.A planner class starts the calculations from the most related to the least related PoIs. It takes two nodes and finds the shortest path between them. If the path contains any element from the related interests of the visitor, it will be added to the visited nodes. The process proceeds until the time to visit the nodes reaches the total time requested by the visitor.

```
run:
According to your interst you can visit: [2, 8, 7, 1, 6, 4]
The Shortest path between each two elements [2, 6, 8]with total distance 26
The Shortest path between each two elements [8, 1, 7]with total distance 40
The Shortest path between each two elements [7, 1]with total distance 20
Time out
your final path is 2 ->6 ->8 ->1 ->7 ->
BUILD SUCCESSFUL (total time: 3 seconds)
```

Figure 31 Results from plan agent

- Avatar Agent Class: The class presents the results and information to the visitor. This agent receives content from the route agent and presents it as a text, image or video, or it uses the avatar “character”.

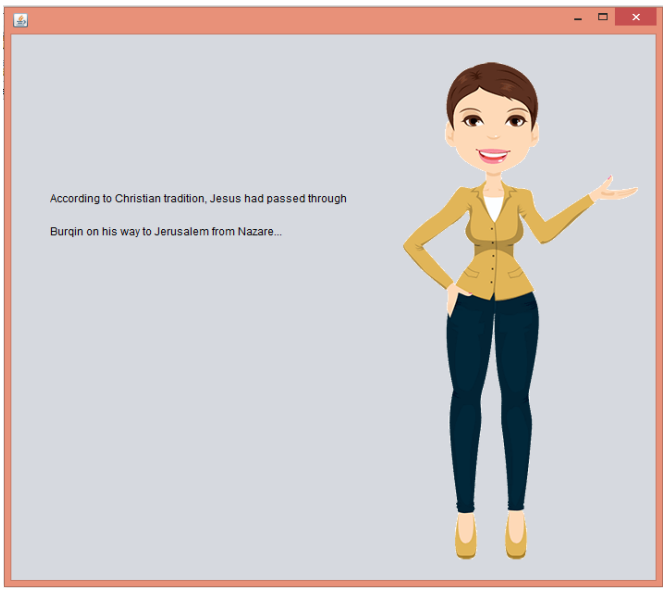

Figure 32 Avatar Agent GUI

```
addBehaviour(new CyclicBehaviour() {
        @Override
        public void action() {
          String content;
ACLMessagemsg = receive();
            if (msg != null) {
               content = msg.getContent();
dospeak(content, "kevin16");
        }else block();
        }
   });
            public void dospeak(String speak, String voicename) {
              String speaktext = speak;

               try {
                  Voice voice;
        VoiceManagervm = VoiceManager.getInstance();
                  voice = vm.getVoice(voicename);
        voice.allocate();
        voice.speak(speaktext);
              } catch (Exception e) {
              }
           }
```

Class 10 Avatar Class

- Capacity Class: this class notifies all the route agents about the full capacity spaces.

```
// search for all route agents " active"
        addBehaviour(new TickerBehaviour(this, 60000) {
                protected void onTick() {
                   try {
        DFAgentDescription template = new DFAgentDescription();
        ServiceDescriptionsd = new ServiceDescription();
        sd.setType("full_capacity");
        template.addServices(sd);
        DFAgentDescription[] result = DFService.search(myAgent, template);
        System.out.println("Found the following Route Agents agents:");
        RouteAgent = new AID[result.length];
                    for (inti = 0; i<result.length; ++i) {
        RouteAgent[i] = result[i].getName();
        System.out.println(RouteAgent[i].getName());
                        String msg = String.valueOf("over capacity");
        ACLMessagecfp = new ACLMessage(ACLMessage.CFP);
                        for (int j = 0; j <RouteAgent.length; ++j) {
        cfp.addReceiver(RouteAgent[j]);
                        }
        cfp.setContent(msg);
        myAgent.send(cfp);
```

```
System.out.print(cfp);
                }
            } catch (FIPAException ex) {
Logger.getLogger(CapacityAgent.class.getName()).log(Level.SEVERE, null,
ex);
            } } });
```

Class 11 Capacity Class

The search results shown in figure 33, represent the active route agents, the capacity agent will use this list to send inform message to notify all route agents about the full capacity PoIs.

```
Found the following Route Agents agents:
route3@192.168.1.6:1099/JADE
route1@192.168.1.6:1099/JADE
route2@192.168.1.6:1099/JADE
```

Figure 33 Results by capacity agent

```
  :sender  ( agent-identifier :name capacity@192.168.1.6:1099/JADE )
  :receiver  (set ( agent-identifier :name route3@192.168.1.6:1099/JADE ) ( agent-identifier :name route1@192.168.1.6:1099/JADE ) ( agent-ide
ntifier :name route2@192.168.1.6:1099/JADE ) )
  :content  "over capacity"
 )
```

Figure 34 Message from capacity to all route agents

6.2. Simulation

The general purpose of the simulation process is to understand the processes of the system, presents and tests theories to track the system's behavior and to model systems that are composed of many components interacting in a complex way. In Multi-Agent System (MAS) simulation visualization is considered as a key factor to identify and understand the behavior models [44].

Simulating agents involves identifying the agents' roles, and their interaction with each other and with the environment [45].

This simulation part was built under NetLogo environment which is a simulator developed for modeling real-world phenomena using Multi-Agent systems. In this environment, agents are represented as turtles, and operate according to a set of rules. NetLogo illustrates the interactions between the agents at the lower (micro) level behaviors of individuals and the higher (macro) level behaviors that emerge from their interactions [46].

We simulate our model to present the visitors' behaviors in the environment. It also presents the increase in the number of visitors, as well as the diversity between them.

This work was done in a cooperation with Ihab Nassra, for a research paper not submitted yet.

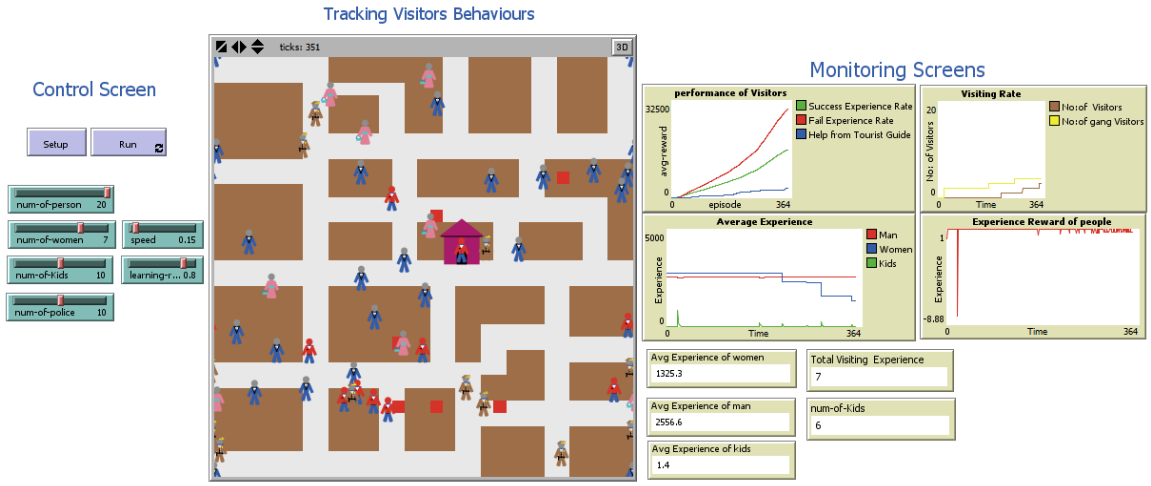

Figure 35 System Simulation

Environment: we represent the location as brown area. Each house represents the PoI. People are divided into four groups: men, women, kids, tourist guides. They randomly move in the location.

We control the following parameters: number-of-persons, movement speed and learning rate.

In the monitoring screens we present the following control capabilities:

1- Visiting rate: count the total number of visitors and measure the visiting rate.

2- Average experience: measures the visiting experience of people by considering how many times they need help from a tourist guide.

3. Average experience of men/women/kids: measures the average acquired experience of men/women/kids.

6. Total visiting experience: measures the total visiting experience of all people's types.

The following is a sample of the code to create visitor agents "people" using netlogo:

```
create-people num-of-person
 [
   set Qnew 0
   set Q 0
   set shape "person business"
   set size 2
   set Experience random 5000
   set age random 50
   set to-walk? false
   set to-be-Visit? false
   set reward 0
   set qtable [0]
   set visits-table []
   set Visited 0
   set avg-Experience-people mean [Experience] of people
   set init-xcor random-xcor
   set init-ycor random-ycor
   if age > 14
   [
     set fight random 2
     show word "fight " fight
   ]
   put-on-empty-road
 ]
```

```
ask people
 [
   move
   set to-walk? true
   set to-be-Visit? false
   learn
   if any? Monuments-on neighbors[visit-monuments]
 ]

to visit-monuments
 if Experience < 1000
 [
   show "going to work"
   move-to one-of Monuments
   show ticks
   set Experience Experience + 500
   show word " Experience after visit " Experience
    ]
end
```

Conclusion

In this chapter we have illustrated the prototype implementation part. This prototype is developed based on JADE platform. We have presented the agent classes, and the results for each one. We have also presented the agent simulation using netlogo to illustrate visitors' behaviors in the environment.

Conclusion and Perspectives

In this thesis, we have introduced the heritage locations as a complex adaptive system. The complexity of such systems raises from the diversity of the visitor's communities and their dynamic behavior inside the location. Moreover the complexity expands due to the process of delivering adaptable plans to these visitors according to their profiles and visit duration.

Throughout this thesis, we have discussed the use of Multi-Agent System (MAS) in the cultural heritage domain and proposed a multi-agent model to reduce the complex interactions between the location, data and visitors.

Our model is composed of a set of agents; visitor, position, avatar, wiki, planner, capacity and route agents. These agents interact and collaborate to fulfill the overall system objectives and they are capable of i) creating and updating the visitor's profile ii) tracking the visitor's position inside the location iii) presenting the content in an interactive way through the animated character's "avatar" iv) providing personalized plans according to the visitor's profile and time v) adapting new propositions according to the changing behavior of the visitors. vi) monitoring and considering the location and system constraints in the proposed plan.

As a final step, we have demonstrated a new application prototype that implements our proposed agents, their behaviors and the interactions between them. We have also presented agent simulation using netlogo to illustrate visitors' behaviors in the environment.

There are several future perspectives to enhance and optimize our users' experiences.

- We can use our thesis to achieve a higher level of interaction inside cultural locations through the collaboration with the augmented reality researchers, historians, multimedia and cultural heritage semantic.
- This thesis can be an entrance to a PhD subject in a complex digital heritage field based on agent technology.

References

[1] Elena Not; Daniela Petrelli, "Balancing Adaptivity and Customisation:," in *22nd Conference on User Modelling, Adaptation and Personalization*, Aalborg, Denmark,, 7-11 July, 2014.

[2] C. Dumcke and M. Gnedovsky, "The Social and Economic Value of Cultural Heritage: literature review," in *European Expert Network on Culture (EENC)*, July 2013.

[3] U. W. H. Center, "World Heritage Information Kit," 2008.

[4] F. N. F. f. Freedom, "Palestine Tourism Sector," in *International Chamber of Commerce*, August 2013.

[5] A. Rattrout and S. Abulhouf, "Modeling interactive multi agent system in cultural heritage locations," in *the third Hyperheritage International Seminar (HIS.3)*, Tunis, 2016.

[6] Y. Ahmad, "The Scope and Definitions of Heritage: From Tangible to Intangible," *International Journal of Heritage Studies*, 2006.

[7] R. Comes, Z. Buna and I. Badiu, "Creation and Preservation of Digital Cultural Heritage," *Journal of Ancient History and Archeology*, 2014.

[8] L. Ardissono, T. Kuflik and D. Petrelli, "Personalization in Cultural Heritage: The Road Travelled and the One Ahead," *User Modeling and User-Adapted Interaction,* vol. 22, 2011.

[9] Z. Noh, M. Sunar and Z. Pan, "A Review on Augmented Reality for Virtual Heritage System," 2009.

[10] A. Foni, N. Papagiannakis and G.Magnenat-Thalmann, "Virtual Hagia Sophia: Restitution, Visualization and Virtual Life Simulation," in *UNESCO World Heritage Congress Proceedings*, 2002.

[11] Z. Noh, M. Sunar and Z. Pan, "A Review on Augmented Reality for Virtual Heritage System," 2009.

[12] M. Linaza, D. Marimón, P. Carrasco, J. M. R. Álvarez, S. Aguilar and G. Diez, "Evaluation of Mobile Augmented Reality Applications for Tourism Destinations," in *Information and Communication Technologies in Tourism*, 2012.

[13] F. Frit, A. Susperregui and M.Linaza, "Enhancing Cultural Tourism experiences with Augmented Reality Technologies," 2005.

[14] N. Mourkoussis, F. Liarokapis, J. Darcy, M.Pettetsson, P. Petridis, P. Lister and M. White, "Virtual and Augmented Reality Applied to Educational and Cultural Heritage Domains," 2002.

[15] T. Pomonis, D. Koutsomitropoulos, S. Christodoulou and T. Papatheodorou, "Combining Semantic Web and Web 2.0 Technologies to Support Cultural Applications for Web 3.0," 2011.

[16] A. Amato, B. D. Martino and S. Venticinque, "BDI Intelligent Agents for Augmented Exploitation of Pervasive Environments," in *WOA*, 2011.

[17] T. Lam and R. Lee, "iJADE freeWalker An Intelligent Ontology," in *Computational Intelligence for Agent-based Systems*, Springer, 2007, pp. 103-125.

[18] I. Ayala, M. Amor, M. Pinto, L. Fuentes and N. Gámez, "iMuseumA: An Agent-Based Context-Aware Intelligent Museum System," *Sensors*, 2014.

[19] N. Shah, J. Siddiqi and B. Akhgar, "An Intelligent Agent Based Approach to MOSAICA'S Pedagogical Framework," in *ICEIS - International Conference on Enterprise Information Systems*, 2007.

[20] T. Kuflik, A. Albertin, P. Busetta, C. Rocchi, O. Stock and M. Zancanaro, "Tsvi Kuflik, Adriano Albertini, Paolo Busetta, Cesare Rocchi, Oliviero Stock, Massimo Zancanaro," in *Ubiquitous and Decentralized User Modeling workshop*, 2007.

[21] I. Lykourentzou, X. Claude, Y. Naudet, A. A. E. Tobias, G. Lepouras and C. Vasilakis, "Improving museum visitors' Quality of Experience through intelligent recommendations: A visiting style-based approach," in *Intelligent Environments*, 2013.

[22] I. Roes, "Personalized Museum Tour with Real-Time Adaptation on a Mobile Device with Multi-Point Touch Interface," July 2010.

[23] N. Jennings, "On agent-based software engineering," *Artificial Intelligence*, 2000.

[24] J. Magdaleno-Palencia, M. Garcia-Valdez, M. Castanon-Puga and B. Marquez, "Agents Based Adaptive Hypermedia System with the Competency Approach," *International Journal on New Computer Architectures and Their Applications*, 2011.

[25] F. Graesser, "Is it an Agent, or just a Program? A Taxonomy for Autonomous Agents," in *Proceedings of the Third International Workshop on Agent Theories*, Springer-Verlag, 1996.

[26] H. S. Nwana, "Software Agents: An Overview," *Software Agents: An Overview*, vol. 11, no. 3, pp. 1-40, 1996.

[27] " JAVA Agent Development Framework," Telecom Italia Lab, [Online]. Available: http://jade.tilab.com. [Accessed 16 November 2015].

[28] "How does GPS work?," MIO GPS Navigation Manufacturer, 5 8 2016. [Online]. Available: http://www.mio.com/technology-what-is-gps.htm. [Accessed 10 7 2016].

[29] J. Montanes, A. Rodriguez and I. Prieto, "Smart Indoor Positioning/Location and Navigation: A lightweight Approach," *International Journal of Artificial Intellignce and Interactive Multimedia*, vol. 2, no. 2, 2013.

[30] i. GmbH, "Indoor Positioning and Navigation A Guide on Technologies and Use Cases," 2016.

[31] S. Lawson, "Ten Ways Your Smartphone Knows Where You Are," IDG News Service, 6 April 2012. [Online]. Available: http://www.pcworld.com/article/253354/ten_ways_your_smartphone_knows_where_you_are.html. [Accessed 10 July 2016].

[32] J. Holland, Hidden Order: How Adaptation Builds Complexity, Massachusetts:

Perseus Books, 1995.

[33] A. AMAD, N. BOUHAI and K. ZREIK, "Data acquisition and enrichment in the context of poorly documented cultural heritage," in *3d Hyperheritage International Conference*, Tunis, 2016.

[34] P. Bourgine, D. Chavalarias and E. Perrier, "French Roadmap for Complex Systems," in *French National Network for Complex Systems*, 2008-2009.

[35] S. Chan, "Complex Adaptive Systems," 31 October 2001. [Online]. Available: http://web.mit.edu/esd.83/www/notebook/Complex%20Adaptive%20Systems.pdf. [Accessed 10 July 2016].

[36] M. Rupert, A. Rattrout and S. Hassas, "The web from a complex adaptive systems perspective," *Journal of Computer and System Sciences*, vol. 74, no. no.2, pp. 133-145, 2008.

[37] H. A. Abbas, S. I. Shaheen and M. H. Amin, "Organization of Multi-Agent Systems: An Overview," *International Journal of Intelligent Information Systems*, vol. 4, pp. 46-57, 2015.

[38] A. Rattrout, "Personalizing Dynamic Information Resources in Complex Web according to Complex Adaptive Systems Perspective," *PHD Thesis*, 2007.

[39] M. wooldridge, N. R. Jennings and D. Kinny, "The Gaia Methodology for Agent-Oriented Analysis and Design," 2000.

[40] R. Cervenka, I. Trencansky and M. Calisti, "Modeling Social Aspects of Multi-Agent Systems: The AML Approach," in *Agent-Oriented Software Engineering*, Springer Science and Business Media, 2006, pp. 28-39.

[41] M. Luck, P. McBurney, O. Shehory and S. Willmott, "Agent Technology: Computing as Interaction," in *the European Coordination Action for Agent-Based Computing*, 2005.

[42] Wikipedia, "Dijkstra's algorithm," [Online]. Available: https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm. [Accessed 7 May 2016].

[43] F. Bellifemine, G. Caire and D. Greenwood, Developing Multi-Agent Systems with JADE, Wiley, 2007.

[44] D. KORNHAUSER, W. RAND and U. WILENSKY, "VISUALIZATION TOOLS FOR AGENT-BASED MODELING IN NETLOGO," in *Proceedings of Agent 2007*, Chicago, 2007.

[45] P.-O. Siebers and U. Aickelin, "Introduction to Multi-agent Simulation," in *Encyclopedia of Decision Making and Decision Support Technologies*, Information Science Reference, 2008, pp. 554-562.

[46] S. Tisue and U. Wilensky, "NetLogo: Design and implementation of a multi-agent modeling environment," in *Agent 2004 Conference on Social Dynamics: Interaction,*, Chicago,, 2004.

الملخص

تمثل هذه الرسالة خطوة متقدمة في مجال تطوير التراث الرقمي من منظور الأنظمة متعددة العملاء بحيث تمثل العلاقة بين العناصر الفاعلة في المواقع التراثية متمثلة في المكان، الزوار والمحتوى. يهدف هذا العمل الى توصيل المحتوى لزوار المواقع الأثرية في الوقت والطريقة المناسبة مع مراعاة الفروقات بين الزوار والتغير في سلوكهم.

في هذا العمل تم دراسة المواقع الأثرية على أنها أنظمة مركبة قادرة على التكيف مع التغيرات. ويرجع السبب في كونها مركبة الى اختلاف الزوار في العديد من الامور مثل: اللغة والعمر والاهتمامات والاختلاف في سلوكهم. بالاضافة الى عدد الزوار المتزايد و تفاعلهم مع القطع التراثية في نفس الوقت. تم استخدام الأنظمة متعددة العملاء للتقليل من تعقيد النظام، ويتكون النظام المقترح من مجموعة من العملاء القادرة على: 1- انشاء سجلات للزوار و تخزين معلوماتهم و تحركاتهم داخل الاماكن الاثرية 2- تزويد الزوار بخطط لجولات داخل المكان مناسبة لاهتماماتهم 3- اكتشاف وحل التعارض في الجولات في وقت حدوثها 4- تسهيل التفاعل بين المستخدم والقطع الاثرية من جهة و تفاعله مع التطبيق الالكتروني من جهة اخرى بحيث يكون التفاعل تكامليا بين العناصر الثلاثة.