

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

Since the 1980s, agent technology has attracted an increasing amount of interest from the research and business communities (Nicholas & Wooldridge, 2012). In particular, the last decade has witnessed a steadily increasing number of different agent theories, architectures, and languages proposed in the literature. This increasing interest in agent technology is mainly due to its potential to significantly improve the development of high-quality and complex systems (Nicholas & Wooldridge, 2012). Indeed, there have been numerous agent-based applications in a wide variety of domains such as air traffic control, space exploration, information management, business process management, e-commerce, holonic manufacturing, and defence simulation (Nicholas & Wooldridge, 2012). Despite its popularity and attractiveness as a research area, agent technology still faces many challenges in being adopted by the industry and possibly taking over from objects technology as the dominant software development technology (Odell, 2012). A key area of research is Software Engineering methodology: One of the most fundamental obstacles to large-scale take-up of agent technology is the lack of mature software development methodologies for agent-based systems. (Nicholas & Wooldridge, 2012).

Indeed, the development of industrial-strength applications requires the availability of software engineering methodologies. These methodologies typically consist of a set of methods, models, and techniques that facilitate a systematic software development process, resulting in increased although many Agent Oriented methodologies have been proposed, few are mature or described insufficient detail to be of real use. None of them is in fact complete- in the sense of covering all of the necessary activities involved in the development of intelligent agents and is able to fully support the industrial needs of agent-based system development.

In addition, although a large range of agent-oriented methodologies are available; there is a lack of appropriate study on evaluation and comparison of the existing methodologies. Several

approaches have been applied to review and classify a large range of agent-oriented methodologies or perform comparisons on a small number of methodologies.

Unfortunately, such evaluations or comparisons are mostly subjective and are solely based on inputs from a single assessor (or group of assessors).

Furthermore, numerous key issues relating to software engineering generally and the agent-oriented paradigm specifically are not addressed in those studies. We believe that the area of agent-oriented methodologies is growing rapidly and that the time has come to begin drawing together the work from various research groups with the aim of developing the next generation of agent-oriented methodology.

A crucial step is to understand the relationship between various key methodologies, including each methodology's strengths, weaknesses, and domain of applicability. An important part of this step is also identifying the key commonalities and differences among the existing agent-oriented methodologies. By doing so, we may contribute towards building a unified approach to agent oriented software development.

Many diverse Agent Oriented Software Engineering (AOSE) approaches and methodologies have been proposed (Jayaratna, 2012). Each of the methodologies has different strengths and weaknesses, and different specialized features to support different aspects of their intended application domains. Clearly no single methodology is “one size fits all. However, as application complexity grows, we expect future projects to have an increasingly large number of aspects that cannot be addressed by a single methodology alone. To provide engineering support for such projects, specialized features to address different aspects must be brought together from different methodologies in a consistent fashion.

It is useful to identify and standardize the common elements of the existing methodologies.

The common elements could form a generic agent model on which specialized features might be based. The remaining parts of the methodologies would represent “added-value” that the methodologies bring to the common elements, and should be “componentized” into modular features. The small granularity of features allows them to be combined into the common models in a flexible manner. By conforming to the generic agent model in the common elements, we expect the semantics of the optional features to remain consistent.

In Agent Technology, individual agent-oriented methodologies are useful for restricted situations, a more flexible approach can be found in the use of an enhanced model. Using an underpinning Meta model, a repository of method fragments can be built up and, from this, a selected number of fragments can be abstracted to form an organization-specific or project specific methodology. Hybrid agent oriented methodology for intelligent agent system will be created from existing individual agent oriented methodologies, such as Prometheus, is demonstrated with further enhancements from other methodologies such as ROADMAP.

Enhanced Multi-Agent System Development (EMASD) methodology, consisting of the common elements identified from MaSE, Prometheus and ROADMAP.

1.2 The Statement of Problem

The multiplicity and variety of agent oriented methodologies result in the following problems:

Most of the research that examined and compared properties of agent-oriented methodologies suggested that none were completely suitable for industrial development of multi-agent systems (Tran and Law, 2010). Therefore, selecting methodology for developing an agent-based system/application becomes a trivial task, in particular for industrial developers which hold specific requirements and constraints.

None of the existing agent-oriented methodologies has itself established as a standard nor have they been commonly accepted (Tran and Law, 2010). As long as there are no standard definitions of an agent, agent architecture, or an agent language, we could think that the existing methodologies will only be used by individual researchers to program their agent-based application using their own agent language, architectures, and theories. The lack of standard agent architectures and agent programming languages is actually the main problem to define models and put them into operation, or providing a useful “standard” code generation. Since there is no standard agent architecture, the design of the agents needs to be customized to each agent architecture. Nevertheless, the analysis models are independent of the agent architectures. They describe what the agent-based system has to do, but not how this is done (Iglesias, 2012). Most of the existing methodologies suffer from a gap between the design models and the existing implementation languages (Scott, 2012).

It is difficult for a programmer to map the developed complex design models onto an implementation. To close this gap, a methodology should either provide refined design models that can be directly implemented in an available programming language or use a dedicated agent-oriented programming language which provides constructs to implement the high-level design concepts.

Most of the existing methodologies do not include an implementation phase. Methodologies that include an implementation phase as an essential phase of its methodology, such as the Tropos methodology, provide an explicit implementation language. This implementation language however does not explain how to implement reasoning about beliefs, goals, plans and reasoning of communication (Scott, 2011).

This leads to difficulties using the methodology. The implementation phase should describe in detail how the belief, goals, plans, and interactions are to be implemented using a specific agent programming language.

One important characteristic of agent behaviour is that the agent may play one or several roles in the system. A few of the existing methodologies support role concept. None of them takes into account that an agent may play more than one role in a system (Rumbaugh, 2011). This aspect gives the agent more flexibility and the ability to complete the work mandated. The agent can benefit from combining the goals and plans for the roles played by the agent and the latter can be exploited to carry out its work in the system.

Therefore, all of the above stated problems provide us with a motivation to come up with a novel approach for development of Intelligent Agents.

1.3 Aim and Objectives of the Study.

The aim of this study is to develop an Enhanced Software Engineering Methodology for Analysis and Design of Intelligent Agents.

Specific objectives of the study include:

- 1) Investigate three prominent agent-oriented methodologies.
- 2) Present the enhanced framework for the Development of Intelligent Agents.
- 3) Apply the enhanced model in developing an Intelligent Agent called modeling agent as a proof of concept.

4) Conduct Performance evaluation of the proposed model with existing Software Engineering Methodology.

1.4 Significance of the Study

The potential of agent-based systems is large and it remains to exploit this new technology. The success of agent-based systems in the future, in contrast, in our view depends on a thorough analysis of its conceptual basis and the construction of structuring principles and methods for the design of agent systems (Lin & Winkoff, 2012). Despite the many proposals in the literature to support the construction and design of agents by means of specific agent architectures or programming languages. (Lin & Winkoff, 2012). Some promising efforts in providing a methodology for agent-oriented software engineering derived from object-oriented methodologies have been proposed (Booch, 2014).

Furthermore, a Traveller Agent System was developed in this research work using the developed enhanced model. Based on an observation that the coalescence of groups of Object Oriented methodologies in the late 1990s led to an increased take-up by industry of the object-oriented paradigm for system development and project management, this research aims to encourage first the coalescence and collaboration between research groups and then, hopefully, more rapid industry adoption of Agent Oriented methodological approaches (Lin & Winkoff, 2012).

Moreover, the significance of this research work cover the identification of those predominant and tested AO methodologies, characterize them, analyze them, and seek some method of unification and consolidation with the hope that, in so doing, the community of scholars supporting Agent Oriented methodologies will soon be able to transfer those innovative ideas into industry acceptance. The importance and value of this research lies in its inherent abilities to conveniently, effectively and efficiently develop a One-Size-fits-all methodology for intelligent agent. Software integration such as the intelligent agent systems is package that is implemented to provide search engine, process data and transfer it to where it will use. It can also help researchers to examine the similarity and the differences among existing agent-oriented methodologies and to analyze the needed attributes of such methodologies. Additionally, setting a scale for grading agent-oriented methodologies, and using the scale in

conjunction with our framework, may result in a selection of the better methodologies, gradually reducing their number (Wooldridge,2012). This selection may eventually converge to a small set of the most-fit agent-oriented methodologies, possibly leading to standardization.

1.5 Scope of the Study.

This study is meant to cover the following individual agent-oriented methodologies **MaSE**, **ROADMAP**, and **Prometheus** that are restricted to specific domains and provide means for addressing these problems by developing an enhanced model. This comparison and evaluation framework may be used by organizations to select a methodology for developing agent-based applications. Such an evaluation would ideally be carried out using a complete framework via different evaluation methods such as feature analysis, structured analysis, etc. Therefore, we try to construct a framework that is complete in the sense that it can be used to fulfill our purposes. It is also important that the evaluation is as objective as possible.

Furthermore, the Enhanced Multi-Agent System Development (EMASD) methodology is designed to work for cross-boundary systems (semi- open systems) where the agent society itself is closed (i.e. the types and behaviours of agents defined in the system are determined beforehand) but external agents may interact with members of the society via the defined and used protocols. Enhanced Multi-Agent System Development (EMASD) methodology is focused on small and medium sized systems which are based on the BDI agent architecture, which is used to design agents for the development process.

Moreover, this methodology follows the traditional top-down approach that starts by identifying the system requirements and ends up by implementing the system

1.6 Limitations of the Study.

Some of the constraints that may have in one way or another affected the outcome of this work include:

The difficulty in obtaining a complete evaluation framework in practice. We tried to construct a framework that is complete in the sense that it can be used to fulfill our purposes. It is also

important that the evaluation is as objective as possible. This means that the results of the evaluation reflect a wide range of viewpoints.

1.7 Definition of terms.

- 1) **Agent:** One that is authorized to act for another. Agents possess the characteristics of delegacy, competency, and amenability.
- 2) **Information Society (IS)** – A country or region where information technology has been fully exploited and is part of everyday life as an enabler of information sharing, communication and diffusion.
- 3) **Information Technology (IT)** – Embraces the use of computers, telecommunications and office systems technologies for the collection, processing, storing, packaging and dissemination of information.
- 4) **Internet Exchange Point (IXP)** – It is a “peering point” for Interconnecting ISPs and/or other IXPs for the purpose of localizing national traffic routing as opposed to using international routes to accomplish Inter-ISP traffic flow.
- 5) **Internet Service Provider (ISP)** – Also known as Internet Access Providers – Is a company that provides infrastructure for access to the Internet or for interconnecting other ISPs and content-based or application-based services on the Internet.
- 6) **Knowledge Based Economy (KBE)** – A country or region where ICT is extensively used to enhance knowledge so that higher human capital brings further improvement to the economy.
- 7) **Delegacy:** Discretionary authority to autonomously act on behalf of the client. Actions include making decisions, committing resources, and performing tasks (Lin & Winkoff, 2012)
- 8) **Competency:** The capability to effectively manipulate the problem domain environment to accomplish the prerequisite tasks. Competency includes specialized communication proficiency (Lin & Winkoff, 2012).
- 9) **Amenability:** The ability to adapt behavior to optimize performance in an often non-stationary environment in responsive pursuit of the goals of the client. Amenability may be combined with accountability (Lin & Winkoff, 2012).

CHAPTER TWO

LITERATURE REVIEW.

2.1 Review of Related Literature

2.1.1 Conceptual Framework

The term agent has become one of the most striking topics in computer science research. The term “Software agent” leads to a wide argument of what a software agent is, and of how it could be clearly distinguished from a program.

Examining the question, “What is a software agent?” raises many arguments about what a software agent is, and what the difference between a software agent and computer program is. Researchers have proposed many definitions for the concept of a software agent. Each of them introduced his/her definition according to their point of view. Some of them concentrated on artificial intelligence approaches, others concentrated on software engineering approaches. We concentrate on the definitions that are well known and most accepted by agent researchers such as M. Wooldridge and N. Jennings etc.

A general definition of a software agent is that it is a computer program that exhibits the characteristics of an agency or a software agency. According to Krupansky's Foundations of Software Agent Technology the software agent is defined as: “A software agent (or autonomous agent or intelligent agent) is a computer program which works toward goals (as opposed to discrete tasks) in a dynamic environment (where change is the norm) on behalf of another entity (human or computational), possibly over an extended period of time, without continuous direct supervision or control, and exhibits a significant degree of flexibility and even creativity in how it seeks to transform goals into action tasks. (Krupansky, 2010)

Here we present another software agent definition, which clearly distinguishes a software agent from any other program. Wooldridge and Jennings (2012) proposed two notions of agency; a weak notion and a strong one. A weak notion of agency is that of hardware or more frequently software-based computer system that provides the following properties:

1. **Autonomy:** is when agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;
2. **Socialability:** is the ability of agents to interact with other agents (and possibly humans) via some kind of agent-communication language;
3. **Reactivity:** The ability of agents to perceive their environment and respond in a timely fashion to changes that occur in it. Here, the agent environment may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, or perhaps all of these combined.
4. **Pro-activeness:** is when agents do simply act in response to their environment, but they are also able to exhibit a goal-directed behaviour by taking the initiative.”

A strong notion of agency is also widespread in artificial intelligence. In addition to the weak notion, the strong notion also uses mental components such as belief, desire, intention, and knowledge and so on. This definition illustrates autonomy of an agent, sensing and acting on a finite environment that an agent is a part of (Wooldridge, 2012). Many agent definitions are proposed by software agents research Maes (2011), Russel& Norvig(2011), Roel(2011), and Wooldridge and Jennings (2012). Most of these definitions were based on certain situations, certain conceptions, or to solve certain problems according to a researcher’s point of view. As an attempt to cover most of the existing agent definition patterns, we conclude the following agent definition and its related concepts to be the foundation for the new methodology. These definitions are stated as follows:

Agent: A persistent computer system that carries out some set of tasks on behalf of a user or a computer system and is capable of:

1. Functioning with some degree of autonomy (autonomy means the agent ability to work with minimum intervention by the real user. The autonomous agents have control over their tasks and resources and will take part in cooperative activities only if they chose to do so).
2. Interacting with others (humans or agents) via specific agent communication language.
3. Perceiving its environment through sensors, acting on the environment, and reacting to the changes of the environment through effectors.

4. Employing its knowledge to make decisions.
5. Cooperating with others (humans and agents) either by negotiation or coordination to achieve common goals.
6. Realizing its goals by performing suitable roles and following suitable plans.
7. Gaining knowledge from experience to store the successful plans.
8. Being adaptable with the environment changes by responding in a timely fashion.
9. Having initiative (self-starting).

Agent role: A set of actions and activities that are assigned to, or expected of an agent to be able to perform in the system. In other words, a role represents an agent behavior that is recognized, providing a means of identifying and placing an agent in a system. The distinction between an agent and a role is that an agent model describes characteristics that are inherent to an agent, whereas a role describes characteristics that an agent takes on. For each agent there is at least one role that should be performed in the system. For each role, there is at least one responsibility that should be performed by this role. For each responsibility, there exists a trigger, which is possibly triggering an action that belongs to the agent capabilities. Responsibilities of a role represent the main activities or tasks that the role performs in order to realize the objectives in the system (Jennings& Wooldridge, 2012).

Agent knowledge: What each agent knows about the environment state, but also what each agent knows about other agents. Agent knowledge represents the informational state of the agent about the environment including it and other agents. It includes agent beliefs and goals.

Agent beliefs: Facts that are believed to be true about the working environment. An agent's beliefs are knowledge, which constitutes a description of the world. An agent's beliefs may be taken to explicitly represent the agent's working environment or even about the agent it or other agents. Using the term belief rather than knowledge recognizes that what an agent believes may not necessarily be true and in fact may even change in the future (Jennings& Wooldridge, 2012).

Agent goals: Goal is defined as an end state, something to be achieved. It describes, "What is to be done". It is the destination itself and not a recipe for how to reach that destination.

Agent goals are informational states of what it is planned to be achieved. The goals represent a mechanism, which leads the agent to achieve its tasks in an orderly and smooth way.

In order to describe the goal the following two questions need to be answered. When are goals initiated or started? When are goals considered satisfied? The goal is started or initiated when its precondition(s) is satisfied. The goal is considered satisfied if and only if at least one of its plans is satisfied then its post condition(s) is satisfied. These pre- and post-conditions are considered the beliefs of the agent. The agent goals are classified into two types of goals long-term and short-term goals. Long-term goals are ones that the agent will achieve over a longer period. Long-term goals often are the most meaningful and important goals. One problem, however, is that the achievement of these goals is usually far in the future. Therefore, the agents should stay focused and maintain a positive attitude towards reaching these goals. Short-term goals are ones that the agent will achieve in the near future. Long-term goals can be decomposed to hierarchical sub-short-term goals. Short-term goals will move the agent along towards its long -term goals. Identifying the following short-term goals will help the agent to create a clear picture of where it is going (Wooldridge & Jennings, 2012).

Plans: An agent's view of the way a modeled agent will achieve its goals. A plan is an organized set of tasks the agent will do to achieve its goals. Each plan is composed of a set of tasks. These tasks will implement the plan and complete the required work (Wooldridge & Jennings, 2012).

Task: An atomic piece of work to be done. It identifies how things are to be done and is clearly seen as an atomic work unit. A Task represents the miniature action that is performed by the agent, which cannot be decomposed into sub-actions. (Wooldridge & Jennings, 2012).

Agent decision: An agent is capable to decide what actions to perform based on its plans, knowledge, and beliefs. An agent decides to perform actions that will change the environment situation, and by doing so, the agent's goals are committed to be satisfied (Wooldridge & Jennings, 2012).

Reactive agent: An agent that reacts to incoming events perceived in the environment. A reactive agent answers to an event by a pre-defined action.

Proactive agent: agents do not simply act in response to their environment; they are able to exhibit goal- directed behavior by taking the initiative. An agent generates goals, tries to achieve them, and does not depend on events occurring in the environment.

Environment: A set of components that describe all the features of the system and its behavior. These components affect each other. These components are stated as follows: the agents that act on this environment, the events that happen in the environment, the interactions that could take place between the agents, and the dependencies between agents in the system. In fact, the environment constitutes the MASs (Wooldridge & Jennings, 2012).

Events: Actions that happens at a given place and time. The action uniquely identifies the event. Location is the place where the event happens. Time is that time when the event happens. They are all perceived and afterwards processed by agents and may launch or trigger plans or goals that should be selected to achieve. An agent may react to events that change its knowledge. Events may change the agent's knowledge because its perception of the environment has changed. A triggering event defines which events may lead to the execution of a particular plan in order to achieve a particular goal (Wooldridge & Jennings, 2012).

Triggers: Represent incoming information from the environment to the agent. The agent reacts according to this information in terms of actions. An agent perceives its environment through sensors that describe triggering information. This triggering information could be events or agent belief changes about the state of the environment. These events or agent belief changes trigger the agent to do actions that may update the agent's knowledge, known as beliefs and goals (Wooldridge & Jennings, 2012).

Agent interactions: The way in which agents exchange information. This exchange amounts to a message passing from one agent to many agents or humans. Interaction enables agents to negotiate and coordinate in achieving their tasks or common goals. These interactions are

managed by communication acts (messages) and organized by communication protocols (Wooldridge & Jennings, 2012).

Message: A unit of information or data that is transmitted from one agent to another. A message can be defined as any information sent as an agent, which interacts with another (Wooldridge & Jennings, 2012).

Protocol: A sequence of rules, which guide the interaction that take place between several agents. These rules determine the format and transmission of messages exchanged between agents. These rules define what messages are possible for any particular interaction state. The set of possible messages is finite. (Wooldridge & Jennings, 2012).

Agent services: A service is a task that an agent is potentially willing to perform on behalf of other agents. A set of services is associated with each agent. For each service that may be performed by an agent, it is necessary to specify its properties such as name, cost, etc. An agent possesses skills (services), which the agent can offer to other agents (Wooldridge & Jennings, 2012).

Agents' relationships: Denotes the degree of influence agents have over each other. They allow us to construct management hierarchies (i.e. who is the boss of who). For example, an agent dependency relationship is a relationship between two agents, a depended, and a dependant. The dependant agent depends on another agent (the dependee) to do or provide something (dependum) in order that the dependant may achieve some goal (Wooldridge & Jennings, 2012).

Multi-Agent System (MAS): A system composed of several agents, capable of reaching goals that are difficult to achieve by an individual agent system.

A multi-agent system is a system showing the following characteristics (Wooldridge & Jennings, 2012):

1. Each agent has incomplete capabilities to solve a problem.
2. There is no global system control.

3. Data is decentralized.
4. Computation is asynchronous.

When several agents interact, they may form a multi-agent system. Characteristically such agents will not have all data or all methods available to achieve an objective and thus will have to collaborate with other agents.

In addition, there may be little or no global control and thus such systems are sometimes referred to as “swarm systems”. As with distributed agents, data is decentralized and execution is asynchronous. MASs evolved from Distributed Artificial Intelligence (DAI), Distributed Problem Solving (DPS), and Parallel Artificial Intelligence (PAI), thus inheriting all characteristics from DAI and Artificial Intelligence (AI). Generally, multi-agent systems can show plainly self-organization and complex behaviors (Wooldridge & Jennings, 2012).

There are two types of agent systems: Closed multi-agent systems and open multi-agent systems:

Closed multi-agent systems: based on static design with components and functions, which are required to be known in advance. In such systems, there is a common language for communication between agents. Each agent is developed as an expert in a particular area has the ability to solve problems, skills, and knowledge. For example, MAS is organization built to contain a group of agents who represent different departments within the organization. Each of these agents has different skills and roles (Wooldridge & Jennings, 2012).

Open multi-agent systems: Often do not have static design beforehand there are only independent agents inside the system. Agents would not necessarily know the experience of other agents or services they offer. Therefore, it is requested that there should be a mechanism to identify agents. Agents may be uncooperative, malicious and unreliable in open systems. An example of open systems is the e-commerce market where it is not necessary for agents representing clients to look for providers in order to obtain services or products they need. This is often done through mediator agents and brokers specially designed for this purpose and who are working as a directory (Wooldridge & Jennings, 2012).

2.1.2 Agent Architectures

Maes proposed agent architecture as a particular methodology for building agents. It specifies how the agent can be decomposed into the construction of a set of component modules and how these modules should be made to interact (Maes, 2011). Agent architectures present a higher level of abstraction for building and viewing agent systems. A numeral of existing agent architectures were reviewed, decomposed using object-oriented techniques, and then classified based on their components, connectors, and overall structural pattern. Five agent architectural styles were found using this approach: Belief Desire Intention (BDI), reactive, planning, knowledge-based, and deliberative. The following sections review the most common types of agent architectures and the components they are constructed from.

2.1.2.1 BDI Agent Architecture

The BDI architecture is one of the most well-known and studied software agents' architectures (Juan, Pierce, & Sterling, 2011). This architecture consists of four basic components: beliefs, desires, intentions, and plans. In this architecture, the agent's beliefs represent information that the agent has about the world, which in many cases may be incomplete or incorrect (Wooldridge & Jennings, 2012). The content of these beliefs can be anything from knowledge about the agent's environment to general facts an agent must know in order to act rationally. The desires of an agent are a set of long-term goals, where a goal is typically a description of a desired state of the environment. An agent's goals simply represent some desired end state. These goals may be defined by a user or may be adopted by the agent. New goals may be adopted by an agent due to an internal state change in the agent, an external change of the environment, or because of a request from another agent. State changes may cause goals or plans to be triggered or new information to be inferred that may cause the generation of a new goal. Requests for information or services from other agents may cause an agent to adopt a goal that it currently does not possess.

An agent's desires provide it with motivations to act. When an agent chooses to act on a specific desire that desire becomes an intention of the agent. The agent will then try to achieve these intentions until it believes the intention is satisfied or the intention is no longer achievable (Wooldridge & Jennings, 2012). The intentions of an agent provide a commitment

to perform a plan. Although not mentioned in the acronym, plans play a significant role in this architecture. A plan is a representation outlining a course of action that, when executed, allows an agent to achieve a goal or desire.

2.1.2.2 Reactive Agent Architectures

Perhaps the simplest among the most widely used agent architectures are reactive architectures. Wooldridge and Jennings (2012) describe a reactive architecture as an architecture that does not have a central world model and does not use complex reasoning. Unlike knowledge-based agents that have an internal symbolic model from which to work, reactive agents act by stimulus-response to environmental states. The agent perceives an environmental change and reacts accordingly. Reactive agents can also react to messages from other agents.

Although reactive agents are basic and can only perform simplistic tasks, they do form a building block from which other, more complex agents can be built. By adding a knowledge base to a simple reactive agent, the agent becomes capable of making decisions that take into account previously encountered state information. By adding goals and a planning mechanism, we can create a rather complex goal directed agent. Although complex patterns of behavior can be developed using reactive agents, their primary goals usually consist of being robust and having a fast response time. Most agent architectures contain a reactive component of some kind. However, they are not actually truly reactive agents. Majority of reactive architectures can be modeled using a basic “IF-THEN” rule structure.

2.1.2.3 Planning Agent Architecture

A number of researchers present different definitions for planning, but all result in the same essential facts. Planning is the process of formulating a list of actions in order to achieve a specified goal (Pollack, Fausto, & Anna, 2010). In artificial intelligence, a planner uses knowledge about the actions it may perform and their consequences. It uses this as well as knowledge about the environment, to formulate a list of acceptable state transforming operators that can transform the agent from an initial state into a goal state. As seen in BDI, planning architectures are usually embedded in other agent architectures to determine the

actions that an agent will perform. Within a given agent architecture, plans may be either synthesized dynamically or predefined in advance and placed in a plan library. In general, plans come in two types; total order and partial order. Total order plans simply consist of a list of steps that an agent must follow to accomplish a set goal. These steps have a definite order that must be followed for the goal to be achieved. Partial ordered plans may have some steps ordered while the order of other steps is arbitrary and inconsequential to reaching the goal. At one more level of abstraction, plans can be fully or partially instantiated. The steps of a plan are generally operators containing parameters that need to be defined to a set value in order for the operator to function. A fully instantiated plan is one in which all of these parameters are defined to a set value.

Russell and Norvig (2011) state that a plan is a formally defined data structure that contains the following components:

- A set of plan steps. Each step is one of the operators of the problem.
- A set of step ordering constraints.
- A set of variable binding constraints.
- A set of causal links to record the purpose(s) of steps in the plan

2.1.2.4 Knowledge-Based Agent Architectures

Even though the BDI architecture has a knowledge base, a large number of architectures that exist are built around a centralized knowledge store.

In general, these are referred to as knowledge-based or expert systems.

Knowledge-based systems use data structures consisting of explicitly represented problem-solving information. This knowledge can be viewed as a set of facts about the world. Three aspects of knowledge-based systems, which make them powerful, are:

1. They can accept new tasks in the form of explicitly described goals.
2. They can achieve competence quickly by being told or learning new knowledge about the environment.
3. They can adapt to changes in the environment by updating the relevant knowledge (Russell & Norvig, 2011).

In general, knowledge-based systems represent knowledge using a formal declarative language. The use of declarative language allows knowledge to be added or deleted from the knowledge base quickly and easily without affecting the rest of the system. Using a declarative language such as first-order logic also allows new information to be derived from the current knowledge stored in the system using inference mechanisms. An inference mechanism can perform two actions. First, given a knowledge base, it can generate new sentences that are necessarily true, given that the old sentences are true. Second, given a knowledge base and a sentence, it can determine whether the sentence was generated by the knowledge base or not (Russell & Norvig, 2011). The relation just described between sentences is called entailment and is used a great deal in knowledge-based systems.

2.1.2.5 Deliberative Agent Architectures

The deliberative agent architecture contains an explicitly represented, symbolic model of the world. Decisions (for example about what actions are to be performed) are made via logical reasoning, based on pattern matching and symbolic manipulation (Genesereth & Nilsson, 2011). In order to build an agent in this way, there are at least two important problems that need to be solved:

1. The transduction problem: that of translating the real world into an accurate, adequate symbolic description, in time for that description to be useful.
2. The representation/reasoning problem: that of how to symbolically represent information about complex real-world entities and processes, and how to get agents to reason with this information in time for the results to be useful.

2.1.3 Agent-Oriented Software Engineering

Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (IEEE Standard Glossary of Software Engineering Terminology, 2010). To date, a wide range of software engineering paradigms (e.g. structured programming, object-oriented paradigms, component-ware approaches, etc.) have been proposed with the aim of either facilitating the engineering process of producing software or increasing the

complexity of the applications that can be built (Hans,2012). Among them, Agent-Oriented Software Engineering (AOSE) has been emerging as a promising approach to achieve these objectives. By definition, AOSE is the application of agents to software engineering in terms of providing a means of analyzing, designing, and building software systems (Wooldridge & Jennings, 2012). In this section, we briefly highlight the trend of software engineering paradigms over the past few decades with the purpose of indicating why and how AOSE has the potential of being an efficient and powerful software engineering approach.

In the early days of programming languages, programmers wrote programs at a level close to the machine. They used to view the whole system as a basic unit of software. Hence, modular design did not exist in those days. This technique is, however, only practical for simple applications. As time went on, software systems became more complex and the old ad hoc programming approach became impractical. Programmers needed to organize their code in a more structured way, making it easier to manage. Structured programming was introduced to answer that demand. According to this software engineering paradigm, the basic units of software are procedures or subroutines. These subroutines are designed to perform a specific task and can be reused in various situations.

Additionally, the concept of encapsulation (i.e. the hiding of implementation details) was introduced since the code inside each subroutine is wrapped" and its state is only determined by external given arguments. This new approach promoted modular design and consequently eased the process of developing and maintaining software. Together with the explosion of information technology in the 80s and the 90s, there was a large demand for having a wide range of software applications that are both of a high quality and meeting complex requirements. However, structured analysis techniques were unable to deal with that demand. In that context, the object-oriented approach was introduced. Its effectiveness resides in many aspects such as information hiding, data abstraction, encapsulation, and concurrency (Wooldridge & Jennings, 2012).

Object-orientation also attempts to close the gap between the real world and its representation, i.e. the software application, by modelling real entities as objects. These useful properties of

object-oriented paradigms bring better maintenance, improved modifiability and increased functionality to software engineering (Wooldridge & Jennings, 2012). As a result, object-oriented programming and design has quickly become the dominant software engineering approach in both academia and industry. Following it, there have been several paradigms which expand object-orientation such as component-ware, design patterns, and application frameworks. They also contribute to an attempt to achieve software reuse.

Even though object-orientation has proven its usefulness and power as a software engineering paradigm, it still seems not to be able to cope with the increasing complexity of software systems. These complexities result from different sources (Wooldridge & Jennings, 2012). One of them is the rapid and radical change of the information system environment. Software systems are now becoming not just more inter-connected, and more decentralized but also more interdependent. These changes are amplified with the increasing popularity of the Internet and the World Wide Web. Furthermore, the complexities within software come from the increasing number of interactions between subcomponents, which is in fact an inherent property of large systems.

Therefore, building high-quality and industrial-strength software becomes more and more difficult. The concept of agents as being autonomous, sociable, flexible, etc. promises a new solution to those issues because it leads to a new way of thinking about software systems. Such a system is no longer a collection of passive objects. Therefore, there has been a growth of interest in agents as a new paradigm for software engineering (Wooldridge & Jennings, 2012).

The credentials of agent-based approaches as a software engineering paradigm are two-fold. Firstly, the technical embodiment of the agency can result in advanced functionalities. Multiagent systems which consist of autonomous agents can expand the complexity and quality of the real-world applications (Wooldridge & Jennings, 2012).

In fact, the autonomy aspect of multiagent systems suits the highly distributed environment where different autonomous agents within a system act and work independently to each other. In addition, the inherently robust and flexible characteristics of multiagent systems allow them to work in a more dynamic and/or open environment with error-prone information sources.

The reliability and failure-tolerance of the system are increased and so is its ability to adapt to changes in the environment.

Secondly, agents with their rich representation capabilities promise more effective and reliable solutions for complex organizational processes (Wooldridge & Jennings, 2012).

Jennings and Wooldridge in pointed out agent-orientation provides three essential tools which assist the developers to manage complexity: decomposition, abstraction, and organization. Firstly, they show that agent-oriented decomposition is the effective means of decomposing the problem space of a complex system.

Secondly, they prove the suitability of agents as an abstraction tool, or a metaphor, for the design and construction of systems.

Thirdly, they indicate the appropriateness of applying the agent-oriented philosophy for modelling and managing organization relationships to deal with the dependencies and interactions that exist in complex systems. Various researchers and software engineers have also come to that agreement (Wooldridge & Jennings, 2014).

2.1.4 Agent-Oriented Methodologies

In order to be able to perform a comprehensive literature review for the agent-oriented methodologies, “what the meaning of the methodology is” needs to be precisely defined before starting this discussion. A good methodology should provide the models for defining the elements of the multi-agent environment (agents, objects and interactions).

A good methodology should also provide the design guidelines for identifying these elements, their components and the relationships between them. Any good methodology aims to provide a set of guidelines that covers the whole lifecycle of the system development. The guidelines should cover both the technical as well as the management aspects. Agent systems have been increasingly recognized as the next important software engineering approach. Methodologies are the means provided by software engineering to facilitate the process of developing software and, as a result, to increase the quality of software products.

By definition, a software engineering methodology is:

A collection of procedures, techniques, tools and documentation aids which will help the systems developers in their efforts to implement a new information system. A methodology will consist of phases, themselves consisting of sub-phases, which will guide the systems developers in their choice of techniques that might be appropriate at each stage of the project and also help them plan, manage, control and evaluate information system projects (Avison & Fitzgerald, 2013). It is also important for a methodology to provide notations and modeling techniques, which allow the developers to model the target system and its environment. Notations are a technical system of symbols used to represent elements within a system. A modeling technique is a set of models that depict a system at different levels of abstraction and the different aspects of the system. Furthermore, an agent methodology should support software engineering issues such as: preciseness, accessibility, expressiveness, modularity, domain applicability, and scalability. Preciseness makes sure that the semantics of modeling techniques of the methodology are unambiguous in order to avoid misinterpretation of the developed models by those who use it.

Accessibility is the understandability of the modeling techniques for both experts and novices. Expressiveness is the ability of the methodology to express the system as whole. It represents the following aspects of the system: structure; encapsulated knowledge; ontology; data flow; control flow; concurrent activities; resource constraints (e.g., time, CPU and memory); the physical architecture; agents' mobility; interaction with external systems; and the user interface definitions. Modularity is the ability to express the methodology in stages. That is, when new specification requirements are added, there is no need to modify previous parts, and these may be used as a part of the new specification. Domain Applicability is the suitability of the methodology for a particular application domain (e.g. real-time, information systems).

Scalability is the ability of the methodology or subsets thereof, to be used to handle various application sizes (Brazier, Jonker, & Treur, 2010). In addition to the methodology, there are also tools that support the use of such methodologies. For example, diagramming editors help developers draw symbols and models, which are described in the methodology.

The Rational Unified Process (RUP) is a good example of a software engineering methodology (Buhr, 2008). It uses the notation described in the Unified Modeling Language (UML) (Burmeister, 2010) and its typical tool support is called Rational Rose. A robust methodology needs to contain sufficient abstractions to entirely model and support agents and MASs. Therefore, software engineers should use agent-oriented concepts to describe the methodology. In turn, this can be used to build agent-oriented systems and MASs. Arguably, simple extensions of object-oriented methodologies to represent agent concepts are highly restricted by object concepts. Thus, an agent-oriented methodology needs to concentrate on an organized society of agents playing roles within an environment. This society of agents is interacting according to protocols determined by agents within the system.

2.1.4.1 Classification of Agent-Oriented Methodologies

Agent-oriented methodologies have several roots. They are classified according to the approach or discipline upon which they are based. A common property of these methodologies is that they are developed based on the approach of extending existing methodologies to include the relevant aspects of agents. They are broadly classified into three categories: agent-based methodologies, object oriented-based methodologies and their extensions, and knowledge engineering-based methodologies (Allan, 2012; Henderson-Sellers and Giorgini 2005). Figure 2.1 illustrates the classifications of agent-oriented methodologies.

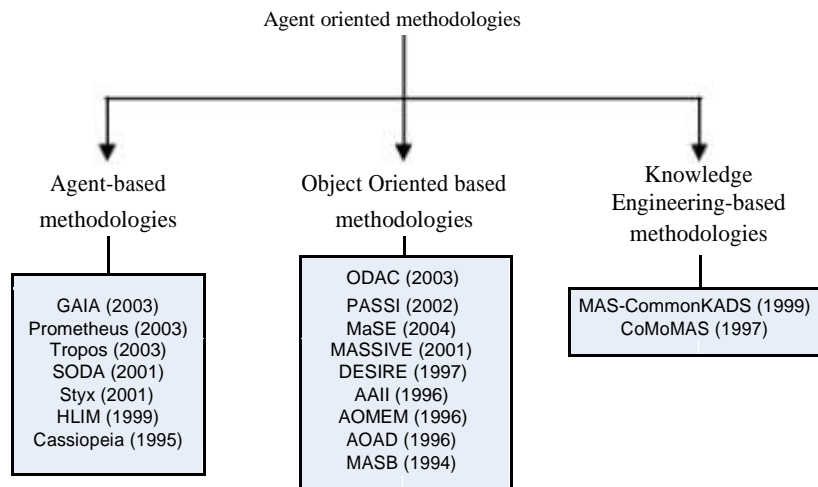


Figure 2.1 Classifications of Agent-Oriented Methodologies (Source: Allan, 2012)

Agent-based methodologies: There are several methodologies that belong to this category such as: GAIA (Jennings, & Wooldridge, 2012), HLIM (Elammari & Lalonde, 2010), Tropos (Bobkowska, 2005), Prometheus (Padgham & Winikoff, 2012), SODA (Omicini, 2012), Styx (Allan, 2012), and Cassiopeia (Collinot, & Treur, 2010). The developers of such methodologies urge that the agent concept should be established without dependency on other traditional methodologies, such as object-oriented methodologies. The main reason is the inherent differences between the two entities; agents, and objects. This is because agents have a higher level of abstraction than objects. Object -oriented approaches cannot offer the same properties as agents do. They also fail to properly capture the autonomous behavior of agents, interactions between agents, and organizational structures (Allan, 2012). In fact, the notions of autonomy, flexibility, and pro-activeness can hardly be found in traditional object-oriented approaches (Odell, 2012). As a result, object-oriented methodologies generally do not provide techniques to model the intelligent behavior of agents (Jennings & Wooldridge, 2012). Therefore, there need to be software engineering methodologies, which are specially tailored to the development of agent-based systems.

Object oriented-based methodologies (Extensions of object-oriented methodologies): The agent-oriented methodologies which belong to this category either extend existing object-oriented methodologies or adapt them to the aim of agent-oriented software engineering. The examples of such methodologies are: ODAC (Glaser, & Francisco, 2010), MaSE (DeLoach, 2012), MASSIVE (Lind, 2011), DESIRE (Brazier, 2010), AAI (Kinny, Georgeff, & Rao, 1996), AOMEM (Kendall, 1996), AOAD (Burmeister, & Cossentino, 2011) and MASB (Moulin, 2011). Some researchers present several reasons for following this approach. Firstly, the agent-oriented methodologies, which extend the object-oriented approach, can benefit from the similarities between agents and objects. Secondly, they can capitalize on the popularity and maturity of object -oriented methodologies. In fact, there is a high chance that they can be learnt and accepted more easily. Finally, several techniques such as use cases and class responsibilities card (CRC), which are used for object identification can be used for agents with a similar purpose (i.e. agent identification) (Iglesias, Garrijo & Gonzalez,2010).

Knowledge Engineering-based methodologies (Extensions of Knowledge Engineering (KE) techniques): There are, however, some aspects of agents that are not addressed in object-

oriented methodologies. For instance, object-oriented methodologies do not define techniques for modeling the mental states of agents. In addition, the social relationship between agents can hardly be captured using object-oriented methodologies. These are the arguments for adapting KE methodologies for agent-oriented software engineering. They are suitable for modeling agent knowledge because the process of capturing knowledge is addressed by many KE methodologies (Iglesias, Garrijo, and Gonzalez 2010). Additionally, existing techniques and models in KE such as ontology libraries, and problem solving method libraries can be reused in agent-oriented methodologies. Examples of such methodologies are: MAS-CommonKADS (Iglesias, Garrijo, Gonzalez & Velasco, 2010) and CoMoMAS (Glaser, 2010). Agent-oriented methodologies should assist the developer in making decisions about the aspects of the analysis, design, and implementation of the agent systems. Some methodologies focus on inter-agent aspects, while others focus on intra-agent aspects. Finally, some methodologies explicitly deal with the environment while others do not. These methodologies differ from each other in many respects. They differ on the software development phases they capture in analysis, design, and implementation phases. In addition, they differ in their premises, covered phases, models, concepts, and the supported multi-agent system properties. Therefore, we selected **three** of the existing agent-oriented methodologies to evaluate. The selection is based on several factors such as the methodology's significance and relevance with respect to the field of agents, and its available resource such as documentation, tool support, etc.

Table 2.1: List of AOSE methodologies introduced before year 2000 Source from Wooldridge & Jennings.

#	Methodology	Year Reference(s)
1	ARCHON	1991 (Cockburn and Jennings, 1996)
2	MADE	1992 (O'Hare and Wooldridge, 1992)
3	DRM	1993 (Singh et al., 1993)
4	TOGA	1993 (Gadomski, 1993)
5	CIAD	1994 (Verharen and Weigard, 1994; Verharen, 1997)
6	Agent Factory	1995 (Collier, 1996, 2002; Collier and O'Hare, 1999; O'Hare and Collier, 1998)
7	AOMfEM	1995 (Kendall et al., 1996)
8	Cassiopeia	1995 (Collinot and Drogoul; 1998, Collinot et al., 1996)
9	AAII (KGR)	1996 (Kinny and Georgeff, 1996; Kinny et al., 1996)
10	AOAD	1996 (Burmeister, 1996)
11	AWIC	1996 (Muller, 1996)
12	CoMoMas	1996 (Glaser, 1996)
13	MASB	1996 (Moulin and Brassard, 1996)
14	MAS-CommonKADS	1996 (Iglesias et al., 1998)
15	ALAADIN	1997 (Ferber, 1997; Ferber and Gutknecht, 1998)
16	AMBSA	1997 (Neal Reilly, 1997)
17	AOIM	1997 (Kindler et al., 1997)
18	CaseLP	1997 (Martelli et al., 1997)
19	DESIRE	1997 (Brazier et al., 1997)
20	Adept	1998 (Jennings et al., 1998)
21	AMBIA	1998 (Gao and Sterling, 1998)
22	AOAaD	1999 (Wooldridge, 1999)
23	HIM	1999 (Elammari, 1999)
24	MaSE	1999 (Deloach, 1999, 2005)
25	MASSIVE	1999 (Lind, 1999, 2001)
26	ZEUS	1999 (Nwana et al., 1999)
27	ASEfIA	2000 (Zamboneli et al., 2000)
28	GAIA	2000 (Wooldridge et al., 2000; Zamboneli et al., 2005)
29	MESSAGE/UML	2000 (Caire et al., 2000; Evans et al., 2001)
30	SODA	2000 (Omicini, 2000)

The three methodologies which were chosen are: **Roadmap, MaSE, and Prometheus.**

In the following sections, we briefly describe each of them. It is noted that most of the examples and gurus given during the discussion of each methodology are based on the design of the same application.

2.1.4.1.1 GAIA (Generic Architecture for Information Availability)

GAIA, proposed originally by M. Wooldridge et al. (2012), where the foundation of analysis is based on an Object Oriented design method called Fusion, from which it borrows terminology and notations.

GAIA is rooted in conceptual organizational modeling (Zambonelli, Jennings, & Wooldridge, 2010) and suggests that developers think about building Agent-based systems as a process of organizational design. The Agent computational organization is viewed similarly to human organization consisting of interacting roles and functions. GAIA is one of the first methodologies which is specifically tailored to the analysis and design of agent-based systems (Wooldridge et al. 2012). Its main purpose is to provide the designers with a modelling framework and several associated techniques to design agent-oriented systems. GAIA separates the process of designing software into two different stages: **analysis** and **design**. Analysis involves building the conceptual models of the target system, whereas the design stage transforms those abstract constructs to concrete entities which have direct mapping to implementation code. Figure 2.1 depicts the main artifacts of each stage: **Role Model** and **Interaction Model** (Analysis), and **Agent Model**, **Services Model**, and **Acquaintance Model** (Design). A detailed description of the process steps which the developers need to follow to build these models is described below.

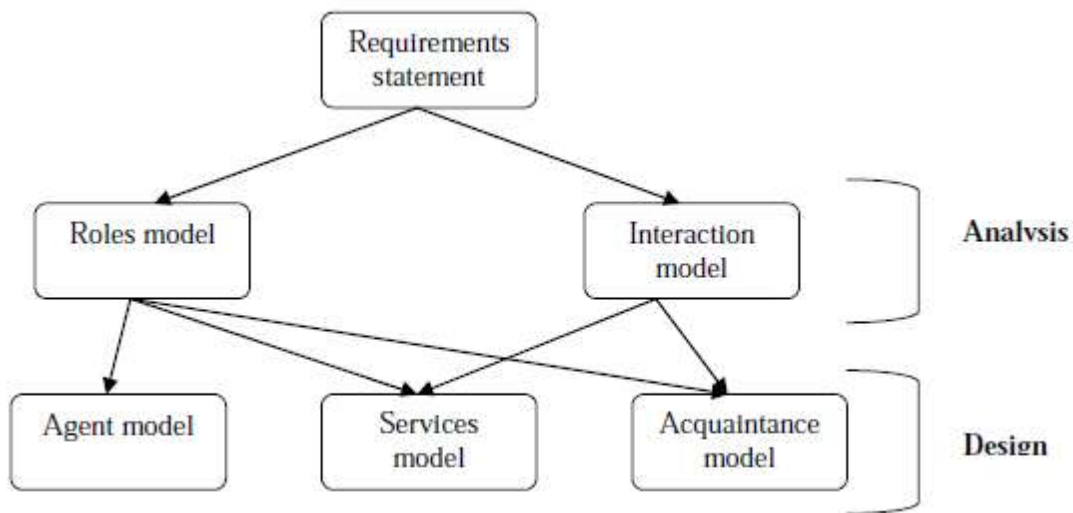


Figure 2.2: Relationship between GAIA's models, source from Wooldridge & Jennings

Analysis

It is noted that, GAIA assumes the availability of a requirements specification. It means that before beginning the GAIA's development process, the analysts need to have a reasonable understanding of what the system should do. These requirements form the overall objectives of the system which influence the analysis and design phase. GAIA encourages the developers to view an agent-based system as an organization. The software system organization is similar to a real world organization. It has a certain number of entities playing different roles. For instance, a university organization has several key roles such as administration, teaching, research, students, etc. These roles are played by different people in the university such as managers, lecturers, students, etc. Inspired by that analogy, GAIA guides the designers to the direction of building agent-based system as a process of organizational design. At the first step of the analysis phase, GAIA requires the analysts to define key roles in the system. At this step, these roles only need to be listed and described in an informal manner. The main purpose of this step understands what roles exist in the system and roughly what they do. GAIA calls the artifact of this step a prototypical roles model.

Different roles in an organization interact with each other to achieve their own goals and also to contribute toward the overall goals of the organization. These interactions need to be defined in the next step of the GAIA analysis phase. The product of this step is the interaction model. This model consists of a set of protocol definitions for each role. Each protocol definition defines the purpose, the initiator, the responder, the inputs, the outputs and the processing during the course of the interaction. The valid sequence of messages involved in a conversation, however, is not required at this stage. Instead, GAIA focuses on the basic nature and purpose of the interaction and abstracts from exact instantiation details. The final step of GAIA analysis phase involves elaborating the key roles identified in the first step. This process includes the identification of permissions of roles, their responsibilities as well as the protocols and activities in which they participate. This detailed description of a role is depicted by a Role Schemata. A set of Role Schemata forms the Role Model, which is considered as the key artifact of this analysis phase. Responsibility defines the functionality of a role and is

divided into two types: liveness properties (something good happens) and safety properties (Arazy, 2012).

Permissions define which resources the agents playing that role can and cannot use when performing a particular action. Activities are private actions which do not involve interactions with other roles. Protocols are actions that involve interactions with other roles and are derived from the protocol model built in the previous step. It is noted that the GAIA analysis process is not purely linear as described. In contrast, the analysts are encouraged to go back to add a new role, new protocols or move forward to add new permissions, activities, etc.

Design

Having finished the analysis phase, the analysts basically complete building the conceptual model of the system with abstract entities. These entities do not necessarily have any direct realization within the system. They can now move to the second phase (i.e. the Design phase) where those abstract entities are transformed into concrete entities which typically have direct counter-parts in the run-time system. The design stage requires the developers to build three models. First, an agent model which includes various agent types is constructed. Agent types are the counterparts of objects in object-oriented approaches. They are basic design units of an agent-based system and their realizations at run-time are agent instances. Agent types in the system under development are defined on the basis of the roles that they play. Therefore, the important feature of this step is to map roles identified in the analysis phase to agent types. A role can be mapped to one or more agent types and vice versa. Some general guidelines are proposed to help this process. For instance, a close relationship between several different roles indicates that they can be grouped together in a single agent type.

The second artifact developed in GAIA's design phase is a service model which depicts the services that each role provides. A service is a coherent, single block of activity in which an agent will engage. Each service should be represented by its properties: inputs, outputs, pre-conditions and post-conditions. Inputs and outputs are derived from the protocol model. Pre-conditions and post-conditions which define the constraints on services are derived from the safety properties of a role. The final model which the designers need to complete is the acquaintance model. It depicts the communication links existing between agent types. It is in

fact a directed graph in which nodes represent agent types and arcs show communication pathways. GAIA does not address implementation and there is no tool support that we, or Michael Wooldridge, one of the authors of GAIA, are aware of.

2.1.4.1.2 THE ROADMAP METHODOLOGY

The ROADMAP (Role Oriented Analysis and Design for Multi-Agent Programming) methodology extends GAIA with the features outlined in the last section. Figure 2.3 shows the structure of the ROADMAP models. The ROADMAP methodology aims to support the engineering of large-scale open systems. It extends the GAIA methodology (Wooldridge, 2012) by introducing use-cases for requirement gathering, explicit models of agent environment and knowledge, and an interaction model based on AUML interaction diagrams (Wooldridge, 2012). The original GAIA role model is also extended with a dynamic role hierarchy. This role hierarchy is carried into design and will have a run-time realization, allowing social aspects to be explicitly modeled, reasoned and modified at run-time.

ROADMAP promotes the view of software systems as computational organizations. Agents in a system are similar to individuals in a human organization, while the roles in ROADMAP encapsulate regulations, processes, responsibilities and team roles by which individuals function within a human organization. They specify, support and constraint an agent's behaviors in the organization. When the expected behaviors in the organization are explicitly represented at run-time, agents can verify each other's behavior, and misbehaving agents can be identified and removed or replaced. A useful level of trust can be established when new agents enter the system. If the new agent has the appropriate knowledge, and behaves according to its role in the correct environment zone, then the other agents in the organization can trust this agent to act to achieve the overall goal of the organization. The relationship between roles and agents is similar to the relationship between interfaces and objects in OO approach. Like interfaces, roles provide an abstract model of the system above concrete implementation of functionalities. However, unlike interface, roles can be changed at run-time given the correct authorization. Instead of an immutable contract of behavior, roles should be considered as a long-term agreement of behavior that can be reasoned and changed. This difference allows a computing organization modeled in roles to be more flexible

The Development Process: The ROADMAP methodology encourages an iterative approach and expects details of the models to be filled in when applicable. During the analysis phase, the six analysis models are created to allow conceptualization of the system as an organization. During the design phase, the initial conceptualization of the system is optimized for the chosen quality goals, such as performance. The original models are modified and refined to reflect the design decisions. In addition, three new design models are created to populate the updated organization with member agents.

Applicability: Before adopting the feature-based approach, ROADMAP prescribes rich models to explicitly deal with roles of agents, the environment and knowledge in the system. The methodology provides strong support for engineering complex open systems, but is less suitable for application not requiring these properties. Consider a stand-alone desktop productivity tool where little knowledge is required outside its functionalities. Given its static nature, simple environment and lack of knowledge, creating the models prescribed by ROADMAP simply causes extra overhead.

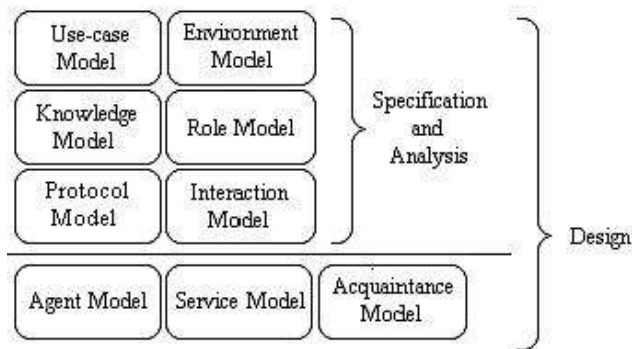


Figure 2.3 The ROADMAP Models: source from Wooldridge & Jennings, 2012

The collection of protocol descriptions is renamed to the protocol model. An interaction model, based on AUML interaction diagrams (Wooldridge, 2012), is added to model the dynamic aspect of the system. The original role model is extended with a role hierarchy. The role hierarchy is represented as a tree and allows arbitrary levels of abstraction. The leaf nodes of the tree are atomic roles. The atomic roles retain their original definition and represent

characteristics of individual agents. All other nodes in the tree are composite roles, defined in terms of other roles, whether atomic or composite. A composite role represents a localized organization of agents. Its attributes model social aspects of that organization, such as the organizational structure, social goals, social tasks and social laws.

All the analysis models are carried into the design phase. They are optimized towards chosen quality goals and updated to reflect design decisions. The extended role model and the protocol model, together define the social aspects within the system as well as characteristics of individual agents, are further carried into implementation. Components of these two models, such as roles, are no longer considered abstract. They are concrete first class entities in design and will have realization in the final implemented system. Similar to roles in GAIA, ROADMAP roles have permissions to access or modify information resources (objects in the environment). To model runtime reflection, we extend the original framework and allow a role to have permissions to access or modify the definition of other roles. Given that roles in ROADMAP have runtime realization, this mechanism allows runtime reasoning, extension and modification of social aspects such as social structure, social laws, and individual agent characteristics and capabilities.

In summary, the system is defined as a computational organization of interacting roles at the analysis stage. The organization is then optimized for quality goals, and populated with agents at the design stage.

THE SPECIFICATION AND ANALYSIS MODELS AND PROCEDURES

In the specification and analysis phase we create the following models sequentially: use-case model, environment model, knowledge model, role model, protocol model and the interaction model. Each model takes all previous models as input. These models are refined iteratively until sufficient information about the system is captured.

The Use-case Model

Creating use-cases has proven to be a very effective and sufficient method to discover requirements. At this stage, this is the only method we use to extract requirements. Adapted from Object Oriented methodologies, the use-case model includes generalized graphical

diagrams and specialized text scenarios. The scenarios outline the system response given a precise sequence of user actions. ROADMAP concepts such as zones and roles can be used in the use-cases. To better capture the richer agent behaviors, the semantics of use-cases are different from the traditional OO approach.

Instead of imagining the user interacting with the software system to do work, we imagine the user interacting with a team of abstract ideal agents. The agents are considered ideal and possess any knowledge, ability or mental states required to provide the best service. The concept of ideal agents with perfect knowledge and ability is unrealistic.

However, with ROADMAP support for open systems, we expect such an ideal agent to be approximated by a dynamic complex multi-agent system. The methodology provides adequate support, allowing the complexity and knowledge in the multi-agent system to scale modularly to any arbitrary level required.

The Environment Model

The environment model is proposed to provide a holistic description of the system environment. Complex open systems usually have highly dynamic and heterogeneous environments. By formally describing the environment, we create a knowledge foundation on which environment changes are handled consistently. The environment model is derived from the use-case model. The model contains a tree hierarchy of zones in the environment, and a set of zone schema to describe each zone in the hierarchy.

In the scenario from Section 2, three primary zones are identified. They are the Internet, local PC, and the physical environment of the house. Sub-zones are then identified by partitioning the Internet to yield the police website and web services, home owner's communication web services at work and commercial web services such as the face recognition service. The physical environment of the house includes sub-zones such as rooms in the house and the garden. A zone schema includes a text description of the zone, and the following attributes: static objects, objects, constraints, sources of uncertainty and assumptions made about the zone. Static objects are entities in the environment whose existences are known to the agents,

with which the agents do not interact explicitly. Objects are similar entities with which agents interact. Sources of uncertainty in the environment are identified and analyzed. The zone hierarchy uses OO-like inheritance and aggregation to relate zones and various objects inside zones. The zone schema is not an exhaustive listing of zone properties. It only contains related information from the use-cases. The developer will go through the scenarios in the use-case model, and for each scenario, identify new zones and update new information into the existing zones. This process builds up a description of the environment iteratively and can be re-used when building future applications in similar environments.

The Knowledge Model

The knowledge model is proposed to provide a holistic description of the domain knowledge in the system. The model consists of a hierarchy of knowledge components, and a description for each knowledge component. From the use-case model and the environment model, the developers identify the knowledge required to deliver the agent behaviors in the appropriate zones for each use-case. The knowledge identified is then decomposed into small coherent blocks. The lifecycles of these knowledge components are analyzed, focusing on how the knowledge component is generated, consumed and stored. The dependencies between knowledge components are analyzed, and the knowledge components are organized into a hierarchy. The creation of the role model happens in parallel and roles are identified from the scenarios. As knowledge components are created, they are assigned to roles; effectively making roles units of knowledge in the system. The knowledge model connects the role model with the use-case model and the environment. When the expected behavior of the system or the environment changes, the knowledge in the roles can be conveniently revised.

The Revised Role Model

The revised role model now consists of two artifacts: a role hierarchy, and a set of role schema describing each role in the hierarchy. The role hierarchy is represented as a tree; Figure 2.3 shows an example. The leaf nodes of the tree are atomic roles. In this example they are D, E, F and G. They retain their original semantics from GAIA and represent characteristics of individual agents.

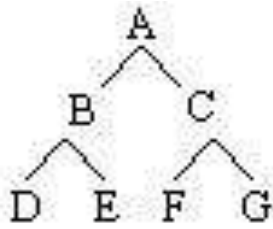


Figure 2.4. A sample Role Hierarchy (Source: Wooldridge, 2012)

All the other roles are composite roles. As shown in the example, they are A, B and C. They are defined in terms of other roles, whether atomic or composite. The mechanism we use is aggregation. The root of the tree, A, is a special composite role; it represents the entire system.

Composing with Aggregation

The semantic of aggregation is different from that of traditional OO approaches. In OO approaches, aggregation usually has a static object containment semantic, assuming only one thread of execution. It takes a bottom-up view and the aggregate has direct control over the components. The class hierarchy is effectively a control hierarchy. To illustrate the autonomous nature of agents and the social aspects in multi-agent systems, a more dynamic and distributed view is needed. Aggregation in the proposed role hierarchy assumes a dynamic teamwork semantic. The individual sub-roles are said to participate in the super-role; each may own a thread of execution, a set of knowledge and functionality. The super-roles are said to involve the sub-roles, with that involvement dynamically adjusted if necessary. A sandwich approach sees sub-role responsibilities interacting to achieve the super-role responsibilities, and sub-role actions interacting so that super-role action emerges. The super-role may not have complete and fine-grained control over sub-roles. The role hierarchy models societies at different levels of abstraction, instead of depicting a control hierarchy.

Inheritance is not used in the methodology for three reasons. First, a relationship expressed as inheritance can be re-expressed as aggregation. Two entities A and B having a common entity C can both aggregate C as a component, rather than inheriting from C as a parent class. Secondly, inheritance represents the “is-a” relationship. This relationship creates more implicit coupling than the “has-a” relationship of aggregation. In a dynamic system with constant

change in organization, inheritance will leave more legacy dependency/coupling between types. This reduces the maintainability and extensibility of the system.

Thirdly, in light of an open system, the architecture of the system is expected to be refactored regularly. Having a uniform mechanism to compose the system instead of two simplifies the refactoring.

Modeling Social Aspects with Composite Roles

A composite role represents a localized organization or society of roles. Its attributes model social aspects, most notably the organization structure, social goals, social tasks or social laws, of that society. The revised role schema is identical to the original schema with the addition of two attributes, namely the Sub-roles and the Knowledge attribute. For a composite role, its Sub-roles attribute lists its sub-roles, representing the local organization structure. Its Knowledge attribute arises from the interaction of sub-role knowledge, and represents local social knowledge. Its protocol and activities are social actions or tasks emergent from the interaction of sub-role protocols and activities. Its liveness responsibilities, still defined as **w**-regular expressions over the sets of activities and protocols the role processes, represent local social goals that the sub-role responsibilities interact to achieve. Its safety responsibilities, still defined as predicates, represent local social laws emergent from sub-role safety responsibilities, hence respected by sub-roles.

Social Participation and Information Hiding

We use the “involved” keyword to denote social participation. Assuming the protocol Protocol_B1 of composite role B is the emergent action from the interaction of protocol Protocol_D1 of role D and activity Activity_E1 of E. We can then depict this participation in the super-role’s (B) role schema, under the protocol attribute, by the statement: Protocol_B1 **involves** D.Protocol_D1 and E.Activity_E1. Let’s look at a similar statement on B’s liveness responsibility resp: resp = (work | wait) involves D.resp1 and E.resp2. The statement defines resp as a **w**-regular expression in terms of B’s activities work and wait. The statement also points out that resp is a result of interaction between D’s liveness responsibility resp1, and E’s liveness responsibility resp2. Any attribute of the super-role can only involve attributes of the same type from the direct sub-roles, with the exception that protocols and activities can both

involve any mixture of sub-role protocols and activities. The participation relationship is intrinsic to the role hierarchy and cannot be removed. Modeling this relationship explicitly highlights the dependencies and simplifies changes frequent to open systems. To a composite role, its internal social participation of sub-roles is essentially its implementation detail. Hence its sub-role participation and internal structure is hidden from and invisible to its super-role, sibling roles under the same super-role and interacting peer roles. This is to prevent these roles to depend on the implementation details. To these roles, the composite role considered appears identical to an atomic role.

Only its sub-roles are allowed to see their participation in achieving mutual goals. In the example of Figure 4, the internal participation in B by D and E is only visible to D and E, not to A, C, F or G. With the above mechanism, a role B, whether atomic or composite, can be easily extended to a multi-role system (composite role) of any complexity. By preserving B's original interface, none of the super-role, sibling roles and interacting peer roles will be affected. This mechanism allows seamless extension to the system.

Modeling Runtime Reflection

Many applications in our scenario are expected to be highly available. To extend and maintain these applications, we need the ability to change the application architecture as well as the ability of individual agents at runtime. A rich body of work exists for reflection and adaptive software (Wooldridge, 2012). At the methodology level we do not subscribe to any particular implementation. We only model it abstractly. To do so, we extend the permissions attribute of roles. We allow a role to have read, write or create permissions on definitions of other roles. For example, a role A might include in its permissions attributes a permission to modify all protocols belonging to another role B. There are three logical levels of reflection, on the entire role, on an attribute, such as protocols, or on a particular member in an attribute, such as the Auction protocol. A star notation signals recursive permission on all sub-roles, direct or indirect. For example, B.protocols* represents reflection on protocols of B and all its sub-roles down the hierarchy.

The Protocol Model and the Interaction Model

Apart from the new name, the protocol model is the same as the original GAIA interaction model. The ROADMAP interaction model is based on the AUML interaction diagrams, with roles and zones represented graphically.

Design Models.

All analysis models are carried into the design phase. During design they are updated to reflect the design decisions. Three design models: the agent model, the service model and the acquaintance model are created from the updated analysis models. These models are refined iteratively until sufficient design information is captured.

Roles as Agreements of Behavior

As an agent takes a role in the system, we can see it as entering a contract with the rest of system to perform certain duties and functions. Hence roles can be seen as contracts of behavior similar to the interfaces in OO programming. The scope of roles covers the social aspects, the knowledge aspects of the organization, as well as the individual agent responsibilities and abilities. It is much richer and more expressive in comparison to OO interfaces, to better constraint richer agent behaviors. The agent classes are similar to the OO classes and agents are similar to the objects. One obvious implication of this analogy is accessing multiple agent classes through their common role to model polymorphism. Unlike interfaces, roles can be changed at runtime, and should be considered as variable-term agreements on agent behaviors, rather than immutable contracts. This difference makes the architecture of role-based systems more flexible at runtime than traditional systems. In general, if an agent behaves according to the appropriate role in the appropriate zone, other agents in the organization can then trust this agent to act to achieve the overall goal of the organization.

The Role Model and the Agent Model

The role model in the specification and analysis phase only aims to provide a conceptual view of the system. Its organization structure has not been optimized as the architecture of the system towards any quality goals. In the design phase the role model is refactored, for the chosen quality goals. The agent model is created in parallel by assigning roles to agent classes as in the original GAIA. At runtime we expect an agent to have a pointer to each of its roles.

This pointer allows us to access agents via their roles at runtime. The role assignments are also subject to change at runtime. For every agent class, a number of associated services are named. For each service, we can specify whether it implements a protocol or activity from the assigned roles, using the “implements” keyword. For each protocol or activities, there must be at least one implementing service. This allows the correct service to be invoked at runtime. Such dependency is intrinsic to the system. By providing formal descriptions of the dependency, applying future changes to the system will require less effort.

Other Design Models

All analysis models are optimized for quality goals in the design stage, following the same model syntax. The service model and the acquaintance model are the same as in GAIA.

The only exception is that roles can now be represented as nodes in the acquaintance diagram like agents.

Critique of ROADMAP Methodology:

1. Goals implicitly coincide with subdivisions of the system, which potentially increase the modeling complexity. There is also no clear guideline on how to derive roles from the organizational model.
2. It is difficult to model Agents entering and exiting sub-organizations; or, adapting to the evolution of organizational structure. There is a lack of dynamic reasoning (Juan, Pearce, & Sterling, 2002)
3. Organizational metaphor is a strongly embedded abstraction coded in the GAIA methodology.

2.1.4.1.3 Multiagent Systems Engineering (MaSE)

MaSE, proposed by DeLoach et al. (2001), stands for Multi-agent System Engineering. MaSE methodology aims to provide developers guidance from requirements to implementation.

Multiagent Systems Engineering (MaSE) (DeLoach, 2012) is an agent-oriented software engineering methodology which is an extension of the object-oriented approach. MaSE does not view agents as being necessarily autonomous, proactive, etc.; rather agents are simple software processes that interact with each other to meet an overall system goal." (DeLoach, 2012). MaSE's authors argue that, regarding agents in this way, one may avoid having to

define what an agent is, which was an arguable topic at the time the methodology was being developed.

In addition, all the components in the system are equally treated regardless of whether they possess intelligence or not. Because of this inherent perspective, MaSE is constructed according to the application of existing object-oriented techniques to the analysis and design of multiagent systems.

As a software engineering methodology, the main goal of MaSE is to provide a complete-lifecycle methodology to assist system developers to design and develop a multi-agent system. The MaSE methodology is a specialization of more traditional software engineering methodologies. The general operation of MaSE follows the phases and steps shown below table 2.2 and figure 2.5.

Table 2.2: MaSE Development Processes and Models Source from Deloach

1. Analysis Phase

	Phases	Model
a	Capturing Goals	Goal Hierarchy
b	Applying use cases	Use case, Sequence Diagrams
c	Refining Roles	Concurrent tasks, Role Model

2. Design Phase

	Phases	Model
a	Creating Agent Classes	Agent Class Diagrams
b	Constructing Conversations	Conversation Diagrams
c	Assembling Agent Classes	Agent Architecture Diagrams
d	System Design	Deployment Diagrams

Similar to GAIA, it also assumes the availability of an initial requirements prior specification to the start of software development under the methodology process. The process consists of seven steps, divided into two phases. The Analysis phase consists of three steps: Capturing Goals, Applying Use Cases, and Refining Roles. The remaining four process steps, Creating Agent Classes, Constructing Conversations, Assembling Agent Classes, and System Design, form the Design phase.

Analysis Phase

Once the concurrent tasks of each role are defined, the Analysis phase is complete. The MaSE Analysis phase is summarized as follows:

1. Identify goals and structure them into a Goal Hierarchy Diagram.
2. Identify Use Cases and create Sequence Diagrams to help identify roles and communications paths.
3. Transform goals into a set of roles.
 - (a) Create a Role Model to capture roles and their tasks.
 - (b) Define role behavior using Concurrent Task Models for each task.

Design Phase

There are four steps to the designing a system with MaSE. The first step is Creating Agent Classes, in which the designer assigns roles to specific agent types. In the second step, Constructing Conversations, the conversations between agent classes are defined while in the third step, Assembling Agents Classes, the internal architecture and reasoning processes of the agent classes are designed.

Finally, in the last step, System Design, the designer defines the number and location of agents in the deployed system.

Once the Deployment Diagrams are finished, the Design phase is complete.

The MaSE Design Phase can be summarized as follows:

1. Assign roles to agent classes and identify conversations.
2. Construct conversations, adding messages/states for robustness.
3. Define internal agent architectures.
4. Define the final system structure using Deployment Diagrams.

Agent Tool

The agent Tool system (DeLoach & Wood, 2001) has been developed to support and enforce MaSE. Currently agent Tool implements all seven steps of MaSE as well as automated design support.

Applications

MaSE has been successfully applied in many graduate-level projects as well as several research projects. The Multiagent Distributed Goal Satisfaction project used MaSE to design the collaborative agent framework to integrate different constraint satisfaction and planning systems. The Agent-Based Mixed- Initiative Collaboration project also used MaSE to design MAS focused on distributed human and machine planning. MaSE has been used successfully to design an agent-based heterogeneous database system as well as a multiagent approach to a biologically based computer virus immune system. More recently, we applied MaSE to a team of autonomous, heterogeneous search and rescue robots (DeLoach et al., 2003). The MaSE approach and models worked very well. The concurrent tasks mapped nicely to the typical behaviors in robot architectures. MaSE also provided the high-level, top-down approach missing in many cooperative robot applications.

Critique of MaSE Methodology:

1. Goal analysis, conducted at the beginning of a MaSE process, reinforces goal preservation through analysis and design phases.
2. It facilitates role and Agent class modeling to focus on clear goal delegation, where every role is responsible for a particular goal to be accomplished.
3. There are tasks that belong to the dedicated goals of roles. In a role refinement step, it is crucial to match goals with roles. Every goal has to be associated with a role. With these roles defined, the design of communication between roles and their corresponding tasks become fixed, lacking dynamic adaptability of goals (and hence roles).

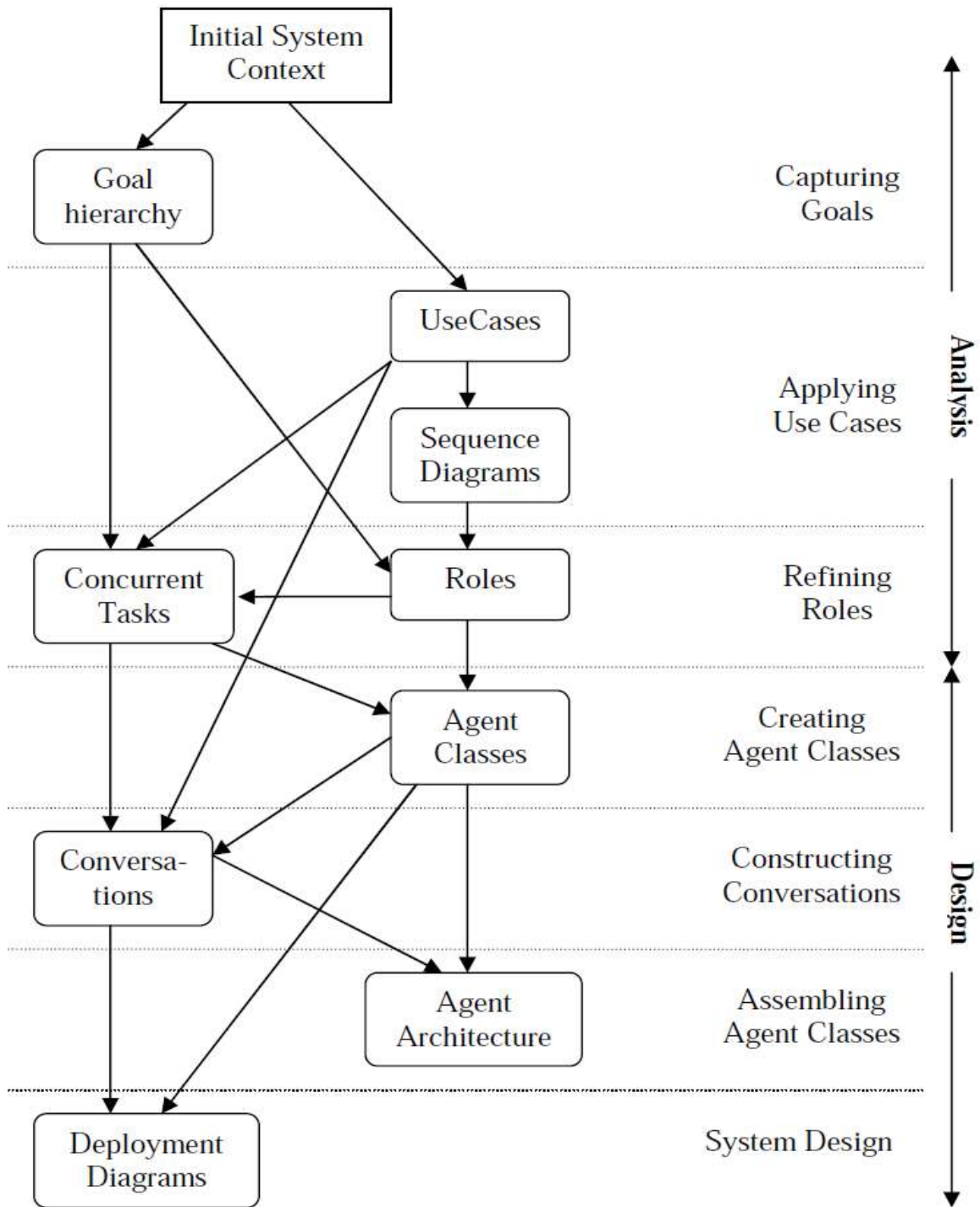


Figure 2.5. MaSE's process steps and artifacts (Source from DeLoach, 2012)

2.1.4.1.4Prometheus Methodology

The Prometheus methodology is a detailed and complete (“start-to-end” process for specifying, designing, and implementing intelligent agent systems, which has been developed over the last few years in collaboration with Agent Oriented Software (A company which markets the agent development software platform JACK (Padgham & Michael,2012), as well as agent solutions). The goal in developing Prometheus is to have a process with defined deliverables, which can be taught to industry practitioners and undergraduate students who do not have a background in agents, and which they can use to develop intelligent agent systems. The Prometheus methodology is a detailed AOSE methodology, which aims to cover all of the major activities required in the developing agent systems (Padgham & Michael, 2012). The aim of Prometheus is to be usable by expert and non-expert users. The methodology uses an iterative process which consists of three phases: system specification, architectural design and detailed design, see figure 2.6. Each of them is elaborated in detail below.

System specification

The system specification is the first phase of Prometheus. Its main purpose is building the system's environment model, identifying the goals and functionalities of the system, and describing key use case scenarios.

Firstly, one of the main characteristics of agents is situatedness. It means that agent systems are situated in an environment that is changing and dynamic. To some extent, situated agents need to interact with the environment. As a result, building the environment model is an important step in this system specification stage. Modelling an environment involves two activities: identifying percepts which are incoming information from the environment and determining actions which are the means by which an agent affects its environment. Percepts and actions are defined using descriptors. Additionally, external resources such as data, information, etc. need to be identified.

Secondly, goals and functionalities of the system need to be captured at this stage. At the first step, system goals are identified mainly based upon the requirements specification. Goals are decomposed into subgoals if necessary. After that, system functionalities that achieve these

goals are defined. Another step which helps the analysts identify the system functionalities is defining use case scenarios.

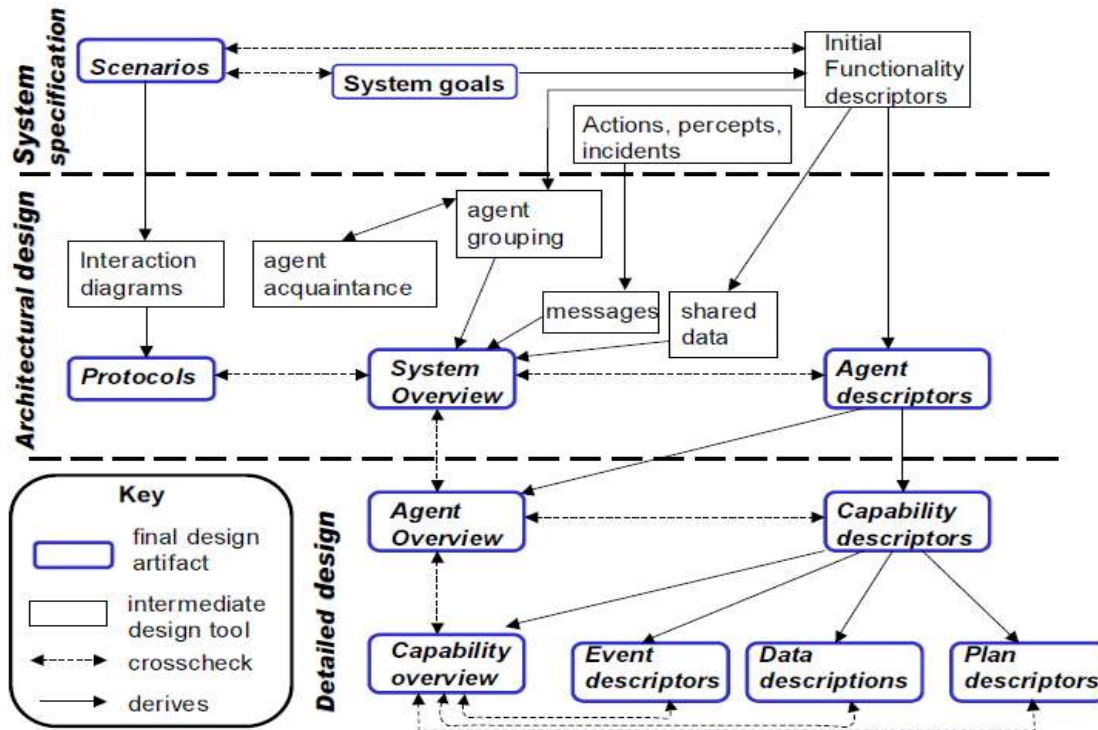


Figure 2.6: Prometheus Overview (extracted from Padgham & Michael, 2012)

Architectural Design

Between the requirements capturing phase and the low level design phase where the system is modeled as computational entities which suit a particular agent platform, Prometheus has an intermediate phase called architectural design. The three main activities involved in this stage are: defining agent types, designing the overall system structure, and defining the interaction between agents. Objects are regarded as the basic entity in object-oriented design and agents are their counterpart in agent-oriented design. Therefore, determining which agents should exist in the target system is an important step. The designers are able to make that decision by grouping the system functionalities which were previously defined in the system specification phase.

Functionalities are grouped based upon two criteria. Functionalities that are related to each other (e.g. using the same data) are likely to be in the same group (cohesive criterion). On the other hand, if there are significant interactions between two functionalities, then there is a high chance that they should be grouped (coupling criterion). Prometheus also provides the data coupling diagram and agent acquaintance diagram as aids to the functionalities grouping process. After agent types are defined, the system's structure needs to be captured in a system overview diagram, which is arguably the single most important design artifact in Prometheus" (Padgham & Michael,2011). The system overview diagram (Figure 2.5) is constructed based on the designers' understanding of the system up to this stage of the development process. It depicts the agent types and the communication links between them and the data used, which was defined in the previous step. Furthermore, it shows the system's boundary and its environment in terms of actions, percepts and external data. In short, the system overview diagram provides the designers and implementers with a general picture of how the system as a whole will function.

The system overview diagram, however, only provides the static structure of the system. At this stage, the designers are also required to capture the dynamic behavior of the system. There are two types of diagrams which Prometheus uses to represent the system dynamics. Interaction diagrams are borrowed from object-oriented design to show interaction between agents. They are developed based upon use cases scenarios that are defined in the system specification stage. At a lower level of detail, interaction protocols define the intended valid sequence of messages between agents

Detailed Design

The final stage of the current Prometheus methodology is the detailed design. This is where the internal structure and behavior of each agent are addressed. This stage emphasizes on defining capabilities, internal events, plans and detailed data structure for each agent type defined in the previous step.

Firstly, an agent's capabilities are depicted via a capability descriptor which contains information such as which events are generated and which events are received.

The capability descriptor also includes a description of the capability, details involving interactions with other capabilities and references to data read and written by the capability. Secondly, at a lower level of detail, there are other types of descriptors: individual plan descriptors, event descriptors, and data descriptors. These descriptors provide the details so that they can be used in the implementation phase.

The detailed design phase also involves constructing agent overview diagrams. These are very similar to the system overview diagram in terms of style but give the top level view of each agent's internals rather than the system as a whole. An agent overview diagram, together with the capability descriptors, provides a high level view of the components within the agent internal architecture as well as their connectors (interactions). They show the top level capabilities of the agent, the flow of tasks between these capabilities and data internal to the agent.

Prometheus is supported by two tools (Padgham & Michael, 2012). The JACK Development Environment (JDE), developed by Agent Oriented Software (www.agent-software.com) includes a design tool that allows overview diagrams to be drawn. These are linked with the underlying model so that changes made to diagrams, for example adding a link from a plan to an event, are reflected in the model and in the corresponding JACK code. The Prometheus Design Tool (PDT) provides forms to enter design entities. It performs cross checking to help ensure consistency and generates a design document along with overview diagrams. Neither PDT nor the JDE currently support the system specification phase.

Applicability: Prometheus supports the engineering of conventional closed systems with controlled and trusted agents. It specifically supports the BDI framework, and focuses on functionalities. Its concrete nature and detailed models and processes allow easy transition from the conventional OOSE approaches and make it very suitable for conventional applications such as an intelligent web server.

However, it lacks support for advanced properties such as openness and is not suitable for systems requiring these properties.

2.1.5 Software Engineering Methodology Evaluation

In the previous section, we have performed a brief literature review on agents and agent-oriented methodologies. In this section, we review the literature involving the evaluation of software engineering methodologies. As we discussed in the previous section, the evolution of software engineering has been taking place from the early days when ad-hoc programming was the dominant method of producing software. During this period, a large number of software engineering methodologies have been offered to the computer science and software engineering communities. On the one hand, they have provided a rich resource but on the other hand the decision of choosing which methodology to use to design and implement a particular system is more difficult and critical (Mike,2013). The decision of adopting a new methodology may affect the success of a software product, the current organization practice as well as cost, training and other issues Having recognized those difficulties involved in the selection of an appropriate methodology, there has been a large amount of effort spent on evaluating and comparing software engineering methodologies. This section briefly discusses various key methods, techniques and frameworks which have been proposed in this research area. Since object-oriented methodologies are considered as the predecessor" of agent-oriented methodologies, we also look back at work on evaluating and comparing a number of object-oriented methodologies.

2.1.6 Methods for Evaluating Methodologies

Making a choice from the apparently very wide range of methods and tools available can in itself be a complex and costly process" (Law & Naem, 2013).

Therefore, it is necessary to carry out systematic evaluations. In answering a pressing need for methods for comparing or evaluating methodologies, there have been a number of major initiatives in this area over the past decades.

Amongst these are:

The book **Methods for Comparing methods"**(Law & Naem, 2013) written by David Law and published by the UK National Computing Centre (NCC) in 1988.

Its main purpose is to provide a scientific approach to the comparison of methodologies, especially regarding how design methodologies and requirements specification methodologies can be usefully evaluated.

The **DESMET (Determining an Evaluation methodology for Software Methods and Tools)** project which started in 1990 and began to publish details in 1994 (Law & Naem, 2013). Its participants included the UK National Computing Centre, The University of North London and several European software consultants. The main contribution of DESMET resides in the effort to develop a common framework for evaluation methods, tools and techniques in the software engineering domain. DESMET has been commonly regarded as a source of inspiration for all aspects concerning both qualitative and quantitative evaluation.

The **NIMSAD (Normative Information Model-based System Analysis and Design)** framework initiated by Jayaratna in his book *Understanding and Evaluating Methodologies: NIMSAD a Systematic Framework* published in 1994. Its significance resides in its difference to other approaches at that time by dealing with the methodology evaluation area from a more general, philosophical or theoretical perspective.

The book **Information Systems Development: Methodologies, Techniques and Tools**" (Avison & Fitzgerald, 2011) written by Avison and Fitzgerald. Its first edition was published in 1988, the second in 1995 and the latest was recently published in 2002. Apart from detailed and well-described concepts relating information system development methodologies, its main contribution to the area of methodology comparison is the review of existing evaluation approaches and the proposal of a generic framework for methodology classification. Besides the above significant works, there have been various approaches (Allan & Wallnau, 2012) which generally either **adapt** them or **extend** and **tailor** them to suit a particular evaluation purpose. In the remainder of this section, we classify the major approaches to methodology evaluation into four main groups:

Feature-based evaluation, Quantitative evaluation, NIMSAD framework and other evaluation approaches.

2.1.6.1 Feature-based evaluation

Feature-based evaluation (also often called **Feature Analysis**) is the most prominent and popular comparison approach which has been used. It is regarded as a qualitative method (Avison & Fitzgerald, 2011). It involves building an evaluation framework that can be represented in terms of a set of properties, qualities, attributes or characteristics (Avison & Fitzgerald, 2011). These features are able to describe the evaluated methodology sufficiently well so that it can be assessed and compared for a particular purpose. They often reflect user requirements for specific tasks or activities performed on a particular domain. Assessing a methodology against a framework of attributes and features involves some judgment of how well it supports or to what extent it has a specific attribute or feature. In other words, the methodology assessment is determined on the basis of its ratings on the different attributes and features.

Furthermore, deriving a set of attributes or features is a difficult task since there is no universal agreement on the standard set of features (Law, 2012). The choice of features may essentially reflect the subjective opinion of a particular group of assessors, which in turn depends on their background, interests and knowledge. There have been several approaches to devise evaluation features (Law & Naem, 2013). For instance, one can consider a software development methodology as a set of the models, processes, techniques and interactions. The task of evaluating a methodology then becomes assessing its support for features of each of these components.

The second approach involves using an expert perspective on what the key features of such a methodology should be. They may be derived either from experience or from the theories and principles of software engineering.

For instance, one can follow the guidelines in (Wood et al, 2012) to derive a number of features that the evaluation purpose. In practice, both techniques are often used together to generate a list of evaluation criteria. The framework of features generally needs to consider not only the technical aspects but also economic, cultural and quality issues.

Devising a set of features and criteria for the evaluation framework is only one of the major concerns in performing a Feature Analysis.

Another concern involves the selection of the evaluation **procedure**, i.e. the way in which the evaluation is organized. Similar to features generation, there are also different ways of organizing a Feature Analysis. They range from a simple comparison performed by a single assessor to a formal process conducted in an organization to select a methodology for its development process. Kitchenham (Wood et al, 2012) has categorized them in four major groups:

1. **Screening Mode approach:** This approach can be performed by a single person who is responsible for both generating the evaluation criteria and assessing the methodologies. The evaluation is solely based on his/her understanding of the methodologies according to their documentation. This technique does not require a large amount of time or effort/cost but it is not reliable. This is due to the fact that the entire evaluation is based on the assessors' subjective opinion which may not be representative of the users of the methodology. Also, the results of the evaluation may not be correct since the assessors may make a wrong assumption on a specific aspect of the methodology.
2. **Case Study approach:** This approach is proposed in (Wood et al, 2012). The evaluated methodologies are used to develop a real project. In contrast to the screening mode, there now are two distinct roles in the evaluation process.

The first role is the evaluator who is responsible for selecting the methodologies, generating the evaluation criteria and selecting the testing project.

The second role is played by the software developers who assess each feature of the methodology based on their experience of using it to develop the trial project. This approach has several advantages such as providing a practical evaluation and the evaluation is performed by actual users of the methodologies.

Nevertheless, its limitations include that the result collected from a doing a project is probably not representative of some specific features that the methodology addresses.

In addition, the assessment is also affected by the background, the ability and learning curves of the software developers in using and understanding the methodologies. Finally, it is also relatively expensive since a certain number of people need to be involved in the evaluation.

3. **Formal Experiment approach:** Similar to the case study approach, there are also at least two different roles in this approach. The evaluators need to select the methodologies, build the set of features, plan and run the experiments, and analyze the results. They also need to choose an appropriate experimental design, select (randomly or deliberately) the experimental subjects (i.e. users) and probably classify them into different classes of users. In addition, the experimental subjects are trained in the use of the methodology if necessary. This approach is likely to produce the most reliable results since it seems to reduce the influence of single assessor differences. It is, however, the most costly and lengthy approach. It may require a large number of participants in the evaluation process.

Survey approach: This approach does not involve the practical use of the evaluated methodologies. Rather, it relies on the assessment of the experts and users who have been using some of the evaluated methodologies. Similar to the above three approaches; the evaluators also need to derive a set of evaluation criteria together with the judgment scale. After that, they design and run the survey. Several tasks are involved with this process, including choosing the type of survey (e.g. web-based survey or personal interview), building the survey documentation (e.g. questionnaire), and identifying people who will be asked to join in the survey.

Finally, the evaluators run the survey and collect and analyzed the responses according to the survey design. The advantages of this approach are its tendency to take less time and effort than the formal experiment approach. In addition, it may reflect the opinions of a wide range of methodologies' users.

Its limitations include the difficulty in finding the right people to ask to participate in the survey, especially, if the evaluated methodologies are still not popular and mature enough to have a large number of users.

In addition, unlike the formal experiment approach the evaluators are not able to control the experience, background and capabilities of the participants.

The feature-based evaluation approach gains its popularity from its high **flexibility**.

Firstly, it can be performed by anyone or any organization without the need to have several evaluation facilities such as a measurement programmed in place.

Secondly, a feature analysis can be conducted to any required level of detail, ranging from very simple evaluations such as screening mode to sophisticated procedures like formal experiments.

Thirdly, this approach can be used not only to assess methodologies but also any type of tools and development processes.

Nonetheless, there are several outstanding limitations relating to feature-based evaluation.

First of all, the subjectivity involved in using the feature-based approach is of concern. As mentioned earlier, subjectivity can come from two sources: the set of evaluation criteria and the judgment of a particular methodology against them.

Secondly, the inconsistency in the judgment of different assessors may pose some issues. It results from the fact that different people have different backgrounds, experiences and ability to understand and use a methodology. For instance, some features have a higher score than others just simply due to the fact that the assessors tend to be more familiar with them.

Thirdly, aggregating the scores of all the evaluations criteria to a single number that represents the quality of a methodology is a difficult task. Besides, the evaluators need to deal with the relative importance among features - some of them significantly represent the quality of a methodology whilst others do not.

Finally, one may face the redundancy crisis where too many features (e.g. more than 100) are produced. Managing the judgment scores of all the features becomes a more difficult task. In addition, the evaluators need to collate and analyze the scores.

Quantitative evaluation approaches

Feature-based evaluation or Feature Analysis is usually a qualitative technique. Even though it can be quantified in the sense of judging scores, assessing scales, weights and aggregating them, it still deals with quality aspects of the methodology. On the other hand, quantitative

evaluations assess a methodology according to some measurable results produced by its use. These can be the software applications produced or the changes in the development process.

Similar to the Feature Analysis, the main procedures to perform a quantitative evaluation are case studies, formal experiments, and surveys. These also proceed in an analogous fashion.

The main difference as proposed in DESMET (Wood et al, 2012) is that before running a case study, an experiment or a survey, the evaluators must **formulate** and **validate** the hypotheses.

Formulating a hypothesis involves defining the consequences the assessors expect the evaluated methodologies to bring when applied to a project or a development process. These effects need to be defined in such a way that they are measurable and detailed. Validating a hypothesis is to ensure that the evaluators are able to correctly interpret the results based on it.

Overall, quantitative evaluation methods tend to produce more reliable results than qualitative or feature-based approaches do. In fact, DESMET classifies them in terms of the extent of risk that an evaluation draws an incorrect conclusion about the methodologies being evaluated. The relative risk related with each quantitative technique ranges from low (Quantitative Case Study method) to very low (Quantitative Formal Experiment).

In contrast, qualitative or feature analysis approaches have from a very high risk (Feature-based screening mode) to low risk (Feature-based experiment). Despite producing higher confidence in evaluation results, quantitative methods are not as exible as feature-based approach. To conduct a quantitative evaluation, an organization needs to have some prior infrastructure such as a measurement program, a set of standards for metrics, etc. Furthermore, there are some aspects of the methodology such as process visibility or controllability that is best evaluated using qualitative methods.

2.1.6.2The NIMSAD framework

The above two approaches are the most practical ways of performing a comparative analysis on system development methodologies. There are also alternative approaches for understanding and evaluating methodologies in a more theoretical way. One of them is the

NIMSAD (Normative Information Model-based System Analysis and Design) framework. It was proposed by Jayaratna in his book *Understanding and Evaluating Methodologies: NIMSAD a Systematic Framework*" (Jayaratna, 2012). Being an evaluation framework, NIMSAD is not in fact a method for practically and efficiently comparing methodologies as such. Rather, it provides an alternative way of understanding and evaluating methodologies on the basis of the models and epistemology of systems thinking perspective" (Jayaratna, 2012). The basic scolding for the framework is constructed based on a fairly wide and encompassing view of an approach to problem solving.

The framework contains four major elements: the methodology context, the methodology user, the methodology itself, and the evaluation of the above three.

The fourth component, the evaluation, is missing from many other frameworks according to Jayaratna. The process of evaluating a methodology involves answering various questions that address the first three components. For instance, the questions relating to the methodology context element deal with the mechanism employed by the methodology in assisting the understanding and identification of the clients, their experiences and commitments, the culture and context of the target system, etc.

The questions concerning the second element, the methodology users, involve their belief, values, and ethical positions, their experiences, skills, and motives, etc.

The questions addressing the third element (the methodology itself) are similar to the evaluation criteria that found in other frameworks. These questions and the way the methodology provides specific assistance for understanding, defining and modelling the problem, implementing the design, etc.

The fourth element, evaluation, is performed at three stages: prior to intervention (i.e. before a methodology is adopted), during intervention (i.e. during its use) and after intervention is complete (i.e. assessment of the success or failure of the methodology).

The NIMSAD framework has been applied to evaluate three well-known methodologies which have different approaches to systems development. These are Structured Systems Analysis and Systems Specification, ETHICS, and SUM (Jayaratna, 2012).

2.1.6.3 Other Approaches

Apart from the above three major evaluation approaches, there are various methods and frameworks that have been proposed in the literature. For instance, David Law suggested that the evaluators are able to employ the set of evaluation criteria and make a direct comparison between methodologies rather than examine all the compared methodologies together. He argued that the latter who involves judging on the basis of a rating scale may be less discerning and sensitive. He indicated that the direct comparison is suitable for quick, subjective expert comparison at a relatively high level of detail. Also mentioned in (David-Law, 2012), the easiest way for an organization to choose a methodology, method or tool is following the recommendations of some institution. This approach, however, is not a safe option since the organization has to be careful in taking account of its own needs, requirements, etc. In a slightly different direction, DESMET (Barbara, 2012) refers to this approach as a Qualitative Effects Analysis where the selection of a method or tool is done on the basis of expert opinion. This, however, assumes the existence of a knowledge base of expert opinion regarding generic methodologies and techniques.

2.1.7 Comparisons of Object-Oriented Methodologies

In the early 90's, Object-Oriented (OO) software engineering experienced a similar period to that which Agent-Oriented (AO) approaches are going through now. Driven by the attractiveness of the OO paradigm, software engineering researchers proposed a large number of Object Oriented methodologies supporting the development of OO systems. As cited in (Frank, 2012), a famous quote of Edward Berard describes this situation as: I have good news and bad news. The good news is that there has been a great deal of work in the area of 'object-oriented software engineering'. The bad news is that there has been a great deal of work in the

area of 'object-oriented software engineering. This major issue led to difficulties in deciding between different methodologies for object-oriented design.

Consequently, a large amount of effort was spent on evaluating, comparing and understanding object-oriented methodologies. Since AO and OO methodologies share the same foundation of software engineering goals and principles, it seems important and useful for us to review the main evaluation approaches applied to OO methodologies. These may provide inspirations and valuable resources for our research relating to AOSE methodologies evaluation.

Comparative studies of OOSE methodologies generally fall into the four major methods of evaluation which we discussed in the previous section. Some of them target a full OO methodology, whereas others target at several aspects or process stages. For instance, in (Frank, 2012), the complete software development process, the modelling language as well as tool support of a methodology are examined. On the other hand, (Frank, 2012) focus on the comparison of requirements specification techniques; whereas object-oriented design and analysis are addressed in (Frank, 2012). In Frank, (2012), only the modeling language (i.e. models and notations) of different object-oriented methodologies are evaluated. Overall, significant work in the area of Object Oriented methodology comparisons can be classified into three styles: comparison against a frame-work (feature analysis), comparison by meta-modelling and comparison by outcome (quantitative evaluation).

2.1.8 Comparison against a framework

Because of the popularity and exhibility of feature analysis as a powerful evaluation approach, this type of comparative study on object-oriented methodologies has attracted more significant work than the other comparison styles. Comparison against a framework basically involves building an evaluation framework containing a hierarchy of attributes or features. These form the evaluation criteria on which the assessment is based. The evaluation framework also varies - some (Frank, 2012) try to construct an ideal OO development methodology against which others can be compared, whereas most of the work constructs their own framework. Overall, an emerging agreement of those comparison frameworks is that they address major

components of an OO methodology such as concepts, models, process, tools, and pragmatics. It is also noted that we found many evaluation criteria in this area that are potentially useful to our purpose of comparing agent-oriented methodologies.

2.1.9 Comparison by meta-modeling

This type of comparative study involves developing a common frame of reference (i.e. meta-modeling) for viewing different participating methodologies. It generally requires the construction of a meta-model based on a combination of different features provided by the compared methodologies. Among the significant works, there is a formal approach to evaluate six different Object Oriented methodologies (Frank, 2012) using the meta-modeling comparison technique, which was initiated by Hong, Goor and Brinkkemper in 1993. They argue that in order to make the comparison accurate and objective, those methodologies should be compared based on a **uniform, formal** and **unbiased** basis. As a result, the meta-modeling evaluation technique is chosen. At the first step, the goals, concepts, techniques, processes and graphical notations of each methodology are identified. The collected information is then used to build a meta-model of each methodology which consists of two sub-models: a meta-process model and a meta-data model.

The meta-process model captures the process of the methodology whereas the meta-data model describes the concepts and techniques relating to it. Based on the meta-models of the selected methodologies, the research describes the process of comparing them according to three major features: the analysis and design steps, the concepts, and the techniques provided. Meta-modelling techniques have their attraction in terms of being more objective and accurate compared with feature-based techniques. However, we found that the meta-models constructed do not capture other issues of software engineering principles such as reusability, maintainability and modifiability. Additionally, meta-modelling techniques tend to be useful only for cases when we want to perform a comparison on a certain number of methodologies. Another representative of this kind of comparative study can also be found in (Frank, 2012).

2.1.10 Comparison by outcome

Comparison of object-oriented methodologies by outcome is a form of quantitative evaluation. Object oriented methodologies are evaluated on the basis of assessing the outcome in terms of several measurable effects. For example, the quality of the product or the complexity of the process as a result of applying the methodologies can be examined. Representatives of this type of study in the object-oriented methodologies are the formal experiment conducted to assess analysis techniques for information requirements determination found in (Hans van Vliet, 2012), and the metrics-based evaluation of object-oriented software development methods in (Dumke & Foltin, 2012) and in (Hans van Vliet, 2012).

2.2 Summary of Literature Review.

As opposed to object-oriented methodologies, there has not been much work in comparing agent-oriented methodologies. Shehory and Sturm performed a feature-based evaluation of several Agent Oriented Software Engineering (AOSE) methodologies. Their criteria included software engineering related criteria and criteria relating to agent concepts. In another paper they used the same techniques in addition to a small experimental evaluation to perform an evaluation of their own Agent Oriented Modelling Techniques (AOMT). In the work by Sturm and Shehory (2003), four dimensions are proposed as four separate divisions of the issues being examined. They are (1) concepts and properties, (2) notations and modeling, (3) process, and (4) pragmatics. In addition to these, support for software engineering and marketability are added into the comparison framework by Dam and Winikoff (2003).

This work suffers from subjectivity in that the criteria they identified are those that they see as important and, naturally, AOMT focuses on addressing these criteria. A framework to carry out an evaluation of agent-oriented analysis and design modelling methods has been proposed by Cernuzzi and Rossi. The proposal makes use of feature-based evaluation techniques but metrics and quantitative evaluations are also introduced. The significance of the framework is the construction of an attribute tree, where each node of the tree represents a software engineering criterion or a characteristic of agent-based system. Each attribute is assigned a

score and the score of attributes on the node is calculated based on those of its children. They have applied that framework to evaluate and compare two AOSE methodologies: the Agent Modelling Techniques for Systems of BDI (Belief, Desire and Intention) Agents and MAS-CommonKADS.

In O'Malley and DeLoach proposed a number of criteria for evaluating methodologies with a view to allowing organizations to decide whether to adopt AOSE methodologies or use existing Object Oriented methodologies. Although they performed a survey to validate their criteria, they do not provide detailed guidelines or a method for assessing methodologies against their criteria. Their example comparison (between MaSE and Booch) gives ratings against the criteria without justifying them. Their work is useful in that it provides a systematic method of taking a set of criteria, weightings for these criteria (determined on a case by case basis), and an assessment of a number of methodologies and determining an overall ranking and an identification of which criteria are critical to the result.

A few frameworks for comparing Agent-oriented methodologies have been suggested. Sabas, Badri, and Delisle (2002) suggest a multi-dimensional framework containing criteria within each of the following aspects: methodology, representation, organization, cooperation, and technology. These criteria are used as differentiators for comparison purposes. The results of comparisons are described by a two-dimensional array containing criteria (row wise) and methodological names (column wise). Each intersection is marked: "Y" for Yes, "N" for No, "P" for possible, or simply blank (' '). Agent-Oriented Software Engineering (AOSE) is a nascent but active field of research (Tveit, 2001). A comprehensive methodology that plays an essential role in software engineering must be robust but easy-to-use. More importantly, it should provide a roadmap to guide engineers in creating Agent-based system.

Recently several Agent-oriented methodologies have been proposed to address the Agent oriented software engineering process.

Thus far, however, software developers have not embraced any single methodology, primarily because AOSE lacks industrial strength tools and standards (Sturm & Shehory, 2003).

Therefore, understanding the limitation of existing AOSE methodologies can permit researchers to develop better solutions. Exploration into building blocks for Agents, fundamental theories and methods as well as available assistance from notations of analysis, architecture design and implementation toolkits should be needed. The ultimate hope is the development of practical AOSE methodologies for building robust, industrial-strength Multi-Agent Systems (MAS). However, given the divergent directions currently being pursued by researchers, the road towards mature Agent software development may be reached most effectively by first gaining a better understanding of the competing approaches.

Furthermore, this dissertation is aimed to develop a complete life-cycle methodology for designing and developing Intelligent Agent Systems. The objective is to evaluate and compare three selected agent oriented methodologies (ROADMAP, MaSE and Prometheus) by performing a feature analysis which is carried out by evaluating the strengths and weaknesses of each participating methodology using an attribute-based evaluation framework. This evaluation framework addresses four major areas of an agent-oriented methodology: concepts, modeling language, process and pragmatics. The comparative study also goes further with a structural analysis where the key commonalities and distinguishing differences of the three selected methodologies are identified in terms of models, techniques, and tools.

This dissertation developed a new agent oriented software engineering methodology called An Enhanced Multi-Agent System Development (EMASD) methodology for development of Intelligent Agent systems. It is based upon strong points of the previous existing methodologies. This new methodology is established based on three fundamental aspects: concepts, models, and process. These three aspects are considered as a foundation for building a solid methodology. This methodology is developed based on the essential software engineering issues such as preciseness, accessibility, expressiveness, domain applicability, modularity, refinement, model derivation, traceability, and clear definitions. System Development Life Cycle (SDLC) and Object Oriented methodology (OOM) were adopted

during the development of Enhanced Multi-Agent System Development (EMASD) methodology. The Enhanced Multi-Agent System Development (EMASD) methodology is illustrated by a case study on an agent-based system – Intelligent Traveller Agent System.

CHAPTER THREE

SYSTEM ANALYSIS

3.1 Materials Required

This chapter highlights the system analysis methods and research methodology used in the course of this study. It attempts to appraise the existing system and refine the abstract description of the project. Before any project is carried out, the system has to be analyzed to suit the description of how and what the system would be doing. The analysis can be on paper for clarity in a sequence and also explicit in a way a layman would easily understand. The system after analysis has to be designed following the series of steps from the formal description of that system. System Analysis is the procedures by which activities in an organization are studied with the aim of determining how to operate it most efficiently, including how to effectively design intelligent agent systems (Lewis, 2014).

The objective of analysis is a realistic and keen insight into a system and its problem areas, so that an improved system can be designed.

3.2 Methodology Adopted

According to Martin (2012), a research methodology is a systematic programming approach of a well-defined procedure that should be followed in carrying out a thorough research project. An adequately suitable methodology that would ensure a very detailed research work and ensured a high degree of accuracy and efficiency is adopted. The research methodology used helps to ensure that a thorough study of the present system is effectively carried out, thus helping the project research team to completely understand the modus operandi of the existing system so as to know how the proposed system should be structured and the functionalities needed in it to address the seemingly problem discovered. This help to know if there should be a total overhauling of the existing system or if only improvements should be made.

Hence, after duly consideration of the above reasons, the systems development life cycle (SDLC) and Object Oriented Methodology are adopted. This is because the Systems Development Life Cycle (SDLC) is a conceptual model used in project management that describes the stages involved in an information system development project, from an initial feasibility study through maintenance of the completed application.

In general, an SDLC methodology follows these steps:

1. If there is an existing system, its deficiencies are identified. The new system is build to handle the deficiencies in the existing system.
2. The new system requirements are defined including addressing any deficiencies in the existing system with specific proposals for improvement.
3. The proposed system is designed. Plans are created detailing the hardware, operating systems, programming, and security issues.
4. The new system is developed. The new components and programs must be obtained and installed. Users of the system must be trained in its use, and all aspects of performance must be tested. If necessary, adjustments must be made at this stage.

3.3 Analysis of the Existing System

Many Agent Oriented methodologies use the metaphor of the human organization in which agents play one or more roles and interact with each other. Human organization models and structures are consequently used to design Multi Agent Systems.

Concepts like role, social dependency, and organizational rules are used not just to model the environment in which the system will work, but the system itself. Given the organizational nature of a Multi Agent System, one of the most important activities in an Agent Oriented methodology results in the definition of the interaction and cooperation models that capture the social relationships and dependencies between agents and the roles they play within the system. Interaction and cooperation models are generally very abstract, and they are concretized implementing interaction protocols in later phases of the design.

Although the Agent-Oriented Programming (AOP) paradigm was introduced more than ten years ago by Yoav Shoam in his seminal work (Shoham, 2013), still there are no Agent Oriented languages used in practice for developing a Multi Agent System.

In analyzing the present system, it would be good to state the problems of all the existing agent oriented methodologies. While individual agent-oriented methodologies are useful for restricted situations, a more flexible approach can be found in the use of an enhanced model. Furthermore, repository of method fragments can be built up and, from this, a selected number of fragments can be abstracted to form an enhanced intelligent model.

The ROADMAP methodology aims to support the engineering of large-scale open systems. It extends the GAIA methodology by introducing use-cases for requirement gathering, explicit models of agent environment and knowledge, and an interaction model based on AUML interaction diagrams (Wooldridge Jennings, 2014). The original GAIA role model is also extended with a dynamic role hierarchy. This role hierarchy is carried into design and will have a run-time realization, allowing social aspects to be explicitly modeled, reasoned and modified at run-time.

ROADMAP promotes the view of software systems as computational organizations. Agents in a system are similar to individuals in a human organization, while the roles in ROADMAP encapsulate regulations, processes, responsibilities and team roles by which individuals function within a human organization. They specify, support and constraint an agent's behaviors in the organization. When the expected behaviors in the organization are explicitly represented at run-time, agents can verify each other's behavior, and misbehaving agents can be identified and removed or replaced. The ROADMAP methodology encourages an iterative approach and expects details of the models to be filled in when applicable. During the analysis phase, the six analysis models are created to allow conceptualization of the system as an organization. During the design phase, the initial conceptualization of the system is optimized for the chosen quality goals, such as performance. ROADMAP prescribes rich models to explicitly deal with roles of agents, the environment and knowledge in the system.

The methodology provides strong support for engineering complex open systems, but is less suitable for application not requiring these properties. The architecture of ROADMAP is shown in Figure 3.1

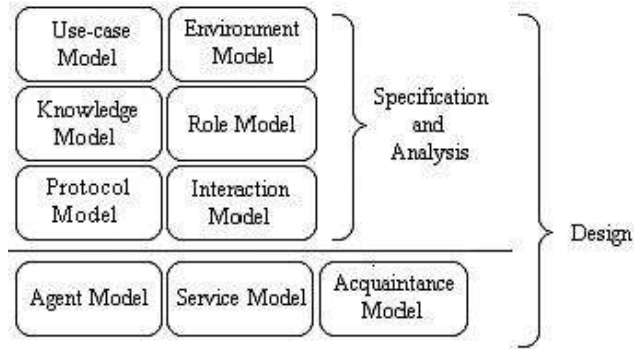


Figure 3.1 The ROADMAP Architecture: source from Wooldridge & Jennings, 2012

MaSE methodology aims to provide developers guidance from requirements to implementation. Multiagent Systems Engineering (MaSE) (DeLoach, 2012) is an agent-oriented software engineering methodology which is an extension of the object-oriented approach. MaSE does not view agents as being necessarily autonomous, proactive, etc.; rather agents are simple software processes that interact with each other to meet an overall system goal. (DeLoach, 2012). MaSE's authors argue that, regarding agents in this way.

In addition, all the components in the system are equally treated regardless of whether they possess intelligence or not. Because of this inherent perspective, MaSE is constructed according to the application of existing object-oriented techniques to the analysis and design of multi agent systems. As a software engineering methodology, the main goal of MaSE is to provide a complete-lifecycle methodology to assist system developers to design and develop a multi-agent system. The architecture of MaSE is shown in Figure 3.2

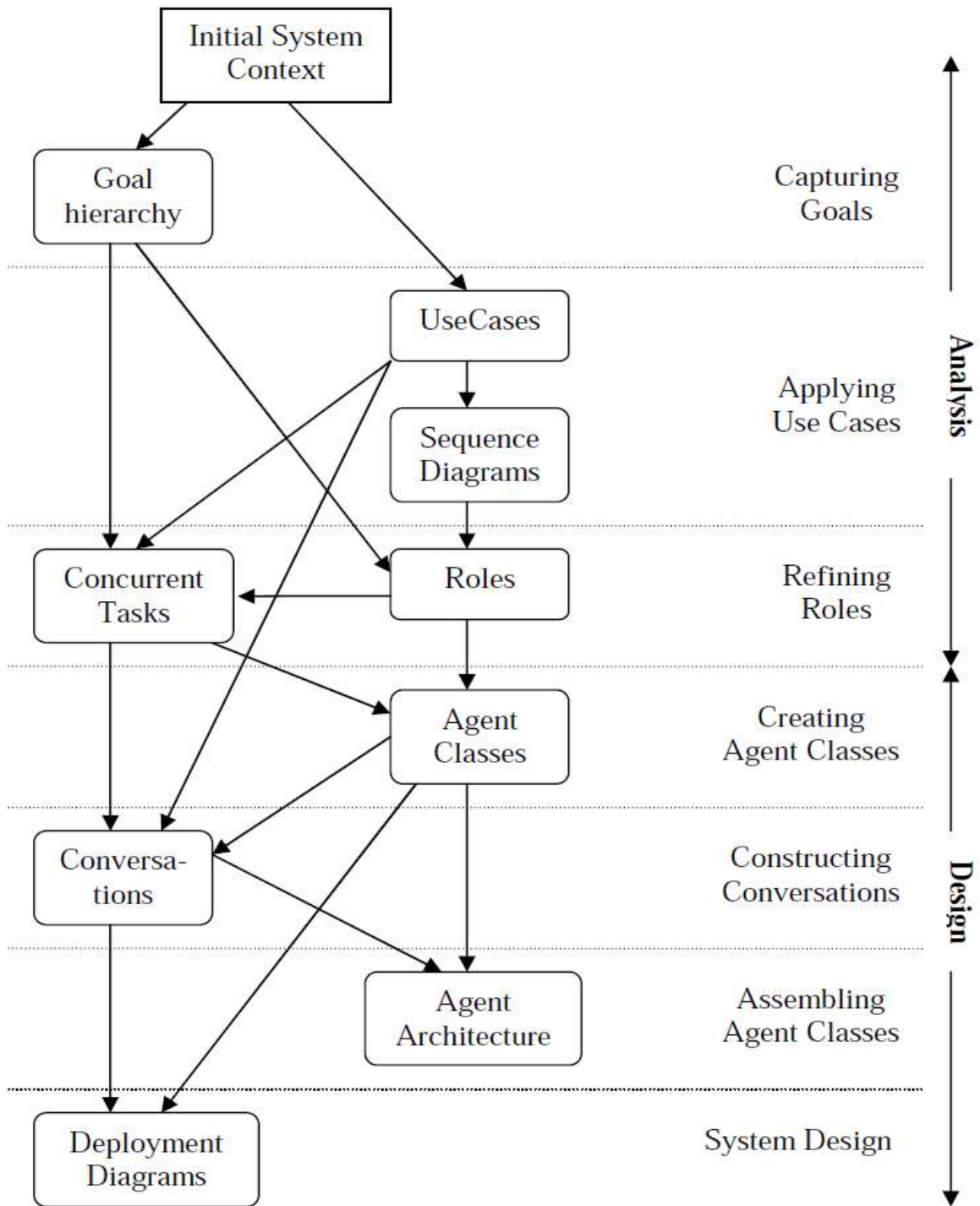


Figure 3.2. MaSE's Architecture (Source from DeLoach, 2012)

The Prometheus methodology is a process for specifying, designing, and implementing intelligent agent systems, which has been developed over the last few years in collaboration with Agent Oriented Software (A company which markets the agent development software platform JACK (Padgham & Michael,2012), as well as agent solutions. The goal in developing Prometheus is to have a process with defined deliverables, which can be taught to industry practitioners and undergraduate students who do not have a background in agents, and which they can use to develop intelligent agent systems.

The Prometheus methodology is a detailed AOSE methodology, which aims to cover all of the major activities required in the developing agent systems (Padgham & Michael, 2012). The aim of Prometheus is to be usable by expert and non-expert users. The methodology uses an iterative process which consists of three phases: system specification, architectural design and detailed design.

Prometheus supports the engineering of conventional closed systems with controlled and trusted agents. It specifically supports the BDI framework, and focuses on functionalities see figure 3.3. Its concrete nature and detailed models and processes allow easy transition from the conventional OOSE approaches and make it very suitable for conventional applications such as an intelligent web server.

However, it lacks support for advanced properties such as openness and is not suitable for systems requiring these properties.

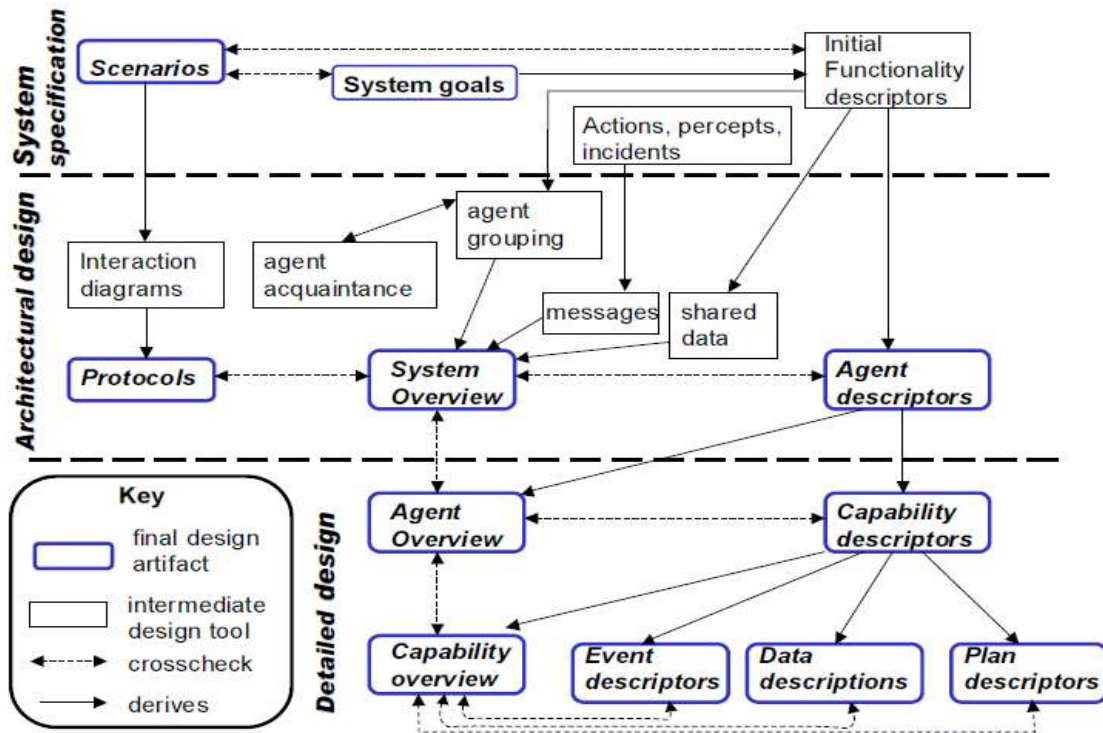


Figure 3.3: Prometheus Overview (extracted from Padgham & Michael, 2012)

3.3.1 Advantages of Existing System.

These are some of the advantages of the existing agent-oriented methodologies which are based on strong disciplined foundations:

- 1) The existing system supports both levels (micro and macro levels) to construct and develop agent systems.
- 2) ROADMAP and MaSE are considered as an extension of the software engineering approach provides a solid base for the development of multi-agents systems.
- 3) Prometheus use well-known techniques such as UML, which is particularly interesting. These techniques facilitate comprehension and communication between the various agents involved during software development (Sabas, Delisle and Badri 2012).
- 4) Prometheus constitute an extension of knowledge-based methods provide models that take into account the agents' internal states much better.
- 5) ROADMAP provides relatively elaborated support for reusable models, which is a valuable aspect for any methodology.

3.3.2 Disadvantages of Existing System.

The existing agent- oriented methodologies suffer from some difficulties which are the main reasons for a number of limitations emerge. Those difficulties prevent agent-oriented methodologies from being utilized and practiced in a wide manner.

- 1) There is no existing agreement or accord on agent theory. Up until this point in time, no agent-oriented standards have been established and accepted as standard. No agent-oriented methodology will be able to spread unless the agent model is standardized. This standardization is refers to what characteristics define an agent, what types of architecture are available for agents, what agent organizations are possible, and what types of interactions there are between agents, etc.
- 2) None of these methodologies are used and none are exploited in a wide manner.
- 3) All research that examined and compared properties of these methodologies has suggested that none of them are completely suitable for industrial development of MAS.
- 4) There is no systematic approach to identify the components of MAS. Most current methodologies require the designers and developers to identify all agents of the system. Therefore, a designer experience is very important and is essential for producing a quality MAS. Designers should be trained beforehand to have the necessary skills for such projects.
- 5) Although, there are new languages for programming agent behavior, there are no adequate development tools for representing agent structure. Languages tend to focus mainly on particular agent architecture.
- 6) Selecting a suitable methodology to be followed for MAS development processes is not an easy task. Therefore, a precise methodology needs to be presented to guide the team of developers towards the achievement of objectives.
- 7) Comparing methodologies is often difficult. This difficulty arises from the fact that it is not easy to evaluate them because they usually differ in their premises, covered phases, models, concepts and the supported multi-agent system properties.
- 8) There is no agreement on what a methodology is and on what it should consist of.

3.4 Analysis of the Proposed System

Building software to solve contemporary business problems is no easy task. Over the last decade there has been an increasing focus on object-oriented notations and modeling languages, perhaps at the expense of a full methodological approach to solving the problem and giving software developers the tools they need to comprehensively create applications within management and market constraints—money, time, quality, and so forth (Brian,2012).

With increasingly sophisticated applications being demanded by businesses aiming for a competitive market advantage, object technologies are being supplemented and complemented by agent technologies. This is especially true in areas such as ambient intelligence, e-business, Web services, peer-to-peer networks, and bioinformatics. These areas demand software that is robust, which can operate within a wide range of environments.

They evolve over time to cope with changing requirements that are highly customizable to meet the needs of a wide range of users, and are sufficiently secure to protect personal data and other assets on behalf of its stakeholders. To fulfill these requirements, builders of systems need an appropriate agent-oriented methodology (Paolo, 2011); this is the focus of this work.

Agent-based systems call for a novel concepts, tools, and techniques for engineering and managing software. In particular, we need an enhanced software development methodology that supports the design and implement organizations of agents able to interact with one another in order to achieve some common or individual goal (Paolo, 2011).

The comparative analysis is one step towards the development of an enhanced agent oriented methodology that may be formed by merging the various existing individual Agent Oriented methodologies. If the creation of a single universally applicable methodology is an attainable goal, then we must work towards the creation of a methodological environment in which the various demands of different software developers might be satisfied.

These requirements are classified into three types of categories that new methodology should comply with: concepts, modeling techniques, and processes.

In this dissertation an Enhanced Software Engineering Methodology was developed for Analysis and Design of Intelligent Agent and is called An Enhanced Multi-Agent System Development (EMASD) methodology for development of Intelligent Agent systems.

This new methodology is established based on concepts, models, and process. These three aspects are considered as a foundation for building a solid methodology. Furthermore, this methodology is developed based on the essential software engineering issues such as preciseness, accessibility, expressiveness, domain applicability, modularity, refinement, model derivation, traceability, and clear definitions.

The Enhanced Multi-Agent System Development (EMASD) methodology is provided by a plain and understandable development process through the methodology phases. It captures the holistic view of the system components, and commutative aspects, which should be recognized before designing the methodology models. The new methodology covers the whole life cycle of agent system development, from requirement analysis, architecture design, and detailed design to implementation.

System Development Life Cycle (SDLC) methodology was adopted during the development of Enhanced Multi-Agent System Development (EMASD) methodology.

3.4.1 Advantages of the proposed System

In this section we present an Enhanced Multi-Agent System Development (EMASD) methodology, consisting of the common elements identified from MaSE, Prometheus and ROADMAP.

These are some of the advantages of the Enhanced Multi-Agent System Development (EMASD) methodology which are based on strong disciplined foundations:

1. The Enhanced Multi-Agent System Development (EMASD) methodology is independent of the implementation architecture and supports analysis and architecture design of the system.
2. It consists of six basic models as in figure 3.2. During the analysis stage, the models are created to conceptualize and document the system requirements. During the architecture design stage, the same models are refined and optimized for the given quality goals. During these two stages, agents are considered as black boxes. At the end of the architecture design phase, implementation architecture, such as the BDI architecture, is chosen for each agent.
3. The process of the Enhanced Multi-Agent System Development (EMASD) methodology is simple. A developer can create the models in order of listing, and fill in details of any model when possible.

4. The Enhanced Multi-Agent System Development (EMASD) methodology features are architecture-independent and can be added to the skeleton methodology to facilitate analysis and high-level design.
5. The Enhanced Multi-Agent System Development (EMASD) methodology is provided by a plain and understandable development process through the methodology phases. It captures the holistic view of the system components, and commutative aspects, which should be recognized before designing the methodology models.
6. The new methodology covers the whole life cycle of agent system development, from requirement analysis, architecture design, and detailed design to implementation.
7. The Enhanced Multi-Agent System Development (EMASD) methodology should be based on robust concepts of agent system and MAS. Therefore, it should have a complete conceptual agent and MAS structure.
8. The Enhanced Multi-Agent System Development (EMASD) methodology should rely on a plain, specific conceptual framework, which is responsible for specifying and linking the concepts during the different construction stages.
This conceptual framework is considered as a foundation in the different phases of construction.
9. The new methodology should be able to model the mental aspects of agents such as beliefs, goals, and plans. Such aspects play a crucial role in determining how rational agents will act.
10. The methodology should be able to support existing agent architectures in order to specify how the agent can be decomposed into a set of component modules and how these modules should be made to interact.
11. The Enhanced Multi Agent System Development (EMASD) methodology bridges the gap between design and implementation. This is achieved by providing refined design models such as an agent container in the design phase. They can be directly transferred into implementation constructs in an available programming language. It provides constructs to directly implement high-level design concepts.
12. The methodology should provide concepts which are used to specify and represent changes in the environment, e.g. events, incidents, etc. Therefore, it should include a trigger concept for agents to represent its autonomy and reactivates characteristics.

3.4.2 Justification for the Proposed System.

From the analysis carries out on the Agent Technology, individual agent-oriented methodologies are useful for restricted situations, a more flexible approach can be found in the use of an enhanced model. Using an underpinning Meta model, a repository of method fragments can be built up and, from this, a selected number of fragments can be abstracted to form an organization-specific or project specific methodology. Hybrid agent oriented methodology for intelligent agent system will be created from existing individual agent oriented methodologies, such as Prometheus, is demonstrated with further enhancements from other methodologies such as ROADMAP.

3.5 Implementation Model of the Proposed System

In the Implementation model for an enhanced agent oriented methodology we need to look at this model as characterized by two views: the multi-agent and single-agent views. The outer level of iteration (dashed arrows) concerns the dependencies between multi-agent and single-agent views. The first (multi agent) view relates to the agents' structure (in terms of cooperation and tasks involved) and behaviors (flows of events depicting cooperation). The second one instead relates to the single-agent structure (attributes, methods, inner classes) and behavior (specified in an appropriate way). The inner level of iteration (Agent Structure Definition – Agent Behavior Description) takes place in both the multi-agent and single-agent views and concerns the dependencies between structural and behavioral matters

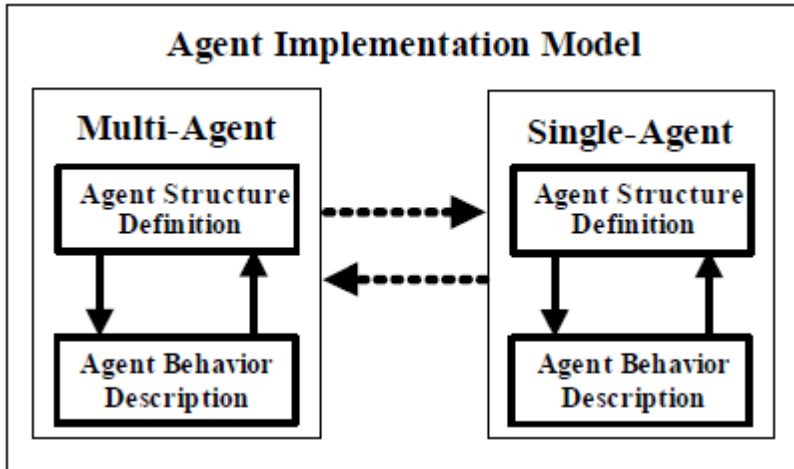


Figure 3.4. Agent Implementation Model.

CHAPTER FOUR

SYSTEM DESIGN AND IMPLEMENTATION

Enhanced Multi-Agent System Development (EMASD) methodology is developed as a reliable systematic approach that proves a milestone for Software Development Life Cycle (SDLC). The proposed methodology covers the most important characteristics of multi-agent systems. The proposed methodology deals with the agent concept as a high-level abstraction capable of modeling a complex system. In addition, it includes well-known techniques for requirement gathering and customer communication and links them to domain analysis and design models such as UCMs (Buhr& Casselman, 2011), UML Use Case Diagrams (UML Use case diagrams), Activity diagrams (UML Specification, 2012). Furthermore, it supports simplicity and ease of use as well as traceability.

The Enhanced Multi-Agent System Development (EMASD) methodology is composed of four main phases; the system requirements phase, the analysis phase, the design phase, and the implementation phase. The next sections present a more discussion of each of the four phases. A Traveller Agent System (TAS) is used to describe the process of EMASD methodology.

The problem consists of building a system that is consulted by a user for booking a flight, buses, hotels and answers with the cheapest available flights, buses and hotels with lowest probability of being delayed. The system will be run by any company, and the information of the flights, bus and hotel will be available from the respective companies.

4.1 Objectives of the Design.

The Principal goal of this design process is to produce an enhanced model for the development of intelligent agents which is the design of the proposed system. Based on the bottlenecks of the existing systems, design requirement specifications were established and aligned with the implementation phase.

The design objectives are to:

1. Bridge the gap between the problems of existing systems and the solutions of the proposed system.
2. Create a platform which is very interactive and users-friendly for all stakeholders
3. Ensure strong security proof for the corporate network.

4.2 Decomposition and Cohesion of the High Level Model

The system requirements phase describes all the details of a system scenario as a high-level design through system scenario model. The system scenario model uses well-known techniques such as Use-Cases Diagrams (UCDs) and Use Case Maps (UCMs) (Buhr 2010) to describe the whole system scenario. Such techniques assist to discover the system components such as agents, objects, roles, resources etc. and their high-level behavior. The system requirements phase produces a model called the **system scenario model**.

System Scenario Model

This model is used as a starting point for generating more detailed visual descriptions. It describes the whole system scenario in terms of what a system does, but it does not specify how it does it. The model captures the components that the system is composed of and the tasks that have to be performed by each component within the system. Then, it illustrates how these components interact with each other and with the external environment. In addition, it captures the behavior of a system as it appears from the point of view of the outside user. To construct this model, some specific, well-known techniques have been used such as Use-Case Diagrams (UCDs) and Use Case Maps (UCMs). These techniques are assembled together in order to understand and obtain a complete system requirement as far as possible.

In the system scenario model, UCDs are exploited to describe the behavior of the system from the user's point of view. It is through this notation that the roles in the system can be recognized. Recognition of roles within a system is very helpful during the analysis and design phases as well as for understanding the system's requirements.

Also, in the system scenario model, UCMs are used as a precise structured notation. UCMs describe system scenario in terms of causal relationships between responsibilities. They also emphasize the most relevant, interesting and critical functionalities of the system. They describe the general behavior of the system in the form of scenarios without referring to any implementation details.

UCMs include adequate information in a summarized form. It has two advantages:

- It enables developers to understand and conceptualize the behavior of the system as a whole.
- It gives an explicit concept overview about how the system operates as a whole.

Analysis Phase

The objective of the analysis phase is to transform the system requirements into a representation of the system that can be forwarded to the design phase. The analysis phase is considered to be the most important process of the methodology. This phase starts with analyzing the system requirements phase; it utilizes the system scenario model that is constructed by UML use-cases and use-case maps. This system scenario model is considered as a base to produce the models of the analysis phase. The analysis phase is concerned with the description of the agent architecture as well as the MAS architecture. It is divided into two stages. The first stage describes the agent architecture. The second stage describes the MAS architecture. The agent architecture stage describes the internal structure (roles, beliefs, goals, plans and triggers) of agents in the system. In contrast, the MAS architecture stage describes the relationships between agents, the conversations and exchanged messages and agent services. This description of the MAS architecture is important in order to facilitate two main functions:

1. To enable negotiation and cooperation between agents.
2. To establish commitments and agreements that the agents should adhere to in order to provide the services to other agents in the system.

Agent Architecture Stage

The agent architecture stage describes the following models:

- Roles model: Discovers the roles that agents play or perform in the system, determines responsibilities for each role and specifies activities for each responsibility.
- Agent model: Identifies agents in the system and assigns roles to them. Refines the roles to fit agent capabilities.
- Beliefs model: identifies agent beliefs.
- Goals model: Identifies agent's goals.
- Plans model: Specifies plans for each goal.
- Triggers model: Identifies the triggers that each agent should be aware of as being events that take place in the system. The EMASD methodology requires the development of all models of

the agent architecture stage. They are always developed even if the proposed agent system is just as a single agent.

Roles Model

The agent role represents an agent behavior that is recognized, providing a means of identifying and placing an agent in a system. Role modeling is appropriate for agent systems (Kendall, 2011) because of the following reasons:

1. Roles and role models provide a new abstraction that can unify diverse aspects of a system. Software agents, objects, processes, organizations, and people can play roles, and this is especially important in applications that encompass all these types of entities, such as information and process management.
2. Role models are patterns that should be documented and shared.
3. Role model synergy integrates roles and may be valuable for agent design.
4. Role model dynamics can be employed to model mobility, adaptive behavior, context switching, and other aspects of agent systems.

Furthermore, the roles model presents the agent system as an organization by considering it as a set of roles that work together. Each role has its own responsibilities. These roles improve and systematize the agent functionality and emphasize social or interactive behavior.

The agent can perform more than one role in the system and more than one agent can perform the role. The roles as encapsulated units can be transferred easily from one agent to another when there is a need. The roles model is the first task in the analysis phase. In this model, the roles that an agent plays in the system are discovered. It includes the following three detailed steps: discovering roles, determining role responsibilities, and specifying activities for each responsibility.

Discovering Roles

This step is responsible for identifying the main roles that are found in the system. In order to be able to capture those roles, UCMs and UML use cases scenarios are to be exploited. In the system scenario model, the UCM components that are involved in the system are identified.

These components could be agents, objects, or actors. Roles are discovered by analyzing path segments that cross UCM components in the system scenario model.

Determining Responsibilities of the Roles

Once roles have been identified then the next step is to determine the duties and **responsibilities** of each role separately. This process starts by tracing scenarios of use case diagrams that have been developed during the system requirements phase, Identifying each actor individually and determine all its use-cases, then transferring them directly (one- to-one) to responsibilities in the role that it plays in the system. The scenario paths of the UCMs are then traversed and all the responsibilities and stubs are individually defined and transferred directly to responsibilities and functions that are carried out by the role. This process is an attempt that most of responsibilities and functions of each role are fully, clearly and accurately captured.

Specifying Activities of Each Responsibility

Once the responsibilities and functions of each role are individually identified, then the following step will identify all the **activities** undertaken by each responsibility. This will in fact, represent the functions of the proposed role to be implemented in the system.

The important attributes of the roles model are: **role name**, **role description**, **responsibilities**, **permissions**, **perceptions**, **obligations** and **constraints**. The **role name** states the name of the role. The **role description** is a textual explanation of the function of the role.

Responsibilities are the activities that the role is responsible to perform. **Obligations** are requirements that should be available to enable the role to start its functionality and carry out its responsibilities and activities. **Permissions** are the authorities related to numbers and types of resources that will be exploited by agents in the system. **Constraints** are restrictions and boundaries that the role must not violate through executing its tasks. Developers systematically apply phrase heuristics to classify the statements as permissions, obligations, or constraints. Heuristics include modality (can, May, must), condition key words (if, unless, except) and English conjunctions (and, or, not). Developers must document their interpretation (e.g.,

“may” indicates a permission) and assign logical meanings to each conjunction. Due to logical disjunctions, each sentence may have multiple obligations, permissions, and constraints.

Agent Model

The agent model describes the internal structure of agents within the system and how these agents employ its internal structure to perform its tasks. In the EMASD methodology, the building process of the agent model is based on BDI agent architectures (Georgeff, Pell, Pollack, Tambe, and Wooldridge 2012). The BDI architecture is used to determine the actions that an agent performs. Each agent possesses one goal or more, which it desires to realize. In addition, an agent has beliefs that it depends on to achieve its goals. It is assumed that agents have a library of goals available to them, each goal containing a set of predefined plans. Each plan contains a set of predefined tasks. Tasks are not necessarily atomic; they could be a single task or a sequence of tasks that form a plan. The term plan is used to achieve a specified goal. Each plan has a set of preconditions and post conditions associated with it. In order for the tasks of that plan to be executable, the preconditions for that plan must be satisfied. These preconditions and post conditions could be considered as the agent’s beliefs that it needs to hold in order for it to be able to select the appropriate plan to achieve the goal. Once the plan has been executed, its post conditions are applied. Executing a plan can cause changes to the state of the environment. Agents also have triggers. These triggers assist them to determine the appropriate goal or plan to be selected. The behavior of the agent is determined solely by its concrete beliefs, goals, and plans. The agent model describes in detail the following steps: Identifying agents, refining roles, beliefs model, goals model, plans model, and triggers model.

Identifying Agents

The agent identification step is performed to extract those agents that are assumed to exist within the system. These agents are identified using use case maps that have been developed during the system requirements phase. Agents are identified by analyzing UCM components. A component can be identified as an agent or several components that can be combined to constitute one agent. Hence, several roles are combined into one agent.

Refining Roles

The refining roles step is merely used to revise the roles that the agent plays within the system. The refinement process consists of two steps. The first step is to match the roles that are captured in the roles model with agents that play these roles according to the agent's capabilities. The role's responsibilities are classified based on who is responsible for performing them. The second step is to separate, or isolate those responsibilities that are to be carried out by real persons from those responsibilities that are to be carried out by agents on their behalf.

The refining roles process keeps the responsibilities that are to be carried out by the agent within the roles model. The responsibilities that are to be carried out by real users are stated as preconditions. These preconditions are translated into beliefs. Agents use these beliefs to keep track of whether the real person performs those responsibilities or not. Agents should be able to sense the environment to check whether these beliefs are changed or not. In other words, an agent may wait for a signal (e.g. a message) that confirms that a task performed by the real user has been completed. This refinement process assists developers to build a clear design that is free from confusion and a responsibility overlap.

Agent Beliefs Model

The agent knowledge is considered one of the most important aspects of the agent system.

It stores relevant facts about the agent and its environment. Agent knowledge may be taken to explicitly represent the agent's beliefs about its environment or even about itself or about other agents. The following sub-sections show how the agent beliefs are identified. The beliefs model in the EMASD methodology is carried out via the system scenario model and the roles model. The agent beliefs are identified either by the preconditions or by postconditions of the agent's plans and goals, or by the obligations, permissions, and constraints that were obtained in the roles model. Furthermore, the beliefs can be obtained by tracing the UCM scenarios.

The stubs and responsibilities are considered as bases of beliefs that are used to trace whether these stubs and responsibilities are achieved by the agent or not. In addition, the beliefs store information about the internal state of agents. Agent beliefs are classified into two types: *constant belief*, these beliefs are set beliefs and not allowed to change, and *variable beliefs*, the values of these beliefs can change many times. Beliefs can be assigned initial values or their

values are computed using some kind of expressions or deduced by inference rules. According to Parsons (2011) it is reasonable to assume that the values of the beliefs are obtained in several ways:

- 1) Initial beliefs (basic facts which represent the agent's initial beliefs).
- 2) Beliefs deduced from previous beliefs by deductive inference rules.
- 3) Beliefs obtained as answers to questions put to the environment by the agent.
- 4) Beliefs perceived by a sensor (facts that the agent perceives in its environment).
- 5) Beliefs communicated by external agents (messages received from other agents).

Also EMASD classified the purposes of the beliefs as the following: Storage belief, when the belief is stored and the agent can use it during its lifecycle; maintain belief, when the agent must keep the belief at a certain value e.g. when the agent must keep the temperature constantly at 20 degrees; and achieve belief, the agent stores a required value of the belief and during its lifecycle tries more than one time to check the value of the belief and run plans if the value is not the required value. The agent may not be able to change can achieve belief to a required value but it must keep the value of a maintained belief true at all times.

These classifications and its purposes assist the developers to identify the mechanism of how the beliefs are stored and exploited. Accordingly, the agents will be able to reason about the beliefs to select the appropriate actions.

The agent beliefs model deals with only some types of beliefs that were mentioned previously. The focus is on those perceived by sensors, those placed as initial beliefs, and those obtained as an answer to questions put to the environment by the agent. The beliefs that are communicated by other agents as the messages received from other agents, are treated as communication messages covered in the agent interaction model. Due to the fact that agents within the system could possess many beliefs, we will provide the beliefs model for the customer agent only for simplicity.

Agent Goals Model

The goal represents a specific target state that the agent is trying to achieve. In a goal-oriented design, goals are considered explicitly as states to be achieved. Therefore, goals also define reasons to execute agent actions. When actions fail, it can be checked if the target state is already achieved, if not, it would be useful to retry the failed action or try out another set of actions to achieve the target state.

The agents' internal structure in the EMASD methodology is based on the BDI architecture (Bratman 2010; Rao and Georgeff 2010; Cohen and Levesque 2010; Kinny, Georgeff, and Rao 2011). The EMASD methodology assumes that the concept of goals in relation to agents has a very strong relation with the BDI architecture. The goals represent the desires and intentions that the agent possesses. The definition of the relationship that links goals with the desires and intentions is formulated as being a similarity or matching relationship. Intentions are considered and are defined as being goals that possess previously prepared plans to be executed. Desires are defined as goals with no plans for future execution.

In this model, goals are identified for each agent in the system. These goals represent a mechanism, which leads the agent to perform its actions in an orderly and smooth way. The MASD methodology supports two types of goals (long-term and short-term) in the form of goals and sub-goals.

The EMASD methodology deals with short-term goals as the goals of the agent, which will be achieved during system runtime. This type of goal is obtained through the methodology process by capturing the agent goals from roles model. More details are available in the agent goals model in the analysis phase. The EMASD methodology deals with long-term goals as the strategic goals of the system. This kind of goal cannot be obtained through the methodology process like the short-term goals. They should, however, be deduced by the designer in order to identify the sub-goals (short-range goals) and then determine the conditions of use.

The goals model specifies how to obtain the goals of the agent through the role or roles that it will play within the system. In order to identify the goals of the agent, we have to convert each responsibility of a given role to a specific goal. Therefore, it can be stated that each responsibility within a specific role is considered a goal for the agent who plays the role. Moreover, each activity within a specific responsibility is the foundation for one plan of the goal. In the agent goals model, the goals that the agent desires to achieve are identified. Each goal and its priorities will be identified. Each goal will be initiated according to its preconditions and a specific priority. The plans, which are prepared by the agent to satisfy the desired goal, will also be identified. This model also contains preconditions and post conditions to initiate the process of achieving goals that the agent desires to realize.

Agent Plans Model

After the goals of the agents we are identified by the previous step, it is time to describe the plans that should be followed by an agent in order to achieve its goal. Since each agent has a goal or set of goals that it wants or wishes to achieve, a plan or a set of plans for each individual goal must exist. Such a plan needs to be adhered to and followed in order for it to be achieved or performed. Each of those plans consists of a set of tasks to be executed.

Specifying Plans for each Goal

Plans are a deliberately prepared means through which agents achieve their goals. A plan is not just a sequence of basic actions, but it may also include sub-goals.

Other plans are executed to achieve the sub-goals of a plan thereby forming a hierarchy of plans. The agent keeps track of the actions and sub-goals carried out by a plan to determine and handle plan failures.

Plans are specified by matching and transforming the activities that belong to the responsibilities within the roles. Each plan consists of a set of tasks. These tasks implement the plan and they will complete the required work. Completion and implementation of these tasks is considered the as success of the plan. A given goal is considered to be accomplished if at least one plan related to it was implemented. Plans may be executed in a sequential manner, according to the priority of each plan, or in parallel manner. In the plans model the plans that should be followed or that have to be selected by an agent during achieving a specific goal are recognized. In other words, every goal has to be achieved through one specific plan or more. Plans are adopted by agents and, once adopted, constrain an agent's behavior and act as intentions.

The plans model consists of six parts: a plan name, preconditions, post conditions, successful internal actions, failed internal actions and a plan body. Optional **preconditions** define the preconditions of the plan, i.e., what must be believed by the agent for a plan to be executable. **Post conditions** are conditions that must be true for the plan when it completes. **Successful internal actions** are the actions that are performed if the plan succeeds. **Failed internal actions** are the actions that are performed if the plan fails. Finally, the **plan body** defines a tree

representing a kind of flow- graph of actions to perform. UML activity diagrams (UML Specification 1997) are used to represent the plan body. Activity diagrams are used to model the workflow of the process of the internal operation of the agent system. Activity diagrams illustrate the dynamic nature of the agent system by modeling the flow of control from one activity to another.

Executing a plan successfully involves traversing the activity diagram from the start node to the end node. Activity diagrams show the dynamic nature of the system, which is emphasized by the representation of the flow of control between the tasks in the plan. No doubt, that being able to represent these tasks in clear and detailed manner will help developers and programmers in representing and implementing them easily and with more flexibility.

Agent Triggers Model

This model identifies and captures triggers that occur during system runtime. The idea of the trigger concept is somewhat similar to the ECA rule (event, condition and action (Dittrich, Gatzju &Geppert, 2011)). Triggers are the events and the changes in the beliefs. All events and the change of beliefs that are expected to occur within the system are identified. This model helps designers and developers to identify these events and select the appropriate reaction for such triggers. Triggers can be caused by information coming from the environment, which has an effect on the behavior of agents. According to that information, the agent performs certain actions as a reaction.

Triggers are obtained by capturing and analyzing the beliefs of each agent that could be changed during runtime. Triggers are also obtained by capturing and identifying the expected events that will occur in the system during runtime. The selection of triggers that prompts a goal or a specific plan is then followed by transferring them into triggers that motivate the agent to perform some given reactions. The triggers model consists of four attributes: The trigger name, trigger type, trigger activator and the actions. Each trigger is identified by a unique **trigger name**. The **trigger type** can be either an event or a change of belief. The **trigger activator** represents the entity that is responsible for causing such trigger. This entity can be an

agent, an object, or a particular resource within the system. **Actions** are either goals to be achieved or plans to be executed.

EMASD Methodology Architecture

Enhanced Multi-Agent System Development (EMASD) methodology is developed as a reliable systematic approach that proves a milestone for Software Development Life Cycle (SDLC). Figure 4.1 illustrates the process of Enhanced Multi-Agent System Development (EMASD) Methodology. The proposed methodology covers the most important characteristics of multi-agent systems. The new methodology deals with the agent concept as a high-level abstraction capable of modeling a complex system.

Furthermore, the Enhanced Multi-Agent System Development (EMASD) methodology architecture is developed under the umbrella of The Belief Desire Intention Architecture. The BDI architecture is one of the most well-known and studied software agents' architectures (Georgeff, Pell, Pollack, Tambe, & Wooldridge 2011). This architecture consists of four basic components: beliefs, desires, intentions, and plans. In this architecture, the agent's beliefs represent information that the agent has about the world, which in many cases may be incomplete or incorrect (d'Inverno, Kinny, Luck & Wooldridge, 2011). The content of these beliefs can be anything from knowledge about the agent's environment to general facts an agent must know in order to act rationally. The desires of an agent are a set of long-term goals, where a goal is typically a description of a desired state of the environment.

In addition, The EMASD methodology is composed of four main phases; the system requirements phase, the analysis phase, the design phase, and the implementation phase. The next sections present a more discussion of each of the four phases.

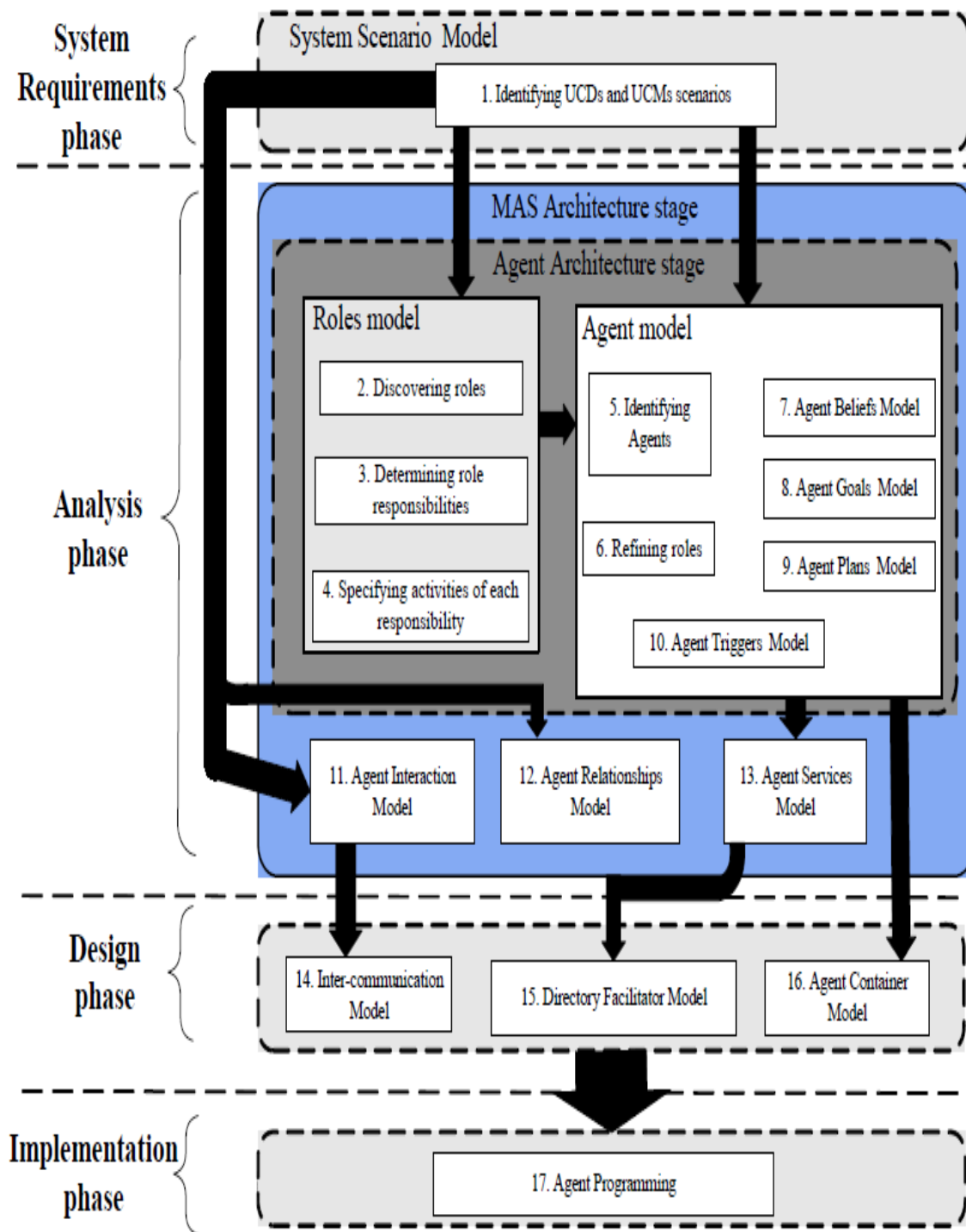


Figure 4.1 Enhanced Multi Agent System Development Methodology

4.2.1 Main Menu.

The main menu of the enhanced model for development of Intelligent Agent shows their feature and structural analysis.

Furthermore, the main menu of the TIAS will show its major operation and how each module will be accessed from the main program and the function of each module in the harmonious working of the entire system.

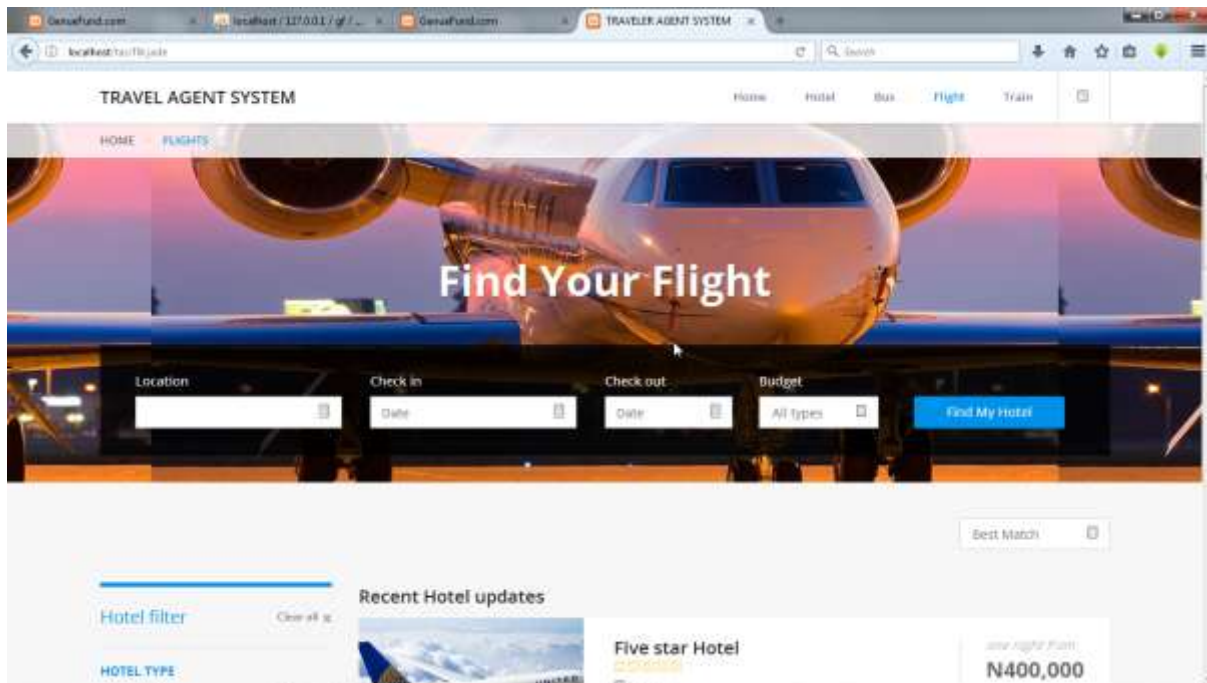


Figure 4.2 Main Menu

4.2.2 The Sub menus/Subsystems

Use case Diagram for Airline Reservation system

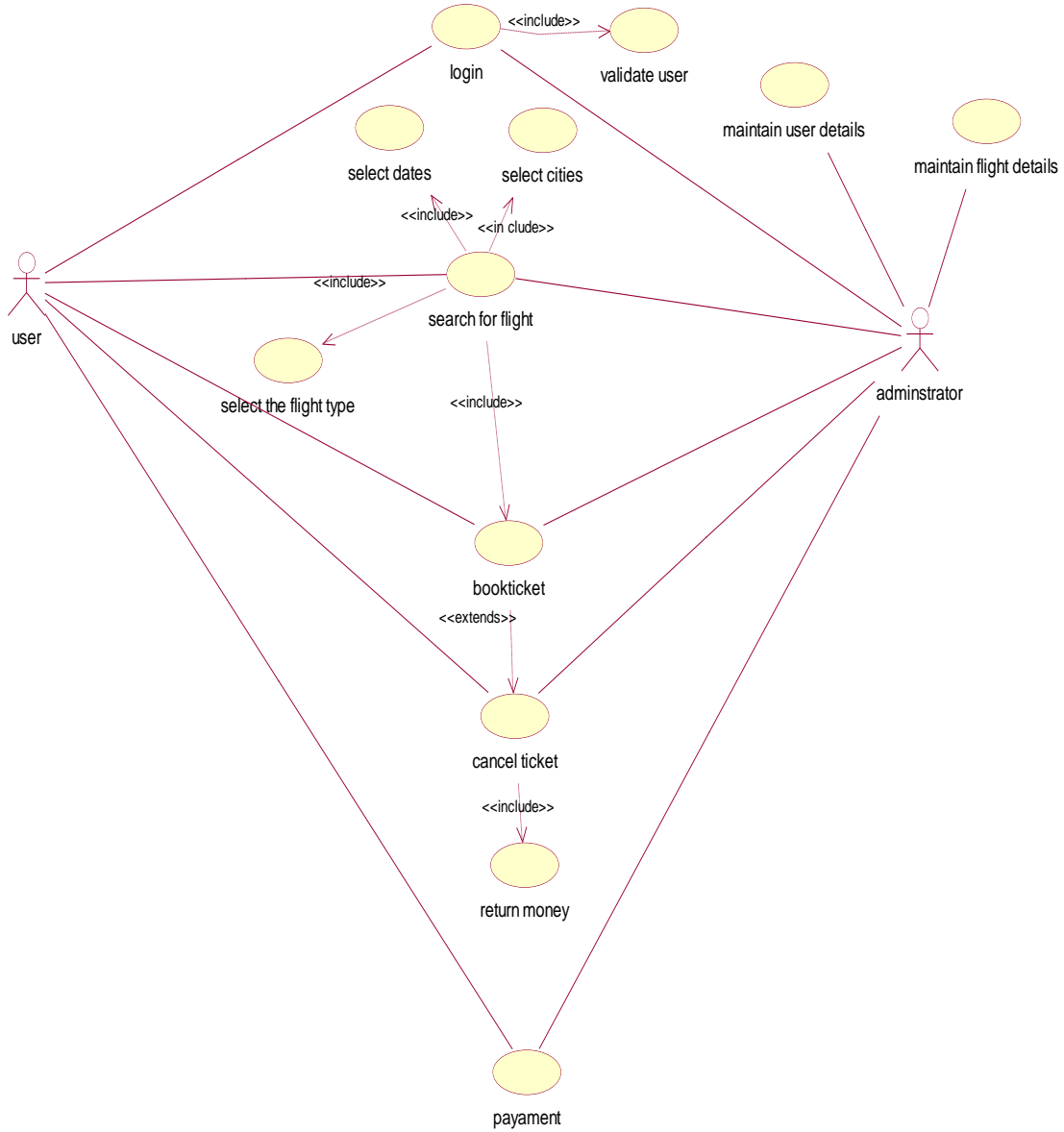


Figure 4.3 Use case Diagram for Airline Reservation system

Subclass Use case Diagram for Airline Reservation Systems:

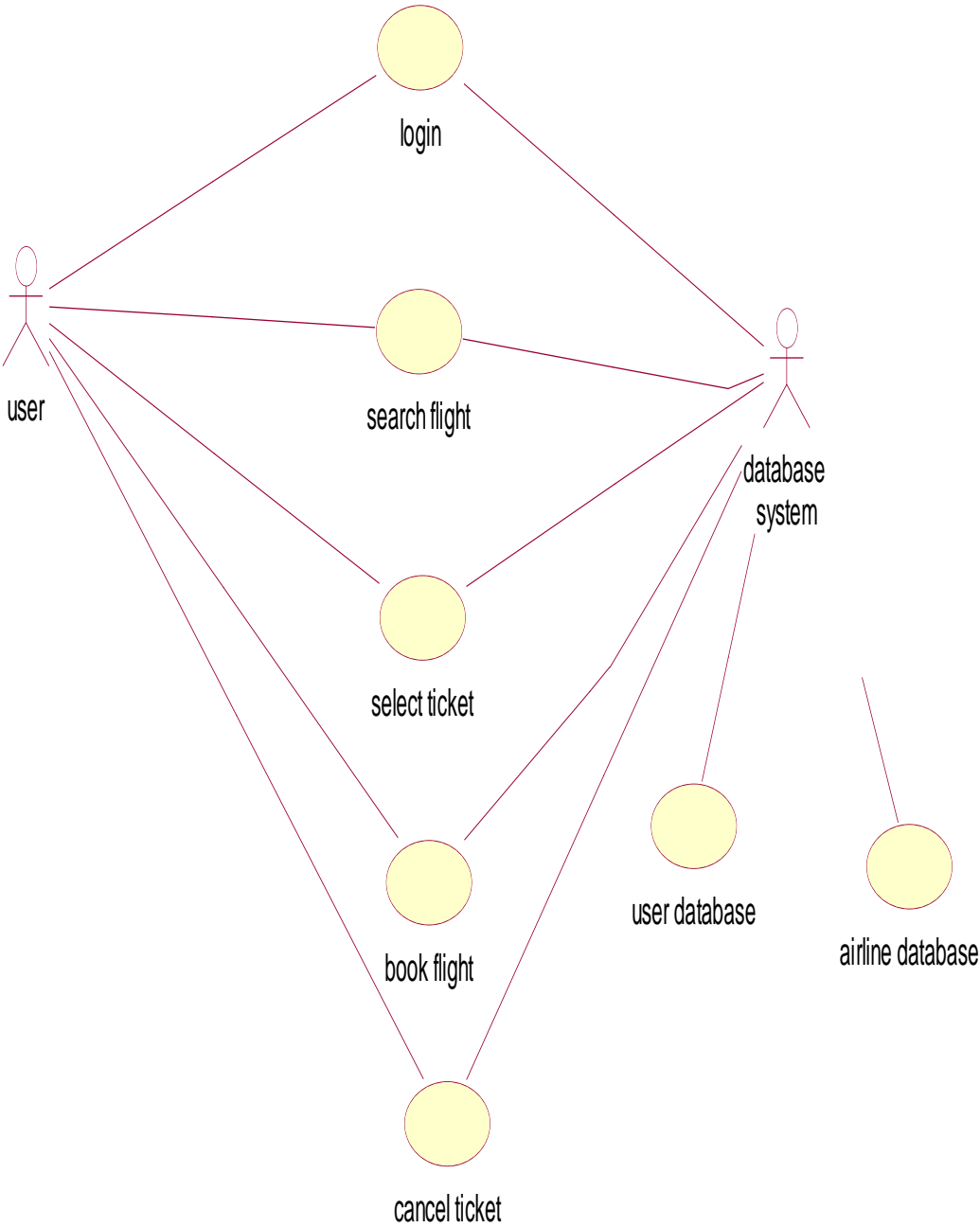


Figure 4.4: Subclass Use case Diagram for Airline Reservation Systems

4.3. Specification

4.3.1 Database Specification

DB DESIGN:

Entities:

1. Centralized password
2. Designation
3. Login
4. Agent Details
5. Action Section
6. Instruction Details
7. Occurrence Details:
8. Users Details
9. Property Details:

Entities with Attributes:

1. Centralized password
2. Designation
 - Instructor
 - Sub Control
 - System Control

3. Login

- User ID
- Password

4. Agents details:

- Agent No.
- Agent Information

5. Action Section

- Environment
- Software
- Actions

6. Section

- Type of Operation
- Contact Points

7. Occurrence of Operation:

- Date
- Time
- Type of Information

8. Property Details:

Property Details

Total Value

Table 4.1: Database Tables

SNO	COLUMN NAME	DATA TYPE (SIZE)	CONSTRAINTS (KEY)	REFERENCES FROM
1	User ID	VARCHAR(15)	PRIMARY KEY	
2	Password	VARCHAR(15)	VARCHAR(15)	

Table 4.2: Record Management:

SNO	COLUMN NAME	DATA TYPE (SIZE)	CONSTRAINTS (KEY)	REFERENCES FROM
1	User ID	VARCHAR(15)	PRIMARY KEY	
2	Password	VARCHAR(15)	NOTNULL	
3	Designation	VARCHAR(15)	NOTNULL	

REPORTS:

Table 4.3: Agent Details

SNO	COLUMN NAME	DATA TYPE (SIZE)	CONSTRAINTS (KEY)	REFERENCES FROM
1	AG_NO	VARCHAR(15)	PRIMARY KEY	
2	AG_INFORMATION	VARCHAR(15)	NOTNULL	

Table 4.4: Action Section Details

SNO	COLUMN NAME	DATA TYPE (SIZE)	CONSTRAINTS (KEY)	REFERENCES FROM
1	Agent no	VARCHAR(15)	PRIMARY KEY	FIR DETAILS(FK)
2	Environment	VARCHAR(15)	NOTNULL	
3	Software	VARCHAR(15)	NOTNULL	
4	Action	INT(4)	NOTNULL	
5	Section	INT(4)	NOTNULL	
6	Type_Of_Operation	VARCHAR(15)	NOTNULL	

Table 4.5: Occurrence Operation

SNO	COLUMN NAME	DATA TYPE (SIZE)	CONSTRAINTS (KEY)	REFERENCES FROM
1	Ag_NO	VARCHAR(15)	PRIMARY KEY	FIR DETAILS(FK)
2	Date	DATETIME	NOTNULL	
3	Time	DATETIME	NOTNULL	
4	Type_Of_Information	VARCHAR(15)	NOTNULL	

Table 4.6: Report Generation by Type of Operation

SNO	COLUMN NAME	DATA TYPE (SIZE)	CONSTRAINTS (KEY)	REFERENCES FROM
1	General details	VARCHAR(15)	NOTNULL	
2	Full Details	VARCHAR(15)	NOTNULL	

Table 4.7: Report Generation by Date

SNO	COLUMN NAME	DATA TYPE (SIZE)	CONSTRAINTS (KEY)	REFERENCES FROM
1	From Date	VARCHAR(15)	NOTNULL	
2	To Date	VARCHAR(15)	NOTNULL	

4.3.2 Use Case Description

Use case specifications for login system:

1. Use case name: Login system
2. Flow of events:
 - 2.1 Basic flow: The customer enters the valid login details in login system.
 - 2.2 Alternate flow:
 - 2.2.1 Invalid user name
The customer enters the invalid values.
3. Special requirements: User can enter as a guest.
4. Pre-conditions: There are no preconditions.
5. Post conditions: There are no post conditions.
6. Extension points: There are no extension points.

1. Use case specification for search flight:

1. Use case name: Search Flight.

This use case is started traveler. It provides the facility to search the flights available.

2. Flow of events:

- 2.1 Basic flow:

This use case is started when the traveler enter the details such as departure city and arrival city. If the names are invalid alternate flow 2.2.1 is executed. The system then checks for the list of flights available and print them. If the flight is not available alternate flow 2.2.2 is executed.

2.2 Alternate flow:

2.2.1 If the traveler enters the city with errors such as “arrival date” is precede departure date or entering the dates is already completed or the cities are invalid then the system informs the traveler that returns the details.

2.2.2 If the flight is not available between the two cities that the user enters then the system display the message that “there is no flight service directly between two cities.

3. Special requirements: There are no special requirements.

4. Pre-conditions; There are no pre conditions.

5. Post conditions: There are no post conditions.

6. Extension points: There are no extension points.

Use case specifications for selecting flight:

1. Use case name: Select Flight

This use case is started by traveler. It provides the facility for traveler to select a flight from a list of available flights

2. Flow of events:

2.1 Basic flow:

This use case is started after the search flight is completed by traveler. The system chooses a flight from the list of available flights if the search system finds any flights between the roots. If there are no seats available alternate flow 2.2.1 is executed.

2.2 Alternate flow:

2.2.1 If there are no seats available on the selected flight then the system informs the traveler to choose another flight.

3. Special requirements: There are no special requirements.

4. Pre-conditions: There are no pre conditions.

5. Post conditions: There are no post conditions.

6. Extension points: There are no extension points.

Use case specifications for booking a flight:

1. Use case name: Book Flight.

This is use case is started by the traveler. It provides the facility for the traveler to book tickets.

2. Flow of events:

2.1 Basic flow;

This use case is started after the traveler chooses a flight. The system then asks the traveler to enter his/her details and credit card number. The system then checks the credit card number. The system then checks the credit card validity through credit card authorization mechanism and books the tickets else alternate flow 2.2.1 is executed. After booking the tickets the systems update databases.

2.2 Alternate flow:

2.2.1 If the credit card is not valid the system asks the traveler to re-enter the credit card number correctly.

3. Special requirements: There are no special requirements.

4. Pre-conditions: There is availability of seats in the flight which is chosen.

5. Post conditions: There are no post conditions.

6. Extension points: There are no extension points.

Use case specification for the cancel flight:

1. Use case name: Cancel Flight.

2. This use case id started by traveler to cancel his or her reservation.

3. Flow of events:

3.1 Basic flow: This use case is started by the traveler if he has some problems with travelling. To cancel the reservation the system asks the traveler his reservation number and confirmation. Else alternate flow 2.2.1 is executed. After the conformation of traveler the system concedes the reservation and update databases.

3.2 Alternate flow:

3.2.1 If the reservation number is in valid the message is displayed in valid number.

4. Special conditions: There are no special conditions.

5. Pre-conditions: User must have the reservation with that number.

6. Post conditions: There are no post conditions.

7. Extension points: There are no extension points.

4.3.3 Input/output Specifications.

Input /Output Format.

Every program has an input as well as output data. These are used mainly to achieve the specific objectives of verifying the processing operation being performed.

The input format is used essentially to state the data elements requested by the user of the program. Below are forms that serve as the input format

Proto type for login system

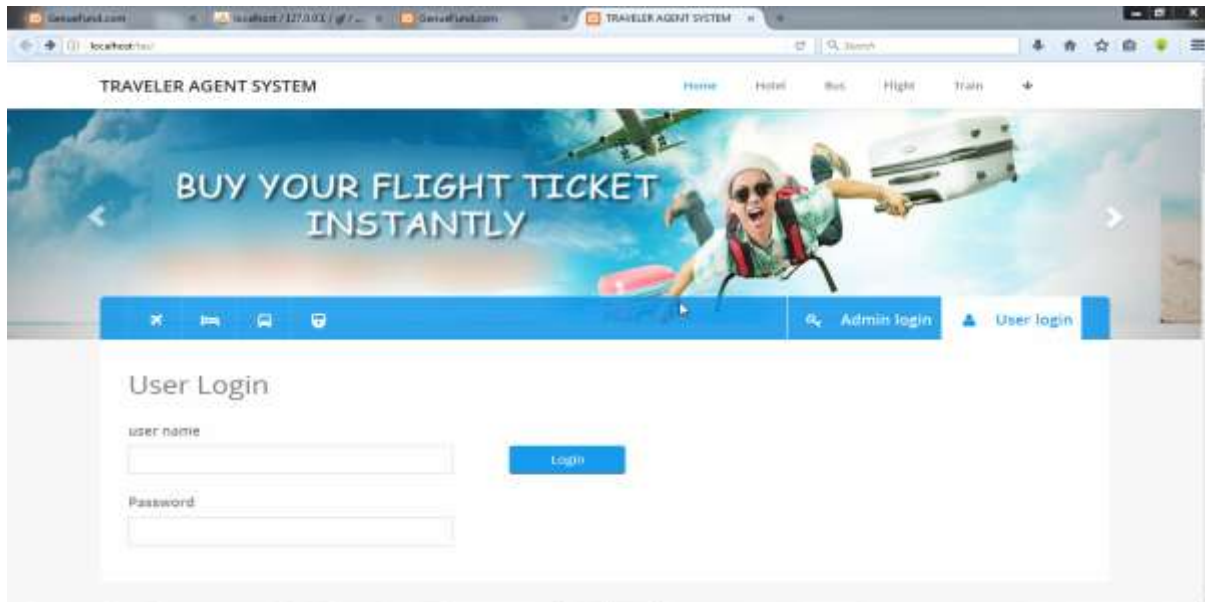
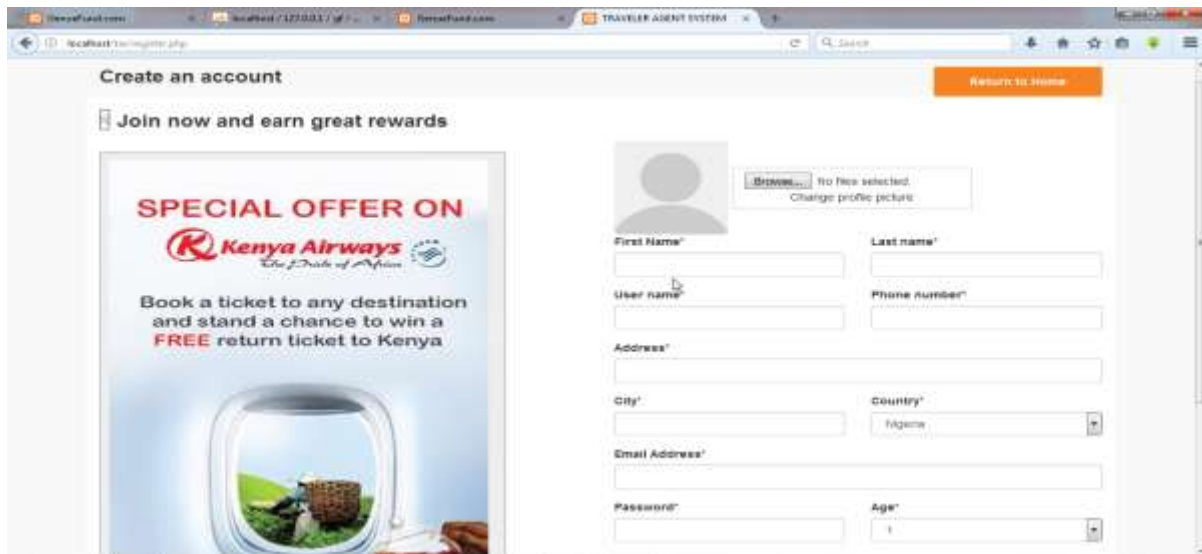


Figure 4.5 login system

This is a platform in which the user is expected to login by entering the username and password to enable him/her perform any of the following operations:

- Search for a flight
- Search for a hotel
- Search for a car to hire
- Book for flight
- Book for a hotel
- Book for a car
- Cancel a flight or hotel or car reservation.

Proto type for registration



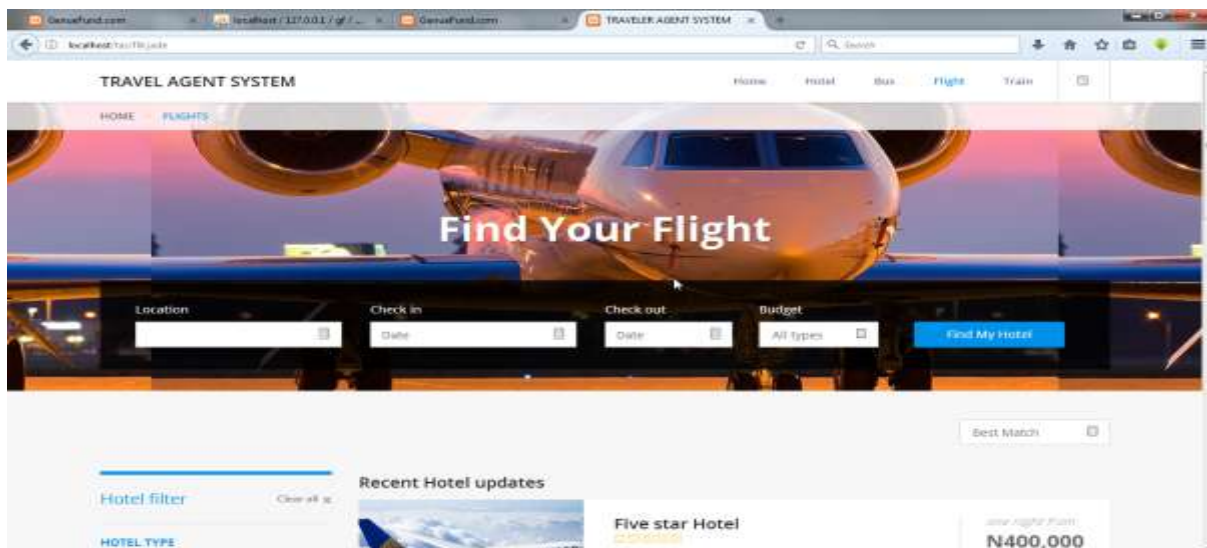
The screenshot displays a web browser window with the URL 'localhost/127.0.0.1/'. The page title is 'TRAVEL AGENT SYSTEM'. The main heading is 'Create an account', with a sub-heading 'Join now and earn great rewards'. On the left, there is a promotional banner for Kenya Airways with the text: 'SPECIAL OFFER ON Kenya Airways The Spirit of Africa. Book a ticket to any destination and stand a chance to win a FREE return ticket to Kenya'. On the right, there is a registration form with the following fields: 'First Name*', 'Last name*', 'User name*', 'Phone number*', 'Address*', 'City*', 'Country*' (with a dropdown menu showing 'Nigeria'), 'Email Address*', 'Password*', and 'Age*' (with a dropdown menu showing '1'). A 'Return to Home' button is located in the top right corner. A profile picture placeholder is shown with the text 'Browse... No file selected. Change profile picture'.

Figure 4.6: Registration

Registration platform is a platform where new user are expected to register by entering vital data about him or and upload his/her passport photograph.

In addition the new user is expected to create a username and password which will give him/her access to select, book or cancel flight, hotels and car.

Prototype for searching flight



The screenshot displays a web browser window with the URL 'localhost/127.0.0.1/'. The page title is 'TRAVEL AGENT SYSTEM'. The main heading is 'Find Your Flight'. The search form includes fields for 'Location', 'Check in' (Date), 'Check out' (Date), and 'Budget' (All types). A 'Find My Hotel' button is visible. Below the search form, there is a 'Hotel filter' section and a 'Recent Hotel updates' section showing a 'Five star Hotel' with a price of 'N400,000'. A 'Best Match' dropdown menu is also present.

Figure 4.7: Searching flight

This is a platform where the user can search for the cheapest flight for the area he/she entail to travel to. This is done by entering the Location where you are going to; the date you want to travel and the amount you budgeted for the flight and click find flight.

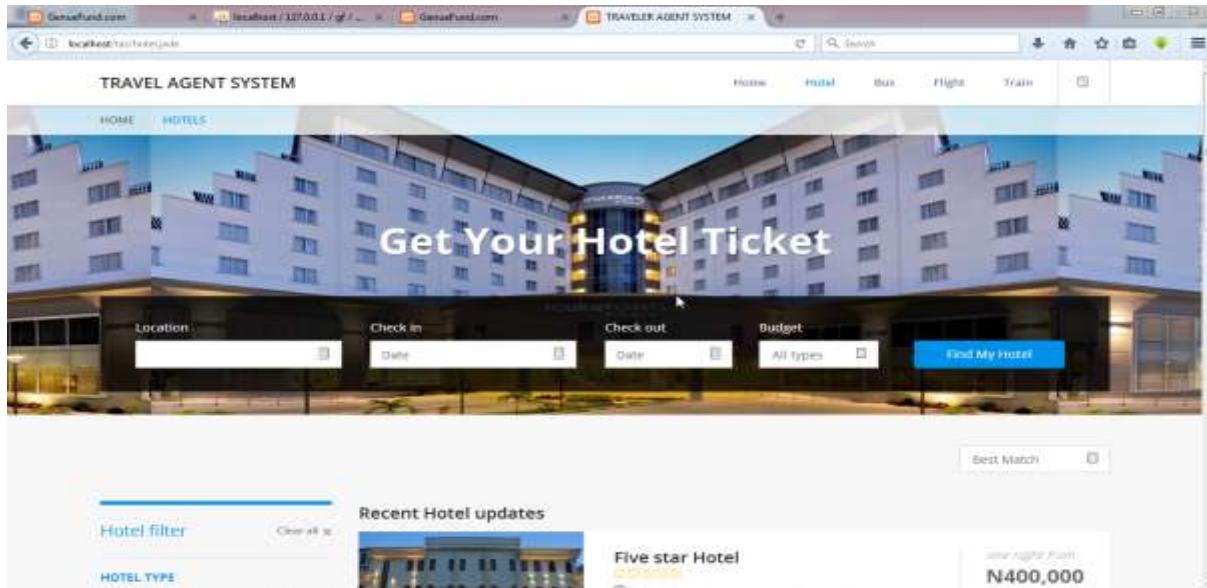


Figure 4.8: Hotel Reservation

This is a platform where the user can search for the cheapest hotel for the area he/she entail to travel to. This is done by entering the Location where you are going to; the date you want to check in, check out and the amount you budgeted for the hotel accommodation and click find hotel.

Prototype for cancel ticket

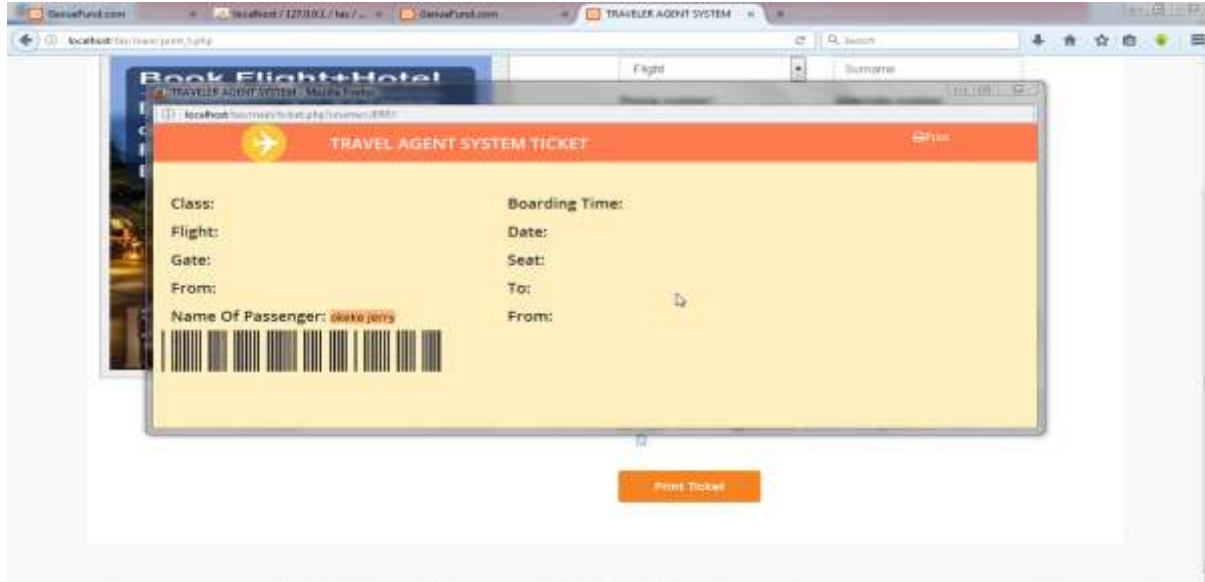


Figure 4.9: Cancel ticket

The traveler if he has some problems with travelling, he/she can cancel the reservation. To cancel the reservation the system asks the traveler his reservation number and confirmation. After the conformation of traveler the system cancels the reservation and update databases.

Output Form.

Prototype for flight details:

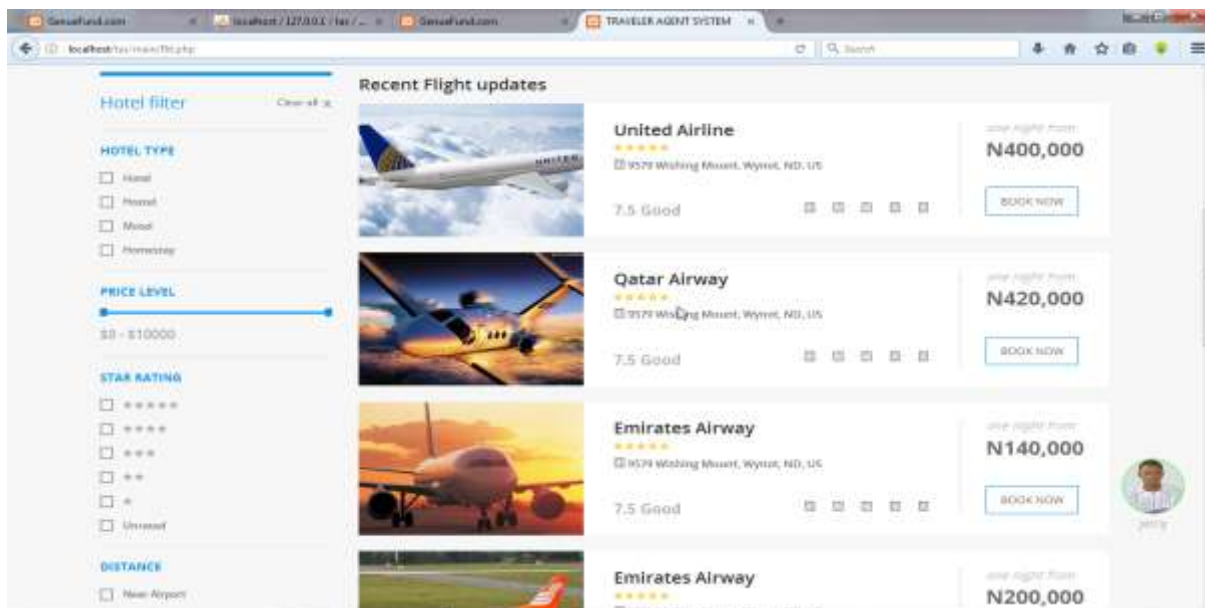


Figure 4.10: Flight Details

Components of the Design.

Activity Diagram

An Activity diagram is a variation of a special case of a state machine, in which the states of activities representing the performance of operations and the transitions are triggered by the completion of the operations. The purpose of Activity diagram is to provide a view of flows and what is going on inside a use case or among several classes. We can also use activity diagrams to model code-specific information such as a class operation. Activity diagrams are very similar to a flowchart because you can model a workflow from activity to activity.

An activity diagram is basically a special case of a state machine in which most of the states are activities and most of the transitions are implicitly triggered by completion of the actions in the source activities.

- Activity diagrams represent the dynamics of the system.
- They are flow charts that are used to show the workflow of a system; that is, they show the flow of control from activity to activity in the system, what activities can be done in parallel, and any alternate paths through the flow.
- At this point in the life cycle, activity diagrams may be created to represent the flow across use cases or they may be created to represent the flow within a particular use case.
- Later in the life cycle, activity diagrams may be created to show the workflow for an operation.

MAIN ACTIVITY BUSINESS DIAGRAM:

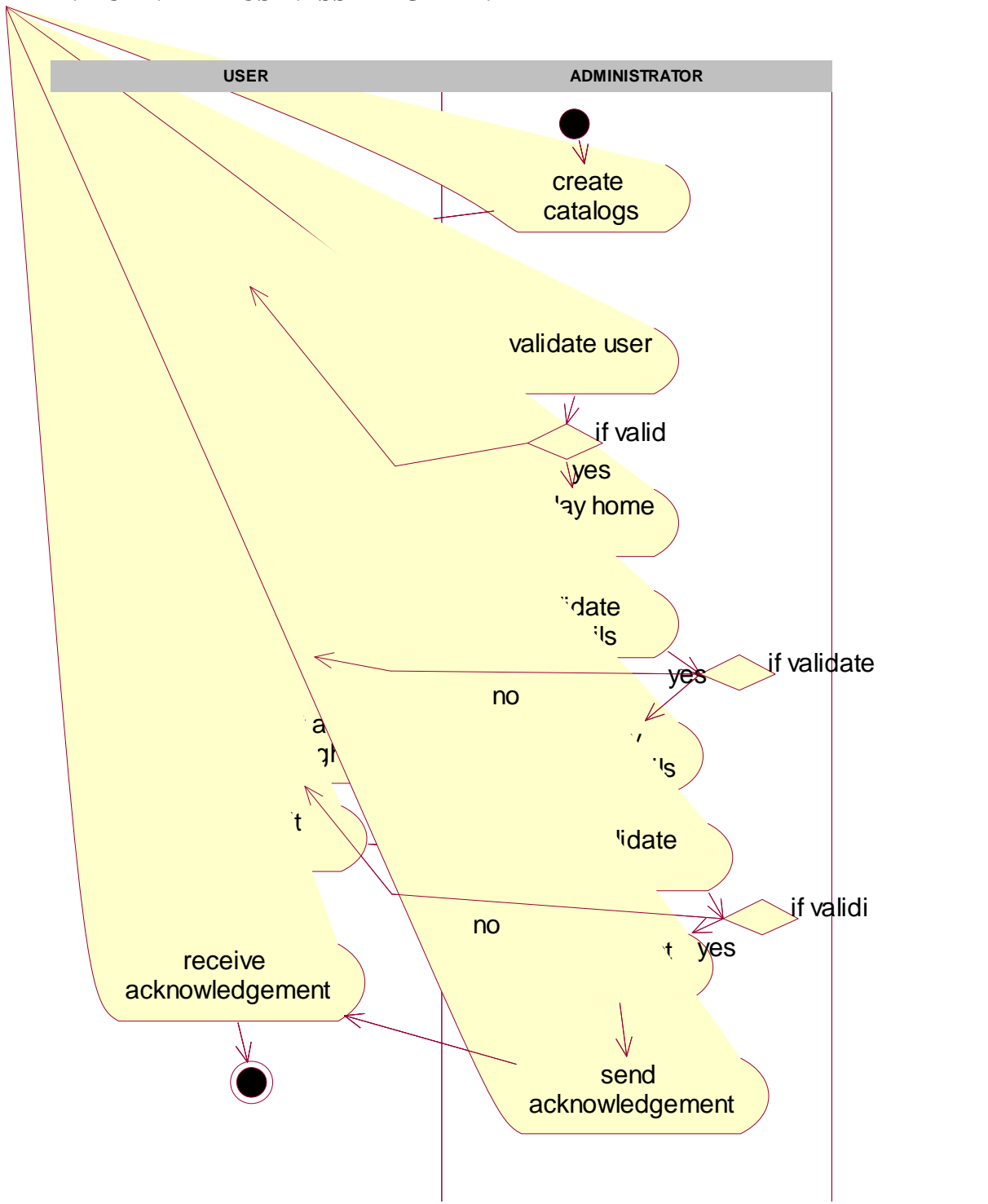


Figure 4.11: Main Activity Business Diagram

Activity Diagram for Log in

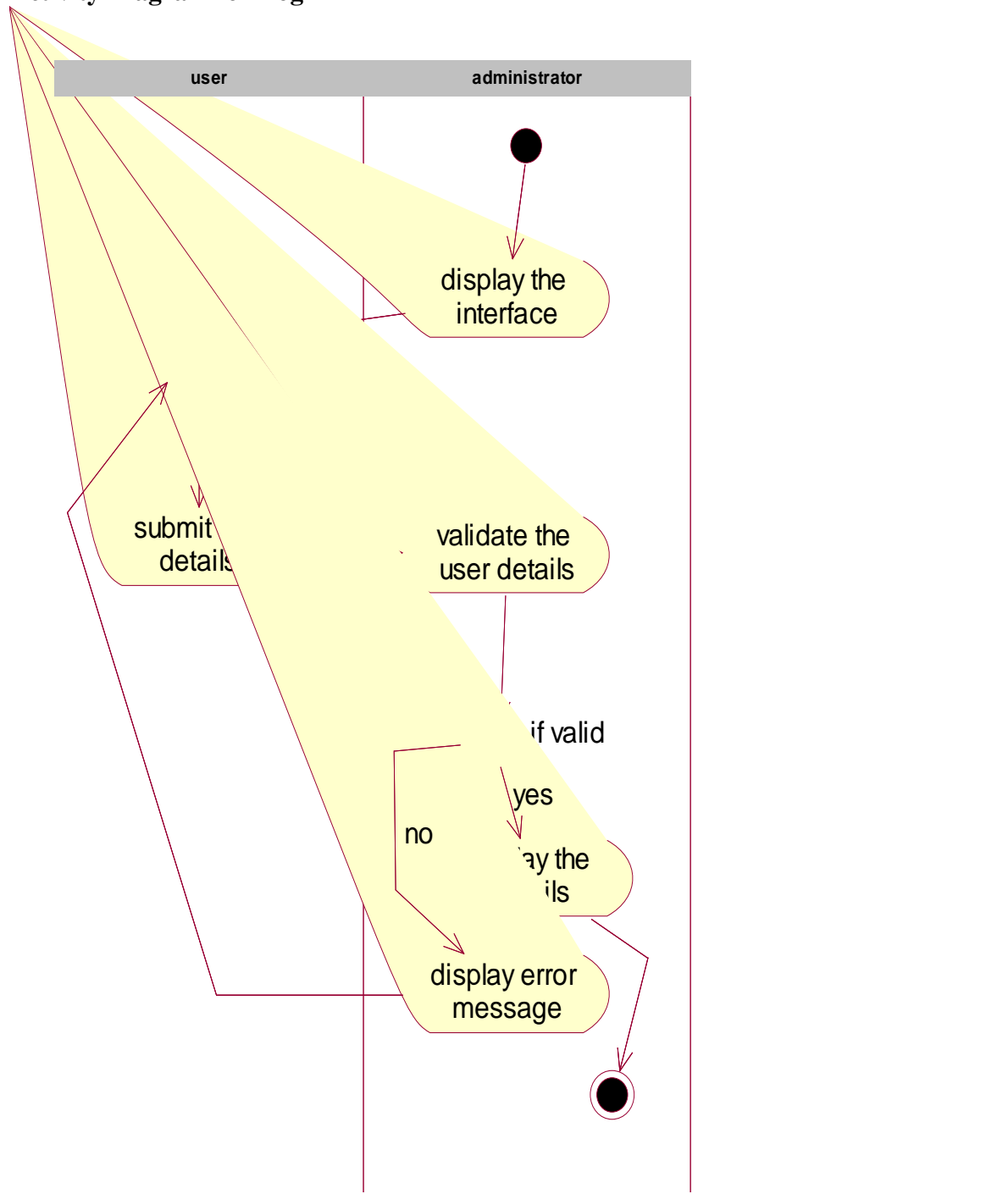


Figure 4.12: Activity Diagram for Login

Activity Diagram for Selecting the Flight

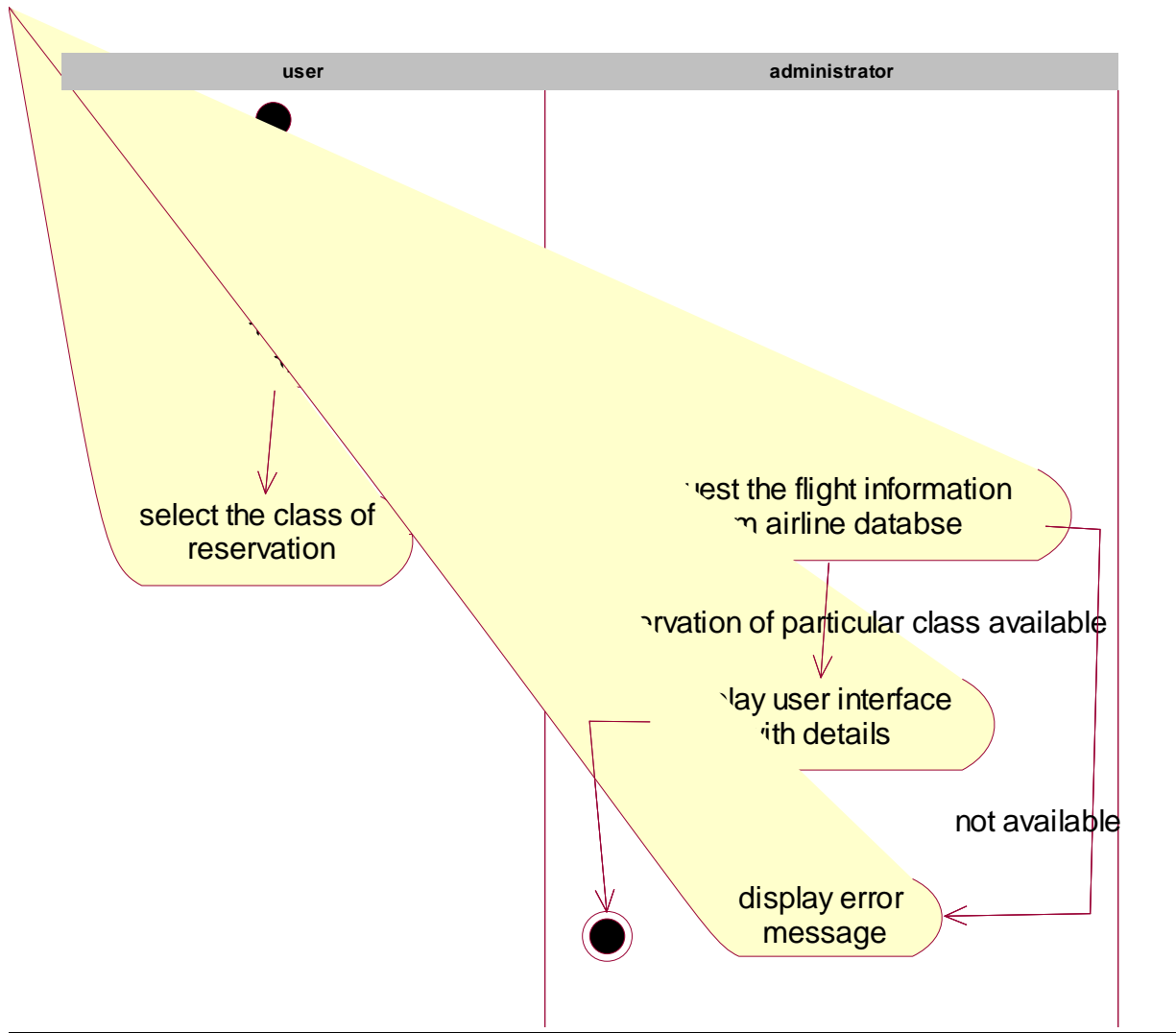


Figure 4.13: Activity Diagram for Selecting the Flight

Activity Diagram for Booking Ticket:

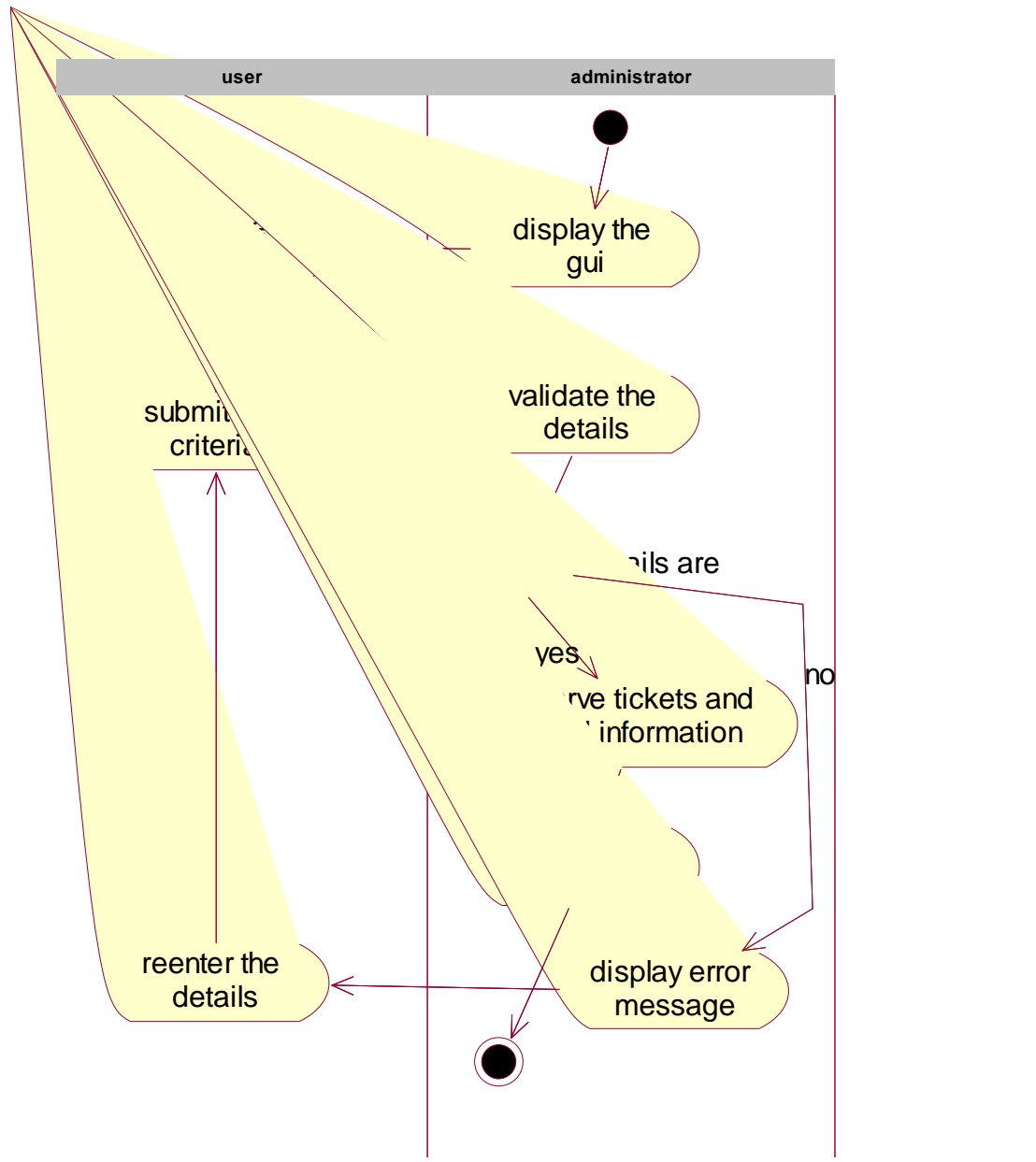


Figure 4.14: Activity Diagram for Booking Flight

ACTIVITY DIAGRAM FOR CANCEL FLIGHT

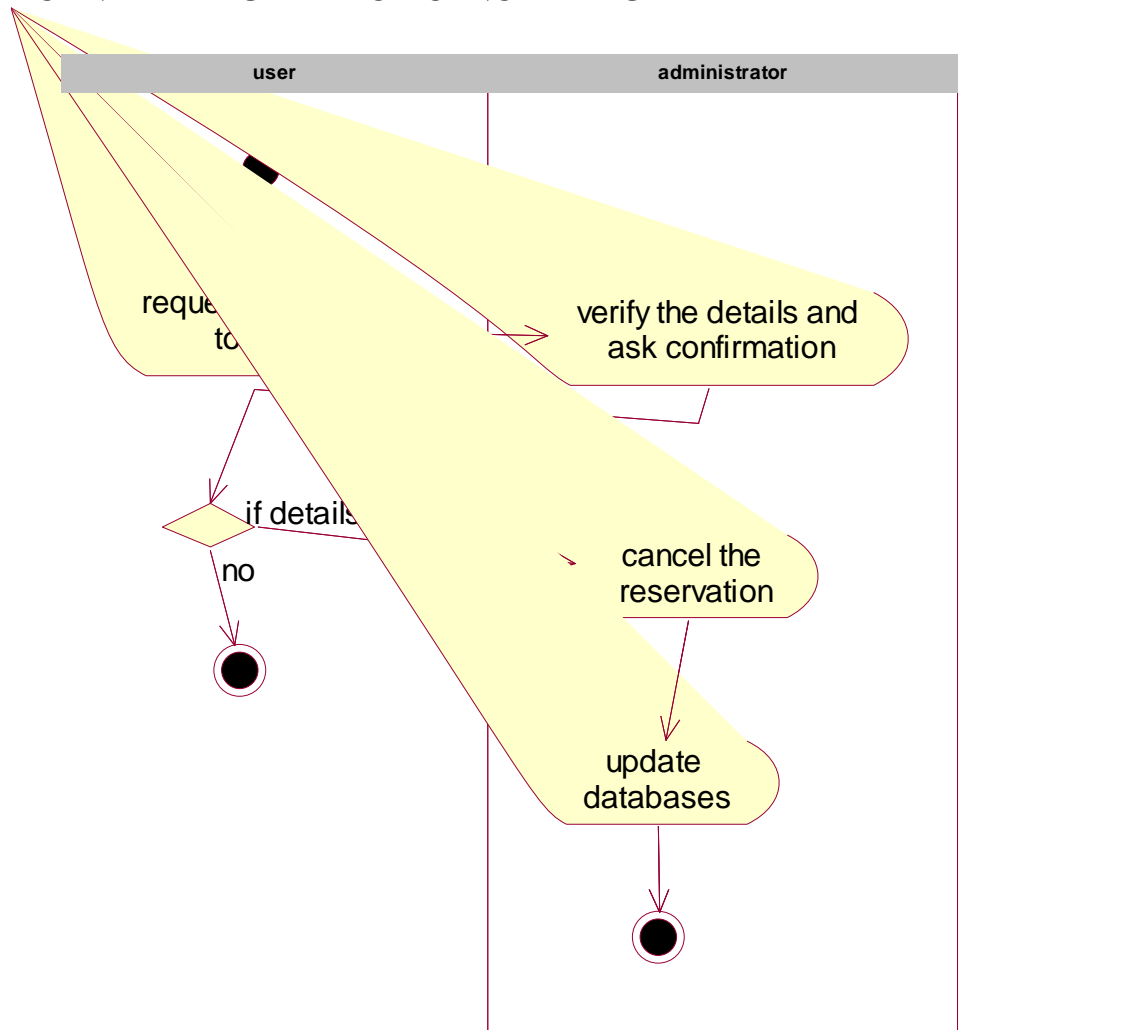


Figure 4.15: Activity Diagram for Cancel Flight

4.3.4 Application Algorithm

Use Case

At the starting, for the identification of classes we need to concentrate completely on uses cases. A further examination of the use cases also helps in identifying operations and the messages that classes need to exchange. However, it is easy to think first in terms of the overall responsibilities of a class rather than its individual operations. A responsibility is a high level description of something a class can do. It reflects the knowledge or information that is available to that class, either stored within its own attributes or requested via collaboration with other classes, and also the services that it can offer to other objects. A responsibility may correspond to one or more operations. It is difficult to determine the appropriate

responsibilities for each class as there may be many alternatives that all appear to be equally justified. Class Responsibility Collaboration (CRC) cards provide an effective technique for exploring the possible ways of allocating responsibilities to classes and the collaborations that are necessary to fulfill the responsibilities.

CRC cards can be used at several different stages of a project for different purposes.

1. They can be used early in a project to help the production of an initial class diagram.
2. To develop a shared understanding of user requirements among the members of the team.
3. CRCs are helpful in modeling object interaction. The format of a typical CRC card is shown table 4.8

Table 4.8 CRC

Class Name:	
Responsibilities	Collaborations
<i>Responsibilities of a class are listed in this section</i>	<i>Collaborations with other classes are listed here, together with a brief description of the purpose of the collaboration</i>

CRC cards are an aid to a group role-playing activity. Index cards are used in preference to pieces of paper due to their robustness and to the limitations that their size (approx. 15cm x 8cm) imposes on the number of responsibilities and collaborations that can be effectively allocated to each class. A class name is entered at the top of each card and responsibilities and collaborations are listed underneath as they become apparent. For the sake of clarity, each collaboration is normally listed next to the corresponding responsibility.

From a UML perspective, use of CRC cards is in analyzing the object interaction that is triggered by a particular use case scenario. The process of using CRC cards is usually structured as follows. Conduct a session to identify which objects are involved in the use case.

Allocate each object to a team member who will play the role of that object.

Act out the use case: This involves a series of negotiations among the objects to explore how responsibility can be allocated and to identify how the objects can collaborate with each other.

Identify and record any missing or redundant objects.

Before beginning a CRC session it is important that all team members are briefed on the organization of the session and a CRC session should be preceded by a separate exercise those identities all the classes for that part of the application to be analyzed.

The team members to whom these classes are allocated can then prepare for the role playing exercise by considering in advance a first-cut allocation of responsibilities and identification of collaborations. Here, it is important to ensure that the environment in which the sessions take place is free from interruptions and free for the flow of ideas among team members.

During a CRC card session, there must be an explicit strategy that helps to achieve an appropriate distribution of responsibilities among the classes.

One simple but effective approach is to apply the rule that each object should be as lazy as possible, refusing to take on any additional responsibility unless instructed to do so by its fellow objects. During a session conducted according to this rule, each role player identifies the object that they feel is the most appropriate to take on each responsibility, and attempts to persuade that object to accept the responsibility. For each responsibility that must be allocated, one object is eventually persuaded by the weight of rational argument to accept it. This process can help to highlight missing objects that are not explicitly referred to by the use case description. When responsibilities can be allocated in several different ways it is useful to role-play each allocation separately to determine which is the most appropriate. The aim normally is to minimize the number of messages that must be passed and their complexity, while also producing class definitions that are cohesive and well-focused.

Consider CRC exercise for the use case allotting buses to routes. This use case involves instances of Bus, and Depot.

The resulting CRC cards are shown in the tables 4.9 and 4.10

Table 4.9: CRC card for Traveler class in search flight.

Class Name : traveler	
Responsibilities	Collaborations
<i>Provide traveling details</i>	<i>Search flight UI provide flights available.</i>

Table 4.10: CRC card for Searching system in Search Flight

Class Name : Searching system	
Responsibilities	Collaborations
Search for the flight available.	Airline database provides the list of available flights

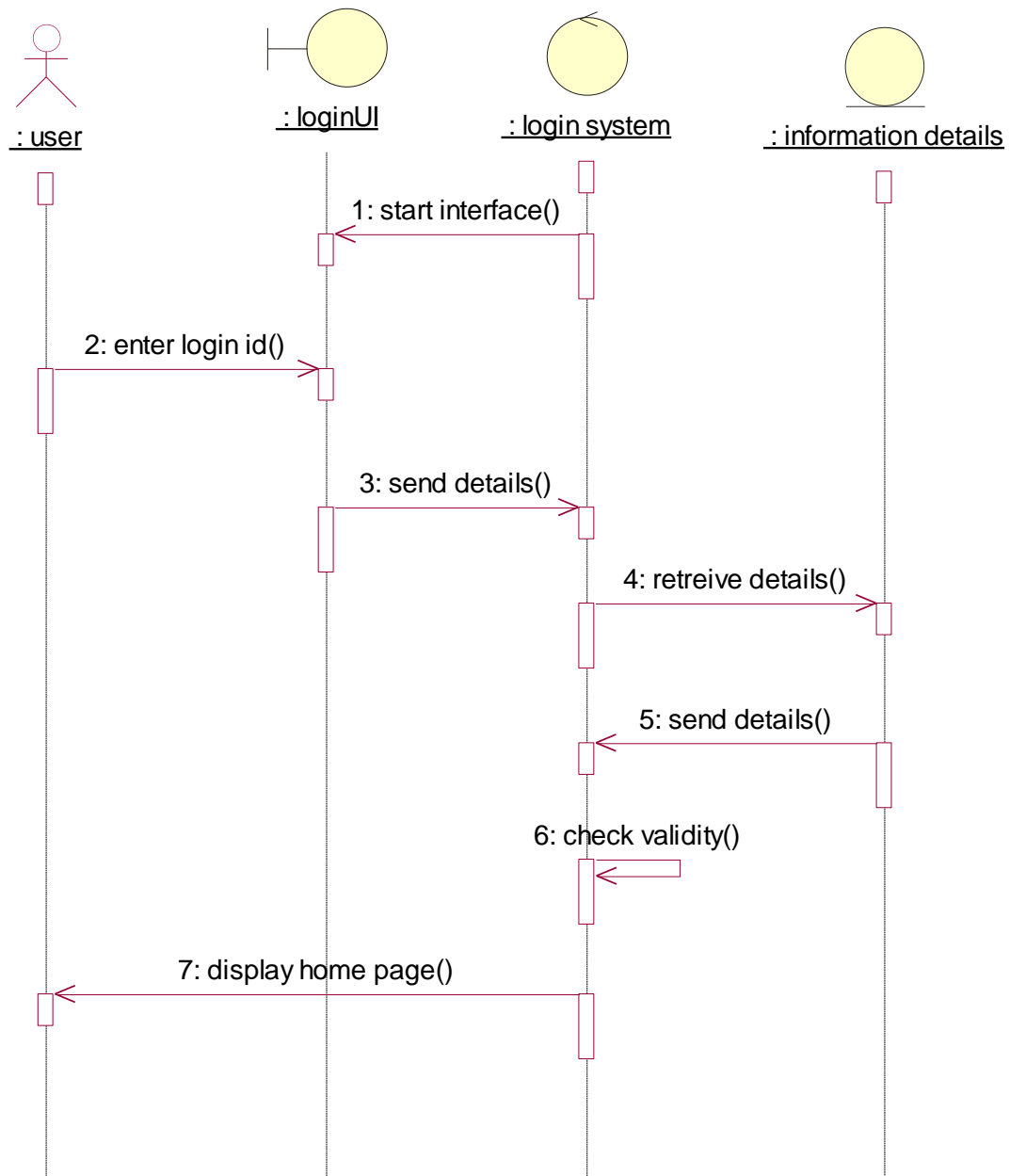


Figure 4.16: Sequence diagram for login system

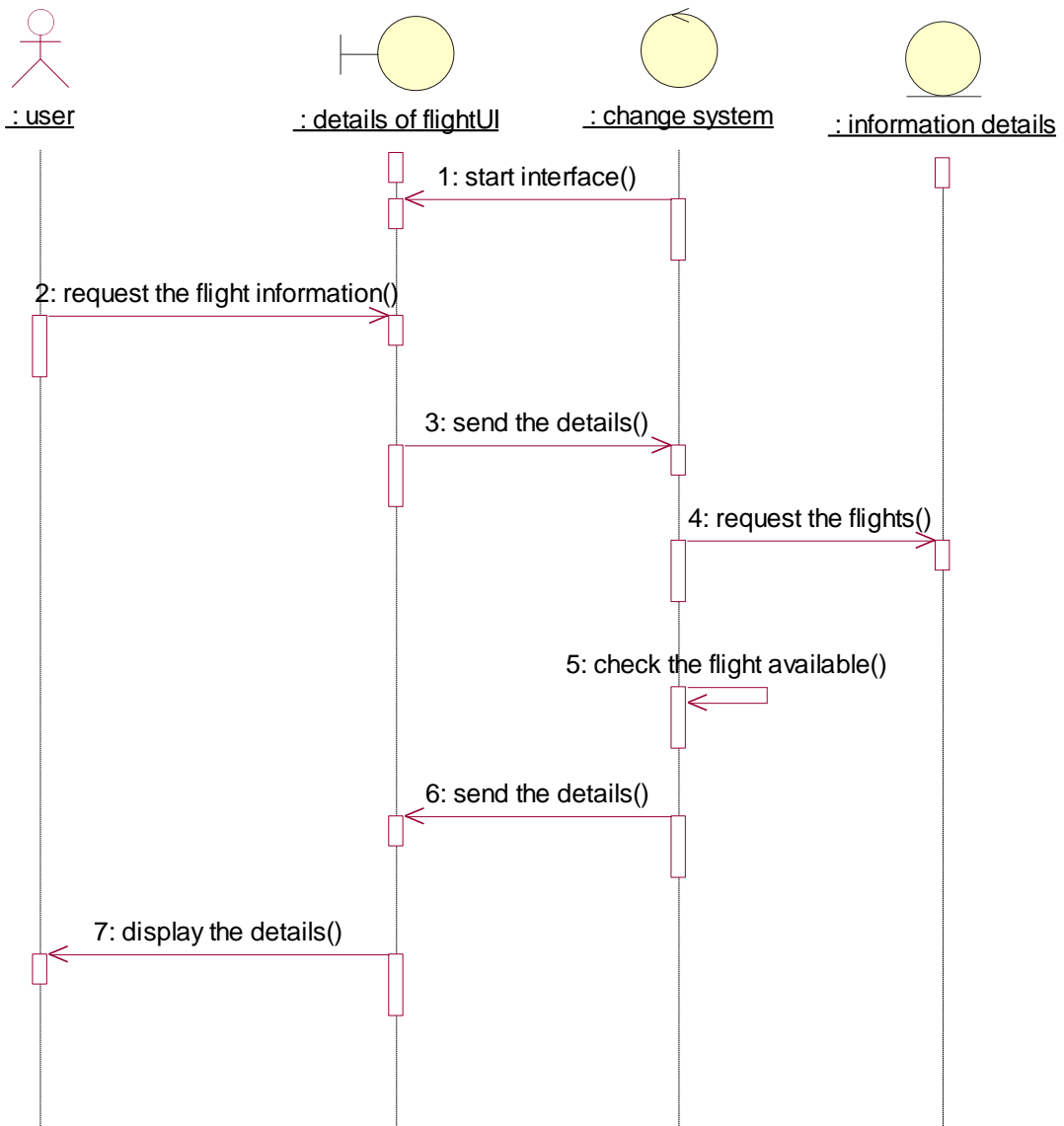


Figure 4.17: Sequence diagram for searching flights

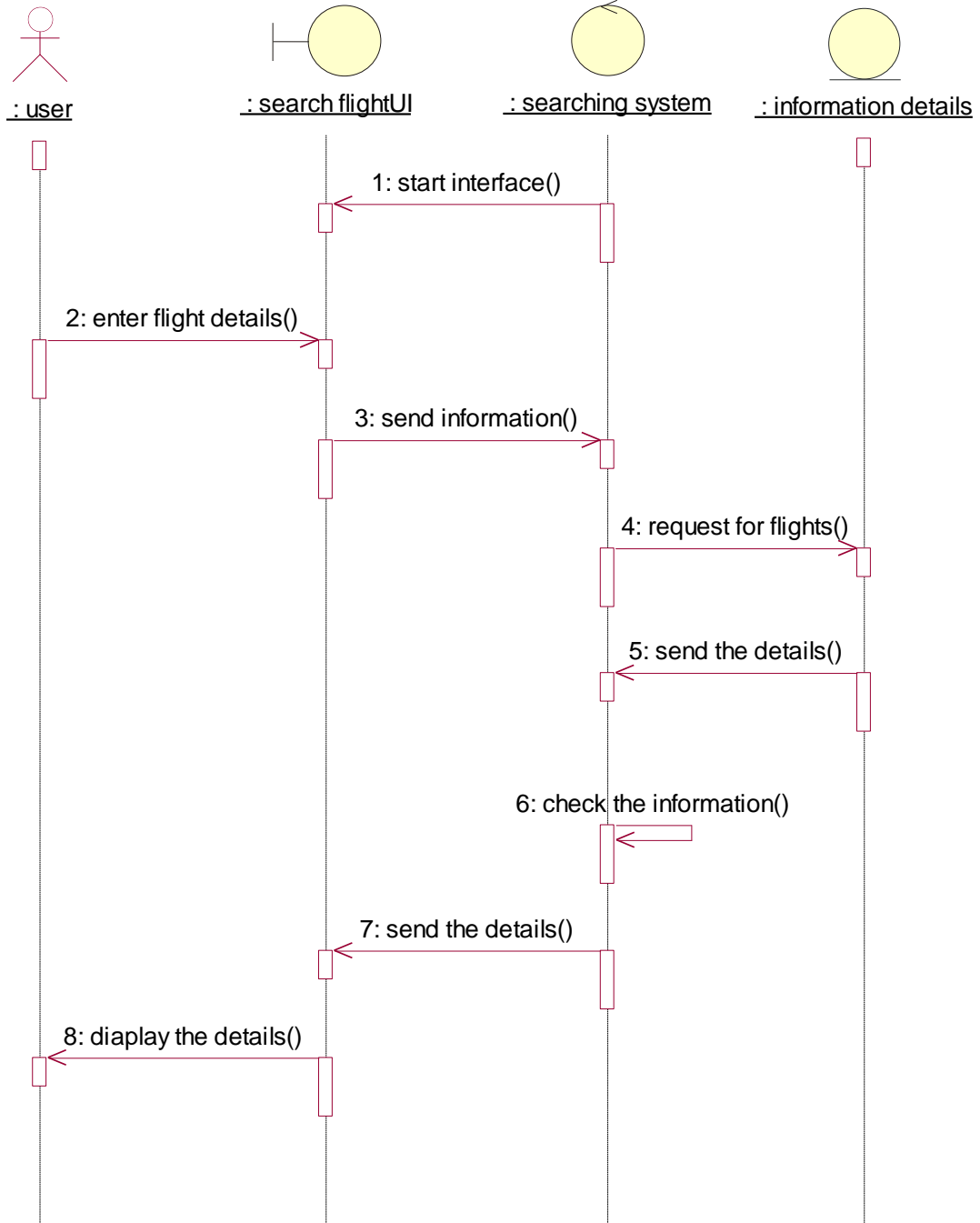


Figure 4.18: Sequence diagram for selecting flight

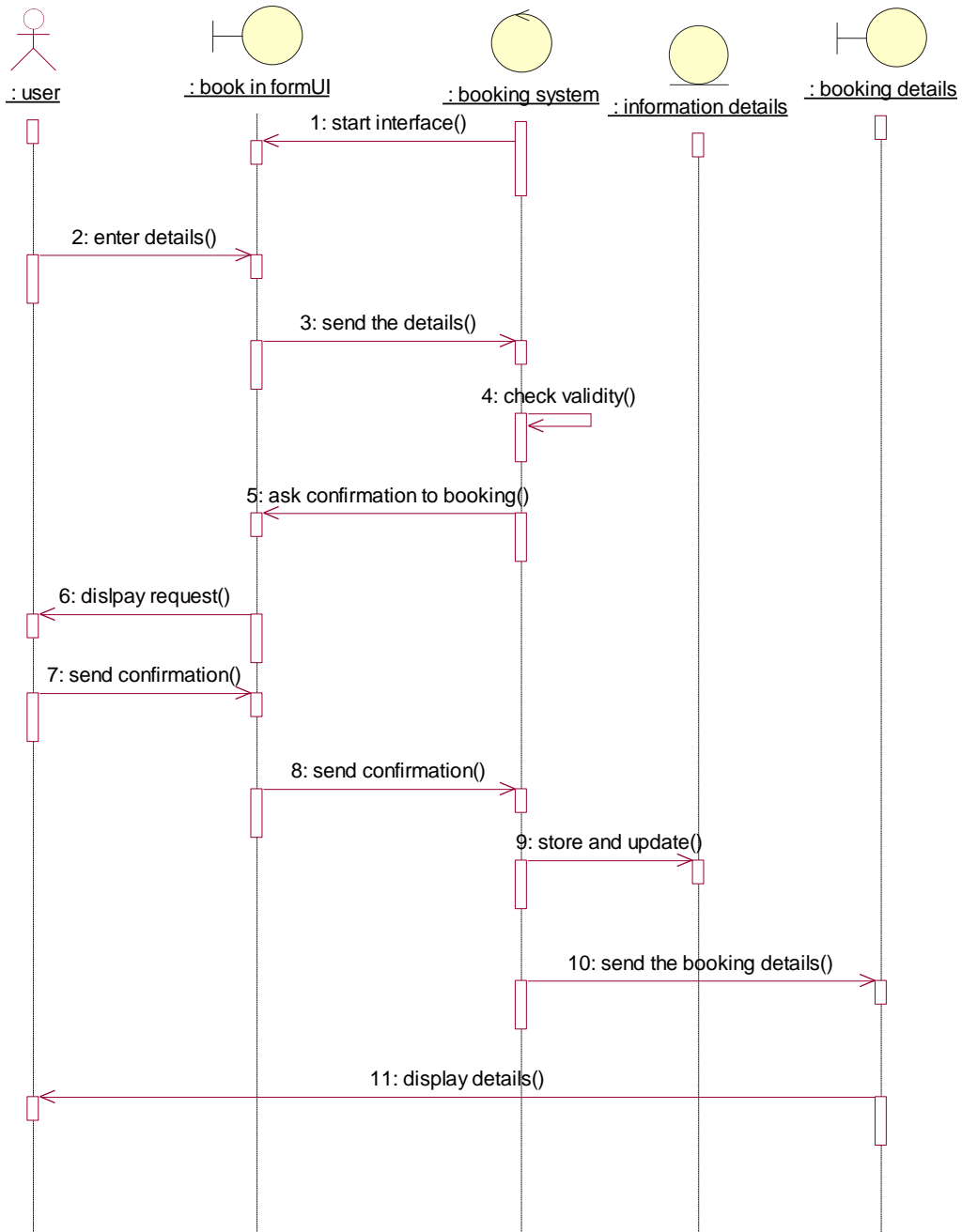


Figure 4.19: Sequence diagram for booking flight

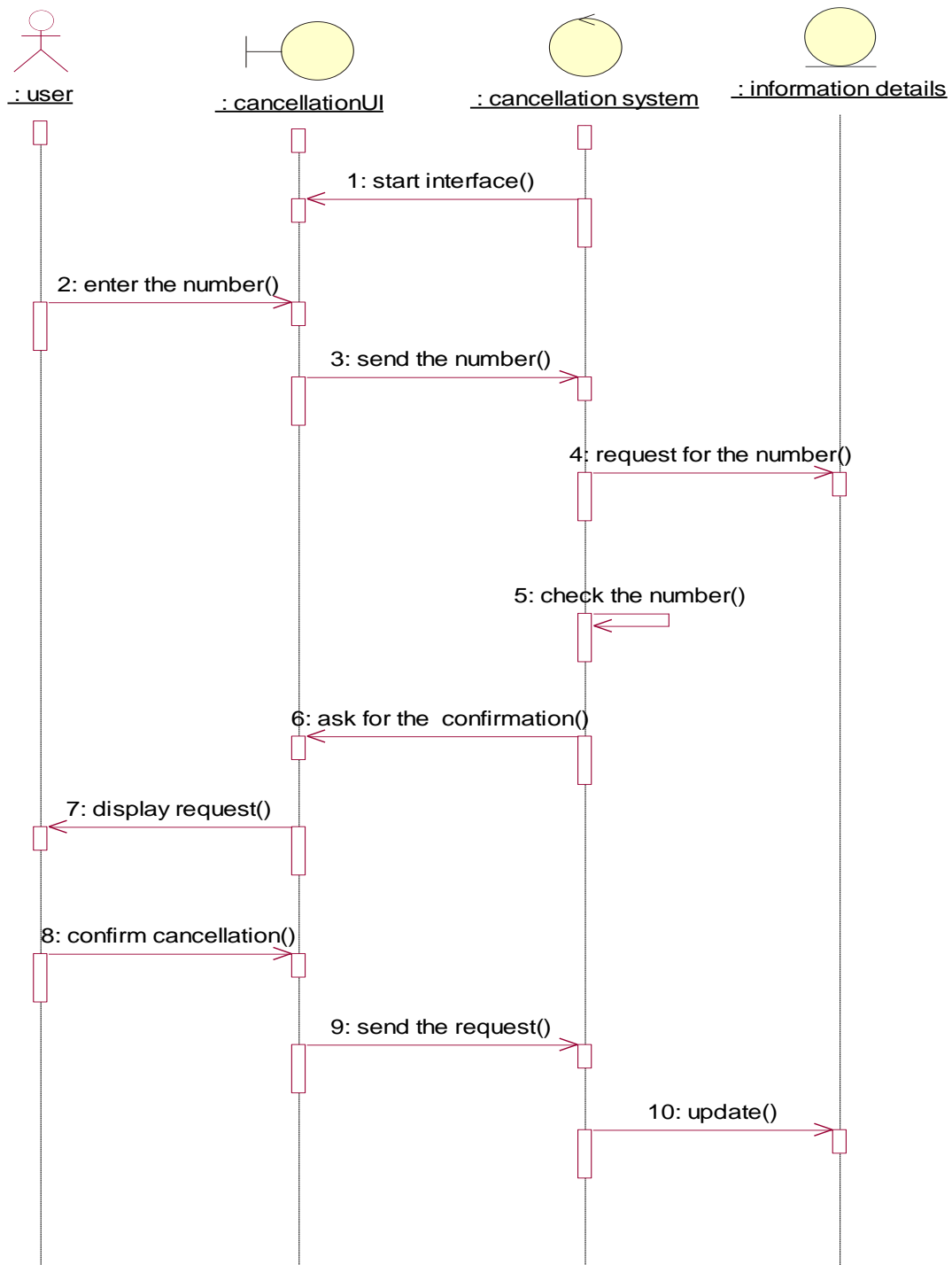


Figure 4.20: Sequence diagram for cancelling flight

Collaboration Diagram

A collaboration diagram is an alternate way to show a scenario. This type of

:

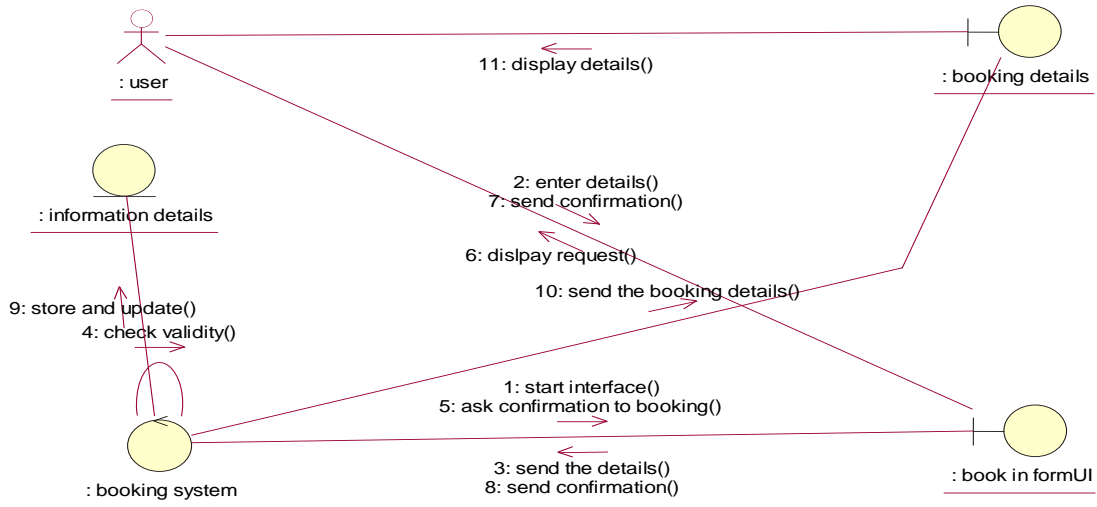


Figure 4.21: Collaboration diagram for booking flight

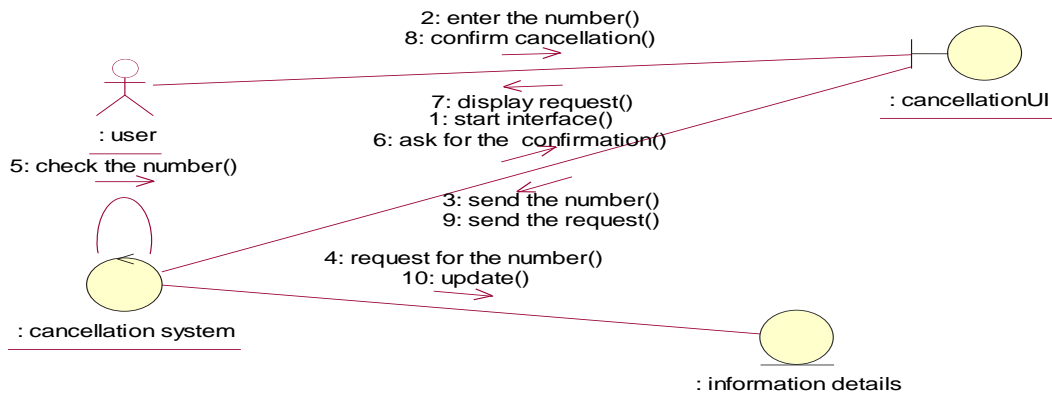


Figure 4.22: Collaboration diagram for cancelling flight

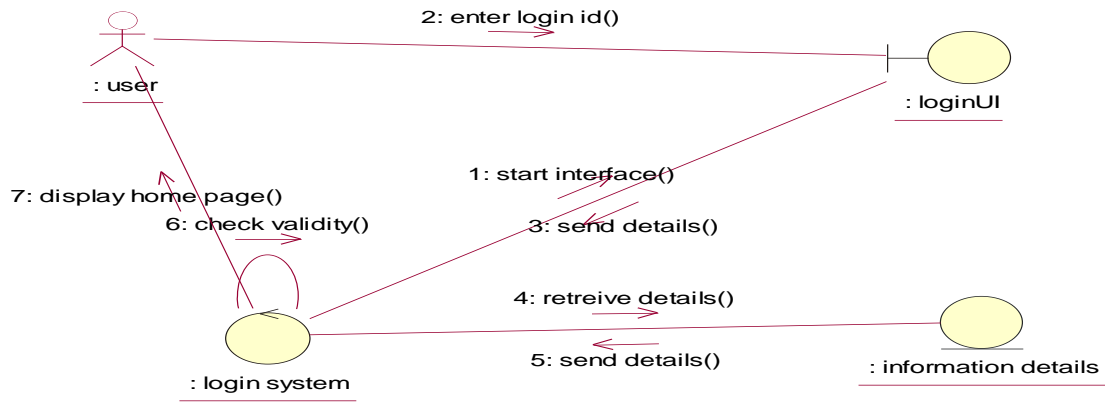


Figure 4.23: Collaboration diagram for login

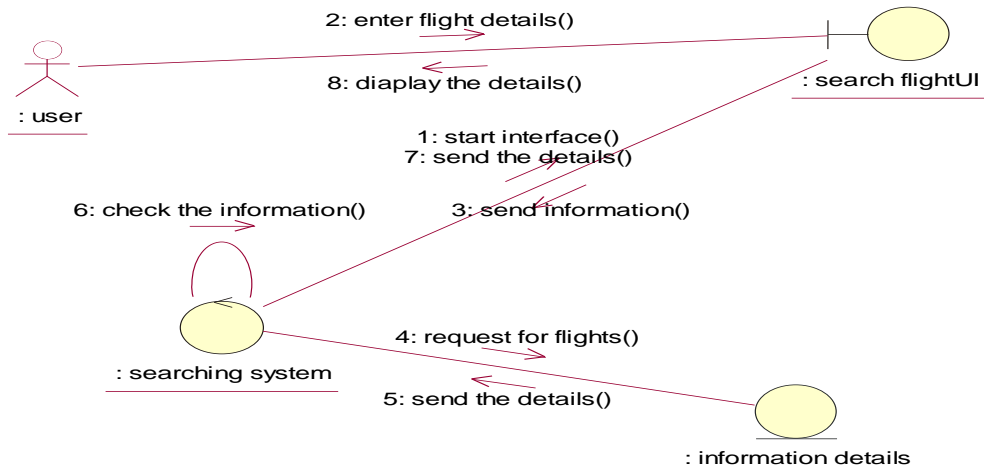


Figure 4.24: Collaboration diagram for searching flight

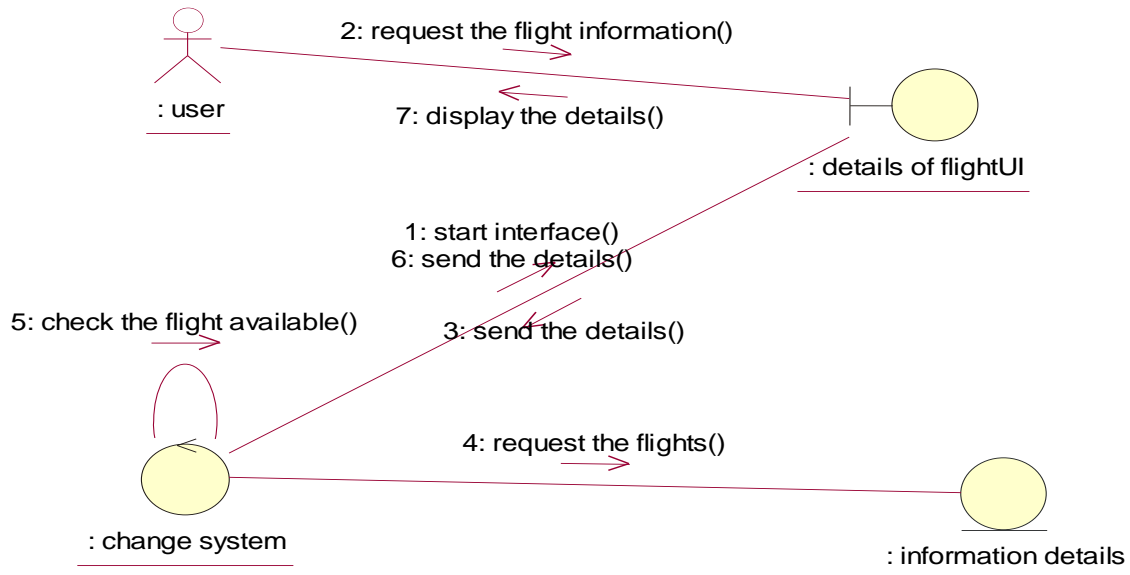


Figure 4.25: Collaboration diagram for booking flight

4.4 Proposed System Requirement

4.4.1 Hardware Requirement

Hardware is the physical equipment or component that makes up a computer system. The selection of hardware is very important in the existence and proper working of this software.

Hardware Requirement.

The Case study designed to test the proposed System is developed on:

- + Processor
- + RAM
- + Hard Disk Drive
- + Key Board
- + Monitor
- + Display Adapter
- + Network Adapter
- + Mouse
- + Flash Drive
- + Uninterruptible Power Supply (UPS)

- ✚ Stabilizer Power Regulator
- ✚ Printer (LaserJet)
- ✚ Surge Protector

4.4.2 Software Requirement.

Software is a program used by Computers to facilitate their operation and utilization. The Software is expected to run under the Microsoft windows' environment. Windows has a variety of tools that make program running under it user friendly. It gives the Computer Capability of doing whatever the user wants. Software can be of two types: namely

- ✚ System Software
- ✚ Application Software.

System Software are programs written by system programmer or manufacturer of Computer systems to interpret instructions contained in the application software and provide instructions to the central processor, so that the various hardware units that make up the computer can function as intended.

Application Software is also known as user programs developed to solve some general problems in a particular field.

4.5 Program Development.

Software development deals with various tools, methods and procedures required for controlling the complexity of software development, project management and its maintenance.

Intelligent Agent Software systems pass through two principal phases during their lifecycle.

- The development phase.
- The operation and maintenance.

Intelligent Agent Software development passes through various phases. They include

Program definition: The first stage in the development process understands the problem in question and its requirements. Requirements include the context in which the problem arises, functionality expected from the system and system constraints.

Analysis: Analysis phase delivers requirement specification. The system specification serves as an interface between the design and the implementer as well as between the implementer and the user.

Coding or Implementation: Once the specification and design of the software is over, the choice of the programming language remains as one of the most critical aspect in producing reliable software.

The other objectives of this system can be summarized as follows:

- Design of a hierarchical framework in terms of positions held thus depicting the organizational hierarchy. Update of the structure of the same, as well as addition of new elements.
- Search for all reservations, bookings, relevant information etc. possible. Also department-wise, level-wise and other parameter based search enabled.
- Communication between Intelligent System and administrator.
- Computerized Reservation generation, manipulation and management.
- Easy management of databases of various sections covering key aspects.

4.6 System Testing

Intelligent Agent program testing and Implementation Diagrams

Component Diagram

Two type's implementation diagrams in UML terminology are

1. Component diagrams
2. Deployment diagrams

In a large project there will be many files that make up the system. These files will have dependencies on one another. The nature of these dependencies will depend on the language or languages used for the development and may exist at compile-time, at link-time or at run-time. There are also dependencies between source code files and the executable files or byte code

files that are derived from them by compilation. Component diagrams are one of the two types of implementation diagram in UML. Component diagrams show these dependencies between software components in the system. Stereotypes can be used to show dependencies that are specific to particular languages also.

A component diagram shows the allocation of classes and objects to components in the physical design of a system. A components diagram may represent all or part of the component architecture of a system along with dependency relationships.

The dependency relationship indicates that one entity in a components diagram uses the services or facilities of another.

- Dependencies in the component diagram represent compilation dependencies.
- The dependency relationship may also be used to show calling dependencies among components, using dependency arrows from components to interfaces on other components.

Different authors use component diagrams in different ways

Here we have the following distinction between them:

- Components in a component diagram should be the physical components of a system.
- During analysis and the early stages of design, package can be used to show the logical grouping of class diagrams or of models that use other kinds of diagrams into packages relating to sub-systems.
- During implementation, package diagrams can be used to show the grouping of physical components into sub-systems.

If component diagrams are used, it is better to keep separate sets of diagrams to show compile-time and run-time dependencies, however, this is likely to result in a large number of diagrams. Component diagrams show the components as types. We wish to show instances of the system components and deployment diagram.

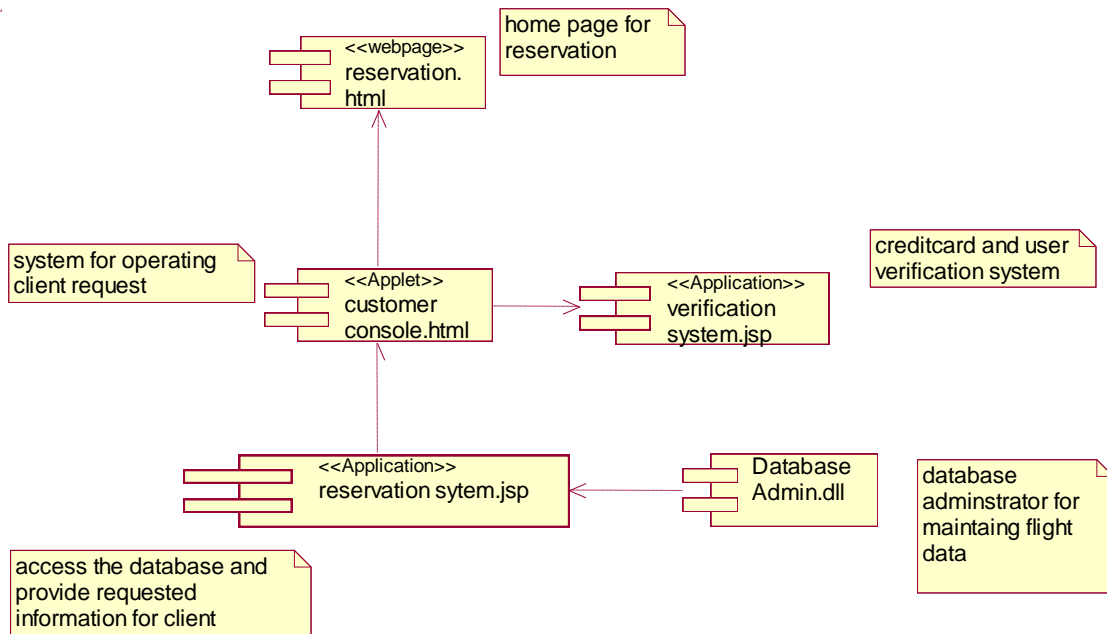


Figure 4.26 Component diagram for AIRLINE RESERVATION MACHINE (Frank, 2012)

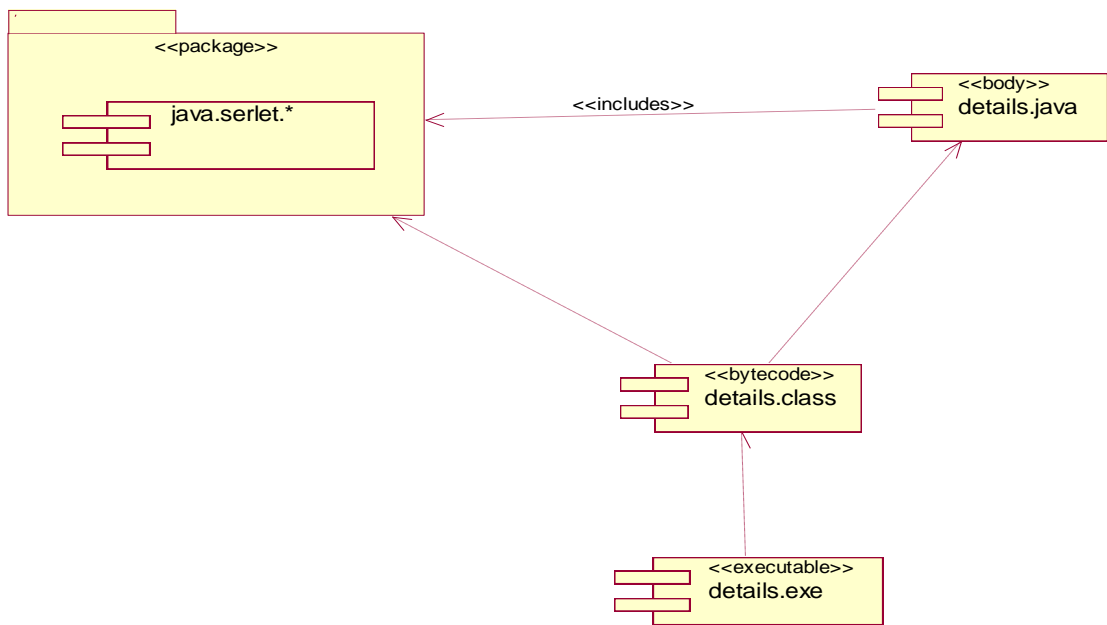


Figure 4.27: Component Diagram for ARS Details (Frank, 2012)

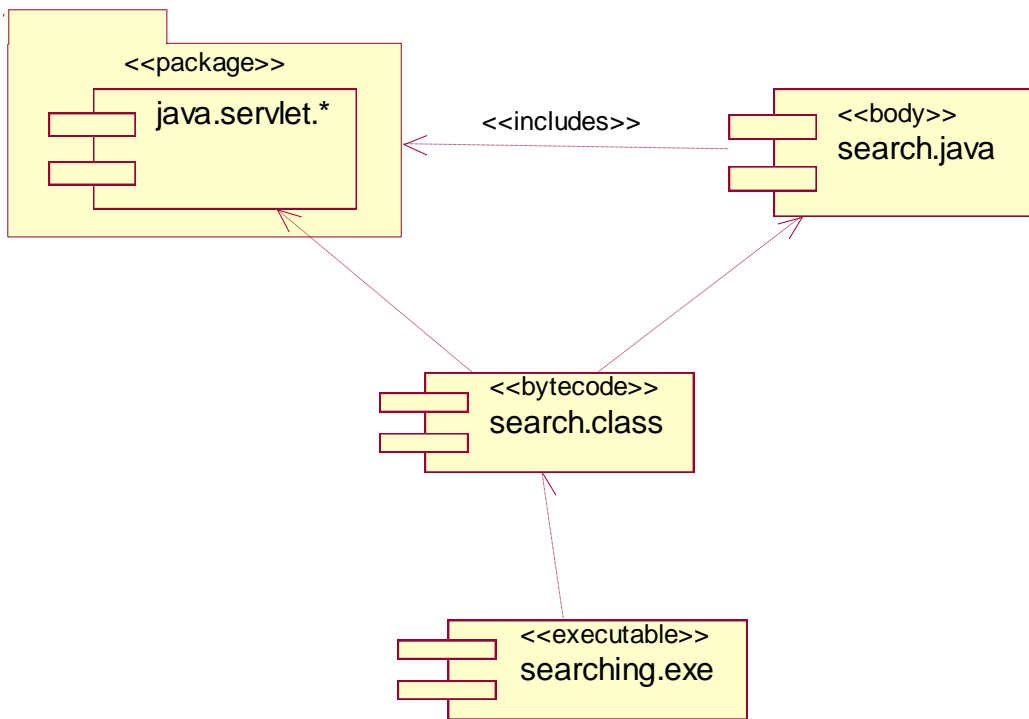


Figure 4.28: Component Diagram for Searching Flights (Frank, 2012)

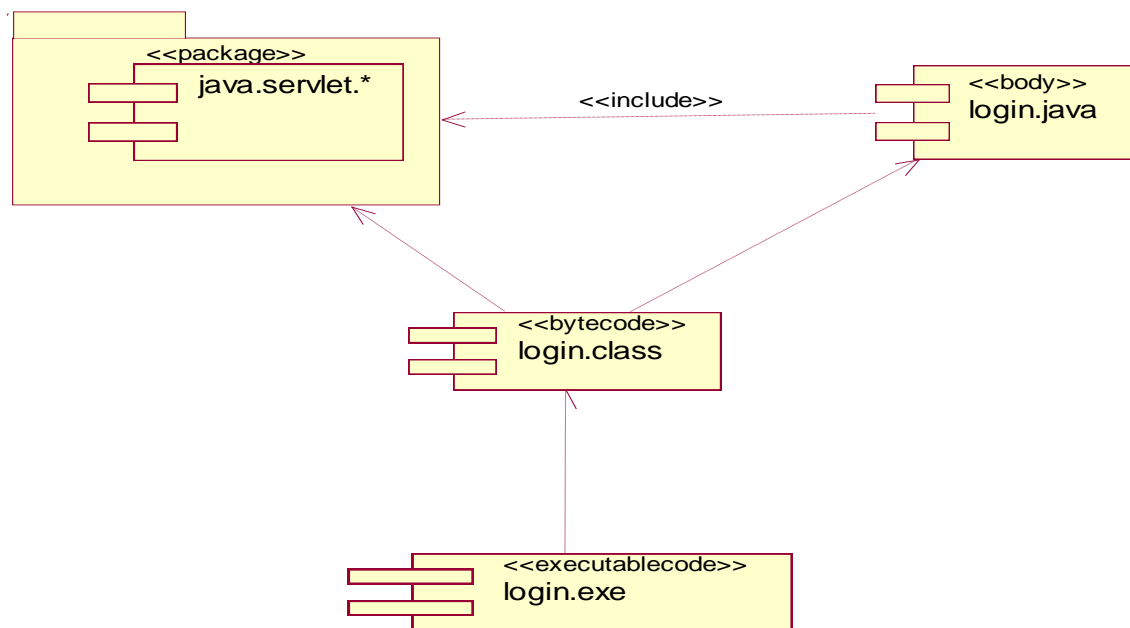


Figure 4.29 Component diagram for Login System (Frank, 2012).

Deployment Diagram

The second type of implementation diagram provided by UML is the deployment diagram. They are used to show the configuration of runtime processing elements and the software components and processes that are located on them. They are made up of nodes and communication associations, see figure 4.30.

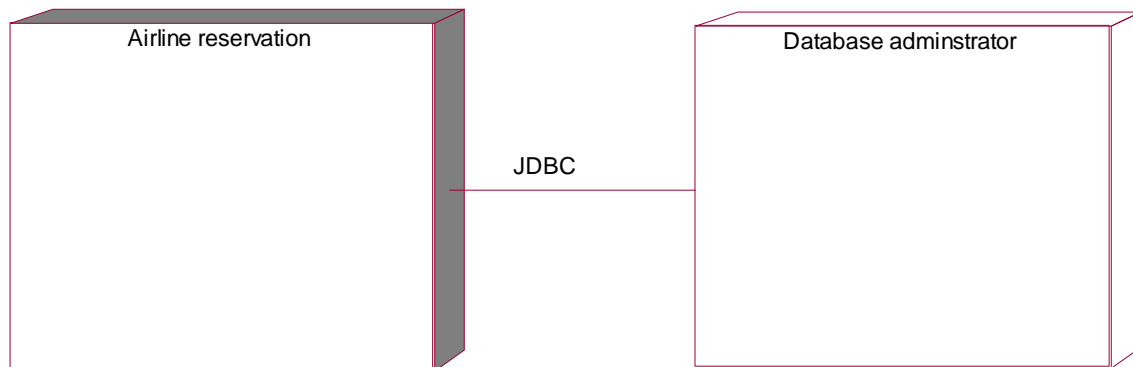


Figure 4.30 Deployment Diagram

4.6.1 Test Plan

The previous chapter provided some insight into the problem which this research attempts to tackle. We have briefly touched on the concepts related to agents and agent-based systems. We also described the process, models, and techniques in three prominent agent-oriented methodologies.

Testing is the process of evaluating a system or system components on manual or automated means to verify that it satisfies the specified requirements. The system is fairly simple in design and implementation.

Test Plan is defined as a strategic document which describes the procedure how to perform various testing on the total application in the most efficient way. This document involves the scope of testing, Objective of testing, Areas that need to be tested, Areas that should not be tested and Scheduling Resource Planning.

Types of Testing

Regression Testing: is one of the best and important testing. Regression testing is the process in which the functionality, which is already tested before, is once again tested whenever some new change is added in order to check whether the existing functionality remains same.

Re-Testing: is the process in which testing is performed on some functionality which is already tested before to make sure that the defects are reproducible and to rule out the environments issues if at all any defects are there.

Static Testing: is the testing, which is performed on an enhanced Intelligent Agent model application when it is not been executed.ex: GUI, Document Testing

Dynamic Testing: is the testing which is performed on an application when it is being executed.ex: Functional testing.

Alpha Testing: it is a type of user acceptance testing, which is conducted on an application when it is just before released to the customer.

Beta-Testing is a type of UAT that is conducted on an application when it is released to the customer, when deployed in to the real time environment and being accessed by the real time users.

Installation Testing is the process of testing in which the tester try to install or try to deploy the module into the corresponding environment by following the guidelines produced in the deployment document and check whether the installation is successful or not.

Furthermore, the methodology evaluation methods and frameworks available in the literature were also reviewed. On this basis, in this chapter we present the methods, including evaluation types and procedures that we employed to assess the three selected agent-oriented methodologies.

We also describe in detail the evaluation framework on which our assessment is based.

4.6.2 Test Data.

Before proceeding with assessment, one need to decide what evaluation methods should be used. This section answers this question by firstly sketching the purpose of our evaluation of agent-oriented methodologies. After that we describe the evaluation types and techniques that we used in this research.

4.6.3 Expected Versus Actual Result.

The expected and the actual results from the test data of the enhanced Intelligent Agent model for an Airline system are summarized in the tables 4.11 and 4.12:

Table 4.11 Subsystem Testing

Test Data	Expected Result	Actual Result
Reservation Testing	It is expect of software to display all the modules in Reservation on user's Interface.	The system does just that as expected.
Booking subsystem testing	It is expect of software to display all the modules in Booking subsystem on user's Interface.	The system does just that as expected.
Search subsystem testing	It is expect of software to display all the modules in Search subsystem on user's Interface.	The system does just that as expected.
Report subsystem	It is expect of software to display all the modules in report subsystem on user's Interface.	The same result was achieved.
Help Subsystem	It displays help modules on user's Interface.	The system does just that as expected.

Table 4.12: Subsystem Testing

Test Data	Expected Result	Actual Result
Create entry module Testing	It is expect of software to display all the controls of create record module on user's Interface.	The same result as expected.

Modify Record Module testing	It is expect of software to display all the controls of modify record modules on user's Interface.	The same result was achieved.
Delete Record Module testing	It is expect of software to display all the controls of delete record modules on user's Interface.	The same result was achieved.
View Reservation Module testing	It is expected of software shows all the controls in view modules on user's Interface.	The same result was achieved.

4.6.4 The Performance Evaluation

There seems to be an agreement among the evaluation methods community (Law, 2012) that the first and very important step of any evaluation is to identify its purpose. Law emphasized that **no rational comparison is possible without defining the purpose of the exercise.** Depending on the purpose, the method of carrying out an evaluation and the results may vary significantly. For example, an organization conducting an evaluation with the aim of adopting a new methodology for its existing development process may require a formal but costly evaluation procedure.

This is due to the importance of the evaluation results with respect to the organization's success. The selected methodology must be best suited to the organization's needs and must require no significant changes to its current practice process.

The purpose of our evaluation of several prominent agent-oriented methodologies is different.

Since the development of AOSE methodologies is still at an early age, practical evaluation purposes such as choose adopting a selected AOSE methodology to the current software development of an industrial organization seem inapplicable. Rather, our aim is:

- Better understand the nature of AOSE methodologies, including their philosophies, objectives, features, etc.
- Identify their strengths and weaknesses as well as the commonalities and differences in order to perform classifications and to improve future agent-oriented software system development.

Furthermore, it is emphasized that we are not trying to search for, in isolation, a best methodology. We believe that it is not always the case that all the AOSE methodologies are **mutually exclusive**. In fact, different methodologies may be appropriate to different situations, thus a methodology should be selected on the basis of considering different issues. These influencing factors can be the context of the problem being addressed, the domain, and the organization and its culture. However, we also expect that the evaluation would help in practical choices such as identifying the domains of applicability of each evaluated methodology.

Implementation and Testing/Debugging

So far, none of the three methodologies has offered sufficiently detailed process and techniques to allow the developers to perform the implementation phase. In our perspective, there is generally a close relationship between the detailed design and the implementation phase. As such, products from the design and analysis phases can be employed and applied to implement the system.

That close relationship also allows implementation to be derived from a detailed design by either automated code generation or performed by hand. In short, it promotes a smooth transition between development phases.

Additionally, during implementation, testing or debugging methods are essential, which should relate to other concepts used in the analysis and design. MaSE and Prometheus also have some recent efforts in supporting the testing/debugging phase although it is not fully

integrated into the methodology yet. Nonetheless, behavior verification (MaSE) (Juan.2012) and interaction protocol debugging (Prometheus) (Jennings, 2012) can be employed under the unified methodology.

4.8 System Conversion

4.8.1 Change over Procedure/Process.

This is the systematic way of changing from the old process – individual agent oriented methodologies to a hybrid methodology.

This could be achieved using any of the following procedures:

- Direct change over
- Parallel change over
- Pilot change over

Direct Change over.

This is the introduction of the new enhanced methodology and completely abandoning the individual agent oriented methodologies.

This has the advantages of reducing the cost of developing agent oriented methodologies.

Parallel Change over.

Here the individual agent oriented methodologies and the enhanced methodology is used simultaneously for some time to ensure that the enhanced methodology has more features than the individual agent oriented methodologies. This offers a much reliable comparison between the two methodologies

Pilot Change over

This is a situation where the enhanced methodology is tried out somewhere while the other parts of the individual agent oriented methodologies are unchanged until the enhanced methodology is proved perfect in its function.

However, after due consideration of all the changeover procedures, parallel-change is recommended and adopted.

4.9 System Maintenance

Every piece of Software from time to time needs to be reviewed and necessary maintenance involves the adjusting and improving the system by having the system upgraded and periodic evaluation and by making changes based on the new conditions.

Maintenance includes not only monitoring the information system but keeping the system running to keep pace with the new products, service regulations and other requirements.

4.10 System Training

The Evaluation Type and Procedure

There are several factors, as proposed in **DESMET (Determining an Evaluation methodology for Software Methods and Tools)**, that may affect the decision of choosing an appropriate type of evaluation and procedure to carry out the evaluation.

These are the **available time**, the level of **confidence** we need to obtain in the results of the evaluation, and the **cost** of the evaluation.

Taking into account the main purpose of our evaluation, we examine each factor below.

Evaluation timescales: Our research takes several months. Therefore, a small Feature Analysis survey and Case Study are likely to be candidates. In contrast, a quantitative Case Study may take more than the four months which are available to us.

Confidence in results: As stated in our purpose, we are not aiming at obtaining very highly reliable results which are required in the context of industry.

Rather, results ranging from medium to high confidence are suitable. Thus, all three forms of Feature Analysis (Case Study, Survey and Formal Experiment) plus Quantitative Survey

Hence, the options which were open for us to make our evaluation are a **Feature Analysis** incorporating a small survey and case study. In addition, to understand the similarities and differences between the methodologies we performed a **Structural Analysis**.

A multi-stage selection is needed if the evaluators face a wide range of methodologies. We were in a similar situation since; there are more than fifteen agent-oriented methodologies in the literature. Since we did not have enough time and resources to examine all these

methodologies in detail, it is necessary to reduce the number of apparently suitable methodologies to a few so that we are able to perform a sufficiently detailed evaluation. Hence, the multi-stage approach to evaluation seems appropriate.

The main evaluation procedures that we performed are described below.

First round qualification

In order to help perform a preliminary qualification round to select several methodologies to evaluate, we imposed several criteria upon which the selection can be based.

- i **Relevance:** To some extent, the methodology must be widely regarded as an agent-oriented methodology rather than, for example, an object-oriented methodology.
- ii **Documentation:** The selected methodology needs to be described in sufficient detail. For example, it needs to have been presented in books, journal papers, or detailed technical reports rather than just a conference paper. In addition, it is also important that we are able to access these descriptions.
- iii **Tool support:** Methodologies that have supporting tools are preferred over those that don't.

Since the evaluation process involves the practical use of each selected methodology to design Traveller Agent System, the availability of tool support is a practical advantage.

It is also a good indication of maturity and of the development effort that has gone into a methodology.

The decision of selecting the three methodologies to evaluate was based on the above criteria.

Feature Analysis

As first step, we built an evaluation framework which contains attributes, features and criteria. These are based on the existing work in the comparison of Object Oriented methodologies. In addition, there are features and attributes that are unique to agent-oriented methodologies.

By including various issues that have been identified as important by a range of authors, we avoid biasing the evaluation by including only issues that we consider important.

Having constructed the framework, we then carried out our evaluation for each methodology. However, it is emphasized that our evaluation mainly focused on the technical features of the methodologies. An organization carrying out a full evaluation may take into account more

features, and assign scores depending on the effect of the different feature in its environment or toward a particular project.

Case Study

The aim of this small case study was to study each methodology's ability to solve a small problem which is a simplified extracted from a large real problem. Its main merit is to explore whether a methodology is in fact understandable and usable. In addition, the evaluated methodologies were employed to design the same application. For these two reasons, the case study is useful since the evaluation results can be made based on a direct comparison between the selected methodologies. Therefore, during the study, the perceived advantages as well as limitations of the methodology were identified.

Structural Analysis

A feature-based evaluation (including case study and survey) generally only addresses how much support an agent-oriented software engineering (AOSE) methodology appears to provide for a feature, i.e. what degree of support seems to be present.

Our purpose in this research is also to attempt to understand the common and different aspects of the three AOSE methodologies. In other words, we tried to explore, what models and processes the three methodologies share and what the distinguishing aspects for each of them are. We believe this will contribute to the development of next generation agent-oriented software engineering methodologies.

The Evaluation Framework

In this section, we describe a methodology evaluation framework within which the feature-based comparison is conducted. The framework consists of a set of criteria which addresses not only classical software engineering attributes but also properties which are uniquely found in AOSE. In order to avoid using an inappropriate comparison framework, the properties in our framework are derived from a survey of work on comparing AOSE methodologies and on

comparing OOSE methodologies. The evaluation framework covers four major aspects of each AOSE methodology:

Concepts, Modeling language, Process, and Pragmatics. This framework is adapted from various frameworks proposed in (John, Colleen, & Suzanne, 2010) for comparing Object-Oriented methodologies. In addition to the above four components, those frameworks consider other areas such as Support for Software Engineering and Marketability. For our framework, we have decided to address "Support for Software Engineering" criteria in various places in the above four major aspects. With regard to Marketability issues, since all of our evaluated AOSE methodologies are still being researched and developed, we do not believe that marketability criteria are applicable.

For each feature or attribute or criterion of the evaluation framework, a brief description is provided together with several guidelines that help us in assessing a methodology against this feature. In addition, there are two kinds of evaluation features.

One indicates how much support time, cost and resources constraints. Rather, the aim was to avoid any particular bias by having a range of viewpoints.

A methodology appears to provide for a feature, i.e. what degree of support seems to be present. For this type of evaluation criteria, we use a judgment scale ranging from 1 to 5 where 1 indicates a low level of support and 5 implies that the methodology provides a high level of support. The other type of evaluation feature indicates **what** is supported by a methodology. These criteria are marked with the text "Narrative" next to them.

For each feature or criterion, we briefly describe its definition together with the relevant sources in the literature where it is discussed. A brief discussion of why it is important is also provided if necessary. Furthermore, we provide some **preliminary** guidance to identify the degree of support of an agent-oriented methodology with respect to a particular feature. The guidance is phrased as questions and refers to the methodology.

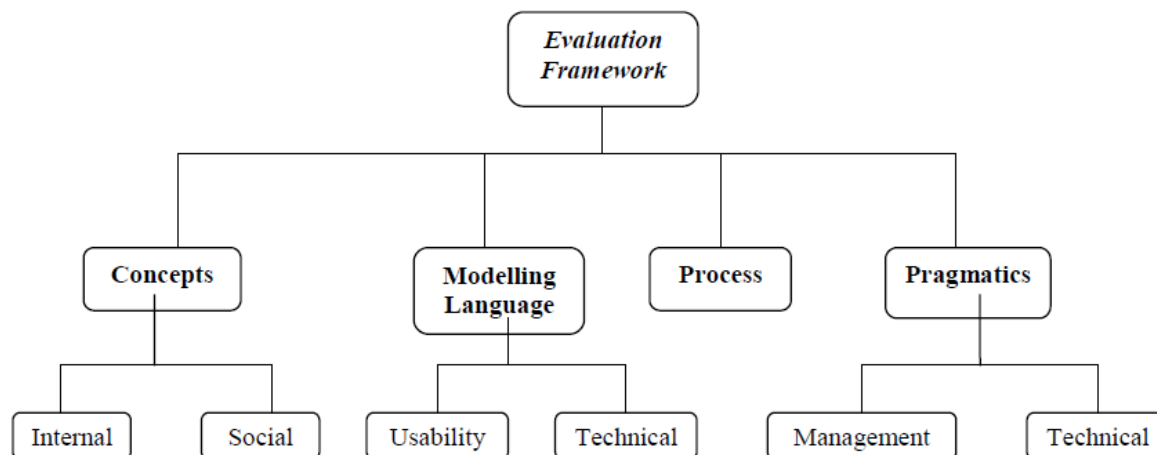


Figure 4.31: Evaluation framework at a high level view :(Verharen & Dignum,2012)

Concepts

The concepts related to agents distinguish an agent-based system from other types of systems. Hence, one of the important facets of evaluating agent-oriented methodologies is an examination of the methodologies' support for agent-based systems' characteristics such as **autonomy, pro-activeness, reactivity**, etc.

In other words, it is necessary to understand to what extent an AOSE methodology supports the development of agent-based systems that possess these characteristics.

We divide the agents' properties into two groups: **internal features** and **cooperation features**. The former addresses the characteristics that are internal to agents, whereas the latter are concerned with the cooperation process between agents. Each of them is described in detail below.

Internal properties

1. **Autonomy**: Agents can operate and make their own decision on which action they should take without direct intervention of humans or other agents. In other words, both agents' internal state and their own behavior are controlled by themselves.
2. **Mental attitudes**: This feature relates to the **strong agency** definition of agents as discussed in section 2.1.1. The three major elements of the Belief-Desire-Intention (BDI) architecture of

agents are an example of it. The BDI architecture defines an agent's internal architecture by its beliefs, the desires or goals it wants to achieve and its intentions or plans to accomplish those goals.

3. **Pro-activeness:** an agent's ability to pursue goals over time

Does the methodology provide a goal modeling technique, capturing the system's goals and the agents' goals?

Does the methodology provide plans and/or tasks models which depict how goals are achieved by an agent, e.g. performing a specific actions, or interacting with other agents, etc?

4. **Reactivity:** an agent's ability to respond in a timely manner to changes in the environment.

Does the methodology provide mechanisms to represent changes in the environment, e.g. events, incidents, etc.?

Does the methodology provide mechanisms to specify and represent the agents' responses to those changes in the environment?

5. **Concurrency:** agent's ability to deal with multiple goals and/or events at the same time. More specifically, agents are able to perform actions/tasks or interact with other agents simultaneously.

Can task be modeled by the methodology in such a way that concurrency or parallelism can be expressed?

Does the methodology provide models and techniques to capture the concurrent characteristics of a conversation between agents? In other words, can the communications between agents described in a fashion that one agent is allowed to interact with more than one other agent at the same time? Does the methodology provide support for detecting and avoiding problems that arise from concurrency such as race conditions, deadlock, and goals and/or resources conflicts?

6. **Situatedness:** agents are situated in an environment. They are able to perceive the environment via their sensors and to initiate actions to affect it using their effectors. Does the methodology support the modeling of the environment where agents are working? For instance, events happening within the environment are captured as well as the actions that the agents can perform in responding to these events. How well does the methodology address modeling the environment through, for example, percepts and actions? What types of

environment does the methodology support? According to Russell & Norvig, the environment can be classified using five different aspects:

Accessible vs. Inaccessible: Percepts do not contain all relevant information about the world in an inaccessible environment.

Deterministic vs. Nondeterministic: current state of the world does not uniquely determine the next in a nondeterministic environment.

Episodic vs. Non episodic: not only the current (or recent) percept is relevant to the situated agent in a non episodic environment.

Static vs. Dynamic: dynamic means that the environment changes while the agent is deliberating.

Discrete vs. Continuous: there are indefinite numbers of possible percepts/actions in a continuous environment.

Social features

In real world organizations, people, especially those who are working on the same project, need to cooperate from time to time. This is because; cooperation and teamwork are proven, in most cases, as the effective ways of tackling large project and making use of distributed expertise. That social behaviour is also borrowed by the agent-oriented paradigms.

A multiagent system consists of a number of various agents that work together to fulfill the common objective design of the whole system. Working together means that the agents existing in the system must **cooperate**.

Similar to humans, the cooperation between agents accelerates the process for analyzing and resolving the problems as well as increases the quality of the solutions or the products (Munindar, 2012).

The importance of cooperation indicates the need for a methodology to provide general principles and techniques to support it. For instance, it is necessary to assist the designers in building and preserving co operations between agents within the system.

The following evaluation criteria examine various issues relating to this dimension of agents.

1. **Methods of cooperation:**(Narrative). This criterion addresses the cooperation models (i.e. how cooperation is to take place) supported by the methodology.

What cooperation modes are supported by the methodology? For example, there are different types of method for cooperation (Frank, 2012) as described below.

Task delegation: Tasks are identified based on the process of decomposing overall goals. There is an agent called "manager" or "facilitator" who is responsible for selecting the agent who will perform each (or a group of) task. Each agent needs to carry out tasks that are assigned to them.

1. Negotiation: Unlike task delegation, there are no centralized agents or mediators to handle the agent cooperation. Instead, a society of self-interested agents uses a negotiation process to reach agreements with respect to co operations.

For example, in a trading environment, a buyer agent and a seller agent often negotiate (e.g. about prices, goods, etc.) with the aim of cooperating to achieve their own goals.

Multiagent planning: Plans are developed to achieve goals. These plans are described in terms of tasks. Tasks are distributed to agents using task delegation method.

Teamwork: a group of agents working toward a common goal.

2. Communication modes:(Narrative) .Interactions between agents is mainly achieved via communication. Communication is also the basis for social organization in multiagent systems, as for human-beings. Without communication, the agent is purely isolated and not able to interact with other agents. This criterion addresses the question of what communication modes are used by the methodology.

What communication modes are supported by the methodology? For example, there are several types of communication (Munindar, 2012)

Direct: communication can take place directly between two agents, e.g. exchanging messages.

Indirect: communication between two agents is done via a third party. For instance, if agent A wants to communicate with agent B, then A need to send a message to agent C, and C passes that message to B.

Synchronous: the sending agent does not continue the conversation until the message is received, e.g. making a phone call.

Asynchronous: In contrast to the above communication type, the sending agent can goes on exchanging messages immediately after sending a message, e.g. sending emails.

3. **Protocols:** This criterion examines the level of support for defining the allowable conversations in terms of valid sequences of messages exchanged between two agents (Munindar, 2012). Therefore, it is useful if the methodology provides models and techniques to define the specification of protocols that characterize agent conversations.

Does the methodology provide textual templates of the communicative act sequence?

Does the methodology have a way of representing the protocols such as finite state machines, AUML sequence diagrams, Petri Nets, etc?

4. Communication language:(Narrative).This concerns the language used for communication between agents.

What communication languages are supported by the methodology? For example, there are two typical agent communication language (Munindar, 2012):

Signals: The communication language is at a low level.

Speech acts: The communication language is at a high level (knowledge level),e.g. Knowledge Query and Manipulation Language (KQML), FIPA Agent Communication Language (ACL).

Modeling Language

Just as agent-oriented concepts are the basis for any AOSE methodology, so the modeling language for representing designs in terms of those concepts is generally the core component of any software engineering methodology. The modelling language, also called model or notation, of a methodology provides the foundation of the methodology's view of the world (Frank.2012). This view is generally an abstract representation of the important aspects of the system under development. Having a good modelling notation the methodology effectively eases the complex tasks of requirement analysis, specification analysis and design. Therefore, measuring the quality of the modelling language of an AOSE methodology plays an important part in our evaluation.

A typical modelling language consists of three main components (Frank,2012): symbols (either graphical or textual representation of the concepts), syntax and semantics. They together are used to fulfill three major purposes of a modelling language of a software engineering methodology (Frank,2012).

Firstly, it is a channel of communication, i.e. providing a means for software developers to exchange their thoughts and ideas.

Secondly, using a modelling representation one is able to capture the essence of a problem or design in such a way that the translation or mapping from it to another form (e.g. implementation) can be done without loss of detail.

Thirdly, the modelling language provides a presentation that gives users clear understanding of the problem. Based on its constituted components and purposes, we categorize the criteria assessing the modelling language of each methodology into two groups: **usability** and **technical** criteria.

Usability criteria reflect the first aim of a modelling language, i.e. providing the way for users to exchange thoughts and ideas.

On the other hand, **technical criteria** aim at the second and third purposes. The evaluation criteria belonging to the two groups are elaborated in detail as follows.

Usability criteria

Usability criteria consist of various measures: clarity and understandability, easy to use, adequacy and expressiveness.

Clarity and understandability: These two criteria are closely related to each other and both of them are fundamental requirements of a modelling language. In fact, a Methodology which provides clear notations tends to increase the users' understandability of the models.

Are symbols and syntax well-defined? Is no overloading of notation elements? For example, notation symbols are not similar to each other, and the most used concepts have simple notation.

Can the models be constructed at various levels of abstractions?

Does the methodology support for capturing different perspectives of the system, e.g. views from programmers, system analysts, managers, users, etc. Increase the level of clarity and understandability of the models?

2. **Adequacy and expressiveness:** These two criteria are related to each other. Differing from the above criteria, these two criteria should be examined relative to the application's purpose.

They measure whether a modelling language represents all the **necessary** aspects of a system in a clear and natural way.

Necessary here also means expressiveness, meaning that there is no need to have modelling constructs that result in an increase of complexity rather than promoting the clarity and understandability (Munindar, 2012).

Is the notation capable of expressing models of both static aspects of the system and dynamic aspects? Static models are those that represent relationships such as aggregation, specialization, structure of the system, and the knowledge encapsulated within the system.

Dynamic models describe the processing, agent interaction; stage changes, timing, and data control flow within the system.

Does the methodology allow the system under development to be modeled from different views such as behavioral, functional and structural views (Munindar, 2012).

Does the methodology provide mechanisms to express various aspects of the system such as exceptional conditions, boundary conditions, error handling, initialization, fault-tolerance, performance, and resource constraints?

Ease of use: It is important for a modelling language not only to be understandable to the users but also to be easy to use. The first step toward using a modelling language is to learn the notation.

Hence, it is desirable that the notation be **easy to learn** by both expert and novice users (96). In addition, the easier the users can remember the notation, the quicker they are able to learn to use it. Therefore, the notation should be as simple as possible.

Furthermore, since people usually sketch models by hand during the process of brain-storming or reviewing designs, it is essential for the notation be **easy to draw and write by hand** (Rumbaugh,2011).

Finally, as mentioned earlier, one of the important purpose of a modelling language is to convey information among the users. Often this is in the form of hard-copy documentation for reading and discussing.

Hence, it is important that the diagrams produced **are easy to read and comprehend when printed**(Rumbaugh,2011).

Does the notation contain symbols that are familiar to users and easy for them to remember?

Is it easy to draw and write by hand the notations provided by the methodology?

Are the diagrams produced using the methodology clear, e.g. containing no distraction or unnecessary marks, and making good use of space? Are important concepts captured more prominently than minor ones?

Technical criteria

Technical criteria consist of three different evaluation considerations: unambiguity, consistency, traceability, refinement, and reusability.

1. **Unambiguity:** symbols and syntax are provided to users so that they can build a representation of a particular concept. Thus, the semantic or meaning of a concept is the users' interpretation of the representation provided.

However, this interpretation can be different from observer to observer, which in turn results in misunderstandings. Therefore, it is important to make sure that a constructed model can be interpreted unambiguously (Frank, 2012).

Are the semantics of the notation clearly-defined? For instance, the mapping between concepts and notation needs to be unambiguous. Common and uniform mapping rules are employed. In addition, it is not recommended to have representations that are more complex than the nature of the relationship that they are trying to describe.

Does the methodology provide techniques for checking the models to make sure that all ambiguities have been eliminated?

2. **Consistency:** models should not contradict each other. This property becomes more important as the design evolves. More specifically, the representation of various aspects of a system such as structure, function and behavior should be consistent (Rumbaugh,2011)

Does the methodology provide guidelines and techniques for consistency checking both within and between models?

Is the methodology supported by tools that provide model consistency checking?

Is data dictionary used to avoid naming clashes between entities?

Traceability: There are relationships between models and between models and the requirements of the target system. Traceability requires that it has to be easy for the designers and the audiences of the design documents to understand and trace through the models. This may increase the users' understanding of the system. Tracing backwards and forwards between models and stages also allow the users to verify that all the requirements of the system are addressed during the analysis and design stages. Traceability also assists the designer produce new models by referring to the models that have been previously constructed. A result of doing this may be increased productivity in the sense that information gathered from one model can be used to construct others (Wood, 2012).

Is there a clear and easily recognizable path from early analysis to implementation via different modelling activities (Frank, 2012)?

Are naming conventions for entities used across models?

Can design decision be recorded in some forms?

Are there rules, either formal or informal, for transforming one model into other models? For instance, transform abstract analysis constructs into more concrete design artifacts.

Refinement: Adding more detail into a model is called refinement. As discussed in refinement is a way of developing a design since it allows the developers to develop and fine-tune design artifacts at different points in the development process. Hence, it is desirable that a methodology should provide mechanisms to support refinement.

Is the modelling language integrated into the development process?

Can a model incrementally built? For instance, the designers can start from the most abstract level to subsequent levels of detail.

Is there a seamless transformation from one level of abstraction to another without causing the loss of semantics?

Reusability: support for the reuse of design components.

Does the methodology provide mechanisms to reuse existing components or to derive new components from existing ones?

Does the methodology support the use of modularization, generalization, design patterns, or application frameworks?

Process

As discussed above, the **modelling language** is considered as a crucial part of any software engineering methodology. However, in constructing a software system, software engineering also emphasizes the series of activities and steps performed as part of the software life cycle (Rumbaugh, 2011). These activities and steps form the **process** which assists system analysts, developers and managers in developing software.

According to Frank, an ideal methodology should cover six stages, which are enterprise modelling, domain analysis, requirements analysis, design, implementation and testing.

Methodologies which cover all aspects of the system development are more likely to be chosen because of the consistency and completeness they provide. More specifically, in evaluating the process" component of an agent-oriented methodology, we consider the following criteria.

1. **Development principles:** This criterion addresses the lifecycle coverage in a broad view. It examines the development stages and their corresponding deliverables described within the methodology. In addition, it examines the supporting software engineering lifecycle model and development perspectives (Rumbaugh, 2011).

What are the development stages supported by the methodology? For example, requirements analysis, architectural designs, detailed design, implementation, testing/debugging, deployment, maintenance, etc. Which software engineering approach does the methodology support? E.g. Sequential, waterfall, iterative development, etc.

Which development perspective is supported by the methodology? Is it a top down or bottom up approach or the combination of both?

2. **Process steps:** Differing to the above criterion, this one measures the lifecycle coverage in more detail. In fact, an important aspect in evaluating whether a methodology covers a particular process step is the degree of detail provided. It is one thing to mention a step (e.g. at this point the designer should do X") and another thing to provide a detailed description of **how** to perform X.

Since design is inherently about trades, detailed descriptions are usually expressed using **heuristics** rather than algorithms, as well as **examples**. Thus, in assessing support for process steps we identify whether the step is mentioned, whether a process for performing the step is given, whether examples are provided, and whether heuristics are given.

Is a particular step mentioned? Is a process for performing the step given? Is there any example provided to illustrate the use of the step? Is there any heuristic supplied?

Does the methodology provide decisions making by management such as when to move between phases, i.e. when a phase is completed and move to the next phase?

- 1. Supporting development context:** This criterion identifies the development context supported by the methodology. A development context specifies a set of constraints within which the software development has to take place.

Which development context is supported by the methodology? Below are number of development contexts described in (Frank, 2012).

Greenfield" is the least constrained in that development can be conducted regardless of existing software.

Prototyping" involves either performing prototyping as part of the software development process or delivering the final product on the basis of successively refining the prototypes.

Reusing" development context also has two facets. One the one hand, the methodology expressly covers the inclusion of reuse products into its process.

On the other hand, reuse requires the methodology to provide process steps for producing reusable products.

Reengineering" is the most constrained. It regards the software development as a process of improving legacy and existing systems with the purpose of making them more useful.

This criterion is important, especially for AOSE because, as emphasized in (Jennings,2011), one of the key pragmatic issues which determines whether the agent-oriented paradigm can be popular as a software engineering paradigm is the degree to which existing software can be integrated with agents.

Regarding this development context, it is important for a methodology to provide techniques to manage legacy systems effectively, to understand their structure design as well as to revive them to achieve a particular requirement of the organization.

Estimating and quality assurance guidelines: These two criteria determine if such guidelines are provided within the methodology process.

Estimating guidelines are important to task planning. Quality assurance guidelines provide the assessors with useful information in evaluating the merit of the delivered product.

Does the methodology provide estimating guidelines for estimating the cost, schedule, number of agents, etc. of the software under development?

Does the methodology provide quality assurance guidelines that describe how the quality of the software is to be assessed? Such qualities are reliability, performance, etc. Techniques to assess the quality can be by reading documents, inspecting or walkthrough.

Pragmatics

In addition to issues relating to notation and process, the choice of an agent-oriented software engineering (AOSE) methodology depends on the **pragmatics** of the methodology. This can be assessed based on two aspects: **management** and **technical** issues.

Management Criteria

Management criteria consider the support that a methodology provides to management when adopting it. They include the cost involved in selecting the new methodology, its maturity and its effects on the current organization business practices (Frank, 2012).

1. **Cost:** There are different types of cost associated with adopting the methodology as described below.

What is the cost of acquiring methodology and tool support? For example, the cost for reference material, software tools for project development, maintenance on software tools, etc.

What is the cost of training to fully exploit the methodology? This cost depends on the expertise required to use the methodology, which relates to other criteria relating to the modelling language (number of models, clarity, usability, understandability) and process (number of steps in each process).

How complex is the methodology? Is the methodology similar to familiar software engineering methodologies such as UML and RUP?

2. **Maturity:** The maturity of a methodology is a factor that can play an important role in determining the quality of a methodology. There are several ways to measure the maturity of a methodology, as described below.

What are available resources supporting the methodology? For example, conference papers, journal papers, text book, tutorial notes, consulting services, training services, etc.

Is the methodology supported by tools? For example, supporting tools can be tools for build models, diagram editor, code generator, design consistency checker, project management, rapid prototyping, reverse engineering, automatic testing, etc.

What is the methodology's experience" such as the history of the methodology use?

For instance, the number of applications that have been built using the methodology. In addition, industrial strength applications should be preferred over ones that have only been used to develop small demonstration projects. Similarly, applications developed by people not associated with the creators of the methodology are more highly rated.

Technical criteria

Differing from management issues, technical criteria look at a methodology from another angle. We use the following criteria, which are discussed in (O'Malley, 2011), to evaluate the technical aspect of the methodology's pragmatics. It is noted that the guidance for each criterion can be extracted from its description.

Domain applicability: This considers whether the methodology is targeted at a specific type of software domain such as information systems, real time systems or component-based systems. With regard to this issue, the methodology that is applicable to a wide range of software domains tends to be more preferred.

Dynamic structure and scalability: This measures the methodology's support for designing systems that are scalable. It means that the system should allow the incorporation of additional resources and software components with minimal user disruption. To the one end, that is an introduction of new components into the system. To the other end, that is the introduction of new agent into the system (i.e. open systems).

Distribution: This criterion measures the methodology's support for designing distributed systems. It means the methodology should provide mechanisms, including techniques and

models, to describe the configuration of processing elements and the connection between them in the running system. It shows not only the physical of the different hardware components that compose a system, but also the distribution of executable programs on this hardware. More specifically, such models need to depict the deployment of agents over the network.

Evaluation Results

We have described the methods that we employed to carry out the evaluation and the constructed attribute-based framework on which the evaluation was based. In this section, we present the results of the evaluation.

The differences and distinguishing differences of the three agent-oriented methodologies which are identified as a result of the **structural analysis** are presented.

Finally, we propose some suggestions to the unification of the three methodologies based on the results we found in the evaluation.

These are based on agent concepts, development phases, artifacts and models, modelling notations, nature of applications, and development tools. Our comparison results are summarized in the following tables:-

Concepts

The results of the evaluation of the five methodologies with respect to their **concepts** are shown in Tables 4.13 to 4.14.

Table 4.13: Illustrates the scale of the details within each development

PHASES	ROADMAP	MASE	PROMETHEUS
System Specification	Detailed	Medium	Detailed
Analysis	Detailed	Detailed	Detailed
Architectural Design	Abstract High -Level	Detailed	Detailed
Detailed Design	Not exist (Architecture)	Not exist (Architecture)	Detailed(BDI agents)

Table 4.14. Presents the measure of agent concept that each methodology support

Concept	ROADMAP	MASE	PROMETHEUS
Autonomy	Medium	Medium	High
Mental Attitude	Use Knowledge Schema	Agent do not have to be intelligent	Agents are Intelligent Agent (BDI)

Table 4.15. Shows the scale of the modelling criteria within each methodology.

CRITERIA	ROADMAP	MASE	PROMETHEUS
Clear Notation	Strongly agree	Strongly agree	Strongly agree
Ease of learning	Strongly agree	Strongly agree	Agree
Ease of use	Agree	Strongly agree	Agree
Adaptability	Strongly agree	Strongly agree	Agree
Traceability	Strongly agree	Agree	Strongly agree
Consistency	Agree	Agree	Agree
Refinement	Agree	Agree	Agree
Scalability	Agree	Do not agree	Do not agree
Concept Overload	Medium	Medium	Low

Table 4.16. Compares the properties of the methodologies.

PROPERTY	ROADMAP	MASE	PROMETHEUS
Openness	High	Low	Medium
Environment	High	Medium	Medium
Abstraction	High	High	High
Traceability	High	High	High
Modelling	Medium	High	High
Complexity	Low	Low	Medium
Ease of Use	Easy, require some	Easy	Complicated
Limitations	Lack of richer notation	Lack of richer notation	Highly
Language	Low	Medium	High
Reusability	High	Medium	Medium

Table 4.17. Illustrates the available activities in each development phase

PHASES	ROADMAP	MASE	PROMETHEUS
System Specification			Stakeholders scenario diagram
Analysis	Environment, Knowledge, Goal, Role, Revised Role Model, Social Model	Use cases, Goal hierarchy, Sequence, Concurrent task diagram	Goal overview, Role, Data coupling diagrams
Architectural Design	Agent Service, Acquaintance model	Agent classes, Conversations	Agent acquaintance, System Overview Agent Descriptors Protocols
Detailed Design		Agent's internal architectures, Deployment diagram	Process , Agent Overview Diagrams, and Capacity, Capability overview, Event, Data, and Plan

ROADMAP (Role Oriented Analysis and Design for Multi-Agent Programming)

Roadmap's analysis phase includes constructing a role model and protocol diagrams. In order to do so, Roadmap suggests the analyst follow the three process steps:

- (1) Make the prototypical roles model,
- (2) Make the protocol diagrams and
- (3) **Make the elaborated roles model.**

The stage is generally well-described in the related journal paper. However, the methodology seems to lack support for helping the analysts identify roles in the system. Involving the design phase, the student responded that even though there are some very general heuristics in helping the designers map roles to agent types, she found some difficulties in performing this step. More detailed and specific guidelines would be more helpful. Furthermore, during the process of constructing the Service Model, she was in fact able to define the permissions and the safety responsibilities of each role more properly and clearly. There is, however, a limitation in the Service Model in that the definition of notations used for the pre-conditions

and post-conditions are not mentioned. The final Roadmap design artifact, the Agent Acquaintance Model, tends to be the easiest one to build, according to her.

MaSE (Multi-Agent System Engineering).

MaSE gave some positive responses to the Analysis phase described in the methodology. MaSE has three process steps (i.e. Capturing Goals, Applying Use Cases, and Refining Roles) well-defined and easy to follow. MaSE is clear and easy to understand and to learn. The provided tool support (agentTool) was very useful in helping her to build analysis and design models and to check the consistency among them. However, there were some minor issues; for example as there is too much text on arcs in **concurrent task diagrams**, which made them hard to read.

Of the four Design steps described in MaSE, only the **Constructing Conversations**" step gave her some difficulties to follow; the other steps are well-defined and easy to apply. Regarding the "Constructing Conversations" step, there is a paper (Frank, 2012) proposing the use of semi-automatic transformation from analysis models to design models. The idea was also implemented in agentTool.

Prometheus

The system specification is the first phase of Prometheus. Its main purpose is building the system's environment model, identifying the goals and functionalities of the system, and describing key use case scenarios.

Firstly, one of the main characteristics of agents is situatedness. It means that agent systems are situated in an environment that is changing and dynamic. To some extent, situated agents need to interact with the environment. As a result, building the environment model is an important step in this system specification stage.

Despite the fact that goals are a new concept introduced in the agent-oriented paradigm, the student found it easy to identify system goals. However, determining the **external interface** (percepts, actions, data stored) was harder for him. In particular, he had some difficulties in identifying the relationship between **functionalities** and **actions** as well as telling the difference between actions and sending messages.

These confusions may suggest that the differences between these concepts need to be made very clear. More examples and heuristics in those cases are likely to be required.

Structural Analysis - The Commonalities

Weak agency regards pro-activeness is one of the key characteristics of agents. Agents are pro-active if they have **goals** that they pursue over time. It means that actions initiated by agents are more or less towards the achievement of their goals. Strong agency emphasizes that **goals** are part of the mentalistic and intentional notions of agents. Hence, agents' goals are arguably one of the most important concepts of agency.

In addition, goals are considered more stable than requirements and less likely to change (Frank,2012). For these reasons, Capturing Goals is regarded as one of the key steps in the agent-oriented development process proposed by the three methodologies which we have evaluated. Indeed, capturing goals is the first step of seven steps in MaSE, whereas identifying goals is part of the system specification in Prometheus. Goals are in the form of roles' responsibilities in Roadmap, which is more concrete than goals in the other methodologies. Overall, for all of the three methodologies, "Capturing Goals" is part of the requirements analysis and is used as a foundation of the identification of agents.

The mechanism of capturing goals basically includes **identifying** goals, **structuring** them and **representing** them. Most of the methodologies take the requirements specification (assuming its availability) as a basis for goal identification. They also tend to agree that system goals are abstractions of requirements (function and non-functional).

Structuring goals ranges from a simple AND/OR decomposition to more complex process. For instance, in MaSE goals are classified into four types: **summary goal** (an abstraction of several goals), **partitioned goal** (is achieved by accomplishing sub-goals), **combined goal** (a sub-goal are a combination of two or more similar parent goals), and **non-functional goals** (derived from non-functional requirements).

Prometheus does mention goal decomposition but does not go into detail. Except for Roadmap and Prometheus, the other methodology provides a graphical representation (goal hierarchy in MaSE) depicting the relationship between goals and sub-goals.

The Role of Use Cases in Requirements Analysis

Use cases have been proven to be an effective technique in object-oriented methodologies in eliciting requirements. Of the three AOSE methodologies, Prometheus, and MaSE make use of this technique. The purpose of using it in all three methodologies is to help the analysts identify the key communications/interactions between entities in the system. Prometheus proposes a structured use case scenario template covering related information such as incoming incident/percept/trigger, goals, message, and events. MaSE does not explicitly define a use case template, hence it is expected that analysts use those from object-oriented design. Nonetheless, MaSE uses UML-like sequence diagrams as a notation to express the communication paths between roles on the basis of the identified use cases.

Roles/Capabilities/Functionalities

Similar to their counterparts (i.e. objects) in object-oriented systems, agents are the key entities in agent-based systems. Therefore, one of the crucial requirements of all the AOSE methodologies is to assist the developers identify the agents constituting the system. The importance of agent identification is amplified when the target system is a multi-agent system, consisting of a certain number of agents.

A common technique used in all three methodologies to deal with agent identification is to start from entities that are "smaller" than agents. These entities are functionalities in Prometheus, and roles in other methodologies. Agents, or also called agent types, are formed by grouping these entities into chunks. There are different techniques and models provided by some of the three methodologies to help the designers group or map these chunks into agents.

Social System - Static Structure and Dynamics

One of the key motivations of introducing the agent-oriented paradigm into software engineering field resides in its attractiveness in designing and implementing complex systems. Therefore, it is very important for an AOSE methodology to provide a useful abstraction for understanding the overall structure of the system. Most of the three methodologies (all except

Roadmap) address this issue fairly well. In MaSE, the role-task-goal-conversation diagram (called Role Model) shows chunk overview as well as goal assignment. It represents the relations between the key constructs in the system such as: roles-tasks, roles-goals, and tasks-conversation. Prometheus, on the other hand, has a data coupling diagram which shows the relations between functionalities and data. More importantly, it has a system overview diagram depicting the social structure of the system as a collection of agents, messages between them, events, and shared data objects and resources in the system and its environment. These relationships are described at the coarse-grained level so that the social system structure is captured at the abstraction level above the concrete implementation.

If the importance of the existence of an overall view on the organizational system structure resides in their ability to represent the static structure, then it is also essential to capture the high-level dynamics of the system. These are the interactions and communication taking place between agents. They include descriptions of the mechanism by which agents coordinate their activities or other complex social interactions such as competition, negotiation, and teamwork. MaSE, Prometheus, and Roadmap describe interactions at two different levels of granularity. They include a set of high level interactions and a more detailed representation in terms of inter-action protocols. Roadmap only provides interactions at the level of the former.

MaSE, and Prometheus, are similar in that they model high level interactions using sequence/interaction diagrams borrowed from UML sequence diagrams. In addition, use cases are used to develop such sequence/interaction diagrams. Nonetheless, there are differences: interaction diagrams in Prometheus show interactions between agents whereas sequence diagrams in MaSE depict those between roles. Roadmap is also similar to MaSE in that interactions are discovered and modeled at the chunk" level (e.g. role) rather than agent-level. Rather, the model consists of purpose/motivation of the interactions, the initiator and responder, inputs/trigger conditions, and outputs/information achieved.

When moving down to the detailed level of interaction protocols, Prometheus, suggest the use of AUML interaction protocol diagrams. MaSE, on the other hand, defines a coordination

protocol using a form of finite state transition diagrams. Nonetheless, the interaction model in all the methodologies defines the legitimate sequence of the messages exchanged between agents.

Individual Agent - Static Structure and Dynamics

Agents are mostly considered as concrete entities in the agent-oriented software engineering methodologies. They are often the final products of the design and are mapped to code at the implementation stage. Therefore, in our view having techniques, guidelines and heuristics for analyzing and designing the internal architecture of each agent in the system are as important as defining the social system. As a result, the availability of a useful abstract model which characterizes the **static structure** and **dynamic behavior** of the agents is very helpful.

Prometheus answers this question fairly well. It has an agent overview diagram which provides the top level view of the agent internals. It depicts the capabilities (i.e. modules) together with their interactions within an agent. Similarly, MaSE's Architecture Diagram describes the architectural components, the connectors between these components within an agent (i.e. inner agent connectors) as well as connections with external resources or other agents (i.e. outer-agent connectors). Roadmap does not have any model describing the internal **structure** of the agent.

Nonetheless, Roadmap does provide a model to capture the **dynamics** within an agent. In the Service Model, the descriptions of the inputs, outputs, pre-conditions, and post-conditions for each function of the agent are provided. However, the techniques for describing agent's planning or scheduling capabilities are not supplied.

In addition, each plan in a capability is also depicted using a UML activity diagram. MaSE also describes this micro-level of dynamics but only for roles and at the level of tasks. Prometheus seem not to have this type of model even though Prometheus does have capability overview diagrams but only shows the relations between plans, events and resources. Prometheus also provides the individual plan, event, and data descriptors. These textual

descriptors provide the details belonging to individual agents, which are necessary to move into implementation.

Agent Acquaintance Model

Additionally, the dependency and interaction (i.e. communication paths) between agents are represented in terms of agent acquaintance diagrams in MaSE, Prometheus and Roadmap. These are directed graphs which include rectangles (representing agents) and lines with arrows (depicting the links/conversations/communications) between two agents. They vary in whether they depict cardinalities (Prometheus), and whether they indicate for each agent the roles included (MaSE). MaSE is also different from Prometheus and Roadmap in terms of specifying the name of the conversations. Whereas, the other two simply define the communication links that exist between agents without defining the actual characteristics of the link. Nonetheless, these links are described in detail later on in the design processes of MaSE and Prometheus.

Therefore, the purpose of having the Agent Acquaintance Model in the three methodologies is to assist designers in identifying the coupling and communication bottlenecks among agents in the systems.

Structural Analysis - The Differences

An agent-based system is placed in an environment which it interacts with. In fact, as situatedness is commonly regarded as one of the key properties of agents.

Therefore, an agent system needs to have a representation or a model of the environment in which it operates. Unfortunately, none of the methodologies answers this question well.

MaSE only mentions that interfacing with external or internal resources requires a separate role to act as an **interface** from a resource to the rest of the system. Tropos represents the resources (physical or information entity) as an entity but no further than that. This limitation involving having a model describing the environment and the interaction between it and the agents situated also occurs in Roadmap. Of the three methodologies, only Prometheus and MaSE address this issue and the way they approach it is different. Prometheus views the

environment from the **system perspective** by providing an interface model which contains a list of percepts and actions possible in the environment of the agents. This view is motivated by the fact that the system in general and the agents in particular perceive the environments via a number of sensors. They also operate on the environment using effectors. Therefore, it is necessary to have a clear input/output specification regarding the characteristic needs and requirements of the system. The model provided by Prometheus only captures this interface. A model of the environment that is internally used by the agents to represent their environment and to reason about is not described in the methodology.

The system perspective of the environment, however, is only one side of the coin. The developers of the system may need to have a different perspective on the environment. In contrast to Prometheus, addresses this need. Its Domain View represents domain specific concepts and relations. For instance, in the Travel Agency domain, concepts such as Travel Arrangements, Accommodation, Journeys, Planes, etc. and the relations between them are represented in the Domain Model. Nonetheless, neither of the models described in Prometheus is sufficient.

They do not deal explicitly with characteristics of the environment, such as whether the environment is inaccessible, nondeterministic, dynamic, continuous, etc. (Frank, 2012). These characteristics of the environment, if they are sufficiently captured, may affect the design decisions of individual agents (e.g. the mechanism of reasoning) as well as the system as a whole.

Deployment Model in MaSE

A **deployment model** is often used to describe the configuration of processing elements, and the connections between them in the running system. In addition, it shows not only the physical layout of the different hardware components that compose a system, but also the distribution of executable programs on this hardware. The use of a deployment model for such purposes is not new to information system development.

It is one of the key models in UML (Jennings, 2012). Its usefulness comes from three different sources.

Firstly, developing a high level deployment provides a foundation for assessing the feasibility of implementing the system. Secondly, during the process of constructing the deployment model, the designers may obtain a better understanding of the **geographical** and **operational** complexity of the system. Last but not least, the use of a deployment model also aims to give an estimate of various aspects such as cost. Despite the above benefits of developing a deployment model, only MaSE integrates it into the methodology's development process. MaSE's Deployment Diagrams, the realization of deployment model, depicts the system on the basis of agent classes and their location on the network.

In constructing the Deployment Diagram, the developers have a chance to fine-tune the system at the design stage by considering agent-related factors such as the number of agents, their types and locations, etc. against performance issues such as processing power, and network bandwidth. MaSE also offers the flexibility in system deployment design by allowing the designers to develop different system configurations and/or to modify them.

Data Coupling Diagram in Prometheus

The three agent-oriented methodologies share a similar technique of identifying the agent types in the system. That is the iterative mechanism of mapping goals to roles or functionalities, and roles or functionalities are assigned to agents. In our view, the second part of this process is not easy and the decision may affect later design activities. The choice of grouping roles or functionalities into one agent depends on several factors and more importantly needs to be supported by a method that guides designers in identifying issues.

However, most of the methodologies do not address this critical issue well. Indeed, the documentation of MaSE (Frank, 2012) and Roadmap(Jennings, 2012) has only a paragraph discussing this issue.

Prometheus is the only methodology that provides clear techniques to deal with agent identification. One of them is the use of agent acquaintance diagrams the other is the development of **data coupling diagrams**. These diagrams consist of the system functionalities

and the external resources in terms of identified data. More importantly, they also capture the **use and being used**" relationship between those functionalities and the data in terms of directed links. On the basis of visually assessing the data coupling diagram plus provided guidelines and techniques, the designers are able to group functionalities into agents. This is based on considerations of both **cohesion** and **coupling** - one wants to reduce coupling and increase cohesion. Putting agents that write or read the same data together seems to reduce coupling between agents. In other words, functionalities that use and/or produce the same data tend to belong to the same agent.

4.16 Performance Evaluation.

In the previous sections, we have looked at our evaluation analysis of the three selected agent-oriented methodologies. They are currently, in our view, among the most prominent AOSE methodologies. We have assessed their strengths and weaknesses based on a feature-based evaluation and a case study. In addition, their similarities and distinguishing differences with respect to their techniques and models were also examined.

On that basis, in this section we provide several proposals towards their unification by combining their strong points as well as avoiding their limitations. We believe such effort is similar in spirit to the one that gave birth to Unified Modelling Language. There was an effort of assembling agent-oriented methodologies from features.

However, their approach is to build a core methodology and to integrate additional features into it choosing from different methodologies. The integration is performed on an application by application basis. Our approach is different in that we Endeavour to make some preliminary suggestions to form a unified methodology based on the three selected methodologies. These suggestions, as we believe, may contribute a step towards developing the next generation" of agent-oriented methodologies. Such a methodology should support in sufficient depth all the features that we have identified.

They are summarized in Table 4.18

Table 4.18: The basic features of an enhanced model for the development of intelligent agent.

Components	Features
Concepts	<p>Internal properties: <i>autonomy, mental attitudes, pro-activeness, Reactivity, concurrency, and situatedness.</i></p> <p>Social properties: <i>methods of cooperation, teamwork, Communication modes, protocols, and communication language.</i></p>
Modelling language	<p>Usability criteria: <i>clarity and understandability, adequacy and Expressiveness and ease of use.</i></p> <p>Technical criteria: <i>unambiguity, consistency, traceability, Refinement and reusability.</i></p>
Process	<p><i>Full life-cycle coverage, iterative development which allows both top-down and bottom-up design approaches</i></p> <p><i>Sufficiently detailed process steps with definitions, examples, Heuristics, management decision making. Estimating and quality assurance guidelines are provided.</i></p> <p><i>Supporting various development contexts such as: Greenfield, Reuse, Prototype and Reengineering</i></p>
Pragmatics	<p>Management criteria: <i>cost effectiveness, and maturity</i></p> <p>Technical criteria: <i>a wide range of domain applicability, support For the design of scalable and distributed applications.</i></p>

As can be seen in the Process feature, a relatively complete and mature AOSE methodology should at least cover the requirements analysis, design, implementation and testing/debugging phases. The models and techniques used in these stages can be formed as a unification of those used in the three existing agent-oriented methodologies that were examined in this research.

Requirements Analysis

It seems to us that there are four steps that are important in this stage:

1. **Capturing Goals:** Identifying and structuring goals is an important stepping in eliciting requirements in most the three methodologies. ROADMAP can be structured and represented as a goal hierarchy diagram similar to the one used in MaSE.
2. **Defining use case scenarios:** The structure of use cases can be the one proposed in Prometheus. Sequence diagrams as in MaSE can be used as a realization of use cases to show the communication path between roles in the system.

3. **Building the environment model:** The environment can be modeled from the perspectives of both the system and the developers. For the former, an Environment Interface Model describing possible percepts and actions in the environment of the agents like the based on the one in Prometheus can be used. Regarding the latter, we can employ the Domain View in MESSAGE which captures specific organizational concepts and relations. In addition, characterization of the environment and the target run-time environment needs to be examined in this step.
4. **Defining roles:** The above two steps seem to provide enough information for the system analysts to determine a number of key roles or a "chunk" of functionalities existing in the system. It seems to us that the Role Schemata that are mainly used in Roadmap and MaSE can be applied to formally define a role. Another alternative is the functionality descriptor used in Prometheus.

The four steps above and their respective artifacts, in our view, provide a strong support for eliciting requirements, identifying system goals, capturing the environment and defining key roles in the system. They promote the developers' understanding of the requirements and form the inputs to the design state.

Architecture Design

The unified methodology may describe the architectural design as the following two steps:

1. **Building social system static structure:** An important aspect of this step is to identify the agent types existing in the system. Two techniques that are commonly used are data coupling diagrams (Prometheus) and agent acquaintance models (Roadmap, MaSE, Prometheus).
In addition, dependency relations describing dependencies between agent types for using resources, performing tasks or achieving goals need to be modeled. Regarding this, the System Overview Diagram in Prometheus can be applied.
2. **Building social system dynamics:** the social behaviors and interactions of the system can be captured at a high level using the AUML interaction protocol diagrams (similar to the ones

employed in Prometheus,). At a lower level, in our opinion, the Concurrent Task Diagrams of MaSE can be employed to model the concurrent interaction protocols.

Detailed Design

The detailed design should focus on constructing individual agents' architecture in terms of defining the components and the connections between them.

1. **Internals of individual agents:** In describing the top level view of the agent internals, the agent overview diagram used in Prometheus can be employed. For further detail, the individual plan, event, and data descriptors (also in Prometheus) can be used. The service model used in GAIA is also a good representation of the inputs, outputs, pre-conditions, and post-conditions of each service (i.e. capability) provided by an agent.
2. **Dynamics of individual agents:** The dynamic behavior of individual agents can be modeled using capability diagrams and plan diagrams.
3. **Deployment model:** The deployment of agents within or across the network can be represented using a deployment model as in MaSE.

4.11 System Documentation

Users' manual provides guidelines on how to use the enhanced methodology. A user manual both in an electronic or hard copy form should be made available for further research and students in the field of Computer Science, Computer Engineering, and Information Technology etc.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary

In this section, we discuss the deficiencies and limitations of existing agent oriented methodologies which we have addressed and solved in this dissertation and those that are not solved as well. The addressed problems are stated first followed by a discussion.

No standard has evolved: Developing and establishing agent standards is not an easy task, because the standardization process shifts the debate from longer term research issues to the ability to practice commercial agent systems. Therefore, Enhanced Multi Agent System Development (EMASD) methodology does not propose any standard definitions, nor agent architecture or any aspects related to agent languages which can be considered as a standard.

Industrial development suitability: Although, Enhanced Multi Agent System Development (EMASD) methodology is still new, it is difficult to assess whether it is suitable for industry or not. However, we assure it will be suitable for industrial development of multi-agent systems and it will be accepted in the industrial domain. That is because of its simplicity, easy to learn and its completion of stages of systems development.

Neediness for formal semantics: Despite the Enhanced Multi Agent System Development (EMASD) methodology is developed based on concepts chosen in precise manner to cover most of the existing agent definition patterns; Enhanced Multi Agent System Development (EMASD) methodology does not have any formal semantics.

Existing gap between design and implementation: The Enhanced Multi Agent System Development (EMASD) methodology bridges the gap between design and implementation. This is achieved by providing refined design models such as an agent container in the design phase. They can be directly transferred into implementation constructs in an available programming language.

It provides constructs to directly implement high-level design concepts.

Implementation phase inclusion: The Enhanced Multi Agent System Development (EMASD) methodology provides an explicit implementation phase and is considered as an essential phase of its process. Most of the existing methodologies do not include an implementation phase.

Support of multiple Roles: One of the most important aspects of the Enhanced Multi Agent System Development (EMASD) methodology is that it considers the role concept as a one of the main concepts that the methodology relies on. Enhanced Multi Agent System Development (EMASD) methodology assumes agents can play one or more roles at a time. This is achieved by providing the roles model in the analysis phase which describes the roles that the agent will play in the system.

Agent-oriented approaches: According to the agent methodology classification discussed in in chapter 2, The Enhanced Multi Agent System Development (EMASD) methodology is considered to be an agent based methodology. Therefore, the agent (and its internal components) developed by the Enhanced Multi Agent System Development (EMASD) methodology is built from scratch as an individual entity without dependence on any other traditional methodologies, such as object-oriented methodologies.

Environment features inclusion: The Enhanced Multi Agent System Development (EMASD) methodology takes into account the environment issues by providing the MAS architecture stage in the analysis phase. This stage describes how to identify relationships and interactions between the entities (agents) that inhabit the environment (MAS), the conversations and exchanged messages and the services that are offered by the agents in the system.

Software engineering issues: the Enhanced Multi Agent System Development (EMASD) methodology is established based on essential software engineering issues such as preciseness,

accessibility, expressiveness, domain applicability, modularity, refinement, model derivation, traceability, and clear definitions. The preciseness issue is represented in Enhanced Multi Agent System Development (EMASD) methodology by providing well known modelling techniques such as UML UCDs, UCMs, and UML activity diagrams which have clear semantics.

The accessibility issue is represented by modelling techniques that are easy to understand and easy to learn such as UCMs and UML UCDs. With respect to the expressiveness issue, the Enhanced Multi Agent System Development (EMASD) methodology addresses this issue by providing a clear step by step development process. This process describes the whole structure of the system.

In addition, Enhanced Multi Agent System Development (EMASD) methodology supports dynamic aspects by providing interaction diagrams and FIPA -ACL protocols. With regards to domain applicability Enhanced Multi Agent System Development (EMASD) methodology is independent and it can be applied to any application domain. The Enhanced Multi Agent System Development (EMASD) methodology addresses the modularity issue by providing organized phases for the MAS development process. Refinement is generally supported by Enhanced Multi Agent System Development (EMASD) methodology.

Enhanced Multi Agent System Development (EMASD) methodology uses iterative activities which are integrated into its development process.

Enhanced Multi Agent System Development (EMASD) methodology supports both model derivation as well as traceability issues. There are clear links between models provided by Enhanced Multi Agent System Development (EMASD) methodology. For instance, roles, agents, goals, and plans are all linked together. The Enhanced Multi Agent System Development (EMASD) methodology models are traceable and can be derived easily from each other, and they provide a mapping from one model or diagram to another. Most of the models in Enhanced Multi Agent System Development (EMASD) methodology are derived from the system scenario model. Enhanced Multi Agent System Development (EMASD) methodology is based on clear definitions.

Misconceptions: Enhanced Multi Agent System Development (EMASD) methodology is established based on precise accepted MAS definitions. These definitions are chosen from the literature to cover most of the existing agent definition patterns.

Incompleteness: Enhanced Multi Agent System Development (EMASD) methodology is considered as a complete development process. It provides a full lifecycle development process for MAS. This process starts with the initial specification, system requirements, and finally producing implementation code.

Incomplete formality: Enhanced Multi Agent System Development (EMASD) methodology does not address any formalism of MAS concepts.

Open systems: Enhanced Multi Agent System Development (EMASD) methodology is not intended to work with open systems. But it is designed to work for cross-boundary systems (semi-open systems) where the agent society itself is closed (i.e. the types and behaviors of agents defined in the system are determined a priori) but external agents may interact with members of the society via defined and common protocols.

5.2 Conclusion.

As agent-oriented approaches represent an emerging paradigm in software engineering, there has been a strong demand to apply the agent paradigm in large and complex industrial applications and across different domains. In doing so, the availability of agent-oriented methodologies that support the software engineers in developing agent-based systems is very important.

In recent years, there have been an increasing number of methodologies developed for agent-oriented software engineering. However, none of them are mature and complete enough to fully support the industrial needs for agent-based system development.

For all those reasons, it is useful to commence gathering together the work of existing agent-oriented methodologies with the aim of developing a future methodology that is mature and complete. Thus, this research is focused on developing a comprehensive design methodology to assist a multi-agent system designer through the entire software development lifecycle, beginning from the system requirement phase and proceeding in a structured manner towards a working code. There are few principal strengths of the methodology developed through this research work. First, it is based on three important aspects: concepts, models, and process, and it is focused toward the specific capabilities of multi-agent systems. At the commencement of research, EMASD methodology combined several techniques and concepts into a single, simple, traceable, and structured methodology. These concepts and techniques are represented

through a set of models. Most of these models used within the methodology have therefore been already justified and validated within the domain of agents and multi- agent systems. EMASD provides an extensive guidance for the process of developing the design and for communicating the design within a work group. It was very clear that the existence of this methodology provides a great assistance in thinking about and deciding on design issues, as well as conveying design decisions.

The EMASD methodology has captured all the requirements of the system in a proper way by combining well known techniques into one extensive model called system scenario model. Moreover, EMASD has introduced MAS concepts through conceptual framework where the concepts are determined, and selected. This conceptual framework has been used to introduce the MAS concepts that the new methodology relies on. EMASD methodology has proposed the use of the trigger concept which has allowed the representation of agent reactivity.

Finally, EMASD has proven its ability to support organizational aspects by utilizing the role concept which provides the work at different levels of abstraction.

Review of Achievements

Therefore, following this comparative analysis, we proposed a unification scheme for three key methodologies.

With the aim of performing a systematic and comprehensive evaluation that is our purpose, we reviewed the literature sources that provide evaluation methods. In fact, such approaches and techniques are numerous, ranging from simple and preliminary comparisons to formal and scientific experiments. In addition, there are different evaluation types:

Qualitative approaches (otherwise known as **Feature Analysis**) assess a methodology in terms of the features provided by the methodology, whereas **quantitative** techniques focus on the measurable effects in terms of reducing production, rework, maintenance time or costs that are delivered by the methodology.

Both types of approaches, however, target the same goal - to evaluate methods of developing software to examine how good they are.

Thus, the focus was kept on the **qualitative attributes** of an AOSE methodology that support the process of engineering. They are the concepts and characteristics that are specific to

agents, the notation supporting modelling activities, the development process, and the pragmatics of the methodology itself. These attributes, features, and properties constitute the **evaluation framework** that we have employed to examine the **strengths** and **weaknesses** of the three selected agent-oriented methodologies.

This evaluation framework includes a range of issues that have been identified as important by a range of authors from software engineering, object-oriented methodologies evaluation, and agent-oriented methodologies evaluation.

By doing so, we ensure that the framework is unbiased and complete, i.e. covers all significant criteria. In addition to the **Feature Analysis**, we addressed in depth the second goal of this evaluation (i.e. identifying commonalities and distinguishing differences) by performing a **Structural Analysis**.

Overall, all three methodologies provide a reasonable support for basic agent-oriented concepts such as autonomy, mental attitudes, pro-activeness, and reactivity. They all are also regarded by their developers and the students as clearly agent-oriented. However, there are several characteristics of agent-based systems that are not addressed or sufficiently addressed in most of the methodologies.

For instance, none of the three methodologies provide explicit support for designing teamwork in agents. In addition, the "situatedness" of agents is not fully addressed in such a way that the environment in which the agents operate can be modeled in sufficient detail.

The notation of the three methodologies is generally good. Most of them have a strong modelling language in terms of satisfying various criteria such as clarity and understandability, adequacy and expressiveness, ease of use, and unambiguity.

In addition, only Prometheus and MaSE provide techniques and tools for maintaining the consistency and traceability between models. For the other three methodologies, there is still more room for improvement with respect to these issues. It is also emphasized that none of the evaluated methodologies explicitly provide techniques, guidelines, or models to encourage the design of reusable components or the reuse of existing components.

Regarding the process, only Prometheus and MaSE provide examples and heuristics to assist developers from requirements gathering to detailed design. Neither Roadmap support detailed

design. Additionally, even though all phases from early requirements to implementation are mentioned in Tropos with examples given, the methodology does not appear to provide heuristics for any phase. Implementation, testing/debugging and maintenance are not clearly well-supported by any methodology.

The three methodologies also share some common features. For instance, all of them, except GAIA, provide mechanisms to identify and structure the system's goals as well as agents' goals. There are also similarities in the models provided by the methodologies to represent the static structure and dynamic behavior of the social system as well as individual agents. Distinguishing features also exist in the three methodologies in having an early requirements phase.

MaSE offers a deployment diagram to capture the agent types and their location within a distributed network. Prometheus provides distinctive support for issues relating agent identification. Its data coupling diagram, which we regard as a very useful technique in terms of mapping functionalities to agents, is unique within the three methodologies.

5.3 Recommendations.

Developing and constructing a complete methodology is not an easy task. The Enhanced Multi Agent Software Development (EMASD) methodology is developed as a step towards a comprehensive version. The Enhanced Multi Agent Software Development (EMASD) methodology is constructed based on a well- defined, structured set of aspects that an agent-oriented methodology should include. These aspects are: the entire set of guidelines and activities, a full lifecycle process, a comprehensive set of concepts, modeling techniques, and process. The new proposed The Enhanced Multi Agent Software Development (EMASD) methodology should help to improve the MAS development process. The proposed methodology is to be distinguished from existing methodologies in several aspects:

- 1) The Enhanced Multi Agent Software Development (EMASD) methodology is based on correct concepts where the concepts were selected and chosen to be a solid foundation for the building of the new methodology.
- 2) The Enhanced Multi Agent Software Development (EMASD) methodology supports several important features such as: flexibility, consistency, simplicity, and ease of use as well as

traceability. This is in contrast to the difficulty of understanding and implementing existing methodologies, resulting in a lack of success.

- 3) The proposed methodology covers the fundamental phases as a full software development lifecycle for building systems. The operation starts at the system requirements phase and extends to the implementation phase. Most of the existing methodologies suffer from the problem of incompleteness.
- 4) The proposed methodology covers the most important characteristics of multi-agent systems. It deals with the agent concept as a high-level abstraction capable of modeling the complex system.
- 5) The new methodology incorporates well-known techniques for requirement gathering and customer communication. It goes further by linking them to the domain analysis and design models. It also supports high-level designs and describes the functional requirement of the system from an external perspective.
- 6) The new methodology supports agent organizational aspects. An agent organization is a group of agents working together to achieve a common purpose. It consists of roles that characterize agents. By utilizing these roles, the methodology allows developers to work at different levels of abstraction. Agent behavior can be specified at both the level of roles and at the level of role characteristics.
- 7) The methodology proposes a new concept called the “trigger concept” which is considered as one of the most important characteristics that represent the agent autonomy and reactivity. The existence of the trigger concept plays an important role in determining a larger part of the behavior of the agent.
- 8) The new methodology presents a clear understanding of MASs and how to build them without referring to implementation detail. It sets the distinctive characteristics of MAS as a system that has its own structure and composition.

5.3.1 Areas of Application

In addition, a small **case study** was conducted in which we developed an intelligent Traveller Agent System using the enhanced methodology.

The successful use of the three methodologies in designing the Traveller Agent System indicates that they are practical and usable. However, the maturity of the methodologies is still

a major concern. Regarding the availability of resources supporting the methodologies, most of them are still in the form of conference papers, journal papers or tutorial notes.

None of the methodologies are published as text books. Only Prometheus and MaSE appear to provide strong tool support which is tightly integrated with the methodology's development process.

Additionally, some important software engineering issues, which are much needed in the industry, such as quality assurance, estimating guidelines, and supporting management decisions are not supported by any of the methodologies.

5.3.2 Suggestion for Future work

This section lists several topics that are not addressed in this dissertation. Each topic would clearly benefit from further investigation and, hopefully, would make the EMASD methodology stronger. Candidate topics for future investigation are:

- How to utilize the methodology with special domains such as web-based application, real time systems, etc
- How to perform testing for the resulting agent system software?, and
- How to deal with issues related to agent project management, such as: metrics, estimation, schedule, risk and quality?
- How to deal with the agent mobility?

5.4 Contribution to Knowledge

The major contribution of this work to knowledge was the design of an enhanced model for development of Intelligent Agents which can more effectively do the following: identify their strengths, weaknesses, similarities and differences among the 3 selected agent oriented methodologies.

References

- Allan, W., & Kurt, C. (2012). *A framework for evaluating software technology*. IEEE Computer Society Press. Pages 129-148.
- Allan, M., & Wallnau, T. (2012). *The Rational Unified Process: An Introduction*. Addison- Wesley Pub Co.
- Arazy, O., & Woo, C. (2012.). Analysis and design of agent-oriented information systems. *The Knowledge Engineering Review*, 17(2). Pp. 109-118.
- Ardis, A., John A., & James V. (2012). A framework for evaluating specification methods for reactive systems: Experience report. *In International Conference on Software Engineering*, Pp. 159-168.
- Avison, D., & Fitzgerald, G. (2013). *Information Systems Development: Methodologies, Techniques and Tools*. McGraw-Hill: New York.
- Avison, D., & Fitzgerald, G. (2011). *Information Systems Development: Agent Oriented Methodologies, Techniques and Tools*. McGraw-Hill: New York.
- Barbara, A., Drogoul, A., & Benhamou, P. (2016). Agent-oriented design of a soccer robot team. In Proceedings of the Second International Conference on *Multi-Agent Systems*. Menlo Park, CA: American Association for Artificial Intelligence. Pages. 15-28.
- Barbara K., (2012). DESMET: a method for evaluating software engineering methods and tools. *Technical Report TR96-09*, University of Keele, U.K.Pp. 115-128.
- Bauer, B., & Odell, J. (2005). UML 2.0 and agents: how to build agent-based systems with the new UML standard, *Journal of Engineering Applications of Artificial Intelligence* Vol. 18, Issue 2, 2005. Pages. 125-138.
- Belina, F., Hofgreffe, D. & Sarma, A.,(2011). *SDL with Applications from Protocol Specification*”, Prentice Hall Int., Hertfordshire, UK, 1991.
- Behling, K., (2010). *Project Management – Theory and Practice*. McGraw-Hill professional.
- Berard E. (2012). A comparison of object-oriented methodologies. *Technical report*, Object Agency Inc. Pp. 115-128
- Bernon, C., & Glize, P. (2012). *The ADELFE methodology for an intranet system design*. Washington, DC: Cato Institute.

- Bobkowska, A. (2005). *Framework for methodologies of visual modeling language evaluation*, Proceedings of the symposia on Met informatics, ACM Press 2005.Pp. 115-128
- Braubach, L., Pokahr, A., Moldt, D. and Lamersdorf W.(2005).*Goal representation for BDI agentsystems*”, In R. Bordini, M. Dastani, J. Dix . and A. El Fallah Seghrouchni, editors,Programming Multi-Agent Systems, second Int. Workshop (ProMAS’04), vol. 3346 of LNAI, Pages 44–65. Springer Verlag.
- Brazier, F., Jonker, C. & Treur, J. (2010).*Principles of compositional multi-agent system development*”, In Proceedings of Conference on Information Technology and Knowledge Systems, Pages 347–360. Austrian Computer Society.
- Bresciani, P.,Giorgin,N., Hiunchiglia, R., Mylopoulos,K., & Perini, A. (2014).*Tropos: An agent-oriented software development methodology*. AutonomousAgents and Multi-Agent Systems.
- Bresciani, P., & Perini, A. (2010).*Tropos: An agent-oriented software development methodology*. AutonomousAgents and Multi-Agent Systems.
- Buhr, R.,(2008). *Use Case Maps as Architectural Entities for Complex Systems*, IEEE Transactions on Software Engineering. Vol. 24, No. 12, Pages 1131-1155, 2008.
- Buhr, R., & Casselman, R.(2011).*Use Case Maps for Object-Oriented Systems*. Prentice- Hall, USA.
- Burmeister, B.(2010). *Models and Methodology for Agent-Oriented Analysis and Design*; In Working Notes of the KI’96 Workshop on Agent-Oriented Programming and Distributed Systems, Saarbrilcken, Germany.
- Booch, G. (2014). *Object-oriented analysis and design*. Redwood City, CA: The Benjamin/Cummings Publishing Company, Inc.
- Booch, G. Rumbaugh, J & Jacobson.I. (2014). *The Unified Modeling Language User Guide*. Addison Wesley.
- Burrafato, P. & Cossentino, M. (2012). *Designing a multi-agent solution for a bookstore with the PASSI methodology*. In P. Giorgini, Y. Lespérance, G. Wagner & E. Yu (Eds.), Proceedings of the Agent-Oriented InformationSystems.
- Burrafato, P., & Cossentino, M. (2012). Designing a multi-agent solution for a bookstore with the PASSI methodology. In P. Giorgini, Y. Lespérance, G. Wagner & E. Yu (Eds.), Proceedings of the *Agent-Oriented Information Systems* .Pp. 135-158.

- Burmeister, F., & Cossentino, M, (2011). Designing a multi-agent solution for a bookstore with the PASSI methodology. In Fourth International Bi-Conference Workshop on *Agent-Oriented Information Systems* (AOIS-2011), Pp. 135-158. Toronto (Ontario, Canada).
- Bush, G., Stephen, C., & Martin P (2011). The Styx agent methodology. *The Information Science Discussion Paper Series 2012*, Pp. 15-28. Department of Information Science, University of Otago, New Zealand.
- Brain, J., & Odell, J. (2010.). *Object-oriented methods: Pragmatics and considerations*. Upper Saddle River, NJ: Prentice-Hall .
- Caire, G., & Leal, F. (2012): Recommendations on supporting tools. *Technical Information Final version, European Institute for Research and Strategic Studies in Telecommunications (EURESCOM), July 2012*.
- Caire, G., & Massonet, P. (2011). Agent-oriented analysis using MESSAGE/UML. In M. Wooldridge, G. Wei, & P. Ciancarini (Eds.), *Agent-oriented software engineering II*.
- Castro, J., Kolp, M., & Mylopoulos, J. (2010). A requirements-driven development methodology. In In Proceedings of the 13th International Conference on *Advanced Information Systems Engineering (CAiSE'01)*, Pp. 155-178. Interlaken, Switzerland.
- Castro, J., Kolp, M., & Mylopoulos, J. (2012). Towards requirements-driven information systems engineering: *The Tropos project in Information Systems*.
- Cavedon, L., & Sonenberg, L. (1998). On social commitment, roles and preferred goals. In Proceedings of the Third International Conference on *Multi-Agent Systems (ICMAS)*, Paris. Pp. 105-110. IEEE Computer Society.
- Cavedon, L., & Sonenberg, L. (2012). On social commitment, roles and preferred goals. In Proceedings of the Third International Conference on *Multi-Agent Systems (ICMAS)*, Paris. Pp. 115-120. IEEE Computer Society.
- Cernuzzi, L., & Rossi, G. (2012). *On the evaluation of agent oriented modeling methods*. In Proceedings of Agent Oriented Methodology Workshop, Seattle. Pp. 105-110
- Cossentino, M., & Potts, C. (2012). A case tool supported methodology for the design of multi-agent systems. In The 2002 International Conference on *Software Engineering Research and Practice (SERP'02)*, Las Vegas (NV), USA.
- Cernuzzi, L., & Rossi, G. (2002). On the evaluation of agent oriented methodologies. In Proceedings of OOPSLA 2002 Workshop on *Agent-Oriented Methodologies*. Sydney, AUS: Centre for Object Technology Applications and Research.

- Cernuzzi, L., & Rossi, G. (2002). On the evaluation of agent oriented methodologies. In Proceedings of OOPSLA 2002 Workshop on *Agent-Oriented Methodologies*. Sydney, AUS: Pp. 105-110. Centre for Object Technology Applications and Research.
- Collinot, C., & Treur, J. (2010). Deliberate normative agents: Principles and architectures. In N. Jennings & Y. Lespérance (Eds.), *Intelligent agents VI Berlin*: Pp. 115-125 Springer-Verlag.
- Chan, K., & Karunasekera, S. (2004). Agent-oriented software analysis. In Proceedings of 2004 Australian *Software Engineering Conference*(pp. 20-27). Los Alamitos, CA: IEEE Computer Society Press.
- Chris, S., & Barbara, A. (2011). Evaluating software engineering methods and tools-part 4: the influence of human factors. ACM SIGSOFT Software Engineering Notes.
- David, L., & Michael W. (2011). Debugging multi-agent systems using design artifacts: The case of interaction protocols. In Proceedings of the First International Joint Conference on *Autonomous Agents and Multi Agent Systems*.
- David,L.(2012).*Agent-Oriented Information Systems*. Redwood City, CA: The Benjamin/Cummings Publishing Company, Inc.
- Debenham, J., & Henderson-Sellers, B.(2010). Full lifecycle methodologies for agent-oriented systems the extended OPEN process framework. In Proceedings of *Agent-Oriented Information Systems*, Toronto.
- DeLoach, S.(2012). Multiagent systems engineering: A methodology and language for designing agent systems. In *Agent-Oriented Information Systems '99 (AOIS'99)*, Seattle WA.
- DeLoach, S. (2012). Analysis and design using MaSE and agentTool. In Proceedings of the 12th Midwest *Artificial Intelligence and Cognitive Science Conference (MAICS 2001)*, 2012.
- Dumke, R.(2011). Metrics-based evaluation of object-oriented software development methods. In Proceedings of the 2nd Euromicro Conference on *Software Maintenance and Reengineering (CSMR'98)*, pages 193-196, Florence, Italy.
- E. Yu (2011). *Modelling Strategic Relationships for Process Reengineering*, University of Toronto, Department of Computer Science, 2011.
- Eckert, G. (2010). *Improving object-oriented analysis*. Information and Software Technology.
- Elammari, M., & Lalonde, W.(2010). “An Agent-Oriented Methodology: High-Level and Intermediate Models”, *HLIM*, Proceedings of AOIS Heidelberg.Pp. 105-110

- Frank, U. (2012). A comparison of two outstanding methodologies for object-oriented design. *Technical Report*.
- Frank, U. (2012). Evaluating modeling languages: relevant issues, epistemological challenges and a preliminary research framework. *Technical Report 15*, 2012.
- Genesereth, C., & Nelson, T. (2011). The importance of dealing with uncertainty in the evaluation of Software engineering methods and tools. In Proceedings of the 14th international Conference on *Software engineering and knowledge engineering*, ACM Press.
- Glaser, C., & Francisco, L. (2010). Agent oriented analysis using MESSAGE/UML. In Michael W., Paolo, C., & Gerhard, W., editors, Second International Workshop on *Agent-Oriented Software Engineering (AOSE-2012)*.
- Hans –Van, V.(2012). A CASE tool supported methodology for the design of multi-agent systems. In H.R. Ababnia & Y. Mun (Eds.), Proceedings of the 2012 International Conference on Software Engineering Research and Practice (SERP'12), Las Vegas.
- Hans-Van, V. (2000). *Software Engineering: Principles and Practice*. John Wiley & Sons, second edition.IEEE Std 610.12. *IEEE Standard Glossary of Software Engineering Terminology*, p.77.
- Iglesias,S.(2012). Evaluating modelling languages: relevant issues, epistemological challenges and a preliminary research framework. *Technical Report 15*, University of Koblenz-Landau.
- James, O.(2012). Objects and agents compared. *Journal of Object Technology*. Toronto
- Jayaratna,N.(2012). *Understanding and evaluating methodologies, NISAD: A systematic framework”*, Maidenhead, UK: McGraw-Hill.
- Jennings, N., Sycara, K., & Wooldridge, M.(2012). A Roadmap of Agent Research and Development; *In Autonomous Agents and Multi-Agent Systems Journal*, Publishers, Boston.
- Jennings, N., & Wooldridge, M.(2012). *Agent-Oriented Software Engineering*, in proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World: Multi-Agent System Engineering (MAAMAW-99), vol. 1647, Pp. 105-110. Springer-Verlag: Heidelberg, Germany.
- Juan, T., Pierce, A., & Sterling, L.(2011). *Roadmap: Extending the GAIA methodology for complex open systems”*, In Proceedings of the 1st ACM Joint Conference on Autonomous Agents and Multi-Agent Systems (Bologna, Italy), ACM, New York.

- Juan, T., Sterling, L. and Winikoff, M.(2002).*Assembling Agent-Oriented Software Engineering Methodologies from Features*”, in the Proceedings of the Third International Workshop on Agent-Oriented Software Engineering, at AAMAS’02, Bologna, Italy, 2002.
- Jefrey, M. (2011). An introduction to software agents. In Jeffrey M. Bradshaw, editor, *Software Agents*, pages 3{46. AAAI Press / The MIT Press, 2011.
- John, F. (2010). *Multi-agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley.
- Kendall E.,(1996). *Agent Roles and Role Models: New Abstractions for Multi-agent System Analysis and Design*”, Proceedings of the International Workshop on Intelligent Agents in Information and Process Management, Bremen, Germany, 1998.
- Kendall, E., Malkoun, M., & Jiang, C. (2010).*A Methodology for Developing Agent Based Systems*”, In Distributed Artificial Intelligence Architecture and Modeling, LNAI 1087.Springer-Verlag, Pages 85-99, Germany.
- Khanh, H., & Michael W.(2010) . *Comparing agent-oriented methodologies*. In To appear at the International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2010), Melbourne, Australia.
- Kruchten, P.(2000).*The Rational Unified Process: An Introduction*. Addison-Wesley Pub Co.
- Krupansky, J. (2010).*Foundations of Software Agent Technology*”, Activity: Advancing the Science of Software Agent Technology.
- Kinny, T., Georgeff, B., & Rao, I.(1996). *Desire: Modeling Multi-Agent Systems in a Compositional Formal Framework*, Int. Journal of Cooperative Information Systems, Vol. 6. Special Issue on Formal Methods in Cooperative Information Systems: Multi-agent Systems.
- Law,D., & Naem,A.(2013). *Methods for Comparing Methods: Techniques in Software Development*. NCC Publications
- Lewis, R.(2014).Project Management. McGraw-Hill professional
- Lin, P., & Michael, W., (2011). Prometheus: A methodology for developing intelligent agents. In Third International Workshop on *Agent-Oriented Software Engineering*, July 2011.

- Lin, P., & Michael, W., (2011). Prometheus: A pragmatic methodology for engineering intelligent agents. In Proceedings of the OOPSLA 2002 Workshop on *Agent-Oriented Methodologies*, pages 97-108, Seattle, November 2011.
- Lin, P., & Michael, W., (2011). Prometheus: Engineering intelligent agents. Tutorial notes, available from the authors, October 2011.
- Lin, P., & Winkoff, A., (2012). *Prometheus: A brief summary*. Technical note.
- Lin, P., & Michael W.(2012). *Prometheus: A methodology for developing intelligent agents*. McGraw-Hill: New York.
- Lind, C., Erik, A., Steve J., Frank H., & Pascale, T. (2011). *Empirical studies of object-oriented artifacts, methods, and processes: State of the art and future directions*. Empirical Software Engineering.
- Maes, J.(2011). *Understanding and Evaluating Methodologies: NIMSAD a Systematic Framework*. McGraw-Hill, New York, 2nd edition.
- Mark, W., & DeLoach. S. (2012). An overview of the multiagent systems engineering methodology. In The First International Workshop on *Agent-Oriented Software Engineering*, Limerick, Ireland.
- Michael, P. (2012). Evaluation of object-oriented modeling languages: A comparison between OML and UML. In Martin Schader and Axel Korthaus, editors, *The Unified Modeling Language Technical Aspects and Applications*., Physical-Verlag, Heidelberg.
- Mike, F. (2012). The Tropos software development methodology: Processes, Models and Diagrams. In Third International Workshop on *Agent-Oriented Software Engineering*.
- Moulin, A.(1994). Multiagent systems engineering. *International Journal of Software Engineering and Knowledge Engineering*.
- Nicholas, A., & Wooldridge, M.(2012). *An Introduction to Multi-Agent Systems*. John Wiley & Sons, 2002.
- Nwana, H. (2012). *Software agents: An overview*. Knowledge Engineering Review. Toronto
- Odell, S. (2012). Requirements of an object-oriented design method. *Software Engineering Journal*, pages 102-113, March 2012.

- O'Malley, S.(2011). Determining when to use an agent-oriented software engineering methodology. In Proceedings of the Second International Workshop On *Agent-Oriented Software Engineering (AOSE-2011)*.
- Omicini, J., Parunak H., & Bauer B. (2012). *Representing Agent Interaction Protocols in UML*. The First International Workshop on Agent-Oriented Software Engineering.
- Padgham, .J., & Winikoff, A. (2012). *A cognitive foundation for comparing object-oriented analysis methods*. In J. F Nunamaker and IEEE Computer Society Press: R. H Sprague, editors, 26th Hawaii International Conference on System Sciences.
- Padgham, J., & Michael, I.(2012). *Agent Oriented Methodology*; Addison Wesley.
- Parson, C., Jazayeri, M., Mandrioli, D.(2011): *Fundamentals of Software Engineering*. PrenticeHall, Englewood Cliffs, N. J. ,pp.12,14.
- Pollack, B., Fausto, M., & Anna P. (2010). *Troops: An agent-oriented software development methodology*. Technical Report DIT-02-0015, University of Trento, Department of Information and Communication Technology.
- Rao, A.(2010). *A methodology and modeling technique for systems of BDI agents*, In Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi- Agent World, LNAI Vol. 1038, Springer-Verlag, Berlin.
- Roel, W. (2011). *A survey of structured and object-oriented software specification methods and techniques*. ACM Computing Surveys.
- Rumbaugh, E.(2011). A comparison of object-oriented methodologies. *Technical report*, Object Agency Inc.
- Russel,M., & Norvig.(2011). Determining when to use an agent-oriented software engineering methodology. McGraw-Hill, New York.
- Sabas,M., Badri,O., & Delisle,M.(2002). *The GAIA methodology for agent-oriented analysis and design*. Autonomous Agents and Multi-Agent Systems.
- Scott, A.(2011). Specifying agent behavior as concurrent tasks: Designing the behavior of social agents. In Proceedings of the Fifth Annual Conference on *Autonomous Agents*, Montreal Canada.
- Scott, A. (2012). *Applying agent oriented software engineering to cooperative robotics*. University of Toronto, Department of Computer Science.
- Scott, A. (2012). Multi agent Systems Engineering. *International Journal of Software Engineering and Knowledge Engineering*.

- Sharble, R., & Cohen, S. (2012). *The object-oriented brewery: A comparison of two object oriented development methods*. SIGSOFT Software Engineering Notes.
- Sturm, A., & Shehory, O. (2012). Towards industrially applicable modeling technique for agent-based systems (poster). In Proceedings of International Conference on *Autonomous Agents and Multi-Agent Systems*, Bologna.
- Tambe, M., & Jennings, R. (2012). *Applications of intelligent agents. Agent Technology: Foundations, Applications, and Markets*
- Trans, O., & Law, M. (2010). *Understanding and Evaluating Methodologies*: McGraw- Hill, New York.
- Vossen, G., & Dignum, A. (2012). Agent-Object-Relationship Modeling. In Proc. of Second International Symposium - from *Agent Theory to Agent Implementation together with EMCRS 2012*, April 2012.
- Wagner, G., (2011). Agent-Oriented Analysis and Design of Organizational Information Systems. In Proc. of Fourth IEEE International Baltic Workshop on *Databases and Information Systems*, Vilnius (Lithuania), May 2011.
- Wood, R. (2012). *Developing Intelligent Agent Systems*. John Wiley & Sons, Ltd.
- Wood, R., Pethia, L. Gold, A., & Firth, D. (2012). *A guide to the assessment of software development methods. Technical Report 88-TR-8*, Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, PA
- Wooldridge, M., & Jennings, R. (2010). *A guide to the assessment of software development methods. Technical Report 88-TR-8*, Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, PA, 2010.
- Wooldridge, M., & Jennings, R. (2014). *Intelligent agents: Theory and practice*. The Knowledge Engineering Review.
- Wooldridge, M., Jennings, R., & Kinny, D. (2011). *A methodology for agent-oriented analysis and design*. In Proceedings of the third international conference on Autonomous Agents.
- Wooldridge, M., Jennings, R., & Kinny, D. (2012). *The GAIA methodology for agent-oriented analysis and design*. Autonomous Agents and Multi-Agent Systems.

APPENDIX A: SAMPLE PROGRAM CODE

```
<!DOCTYPE html>
<html lang="en"><head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<meta charset="utf-8">
<title>TRAVELER AGENT SYSTEM</title>
<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">
<meta name="format-detection" content="telephone=no">
<meta name="apple-mobile-web-app-capable" content="yes">
<link href="css/css.css" rel="stylesheet" type="text/css">
<link href="css/css_002.css" rel="stylesheet" type="text/css">
<link href="css/css_003.css" rel="stylesheet" type="text/css">
<link rel="stylesheet" type="text/css" href="css/bootstrap.css">
<link rel="stylesheet" type="text/css" href="css/font-awesome.css">
<link rel="stylesheet" type="text/css" href="css/awe-booking-font.css">
<link rel="stylesheet" type="text/css" href="css/owl.css">
<link rel="stylesheet" href="font-awesome/css/font-awesome.min.css" type="text/css">
<link rel="stylesheet" type="text/css" href="css/magnific-popup.css">
<link rel="stylesheet" href="css/animate.css">
<link rel="stylesheet" type="text/css" href="css/jquery-ui.css">
<link rel="stylesheet" type="text/css" href="css/settings.css">
<link rel="stylesheet" type="text/css" href="css/animate.min.css">
<link rel="stylesheet" type="text/css" href="css/style.css">
<link rel="stylesheet" type="text/css" href="css/demo.css">
<link id="colorreplace" rel="stylesheet" type="text/css" href="css/blue.css">
<script src="js/analytics.js" async=""></script>
<script src="js/fbevents.js" async=""></script>
<script>

        !function(f,b,e,v,n,t,s){if(f.fbq)return;n=f.fbq=function(){n.callMethod?
allMethod?

        n.callMethod.apply(n,arguments):n.queue.push(arguments)};if(!f._fbq)f._fbq=n;

        n.push=n;n.loaded=!0;n.version='2.0';n.queue=[];t=b.createElement(e);t.async=!0;

        t.src=v;s=b.getElementsByTagName(e)[0];s.parentNode.insertBefore(t,s)}(window,

        document,'script','//connect.facebook.net/en_US/fbevents.js');

        fbq('init', '1031554816897182');
        fbq('track', "PageView");</script>
<noscript></noscript>
<style>
.theiaStickySidebar:after {
  content: "";
  display: table;
  clear: both;
}
</style>
</head>
<body>
<divid="page-wrap">
<divstyle="display: none;" class="preloader"></div>
<headerid="header-page">
  <divstyle="transform: translateY(0px);" class="header-page__inner header-page__fixed">
    <divclass="container"><divclass="logo">
      <a href="#">
        <h5class="animated pulse">TRAVELER AGENT
SYSTEM</h5>
      </a>
    </div>
    <navstyle="height: auto;" class="navigation awe-n
navigation" data-responsive="1200">
      <ulclass="menu-list">
        <liclass="menu-item-has-children current-
menu-parent">
          <a href="index.php">Home</a>
        </li>
        <liclass="menu-item-has-children">
          <a href="hotel.jade">Hotel</a>
        </li>
        <liclass="menu-item-has-children">
          <a href="bus.jade">Bus</a>
        </li>
        <liclass="menu-item-has-children">
          <a href="flit.jade">Flight</a>
        </li>
        <liclass="menu-item-has-children">
          <a href="train.jade">Train</a>
        </li>
        <liclass="menu-item-has-children
pull-right">
          <a href="#"><iclass="awe-icon fa fa-
arrow-down"></i></a>

```

```

        <ulclass="sub-menu">
            <li>
                <a href="register.php"
aria-labelledby="ui-id-5">Register</a>
            </li>
        </ul>
    </li>
</ul>
</nav>
</div>
</header>

<!--slide_caurasoul-->
<divid="carousel-example-generic"
class="carousel slide" data-ride="carousel">
    <!-- Indicators -->
    <olclass="carousel-indicators">

    </ol>

    <!-- Wrapper for slides -->
    <divclass="carousel-inner"
role="listbox">
        <divclass="item active">
            <imgsrc="images/bg/1.jpg" alt="...">
            <divclass="carousel-caption">
                ...
            </div>
        </div>
        <divclass="item">
            <imgsrc="images/bg/2.jpg" alt="...">
            <divclass="carousel-caption">
                ...
            </div>
        </div>
        <divclass="item">
            <imgsrc="images/bg/3.jpg" alt="...">
            <divclass="carousel-caption">
                ...
            </div>
        </div>
        ...
    </div>

    <!-- Controls -->

```

```

        <a class="left carousel-control"
href="#carousel-example-generic" role="button" data-slide="prev">
        <span class="glyphicon glyphicon-
chevron-left" aria-hidden="true"></span>
        <span class="sr-only">Previous</span>
        </a>
        <a class="right carousel-control"
href="#carousel-example-generic" role="button" data-slide="next">
        <span class="glyphicon glyphicon-
chevron-right" aria-hidden="true"></span>
        <span class="sr-only">Next</span>
        </a>
    </div>

```

```

<section>
    <div class="container" style="opacity:0.9;">
        <div class="awe-search-tabs tabs ui-tabs ui-widget ui-
widget-content ui-corner-all">
            <ul role="tablist" class="ui-tabs-nav ui-helper-
reset ui-helper-clearfix ui-widget-header ui-corner-all">
                <li aria-selected="true" aria-
labelledby="ui-id-1" aria-controls="awe-search-tabs-1" tabindex="0"
role="tab" class="ui-state-default ui-corner-top ui-tabs-active ui-
state-active">
                    <a id="ui-id-1" tabindex="-1"
role="presentation" class="ui-tabs-anchor" href="#awe-search-tabs-1">
                        <i class="awe-icon fa fa-
plane"></i>
                    </a>
                </li>
                <li aria-selected="false" aria-
labelledby="ui-id-2" aria-controls="awe-search-tabs-2" tabindex="-1"
role="tab" class="ui-state-default ui-corner-top">
                    <a id="ui-id-2" tabindex="-1"
role="presentation" class="ui-tabs-anchor" href="#awe-search-tabs-2">
                        <i class="awe-icon fa fa-
hotel"></i>
                    </a>
                </li>
                <li aria-selected="false" aria-
labelledby="ui-id-3" aria-controls="awe-search-tabs-3" tabindex="-1"
role="tab" class="ui-state-default ui-corner-top">
                    <a id="ui-id-3" tabindex="-1"
role="presentation" class="ui-tabs-anchor" href="#awe-search-tabs-3">
                        <i class="awe-icon fa fa-
bus"></i>
                    </a>
                </li>
            </ul>

```



```
</div>
<divclass="form-elements">

    <label>To</label>
    <divclass="form-item">
    <divclass="awe-select-wrapper">

        <selectclass="awe-select" name="to">

            <option>Nigeria</option>
<option>Albania</option>
<option>Andorra</option>
<option>Antigua</option>
<option>Argentina</option>
<option>Armenia</option>
<option>Aruba</option>
<option>Australia</option>
<option>Austria</option>
<option>Azerbaijan</option>
<option>Bahamas</option>
<option>Bahrain</option>
<option>Barbados</option>
<option>Belarus</option>
<option>Belgium</option>
<option>Benin</option>
<option>Bermuda</option>
<option>Bolivia</option>
<option>Bosnia Herzegovina</option>
<option>Botswana</option>
<option>Brazil</option>
<option>British Virgin Islands</option>
<option>Brunei Darussalam</option>
<option>Bulgaria</option>
<option>Cambodia</option>
<option>Cameroon</option>
<option>Canada</option>
<option>Cayman Islands</option>
<option>Chile</option>
<option>China</option>
<option>Colombia</option>
<option>Congo</option>
<option>Cook Islands</option>
<option>Costa Rica</option>
<option>Cote d Ivoire</option>
<option>Croatia</option>
<option>Cuba</option>
<option>Cyprus</option>
<option>Czech Republic</option>
<option>Denmark</option>
<option>Dominican Republic</option>
<option>Ecuador</option>
<option>Egypt</option>
```

<option>Estonia</option>
<option>Ethiopia</option>
<option>Fiji</option>
<option>Finland</option>
<option>France</option>
<option>French Polynesia</option>
<option>Gabon</option>
<option>Gambia</option>
<option>Georgia</option>
<option>Germany</option>
<option>Ghana</option>
<option>Gibraltar</option>
<option>Greece</option>
<option>Greenland</option>
<option>Grenada</option>
<option>Guadeloupe</option>
<option>Guam</option>
<option>Guatemala</option>
<option>Haiti</option>
<option>Honduras</option>
<option>Hong Kong</option>
<option>Hungary</option>
<option>Iceland</option>
<option>India</option>
<option>Indonesia</option>
<option>Iran</option>
<option>Iraq</option>
<option>Ireland (Republic of)</option>
<option>Israel</option>
<option>Italy</option>
<option>Jamaica</option>
<option>Japan</option>
<option>Jordan</option>
<option>Kazakhstan</option>
<option>Kenya</option>
<option>Kuwait</option>
<option>Kyrgyzstan</option>
<option>Laos</option>
<option>Latvia</option>
<option>Lebanon</option>
<option>Libya</option>
<option>Liechtenstein</option>
<option>Lithuania</option>
<option>Luxembourg</option>
<option>Macau</option>
<option>Macedonia</option>
<option>Malaysia</option>
<option>Maldives</option>
<option>Mali</option>
<option>Malta</option>
<option>Mauritius</option>
<option>Mexico</option>

<option>Micronesia</option>
<option>Moldova</option>
<option>Monaco</option>
<option>Mongolia</option>
<option>Morocco</option>
<option>Mozambique</option>
<option>Myanmar</option>
<option>Namibia</option>
<option>Nepal</option>
<option>Netherlands</option>
<option>Netherlands Antilles</option>
<option>New Caledonia</option>
<option>New Zealand</option>
<option>Nicaragua</option>
<option>Norfolk Island</option>
<option>North Korea</option>
<option>Northern Mariana Island</option>
<option>Norway</option>
<option>Oman</option>
<option>Pakistan</option>
<option>Palau</option>
<option>Panama</option>
<option>Paraguay</option>
<option>Peru</option>
<option>Philippines</option>
<option>Poland</option>
<option>Portugal</option>
<option>Puerto Rico</option>
<option>Qatar</option>
<option>Romania</option>
<option>Russia</option>
<option>Rwanda</option>
<option>Saint Barthelemy</option>
<option>Samoa</option>
<option>San Marino</option>
<option>San Martin (F)</option>
<option>Saudi Arabia</option>
<option>Senegal</option>
<option>Serbia and Montenegro</option>
<option>Seychelles</option>
<option>Singapore</option>
<option>Slovakia</option>
<option>Slovenia</option>
<option>South Africa</option>
<option>South Korea</option>
<option>Spain</option>
<option>Sri Lanka</option>
<option>St Kitts and Nevis</option>
<option>St Lucia</option>
<option>St Vincent and Grenadines</option>
<option>Swaziland</option>
<option>Sweden</option>

```

<option>Switzerland</option>
<option>Syria</option>
<option>Tadjikistan</option>
<option>Taiwan</option>
<option>Tanzania</option>
<option>Thailand</option>
<option>Togo</option>
<option>Tonga</option>
<option>Trinidad and Tobago</option>
<option>Tunisia</option>
<option>Turkey</option>
<option>Turkmenistan</option>
<option>Turks and Caicos Island</option>
<option>Uganda</option>
<option>Ukraine</option>
<option>United Arab Emirates</option>
<option]>United Kingdom</option>
<option>United States</option>
<option>Uruguay</option>
<option>Uzbekistan</option>
<option>Vanuatu</option>
<option>Venezuela</option>
<option>Vietnam</option>
<option>Virgin Islands (USA)</option>
<option>Yemen Republic</option>
<option>Zaire</option>
<option>Zambia</option>
<option>Zimbabwe</option>

```

```

</select>

```

```

<iclass="fa fa-caret-down"></i>

```

```

</div>

```

```

</div>

```

```

</div>

```

```

</div>

```

```

<divclass="form-group">

```

```

  <divclass="form-elements">

```

```

    <label>Budget</label>

```

```

  <divclass="form-item">

```

```

    <divclass="awe-select-wrapper">

```

```

      <selectclass="awe-select">

```

```

        <option>N100,000</option>

```

```

        <option>N150,000</option>

```

```

        <option>N200,000</option>

```

```

        <option>N350,000</option>

```

```

        <option>N400,000</option>
        <option>N450,000</option>
        <option>N500,000</option>
        <option>N650,000</option>
        <option>N750,000</option>
        <option>N840,000</option>
        <option>N850,000</option>
        <option>N1000,0000</option>
        <option>N1500,0000</option>
    </select>

    <i>class="fa fa-caret-down"</i>
</div>
</div>
</div>
<divclass="form-elements">
<label>Flight period</label>
<divclass="form-item">

    <divclass="awe-select-wrapper">
    <selectclass="awe-select">

</select>
    <i>class="fa fa-caret-down"</i>

</div>
</div>
</div>
</div>
<divclass="form-actions">
<inputvalue="Find My Flight" name="search" type="submit">
</div>
    </form>
</div>
<divaria-hidden="true" aria-expanded="false" style="display: none;"
role="tabpanel" aria-labelledby="ui-id-2" id="awe-search-tabs-2"
class="search-hotel ui-tabs-panel ui-widget-content ui-corner-bottom">
<h2>Where would you like to go?</h2>
<form>
<divclass="form-group">

```

```

<divclass="form-elements">
<label>Destinations</label>
<divclass="form-item">
<iclass="awe-icon"></i>
<inputvalue="Country, city, airport..." type="text">
</div>
</div>
</div>
<divclass="form-group">
<divclass="form-elements">
<label>Check in</label>
<divclass="form-item">
<iclass="awe-icon fa fa-calendar"></i>
<inputid="dp1478112976534" class="awe-calendar hasDatepicker"
value="Date" type="text">
</div>
</div>
<divclass="form-elements">
<label>Check out</label>
<divclass="form-item">
<iclass="awe-icon fa fa-calendar"></i>
<inputid="dp1478112976535" class="awe-calendar hasDatepicker"
value="Date" type="text">
</div>
</div>
</div>
<divclass="form-group">
<divclass="form-elements">
<label>Guest</label>
<divclass="form-item">
<divclass="awe-select-wrapper">
<selectclass="awe-select">
<optionselected="selected">All types</option>
<option>1</option>
<option>2</option>
<option>3</option>
</select>
<iclass="fa fa-caret-down"></i>
</div>
</div>
</div>
</div>
<divclass="form-actions">
<inputvalue="Find My Hotel" type="submit">
</div>
</form>
</div>
<divaria-hidden="true" aria-expanded="false" style="display: none;"
role="tabpanel" aria-labelledby="ui-id-3" id="awe-search-tabs-3"
class="search-flight ui-tabs-panel ui-widget-content ui-corner-
bottom">
<h2>Search Flight</h2>

```

```

<form>
<divclass="form-group">
<divclass="form-elements">
<label>From</label>
<divclass="form-item">
<iclass="awe-icon"></i>
<inputvalue="Ho Chi Minh, Hanoi, Vietnam" type="text">
</div>
</div>
<divclass="form-elements">
<label>To</label>
<divclass="form-item">
<iclass="awe-icon"></i>
<inputvalue="Ankara, Turkey" type="text">
</div>
</div>
</div>
<divclass="form-group">
<divclass="form-elements">
<label>Depart on</label>
<divclass="form-item">
<iclass="awe-icon fa fa-calendar"></i>
<inputid="dp1478112976536" class="awe-calendar hasDatepicker"
value="Check in" type="text">
</div>
</div>
<divclass="form-elements">
<label>Return on</label>
<divclass="form-item">
<iclass="awe-icon fa fa-calendar"></i>

</div>
</div>
<divclass="form-elements">
<label>Adult</label>
<divclass="form-item">
<divclass="awe-select-wrapper">
</div>
<span>12 yo and above</span>
</div>
<divclass="form-elements">

<label>Kids</label>

<divclass="form-item">
<divclass="awe-select-wrapper">

</div>
</div>

<span>0-11 yo</span>
</div>

```

```

</div>
<divclass="form-group">
<divclass="form-elements">

    <label>Budget</label>

    <divclass="form-item">
    <divclass="awe-select-wrapper">
    </div>
    </div>
    </div>
    <divclass="form-actions">
    <inputvalue="Find My Flight" type="submit">
    </div>
    </form>
    </div>
    <divaria-hidden="true" aria-expanded="false" style="display:
none;" role="tabpanel" aria-labelledby="ui-id-4" id="awe-search-tabs-
4" class="search-flight ui-tabs-panel ui-widget-content ui-corner-
bottom">
    <h2>Search Train</h2>
    <form>
    <divclass="form-group">
    <divclass="form-elements">
    <label>From</label>

    <divclass="form-item">
    <iclass="awe-icon"></i>
    </div>
    </div>
    <divclass="form-elements">

    <label>To</label>

    <divclass="form-item">
    <iclass="awe-icon"></i>
    <inputvalue="Ankara, Turkey" type="text">

    </div>
    </div>
    </div>
    <divclass="form-group">
    <divclass="form-elements">

    <label>Depart on</label>

    <divclass="form-item">
    <iclass="awe-icon fa fa-calendar"></i>

    </div>
    </div>

```



```
<divclass="form-elements">

<label>Return on</label>

<divclass="form-item">
<iclass="awe-icon fa fa-calendar"></i>

</div>
</div>
<divclass="form-elements">

<label>Adult</label>

<divclass="form-item">
<divclass="awe-select-wrapper">
<optionselected="selected">0</option>
</div>

</div>

<span>12 yo and above</span>
</div>
<divclass="form-elements">

<label>Kids</label>

<divclass="form-item">
<divclass="awe-select-wrapper">

<selectclass="awe-select">
```

```
</div>

</div>

<span>0-11 yo</span>
</div>
</div>
<divclass="form-group">
<divclass="form-elements">

<label>Budget</label>
<divclass="form-item">
<divclass="awe-select-wrapper">

<selectclass="awe-select">
<optionselected="selected">All types</option>

<option>1</option>

<option>2</option>

<option>3</option>

</select>

<iclass="fa fa-caret-down"></i>
</div>
</div>
</div>
</div>
<divclass="form-actions">
<inputvalue="Find My Flight" type="submit">
</div>
</form>
</div>
```

```

<div aria-hidden="true" aria-expanded="false" style="display: none;"
role="tabpanel" aria-labelledby="ui-id-5" id="awe-search-tabs-5"
class="search-car ui-tabs-panel ui-widget-content ui-corner-bottom">
<h2>User Login</h2>
<form action="index.php" method="POST">
<div class="form-group">
<div class="form-elements">
<label>user name</label>
<div class="form-item">
<i class="awe-icon"></i>

</div>
</div>
<div class="form-elements">
<label>Password</label>
<div class="form-item">
<i class="awe-icon"></i>

</div>
</div>
</div>
<div class="form-actions">
<input value="Login" name="move" type="submit">
</div>
</form>
</div>
<div aria-hidden="true" aria-expanded="false" style="display: none;"
role="tabpanel" aria-labelledby="ui-id-6" id="awe-search-tabs-6"
class="search-bus ui-tabs-panel ui-widget-content ui-corner-bottom">
<h2> Adin Login</h2>
<form action="index.php" method="POST">
<div class="form-group">
<div class="form-elements">
<label>user name</label>
<div class="form-item">
<i class="awe-icon"></i>

</div>
</div>
</div>
<div class="form-elements">

<label>Password</label>

<div class="form-item">
<i class="awe-icon"></i>

</div>
</div>
<div class="form-actions">
<input value="Login" name="admin" type="submit">
</div>

</form>

```

```

                </div>
            </div>
        </div>
    </div>
</section>
<sectionclass="masonry-section-demo">
    <divclass="container">
        <divclass="destination-grid-content">
            <divclass="section-title">
                <h3>
                    Top Hotels
                    <a href="#"></a>
                </h3>
            </div>
            <divclass="row">
                <divstyle="position: relative;
height: 877.5px;" class="awe-masonry item-9">
                    <divstyle="position:
absolute; left: 0px; top: 0px;" class="awe-masonry__item">
                        <a href="hotel.php">
                            <divclass="image-wrap image-cover">
                                <imgstyle="height: 100%; width: auto;" src="img/1_002.jpg" alt="">
                            </div>
                        </a>
                        <divclass="item-title">
                            <h2>
                                <a href="hotel.php">Enugu</a>
                            </h2>
                            <divclass="item-cat">
                                <ul>
                                    <li><a href="hotel.php">Nigeria</a>
                                    </li>
                                </ul>
                            </div>
                            <divclass="item-available">
                                <spanclass="count">845</span> available Rooms
                            </div>
                        </div>
                    <divstyle="position: absolute; left: 292px; top: 0px;" class="awe-
masonry__item">
                        <a href="hotel.php">
                            <divclass="image-wrap image-cover">
                                <imgstyle="height: 100%; width: auto;" src="img/2.jpg" alt="">
                            </div>
                        </a>
                        <divclass="item-title">
                            <h2>
                                <a href="hotel.php">London</a>
                            </h2>
                            <divclass="item-cat">
                                <ul>

```

```

<li>
<a href="hotel.php">England</a>
</li>
</ul>
</div>
</div>
<div class="item-available">
<span class="count">132</span> available Rooms
</div>
</div>
<div style="position: absolute; left: 585px; top: 0px;" class="awe-
masonry__item">
<a href="hotel.php">
<div class="image-wrap image-cover">

</div>
</a>
<div class="item-title">
<h2>
<a href="hotel.php">Lagos</a>
</h2><div class="item-cat">
<ul>
<li>
<a href="hotel.php">Nigeria</a>
</li>
</ul>
</div>
</div>
<div class="item-available">
<span class="count">2341</span> available Rooms
</div>
</div>
<div style="position: absolute; left: 0px; top: 292px;" class="awe-
masonry__item">
<a href="hotel.php">
<div class="image-wrap image-cover">

</div>
</a>
<div class="item-title">
<h2>
<a href="hotel.php">New york</a>
</h2>
<div class="item-cat">
<ul>
<li>
<a href="hotel.php">USA</a>
</li>
</ul>
</div>
</div>
</div>
<div class="item-available">

```

```

<spanclass="count">2531</span> available Rooms
</div>
</div>
<divstyle="position: absolute; left: 292px; top: 292px;" class="awe-
masonry__item">
<a href="hotel.php">
<divclass="image-wrap image-cover">
<imgstyle="height: 100%; width: auto;" src="img/5\_002.jpg" alt="">
</div></a>
<divclass="item-title">
<h2>
<a href="hotel.php">Abuja</a>
</h2>
<divclass="item-cat">
<ul>
<li>
<a href="hotel.php">Nigeria</a>
</li>
</ul>
</div>
</div><divclass="item-available">
<spanclass="count">2531</span> available Rooms
</div>
</div>
<divstyle="position: absolute; left: 0px; top: 585px;" class="awe-
masonry__item"><a href="hotel.php">
<divclass="image-wrap image-cover">
<imgstyle="height: 100%; width: auto;" src="img/6.jpg" alt="">
</div>
</a>
<divclass="item-title">
<h2><a href="hotel.php">Paris</a>
</h2><divclass="item-cat">
<ul>
<li>
<a href="hotel.php">France</a>
</li>
</ul>
</div>
</div>
<divclass="item-available">
<spanclass="count">2531</span> available Rooms
</div>
</div>
<divstyle="position: absolute; left: 292px; top: 585px;" class="awe-
masonry__item">
<a href="hotel.php">
<divclass="image-wrap image-cover">
<imgstyle="height: 100%; width: auto;" src="img/7.gif" alt="">
</div>
</a>
<divclass="item-title">

```

```

<h2><a href="hotel.php">Calabar</a></h2>
<div class="item-cat">
<ul>
<li>
<a href="hotel.php">Nigeria</a>
</li>
</ul>
</div>
</div>
<div class="item-available">
<span class="count">2531</span> available Rooms
</div>
</div>
<div style="position: absolute; left: 585px; top: 585px;" class="awe-
masonry__item">
<a href="hotel.php">
<div class="image-wrap image-cover">

</div>
</a>
<div class="item-title">
<h2>
<a href="hotel.php">Ghana</a>
</h2><div class="item-cat">
<ul>
<li>
<a href="hotel.php">Accra</a>
</li>
</ul>
</div>
</div>
<div class="item-available">
<span class="count">2531</span> available Rooms
</div>
</div>
<div style="position: absolute; left: 877px; top: 585px;" class="awe-
masonry__item">
<a href="hotel.php">
<div class="image-wrap image-cover">

</div>
</a>
<div class="item-title">
<h2>
<a href="hotel.php">Dubia</a>
</h2>
<div class="item-cat">
<ul>
<li>
<a href="hotel.php">Arab Emirates</a>
</li>
</ul>

```

```

</div>
</div>
<divclass="item-available">
<spanclass="count">2531</span> available Rooms
</div>
</div>
</div>
</div>
<divclass="more-destination">
<a href="hotel.php">More Hotels</a>
</div>
</div>
</div>
</section>

<sectionclass="sale-flights-section-demo">
<divclass="container">
<divclass="row">
<divclass="col-md-8">
<divclass="sale-flights-tabs tabs ui-tabs ui-widget ui-widget-content
ui-corner-all">
<ulrole="tablist" class="ui-tabs-nav ui-helper-reset ui-helper-
clearfix ui-widget-header ui-corner-all">
<liaria-selected="true" aria-labelledby="ui-id-7" aria-controls="sale-
flights-tabs-1" tabindex="0" role="tab" class="ui-state-default ui-
corner-top ui-tabs-active ui-state-active">
<aid="ui-id-7" tabindex="-1" role="presentation" class="ui-tabs-
anchor" href="#sale-flights-tabs-1">Avaliable Flights</a>
</li>
<liaria-selected="false" aria-labelledby="ui-id-8" aria-
controls="sale-flights-tabs-2" tabindex="-1" role="tab" class="ui-
state-default ui-corner-top">
<aid="ui-id-8" tabindex="-1" role="presentation" class="ui-tabs-
anchor" href="#sale-flights-tabs-2">Avaliable Bus
</a>
</li>
</ul>
<divclass="sale-flights-tabs__content tabs__content">
<divaria-hidden="false" aria-expanded="true" role="tabpanel"
class="ui-tabs-panel ui-widget-content ui-corner-bottom" aria-
labelledby="ui-id-7" id="sale-flights-tabs-1">

<divclass="item-media">
<divclass="image-cover">
<imgsrc="img/3.jpg" alt="">
</div>
<divclass="trip-icon">
<imgsrc="img/trip.jpg" alt="">
</div>
</div>
<divclass="item-body">
<divclass="item-title">

```



```

<h2><a href="#">Emirates Airway</a></h2>
</div>
<div class="">
<span>Nnamdi Azikiwe Airport</span>
<ul>

<li>6 hours</li>
</ul>
</div>
<div class="item-footer">
<div class="item-rate">
<span>12.pm</span>
</div>
<div class="item-icon">
<i class="awe-icon awe-icon-gym"></i>
<i class="awe-icon awe-icon-car"></i>
<i class="awe-icon awe-icon-food"></i>
<i class="awe-icon awe-icon-level"></i>
<i class="awe-icon awe-icon-wifi"></i>
</div>
</div>
</div>
<div class="item-price-more">
<div class="price">
<ins>
<span class="amount">N200,000</span>
</ins>
<del>
<span class="amount">N400,000</span>
</del>
</div>
<a href="register.php" class="awe-btn" class="btn btn-primary btn-
lg">Book now</a>
</div>
</div>
</div>
<div class="trip-item">
<div class="item-media">
<div class="image-cover">

</div>
<div class="trip-icon">

</div>
</div>
<div class="item-body">
<div class="item-title">
<h2><a href="#">GLOBAL AIRWAY</a></h2>
</div>
<div class="">
<span>BANANA AIR PORT</span>
<ul>

```

```

<li>30minuts</li>
</ul>
</div>
<divclass="item-footer">
<divclass="item-rate">
<span>13-24-23pm</span>
</div>
<divclass="item-icon">
<iclass="awe-icon awe-icon-gym"></i>
<iclass="awe-icon awe-icon-car"></i>
<iclass="awe-icon awe-icon-food"></i>
<iclass="awe-icon awe-icon-level"></i>
<iclass="awe-icon awe-icon-wifi"></i>
</div>
</div>
</div>
<divclass="item-price-more">
<divclass="price">Economy comfort
<ins>
<spanclass="amount">N133, 0000</span>
</ins>
<del>
<spanclass="amount">N400,000</span>
</del>
</div>
<a href="register.php" class="awe-btn" class="btn btn-primary btn-
lg">Book now</a>
</div>
</div>
</div>
<divaria-hidden="true" aria-expanded="false" style="display: none;"
role="tabpanel" class="ui-tabs-panel ui-widget-content ui-corner-
bottom" aria-labelledby="ui-id-8" id="sale-flights-tabs-2">

</div>
</div>
</div>
</div>

<divstyle="position: relative; overflow: visible; box-sizing: border-
box; min-height: 1px;" class="col-md-4">
<divstyle="padding-top: 0px; padding-bottom: 1px; position: static;
top: 0px; left: 884.5px;" class="theiaStickySidebar">
<divclass="awe-services">
<h2>Best flights</h2>
<ulclass="awe-services__list">
<li>
<a href="#">
<iclass="fa fa-check"></i>
<iclass="fa fa-arrow-right"></i> 100% trusted reviews
<span>We verify them in person</span>

```

```

</a>
</li>
<li>
<a href="#">
<i class="fa fa-check"></i>
<i class="fa fa-arrow-right"></i> 100% trusted reviews
<span>We verify them in person</span>
</a>
</li>
<li>
<a href="#">
<i class="fa fa-check"></i>
<i class="fa fa-arrow-right"></i> 24/7 global support
<span>anytime and any where</span>
</a>
</li>
<li>
<a href="#">
<i class="fa fa-check"></i>
<i class="fa fa-arrow-right"></i> 24/7 global support
<span>anytime and any where</span>
</a>
</li>
<li>
<a href="#">
<i class="fa fa-check"></i>
<i class="fa fa-arrow-right"></i> Manage your bookings online
<span>anytime and any where</span>
</a>
</li>
</ul>
</div>
</div>
</div>
<div style="position: relative; overflow: visible; box-sizing: border-
box; min-height: 1px;" class="col-md-4">
<div style="padding-top: 0px; padding-bottom: 1px; position: static;
top: 0px; left: 884.5px;" class="theiaStickySidebar">
<div class="awe-services">
<h2>Best Bus transports</h2>
<ul class="awe-services__list">
<li>
<a href="#">
<i class="fa fa-check"></i>
<i class="fa fa-arrow-right"></i> 100,000 real deals
<span>No booking fees . No fake</span>
</a>
</li>
<li>
<a href="#">
<i class="fa fa-check"></i>
<i class="fa fa-arrow-right"></i> 100% trusted reviews

```

```

<span>We verify them in person</span>
</a>
</li>
<li>
<a href="#">
<i class="fa fa-check"></i>
<i class="fa fa-arrow-right"></i> 24/7 global support
<span>anytime and any where</span>
</a>
</li>
<li>
<a href="#">
<i class="fa fa-check"></i>
<i class="fa fa-arrow-right"></i> 24/7 global support
<span>anytime and any where</span>
</a>
</li>
<li>
<a href="#">
<i class="fa fa-check"></i>
<i class="fa fa-arrow-right"></i> Manage your bookings online
<span>anytime and any where</span>
</a>
</li>
</ul>
</div>
</div>
</div>
<div style="position: relative; overflow: visible; box-sizing: border-
box; min-height: 1px;" class="col-md-4">
<div style="padding-top: 0px; padding-bottom: 1px; position: static;
top: 0px; left: 884.5px;" class="theiaStickySidebar">
<div class="awe-services">
<h2>Best train transport</h2>
<ul class="awe-services__list">
<li>
<a href="#">
<i class="fa fa-check"></i>
<i class="fa fa-arrow-right"></i> 100,000 real deals
<span>No booking fees . No fake</span>
</a>
</li>
<li>
<a href="#">
<i class="fa fa-check"></i>
<i class="fa fa-arrow-right"></i> 100% trusted reviews
<span>We verify them in person</span>
</a>
</li>
<li>
<a href="#">
<i class="fa fa-check"></i>

```



```

        </div>
        <divclass="col-md-2">
            <divclass="widget widget_about_us">
                <divclass="widget_content">
                    </div>
                </div>
            </div>
            <divclass="col-md-2">
                <divclass="widget widget_categories">
                    <ul>
                    </ul>
                </div>
            </div>
            <divclass="col-md-2">
                <divclass="widget widget_recent_entries">
                    <ul>
                    </ul>
                </div>
            </div>
            <divclass="col-md-3">
                <divclass="widget widget_follow_us">
                    <divclass="widget_content">
                        <p>For Special booking request, please call</p>
                        <spanclass="phone">08055284465</span>
                        <divclass="awe-social">
                            <a href="#">
                                <i class="fa fa-twitter"></i>
                            </a>
                            <a href="#">
                                <i class="fa fa-pinterest"></i>
                            </a>
                            <a href="#">
                                <i class="fa fa-facebook"></i>
                            </a>
                            <a href="#">
                                <i class="fa fa-youtube-play"></i></a>
                        </div></div>
                    </div></div>
                </div><divclass="copyright"><p>©2017 TRAVELER AGENT SYSTEM™ All rights reserved.</p>
            </div></div></footer></div>
            <scripttype="text/javascript" src="js/jquery-1.js"></script>
            <scripttype="text/javascript" src="js/masonry.js"></script>
            <scripttype="text/javascript" src="js/jquery_002.js"></script>
            <scripttype="text/javascript" src="js/jquery.js"></script>

```

```

<scripttype="text/javascript" src="js/theia-sticky-
sidebar.js"></script>
<scripttype="text/javascript" src="js/jquery 004.js"></script>
<scripttype="text/javascript" src="js/jquery.min.js"></script>
<scripttype="text/javascript" src="js/bootstrap.min.js"></script>
<scripttype="text/javascript" src="js/jquery-ui.js"></script>
<scripttype="text/javascript" src="js/scripts.js"></script>
<script>(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]
]||function(){
(i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new
Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
})(window,document,'script','//www.google-
analytics.com/analytics.js','ga');
ga('create','UA-20585382-5','megadrapal.com');
ga('send','pageview');</script>
<scripttype="text/javascript" src="js/jquery 003.js"></script>
<scripttype="text/javascript" src="js/jquery 005.js"></script>
<scripttype="text/javascript">if($('#slider-revolution').length) {
    $('#slider-revolution').show().revolution({
        ottedOverlay:"none",
        delay:10000,
        startwidth:1600,
        startheight:650,
        hideThumbs:200,
        thumbWidth:100,
        thumbHeight:50,
        thumbAmount:5,
        simplifyAll:"off",
        navigationType:"none",
        navigationArrows:"solo",
        navigationStyle:"preview4",
        touchenabled:"on",
        onHoverStop:"on",
        nextSlideOnWindowFocus:"off",
        swipe_threshold: 0.7,
        swipe_min_touches: 1,
        drag_block_vertical: false,
        parallax:"mouse",
        parallaxBgFreeze:"on",
        parallaxLevels:[7,4,3,2,5,4,3,2,1,0],
        keyboardNavigation:"off",
        navigationHAlign:"center",
        navigationVAlign:"bottom",
        navigationHOffset:0,
        navigationVOffset:20,
        soloArrowLeftHalign:"left",
        soloArrowLeftValign:"center",
        soloArrowLeftHOffset:20,
        soloArrowLeftVOffset:0,
        soloArrowRightHalign:"right",

```

```

        soloArrowRightValign:"center",
        soloArrowRightHOffset:20,
        soloArrowRightVOffset:0,
        shadow:0,
        fullWidth:"on",
        fullScreen:"off",
        spinner:"spinner2",
        stopLoop:"off",
        stopAfterLoops:-1,
        stopAtSlide:-1,
        shuffle:"off",
        autoHeight:"off",
        forceFullWidth:"off",
        hideThumbsOnMobile:"off",
        hideNavDelayOnMobile:1500,
        hideBulletsOnMobile:"off",
        hideArrowsOnMobile:"off",
        hideThumbsUnderResolution:0,
        hideSliderAtLimit:0,
        hideCaptionAtLimit:0,
        hideAllCaptionAtLimit:0,
        startWithSlide:0
    });
}
</script>
<div id="ui-datepicker-div" class="ui-datepicker ui-widget ui-widget-
content ui-helper-clearfix ui-corner-all"></div>
</body>
</html>

```

```

<!DOCTYPE html>
<html lang="en"><head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<meta charset="utf-8">
<title>TRAVELER AGENT SYSTEM</title>
<meta name="viewport" content="width=device-width,initial-scale=1,maximum-
scale=1,user-scalable=no">
<meta name="format-detection" content="telephone=no">
<meta name="apple-mobile-web-app-capable" content="yes">
<link href="css/css.css" rel="stylesheet" type="text/css">
<link href="css/css_002.css" rel="stylesheet" type="text/css">
<link href="css/css_003.css" rel="stylesheet" type="text/css">
<link rel="stylesheet" type="text/css" href="css/bootstrap.css">
<link rel="stylesheet" type="text/css" href="css/font-awesome.css">
<link rel="stylesheet" type="text/css" href="css/awe-booking-font.css">
<link rel="stylesheet" type="text/css" href="css/owl.css">
<link rel="stylesheet" type="text/css" href="css/jquery-ui.css">
<link rel="stylesheet" type="text/css" href="css/style.css">
<link rel="stylesheet" type="text/css" href="css/demo.css">
<link id="colorreplace" rel="stylesheet" type="text/css" href="css/blue.css">
<script src="css/analytics.js" async=""></script>
<script src="css/fbevents.js" async=""></script>
</script>

```



```

!function(f,b,e,v,n,t,s){if(f.fbq)return;n=f.fbq=function(){n.callMethod?
n.callMethod.apply(n,arguments):n.queue.push(arguments)};if(!f._fbq)f._fbq=n;
n.push=n;n.loaded=!0;n.version='2.0';n.queue=[];t=b.createElement(e);t.async=
!0;
t.src=v;s=b.getElementsByTagName(e)[0];s.parentNode.insertBefore(t,s)}(window
,
document,'script','//connect.facebook.net/en_US/fbevents.js');

fbq('init', '1031554816897182');
fbq('track', "PageView");</script>
<noscript>
<img height="1" width="1" style="display:none"/></noscript>
</head>
<body>
<div id="page-wrap">
<div style="display: none;" class="preloader"></div>
<header id="header-page">
<div style="transform: translateY(0px);" class="header-page__inner
header-page__fixed">
<div class="container"><div class="logo">
<a href="index.php">
<h5 class="animated shake">TRAVEL AGENT
SYSTEM</h5>
</a>
</div>
<nav style="height: auto;" class="navigation awe-navigation"
data-responsive="1200">
<ul class="menu-list">
<li class="menu-item-has-children">
<a href="index.php">Home</a>
</li>
<li class="menu-item-has-children current-menu-parent">
<a href="hotel.jade">Hotel</a>
</li>
<li class="menu-item-has-children">
<a href="bus.jade">Bus</a>
</li>
<li class="menu-item-has-children">
<a href="flit.jade">Flight</a>
</li>
<li class="menu-item-has-children">
<a href="train.jade">Train</a>
</li>
</ul>
</li>
</ul>
</nav>
<div class="search-box">
<span class="searchtoggle">
<i class="awe-icon awe-icon-search"></i>
</span>
<form style="right: 0px; width: 1140px;" class="form-search">

```

```

        <div class="form-item">
            <input value="Search & hit enter"
type="text">
        </div>
    </form>
</div>
<a style="display: none;" class="toggle-menu-responsive"
href="#">
    <div class="hamburger">
        <span class="item item-1"></span>
        <span class="item item-2"></span>
        <span class="item item-3"> </span>
    </div>
</a>
</div>
</div>
</header>
    <section style="background-position: 50% -76px;" class="category-
heading-section-demo">
    <div class="awe-overlay"></div>
    <div class="container">
    <div class="category-heading-content category-heading-content__2 text-
uppercase">
    <div class="breadcrumb">
    <ul>
    <li>
    <a href="#">Home</a>
    </li>
    <li>
    <span>Hotels</span>
    </li>
    </ul>
    </div>
    <div class="find">
    <h2 class="text-center">Get your hotel ticket</h2>
    <form>
    <div class="form-group">
    <div class="form-elements">
    <label>Location</label>
    <div class="form-item">
    <i class="awe-icon awe-icon-marker-1"></i>
    <input type="text">
    </div>
    </div>
    <div class="form-elements">
    <label>Check in</label>
    <div class="form-item">
    <i class="awe-icon awe-icon-calendar"></i>
    <input id="dp1478113312928" class="awe-calendar hasDatepicker"
value="Date" type="text">
    </div>
    </div>
    <div class="form-elements">
    <label>Check out</label>
    <div class="form-item">
    <i class="awe-icon awe-icon-calendar"></i>

```

```

        <input id="dp1478113312929" class="awe-calendar hasDatepicker"
value="Date" type="text">
    </div>
</div>
<div class="form-elements">
<label>Budget</label>
<div class="form-item">
<div class="awe-select-wrapper">
<select class="awe-select">
<option selected="selected">All types</option>
<option>1</option>
<option>2</option>
<option>3</option>
</select>
<i class="fa fa-caret-down"></i>
</div>
</div>
</div>
<div class="form-actions">
<input value="Find My Hotel" type="submit">
</div>
</div>
</form>
</div>
</div>
</div>
</div>
</section>
<section class="filter-page">
<div class="container">
<div class="row">
<div class="col-md-12">
<div class="page-top">
<div class="awe-select-wrapper">
<select class="awe-select">
<option selected="selected">Best Match</option>
<option>Best Rate</option>
</select>
<i class="fa fa-caret-down"></i>
</div>
</div>
</div>
<div class="col-md-9 col-md-push-3">
<div class="filter-page__content">
<div class="filter-item-wrapper">

<h5>Recent Hotel updates</h5>
<div class="hotel-item">
<div class="item-media">
<div class="image-cover">
</div>
</div>

<div class="item-body">
<div class="item-title"><h2><a href="#">Five star Hotel</a></h2></div>
<div class="item-hotel-star">
<i class="fa fa-star"></i>

```

```

<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
</div><div class="item-address">
<i class="awe-icon awe-icon-marker-2"></i> Lagos
</div>
<div class="item-footer">
<div class="item-rate">
<span>2725 Rooms</span>
</div><div class="item-icon">
<i class="awe-icon awe-icon-gym"></i>
<i class="awe-icon awe-icon-car"></i>
<i class="awe-icon awe-icon-food"></i>
<i class="awe-icon awe-icon-level"></i>
<i class="awe-icon awe-icon-wifi"></i>
</div>
</div>
</div>
<div class="item-price-more">
<div class="price">one night from
<span class="amount">N400,000</span>
</div>
<a href="register.php" class="awe-btn">Book now</a>
</div>
</div>

<div class="hotel-item">
<div class="item-media">
<div class="image-cover">
</div>
</div>

<div class="item-body">
<div class="item-title"><h2><a href="#">Protea Hotel</a></h2></div>
<div class="item-hotel-star">
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
</div><div class="item-address">
<i class="awe-icon awe-icon-marker-2"></i> Asaba
</div>
<div class="item-footer">
<div class="item-rate">
<span>1425 Rooms</span>
</div><div class="item-icon">
<i class="awe-icon awe-icon-gym"></i>
<i class="awe-icon awe-icon-car"></i>
<i class="awe-icon awe-icon-food"></i>
<i class="awe-icon awe-icon-level"></i>
<i class="awe-icon awe-icon-wifi"></i>
</div>
</div>
</div>

```

```

<div class="item-price-more">
<div class="price">one night from
<span class="amount">N420,000</span>
</div>
<a href="register.php" class="awe-btn">Book now</a>
</div>
</div>

<div class="hotel-item">
<div class="item-media">
<div class="image-cover">
</div>
</div>

<div class="item-body">
<div class="item-title"><h2><a href="#">Four Point
Hotel</a></h2></div>
<div class="item-hotel-star">
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
</div><div class="item-address">
<i class="awe-icon awe-icon-marker-2"></i> Abuja
</div>
<div class="item-footer">
<div class="item-rate">
<span>1725 Rooms</span>
</div><div class="item-icon">
<i class="awe-icon awe-icon-gym"></i>
<i class="awe-icon awe-icon-car"></i>
<i class="awe-icon awe-icon-food"></i>
<i class="awe-icon awe-icon-level"></i>
<i class="awe-icon awe-icon-wifi"></i>
</div>
</div>
</div>
<div class="item-price-more">
<div class="price">one night from
<span class="amount">N140,000</span>
</div>
<a href="register.php" class="awe-btn">Book now</a>
</div>
</div>

<div class="hotel-item">
<div class="item-media">
<div class="image-cover">
</div>
</div>

<div class="item-body">
<div class="item-title"><h2><a href="#">Nicon Hotel</a></h2></div>
<div class="item-hotel-star">

```

```

<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
</div><div class="item-address">
<i class="awe-icon awe-icon-marker-2"></i> Lagos
</div>
<div class="item-footer">
<div class="item-rate">
<span>725 Rooms</span>
</div><div class="item-icon">
<i class="awe-icon awe-icon-gym"></i>
<i class="awe-icon awe-icon-car"></i>
<i class="awe-icon awe-icon-food"></i>
<i class="awe-icon awe-icon-level"></i>
<i class="awe-icon awe-icon-wifi"></i>
</div>
</div>
</div>
<div class="item-price-more">
<div class="price">one night from
<span class="amount">N200,000</span>
</div>
<a href="register.php" class="awe-btn">Book now</a>
</div>
</div>
</div>
</div>
<div class="col-md-3 col-md-pull-9">
<div class="page-sidebar">
<div class="sidebar-title">
<h2>Hotel filter</h2>
<div class="clear-filter">
<a href="#">Clear all</a>
</div>
</div>
<div class="widget widget_has_radio_checkbox">
<h3>Hotel Type</h3>
<ul>
<li>
<label>
<input type="checkbox">
<i class="awe-icon awe-icon-check"></i>
Hotel
</label>
</li>
<li>
<label>
<input type="checkbox">
<i class="awe-icon awe-icon-check"></i>
Hostel
</label>
</li>
<li>

```

```

<label>
<input type="checkbox">
<i class="awe-icon awe-icon-check"></i>
Motel
</label>
</li>
<li>
<label>
<input type="checkbox">
<i class="awe-icon awe-icon-check"></i>
Homestay
</label>
</li>
</ul>
</div>
<div class="widget widget_price_filter"><h3>
Price Level
</h3>
<div class="price-slider-wrapper">
<div aria-disabled="false" class="price-slider ui-slider ui-slider-
horizontal ui-widget ui-widget-content ui-corner-all">
<div style="left: 0%; width: 100%;" class="ui-slider-range ui-widget-
header ui-corner-all">
</div>
<a style="left: 0%;" class="ui-slider-handle ui-state-default ui-
corner-all" href="#"></a>
<a style="left: 100%;" class="ui-slider-handle ui-state-default ui-
corner-all" href="#"></a>
</div>
<div class="price_slider_amount">
<div class="price_label">
<span class="from">$0</span> - <span class="to">$10000</span>
</div>
</div>
</div>
</div>
</section>
<footer id="footer-page">
<div class="container">
<div class="row">
<div class="col-md-3">
<div class="widget widget_contact_info">
<div class="widget_background">
<div
class="widget_background_half">
<div class="bg"></div>
</div>
<div
class="widget_background_half">
<div class="bg"></div>
</div>
</div>
<div class="logo">
<h5 style="color:
#fff;">TRANSPORT AGENT SYSTEM</h5>

```

```

        </div>
        <div class="widget_content">
            <p>25 California Avenue, Santa
Monica, California. USA</p>
            <p>08055284465</p>
            <a
href="#">contact@gofar.com</a>
        </div>
    </div>
</div>
<div class="col-md-2">
    <div class="widget widget_about_us">
        <div class="widget_content">
            </div>
        </div>
</div>
<div class="col-md-2">
    <div class="widget widget_categories">
        <ul>
        </ul>
    </div>
</div>
<div class="col-md-2">
    <div class="widget widget_recent_entries">
        <ul>
        </ul>
    </div>
</div>
<div class="col-md-3">
    <div class="widget widget_follow_us">
        <div class="widget_content">
            <p>For Special booking
request, please call</p>
            <span
class="phone">08055284465</span>
<div class="awe-social"><a href="#"><i class="fa fa-twitter"></i>
</a><a href="#">
<i class="fa fa-pinterest"></i>
</a><a href="#">
<i class="fa fa-facebook"></i>
</a>
<a href="#">
<i class="fa fa-youtube-play"></i>
        </a>
    </div>
</div>
</div>
</div>
<div class="copyright">

```



```

        <p>©2016 TRANSPORT AGENT SYSTEM™ All rights
reserved.</p>
    </div>
</div>
</footer>
</div>
<script type="text/javascript" src="js/jquery-1.js"></script>
<script type="text/javascript" src="js/masonry.js"></script>
<script type="text/javascript" src="js/jquery_002.js"></script>
<script type="text/javascript" src="js/jquery.js"></script>
<script type="text/javascript" src="js/theia-sticky-sidebar.js"></script>
<script type="text/javascript" src="js/jquery_004.js"></script>
<script type="text/javascript" src="js/jquery.min.js"></script>
<script type="text/javascript" src="js/bootstrap.min.js"></script>
<script type="text/javascript" src="js/jquery-ui.js"></script>
<script type="text/javascript" src="js/scripts.js"></script>
<script>(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||func
tion(){
(i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new
Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,
m)
})(window,document,'script','//www.google-analytics.com/analytics.js','ga');
ga('create','UA-20585382-5','megadrupal.com');
ga('send','pageview');</script>
<script type="text/javascript" src="js/jquery_003.js"></script>
<script type="text/javascript" src="js/jquery_005.js"></script>
<script type="text/javascript">if($('#slider-revolution').length){
    $('#slider-revolution').show().revolution({
        ottedOverlay:"none",
        delay:10000,
        startwidth:1600,
        startheight:650,
        hideThumbs:200,
        thumbWidth:100,
        thumbHeight:50,
        thumbAmount:5,
        simplifyAll:"off",
        navigationType:"none",
        navigationArrows:"solo",
        navigationStyle:"preview4",
        touchenabled:"on",
        onHoverStop:"on",
        nextSlideOnWindowFocus:"off",
        swipe_threshold: 0.7,
        swipe_min_touches: 1,
        drag_block_vertical: false,
        parallax:"mouse",
        parallaxBgFreeze:"on",
        parallaxLevels:[7,4,3,2,5,4,3,2,1,0],
        keyboardNavigation:"off",
        navigationHAlign:"center",
        navigationVAlign:"bottom",
        navigationHOffset:0,
        navigationVOffset:20,
        soloArrowLeftHalign:"left",
        soloArrowLeftValign:"center",

```

```

        soloArrowLeftHOffset:20,
        soloArrowLeftVOffset:0,
        soloArrowRightHalign:"right",
        soloArrowRightValign:"center",
        soloArrowRightHOffset:20,
        soloArrowRightVOffset:0,
        shadow:0,
        fullWidth:"on",
        fullScreen:"off",
        spinner:"spinner2",
        stopLoop:"off",
        stopAfterLoops:-1,
        stopAtSlide:-1,
        shuffle:"off",
        autoHeight:"off",
        forceFullWidth:"off",
        hideThumbsOnMobile:"off",
        hideNavDelayOnMobile:1500,
        hideBulletsOnMobile:"off",
        hideArrowsOnMobile:"off",
        hideThumbsUnderResolution:0,
        hideSliderAtLimit:0,
        hideCaptionAtLimit:0,
        hideAllCaptionAtLilmit:0,
        startWithSlide:0
    });
}
</script>
<div id="ui-datepicker-div" class="ui-datepicker ui-widget ui-widget-content
ui-helper-clearfix ui-corner-all"></div>
</body>
</html>

```

```

<!DOCTYPE html>
<html lang="en"><head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<meta charset="utf-8">
<title>TRAVELER AGENT SYSTEM</title>
<meta name="viewport" content="width=device-width,initial-scale=1,maximum-
scale=1,user-scalable=no">
<meta name="format-detection" content="telephone=no">
<meta name="apple-mobile-web-app-capable" content="yes">
<link href="css/css.css" rel="stylesheet" type="text/css">
<link href="css/css 002.css" rel="stylesheet" type="text/css">
<link href="css/css 003.css" rel="stylesheet" type="text/css">
<link rel="stylesheet" type="text/css" href="css/bootstrap.css">
<link rel="stylesheet" type="text/css" href="css/font-awesome.css">
<link rel="stylesheet" type="text/css" href="css/awe-booking-font.css">
<link rel="stylesheet" type="text/css" href="css/owl.css">
<link rel="stylesheet" type="text/css" href="css/jquery-ui.css">
<link rel="stylesheet" type="text/css" href="css/style.css">
<link rel="stylesheet" type="text/css" href="css/demo.css">
<link id="colorreplace" rel="stylesheet" type="text/css" href="css/blue.css">
<script src="css/analytics.js" async=""></script>
<script src="css/fbevents.js" async=""></script>
<script>

```

```

!function(f,b,e,v,n,t,s){if(f.fbq)return;n=f.fbq=function(){n.callMethod?
n.callMethod.apply(n,arguments):n.queue.push(arguments)};if(!f._fbq)f._fbq=n;
n.push=n;n.loaded=!0;n.version='2.0';n.queue=[];t=b.createElement(e);t.async=
!0;
t.src=v;s=b.getElementsByTagName(e)[0];s.parentNode.insertBefore(t,s)}(window
,
document,'script','//connect.facebook.net/en_US/fbevents.js');

fbq('init', '1031554816897182');
fbq('track', "PageView");</script>
<noscript>
<img height="1" width="1" style="display:none"/></noscript>
</head>
<body>
<div id="page-wrap">
<div style="display: none;" class="preloader"></div>
<header id="header-page">
<div style="transform: translateY(0px);" class="header-page__inner
header-page__fixed">
<div class="container"><div class="logo">
<a href="index.php">
<h5 class="animated shake">TRANSPORT AGENT
SYSTEM</h5>
</a>
</div>
<nav style="height: auto;" class="navigation awe-navigation"
data-responsive="1200">
<ul class="menu-list">
<li class="menu-item-has-children">
<a href="index.php">Home</a>
</li>
<li class="menu-item-has-children">
<a href="hotel.jade">Hotel</a>
</li>
<li class="menu-item-has-children
current-menu-parent">
<a href="bus.jade">Bus</a>
</li>
<li class="menu-item-has-children">
<a href="flit.jade">Flight</a>
</li>
<li class="menu-item-has-children">
<a href="train.jade">Train</a>
</li>
</ul>
</li>
</ul>
</nav>
<div class="search-box">
<span class="searchtoggle">
<i class="awe-icon awe-icon-search"></i>
</span>
<form style="right: 0px; width: 1140px;" class="form-search">

```

```

        <div class="form-item">
            <input value="Search & hit enter"
type="text">
        </div>
    </form>
</div>
<a style="display: none;" class="toggle-menu-responsive"
href="#">
    <div class="hamburger">
        <span class="item item-1"></span>
        <span class="item item-2"></span>
        <span class="item item-3"> </span>
    </div>
</a>
</div>
</div>
</header>
<section style="background-position: 50% -76px;" class="page-heading-
demo">
    <div class="awe-overlay"></div>
    <div class="container">
        <div class="category-heading-content category-heading-content__2 text-
uppercase">
            <div class="breadcrumb">
                <ul>
                    <li>
                        <a href="#">Home</a>
                    </li>
                    <li>
                        <span>Hotels</span>
                    </li>
                </ul>
            </div>
            <div class="find">
                <h2 class="text-center">Find Your Hotel</h2>
                <form>
                    <div class="form-group">
                        <div class="form-elements">
                            <label>Location</label>
                            <div class="form-item">
                                <i class="awe-icon awe-icon-marker-1"></i>
                                <input type="text">
                            </div>
                        </div>
                        <div class="form-elements">
                            <label>Check in</label>
                            <div class="form-item">
                                <i class="awe-icon awe-icon-calendar"></i>
                                <input id="dp1478113312928" class="awe-calendar hasDatepicker"
value="Date" type="text">
                            </div>
                        </div>
                        <div class="form-elements">
                            <label>Check out</label>
                            <div class="form-item">
                                <i class="awe-icon awe-icon-calendar"></i>

```

```

        <input id="dp1478113312929" class="awe-calendar hasDatepicker"
value="Date" type="text">
    </div>
</div>
<div class="form-elements">
<label>Budget</label>
<div class="form-item">
<div class="awe-select-wrapper">
<select class="awe-select">
<option selected="selected">All types</option>
<option>1</option>
<option>2</option>
<option>3</option>
</select>
<i class="fa fa-caret-down"></i>
</div>
</div>
</div>
<div class="form-actions">
<input value="Find My Hotel" type="submit">
</div>
</div>
</form>
</div>
</div>
</div>
</div>
</section>
<section class="filter-page">
<div class="container">
<div class="row">
<div class="col-md-12">
<div class="page-top">
<div class="awe-select-wrapper">
<select class="awe-select">
<option selected="selected">Best Match</option>
<option>Best Rate</option>
</select>
<i class="fa fa-caret-down"></i>
</div>
</div>
</div>
<div class="col-md-9 col-md-push-3">
<div class="filter-page__content">
<div class="filter-item-wrapper">
<h5>Recent Hotel updates</h5>
<div class="hotel-item">
<div class="item-media">
<div class="image-cover">
</div>
</div>
</div>
<div class="item-body">
<div class="item-title"><h2><a href="#">Five star Hotel</a></h2></div>
<div class="item-hotel-star">
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>

```

```

<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
</div><div class="item-address">
<i class="awe-icon awe-icon-marker-2"></i> Lagos
</div>
<div class="item-footer">
<div class="item-rate">
<span>2725 Rooms</span>
</div><div class="item-icon">
<i class="awe-icon awe-icon-gym"></i>
<i class="awe-icon awe-icon-car"></i>
<i class="awe-icon awe-icon-food"></i>
<i class="awe-icon awe-icon-level"></i>
<i class="awe-icon awe-icon-wifi"></i>
</div>
</div>
</div>
<div class="item-price-more">
<div class="price">one night from
<span class="amount">N400,000</span>
</div>
<a href="register.jade" class="awe-btn">Book now</a>
</div>
</div>

<div class="hotel-item">
<div class="item-media">
<div class="image-cover">
</div>
</div>

<div class="item-body">
<div class="item-title"><h2><a href="#">Protea Hotel</a></h2></div>
<div class="item-hotel-star">
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
</div><div class="item-address">
<i class="awe-icon awe-icon-marker-2"></i> Asaba
</div>
<div class="item-footer">
<div class="item-rate">
<span>1425 Rooms</span>
</div><div class="item-icon">
<i class="awe-icon awe-icon-gym"></i>
<i class="awe-icon awe-icon-car"></i>
<i class="awe-icon awe-icon-food"></i>
<i class="awe-icon awe-icon-level"></i>
<i class="awe-icon awe-icon-wifi"></i>
</div>
</div>
</div>
<div class="item-price-more">

```

```

<div class="price">one night from
<span class="amount">N420,000</span>
</div>
<a href="register.jade" class="awe-btn">Book now</a>
</div>
</div>

<div class="hotel-item">
<div class="item-media">
<div class="image-cover">
</div>
</div>

<div class="item-body">
<div class="item-title"><h2><a href="#">Four Point
Hotel</a></h2></div>
<div class="item-hotel-star">
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
</div><div class="item-address">
<i class="awe-icon awe-icon-marker-2"></i> Abuja
</div>
<div class="item-footer">
<div class="item-rate">
<span>1725 Rooms</span>
</div><div class="item-icon">
<i class="awe-icon awe-icon-gym"></i>
<i class="awe-icon awe-icon-car"></i>
<i class="awe-icon awe-icon-food"></i>
<i class="awe-icon awe-icon-level"></i>
<i class="awe-icon awe-icon-wifi"></i>
</div>
</div>
</div>
<div class="item-price-more">
<div class="price">one night from
<span class="amount">N140,000</span>
</div>
<a href="register.jade" class="awe-btn">Book now</a>
</div>
</div>

<div class="hotel-item">
<div class="item-media">
<div class="image-cover">
</div>
</div>

<div class="item-body">
<div class="item-title"><h2><a href="#">Nicon Hotel</a></h2></div>
<div class="item-hotel-star">
<i class="fa fa-star"></i>

```

```

<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
</div><div class="item-address">
<i class="awe-icon awe-icon-marker-2"></i> Lagos
</div>
<div class="item-footer">
<div class="item-rate">
<span>725 Rooms</span>
</div><div class="item-icon">
<i class="awe-icon awe-icon-gym"></i>
<i class="awe-icon awe-icon-car"></i>
<i class="awe-icon awe-icon-food"></i>
<i class="awe-icon awe-icon-level"></i>
<i class="awe-icon awe-icon-wifi"></i>
</div>
</div>
</div>
<div class="item-price-more">
<div class="price">one night from
<span class="amount">N200,000</span>
</div>
<a href="register.jade" class="awe-btn">Book now</a>
</div>
</div>
</div>
</div>
<div class="col-md-3 col-md-pull-9">
<div class="page-sidebar">
<div class="sidebar-title">
<h2>Hotel filter</h2>
<div class="clear-filter">
<a href="#">Clear all</a>
</div>
</div>
<div class="widget widget_has_radio_checkbox">
<h3>Hotel Type</h3>
<ul>
<li>
<label>
<input type="checkbox">
<i class="awe-icon awe-icon-check"></i>
Hotel
</label>
</li>
<li>
<label>
<input type="checkbox">
<i class="awe-icon awe-icon-check"></i>
Hostel
</label>
</li>
<li>
<label>

```



```


</i>
Motel
</label>
</li>
<li>
<label>

</i>
Homestay
</label>
</li>
</ul>
</div>
<div class="widget widget_price_filter"><h3>
Price Level
</h3>
<div class="price-slider-wrapper">
<div aria-disabled="false" class="price-slider ui-slider ui-slider-
horizontal ui-widget ui-widget-content ui-corner-all">
<div style="left: 0%; width: 100%;" class="ui-slider-range ui-widget-
header ui-corner-all">
</div>
<a style="left: 0%;" class="ui-slider-handle ui-state-default ui-
corner-all" href="#"></a>
<a style="left: 100%;" class="ui-slider-handle ui-state-default ui-
corner-all" href="#"></a>
</div>
<div class="price_slider_amount">
<div class="price_label">
<span class="from">$0</span> - <span class="to">$10000</span>
</div>
</div>
</div>
</div>
</div>
</section>
<footer id="footer-page">
<div class="container">
<div class="row">
<div class="col-md-3">
<div class="widget widget_contact_info">
<div class="widget_background">
<div
class="widget_background_half">
<div class="bg"></div>
</div>
<div
class="widget_background_half">
<div class="bg"></div>
</div>
</div>
<div class="logo">
<h5 style="color:
#fff;">TRANSPORT AGENT SYSTEM</h5>
</div>

```

```

                <div class="widget_content">
                    <p>25 California Avenue, Santa
Monica, California. USA</p>
                    <p>08055284465</p>
                    <a
href="#">contact@gofar.com</a>
                </div>
            </div>
        </div>
        <div class="col-md-2">
            <div class="widget widget_about_us">
                <div class="widget_content">
                    </div>
                </div>
            </div>
        </div>
        <div class="col-md-2">
            <div class="widget widget_categories">
                <ul>
                    </ul>
            </div>
        </div>
        <div class="col-md-2">
            <div class="widget widget_recent_entries">
                <ul>
                    </ul>
            </div>
        </div>
        <div class="col-md-3">
            <div class="widget widget_follow_us">
                <div class="widget_content">
                    <p>For Special booking
request, please call</p>
                    <span
class="phone">08055284465</span>
                </div>
            <div class="awe-social">
                <a href="#">
                    <i
class="fa fa-twitter"></i>
                </a>
                <a href="#">
                    <i
class="fa fa-pinterest"></i>
                </a>
                <a href="#">
                    <i
class="fa fa-facebook"></i>
                </a>
                <a href="#">
                    <i class="fa fa-youtube-play"></i>
                </a></div>
            </div></div>

```

```

</div>
                </div>
                <div class="copyright">
<p>©2016 TRANSPORT AGENT SYSTEM™ All rights reserved.</p>
                </div></div>
        </footer>
</div>
<script type="text/javascript" src="js/jquery-1.js"></script>
<script type="text/javascript" src="js/masonry.js"></script>
<script type="text/javascript" src="js/jquery_002.js"></script>
<script type="text/javascript" src="js/jquery.js"></script>
<script type="text/javascript" src="js/theia-sticky-sidebar.js"></script>
<script type="text/javascript" src="js/jquery_004.js"></script>
<script type="text/javascript" src="js/jquery.min.js"></script>
<script type="text/javascript" src="js/bootstrap.min.js"></script>
<script type="text/javascript" src="js/jquery-ui.js"></script>
<script type="text/javascript" src="js/scripts.js"></script>
<script>(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
(i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new
Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,
m)
})(window,document,'script','//www.google-analytics.com/analytics.js','ga');
ga('create','UA-20585382-5','megadrupal.com');
ga('send','pageview');</script>
<script type="text/javascript" src="js/jquery_003.js"></script>
<script type="text/javascript" src="js/jquery_005.js"></script>
<script type="text/javascript">if($('#slider-revolution').length){
    $('#slider-revolution').show().revolution({
        ottedOverlay:"none",
        delay:10000,
        startwidth:1600,
        startheight:650,
        hideThumbs:200,
        thumbWidth:100,
        thumbHeight:50,
        thumbAmount:5,
        simplifyAll:"off",
        navigationType:"none",
        navigationArrows:"solo",
        navigationStyle:"preview4",
        touchenabled:"on",
        onHoverStop:"on",
        nextSlideOnWindowFocus:"off",
        swipe_threshold: 0.7,
        swipe_min_touches: 1,
        drag_block_vertical: false,
        parallax:"mouse",
        parallaxBgFreeze:"on",
        parallaxLevels:[7,4,3,2,5,4,3,2,1,0],
        keyboardNavigation:"off",
        navigationHAlign:"center",
        navigationVAlign:"bottom",
        navigationHOffset:0,
        navigationVOffset:20,
        soloArrowLeftHalign:"left",

```

```

        soloArrowLeftValign:"center",
        soloArrowLeftHOffset:20,
        soloArrowLeftVOffset:0,
        soloArrowRightHalign:"right",
        soloArrowRightValign:"center",
        soloArrowRightHOffset:20,
        soloArrowRightVOffset:0,
        shadow:0,
        fullWidth:"on",
        fullScreen:"off",
        spinner:"spinner2",
        stopLoop:"off",
        stopAfterLoops:-1,
        stopAtSlide:-1,
        shuffle:"off",
        autoHeight:"off",
        forceFullWidth:"off",
        hideThumbsOnMobile:"off",
        hideNavDelayOnMobile:1500,
        hideBulletsOnMobile:"off",
        hideArrowsOnMobile:"off",
        hideThumbsUnderResolution:0,
        hideSliderAtLimit:0,
        hideCaptionAtLimit:0,
        hideAllCaptionAtLilmit:0,
        startWithSlide:0
    });
}
</script>
<div id="ui-datepicker-div" class="ui-datepicker ui-widget ui-widget-content
ui-helper-clearfix ui-corner-all"></div>
</body>
</html>

```

```

<!DOCTYPE html>
<html lang="en"><head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<meta charset="utf-8">
<title>TRAVELER AGENT SYSTEM</title>
<meta name="viewport" content="width=device-width,initial-scale=1,maximum-
scale=1,user-scalable=no">
<meta name="format-detection" content="telephone=no">
<meta name="apple-mobile-web-app-capable" content="yes">
<link href="css/css.css" rel="stylesheet" type="text/css">
<link href="css/css 002.css" rel="stylesheet" type="text/css">
<link href="css/css 003.css" rel="stylesheet" type="text/css">
<link rel="stylesheet" type="text/css" href="css/bootstrap.css">
<link rel="stylesheet" type="text/css" href="css/font-awesome.css">
<link rel="stylesheet" type="text/css" href="css/awe-booking-font.css">
<link rel="stylesheet" type="text/css" href="css/owl.css">
<link rel="stylesheet" type="text/css" href="css/jquery-ui.css">
<link rel="stylesheet" type="text/css" href="css/style.css">
<link rel="stylesheet" type="text/css" href="css/demo.css">
<link id="colorreplace" rel="stylesheet" type="text/css" href="css/blue.css">
<script src="css/analytics.js" async=""></script>
<script src="css/fbevents.js" async=""></script>

```

```

<script>
!function(f,b,e,v,n,t,s){if(f.fbq)return;n=f.fbq=function(){n.callMethod?
n.callMethod.apply(n,arguments):n.queue.push(arguments)};if(!f._fbq)f._fbq=n;
n.push=n;n.loaded=!0;n.version='2.0';n.queue=[];t=b.createElement(e);t.async=
!0;
t.src=v;s=b.getElementsByTagName(e)[0];s.parentNode.insertBefore(t,s)}(window
,
document,'script','//connect.facebook.net/en_US/fbevents.js');

fbq('init', '1031554816897182');
fbq('track', "PageView");</script>
<noscript>
<img height="1" width="1" style="display:none"/></noscript>
</head>
  <body>
    <div id="page-wrap">
      <div style="display: none;" class="preloader"></div>
      <header id="header-page">
        <div style="transform: translateY(0px);" class="header-page__inner
header-page__fixed">
          <div class="container"><div class="logo">
            <a href="index.php">
              <h5 class="animated shake">TRAVEL AGENT
SYSTEM</h5>
            </a>
          </div>
          <nav style="height: auto;" class="navigation awe-navigation"
data-responsive="1200">
            <ul class="menu-list">
              <li class="menu-item-has-children">
                <a href="index.jade">Home</a>
              </li>
              <li class="menu-item-has-children">
                <a href="hotel.jade">Hotel</a>
              </li>
              <li class="menu-item-has-children">
                <a href="bus.jade">Bus</a>
              </li>
              <li class="menu-item-has-children
current-menu-parent">
                <a href="flit.jade">Flight</a>
              </li>
              <li class="menu-item-has-children">
                <a href="train.jade">Train</a>
              </li>
            </ul>
          </li>
        </ul>
      </nav>
      <div class="search-box">
        <span class="searchtoggle">
          <i class="awe-icon awe-icon-search"></i>
        </span>

```

```

        <form style="right: 0px; width: 1140px;" class="form-search">
            <div class="form-item">
                <input value="Search & hit enter"
type="text">
            </div>
        </form>
    </div>
    <a style="display: none;" class="toggle-menu-responsive"
href="#">
        <div class="hamburger">
            <span class="item item-1"></span>
            <span class="item item-2"></span>
            <span class="item item-3"> </span>
        </div>
    </a>
</div>
</div>
</header>
    <section style="background-position: 50% -76px;" class="register-page-
demo">
    <div class="awe-overlay"></div>
    <div class="container">
    <div class="category-heading-content category-heading-content__2 text-
uppercase">
    <div class="breadcrumb">
    <ul>
    <li>
    <a href="#">Home</a>
    </li>
    <li>
    <span>Flights</span>
    </li>
    </ul>
    </div>
    <div class="find">
    <h2 class="text-center">Find Your Flight</h2>
    <form>
    <div class="form-group">
    <div class="form-elements">
    <label>Location</label>
    <div class="form-item">
    <i class="awe-icon awe-icon-marker-1"></i>
    <input type="text">
    </div>
    </div>
    <div class="form-elements">
    <label>Check in</label>
    <div class="form-item">
    <i class="awe-icon awe-icon-calendar"></i>
    <input id="dp1478113312928" class="awe-calendar hasDatepicker"
value="Date" type="text">
    </div>
    </div>
    <div class="form-elements">
    <label>Check out</label>
    <div class="form-item">
    <i class="awe-icon awe-icon-calendar"></i>

```

```

        <input id="dp1478113312929" class="awe-calendar hasDatepicker"
value="Date" type="text">
    </div>
</div>
<div class="form-elements">
<label>Budget</label>
<div class="form-item">
<div class="awe-select-wrapper">
<select class="awe-select">
<option selected="selected">All types</option>
<option>1</option>
<option>2</option>
<option>3</option>
</select>
<i class="fa fa-caret-down"></i>
</div>
</div>
</div>
<div class="form-actions">
<input value="Find My Hotel" type="submit">
</div>
</div>
</form>
</div>
</div>
</div>
</div>
</section>
<section class="filter-page">
<div class="container">
<div class="row">
<div class="col-md-12">
<div class="page-top">
<div class="awe-select-wrapper">
<select class="awe-select">
<option selected="selected">Best Match</option>
<option>Best Rate</option>
</select>
<i class="fa fa-caret-down"></i>
</div>
</div>
</div>
<div class="col-md-9 col-md-push-3">
<div class="filter-page__content">
<div class="filter-item-wrapper">
<h5>Recent Hotel updates</h5>
<div class="hotel-item">
<div class="item-media">
<div class="image-cover">
</div>
</div>
</div>
<div class="item-body">
<div class="item-title"><h2><a href="#">Five star Hotel</a></h2></div>
<div class="item-hotel-star">
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>

```

```

<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
</div><div class="item-address">
<i class="awe-icon awe-icon-marker-2"></i> Lagos
</div>
<div class="item-footer">
<div class="item-rate">
<span>2725 Rooms</span>
</div><div class="item-icon">
<i class="awe-icon awe-icon-gym"></i>
<i class="awe-icon awe-icon-car"></i>
<i class="awe-icon awe-icon-food"></i>
<i class="awe-icon awe-icon-level"></i>
<i class="awe-icon awe-icon-wifi"></i>
</div>
</div>
</div>
<div class="item-price-more">
<div class="price">one night from
<span class="amount">N400,000</span>
</div>
<a href="register.php" class="awe-btn">Book now</a>
</div>
</div>

<div class="hotel-item">
<div class="item-media">
<div class="image-cover">
</div>
</div>

<div class="item-body">
<div class="item-title"><h2><a href="#">Protea Hotel</a></h2></div>
<div class="item-hotel-star">
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
</div><div class="item-address">
<i class="awe-icon awe-icon-marker-2"></i> Asaba
</div>
<div class="item-footer">
<div class="item-rate">
<span>1425 Rooms</span>
</div><div class="item-icon">
<i class="awe-icon awe-icon-gym"></i>
<i class="awe-icon awe-icon-car"></i>
<i class="awe-icon awe-icon-food"></i>
<i class="awe-icon awe-icon-level"></i>
<i class="awe-icon awe-icon-wifi"></i>
</div>
</div>
</div>
<div class="item-price-more">

```



```

<div class="price">one night from
<span class="amount">N420,000</span>
</div>
<a href="register.php" class="awe-btn">Book now</a>
</div>
</div>

<div class="hotel-item">
<div class="item-media">
<div class="image-cover">
</div>
</div>

<div class="item-body">
<div class="item-title"><h2><a href="#">Four Point
Hotel</a></h2></div>
<div class="item-hotel-star">
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
</div><div class="item-address">
<i class="awe-icon awe-icon-marker-2"></i> Abuja
</div>
<div class="item-footer">
<div class="item-rate">
<span>1725 Rooms</span>
</div><div class="item-icon">
<i class="awe-icon awe-icon-gym"></i>
<i class="awe-icon awe-icon-car"></i>
<i class="awe-icon awe-icon-food"></i>
<i class="awe-icon awe-icon-level"></i>
<i class="awe-icon awe-icon-wifi"></i>
</div>
</div>
</div>
<div class="item-price-more">
<div class="price">one night from
<span class="amount">N140,000</span>
</div>
<a href="register.php" class="awe-btn">Book now</a>
</div>
</div>

<div class="hotel-item">
<div class="item-media">
<div class="image-cover">
</div>
</div>

<div class="item-body">
<div class="item-title"><h2><a href="#">Nicon Hotel</a></h2></div>
<div class="item-hotel-star">
<i class="fa fa-star"></i>

```

```

<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
</div><div class="item-address">
<i class="awe-icon awe-icon-marker-2"></i> Lagos
</div>
<div class="item-footer">
<div class="item-rate">
<span>725 Rooms</span>
</div><div class="item-icon">
<i class="awe-icon awe-icon-gym"></i>
<i class="awe-icon awe-icon-car"></i>
<i class="awe-icon awe-icon-food"></i>
<i class="awe-icon awe-icon-level"></i>
<i class="awe-icon awe-icon-wifi"></i>
</div>
</div>
</div>
<div class="item-price-more">
<div class="price">one night from
<span class="amount">N200,000</span>
</div>
<a href="register.php" class="awe-btn">Book now</a>
</div>
</div>
</div>
</div>
<div class="col-md-3 col-md-pull-9">
<div class="page-sidebar">
<div class="sidebar-title">
<h2>Hotel filter</h2>
<div class="clear-filter">
<a href="#">Clear all</a>
</div>
</div>
<div class="widget widget_has_radio_checkbox">
<h3>Hotel Type</h3>
<ul>
<li>
<label>
<input type="checkbox">
<i class="awe-icon awe-icon-check"></i>
Hotel
</label>
</li>
<li>
<label>
<input type="checkbox">
<i class="awe-icon awe-icon-check"></i>
Hostel
</label>
</li>
<li>
<label>

```

```


</i>
Motel
</label>
</li>
<li>
<label>

</i>
Homestay
</label>
</li>
</ul>
</div>
<div class="widget widget_price_filter"><h3>
Price Level
</h3>
<div class="price-slider-wrapper">
<div aria-disabled="false" class="price-slider ui-slider ui-slider-
horizontal ui-widget ui-widget-content ui-corner-all">
<div style="left: 0%; width: 100%;" class="ui-slider-range ui-widget-
header ui-corner-all">
</div>
<a style="left: 0%;" class="ui-slider-handle ui-state-default ui-
corner-all" href="#"></a>
<a style="left: 100%;" class="ui-slider-handle ui-state-default ui-
corner-all" href="#"></a>
</div>
<div class="price_slider_amount">
<div class="price_label">
<span class="from">$0</span> - <span class="to">$10000</span>
</div>
</div>
</div>
</div>
</div>
</section>
<footer id="footer-page">
<div class="container">
<div class="row">
<div class="col-md-3">
<div class="widget widget_contact_info">
<div class="widget_background">
<div
class="widget_background_half">
<div class="bg"></div>
</div>
<div
class="widget_background_half">
<div class="bg"></div>
</div>
</div>
<div class="logo">
<h5 style="color:
#fff;">TRANSPORT AGENT SYSTEM</h5>
</div>

```

```

                <div class="widget_content">
                    <p>25 California Avenue, Santa
Monica, California. USA</p>
                    <p>08055284465</p>
                    <a
href="#">contact@gofar.com</a>
                </div>
            </div>
        </div>
        <div class="col-md-2">
            <div class="widget widget_about_us">
                <div class="widget_content">
                    </div>
                </div>
            </div>
        </div>
        <div class="col-md-2">
            <div class="widget widget_categories">
                <ul>
                    </ul>
            </div>
        </div>
        <div class="col-md-2">
            <div class="widget widget_recent_entries">
                <ul>
                    </ul>
            </div>
        </div>
        <div class="col-md-3">
            <div class="widget widget_follow_us">
                <div class="widget_content">
                    <p>For Special booking
request, please call</p>
                    <span
class="phone">08055284465</span>
                </div>
                <div class="awe-social">
                    <a href="#">
                        <i
class="fa fa-twitter"></i>
                    </a>
                    <a href="#">
                        <i
class="fa fa-pinterest"></i>
                    </a>
                    <a href="#">
                        <i class="fa fa-facebook"></i></a><a href="#">
                        <i class="fa fa-youtube-play"></i>
                    </a>
                </div>
            </div>
        </div>
    </div>

```

```

        </div>
        <div class="copyright"
<p>©2016 TRANSPORT AGENT SYSTEM™ All rights reserved.</p>
        </div>
    </div>
</footer>
</div>
<script type="text/javascript" src="js/jquery-1.js"></script>
<script type="text/javascript" src="js/masonry.js"></script>
<script type="text/javascript" src="js/jquery_002.js"></script>
<script type="text/javascript" src="js/jquery.js"></script>
<script type="text/javascript" src="js/theia-sticky-sidebar.js"></script>
<script type="text/javascript" src="js/jquery_004.js"></script>
<script type="text/javascript" src="js/jquery.min.js"></script>
<script type="text/javascript" src="js/bootstrap.min.js"></script>
<script type="text/javascript" src="js/jquery-ui.js"></script>
<script type="text/javascript" src="js/scripts.js"></script>
<script>(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
(i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new
Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,
m)
})(window,document,'script','//www.google-analytics.com/analytics.js','ga');
ga('create','UA-20585382-5','megadrupal.com');
ga('send','pageview');</script>
<script type="text/javascript" src="js/jquery_003.js"></script>
<script type="text/javascript" src="js/jquery_005.js"></script>
<script type="text/javascript">if($('#slider-revolution').length){
    $('#slider-revolution').show().revolution({
        ottedOverlay:"none",
        delay:10000,
        startwidth:1600,
        startheight:650,
        hideThumbs:200,
        thumbWidth:100,
        thumbHeight:50,
        thumbAmount:5,
        simplifyAll:"off",
        navigationType:"none",
        navigationArrows:"solo",
        navigationStyle:"preview4",
        touchenabled:"on",
        onHoverStop:"on",
        nextSlideOnWindowFocus:"off",
        swipe_threshold: 0.7,
        swipe_min_touches: 1,
        drag_block_vertical: false,
        parallax:"mouse",
        parallaxBgFreeze:"on",
        parallaxLevels:[7,4,3,2,5,4,3,2,1,0],
        keyboardNavigation:"off",
        navigationHAlign:"center",
        navigationVAlign:"bottom",
        navigationHOffset:0,
        navigationVOffset:20,
        soloArrowLeftHalign:"left",

```

```

        soloArrowLeftValign:"center",
        soloArrowLeftHOffset:20,
        soloArrowLeftVOffset:0,
        soloArrowRightHalign:"right",
        soloArrowRightValign:"center",
        soloArrowRightHOffset:20,
        soloArrowRightVOffset:0,
        shadow:0,
        fullWidth:"on",
        fullScreen:"off",
        spinner:"spinner2",
        stopLoop:"off",
        stopAfterLoops:-1,
        stopAtSlide:-1,
        shuffle:"off",
        autoHeight:"off",
        forceFullWidth:"off",
        hideThumbsOnMobile:"off",
        hideNavDelayOnMobile:1500,
        hideBulletsOnMobile:"off",
        hideArrowsOnMobile:"off",
        hideThumbsUnderResolution:0,
        hideSliderAtLimit:0,
        hideCaptionAtLimit:0,
        hideAllCaptionAtLilmit:0,
        startWithSlide:0
    });
}
</script>
<div id="ui-datepicker-div" class="ui-datepicker ui-widget ui-widget-content
ui-helper-clearfix ui-corner-all"></div>
</body>
</html>

```

```

<!DOCTYPE html>
<html lang="en"><head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<meta charset="utf-8">
<title>TRAVELER AGENT SYSTEM</title>
<meta name="viewport" content="width=device-width,initial-scale=1,maximum-
scale=1,user-scalable=no">
<meta name="format-detection" content="telephone=no">
<meta name="apple-mobile-web-app-capable" content="yes">
<link href="css/css.css" rel="stylesheet" type="text/css">
<link href="css/css 002.css" rel="stylesheet" type="text/css">
<link href="css/css 003.css" rel="stylesheet" type="text/css">
<link rel="stylesheet" type="text/css" href="css/bootstrap.css">
<link rel="stylesheet" type="text/css" href="css/font-awesome.css">
<link rel="stylesheet" type="text/css" href="css/awe-booking-font.css">
<link rel="stylesheet" type="text/css" href="css/owl.css">
<link rel="stylesheet" type="text/css" href="css/jquery-ui.css">
<link rel="stylesheet" type="text/css" href="css/style.css">
<link rel="stylesheet" type="text/css" href="css/demo.css">
<link id="colorreplace" rel="stylesheet" type="text/css" href="css/blue.css">
<script src="css/analytics.js" async=""></script>
<script src="css/fbevents.js" async=""></script>

```

```

<script>
!function(f,b,e,v,n,t,s){if(f.fbq)return;n=f.fbq=function(){n.callMethod?
n.callMethod.apply(n,arguments):n.queue.push(arguments)};if(!f._fbq)f._fbq=n;
n.push=n;n.loaded=!0;n.version='2.0';n.queue=[];t=b.createElement(e);t.async=
!0;
t.src=v;s=b.getElementsByTagName(e)[0];s.parentNode.insertBefore(t,s)}(window
,
document,'script','//connect.facebook.net/en_US/fbevents.js');

fbq('init', '1031554816897182');
fbq('track', "PageView");</script>
<noscript>
<img height="1" width="1" style="display:none"/></noscript>
</head>
  <body>
    <div id="page-wrap">
      <div style="display: none;" class="preloader"></div>
      <header id="header-page">
        <div style="transform: translateY(0px);" class="header-page__inner
header-page__fixed">
          <div class="container"><div class="logo">
            <a href="index.php">
              <h5 class="animated shake">TRAVEL AGENT
SYSTEM</h5>
            </a>
          </div>
          <nav style="height: auto;" class="navigation awe-navigation"
data-responsive="1200">
            <ul class="menu-list">
              <li class="menu-item-has-children">
                <a href="index.jade">Home</a>
              </li>
              <li class="menu-item-has-children">
                <a href="hotel.jade">Hotel</a>
              </li>
              <li class="menu-item-has-children">
                <a href="bus.jade">Bus</a>
              </li>
              <li class="menu-item-has-children">
                <a href="flit.jade">Flight</a>
              </li>
              <li class="menu-item-has-children
current-menu-parent">
                <a href="train.jade">Train</a>
              </li>
            </ul>
          </li>
        </ul>
      </nav>

```

```

    <div class="search-box">
      <span class="searchtoggle">
        <i class="awe-icon awe-icon-search"></i>
      </span>
      <form style="right: 0px; width: 1140px;" class="form-search">
        <div class="form-item">
          <input value="Search & hit enter"
type="text">
        </div>
      </form>
    </div>
    <a style="display: none;" class="toggle-menu-responsive"
href="#">
      <div class="hamburger">
        <span class="item item-1"></span>
        <span class="item item-2"></span>
        <span class="item item-3"> </span>
      </div>
    </a>
  </div>
</div>
</header>
  <section style="background-position: 50% -76px;" class="login-page-
demo">
    <div class="awe-overlay"></div>
    <div class="container">
      <div class="category-heading-content category-heading-content__2 text-
uppercase">
        <div class="breadcrumb">
          <ul>
            <li>
              <a href="#">Home</a>
            </li>
            <li>
              <span>Hotels</span>
            </li>
          </ul>
        </div>
        <div class="find">
          <h2 class="text-center">Get your Train Ticket</h2>
          <form>
            <div class="form-group">
              <div class="form-elements">
                <label>Location</label>
                <div class="form-item">
                  <i class="awe-icon awe-icon-marker-1"></i>
                  <input type="text">
                </div>
              </div>
              <div class="form-elements">
                <label>Check in</label>
                <div class="form-item">
                  <i class="awe-icon awe-icon-calendar"></i>
                  <input id="dp1478113312928" class="awe-calendar hasDatepicker"
value="Date" type="text">
                </div>
              </div>
            </div>
          </form>
        </div>
      </div>
    </div>
  </div>

```



```

    <div class="form-elements">
    <label>Check out</label>
    <div class="form-item">
    <i class="awe-icon awe-icon-calendar"></i>
    <input id="dp1478113312929" class="awe-calendar hasDatepicker"
value="Date" type="text">
    </div>
    </div>
    <div class="form-elements">
    <label>Budget</label>
    <div class="form-item">
    <div class="awe-select-wrapper">
    <select class="awe-select">
    <option selected="selected">All types</option>
    <option>1</option>
    <option>2</option>
    <option>3</option>
    </select>
    <i class="fa fa-caret-down"></i>
    </div>
    </div>
    </div>
    <div class="form-actions">
    <input value="Find My Hotel" type="submit">
    </div>
    </div>
    </form>
    </div>
    </div>
    </div>
    </section>
    <section class="filter-page">
    <div class="container">
    <div class="row">
    <div class="col-md-12">
    <div class="page-top">
    <div class="awe-select-wrapper">
    <select class="awe-select">
    <option selected="selected">Best Match</option>
    <option>Best Rate</option>
    </select>
    <i class="fa fa-caret-down"></i>
    </div>
    </div>
    </div>
    <div class="col-md-9 col-md-push-3">
    <div class="filter-page__content">
    <div class="filter-item-wrapper">
    <h5>Recent Hotel updates</h5>
    <div class="hotel-item">
    <div class="item-media">
    <div class="image-cover">
    </div>
    </div>
    </div>
    </div>
    <div class="item-body">

```

```

<div class="item-title"><h2><a href="#">Five star Hotel</a></h2></div>
<div class="item-hotel-star">
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
</div><div class="item-address">
<i class="awe-icon awe-icon-marker-2"></i> Lagos
</div>
<div class="item-footer">
<div class="item-rate">
<span>2725 Rooms</span>
</div><div class="item-icon">
<i class="awe-icon awe-icon-gym"></i>
<i class="awe-icon awe-icon-car"></i>
<i class="awe-icon awe-icon-food"></i>
<i class="awe-icon awe-icon-level"></i>
<i class="awe-icon awe-icon-wifi"></i>
</div>
</div>
</div>
<div class="item-price-more">
<div class="price">one night from
<span class="amount">N400,000</span>
</div>
<a href="register.php" class="awe-btn">Book now</a>
</div>
</div>

<div class="hotel-item">
<div class="item-media">
<div class="image-cover">
</div>
</div>

<div class="item-body">
<div class="item-title"><h2><a href="#">Protea Hotel</a></h2></div>
<div class="item-hotel-star">
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
</div><div class="item-address">
<i class="awe-icon awe-icon-marker-2"></i> Asaba
</div>
<div class="item-footer">
<div class="item-rate">
<span>1425 Rooms</span>
</div><div class="item-icon">
<i class="awe-icon awe-icon-gym"></i>
<i class="awe-icon awe-icon-car"></i>
<i class="awe-icon awe-icon-food"></i>
<i class="awe-icon awe-icon-level"></i>
<i class="awe-icon awe-icon-wifi"></i>

```

```

</div>
</div>
</div>
<div class="item-price-more">
<div class="price">one night from
<span class="amount">N420,000</span>
</div>
<a href="register.php" class="awe-btn">Book now</a>
</div>
</div>

<div class="hotel-item">
<div class="item-media">
<div class="image-cover">
</div>
</div>

<div class="item-body">
<div class="item-title"><h2><a href="#">Four Point
Hotel</a></h2></div>
<div class="item-hotel-star">
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
</div><div class="item-address">
<i class="awe-icon awe-icon-marker-2"></i> Abuja
</div>
<div class="item-footer">
<div class="item-rate">
<span>1725 Rooms</span>
</div><div class="item-icon">
<i class="awe-icon awe-icon-gym"></i>
<i class="awe-icon awe-icon-car"></i>
<i class="awe-icon awe-icon-food"></i>
<i class="awe-icon awe-icon-level"></i>
<i class="awe-icon awe-icon-wifi"></i>
</div>
</div>
</div>
<div class="item-price-more">
<div class="price">one night from
<span class="amount">N140,000</span>
</div>
<a href="register.php" class="awe-btn">Book now</a>
</div>
</div>

<div class="hotel-item">
<div class="item-media">
<div class="image-cover">
</div>
</div>

```

```

<div class="item-body">
<div class="item-title"><h2><a href="#">Nicon Hotel</a></h2></div>
<div class="item-hotel-star">
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
</div><div class="item-address">
<i class="awe-icon awe-icon-marker-2"></i> Lagos
</div>
<div class="item-footer">
<div class="item-rate">
<span>725 Rooms</span>
</div><div class="item-icon">
<i class="awe-icon awe-icon-gym"></i>
<i class="awe-icon awe-icon-car"></i>
<i class="awe-icon awe-icon-food"></i>
<i class="awe-icon awe-icon-level"></i>
<i class="awe-icon awe-icon-wifi"></i>
</div>
</div>
</div>
<div class="item-price-more">
<div class="price">one night from
<span class="amount">N200,000</span>
</div>
<a href="register.php" class="awe-btn">Book now</a>
</div>
</div>
</div>
</div>
<div class="col-md-3 col-md-pull-9">
<div class="page-sidebar">
<div class="sidebar-title">
<h2>Hotel filter</h2>
<div class="clear-filter">
<a href="#">Clear all</a>
</div>
</div>
<div class="widget widget_has_radio_checkbox">
<h3>Hotel Type</h3>
<ul>
<li>
<label>
<input type="checkbox">
<i class="awe-icon awe-icon-check"></i>
Hotel
</label>
</li>
<li>
<label>
<input type="checkbox">
<i class="awe-icon awe-icon-check"></i>
Hostel

```

```

</label>
</li>
<li>
<label>
<input type="checkbox">
<i class="awe-icon awe-icon-check"></i>
Motel
</label>
</li>
<li>
<label>
<input type="checkbox">
<i class="awe-icon awe-icon-check"></i>
Homestay
</label>
</li>
</ul>
</div>
<div class="widget widget_price_filter"><h3>
Price Level
</h3>
<div class="price-slider-wrapper">
<div aria-disabled="false" class="price-slider ui-slider ui-slider-
horizontal ui-widget ui-widget-content ui-corner-all">
<div style="left: 0%; width: 100%;" class="ui-slider-range ui-widget-
header ui-corner-all">
</div>
<a style="left: 0%;" class="ui-slider-handle ui-state-default ui-
corner-all" href="#"></a>
<a style="left: 100%;" class="ui-slider-handle ui-state-default ui-
corner-all" href="#"></a>
</div>
<div class="price_slider_amount">
<div class="price_label">
<span class="from">$0</span> - <span class="to">$10000</span>
</div>
</div>
</div>
</div>
</div>
</section>
<footer id="footer-page">
<div class="container">
<div class="row">
<div class="col-md-3">
<div class="widget widget_contact_info">
<div class="widget_background">
<div
class="widget_background_half">
<div class="bg"></div>
</div>
<div
class="widget_background_half">
<div class="bg"></div>
</div>
</div>
</div>

```

```

                                <div class="logo">
                                    <h5 style="color:
#fff;">TRANSPORT AGENT SYSTEM</h5>
                                </div>
                                <div class="widget_content">
                                    <p>25 California Avenue, Santa
Monica, California. USA</p>
                                    <p>08055284465</p>
                                    <a
href="#">contact@gofar.com</a>
                                </div>
                            </div>
                        </div>
                    <div class="col-md-2">
                        <div class="widget widget_about_us">
                            <div class="widget_content">
                                </div>
                            </div>
                        </div>
                    </div>
                    <div class="col-md-2">
                        <div class="widget widget_categories">
                            <ul>
                                </ul>
                        </div>
                    </div>
                    <div class="col-md-2">
                        <div class="widget widget_recent_entries">
                            <ul>
                                </ul>
                        </div>
                    </div>
                    <div class="col-md-3">
                        <div class="widget widget_follow_us">
                            <div class="widget_content">
                                <p>For Special booking
request, please call</p>
                                <span class="phone">08055284465</span>
                                <div class="awe-social">
                                    <a href="#">
                                        <i class="fa fa-twitter"></i>
                                    </a>
                                    <a href="#">
                                        <i class="fa fa-pinterest"></i>
                                    </a><a href="#">
                                        <i class="fa fa-facebook"></i>
                                    </a>
                                    <a href="#">
                                        <i class="fa fa-youtube-play"></i>
                                    </a>
                                </div>
                            </div>
                        </div>
                    </div>

```

```

</div>
</div>
</div>
<div class="copyright">
<p>©2016 TRANSPORT AGENT SYSTEM™ All rights reserved.</p>
</div>
</div>
</footer>
</div>
<script type="text/javascript" src="js/jquery-1.js"></script>
<script type="text/javascript" src="js/masonry.js"></script>
<script type="text/javascript" src="js/jquery_002.js"></script>
<script type="text/javascript" src="js/jquery.js"></script>
<script type="text/javascript" src="js/theia-sticky-sidebar.js"></script>
<script type="text/javascript" src="js/jquery_004.js"></script>
<script type="text/javascript" src="js/jquery.min.js"></script>
<script type="text/javascript" src="js/bootstrap.min.js"></script>
<script type="text/javascript" src="js/jquery-ui.js"></script>
<script type="text/javascript" src="js/scripts.js"></script>
<script>(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
(i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new
Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,
m)
})(window,document,'script','//www.google-analytics.com/analytics.js','ga');
ga('create','UA-20585382-5','megadrupal.com');
ga('send','pageview');</script>
<script type="text/javascript" src="js/jquery_003.js"></script>
<script type="text/javascript" src="js/jquery_005.js"></script>
<script type="text/javascript">if($('#slider-revolution').length){
$('#slider-revolution').show().revolution({
ottedOverlay:"none",
delay:10000,
startwidth:1600,
startheight:650,
hideThumbs:200,
thumbWidth:100,
thumbHeight:50,
thumbAmount:5,
simplifyAll:"off",
navigationType:"none",
navigationArrows:"solo",
navigationStyle:"preview4",
touchenabled:"on",
onHoverStop:"on",
nextSlideOnWindowFocus:"off",
swipe_threshold: 0.7,
swipe_min_touches: 1,
drag_block_vertical: false,
parallax:"mouse",
parallaxBgFreeze:"on",
parallaxLevels:[7,4,3,2,5,4,3,2,1,0],
keyboardNavigation:"off",
navigationHAlign:"center",
navigationVAlign:"bottom",
navigationHOffset:0,

```

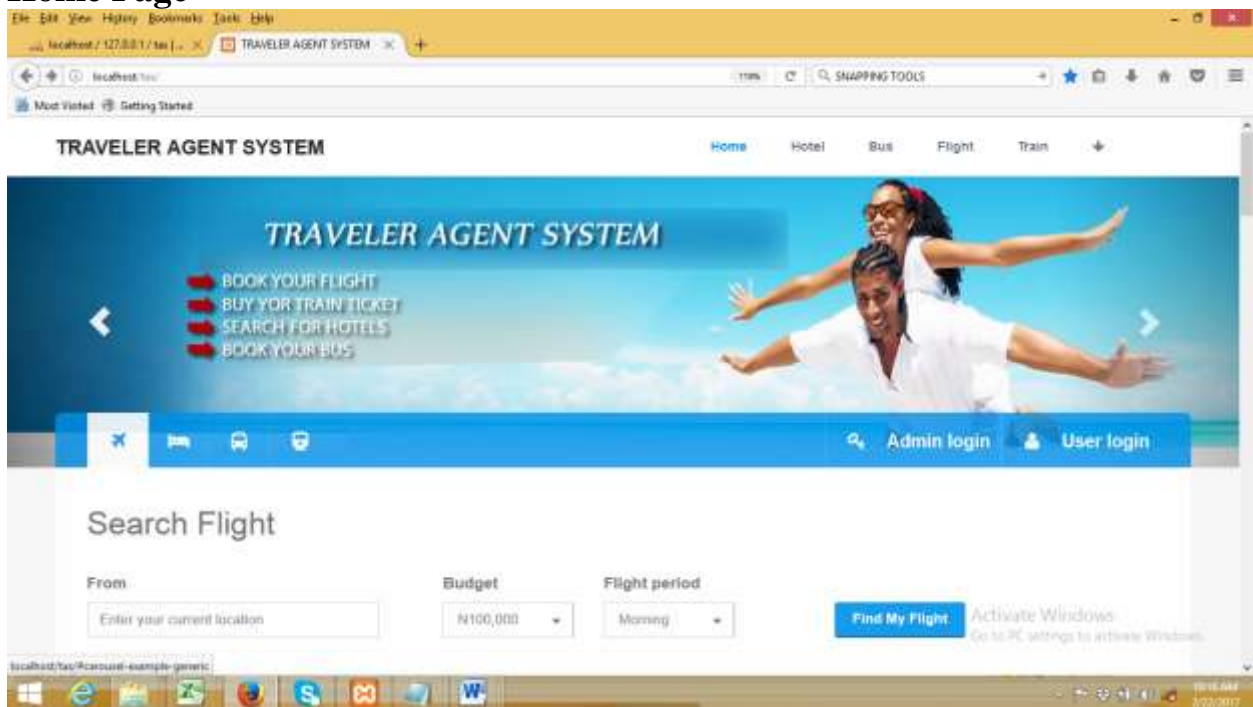
```

navigationVOffset:20,
soloArrowLeftHalign:"left",
soloArrowLeftValign:"center",
soloArrowLeftHOffset:20,
soloArrowLeftVOffset:0,
soloArrowRightHalign:"right",
soloArrowRightValign:"center",
soloArrowRightHOffset:20,
soloArrowRightVOffset:0,
shadow:0,
fullWidth:"on",
fullScreen:"off",
spinner:"spinner2",
stopLoop:"off",
stopAfterLoops:-1,
stopAtSlide:-1,
shuffle:"off",
autoHeight:"off",
forceFullWidth:"off",
hideThumbsOnMobile:"off",
hideNavDelayOnMobile:1500,
hideBulletsOnMobile:"off",
hideArrowsOnMobile:"off",
hideThumbsUnderResolution:0,
hideSliderAtLimit:0,
hideCaptionAtLimit:0,
hideAllCaptionAtLimit:0,
startWithSlide:0
});
</script>
<div id="ui-datepicker-div" class="ui-datepicker ui-widget ui-widget-content
ui-helper-clearfix ui-corner-all"></div>
</body>
</html

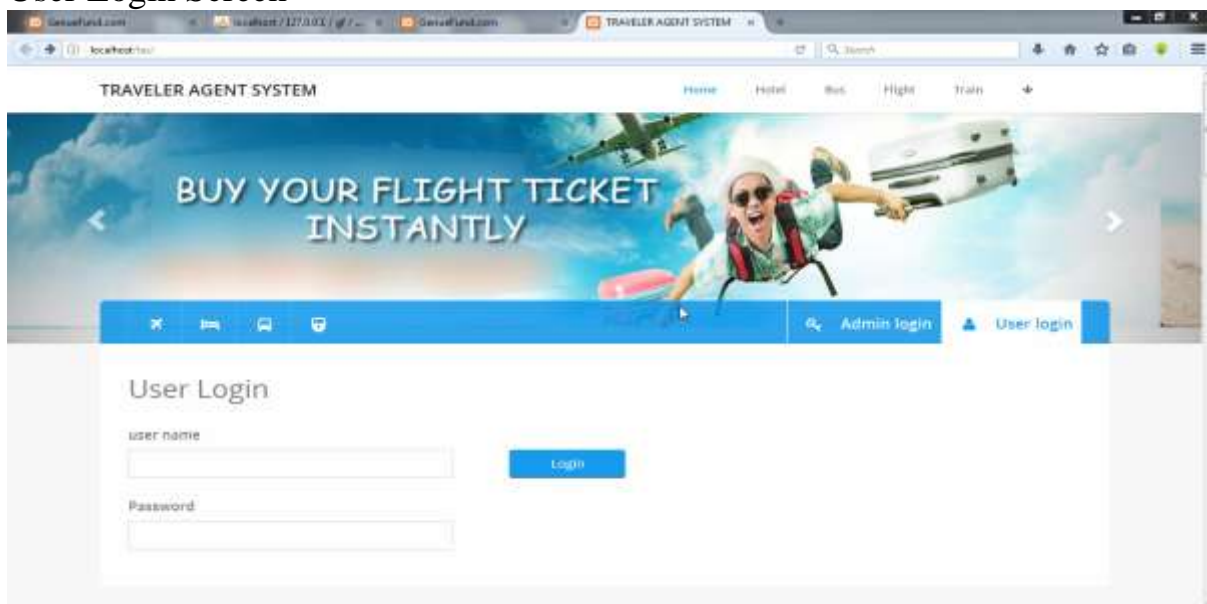
```


APPENDIX B: SAMPLE OUTPUTS.

Home Page



User Login Screen



Create an account Screen.

Create an account [Return to Home](#)

Join now and earn great rewards

SPECIAL OFFER ON
Kenya Airways
The Birth of Africa

Book a ticket to any destination and stand a chance to win a **FREE** return ticket to Kenya

No file selected. Change profile picture

First Name*

Last name*

User name*

Phone number*

Address*

City*

Country*

Email Address*

Password*

Age*

Search Flight Screen

TRAVEL AGENT SYSTEM [Home](#) [Hotel](#) [Bus](#) [Flight](#) [Train](#)

HOME [FLIGHTS](#)

Find Your Flight

Location

Check in

Check out

Budget

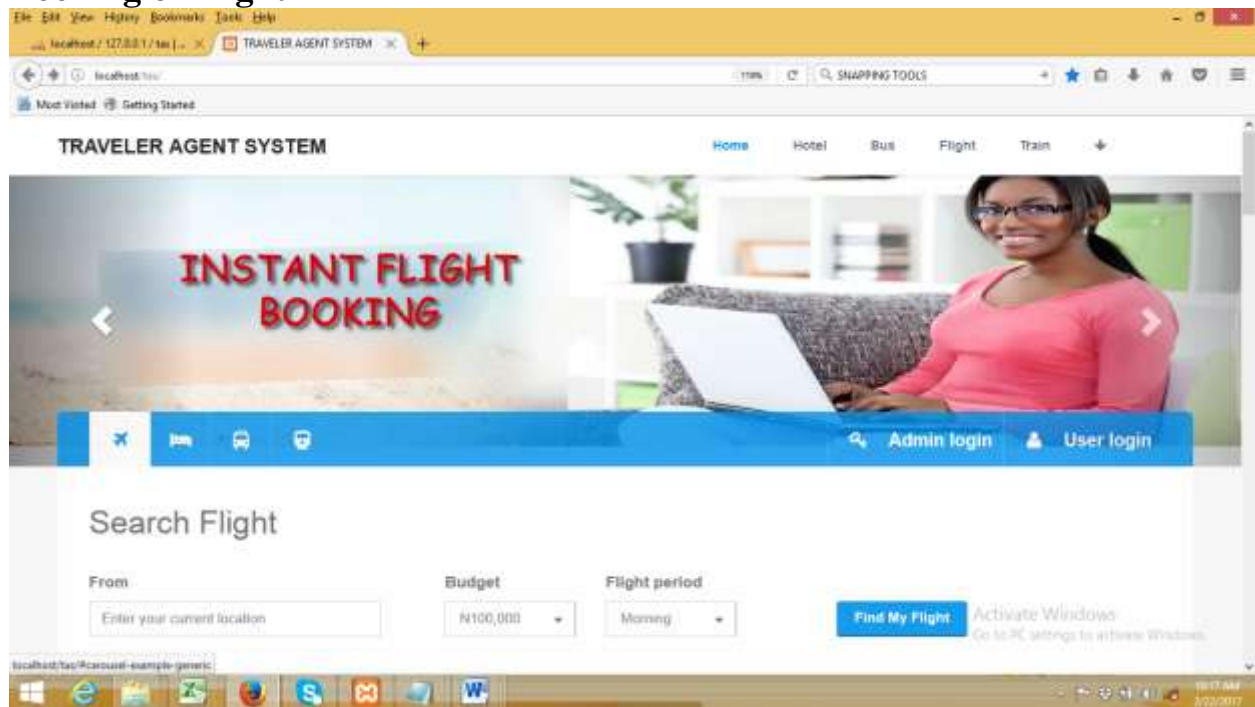
[Hotel filter](#) [Clear all](#)

Recent Hotel updates

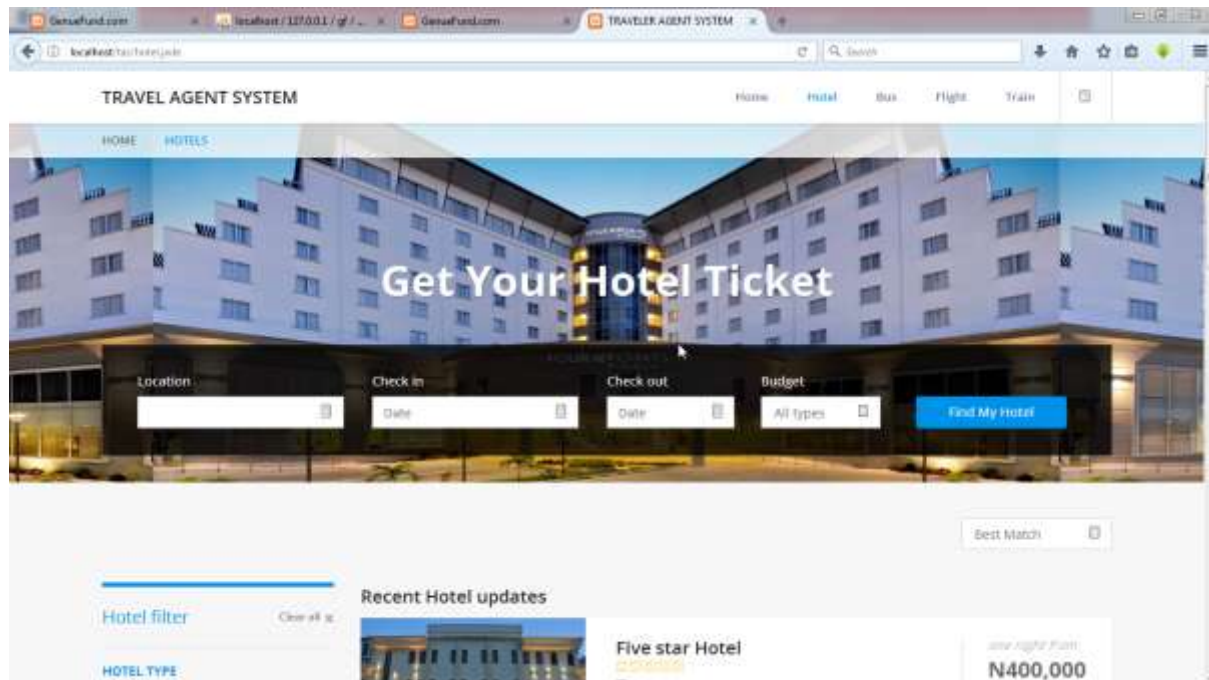
Five star Hotel

N400,000

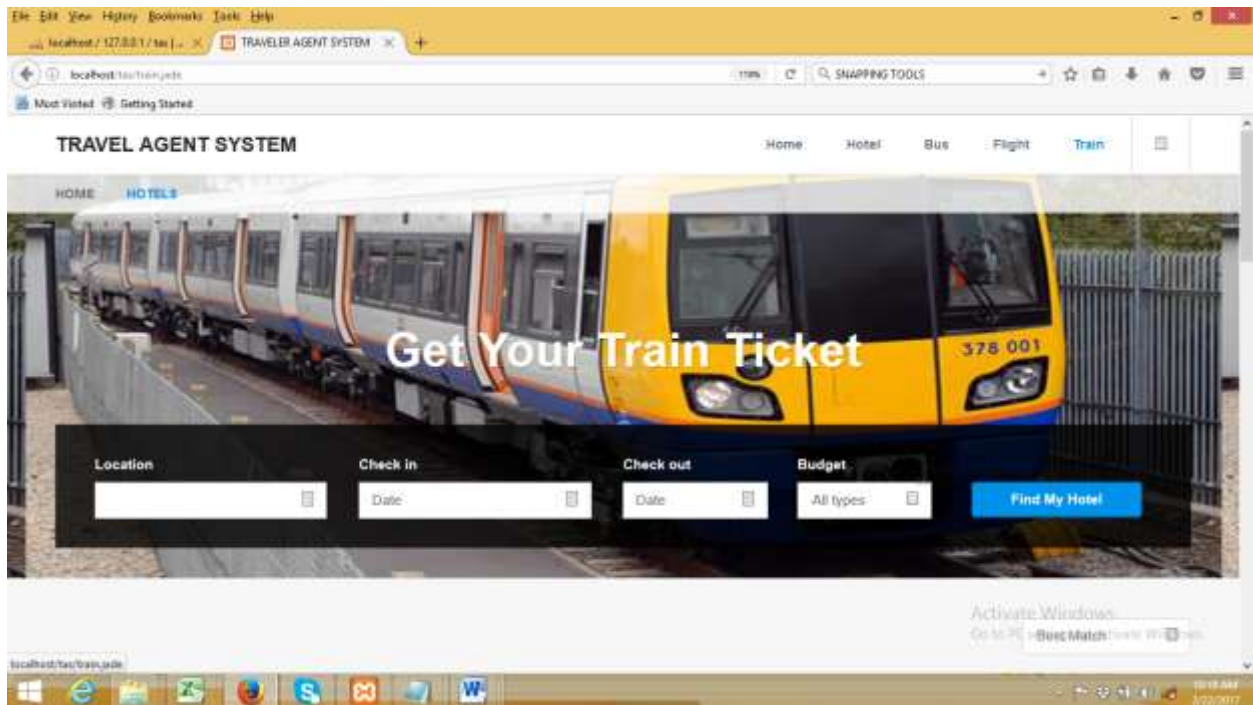
Booking of Flight



Search for Hotel Screen



Search for train.



Cancel flight prototype

