BUILDING A COLLABORATION PLATFORM USING AGENT-BASED MODELLING

Capstone Design

December 2016

Chaimae Aklibous

Supervised by Professor Mourhir Asmaa
ACKNOWLEDGEMENTS

Before concluding the journey of my four years and a half of studies, I am filled with enthusiasm and gratitude to write this note for all the people who supported and helped me to realize this capstone project. I would like first and foremost to direct my appreciation to Professor Asmaa Mourhir who was always there to answer my questions and provide guidance.

Also, I take this opportunity to express the profound gratitude from my deep heart to my beloved parents for their love and continuous support—both spiritually and materially. Also, I would like to express my special appreciation and thanks to my two friends Foutouh Najoua and KARAZI Meryam for their support. I do not think this would have been possible without them.
ABSTRACT

The purpose of this capstone project is building a collaboration platform using Agent Modelling to arrange and interpret the concepts related to the modelling of environmental issues and the development of scenarios. Motivated towards this goal, the platform shall include a set of features that will allow stakeholders or users to create models, generate scenarios, and share the results among them. Agent-based modelling represents a major tool to create complex models and run simulations, which gives specific observations about certain environmental issues. Also, it has other important uses, such as message passing, and this the feature that will be exploited within this project.
Table of content

1. INTRODUCTION........................................................................................................3
2. STEEPLE Analysis..................................................................................................4
3. Methodology ........................................................................................................5
4. Requirements Specification ...................................................................................6
   4.1. Functional requirements .................................................................................6
   4.2. Non-functional requirements .........................................................................7
5. Feasibility Study ....................................................................................................9
   5.1. Agent-Based Modeling ..................................................................................9
      Definition of criteria ..........................................................................................9
      Platforms comparison .......................................................................................10
      5.1.1. JAVA SWARM ......................................................................................10
      5.1.2. REPAST ...............................................................................................11
      5.1.3. MASON ...............................................................................................13
      5.1.4. CORMAS .......................................................................................14
      5.1.5. ANYLOGIC ......................................................................................15
      5.1.6. JIAC V ...............................................................................................16
      5.1.7. NETLOGO ..........................................................................................17
      5.1.8. JADE .................................................................................................18
      5.1.9. JAMES ...............................................................................................18
   4.1.1. Platforms Characteristics Summary ..................................................................20
   4.1.2. Conclusion ...............................................................................................21
5.2. Schedule ............................................................................................................23
6. Design and analysis ..............................................................................................23
   6.1. System Architecture .....................................................................................23
   6.2. Use Case Diagram .......................................................................................24
   6.3. Entity relationship diagram ..........................................................................24
   6.4. Sequence diagrams .....................................................................................27
   6.5. Class diagram ..............................................................................................28
7. Technology Enablers ............................................................................................30
8. Development and Implementation ........................................................................30
   8.1. Agents Message Passing .............................................................................31
      8.1.1. Structure of Jade message ....................................................................31
      8.1.2. Sending and receiving messages .............................................................32
      8.1.3. Screenshots of the application .................................................................33
9. CONCLUSION AND FUTURE WORK ................................................................38
10. REFERENCES .......................................................................................................39
**Table of figures**

| Figure 1 | Collaboration Platform System Architecture | 23 |
| Figure 2 | Use case diagram | 24 |
| Figure 3 | Entity Relationship Diagram | 25 |
| Figure 4 | Sequence diagram for authentication | 27 |
| Figure 5 | Sequence diagram for adding an FCM | 28 |
| Figure 6 | Class diagram | 29 |
| Figure 7 | Message sending method | 32 |
| Figure 8 | Message receiving method | 33 |
| Figure 9 | Login interface | 33 |
| Figure 10 | Registration interface | 34 |
| Figure 11 | Main Page interface | 34 |
| Figure 12 | Adding FCM Interface | 35 |
| Figure 13 | Viewing FCMs Interface | 35 |
| Figure 14 | Adding Scenario Interface | 36 |
| Figure 15 | Profile Interface | 36 |
| Figure 16 | Jade platform with the agents created | 37 |
| Figure 17 | sending a message through the Jade platform | 37 |
1. INTRODUCTION

Agent-based modeling is one of the best tools used in today’s world. It allows creating experiments in the form of models and running those models to give as output the representation of the interaction between the agents. Agent-based modeling can be used in different fields such as, sociology, psychology, environment, health and so on. It allows the study of models that require complex computations.

The application I am working on is a desktop one and it is a collaboration platform. This platform uses agent-based modeling and fuzzy cognitive maps (FCMs) to allow the different users to interact and share information. The main goal behind this platform is giving to the users the necessary tools to model environmental issues using FCMs and simulate and share scenarios within a group of users. Thus, the report will cover in details a feasibility study of the tools used in agent-based modeling, the requirements specifications, the steeple analysis, the design and architecture of the platform, and finally the implementation.

The collaboration platform allows creating FCMs and generation scenarios. The FCM, Fuzzy Cognitive Map, can be considered as a representation of the initial situation of the environmental issue encountered. The FCM is a signed graph that contains concepts (nodes) connected through edges where each edge or connection has a weight. If the weight is positive, it means that there is a positive relation between the concepts $C_i$ and $C_j$. When the first one increases the second one increases as well. If the weight is negative, it means that there is a negative relation. When the first concept increases, the second one decreases. If the weight is 0, it means that the two concepts have no impact on each other. After defining the FCM components in the platform, a user can define and simulate scenarios. This output shows how the related scenario can have impact on a specific environmental issue when the values of the concepts are changed.

Agent-based modeling refers to the use of computational models in order to analyze a system and understand its behavior. It identifies agents that represent entities or individuals and the interactions between them through the exchange of messages. Agent-based modeling is meant for very complex systems, in which practical reasoning does not lead to direct observations. There are many tools that are used to create models and run simulations, as I will be showing in the feasibility study.
2. STEEPLE Analysis

The purpose of this capstone project is building a collaboration platform using Agent Modelling to arrange and interpret the concepts related to the modelling of environmental issues and the development of scenarios. Therefore, the steeple analysis will give a very good insight on the different factors that the project has impact on.

Societal considerations

The collaboration platform allows different users to create scenarios and share them. These scenarios reflect how each user believes that a specific concept change of value could impact the whole situation.

Technological considerations

The application is based on existing technologies and follows the casual software engineering lifecycle.

Environmental considerations

The application will allow tackling the different environmental issues, generating scenarios, and exploring how the impacts of environmental degradation can be reduced or mitigated.

Ethical considerations:

All the features provided by the application are ethical.

Political considerations:

The platform does not have any political goals.

Legal considerations

The application does not involve a legal aspect.

Economic considerations

Since the stakeholders can cooperate through the platform, the costs related to physical meetings will be reduced.
3. Methodology

Since this capstone project involves the integration of an agent-based modeling platform called JADE with the implemented collaboration platform, a detailed study of the different tools used in agent-based modeling was required. There is a plenty of software provided to build models and generate simulations; however, not all of them are suitable to the type of application we are building. A big part of my work consisted in performing a technical feasibility by evaluating and comparing the different agent-based modeling APIs or platforms and choosing the right one.

After defining and analyzing the functional and non-functional requirements for the requirements specification phase, I proceeded with a technical feasibility study to choose the platform that is best to the project. Finally, I implemented a proof of concept to demonstrate how agent-based modeling can be used to build a collaboration application.
4. Requirements Specification

4.1. Functional requirements

User Account Management

1. User registration
   - The user will be able to register in the system by providing his first name, last name, email and password.

2. User authentication:
   - The user will be able to authenticate to the system using his email and password.

3. User sign out
   - The user will be able to disconnect his account from the application.

User Profile Management

1. User profile creation
   - Once an account is created, the user will be able see his profile and modify his personal information. The profile will include the first name, last name, phone number and email.

2. User profile modification
   - The user shall be able to modify his personal information

3. User profile deletion
   - The user can delete his profile.

Cognitive Mapping

1. Creation of a FCM
   - A user can create a Fuzzy Cognitive Map, which consists of a title, description, concepts and their weights.

2. FCMs display
   - The user can view all the FCMs created by other users.
3. Modification of an FCM
   – The user can modify an FCM he created.

4. Deletion of an FCM
   – The user can delete one or more FCMs he has added.

Scenario Management

5. Scenario creation
   – The user can define a scenario. The scenario contains a title and a list of concepts along with their activation values.

6. Scenarios display
   – The user can view all the suggested scenarios, make modifications on them and share them with the other users.

7. Scenarios comment
   – When the user creates a scenario, he/she can add a comment to the comment section. All the users will be able to see all the comments.

8. Scenarios deletion
   – The user can choose to delete one or more scenarios he has created.

4.2. Non-functional requirements

1. Performance
The application performance should be optimized throughout reducing the response time. This is achieved through locally caching the data that is frequently accessed by the users.

2. Scalability
Scalability is highly important since the application allows many users to access the same database and perform operations in parallel. The platform shall preserve a decent level of performance when the load increases at a lower cost.
3. **Integration**

The application shall integrate some external libraries, such as Jade library for agent-based modeling and JUNG for graphs creation.

4. **Security**

The application shall provide high availability of the data, make users information confidential and support integrity.
5. Feasibility Study

Building a collaboration platform using Agent-based Modelling requires choosing the right framework and library paradigm. The framework includes the necessary tools for designing and describing the ABM while the library provides an implementation of the framework along with the simulation tools. (Railsback et al. 2006) Therefore, it is substantial to make a comparison/contrast between the different platforms and toolkits that are used today in Agent-Based Modelling in order to choose the one that is more suitable to the modelling of environmental issues and the development of scenarios.

5.1. Agent-Based Modeling

The number of software platforms dedicated to performing agent-based simulations is increasing significantly, which puts more pressure on scientific researchers to choose the platform with the best features and the one that is most appropriate to the field of study. Besides, while some platforms are developing and releasing new versions every period of time, others were abandoned. Thus, when it comes to analysing these platforms, three main issues should be taken into consideration by responding to the following questions:

- Does the platform provide good and updated tools to build the models?
- Does the platform provide the necessary tools to execute and observe the simulation experiments?
- Does the platform provide a good documentation explaining how it can be used?

By answering the questions above, it will be easier to decide which platform can better arrange and interpret concepts related to environmental issues as well as support information collaboration and sharing among stakeholders. In addition, some platforms were developed for a general-purpose use while others were created to conduct experiments in some specific domains. Therefore, this criterion should also be taken into consideration when comparing existing platforms knowing that our field of research is oriented towards solving environmental issues. In the coming sections, ABM Platforms will be evaluated based on their characteristics, usability, operating ability, and security.

Definition of criteria
• **USABILITY**

Usability in the domain of agent-based modelling refers to the ability of the platform to develop models and generate simulations. It can be evaluated through two main characteristics, simplicity and scalability. Simplicity is the ease of understanding the language the platform uses and how fast any user could learn it. Scalability, on the other hand, reflects the ability of the platform to handle the growing amount of work and find optimal solutions to any possible overloads (Le Page et al. 2012).

• **OPERATING ABILITY**

The operating ability is an essential criterion, which is concerned by the execution process of the platform. Two main characteristics were taken into consideration while evaluating the different platforms, robustness and stability. Robustness copes with error detection and recovery during the execution of a model. Thus, a good platform is the one that provides tools for testing and detecting bugs in the code. Stability refers to how the output of a program changes when the environment changes and how the quality is affected when modifications are made (Le Page et al. 2012).

• **SECURITY**

Security management is very important when evaluating an agent-based modelling platform. It reflects whether the platform is safe or not and to what extent it can deal with security issues. Very good platforms provide protocols and mechanisms to enhance the security of the models (Le Page et al. 2012).

**Platforms comparison**

**5.1.1. JAVA SWARM**

Swarm is one of the first ABM platforms developed, which was created by Santa Fe Institute, and it is the basis of many current ABM platforms. (Berryman, 2008) Swarm was originally implemented using Objective-C and it is best used for hierarchical models, which require “the use of multi-level feedback between agents, groups of agents, and the environment”. (Berryman, 2008) Since the Objective-C version of Swarm was not easy to use especially by inexperienced programmers, a Java implementation was provided later on to satisfy Java users. The Java Swarm mainly passes messages to the Objective-C library through Java. (Railsback et al. 2006)
ADVANTAGES
• Very fast for models with complex computations and less method calls
• An IDE can be used to write, debug, and manage libraries

LIMITATIONS
• It’s difficult to debug the errors occurring in the Objective-C libraries (Railsback et al. 2006).
• Very slow for models with less computations and more method calls
• Lack of documentation

APPLICATIONS
• Social science
Swarm was used in an agent-based experiment in the field of negotiation and exchange. The experiment shows how prices change in a model where vendors and consumers interact (Terna).

• Traffic management
Swarm was used as a tool to solve the problem of traffic congestion. An algorithm called ACO, based on Swarm, was used to build a model and run a simulation for the traffic signal control. The results showed that this later could be very effective in regulating the traffic flow and reduce congestion (Renfrew et al, 2009).

• Health Care
The purpose of this application is creating a prototype for the agent-based distributed medical system. In this model, agents are represented by both individuals and organizations, which take part in the patient’s care process. These agents have a knowledge system with their skills and a communication manager (Jemal et. al, 2015).

5.1.2. REPAST

Repast -Recursive Porous Agent Simulation Toolkit- was firstly developed at the computer lab of Chicago Social Science Research University (Bassiliades & Kravari, 2015). It is an open source platform providing different implementations using different programming languages. Repast was firstly created to provide a Swarm platform based on Java and it was mainly dedicated to researchers in the field of social sciences; however, it can be used to implement models in other areas (Railsback et al. 2006).
ADVANTAGES
- Easy to use even for unexperienced users
- Allows resetting and restarting models from the UI
- Good execution speed compared to other platforms
- There are two editions of Repast and each edition provides different ways of writing models in order to cover many cases
- Simple models with little experience in programming can be implemented with the help of the RepastPy facility
- 2D and 3D models can be created using Repast and Repast Symphony
- The latest version of Repast allows using Colt Java libraries for statistics
- Repast Symphony includes the use of R statistical suite (Berryman, 2008).

LIMITATIONS
- It does not provide a method which allows executing actions in a random order over agents
- Lack of documentation

APPLICATIONS
- Pedestrian movement
  The purpose of this research is studying the factors that affect the pedestrian movement. The model implemented included two main parts. The first part is related to land value and use in informal shopping centres in addition to rent rates. The second part tackled the impact of pedestrians on land use due to decisions, such as trip’s offers or demands (Castle et. al).

- Urban dynamics
  A research related to urban dynamics in Latin American cities was conducted using Repast. The purpose of the research is exploring how urban growth impacts the levels of poverty is those cities. Many agent-based simulations were developed to support this study (Castle et. al).

- Segregation
  Segregation was studied through exploring how personal preferences and tastes can be a trigger for social inequality within neighbourhoods. The model implemented considers agents as virtual households that possess ethnic status and ability to search and make residential choices (Crooks).
A research was conducted to test whether children can be allowed to go out without being accompanied by an adult. The purpose of the research is to examine how children use the local environment and how this environment affects their special behaviour. An agent-based model was implemented, using Repast, to study the children’s special movements (Castle et. al).

5.1.3. MASON

MASON - Multi-agent Simulator Of Neighbourhoods/Networks - is a general-purpose Java library for ABMs, which was created by George Mason University's Evolutionary Computation Laboratory and the GMU Centre for Social Complexity. The main goal behind developing MASON is having a good alternative to Repast, which is characterized by high speed and good performance (Railsback et al. 2006). MASON is dedicated to creating models that are complex and require many computations.

ADVANTAGES

- Very fast for models that require executing agents over many iterations with complex computations
- The ability to attach and re-attach graphical interfaces
- The ability to pause simulations and move it among computers
- The ability to realize 2D and 3D visualization

LIMITATIONS

- The platform only provides very basic features. For example, graphing and random numbers distributions are not implemented yet.
- The terminal window cannot be accessed for debugging when using Eclipse IDE

APPLICATIONS

- Network Intrusion and Countermeasures

The purpose of the network intrusion model is studying the security of a network and exploring the issues related to it. The model is composed of two types agents, computers, which are connected by an IP address space and a remote login space, and hackers. (Luke et. al, 2005) Each computer is protected by a set of security policies. The simulation of this model defines whether the system is safe or threatened by the hackers, when security measures change. (Luke et. al, 2005)
• Urban Traffic Simulation

The aim of this simulation is facilitating the traffic movement during rush hours and sudden “unexpected floods of traffic” (Luke et. al, 2005). The model is based on a graph, where nodes road intersections represent nodes and roads represent edges. The results of this simulation were used to examine other environments, such as packet routing (Luke et. al, 2005).

• Anthrax Propagation in the Human Body

Inhalation anthrax is a deadly infection; therefore, live laboratory studies are not possible in this case. Implementing a model to study how inhalation anthrax spreads in the human body became a necessity. The human organs that can be affected by this disease represent the agents in the model. At the end, the simulation shows how inhalation anthrax flows through the body throughout time. (Luke et. al, 2005)

5.1.4. CORMAS

Cormas -Common-pool Resources and Multi-Agent Simulations- is a purely object-oriented platform based on Visual Works programming environment. It was built using Smalltalk to mainly tackle natural resource issues (Le Page et al. 2012). The sole purpose of CORMAS is “representing the interactions between stakeholders about the use of natural renewable resources.” (Kravari & Bassiliades, 2015). This platform provides all the necessary resources to implement agent-based models and analyse the simulations.

ADVANTAGES

• The logic of the platform is easy to understand
• It provides a built-in method to shuffle the list of agents (Bajracharya & Duboz 2013)
• It provides a well-organized structure that shows the user how to build the model starting from implementing the model up to running the simulation
• It provides a live inspector window, which gives hints about the bugs and how to solve them.

LIMITATIONS

• It is not easy to control the scheduling order of actions in Cormas

APPLICATIONS

• Water management
The implemented model evaluates the impact of rainfall on the spatiotemporal distribution of water resources. It was mainly designed for water storage basins, which serve large sectors. The output of the model gives an idea about water availability and use (Cormas website).

- Economic exchange (Market performance)
A study related to stockbreeding wastewater in Reunion Island, was conducted to define the pollution problems this activity can provoke. The goal of the model implemented is to demonstrate the efficiency of the different scenarios proposed in managing wastewater (Cormas website).

- Cellular automata - Fire
The model Fire Automata studies a major environmental problem, which is the spread of fire in the forest. The spatial entities of this model are represented by a FireAutomata_cell that can take four different states. The color red represents the fire, green represents the tree, grey represents the ash, and white represents emptiness. Every cell in the model moves from one state to the other throughout time. (Cormas website)

5.1.5. ANYLOGIC

The name AnyLogic was inferred from the ability of this platform to support three different methodologies in the same model: system dynamics, discrete-event, and agent-based modelling. The fact that these three approaches can be used to implement a single model, allows the user to create complex models in different areas (AnyLogic Website). AnyLogic is based on the Java development environment and therefore uses the object-oriented design paradigm, which provides an organized structure for creating models.

ADVANTAGES
- AnyLogic is considered to be the only system, which supports implementing models and simulations using three methodologies: system dynamics, discrete-event, and agent-based modelling.
- It provides many tools and libraries that help implementing models in different areas.
- AnyLogic provides some design patterns, which make the development of agent-based model easier.
- It supports 3D animations
• AnyLogic models can be exported as Java standalone applications and can run on any machine without the use of AnyLogic software (AnyLogic Website).
• External libraries and data sources can be used to support and extend AnyLogic tools.
• Excellent documentation and support services are provided to the user of AnyLogic to facilitate the task of implementing models and running simulations.
• Supports Windows, Mac, and Linux operating systems.

APPLICATONS

• Healthcare

Cardinal is a healthcare supply chain, which manages products of different brands and distributes them over 30000 locations. Using agent-based modelling, Cardinal has implemented a model with AnyLogic that is concerned with the activities of the employees and their interactions with one another. The solution allows “capturing work time allocation, congestion wait time, cycle times, distance travelled, worker variability and other important metrics.” (Anylogic Website).

• Food security

An international civil agency used AnyLogic to build a model, which explores the impacts of food insecurity. The main goal of the model is examining how mitigation strategies can be used to eliminate the impact of food insecurity on humans in the developing countries (Anylogic Website).

5.1.6. JIAC V

JIAC - Java-based Intelligent Agent Componentware- is a framework that supports the full lifecycle of designing, implementing, and deploying agent-based models.

ADVANTAGES

• Supports the reuse and modification of models
• Contains libraries with ready-to-use components, which can be integrated in the building of the models
• JIAC is based on ActiveMQ architecture, which facilitates the distribution of agents over a network (Lützenberger et. al 2014)
• Modifications on the model can be performed during runtime

APPLICATONS

• NESS
NESS is a network security simulator, which was developed to check the security of computer networks as well as other networks, such as the power grid and the ICT networks.

- Driver simulation

A model was implemented using JIAC to “simulate a special form of human traffic behaviour, namely strategic compensatory behaviour” (Lützenberger et. al, 2013).

5.1.7. NETLOGO

NetLogo – Network Logo- is a functional programming language and is considered one of the best educational tools. It provides a structure, which allows separating between the implementation of models and their graphical display (Railsback et al. 2006). NetLogo includes a library with models for different domains, such as economics, biology, physics, chemistry, psychology and system dynamics. These models can be modified and adapted to one’s personal needs (Kravari & Bassiliades, 2015).

DEFINITION OF HUBNET

HubNet is an application that is defined as a “technology that lets you use NetLogo to run participatory simulations in the classroom. In a participatory simulation, a whole class takes part in enacting the behaviour of a system as each student controls a part of the system by using an individual device, such as a networked computer or Texas Instruments graphing calculator.” (NetLogo documentation).

ADVANTAGES
- This platform provides high level structures that minimize the coding process and makes of it easy to use
- A very good documentation of the platform is provided to the users
- An errors checker is provided
- The ability to realize 2D and 3D visualization
- The ability to make modifications on existing models
- Running participatory models using HubNet

LIMITATIONS
- The whole code is saved in one file
- Some NetLogo models are not easy to extend
- NetLogo does not provide access to the algorithms used to implement its primitives
- NetLogo is not based on object-oriented which makes it difficult to read and understand
5.1.8. **JADE**

JADE -Java Agent Development Framework- is a software framework fully implemented in Java and which works in compliance with FIPA\(^1\) specifications (Bellifemine et. al 2007). JADE was distributed in 2000 by Telecom Italia as an open source with the goal of providing a service to developers without knowledge of the FIPA specifications (Bellifemine et. al 2007).

**ADVANTAGES**
- It is possible to distribute the agent-platform over machines, which do not share the same operating system.
- The configuration over the model built can be changed during run-time
- JADE can be used on devices with limited resources since the minimal Java requirement is the version 1.2
- JADE works in compliance with FIPA specifications
- JADE is a distributed system where each agent is considered as a separate thread
- It provides useful tools for debugging and monitoring agents’ execution.

**LIMITATIONS**
- JADE has some limitations when it comes to deploying it on mobile devices. They can be summed up in hardware limitations, Java limitations (Java virtual machine), and Network limitations.

5.1.9. **JAMES**

JAMES II is a Java framework for developing models and simulations. It is based on a plugin architecture, which means that any modeling technique can be integrated into the framework through plugins. (Le Page et al. 2012) The main goal of JAMES II platform is increasing efficiency through providing solid tools and algorithms. Also, users can choose from within a variety of plugins and select the functionalities they want to apply (Le Page et al. 2012).

**ADVANTAGES**

---

\(^1\) "The Foundation for Intelligent Physical Agents (FIPA) [5] is an international non-profit association of companies and organizations sharing the effort to produce specifications of generic agent technologies. FIPA is envisaged not just as a technology for one application but as generic technologies for different application areas, and not just as independent technologies but as a set of basic technologies that can be integrated by developers to make complex systems with a high degree of interoperability." (Bellifemine et. al 2001)
• There is a separation between the model and the simulator, which facilitates the change of simulators during a simulation run in case there is one that is more efficient (Himmelspach et al. 2008).
• Many code samples are provided to help the user start building the model and the simulation.
• Multiple simulations can be run in parallel
### Platforms Characteristics Summary

<table>
<thead>
<tr>
<th>Primary domain</th>
<th>JAVA SWARM</th>
<th>REPAST</th>
<th>MASON</th>
<th>NETLOGO</th>
<th>CORMAS</th>
<th>JADE</th>
<th>AnyLogic</th>
<th>JIAC</th>
<th>JAMES II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary domain</td>
<td>General-purpose agent based/Natural resources &amp; environment</td>
<td>Biological &amp; social studies/General-purpose agent based</td>
<td>General-purpose agent based simulations</td>
<td>General-purpose agent based simulations</td>
<td>Natural resources and agent-based simulations</td>
<td>Distributed application s composed of autonomou s entities</td>
<td>General purpose, distribute d agent based simulatio ns</td>
<td>Large-scale distribut ed systems</td>
<td>General purpose agent based modelling and simulations</td>
</tr>
<tr>
<td>Usability</td>
<td>Simplicity Average</td>
<td>Simple</td>
<td>Average</td>
<td>Simple</td>
<td>Simple</td>
<td>Simple</td>
<td>Average</td>
<td>Average</td>
<td>Simple</td>
</tr>
<tr>
<td></td>
<td>Scailability Average</td>
<td>Good</td>
<td>Average</td>
<td>Good</td>
<td>Good</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Operating ability</td>
<td>Stability Average</td>
<td>High</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>High</td>
<td>High</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td></td>
<td>Robustness Low</td>
<td>High</td>
<td>Average</td>
<td>Average</td>
<td>Good</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Security</td>
<td>Weak</td>
<td>Average</td>
<td>Average</td>
<td>Strong</td>
<td>Strong</td>
<td>Strong</td>
<td>Average</td>
<td>Average</td>
<td>Average</td>
</tr>
<tr>
<td>Programming language</td>
<td>Java</td>
<td>Lisp/ Prolog</td>
<td>Java</td>
<td>Logo programming language</td>
<td>SmallTalk</td>
<td>Java</td>
<td>Java</td>
<td>Java</td>
<td>Java</td>
</tr>
<tr>
<td>Web compatibility</td>
<td>Not offered</td>
<td>Not offered</td>
<td>Not offered</td>
<td>Offered</td>
<td>Not offered</td>
<td>Offered</td>
<td>Not offered</td>
<td>Not offered</td>
<td>Not offered</td>
</tr>
<tr>
<td>License</td>
<td>Open source</td>
<td>Open source</td>
<td>Open source</td>
<td>Open source</td>
<td>Open source</td>
<td>Open source</td>
<td>Not free</td>
<td>Free</td>
<td>Free</td>
</tr>
</tbody>
</table>
4.1.2. Conclusion

Each Agent-Based Modelling platform has its own advantages as well as limitations. Yet, these variations between the platforms determine the quality of models and simulations generated. The fact that some platforms are limited to some specific domains impacts the process of creating models and affects other characteristics, such as execution speed and errors checking. Furthermore, even though some platforms provide excellent features/tools for agent-based modelling, the programming language they require is either complicated or not used anymore by new technologies, which might impact the process of developing a model.

In this project, the main criteria that should be taken into consideration to decide which platform should be adopted are: security, performance, and programming language. The goal of the project is basically building a collaboration platform to analyse and interpret environmental issues and allow sharing of information among stakeholders. Therefore, the toolkits chosen should take into consideration the criteria mentioned earlier altogether. So, after comparing all the various criteria, we ended up with two mother platforms, NetLogo and JADE. Both platforms provide tools to be integrated in a Web system with good documentation and tutorials. They are both open source and easy to use. However, if we compare security and the operating ability of each one of them, JADE tends to be more powerful with a strong base (FIPA compliant). It provides many tools to enhance the security of the programs implemented and improve their performance. For example, there is a security add-on for JADE, which provides features, such as end-to-end signature and encryption. Therefore, the platform chosen, and which satisfies most of the project requirements, is JADE.

For the Fuzzy Cognitive Maps creation, the only available API is the JFCM. Unfortunately, this library was not updated since 2014 and the code it provides is not well structured and it is difficult to extend. Therefore, I have used a library that was implemented by my professor Asmaa Mourhir to generate FCMs. Also, I have used the JUNG library to generate graphs for the scenarios. JUNG-Java Universal Network Graph- is a java library, which allows the modelling and visualisation of data that can be represented as a graph (JUNG website).

For the database creation, I chose to work with PostgreSQL mainly because I faced problems while installing the Microsoft SQL server and running it on my laptop. The best
alternative was PostgrSQL, which is also an object-relational database and the default one for the mac operating systems. This DBMS is known for being very powerful since it provides high reliability, data integrity, and correctness.
5.2. Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Dates</th>
<th>Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>29/8-2/9</td>
<td>Project selection</td>
</tr>
<tr>
<td>2</td>
<td>5-9/9</td>
<td>Project initial specification</td>
</tr>
<tr>
<td>3-5</td>
<td>12-30/10</td>
<td>Feasibility study and analysis</td>
</tr>
<tr>
<td>6-7</td>
<td>3-14/10</td>
<td>Design and architecture of the system</td>
</tr>
<tr>
<td>7-11</td>
<td>17/10-11/11</td>
<td>Development of the application</td>
</tr>
<tr>
<td>12</td>
<td>14-18/11</td>
<td>Testing</td>
</tr>
<tr>
<td>13</td>
<td>23-27/11</td>
<td>Final report</td>
</tr>
<tr>
<td>14</td>
<td>28/11-2/12</td>
<td>Project defense</td>
</tr>
<tr>
<td>15</td>
<td>5-9/12</td>
<td>Updated final report</td>
</tr>
</tbody>
</table>

6. Design and analysis

6.1. System Architecture

The system architecture is based on a two-tier architecture. The client application interacts directly with the database server using JDBC. The first tier, which is the client’s application, contains the user interface and the business logic. The second tier is the database server where data of the application can be stored.

![Collaboration Platform System Architecture](image)

Figure 1: Collaboration Platform System Architecture
6.2. Use Case Diagram

The use case diagram gives a general description of the system components and features. In this application, there is only one actor who is the user and who can perform different operations provided by the system once registered and logged in.

Figure 2: Use case diagram

6.3. Entity relationship diagram
Figure 3: Entity Relationship Diagram
The table below provides the description for each entity and its attributes:

<table>
<thead>
<tr>
<th>Entity Name</th>
<th>Attribute</th>
<th>Attribute description</th>
<th>Entity description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User_db</td>
<td>id</td>
<td>The user identifier</td>
<td>The user entity is where information about users of the application is stored.</td>
</tr>
<tr>
<td></td>
<td>First_name</td>
<td>The first name of the user</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last_name</td>
<td>The last name of the user</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Email</td>
<td>The email of the user</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Password</td>
<td>The password of the user</td>
<td></td>
</tr>
<tr>
<td>FcmTable</td>
<td>id</td>
<td>The fcm identifier</td>
<td>The fcm table entity is used to keep track of all the fcms created by the users.</td>
</tr>
<tr>
<td></td>
<td>Fcm_title</td>
<td>The title of the fcm created</td>
<td></td>
</tr>
<tr>
<td></td>
<td>User_id</td>
<td>The id of the user who added the fcm.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>description</td>
<td>The description of the fcm created</td>
<td></td>
</tr>
<tr>
<td>FcmElements</td>
<td>id</td>
<td>The identifier of the element added</td>
<td>The fcm_elements entity includes all the concepts along with their details added for a specific FCM.</td>
</tr>
<tr>
<td></td>
<td>Fcm_id</td>
<td>Foreign key references the id of the FCM table to which the element belongs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Concept_title</td>
<td>The title given to a concept</td>
<td></td>
</tr>
<tr>
<td></td>
<td>concept</td>
<td>The concept identifier</td>
<td></td>
</tr>
<tr>
<td>Fcm_concepts</td>
<td>Id</td>
<td>Identifier of the concept</td>
<td>The fcm_concepts entity defines all the concepts linked to one another.</td>
</tr>
<tr>
<td></td>
<td>Fcmelement_id</td>
<td>References the id of the FCM concept in the fcmelements table</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Link_concept</td>
<td>All the concepts linked to a specific concept.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>weight</td>
<td>The weight of the connection between the two concepts</td>
<td></td>
</tr>
<tr>
<td>ScenarioTable</td>
<td>id</td>
<td>Unique identifier for each concept added</td>
<td>The scenarioTable contains all the scenarios created by the users.</td>
</tr>
<tr>
<td></td>
<td>fcmid</td>
<td>Foreign key references the id of the FCM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>User_id</td>
<td>Foreign key references the user who added the scenario</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scenario_title</td>
<td>The title given to a scenario</td>
<td></td>
</tr>
<tr>
<td>Scenario_elements</td>
<td>id</td>
<td>The identifier of a scenario element</td>
<td></td>
</tr>
</tbody>
</table>
## 6.4. Sequence diagrams

The sequence diagram mainly shows the interaction between objects.

![Sequence diagram for authentication](image)

**Figure 4**: Sequence diagram for authentication
Figure 5: Sequence diagram for adding an FCM

6.5. Class diagram

The class diagram represents the different classes implemented in the project. Each class defines a set of attributes and methods and has relationships with one or more classes. The figure below illustrates the class diagram for the Collaboration Platform.
7. Technology Enablers

<table>
<thead>
<tr>
<th><strong>Java</strong></th>
<th>Java, which is an Object Oriented Programming language, was used to implement the application and the business logic.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NetBeans IDE 8.1</strong></td>
<td>NetBeans IDE 8.1, which is a software development platform written in Java, was used to implement the application and the business logic.</td>
</tr>
<tr>
<td><strong>PostgreSQL</strong></td>
<td>PostgreSQL is an object-relational database, which allows securely storing and retrieving data. It was used to create and populate the tables needed in the database.</td>
</tr>
<tr>
<td><strong>PSequel</strong></td>
<td>PSequel was the IDE used to implement the database. It provides a simple interface that allows performing PostgreSQL tasks.</td>
</tr>
<tr>
<td><strong>Jade</strong></td>
<td>Jade is a software framework fully implemented in Java. It allows the implementation of multi-agent systems.</td>
</tr>
</tbody>
</table>

8. Development and Implementation

The implementation phase of the application was split into two major parts. The first part was about designing a user interface for the different views the application provides and the second part was about connecting the application to the database. Since the application is about building a Collaboration Platform, the first required step is creating a shared database, which will contain the set of tables to store the users’ data, the Fuzzy Cognitive Maps information, and the scenarios generated.

As I mentioned earlier, I have used PostgreSQL to create the database and the tables. Unfortunately PostgreSQL does not provide an interface to perform the needed operations, only the command line can be used to do so. Therefore, I installed PSequel, which offers a simple and easy-to-use GUI for creating/deleting and displaying tables in a database.
For the multi-agents part, the main advantage provided by the platform Jade is the ability to make agents residing in different platforms communicate and share messages. This feature is very important in our project because we need the different stakeholders using the platform to share the output of the simulations generated and exchange scenarios and comments about them.

8.1. Agents Message Passing

To achieve the full power of multi-agents systems, it is necessary for agents to communicate and interact. This communication is done through the exchange of messages, which have to follow a specific format and semantics agreed upon between agents. In Jade, a standard called ACL, Agent Communication Language, is followed and which allows the transmission of a set of knowledge expressed in a content language. The content of the message can be either treated as a serialized Java object using ontologies or simply use a String format.

8.1.1. Structure of Jade message

As I mentioned earlier, Jade follows a specific message format for the communication of agents. This message format can have one or more of the following attributes:

- **Persormative**: this attribute specifies the type of the communication or the FIPA message type and it is the only required field. This field can take many different values, such as INFORM, QUERY, PROPOSE, REQUEST and so on.
- **Addressing**: this field allows the definition of the IDs of the communicating agents. One ID is specified for the sending agent, and one or more IDs are specified for receiving agents.
- **Content**: the content of the message
- **ConversationID**: this attribute is used to link the messages that belong to the same conversation.
- **Language**: this field specifies the language of the messages exchanged
- **Ontology**: specifies the ontology that is used in the content; it gives meaning to the expressions used.
- **Protocol**: identifies the recipient of a message reply
8.1.2. Sending and receiving messages

The exchange of messages between agents in Jade is performed by the FIPA-ACL standard. This operation is asynchronous where each agent has a unique queue for the incoming messages. To send a message, the first step is identifying the AID of the destination agent, then calling the two methods addReceiver() and send() to complete the operation.

Figure 7: Message sending method

To receive a message, the receive() method is called. This method gets the first message from the queue or returns null if the queue is empty. In this case the receiver blocks waiting for the message to arrive to the queue.
public class SecondAgent extends Agent {

    @Override
    protected void setup()
    {
        addBehaviour(new OneShotBehaviour()
            @Override
            public void action()
            {
                //Receive message from agent
                ACLMessage msg = receive();
                if(msg != null){
                    // Get the content of the message
                    String text = msg.getContent();
                } // if the queue is empty, the receiver blocks waiting for the message
                else { block(); }
            });
    }
}

**Figure 8: Message receiving method**

### 8.1.3. Screenshots of the application

![Collaboration Platform](image)

**Figure 9: Login interface**
Figure 10: Registration interface

Figure 11: Main Page interface
**Figure 12: Adding FCM Interface**

![Collaboration Platform](image)

**Figure 13: Viewing FCMs Interface**

![Collaboration Platform](image)
Figure 14: Adding Scenario Interface

Figure 15: Profile Interface
Figure 16: Jade platform with the agents created

Figure 17: Sending a message through the Jade platform
9. CONCLUSION AND FUTURE WORK

The main mission behind my capstone project is, first building a collaboration platform and second integrating a multi-agents software with the platform to allow exchanging of messages. I have used different tools for the implementation of the two parts. For the platform, I used Java to build the classes and the interfaces and PostgreSQL to create the shared database. For the multi-agents part, I used Jade software to create agents and the communication between them.

The collaboration platform I created provides many good features, which facilitate the tasks of stakeholders when it comes to discussing and sharing issues related to the environment. They can add Fuzzy Cognitive maps, and share them with the platform users. Also, it is easy for any stakeholder to create and run his/her own scenario based on an existing FCM and share the output with the others. Also, it is possible to add comments for a specific scenario that others can see and react to.

The first and the original idea of my capstone project is building an e-collaboration platform using agent based modeling. I was supposed to build a web application providing the same features as the desktop one; however, I could not achieve that within the scope of my capstone. Therefore, as a future work, I will be investigating the tools used in building web applications and find the ones that are more suitable to my project. Also, I will be learning more about multi-agents platforms and figure out how they are integrated with online websites.
10. REFERENCES


