Command-line and Visual Simulation of Kitchen Domain

Zheng Huang, Ganesh Arunachalam and Ranjani Nagarajan
{hz, ganesha, r.nagarajan}@pgrad.unimelb.edu.au

Abstract

In this paper, we attempt to simulate a Kitchen domain providing a visual as well as command-line interface. The main focus of this work is to implement the agents as realistically as possible adhering to the design developed as a part of previous project. The resources and environments have also been modelled and implemented as a part of this work. The command-line simulator is implemented in Java/JADE and the visual simulator has been implemented using GDI+ API on the .NET platform to allow simulation of Kitchen Domain (including all activities of agents and messages exchanged between agents). The visual simulator application has been designed to run the simulation at a later time based on the simulation log generated by the Kitchen Domain command-line simulation written using JADE on Java.

1 Introduction

The objective of our project is to simulate/implement cooking domain. The implementation captures the communication model between the different agents apart from modelling their behaviour. It also implements various resources and environments involved. The command-line simulation application has been implemented using agent programming platform -Java Agent Development Framework (JADE) [1, 2, 3, 4].

The visual simulator application has been implemented using the GDI+ library on .NET. This application may load XML files generated by real agent programming language and represent the raw data with animated agent roles. The format of XML file must comply with the format requested by the application. Alternatively, it may also produce data by its own agent simulator.

2 Design

For the sake of clarity, the goal model of the design has been provided in Figure 1. The sections 2.1 - 2.4 describe the various aspects of the domain being modelled and simulated in this paper.

2.1 Agents

The various agents in our system are:

1. **Kitchen Manager (KM):** This agent is responsible for organizing the dinner, deciding the menu and interacting with the guests. Once all the guests leave it closes down the restaurant.

2. **Cook (CK):** This agent is responsible for cooking the various dishes. It interacts with Kitchen Helper and Cleaner. It requests the Kitchen Helper for preparing raw materials and the cleaner for cleaning utensils. Based on the required amount of dishes to be prepared, this agent chooses one at a time (starting from Entrée to Main-Course to Dessert) and prepares and sends it to the prepared dishes. The synchronization between cooks is achieved by exchanging information about what a particular cook is cooking currently.
3. **Kitchen Helper (KH):** This agent caters to needs of the cooks except for cleaning utensils. It finishes preparing a particular raw material it informs the cooks about the same. It also informs the cleaners about the dirty utensils.
4. **Cleaner (CR):** This agent is responsible for cleaning utensils. It receives messages from the kitchen helper and cooks for cleaning utensils. It cleans the utensils in the dirty utensils and put it to clean utensils.

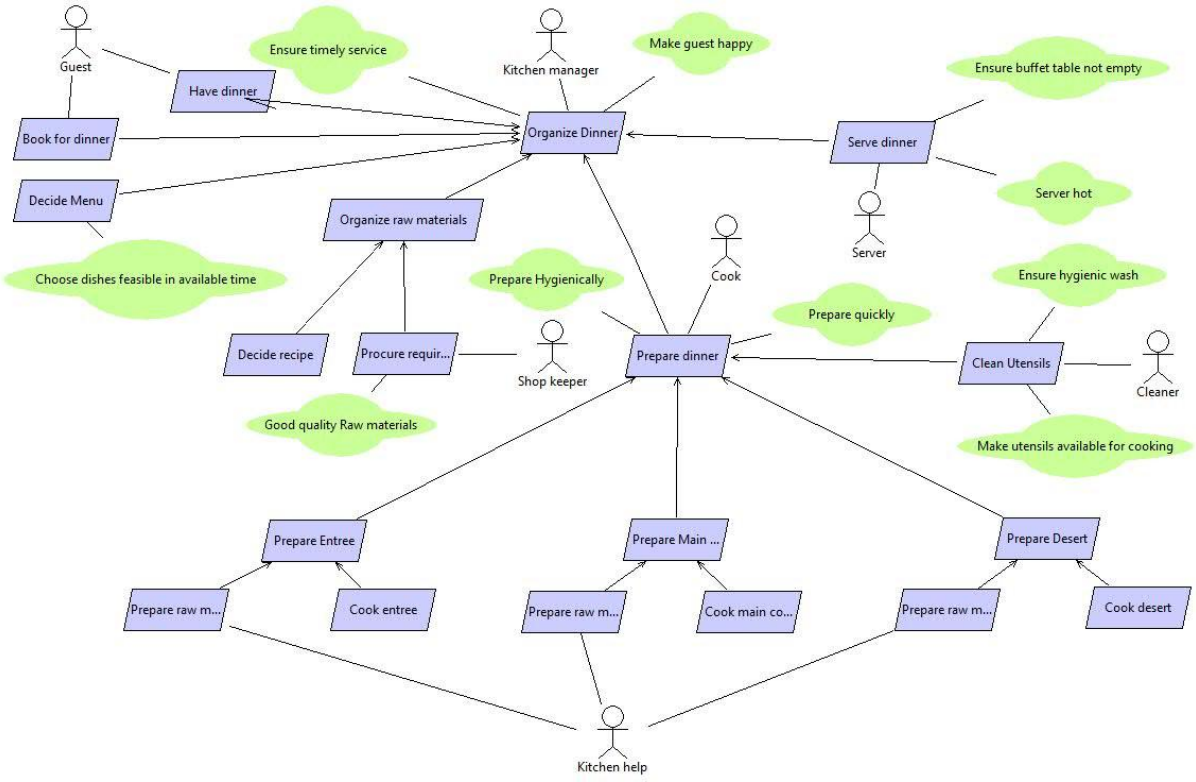

Figure 1: Goal Model

5. **Server (SR):** This agent serves the prepared dishes to the buffet table. It replenishes the buffet with the prepared dishes if any of the dishes on the buffet table is below a certain threshold value.
6. **Guest:** This agent is responsible for having dinner. It consumes the prepared dishes in a very random fashion. We have tried to model guest in a very realistic way; it makes intelligent choice of which dish to eat after looking at the menu. It is also possible that it consumes more of a certain dish than the other dishes, and does not eat a particular dish at all.

2.2 Resources

There are four different types of resources that are modelled as a part of this system.

1. **Raw Materials:** The term 'raw materials' has been used to indicate the ingredients in their basic form. For example- lettuce before it is washed and cut is a raw material in this project. These are the materials that the kitchen helpers work on. Cooks add raw materials and inform kitchen helps about the same. The kitchen helps prepare one raw material at a time and send it to prepared-raw-materials stack.
2. **Prepared Raw Materials:** These are the raw materials that are ready for cooking, for example-lettuce that is washed and cut is a prepared raw material in this project. The cooks remove the all the prepared raw materials for a particular dish and prepare the dish.
3. **Prepared Dishes:** This contains all the cooked dishes. Once the cooks complete a particular dish they send it to the prepared dishes for the server to serve to the buffet.
4. **Buffet:** This is the only resource that is accessible to the guests. They consume food from the buffet by making choices between various food types, and whether to eat more of a certain dish.

2.3 Environment

1. **Dining hall:** This is the environment which simulates the Dining Hall of the Cooking Domain. Agents who are present in the dining Hall Environment are- the kitchen manager, guests and server. The only resource available in this environment is the buffet table.
2. **Kitchen:** The agents present in this environment the cooks, kitchen helpers, cleaners, server. The resources present in this environment are raw materials, prepared raw materials, and prepared dishes, clean and dirty utensils.

2.4 Assumptions

While implementing this project, we made some assumptions and also specified some constraints. We have listed these below.

1. It is a Buffet dinner arrangement.
2. One plate of food is considered as one unit. While cooking, the cooks assume that the guests would have one unit per dish per person. The guests behave differently- eating more of some dishes and none of some other.
3. A cook can cook up to 5 units at a time.
4. There is no cleaning at the end of the dinner.
5. Cutlery, chairs, tables and money have not been implemented in this system.

3 JADE Platform [1]

JADE is multi-agent programming framework written in Java programming language. It allows for multi-agent communication by following Foundation for Intelligent Physical Agents (FIPA) specifications. JADE provides an API to cater to the various requirements of agent-oriented programming. It also provides a GUI to manipulate and monitor agents, including remote agents. Remote Method Invocation (RMI) is used to communicate between agents on different machines.

JADE supports three modes of communication between agents,

1) **Sending/Receiving Messages:** This is the most basic form of communication which sends/receives string type as the message content. This is useful for atomic data but not for abstract concepts and structured data. The 'receive' can be blocking or non-blocking.

2) **Sending/Receiving Objects**: This form of communication uses the Java Serializable interface to send/receive objects between agents. This can be used to represent the structure of data but is not human readable. Here again, the 'receive' can be blocking or non-blocking.
3) **Ontologies:** This form of communication allows definition of objects to be transferred as set of predefined classes adhering to FIPA format. This can be used to represent abstract concepts, predicates and structure of data.

In our project we are using the fist two modes of communication-sending/receiving messages and sending/receiving objects. We have used both blocking and non-blocking receives. We have also used a *broadcast* style of communication where there is one sender and many receivers.

4 Implementation

The following sections describe the implementation details of both the applications.

4.1 Command-line Simulator – using JADE/Java

We have simulated a buffet dinner. The dishes are broadly classified as Entrée, Main-Course and Dessert. There are different types of agents used in this project (explained in the next section). They are- Kitchen Manager, Cook, Kitchen Helper, Server, Cleaner and Guest. The cooking is flagged off by the kitchen manager and the cooks prepare the dishes according to the menu specified by the kitchen manager. The prepared dishes are served by the server to the buffet. The guests come in at intervals and consume dishes.

We have implemented the project using JAVA 5 and JADE 4.6 API. We have provided a GUI for the user to specify the number of guests, cooks, servers, cleaners and kitchen helpers at run-time.

4.2 Visual Simulator- using .NET

The application is broadly a combination of two parts:

1) Functional engines, including Rendering Engine, XML Translator, and Graph Engine. The Rendering Engine creates all those sprites graphic items, and paints them onto windows desktop application panels. XML Translator reads in the XML files and translates them into proper data structures that the agent simulator can understand. The Graph Engine contains the physical structure of the kitchen domain and provides path finding information to Agent Simulator.
2) Agent Simulator, including Action Engine and Role Manager. In order to simulate multiple-Agent environment, each role, to some extend, needs to have Artificial Intelligence feature. This was achieved by the Role Manager. Raw data from AI will be processed by the Action Engine and then translate into movements / messages / events, which can be understood by other functional engines.

The whole system was developed under the environment of .Net Framework 2.0, with the IDE Visual Studio 2005.

Here is a brief architecture view of our visual simulator (Figure 2).

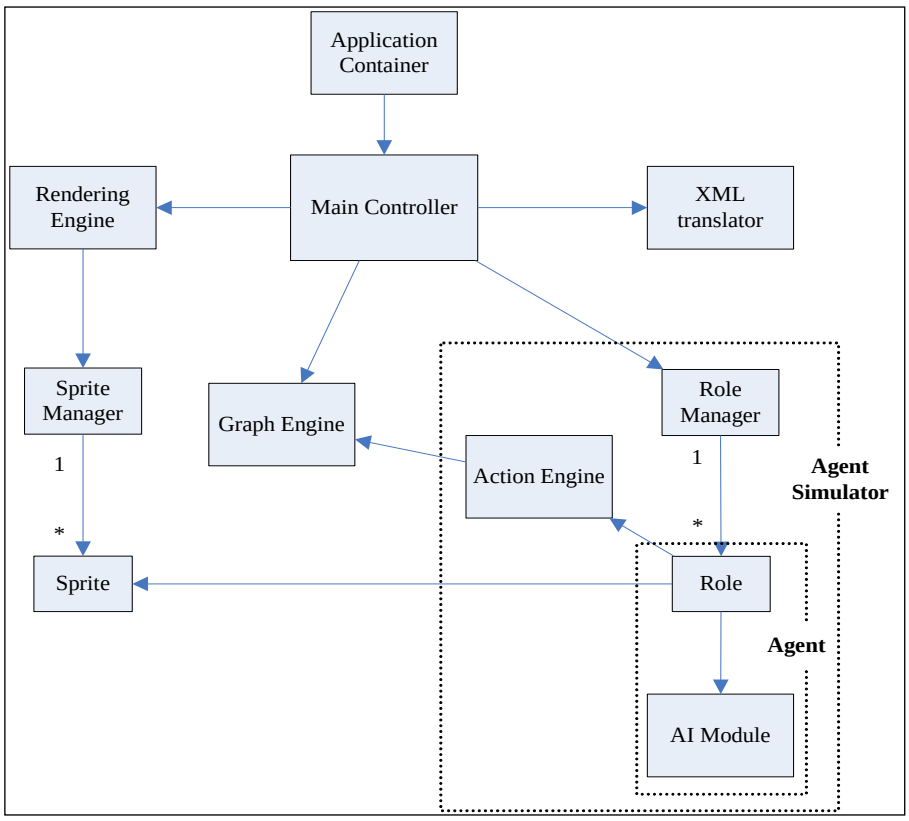

Figure 2: Architecture

4.3 Screen-shots

Screenshots of the command-line as well as visual simulator have been provided below (Figures 3-5).

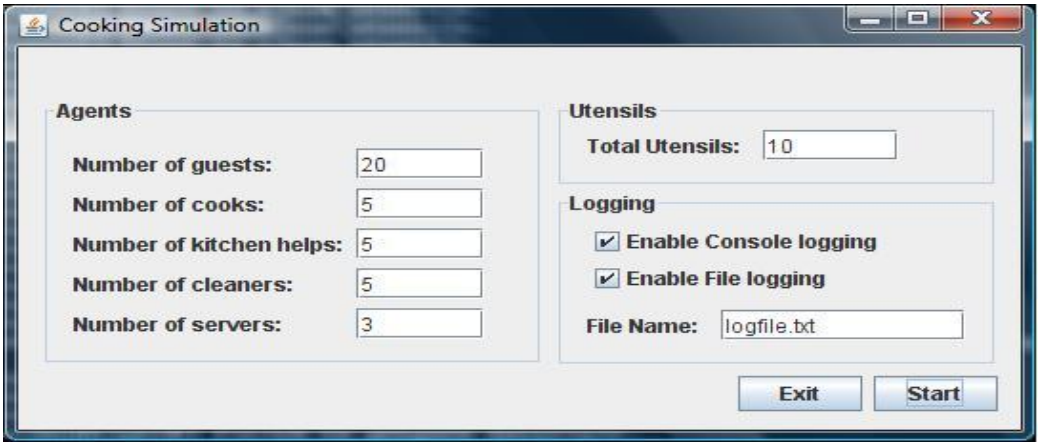

Figure 3: Interface to start command-line simulator

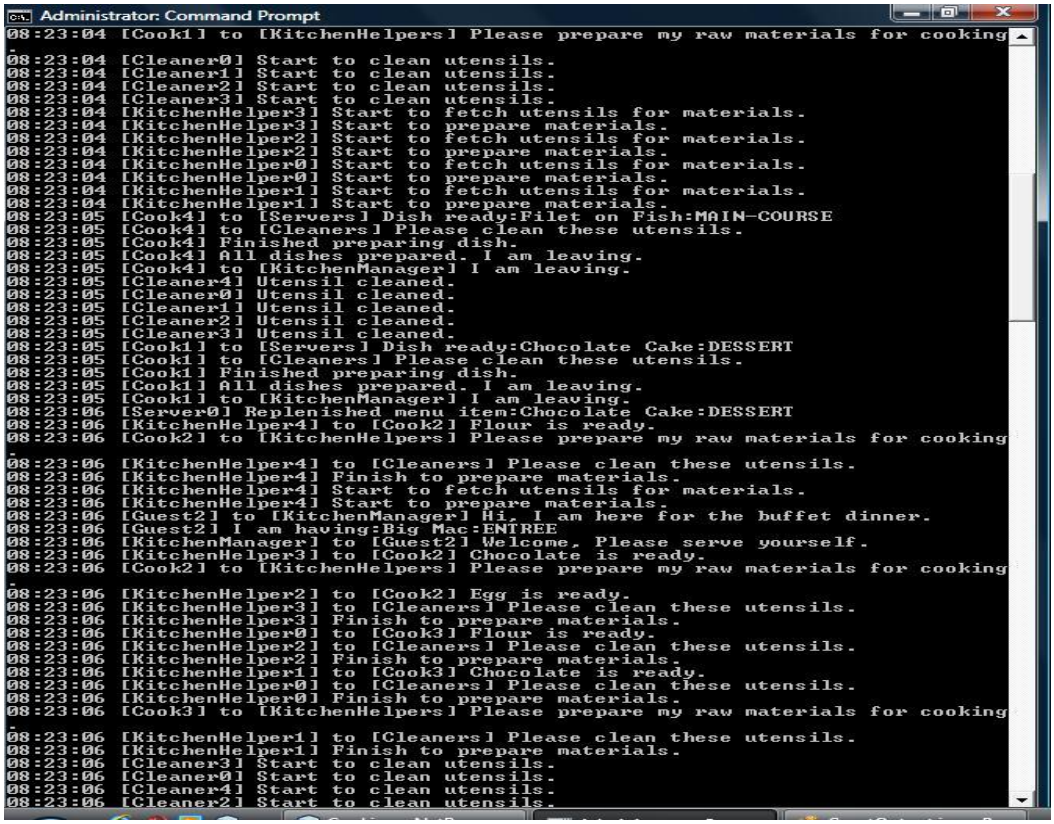

Figure 4: Output of the command-line simulator

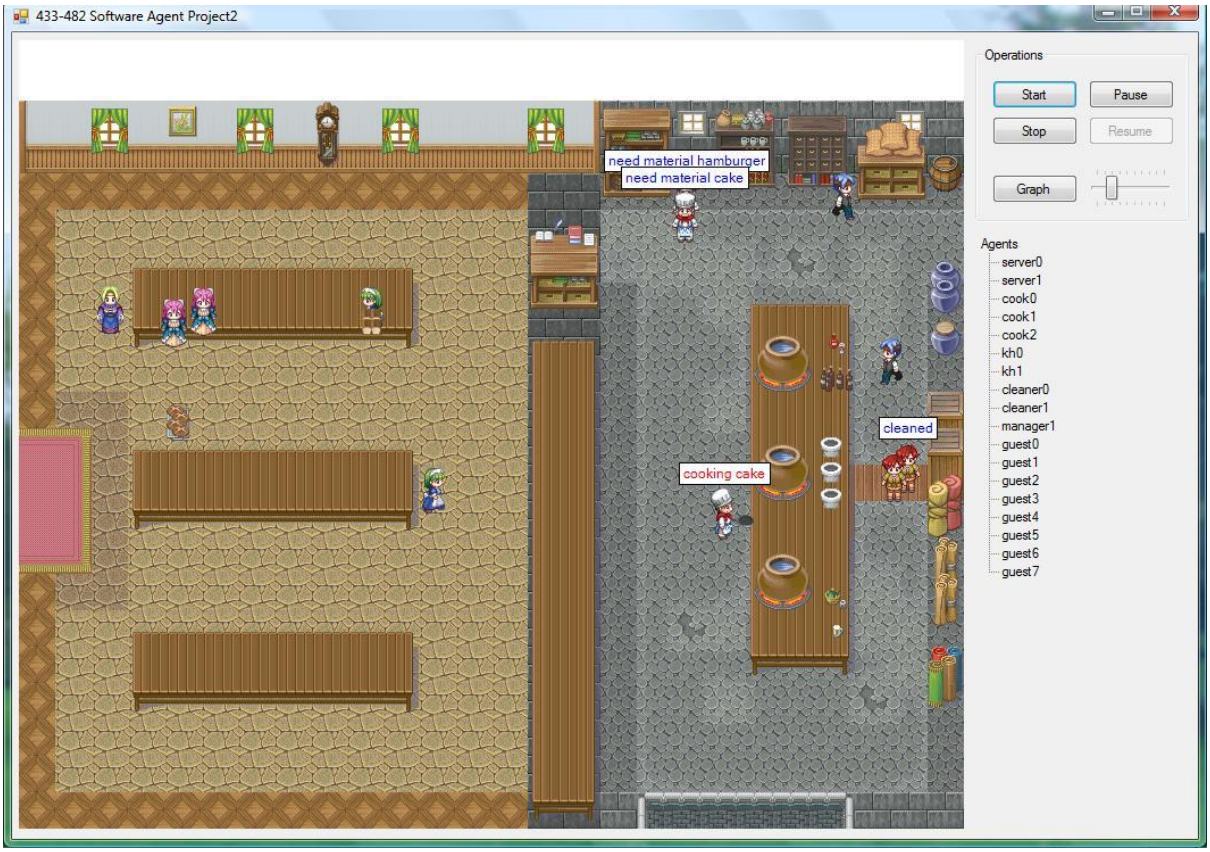

Figure 5: Visual Simulator

5 Challenges

While implementing this project we faced various challenges. Firstly, being staunch object-oriented believers we had to undergo a paradigm shift from object-oriented to agent-oriented design and development. This paradigm shift was initially difficult to cope with and led to some rework to be done. It took a while for us to start thinking of agents as intelligent entities, and information sharing between agents must happen using message passing and not method invocation as in object –oriented style of programming. But eventually it was a rewarding experience. Another problem we encountered was due to concurrent access of shared resources by different agents. Debugging multi-threaded applications is not straightforward at times. 'Synchronized' access to all resources was enforced which alleviated the problem.

6 Conclusion

A simplified Kitchen Domain was implemented in this project. We got a hands-on experience in implementing agent behaviour and learnt quite a lot about agent-oriented programming from this experience. A lot of improvement can be made to this system in terms of making more realistic and introducing more resources and also al a carte mode of dining. A more extensive GUI can also be implemented that depicts graphically the state and current activity of each agent.

References

[1] JADE Primer: http://www.iro.umontreal.ca/~vaucher/Agents/Jade/JadePrimer.html
[2] JADE Documentation: http://jade.tilab.com/papers-index.htm
[3] JADE Tutorials: http://www.ryerson.ca/~dgrimsha/courses/cps720/JADEAdmin/jadeStart.html

[4] JADE Resources: http://agents.cs.bath.ac.uk/cm30174/jade/index.html