Chapter 3

Theoretical Foundation of Autonomous Curriculum Design

3.1 Introduction

Previous chapter discussed about the state of the art in curriculum design and hence identified the issues in the area of curriculum design. Among those issues it postulate the problem been address in this project. This chapter will describe the theoretical foundation for autonomous curriculum design.

Nowadays rational for curriculum design, change over the time due to the rapid growth of world knowledge. At present, there are so many software solutions for curriculum design under different technology. But it is not argued that multi agent technology is ideal for automating curriculum design. In this case, proposed autonomous solution for curriculum design mainly focuses on the multi agent technology based on JADE agent development platform. And it is evident from literature ontology is suitable for providing better knowledge base for curriculum development area. Therefore curriculum ontology has developed based on Protégé.

3.2 Multi Agent Systems

In artificial intelligence research, agent-based systems technology has been hailed as a new paradigm for conceptualizing, designing, and implementing software systems[24]. Agents are sophisticated computer programs that act autonomously on behalf of their users, across open and distributed environments, to solve a growing number of complex problems. Increasingly, however, applications require multiple agents that can work together. A multi-agent system (MAS) is a loosely coupled network of software agents that interact to solve problems that are beyond the individual capacities or knowledge of each problem solver.

Advantages of a Multi-Agent Approach

MAS has the following advantages over a single agent or centralized approach:

- An MAS distributes computational resources and capabilities across a network of interconnected agents. Whereas a centralized system may be plagued by resource limitations, performance bottlenecks, or critical failures, an MAS is decentralized and thus does not suffer from the "single point of failure" problem associated with centralized systems.
- An MAS allows for the interconnection and interoperation of multiple existing legacy systems. By building an agent wrapper around such systems, they can be incorporated into an agent society.
- An MAS models problems in terms of autonomous interacting component-agents, which is proving to be a more natural way of representing task allocation, team planning, user preferences, open environments, and so on.
- An MAS efficiently retrieves, filters, and globally coordinates information from sources that are spatially distributed.
- An MAS provides solutions in situations where expertise is spatially and temporally distributed.
- An MAS enhances overall system performance, specifically along the dimensions of computational efficiency, reliability, extensibility, robustness, maintainability, responsiveness, flexibility, and reuse.

Agents can be model in different levels according to the needs of users. There are several types of agents[15]. Some of them are as follows.

Simple reflex Agents

In simple reflex systems, the choice of appropriate actions depends on the agent's immediate perception.

Model based Agents

Here the action is based on a long term knowledge model. So the knowledge about past becomes more important that immediate perceptions.

Goal based Agent

Information about the goal is use to filter out the best action in a particular

occasion. Here the agents are work for a common goal.

Utility based Agents.

Utility based agents quantify the value of states. This lets them to decide what to do when the next action is not certain.

Learning Agents

These agents gradually improve the performance by learning from the experience.

Also Agents communication is a form of interaction in which the dynamic relationship between agents is expressed through the intermediary of signals, which, once interpreted, will affect these agents [4]. The communication will active by sending some information from a sender to a set of receivers through the message space. This information is encoded with the help of languages and decoded upon arrival by the receivers. In this project we refer to the abstract communication model of FIFA that derives from speech act theory. In this model communication occurs through the exchange of asynchronous message corresponding to communicative acts. The ACL language format defines the format of these messages. ACL message can be describe by: Intention, Attendees, A Content, Content description, Conversation control. Based on these communicative acts, FIPA has defined a set of interaction protocols, each consisting of a sequence of communicative acts to coordinate multi-message actions, such as the contract net for establishing agreements and several types of auctions. We used FIPA ACL messages to communicate between agents in our system.

Intention	type of message such as INFORM, REQUEST, QUERY_REF

Attendees	The sender and the set of receivers
Content	The actual information that is exchanged
Content description	An indication of the content language used to express the content and the ontology
Conversation control	includes interaction protocol and conversation identification

Table 3.1: Characteristics of ACL message

Undoubtly, these features are used by the system to support designing in a productive manner. In broader sense agent systems are working as follows. Agent receive request from client then system assign recourses by creating many agents upon the request. Beforehand there are no agents except a reusable skeleton of a class. The communications among agents are implemented on a message space and required knowledge is stored in the ontology. Finally when all requests are met by assigning resources or resource cannot be assign the sessions terminate and all agents are destroy.

Nowadays, MAS developers have used ontology to implement effective communication among agents. Currently, number of tools available to facilitate the creation of agent based systems. Java Agent Development Environment is one of the major agent development environments available today.

3.3 JADE

The Java Agent Development Environment (JADE) is a FIPA compliant agent development framework to make easier the agent development application for intelligent multi-agent systems and interoperable both in wired and wireless environment [21]. The objective of JADE is to simplify development of the agent while ensuring standard compliance. FIPA-compliant Agent Platform includes Agent Management System, Directory Facilitator and Agent Communication Channel [2]. JADE is fully developed in Java language and is made by various Java packages and it is giving to application programmers both ready-made piece of functionality and

abstract interface for application dependent task.

3.3.1 Content Languages and Content Ontologies

Typically, FIPA-compliant agents will exchange messages with well-defined structured content, represented in some Content Language and described by an Ontology which will allow agents to exchange relatively high-level concepts without the risk of misunderstanding. Content languages are domain independent while content ontologies are domain specific. Therefore unlike ontologies that normally must be defined ad hoc for the domain addressed by an agent application, content language is typically selected among those already available without the need of defining a new one. JADE provide codec for the two content languages. Those are the FIPA SL language and FIPA LEAP language. FIPA SL is a human readable string encoded content language and LEAP is a non-human readable byte encoded content language. The SL language is particularly indicated in open applications where agents from different developer and different environments have to communicate. The property of been human readable can be very helpful when debugging and testing the application. Because of those features this project aims to use FIPA SL language as the content language.

The message content ontology facilitates agents to describe facts, beliefs, hypothesis and predication about a domain. Jade contains extensive support for defining and using content languages based on externally defined ontologies, and includes support for automatic translation between messages represented in a given content language and Java objects. The content ontology describes the structure and some semantics of the message content, such as properties of content elements, relationships between contents. A number of languages exist for representing ontologies for use in computer systems, not limited to DAML+OIL and OWL. At present JADE does not support these directly instead ontologies are encoded as java classes, either written by hand or generated automatically using tools like Protégé.

3.3.2 GUI enabled agent in JADE

Since the many systems are developed in user friendly manner, JADE also provides facilities to create Graphical User Interface. JADE includes the package jade.gui for this specific purpose. This package contains set of generic classes useful to create GUIs to display and Agent-Identifiers, Agent Descriptions, ACL Messages and etc [8]. JADE is providing the abstract class GuiAgent that extend the Agent class. This class has two specific methods: postGuiEvent() and onGuiEvent(). These two are the methods that allow handling the interaction between a GUI and an agent program. This feature of JADE has been used to interact with the users in the curriculum design system.

3.3.3 Agent Behaviors

Nature of agents is to operate independently and to execute in parallel with other agents. In order to support efficiently parallel activities within an agent, JADE has introduced a concept called Behaviour. A behavior is basically an event handler, a method which describes how agent reacts to an event. In JADE bahaviour are classes and the event handler code is placed in the method called action. Basically there are 2 kinds of behaviour classes: primitive behaviours and composite ones which can combine both simple and composite behaviours to execute in sequence or in parallel. There are three main types of primitive behaviours and two main types of composites behaviours.

Primitive behaviours

1. SimpleBehaviour: an under-rated basic class that you can extend in various ways and which often turns out to be the best solution when other promising Behaviours are found to have some hidden quirks.
2. CyclicBehaviour: This behaviour stays active as long as its agent is alive and will be called repeatedly after every event. Quite useful to handle message reception.
    - TickerBehaviour: a cyclic behaviour which periodically executes some user-defined piece of code.

3. OneShotBehaviour: This executes ONCE and dies Not really that useful since the one shot may be triggered at the wrong time.
    - WakerBehaviour: This executes some user code once at a specified time.
    - ReceiverBehaviour: which triggers when a given type of message is received (or a timeout expires).

Composite Behaviours

- ParallelBehaviour: controls a set of children behaviours that execute in parallel. The important thing is the termination condition: we can specify that the group terminates when ALL children are done, N children are done or ANY child is done.
- SequentialBehaviour: this behaviour executes its children behaviours one after the other and terminates when the last child has ended.

3.4 Ontology

For many countries, philosophers have recognized ontologies as the artifact for enabling communication among human. Depending on the ontology domain knowledge will differ. For this project used ontology to implement effective communication among agents. Ontology is a formal representation of a set of concept within a domain and the relationships between those concepts. It is used to reason about the properties of that domain and may be used to define the domain [20]. Ontology provides shared vocabulary, which can be used to model the particular domain. It includes machine interpretable definitions of basic concepts in the domain and relations among them. Sharing common understanding of the structure of information among people or software agents is one of the common goal in developing ontologies[12]. Therefore ontology is ideal for creating required knowledge for the agents in curriculum design system.

3.5 Protégé ontology

Protégé is an integrated software tool used by system developers and domain experts to develop knowledge based systems [22]. Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies. At its core, Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Protégé can be customized to provide domain-friendly support for creating knowledge models and entering data. Further, Protégé can be extended by way of a plug-in architecture and a Java-based Application Programming Interface (API) for building knowledge-based tools and applications. The Protégé platform supports two main ways of modeling ontologies: Protégé-frames Editor and Protégé-OWL Editor. The Protégé-Frames editor enables users to build and populate ontologies that are frame-based, in accordance with the Open Knowledge Base Connectivity protocol (OKBC).. The Protégé-OWL editor enables users to build ontologies for the Semantic Web, in particular in the W3C's Web Ontology Language (OWL).

The ontology consists of classes, slots, facets and axioms. Classes are concept in a domain of discourse. A taxonomic hierarchy can be constructed with the classes. Slots descried the properties or attributes of the classes. A slot in it self is a frame that has a type. The type can be primitive class, String, Integer, Float, or an instance of another classes. Furthermore a slot has a value. Facets describe the properties of slots. These properties include cardinality of a slot, allowed values, numeric boundaries and required or optional of a slot.

3.6 Summary

In this chapter we discussed about the theoretical foundation of autonomous curriculum design with Multi Agent technology and ontology. FIPA standards are well defined and mostly recognized in agent community. It provides Agent communication, Agent management, and Agent system architecture standards. As an agent development platform JADE has many features. JADE provides many tools to develop, control, manage, test and monitor Agents. Ontology been generator provide facility to convert protégé ontology in to JADE. Multi agents were based on JADE platform and the ontology based on Protégé. Next chapter describe how we proceeds towards the solution using the technologies discussed in this chapter.