Chapter 5

THE SIMULATION TOOLS AND RESEARCH METHODOLOGY

5.1 Introduction

Now a days, complex software applications are being developed at an exponential rate and further, the strong desire is to have a united system connecting multiple organizations operating autonomously and cooperatively as well. Such organizations and applications make use of distributed technologies and in fact, multiagent systems are playing vital role in making such a requirement a reality. Agent based systems are finding vital role in information retrieval systems, search engines, office automation, ecommerce, just to list a few. Agents in a multiagent system must be developed considering the requirements of applications. In fact, the study of literature revealed that that various agent developing platforms are available.

The existence of various platforms is primarily due to heterogeneous agents and as well as environments in which the agents operate. Therefore, the choice of most suitable platform for developing and simulating an application is still a challenging task. The chapter begins by exploring various agent development platforms and later compares them for finding the suitability of an agent based development framework for KQML based agents. Since the primary focus of this research work is to improve the KQML, hence an agent building platform that supports communication amongst KQML agents is highly desired.

In order to compare various platforms, lot of research efforts (251,252,253,254) have been put forth. To summarize, the articles present various criteria and metrics for comparing the simulation tools. Out of the various metrics listed in the literature, the
most dominating factors are platform properties, usability, operating ability, pragmatics and security management (see figure 5.1) (255).

<table>
<thead>
<tr>
<th>Metrics for Comparing Simulation Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>platform properties</td>
</tr>
<tr>
<td>usability</td>
</tr>
<tr>
<td>operating ability</td>
</tr>
<tr>
<td>pragmatics</td>
</tr>
<tr>
<td>security management</td>
</tr>
</tbody>
</table>

Figure 5.1 : Metrics for Comparing Simulation Tools

Here, platform properties imply the basic characteristics of a platform required for developing an application. Usability mentions the appropriateness of the platform for the building of agent-based uses. Operating ability covers the potential of platform for executing various applications. Pragmatics indicates the practical utility of the development platform and it is desirable that the agent platform would address the basic security related issues.

This primary agenda of this chapter is to explore various agent platforms. The literature suggested that few platforms are more popular as compared to others and this is primarily due to the type of applications being developed. A brief description of agent development platforms that can serve as simulation tools for the current study is being given in the upcoming section.
5.2 The Agent Development Frameworks

Figure 5.2 depicts various frameworks that are suitable for developing and designing agents and for running an agent based application as well. A brief description of each of the tool is given as follows:

A) Agent Factory

Agent Factory (256) is an open source agent building framework that offers a platform, and languages that support the development and deployment of multi-agent systems. There are two versions of Agent Factory namely, Agent Factory Standard Edition (AFSE), and Agent Factory Micro Edition (AFME) (see figure 5.3). In contrast to AFSE which is suitable for deploying agents on computers and laptops, AFME is a light weight framework that allows agent deployment on mobile and other constrained devices. AFME is compatible with J2ME-CDLC. In fact, the tool is finding its applicability in mobile computing, robotics and many other projects.
**B) AgentBuilder**

AgentBuilder\(^1\) is a KQML compatible simulation tool suitable for developing agents. It requires no background knowledge pertaining to the theory and development of agent based systems and in fact, with agentbuilder, developers can develop agents and related applications very easily. The framework is primarily based on the concept of mental states in agents and is the most suitable for KQML based agents. The detailed discussion about the same is presented later in this chapter.

---

Figure 5.2: Agent Development Frameworks
A) AgentScape

AgentScape (257) is an agent development platform suitable for building complex and large scale multiagent systems. AgentScape supports development and deployment of heterogeneous agents which can be deployed at various locations, operate in distributed manner and can also communicate with each other. The platform offers a scalable and autonomous environment and is addresses the issues related to distributed heterogeneous systems.

B) AGLOBE

AGLOBE (258) is a lightweight agent developing podium that in addition to basic features required for designing agents, also can handle issues pertaining to finding the location of agent and also issues related to establishing communication among agents. In fact, it offers the fast communication amongst agents. AGLOBE offers basic features required for agent existence such as communication related functionality, storage for agents, directory facility, and facility to migrate from one location to another and so on. Although, it complies with FIPA partially and lacks features such as inter-platform communication. AGLOBE is very much in demand for real-world simulations and mobility based devices.

C) AnyLogic

AnyLogic (259) is a GUI based and object-oriented model suitable for agent-based general purpose simulations. The tool can handle dynamics of a multiagent system and also allows discrete event modeling. AnyLogic is flexible and thus can handle the intricacies and diversities in multiagent systems operating to handle various real world systems. AnyLogic's is finding applications in
industrialized production systems, commerce processes, manpower management, end user and hospital management.

**D) Cormas**

Cormas\(^2\) is an imitation tool which finds its basis in the VisualWorks programming environment. The tool allows the design of agent based applications in the Smalltalk, an object oriented language. It contains Smalltalk generic classes which can further be used for specialization and refining. It further allows users to create agents as per user own requirements. It enables not only the building of multiagent application but also the design, observing and investigating of multiagent scenarios. Cormas was chiefly concerned with the depiction of relations between participants around the use of normal renewable resources. Cormas is finding its application in real-life application.

**E) Cougaar**

Cougaar\(^3\) stands for Cognitive Agent Architecture. It is funded by DARPA and is an open-source agent simulation tool. It offers distinct provision for problems pertaining to logistics in particular. Although the tool is not as per the regulations of FIPA but it enables the design of distributed and complex multiagent applications. Since Cougaar's supports cognitive behavior which is its unique attribute as compared to other available tools, it acts as an important tool when human behavior is required to be emulated by agents.


F) CybelePro

CybelePro\textsuperscript{4} delivers its operators with a strong structure for fast expansion and placement of extensive and active agents in a systems. The tools is known to be the marketable proclamation of Intelligent Automation Inc. The simulation tool has been used widely by the administration, business and university for solicitations such as armed logistics, demonstrating, modeling and governance of carriage and communication nets.

G) EMERALD

In the series of tools listed above, EMERALD\textsuperscript{5} is relatively a latest tool that facilitates logic among interoperable agents especially in the domain of Semantic Web. In this case it uses third-party services which are trustworthy and have the reasoning abilities. EMERALD allows exchange of location of agents amongst agents, irrespective of the fact that exchanging agents follows the same rules, are homogenous or heterogeneous. The tool has been designed in JADE and hence completely complies with FIPA. It ropes a diversity of reasons and high level languages such as Java, Prolog, XML etc. The primary and unique contribution of this tools is that it has mechanisms to evaluate trust and reputation and hence offers better decision making. EMERALD is found to be useful in trading applications.

H) GAMA

GAMA\textsuperscript{6} is an agent infrastructure which offers a ful solution for modeling agent-based systems. It has been found useful by the reserchers and scientists in the

\textsuperscript{4} CybelePro (2014). Available at \url{http://www.i-a-i.com/}
\textsuperscript{5} EMERALD (2014). Available at \url{http://lpis.csd.auth.gr/systems/emerald/}
\textsuperscript{6} GAMA (2014). Available at \url{https://code.google.com/p/gama-platform/}. 
domain of multiagent systems especially which are spatially separated. It offers abilities about the close-fitting grouping of 3D imagining, Geographical Information Systems Data Management (GISDM) and modeling of multilevel systems.

The tool provisions abilities for erecting large scale agent-based models written in the GAML language. GAML has a graphical modeling tool that facilitates quick development of agents. Users can activate agents from GISDM and can further run very large-scale simulation.

I) **INGENIAS**

The simulation framework for designing agents is based on INGENIAS\(^7\) approach which is a model driven approach based on INGENIAS Meta-Editor (INGENME). The meta editor in-turn produces self-sufficient graphic executive editor for languages defined using an XML file. In fact, INGENME is suitable for the systems where the schemes' stipulations are treated to yield software design, html forms, or any other essential products. The tool itself overcomes the prevailing manufacturing disputes.

J) **JACK**

JACK\(^8\) is an inter-platform tool for constructing, executing and incorporating business level agent oriented systems. JACK is based on BDI architecture of agents. Alike, BDI agents, it provides a suitable abstraction hiding the complexity of complex system JACK agents are designed in JAVA and therefore is platform independent. Such agents are portable and can execute on any system ranging from a mobile phone to multiple and distributed systems across the network. The tool is one of the prominent BDI-agent based simulation tool that

\(^7\) INGENIAS Development Kit (2014). Available at <http://ingenias.sourceforge.net>

\(^8\) JACK (2014). Available at <http://www.aosgrp.com>
employs an natural language lengthening the Java. Although it offers many advantages still the same is not a FIPA compliant tool.

\textbf{K) JADE}

JADE\textsuperscript{9} which stands for JAVA Agent Development Environment is a free and open source framework. It is written and implemented in Java. It acts as an interface and is a fully FIPA complaint tool. Being developed in JAVA, the tool is portable, platform independent and can be distributed across the network. It can be remotely controlled. JADE can run on devices such as mobile phones, laptops, palmtops as it is a lightweight application. It delivers a well-organized, ascendable, scattered environment in accordance to FIPA specifications. JADE was designed to ease the process of developing agent applications and it uses several mechanisms for accomplishing it. Each JADE agent runs as a single thread within an agent container, and collections of containers run within a single JVM platform as shown in figure 5.4.

\begin{figure}[h]
\centering
\includegraphics[scale=0.5]{jade_diagram.png}
\caption{Components of JADE Simulation Toolkit}
\end{figure}

\textsuperscript{9} JADE (2014). Available at <http://jade.tilab.com/>
L) Jadex

Similar to JACK, Jadex\(^{10}\) is also based on BDI architecture and simplifies development of agents which are intelligent too. The agents are thus designed in XML and JAVA. Jadex is finding importance in scientific applications, academia and industry. The applications thus designed encompass mobile computing, scheduling etc. The Jadex is based on active components based on service component architecture, more popularly known as SCA. It hierarchically arranges active agents interacting via services and thus controls the complication of the system.

M) JAMES II

Java Framework for Modeling & Simulation (JAMES)\(^{11}\) is a pure Java framework. It do not depend any external sources for modeling and simulation. In fact, JAMES II follows plug-in architecture offering maximum flexibility and a clear abstraction of algorithms, workflows and tools. The abstraction and plugin attributes makes it an efficient tool. A user can choose any plugin from the vast list of available plugins. It is flexible as it offers users to pick and choose the plugin as per their application. Moreover, it lets complete control over testing kinds and factors, such as scanning of parameters, optimization, calculation and number of repetitions. JAMES II is finding its utility in applications ranging from laptop based to big cluster sized applications.

\(^{10}\) JADEX (2014). Available at <http://www.activecomponents.org>

\(^{11}\) JAMES II (2014). Available at <http://wwmosi.informatik.uni-rostock.de/jamesii.org/>
N) JAS

JAS\textsuperscript{12} stands for Java Agent-Based Simulation Library. In contrast to JAMES II which is a pure agent development framework, JAS is not a pure agent development infrastructure. It is a simulation platform especially developed for modeling and simulation of agent-based systems. It is a originally developed at Santa Fe Institute. JAS’s simulation engine follows discrete event simulation paradigm. The discrete event paradigm allows management of time very precisely and at multi-scale level. JAS depends on open source third party libraries.

O) Jason

Jason (260) is actually an interpreter for extended version of AgentSpeak where AgentSpeak is a Belief Desire Intention based agent development language. Jason is based on Java. Further, Jason considers implementing the functioning semantics and offers a stage for the expansion of agent-based systems. Jason offers various features which can be easily customized by users. Few prominent features of the toolkit are :

- The agent communication is analogous to speech acts
- Plan labels can be annotated
- Selection function are completely customizable
- Trust functions
- Agent architecture is complete in itself
- Actions can be user defined as well as extensible
- A Java based multi-agent environment

P) JIAC

Java-based Intelligent Agent Component, more commonly known as JIAC\textsuperscript{13} is an agent development framework that simplifies the design of very large scale and distributed agent based systems. As the name suggests, it is a Java-based platform. The outline provisions the plan, application, and distribution of software entities in entirety. It also lets for the option of reprocessing uses and facilities, and even amending them dynamically.

Q) MaDKit

MaDKit\textsuperscript{14} stands for Multiagent Development Kit. It is an open source platform implemented in JAVA. It is further integrated and accessible multiagent platform and is based on Agent/Group/Role (AGR) organizational model. Agents designed using MaDKit can operate in groups. It offers various agent related services such as management of lifecycle of agent, message passing and distribution. MaDKit addresses heterogeneity in agent architectures, communication languages and also allows customizations.

R) NetLogo

It is programmable simulation framework for multiagent systems. NetLogo\textsuperscript{15} finds its motivation from Logo programming language. It follows the principle of "low threshold and no ceiling". NetLogo allows examination of developing occurrences. It is possessed with a wide-ranging library encompassing various domains, such as economics, biology, physics, chemistry, just to list a few.

\textsuperscript{13} JIAC (2014). Available at \url{http://www.jiac.de}
\textsuperscript{14} MaDKit (2014). Available at \url{http://www.madkit.net}
\textsuperscript{15} NetLogo (2014). Available at \url{http://ccl.northwestern.edu}
NetLogo finds its utility in academics as well as industry for modeling and simulation of existing models.

S) **MASON**

MASON\(^{16}\) is a Java based agent development tool kit. It is relatively fast, extendable, process-oriented simulation tool. It is finding utility in simulating applications based on swarm robotics, machine learning, socio-technical projects. MASON offers dynamic connection and disconnection between model and visual tools.

T) **Repast Suite**

Repast (261) is an open source platform for modeling and simulating agent based systems. It is an advanced category of tool that has taken more than 14 years to develop. It is a free-ware and offers cross-platform compilation facility. It has been implemented in many languages and has numerous integral adaptive types.

U) **SeSAm**

SeSAm\(^{17}\) stands for Shell for Simulated Agent Systems. It is a general platform meant for designing and modeling of multiagent systems. It’s a visual programming aid that makes it easy for developers to develop the agents. SeSAm comprises of agents, resources and the world. SeSAm is scalable and can be executed into different runs and results can be compiled later. It offers experimental and theoretical analysis pertaining to computing the complexity of


\(^{17}\) SeSAm (2014). Available at <http://www.simsesam.de>
a system. It was initially developed using Lisp and later transformed to a Java-based tool.

V) Swarm

Swarm\textsuperscript{18} is a reusable software tool that supports agent based modeling and simulation. The tool is intended for artificially intelligent systems, in particular. The complex multiagent systems with adaptive abilities can be simulated using this tool.

A comparison of all the simulation tools is delineated in table 5.1. As shown in figure 5.5, agent development tools are finding applications in the popular core areas. On studying and understanding the various agent simulation tools, it is discovered that since our research study is focused around KQML based agents and for simulating the same, AgentBuilder is found to be the most suitable tool. A brief description of the same is being presented in the next section.

\textsuperscript{18} Swarm (2014). Available at <http://www.swarm.org>
Figure 5.5: Applications of Agent Simulation Tools
5.3 AgentBuilder : The Simulation Tool

AgentBuilder is an integrated tool suite for constructing intelligent software agents. As shown in figure 5.6, AgentBuilder consists of two major components - the Toolkit and the Run-Time System.

![Figure 5.6 : Components of AgentBuilder Simulation Tool](image)

The AgentBuilder Toolkit includes tools for managing the agent-based software development process, analyzing the domain of agent operations, designing and developing networks of communicating agents, defining behaviors of individual agents, and debugging and testing agent software. The Run-Time System includes an agent engine that provides an environment for execution of agent software. Agents constructed using AgentBuilder communicate using the Knowledge Query and Manipulation Language (KQML) and support the performatives defined for KQML. In addition, AgentBuilder allows the developer to define new interagent communications commands that suit his particular needs.
All components of both the AgentBuilder Toolkit and the Run-Time System are implemented in Java. This means that agent development can be accomplished on any machine or operating system that supports Java and has a Java development environment. Likewise, the agents created with the AgentBuilder toolkit are Java programs so they can be executed on any Java virtual machine. Software developers can create powerful intelligent agents in Java that execute on a wide variety of computer platforms and operating systems. The toolkit and the run-time system are described in the following paragraphs. Figure 5.7 shows the architecture of AgentBuilder and their relation of various components to each other. The AgentBuilder toolkit is designed to provide the agent software developer with an integrated environment for quickly and easily constructing intelligent agents and agent-based software.

A) Project Control Tools

The Project Control Tools are provided to help the agent developer manage the overall agent development process. These tools include the Project Manager, the Project Dictionary Tool, and the Project Repository Manager.

B) Ontology Manager

The Ontology Manager assists the developer in analyzing the agent application problem domain and in identifying and defining the concepts relevant to the agent's operation in that domain.

C) Agency Manager

The Agency Manager is designed to help the developer construct an agency. An agency consists of two or more agents that communicate and cooperate with each other to perform some task. The agents may be identical or specialized for performing different functions. The Agency Manger allows the developer to identify and characterize all of the agents and agent types in the system under development. The Agency Manager provides a run-time window for viewing the operation of a system of agents. Thus, the developer can monitor interagent
communications, control the agents, or run the agent debugger to examine the state of any or all agents.

Figure 5.7: The Architecture of AgentBuilder
D) Agent Manager

The Agent Manager provides tools for defining an individual agent's initial mental model and behavior. The agent definition tools include graphical editors for defining the various mental constructs that make up the agent: initial beliefs, initial commitments, initial intentions, capabilities and behavioral rules. In addition, the Agent Manager supports tools for adding planning and learning capabilities to an agent.

E) Planning and Learning

The AgentBuilder toolkit provides assistance for adding a learning and planning capability to the agents. The planning and learning editors provide the developer with the ability to build domain specific planning and learning modules.

F) Agent Debugger

The Agent Debugger provides run-time tools for communicating with an executing agent. The debugger provides the software developer with graphical output describing the agent's mental model, input and output messages. These tools allow the developer to set breakpoints and step through the operation of an agent.

5.3.1 Salient Features of AgentBuilder

AgentBuilder allows software designers having little or zero knowledge regarding development of agents to rapidly and effortlessly construct agents and the allied applications. AgentBuilder reduces development time and development cost and
simplifies the development of high-performance, robust agent-based systems. Salient features of AgentBuilder are listed as follows.

- Requires no special expertise in intelligent agent technology or network communications
- Constructs agents with built-in capabilities for autonomous operation, monitoring their environments, reasoning, and communicating with other agents and users.
- Provides a suite of graphical programming tools for specifying agent behavior and operation
- Utilizes a high-level, agent-oriented programming language. Programming is accomplished by specifying intuitive concepts such as beliefs, commitments and actions
- Provides tools for analyzing the problem domain.
- Provides tools for defining agencies -collections of intelligent agents and their interactions
- Provides tools for testing and debugging agents and agencies
- Java-based, cross-platform toolkit for creating cross-platform, agent-based applications. Creates agents that are Java programs.
- Supports easy integration and use of existing software libraries (Java, C and C++)
- Built-in KQML Agent Communication Language (ACL).
- Supports CORBA and IIOP protocols.
- Supports TCP/IP sockets

As shown in figure 5.8, AgentBuilder is available in two different versions to meet a wide variety of developer needs: Lite is an entry-level product for agent software developers where as Pro, is intended for serious multi-agent development.
5.3.2 Salient Features of AgentBuilder Lite

It provides tools for constructing single-agent stand-alone applications and small agencies of agents. Following are some of the salient features of AgentBuilder Lite:

 ✓ Project control tools including a Project Manager for process management.
 ✓ Ontology tools including concept mapping and object modeling tools.
 ✓ Agent manager tools for creating agent programs using an Agent Definition Language (ADL)
 ✓ Agent debugger with capability to debug a single agent.
 ✓ Run-time engine.

5.3.3 Salient Features of AgentBuilder Pro

AgentBuilder Pro is the ideal tool for developers building multi-agent applications. AgentBuilder Pro includes all of the tools in AgentBuilder Lite plus:

 ✓ Agency manager provides tools for creating and managing multiple software agents.
 ✓ Agency viewer tools allow real-time examination of remote agent operation.
 ✓ Role editor defines roles for agents.
Protocol editors specify inter-agent conversations.

Agency debugging support.

Support for integrating a variety of learning and planning modules.

5.3.4 System Requirements

AgentBuilder is coded in Java and produces Java-based agents. AgentBuilder is distributed with the latest JRE (Java Runtime Environment) for each supported platform. Both the AgentBuilder toolkit and the run-time system execute on the Java Virtual Machine (VM) included with the JRE. AgentBuilder distributions are available for the popular platforms namely, Microsoft Windows NT/2000/XP, LINUX, Sun Solaris. Moreover, AgentBuilder will run on any platform with a Java Virtual Machine. Minimum AgentBuilder development environment requirements are a 266 MHz Pentium-class machine with 128 MB of RAM. The runtime engines are lightweight processes requiring less than 300K bytes of RAM.

5.3.5 Sample ScreenShots
Now turning our attention to the research methodology adopted to achieve the stated objectives, the next section presents a brief description of research methodology.
5.4 Research Methodology

Research methodology is an approach to scientifically solve the research problems. It may be considered as a science of learning how research is done technically. In it we study a variety of steps that are normally adopted to crack the problem along with the logic behind them. To cater the defined problem, we explored the literature thoroughly and studied the articles published in national and international journals and conferences ranging from IEEE, Springer, Scopus, Science Direct and ACM digital library, white papers and other online sources.

The basic approach of this research study is to develop an improved design of KQML including the design of novel communication protocol along with updates in performatives and list of parameters to strengthen the basic structure of KQML. Figure 5.9 depicts the flow diagram of the activities being carried out during the course of this work.

5.5 Conclusion

This chapter enlisted and detailed the comparison between various simulation tools suitable for the development of agent based systems. The comparison of various tools unveiled the fact AgentBuilder simulation tool is the most suitable tool for KQML based agent communication and systems. Since, the primary focus of this work improving the basic KQML, therefore, AgentBuilder was further explored and studied. The detailed study of the same reflected that AgentBuilder is simple and good choice even for the naïve agent developers. A small section focusing on the research methodology adopted to achieve the stated objectives was also presented.
Analysis of Literature survey

Establishing Aims and Objectives

Conceptual Framework Development

Development of Communication Protocol

Updates in KQML

Interpretations and Findings

Evaluation of Improved KQML

Publication of Research Papers

Thesis Writing

Figure 5.9: The Adopted Research Methodology