3rd International Conference on Science,Technology and Management
India International Center, New Delhi (ICSTM-16)
17 January 2016, www.conferenceworld.in 978-81-932074-0-6

AGENT ORIENTED SOFTWARE FRAMEWORK: A REVIEW

Dinesh Kumar Singh[1] , Dr. Devesh Katiyar[2]

[1] *Assistant Professor , Dept of Information Technology, DSMNRU Lucknow (India)*

[2]*Assistant Professor, Dept of Computer Science , DSMNRU Lucknow (India)*

ABSTRACT

Agent technology encompasses the theory of agent acts, functionalities, implementation and the communication languages. Various methodologies have been developed to model the agent acts such as: speech act theory, Belief Desire and Intention (BDI) theory and Agent-0. Different Agent Communication Language (ACL) has been designed for the agent acts in general and communication per formatives in particular.

Various tools and platforms have been developed by different agencies to incorporate the agent functionality in Java based or other imperative and declarative languages. This paper attempts to bring together in a cohesive manner the agent oriented programming framework that includes tools and platform, languages for agent act and communication. A comparative view is presented for various tools and platform with their URL, functionalities and uses. Apart from Java based the other .Net platform is also discussed.

The comparative view of many other agent oriented software products is presented.

The comparative view of makes a quick and comprehensive view of the various platforms, software product comprised of various tools, platform and languages.

Keywords: Agent, Agent Communication Language (ACL), Java, Platform, .NET

I. INTRODUCTION

Agent, Agent based system, Agent Oriented technology, Agent mediated system and Multi agent system are nowadays every where pervading (omni present) terminologies (jargon) in the research, businesses and industries communities. As these exhibit the characteristics of human behavior on many fronts, such as computation, coordination, mobility communication and cognition; it has transformed the computation paradigm from solitary to social settings. The conventional design methodology of software: function based, object oriented, design patterns and component based incorporate the agent oriented concepts, model per formatives, primitive constructions and different level of abstraction i.e. control, communication and languages.

The software product as a program entity displays personified act of human for interaction and interface with external world as well as changes in its cognitive faculty to adapt itself to the working environment.

Agent based architecture may be thought of as programs that perform tasks on behalf of some party. Hence agent that performs work for users, usually storing and refining a set of tasks according to a set of preferences; using artificial intelligence to solve tasks. The agent must have degree of independence that allows them to initiate tasks according to their own way in distributed environment and help the other agents. Hence an agent

based framework is that which can provides an environment for working an agent effectively [9]. One aim of the agent frame work is therefore to isolate the agent dependent behavior logic underlying support code which may common to all of the agents in the simulation. The agent frame work can maintain a separation between mutually dependent functional areas to extent that one area could be replaced with out modifying the other. The agent based framework provide solution to develop software tools to use cooperating intelligent entities to represent domain specific knowledge and make decisions through a negotiation process. Such an implementation is possible by using concept of multivalent system. Agent system environment offers a reliable communication infrastructure enabling interoperability among agents which can rely on the well known agent communication languages [10, 37].

The agent programming framework covers and expresses many underlying and inter level concepts such as : agent platforms , agent toolkit , agent development tool agent development environment , agent infrastructure above all multi agent paradigm characteristics: mainly coordination, cooperation and contracting [7,16].

Two main platforms apart from mobile agent [29] we describe here .NET [1] and JAVA .NET commonly known as component based programming platform is a support infrastructure enabling the development of distributed application.

It is easy to program efficiently and reliably the agent framework with the user application domains coupled with the language C# for the enterprise agent based application [1].

The salient features of some of the product related to O/S and platform are described. Most of the agent based programming are related with Java framework and some are related with .NET framework.Actually a huge number of development tools exits for the design and realization of agent system.Each developing tool depends upon agent programming framework .

In this paper a review and assessment have been performed to enumerate the different software products in MAS paradigm working on different platform /environment, interfaces and with its own functionality, framework and architecture. We emphasize the product software environment languages, platform, functions, features and applications for study. Different tools offer different ability depending on agent definition and type.

The paper is divided into different sections. Apart from introduction in section 1 the next sections as follows. Section 2 deals with the description of some of the important and practicing platforms with their functions, features, implementation and interface. Section 3 covers the details of agent oriented programming languages for agent acts and communication. A comparative view of different tools and platform with their URL, functions and utilities is presented in section4. Conclusion is mentioned in section 5.

II. PLATFORMS

2.1 NET Based Framework

The .NET framework is related with the Common Language Infrastructure(CLI) .Each CLI is rich collection of libraries .The .NET framework is a platform that simplifies application development in the highly distributed environment of the internet. Common Language Runtime (CLR) is the virtual machine for .NET version [1]. Since the CLR's works are directly associted with tools of MSDN compilers, debuggers and profilers so developer's job much simpler. On the basis of CTS and metadata CLR support the multi languages . The

language interoperability components developed by the rules provided by the Common Language Specification(CLS) .

In .NET the message transport is done using meta data. Metadata allows .NET language to describe themselves automatically in a language neutral manner , unseen by both the developer and user. It is useful for mobile agent when they are moving between agents platforms. The .NET framework platform use C# language which is an object oriented and type safe programming language. The .NET framework offers three predefine types of formatters: XML, binary and ActiveX[32].

For development of agent based software in which tools provide by Java , same type of tools are provided by .NET framework. But more application programs are developed in Java ,only NOMAD is a framework which used .NET based framework.The Java based framework is used due to its distributed application support , message transport support ,serilization .Similar types of facialities also developed by using .NET framework.

2.2 Mobile Agent Platforms

Odyssey, Voyager, Concordia is Java –based system that promotes agent migration with state presentation and that provide an agent "context" or "text" or "server to host mobile agents" but uncommon to these three are approaches to handle messaging, persistence, security and management. A brief description of their salient features is given below [29].

2.2 Java based Framework

The Java based framework is software development framework aimed at developing agent systems and applications conforming to FIPA standards for intelligent agent. It support the Agent Communication Language(ACL) and ACL message structure.

Java based framework is written in Java language due to its attractive features like object serialization ,reflection API and remotemethod invocation(RMI).

Java based frame work use Interface Definition Language (IDL) module define by FIPA and reflection API for message transport. JVM is used as virtual machine . Each JVM provides a complate runtime environment for agent execcution . Mostly Java based framework features described so for allow complate instructions between FIPA system agents and user define agents ie they are FIPA standards. A comparative view features of .NET Framework and Java Based Frameworkis shown in Table1.

	.NET Framework	**Java Based Framework**
Language support	C#	Java
Virtual Machine	CLR	JVM
Distributed application support	Remoting framework built on the CLR. Using system runtime remoting.	RMI
Message transport	Metadata	IDL module defines by FIPA. Reflection API.

Table **1** Comparative view between .NET Framework and **Java** Based Framework

2.3 Jade

Architecture/Function: JADE fully complies with the reference architecture of a FIPA agent platform. JADE architecture has an Agent Management System (AMS) which has supervisory control over agent platform execution. AMS has a directory of agent identifiers and the state of agent AMS also provide life cycle of an agent. The Agent Communication Channel (ACC) is the software component which controls the exchange of messages with in the platform including remote platform i.e. part of this architecture. JADE is an enabling technology, a middle ware for the development and run time execution of peer to peer applications which are based on the agent's paradigm. JADE works as an operating system for the agent based system. JADE is composed with the main functionalist (packages).To implement the systems kernel the package is JADE core. It has sub package which contain behavior class hierarchy for implement the task on intentions of agent. JADE contains different packages for communication, massage transport, display and edit etc.

2.4 Jena

Jena is a Java based application developed by HP lab in semantic web language.

The central part of Jena is RDF API.

Architecture/Function: RDF API is the core part of Jena. The information model which is defined by RDF is represented by directed graph that provides greater flexibility for reading, writing, merging the graph, manipulating the graph together by using set of triples. Each triple identifies node which is placed at the sharp end of the arc itself. The node may be labeled or unlabeled. The Jena has features to integrate the graph using operation of set theory union intersection and difference. Each RDF graph use as resource, some of which show the behavior, which are attached by developers. ARP[3], a parser that reads the RDF graph which is represented by standard language, RDF/XML .Jena use the RDF/XML writer with compact output and limited scalability.Jena store the data in main memory, relational database and a new type of stores generated by system programs .It support the DAML+ OIL ontology model.

Jena is buildup for inference and query optimization.

2.5 Jadex

Architecture/Function/Features: Jadex is a platform implemented by JADE agent platform. Jadex implements the BDI architecture using mentalistic concept for system design .Hence Jadex integrates the mentalist concept in the system with JADE platform.[14]. In JADEX the belief of an agent depends up on the informational attitudes which are domain dependent abstract entity, where as the central part of the BDI is goal which depends on the attitude of the agents by which agent can reach the goal. One of the Jadex attitude is plan which is deliberative attitudes means how agent achieve their goals.The Jadex agent has a belief base in which facts are stored .A translation agent is present in Jadex which maintains the belief base. Jadex has three types of goals: achieve, maintain and perform.

The translation agent registers its services using the achieve goal. Jadex has a plan library to represent the plan of an agent ; plans can be reused in different agents and can in cooperate functionality implemented in the other

Java classes .Hence the objective of JADEX is to implement the BDI architecture into the JADE agent platform. Jadex exhibits an explicit representation of agent's goals. Jadex agent follows the FIPA specified life cycle.

III. AGENT COMMUNICATION LANGUAGES

The development of powerful and general purpose programming technology enabled the design of agent based programming language to implement the specification characteristic and functionalities of MAS.The beliefs of agents can be represented as arrays and objects or by tables or databases. The SQL like queries are used to examine the beliefs of an agent .In the dynamic model of belief and goals of agent changing with time due to their observation and interaction are managed by different techniques to update, revise, and merge. In 3APL and Jason simple mechanism and deletion of atomic formulae have been used to update an agent's beliefs and goals. To reason about agents action and plans are implemented with AOP (Agent Oriented Programming) such as: FLUX [23] through specific programming constructs. AOP should address three issues; at implementation level related to individual agents beliefs, goals, plans and a decision making process that deliberates on these entities to determine to action to be performed: at environment Level issues are the effects of the actions that are performed by their agents and the access to recourse and services. At the multi agent and social level, the issues are the coordination of action, communication, mobility and security. These issues are implemented as coordination artifacts which facilitate interaction services and access to resources and services (access control).Implementation with Java, Prolog, and C++ do not provide dedicated programming constructs and functionalities like AOP that facilitate the direct and effective implementation of the three issues.

3.1 Agent Specification

Agents are viewed as intentional system where behavior can be predicted and explained in term of BDI: belief, desire and intention modalities.

AOP model which depicts some what different modalities than BDI theory. It advocates mental state or mentalist notation of belief, commitment (choice) capability.

The control mechanism which implements some model of rational agency; which was developed by Cohen-Levesque theory of intention [30].

3.2 Speech Act Model

3.2.1 Speech Acts

A formal way to describe communication is by interacting each communication primitive as an action that updates the knowledge of an agent about all the aspects of the state. The communication primitives that are exchanged among agents are typically referred to as communicative acts or speech acts.

The theory of speech acts is generally recognized as having begun in the work of the philosopher John Austin .Austin noted that a certain class of natural language utterances (here after referred to as *speech acts*) had the characteristics of actions, in the sense that they change the state of the world in a way analogous to physical actions. Cohen and Perrault gave an account of the [31]semantics of speech acts by using techniques developed in AI planning research .The aim of their work was to develop a theory of speech acts: modeling them in a

planning system as operators defined in terms of speakers and hearers beliefs and goals. Thus speech acts are treated in the same way as physical actions.

In the following paragraphs functions and features of agent speech act based model such as: AgentSpeak(L) Agent-0 and 3APL are given below.

3.2.2 AgentSpeak(L)

It is a programming language based on a restricted first order language with events and actions. The specification language consists of a set of base beliefs and a set of plans which looks similar to the definite clauses of logic programming. At run time an agent can be viewed as consisting of set of beliefs, set of plans, a set ofintentions, a set of actions, and a set of selection functions [2].The syntax consists of variables, constants, function symbols, predicate symbols, connectives, quantifiers and punctuation symbols.

Semantics of speech act Cohen – Levesque [30] used their dynamic logic formalism to develop a theory of intention to formalize several speech acts .The elements of communicative plans are called speech acts .The speech acts can be axiomatic it with the use of model operators that describe the states of belief ,knowledge ,wanting etc. For example, we can define speech act

A request B of R as follows

REQUEST (A, B, R): A request B to perform R

Precondition:

- Know-what (A, Location(B))
- Can perform(B,R)
- Willing to perform (B,R)

Post condition:

- Will (Perform(B,R))

A must know where B is and B can perform the request and B is willing to perform request .Then as a result B will perform the request.

In the model of Cohen – Levesque, the above speech act i.e. request can be implemented with the help the some constructs such as: alternating belief, mutual belief, attempt and helpfulness. An example of the semantics of Cohen – Levesque speech act is given below

Persistent goal:

(P-goal xp)=(goal x (later p))^bel ¬p)^

[Before ((Bel x p)v Bel x □ ¬p))¬goal x (Later p))]

It states that an agent has a persistent goal of p if:

It has goal that eventually becomes and believes that p is not currently true.

One of the following conditions must hold before it drops the goal: the agent believes the goal has been fulfilled; the agent believes the goal will never be fulfilled intention:

(Intend xα = (p-goal x) [Done x (Bel x (happens α))?;α]

It states that an agent has an intention to do α: it has a persistent goal to have believed it was about to do α, and then done α.

Agent-0

Belief, desire, choice, intentions and motivation these are the latent attribute of an agent. Different combination of these factors has been proposed by the researchers such as BDI; Belief, Desire and Intention, BD: Belief and Desire and BCC: Belief Commitment and Capability [12]. In the agent orient programming (AOP) belief is common to the other modalities but commitment is related to the obligation, choice and decision. The capabilities are not a behavior term but a physical measure but in the control it is in the conjunction with the behavioral quantities to form a modality. Capability of course, has its behavioral ingredients such as desire, commitment and motivation [38].

Time based temporal topic is used to model time based activities. For example holding (robot, cup) ^{t} means that robot is holding the cup at time t.

Belief: a model operator, B, denotes Belief. The general form of belief statement is B^{t}_{a} φ. Which means when agent is a, t is time and φ is recursively-defined sentence. An agent believes things both at certain times and about certain times. Another term is defined as B^{3}_{a} B $^{10}_{b}$ like (a,b)^{7} which means that at time 3 agent a belief that at time b will believe at time 7 a liked b.

Commitment: (Obligation): A new model CMT is defined which has no more argument than B.

$C^{t}_{a,b}$φ: At time agent a is obliged, or committed to agent b about φ this inherent flavor of the commitment contracts with part accounts of the same concepts, which viewed t as an intra agent phenomenon. The 'Commitment' here does not cannon ate all the multifaceted concept of motivation.

Decision (Choice): the current definition of the obligation provides an alternative; however decision or choice is defined to be simply obligation, or commitment, to oneself

$$DEC^{t}_{a}\varphi = {}_{def}CMT^{t}_{2,2}\varphi$$

The term 'choice' carries many connotation but here choice is in the sense od 'decision' an agent has chosen something if he has decided that some thing be true.

Capabilities: At time t, 2 is capable of φ is denoted by the notion CAN^{t}_{2} φ. CAN or capabilities is mental notion or net, it is a debatable fact. But it is assumed that 2 take a function U to act only when he believes that he would be able to the function and he intends to do. If one has intention to do, one has desire to do and also one believes that he can do, then one is bale to do. And thus capabilities require certain behavioral components t be active.

Capabilities refers to specific terms such as

$$CAN^{5}_{robot} \text{ open (door)}^{8}$$

Thus at the time 5, the robot might be able to ensure that the door is open at time 8, but at time 6 it might no longer have the capability

An immediate version of CAN is BLE. For any sentenceφ, time (φ) be the outermost time occurring it; for example time (open (door) ^{t}) = time (B^{t}_{2}φ)=t;

Now ABLE is defined as ABLE $_{2}$ φ = $CAN^{time\varphi}_{2}$φ and time ABLE $_{robot}$ open (door) ^{5} = CAN^{5}_{robot} (door) ^{5}

3APL

It is an Abstract Agent Programming Language that implements agent's mental attributes like beliefs, goals and plans and actions. The agent's reasoning rules modify their mental attributes. The salient features of 3APL as follows

Salient Features

The 3APL includes all the classical elements of theory agents i.e. beliefs, goals and plans. The interpreter for 3APL is already available with its extension, the evaluation of effectiveness of the language for the complex problems.

Comparison with Agent-0 and AgentSpeak(L) , it is based on Dribble [17] , a propagation language without variables and it constitute a synthesis between declarative and procedural approaches . So 3APL provides formal constructs to implement an agent's beliefs, goals, plans and intentions.

Syntax: The beliefs, goals and plans can be represented by a first order domain language in which the terms represent the objects and its formulae, represents the relation between domain languages.

Semantics: The semantics of 3APL is defined in the terms of transition system which is a set of derivational rule for deriving transition. Transition is the transformation of one configuration to another configuration in a single compilation step.

The semantics of G ∅ is defined in terms of semantic rules, variables and their values are framed and applied as top syntactic elements. The rules are framed for the execution of plan base, basic action, test, composite plans. Another set of rules is application such as: goal rule application, plan rule application and interactions rule application.

Comparative view of agent oriented programming languages such as : Agent-0- 3APL in Table2, Agent-0 - AgentSpeak(L) and AgentSpeak(L) -3APL are given below

Agent-0	**3APL**
Commitments	Goal
Commitment rules	Practical reasoning rules
Agent –0 lacks constructs like those of imperative programming	Construct for imperative programming.
Explicit representation of time and global clock for control flow i.e. time stamps.	Regular programming control flow.
Non formal semantics.	Formal operational semantics.
Simple mechanism of revision of agent's commitment built in the control structure which can be programmed in mental language.	No such revision of commitments (goal) but only addition of new one. Rules of 3APL can be used to modify or revise.
Each cycle executes all its commitments goals.	Focus on the specific goal.
Primitives for communication.	Extended of multi-agent language with communication primitives.

Table 2: Agent-0 - 3APL comparison

Agent-0 and 3APL: Both are rule based languages. Agent-0 only provides plan rules thus the range of rules in Agent-0 is lesser than 3APL.

Agent-0 and AgentSpeak(L) : Agent -0 can not revise or modify goals . AgentSpeak(L) has formal operational semantics. Using bisimulation AgentSpeak(L) can be simulated in 3APL.

AgentSpeak(L)-3APL

Agent Speak(L) is similar to that of 3APL i.e. each cycle one plan rule and a corresponding a goal .The filter phase in the control structure of 3APL lacks in AgentSpeak(L)[36].

In many respects AgentSpeak(L) and 3APL are similar **.** Extension of agent language with communication and other multi agent features are desired**.**

A very short description of other ACL are given below

Concurrent MetateM: It is based on temporal logics which allow specifying the intended behaviors of an agent [18].

ConGolog: This logic based language is initially designed for high-level robot programming [11].

ARCOL: A Rtimis Communication Language is used for ARTIMIS system, which has a set of primitives that can be composed [26].

KIF: Knowledge Interchange Format is used for any type of knowledge and

meta-knowledge .It is a logic based language for content communication where as KQML FIFA-ACL are for intention communication [19].

COOL: It focuses on rule based conversation management. COOL as a protocol sensitive layer above intention communication. It is applying for coordination knowledge and explicitly representing knowledge for multi agent system.[24]

Agent Talk: Agent Talk is a coordination protocol description language for multi agent systems. In the distributed artificial intelligence area, many coordination protocols such as the contract net protocol have been proposed, and many application-specific protocols will be required as more software agents start to be built up. Agent Talk allows coordination protocols to be defined incrementally and to be easily customized to suit application domains by incorporating an inheritance mechanism [4].

SDML: SDML is an acronym for Strictly Declarative Modeling Language. It is a modeling language with the many features like; knowledge is represented on rule bases, databases all knowledge is declarative, models can be constructed from many interacting agent [34].

LuCe: It is based on first order logic and adopts tuple centers as coordination media. It is a coordination language [7].

STL++: This language is used for describing a framework for organizational structure of a multi agent system [27].

KAOS: Knowledge Agent oriented system is dependent on object oriented programming language and utilizes CORBA based message delivery mechanism. It deals with persistent integration of agents with proper communicative primitives content message and applicable conversation policies. It is a project to build infrastructure for agent development [12].

3.3 ACL- supporting systems and applications

3.3.1 Agent Communication Language:

The agent communication language (ACL) provides a means for exchanging information and knowledge. It handles proportions rules and action instead of simple objects with no semantics associated with them. Its

message describes a desired state in a declarative language, rather than a procedure or method. It contains the key components as the per formative, service content and control levels.

3.3.2 Comparison of ACL with KQML

Both are similar on basic concepts but differ in semantic framework.

- Mappings or transformation between this two or either equivalent is impossible.
- These two have identical syntax but with different names for communication primitives.
- FIPA changed the language's original Prolog like syntax to match KQML's to facilitate the transition of KQML systems to FIPA ACL.
- Semantic descriptions of KQML are preconditions, post conditions and computation conditions.
- For FIPA ACL, the semantic descriptions are feasibility condition and rational effects.
- There are differences in the treatment of the registration and facilitation of the primitives i.e. programtive issues such as: updating registration information, and finding other agents to assist the processing of the requests.
- KQML introduces a special class of agents called communication facilitator which maintains the registers of service names, forwarding messages to named services , routing messages based on context, matchmaking between information providers and clients , and provides mediator and translation services.

FIFA ACL's SL is a quantified multimodal logic, based on other and Levesque [30] BDI theory having the model operators for Belief (B), Desires (D), Uncertain Beliefs (U), and intentions i.e. Persistent Goals (PG). For a given communication act (CA), a set of SL formulae is specified that describe act's feasibility and its rational effect. FP(a) is a feasibility precondition for describing the necessary conditions for the sender of the CA(a). The rational effect represents the effect an agent can expert to occur as a result of performing the action.

IV. TOOLS AND PLATFORM

Short description of some of the tools in practice and comparative view of Java based platform tools with their functions URL and utilities are given below.

4.1 Short Description

ZEUS: The ZEUS toolkit, which provides a library of software components and tools that facilitate the rapid design, development and deployment of agent systems [39].

RETSINA: The Intelligent Software Agents Lab at Carnegie Mellon University's Robotics Institute envisions a world in which autonomous, intelligent software programs, known as software agents, undertake many of the operations performed by human users of the World Wide Web, as well as a multitude of other tasks. The Software Agents Lab has developed the RETSINA multi-agent system infrastructure and has applied that infrastructure and its agents to many domains [33].

JATLite: JATLite (Java Agent Template Lite) is a package of programs written in the Java language that allow users to quickly create new software "agents" that communicate robustly over the Internet. JATLite also provides a basic infrastructure in which agents register with an Agent Message Router, connect/disconnect from

the Internet, send and receive messages, transfer files with FTP, and generally exchange information with other agents on the various computers where they are running [13].

LEAP: Leap (Lightweight Extensible Agent Platform) has been developed as a library, which cannot be used without being combined with JADE. Starting from version 3.0 of JADE, the LEAP libraries are now distributed as an add-on of JADE [15].

MADKIT: MadKit is a Java multi-agent platform built upon an organizational model. It provides general agent facilities lifecycle management, message passing, distribution, and allows high heterogeneity in agent architectures and communication languages, and various customizations [16].

4.2 Graphical Comparisons of Agent Software

Figure 1 shows a comparative view year wise development of Java based and Non Java based development of agent based toolkit; figure 2 shows year wise development of Languages; figure 3 shows year wise development of platforms of Java based and Non Java based. Figure 4 shows the comparison of simulation packages and figure 5 shows comparison of agent acts, communication languages and applications. All values are in percentage.

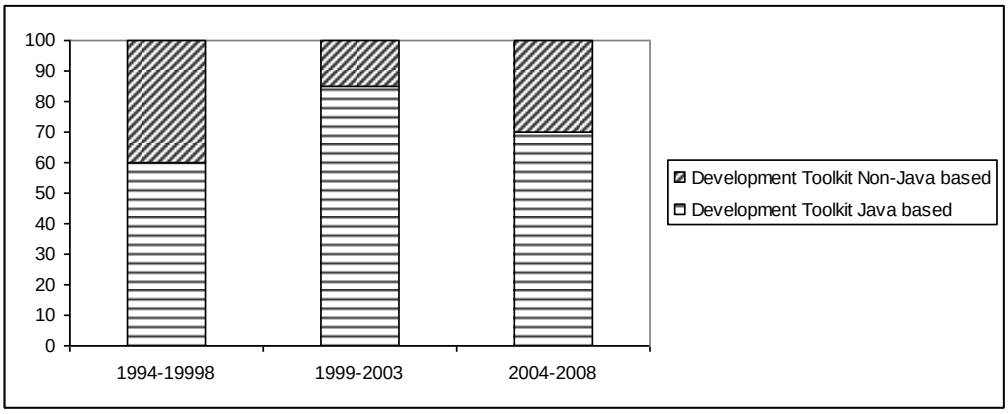

Figure 1: Development Toolkit

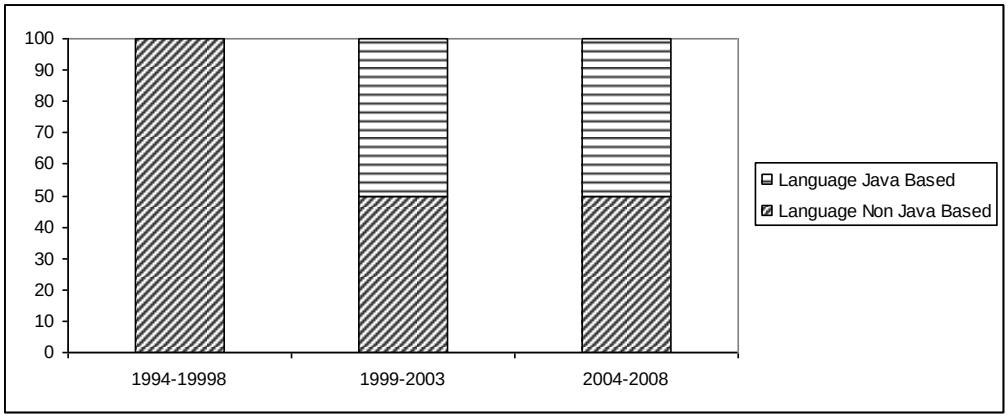

Figure 2: Languages

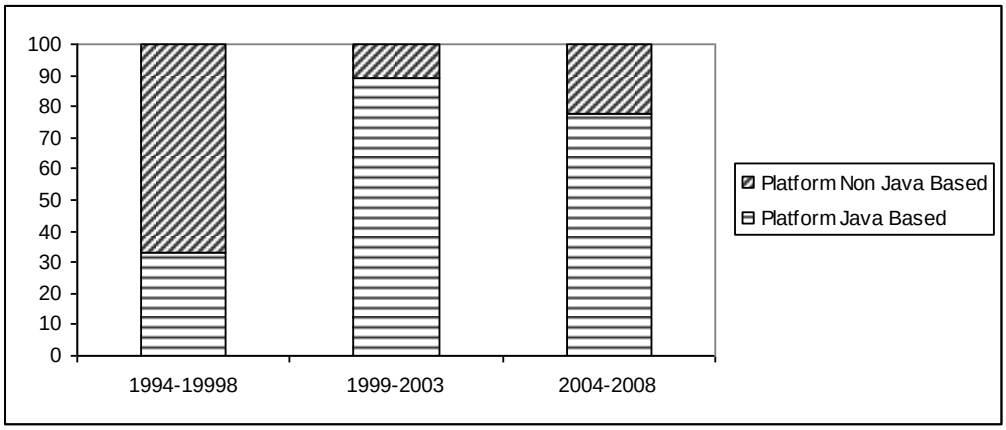

Figure 3: Platforms

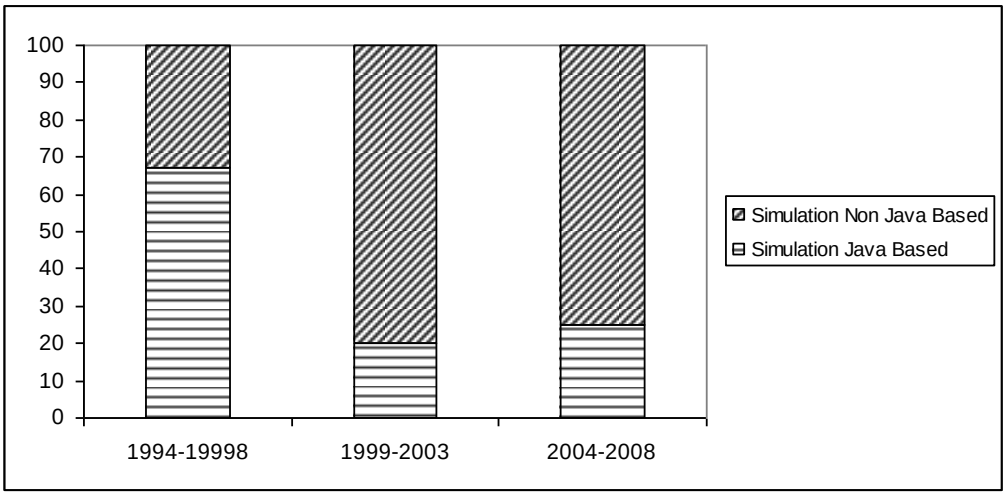

Figure 4: Simulation Packages

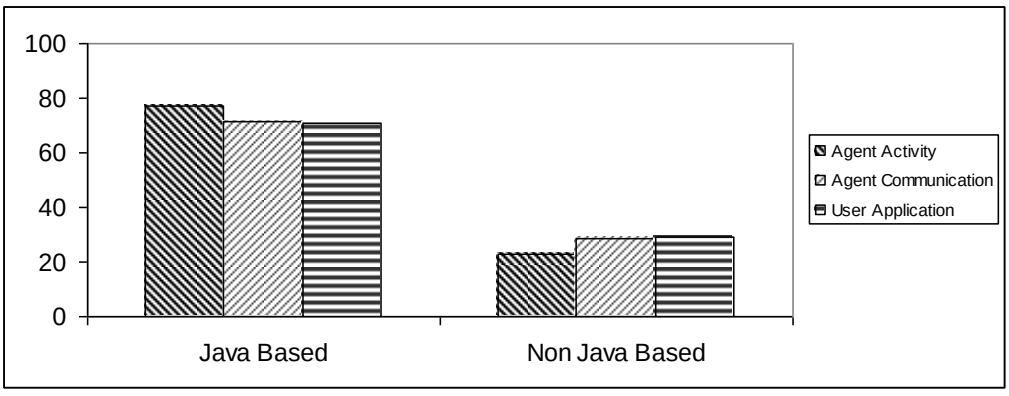

Figure 5: Development of different Agent Software

4.3 Observation and Discussion

We have made a comparative view of the Agent Development Tool kits, Agent Languages, Agent Platforms, Agent Simulation and Agent Software on the basis of yearly development as shown in figures 1-5 respectively. It is found from figure 1 that differences in the development of Java based and Non- Java based tool kits decreases in time up to 2008. Initially Non-Java based languages were much ahead of Java based language but

around 2008 Java based languages are ahead and almost equal up to Non- Java based as shown in figure 2.More alternation on platform dependency was paid 1998 initially Non Java based platform were comparable but around 2008 it is negligible in comparison Java based platform as shown in figure 3. From figure 4 it is clear that presently Java based simulation packages decrease in time up to 2008 in comparison to Non -Java based .Figure 5 is drawn on the basis of software for Agent activity (Agent acts), Agent Communication and agent based applications .In all the three aspects of software presently Java based software are much more ahead of three aspects in Non –Java based software.

V. CONCLUSION

The paper reviews the various aspects of agent oriented framework of software product. Initially the description of important Agent oriented Software product based on Java framework such as:

JADE, JENA, JADEX make their comparative view of any kind of user novice, expert or between the levels. The detailed description and comparative features of the ACL is given more understanding for novice reader. Although these details are available in the concerned literature but joint view makes more appealing and comprehensive. Many platforms, languages mentioned just in short have fewer roles in practice of these products in market. The comparative view with URL, functionality and uses may enable a user to select a particular tool for his applications in mind and practice. Similarly the tools based on Non-Java framework have been described in same tabular form.

There are various websites which speak of the tools, platforms and language. But they deal only with URL and few words about them. In this paper a comprehensive view of the platform, tools and agent programming languages have been described for the users.

The further attempt would be in this direction to determine the common features among tools, platform and languages respectively and to rate their in terms of their uses. It needs the feedback data from the users regarding their effectiveness and fulfillment of the job requirement or the purpose they claim to serve.

REFERENCES

[1] A.Grosso, A.Gozzi, M.Coccoli A.Boccalatte,(2003) “An Agent Programming Framework Based on the C# Language and the CLI”, Union Agency – Science Press ISBN 80-903100-3-6, 2003

[2] AS Rao, “AgentSpeak(L): BDI agents speak out in a logical computable language” Agents Breaking Away:Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World(MAAMAW-96) 42–55,1996.

[3] B.McBride, Jena: A Semantic Web Toolkit ,Hewlett-Packard Laboratories, Bristol, UK IEEE Internet Computing pp 55-59, 2002.

[4] D Kuwabara, T Ishida, and N Osato, “AgentTalk: coordination protocol description for multiagent systems” Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95) 455 1995.

[5] D. Wong, N. Paciorek, T. Walsh, J. DiCelie, M. Young, and B. Peet,"Concordia: An Infrastructure for Collaborating Mobile Agents,"Proc. First Int'l Workshop Mobile Agents, pp. 86-97, Berlin, K.Rothermel and R. Popescu-Zeletin, eds., Springer-Verlag, Berlin,Apr. 1997.

[6] Friedman-Hill E. Jess in Action: Java Rule-Based Systems , Manning Publications Company, 2003.

[7] E Denti, and A Omicini, "Engineering multi-agent systems in LuCe" Proceedings of the ICLP'99 International Workshop on Multi-Agent Systems in Logic Programming (MAS'99),1999.

[8] F Bellifemine, G. Caire, A. Poggi, , G Rimassa,"JADE Awhitepaper", .http://exp.telecomitalialab.com ,2003

[9] F. Bellifemine, G. Caire, T. Trucco., , JADE Programmer's Guide. ,TILab S.p.A, 2003

[10] FIPA, (1997)''Specification Part 2:Agent Communication Language,'' The text refers to the specification dated 23 October 1997.

[11] G De Giacomo, Y Lespérance, and V Levesque, "ConGolog: a concurrent programming language based on the situation calculus" Artificial Intelligence 121 109–169, 2000.

[12] Gerhard Weib "Agent orientation in software engineering" The Knowledge Engineering Review, Vol. 16:4, 349–373.,Cambridge University PressDOI:10.1017/S026988890100025X ,2001.

[13] JATLite, http://Java.stanford.edu/Java_agent/html/ 2000.

[14] L. Braubach, W. Lamersdorf, A.Pokahr, Jadex : Implementing a BDI –Infrastructure for JADE Agents, http://exp.telecomitalialab.com 2002.

[15] LEAP, http://leap.crm-paris.com/. 2000

[16] MADKIT, Multi-Agent Development KIT, http://www.madkit.org/. MADKIT, 1999

[17] M Dastani, Birna van Riemsdijk , Frank Dignum J J Meyer "A Pogramming Language for Cognitive Agents Goal Directed 3APL", ACM 200X

[18] M Mulder, J Treur, and M Fisher, "Agent modelling in Metatem and DESIRE" Intelligent Agents IVProceedings of the Fourth International Workshop on Agent Theories, Architectures, and Languages (ATAL-97) 193–208, 1998.

[19] MR Genesereth, and RE Fikes, "Knowledge Interchange Format. Version 3.0, Reference Manual"Technical Report Logic-92–1, Computer Science Department, Stanford University 1992.

[20] M..Wooldridge, "Semantics issues in the verification of agent communication languages", Autonomous Agents and Multi-Agents systems, V 3, pp 9-31,2000.

[21] M.Wooldridge and N. R. Jennings, ''Intelligent agents: Theory and practice,'' The Knowledge Engineering Review, vol. 10-2. pp. 115-152, 1995.

[22] MJ Wooldridge, and P Ciancarini, "Agent-oriented software engineering: the state of the art" Agent-Oriented Software Engineering: Proceedings of the First International Workshop (AOSE-2000) 1–28 2001.

[23] M Threlscher, " FLUX : a logic programming method for reasoning agents ." Journal of theory and practice of logic programming 5(4),533-565,2005.

[24] M Barbuceanu, and MS Fox, "Capturing and modeling coordination knowledge for multiagent systems"International Journal of Cooperative Information Systems 5(2–3) 275–314, 1996

[25] M Dastani, J Hulstijn,F Dignum, and Meyer, Issues in Multiagent System development, In proceedings of the third international joint conference on autonomous agents and multiagent systems(AAMAS'04). 2004

[26] MD Sadek, "Dialogue acts are rational plans" Proceedings of the ESCA/ETRW Workshop on the Structure of multimodal Dialogue 1–29, 1991.

[27] M Schumacher, F Chantemargue, and B Hirsbrunner, "The STL + + coordination language: a base for implementing distributed multi-agent applications" Proceedings of the Third International Conference on Coordination Languages and Models (COORDINATION'99) 399–414, 1999.

[28] Odyssey URL: http://www.genmagic.com/technology/odyssey.html.

[29] P. Dasgupta, N. Narasimhan, L.E. Moser, and P.M. Melliar-Smith,"A Supplier-Driven Electronic Marketplace Using MobileAgents," Proc. First Int'l Conf. Telecommunications and ElectronicCommerce, Nashville, Tenn., pp. 42-50, Nov. 1998.

[30] P.R. Cohen and H.J. Levesque, ''Intention is choice with commitment,'' Artificial Intelligence, vol.42, pp. 213-261, 1990.

[31] P.R. Cohen and C.R. Perrault, ''Elements of a plan based theory of speech acts,'' Cognitive Science, vol. 3, pp. 177-212, 1979.

[32] Rama Jiten , Judith Bishop ; "Towards a mobile agent framework for Nomad using .NET", proceedings of, pages111-113 , 2004.

[33] RETSINA, 2000, http://www-2.cs.cmu.edu/ ~ softagents/.

[34] S Moss, H Gaylard, S Wallis, and B Edmonds, "SDML: a multi-agent language for organizational modelling" CPM Report 97–19, Centre for Policy Modelling, Manchester Metropolitan University, 1996.

[35] VoyagerURL:http://www.objectspace.com/voyager/whitepapers/VoyagerTechOview.pdf.

[36] V. Koen Hindriks , S Frank der Boeer , Wiebe vander Hoek , and John jules Ch. Meyer , "Agent Programming in 3APL." Formal Semantics of the core of AGENT-0 ECAI' 98 Workshop on Practical Reasoning and Rationality , pages 20-29 ,1998.

[37] Y Labrou , J Mayfield and T Finin "KQML as an agent communication language" "Software Agents" ,MIT Press ,Cambridge ,1995.

[38] Y Shoham, "Agent-oriented programming" Artificial Intelligence 60(1) 51–92 , 1993.

[39] ZEUS, 1999, http://www.labs.bt.com/projects/agents/zeus/index.htm.