cisdi Journal

Computing, Information Systems & Development Informatics Journal
Vol. 11 No. 2, 2020 - www.isteams.net/cisdijournal

A Review of Wireless Agent Systems and Architecture

Ezea, I. L.
Department of Math/Computer Science/Statistics/Informatics
Federal University
Ndufu Alike, Ikwo, Ebonyi State, Nigeria
E-mail: ezeaikenna@yahoo.com

ABSTRACT

The society is becoming smarter owing to the presence of intelligent agents that are being deployed in smart devices to perform various activities. The popularity of the agent is due to its ability to perform intelligent tasks. The choice of the agent's architecture determines how intelligent the agent will function in any given environment. Thus, the aim of this research is to review all the agent's architectures to aid understanding of agents and the architectures. This research work reviewed the agents based on its environment, the architecture, the development approach, strength and the limitations.

Keywords: Wireless, Agent, Systems, Architecture, Environment, Intelligence

CISDI Journal Reference Format
Ezea, I.L. (2020): A Review of Wireless Agent Systems and Architecture. Computing, Information Systems, Development Informatics & Allied Research Journal. Vol 11 No 2, Pp 133-148 Available online at www.isteams.net.cisdijournal

1. INTRODUCTION

Since the introduction of intelligent agent by McCarthy between 1956 and 1958 the field has attracted several researchers. Ever since then many authors have given the definition of intelligent agent but until date, none has been credited to have given a universally acceptable definition of the term intelligent agent. Though no definition has been universally accepted, many still believe that autonomy is an attributes that is central to all agents. This is shown in the work of Maes (1995) where she defines autonomous agent as "the computational systems that inhabit some complex, dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed" (Maes, 1995). The environment presents a platform from which the agent senses and acts. Without the environment, the agent will not have a place to execute its actions, and from the definition of Maes (1995) it is obvious that the environment creates a podium for the agent to sense and act.

Furthermore, to support the importance of autonomy in agent computing Mark and Gordana (2009) mentioned autonomy as one of the important concepts in the development of a variety of fields in computing (Burgin & Dodig-Crnkovic, 2009). The definition given by Brustoloni (1991)also centered on autonomy (Brustoloni, 1991). In his definition, he defined autonomous agents as "systems capable of autonomous, purposeful action in the real world." In the definition given in an online white paper by Virdhagriswaran of Crystaliz, Inc. the term agent was defined to represent two orthogonal concepts "The first is the agent's ability for autonomous execution. The second is the agent's ability to perform domain oriented reasoning" in this definition also autonomy is still the focal point. Aaron and Lakshmi say that one of the purposes of an agent is to satisfy its user by autonomously and continuously performing his (the user's) task.

Autonomy is central here because the agent has to satisfy the users request without the user being present (Hector & Narasimhan, 2005). In the definition given by Wooldridg, he states that an agent is "a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives." (Wooldridg, 2002) Like we mentioned earlier the agent needs an environment to receive input and perform some actions. Just like Wooldridge, Russell and Norvig also gave a definition of an agent to include how the agent perceives and acts on the environment, they defined an agent to be "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators" (Stuart J. Russell, Peter Norvig, 2010), this definition has been adopted by many authors though (Stan Franklin, Art Graesser, 1996) criticized it saying that since this definition depends on environment, sensing and acting it will make every program to be an agent. Since the environment acts as a means through which the program receives input and sends output. The means of receiving input can be called sensing while sending output can be called acting. As we have seen many authors have incorporated environment to the definition of agent, now let's look at how the agent relate to the environment.

2. TRENDS IN AGENT'S ARCHITECTURAL DEVELOPMENT

The agent architecture is the foundation through which all agents are built. It helps in the modeling of the agent's behavior. Every reasoning of the agent is dependent on the architecture as this forms the building block of the agent's intelligence. The agent architecture uses the properties of the agent to enable it make a decisions. According to Wooldridg (2002), it is a software architecture with the intension of supporting the agent's decision making through the properties; reactive, proactive and autonomy (Wooldridg, 2002). Maes (1995) said that it shows the interaction between the sets of components which decomposition is supported by the techniques and algorithms included in the agent architecture (Maes, 1991). The architecture makes the agent to conceptualize the environment before it acts based on the information it has on its data structure or it may respond immediately to what is happening in the environment based on its impulse. The action the agent takes depends on the agent's architecture.

Over the years research has been going on in the area of agent architecture. The first agent architecture was symbolic or logic based architecture, this architecture has been around since 1956 though it has some draw backs which led to the discovery of another type of architecture called reactive architecture, this architecture has been around since 1985. Another architecture that came around 1990 was hybrid architecture, it combines the advantages of logic based architecture and reactive architecture. Another architecture developed by Bratman in 1987 is the BDI architecture. Broadly speaking the four architectures belong to the classical architecture. Other categories are the cognitive and semantic agent architecture. The BDI architecture is a deliberative agent architecture that has its basics on the three state mental characteristics; the belief, desire and intention.

2.1 Logic Based Architecture

Logic Based Architecture also known as Symbolic or Deliberative Architecture is built on the foundation laid by Newell and Simon on their Physical Symbol System Hypothesis (PSSH) (Newell & Simon, 1976). According to Wooldridge and Jennings a Physical Symbol System Hypothesis is defined "to be one that contains an explicitly represented, symbolic model of the world, and in which decisions (for example about what actions to perform) are made via logical (or at least pseudo-logical) reasoning, based on pattern matching and symbolic manipulation." (Michael & Nicholas, 1995). This classical architecture requires an agent to have an internal symbolic representation of the environment and the desired behavior such that the agent's decision is based on the manipulation of those symbols. This architecture did not thrive due to some criticisms by several authors. Their arguments according to Nilsson are based on four main themes. (Nilsson, 2007). The first argument is that the computer does not have an understanding of the symbol it manipulates, they claim that for the symbol to have meaning to the computer there should be a connection between the symbol and the environment through perception and action.

This action (grounding the symbol with the world) is to be achieved through embodiment which requires that the intelligence should reside within the body in order to aid sensing and action. The second theme was based on the fact that much of the intelligent actions especially perception exhibited by man does not necessarily need symbolic processing. The third theme claims that the model of intelligence provided through computation is not the appropriate model for human intelligence. The forth theme claims that some intelligent behavior (the type exhibited by plants and insects as they get along with the complex environment) does not reside within the mind. They adapt and respond intelligently well with the challenging situation of the environment even though they don't manipulate any symbol.

2.2 The Reactive Architecture

Judging by the unsolved problems of Symbolic AI, researchers started questioning the viability of the paradigm and this led to the development of an alternative approach called the reactive architecture. Reactive architecture is based on exhibiting intelligence without explicit representation and manipulation of symbols from the environment like is the case in symbolic AI. It is the direct mapping of perception to action.

Rodney Brooks also criticized and advocated for a change from Physical Symbol System Hypothesis to another approach, which he called "Nouvelle AI … and it is based on the physical grounding hypothesis. This hypothesis states that to build a system that is intelligent it is necessary to have its representations grounded in the physical world" (Nilsson, 2007). Brooks' hypothesis led to another type of architecture known as subsumption architecture. Subsumption architecture according to Brooks "is a parallel and distributed computational formalism for connecting sensors to actuators in Robots" (Brooks., 1991). This architecture is a class of reactive architecture. The need for Brooks alternative architecture started with his works where he propounded three key thesis (Michael & Nicholas, 1995):

1. One can generate intelligent behavior without explicit symbolic representation like is being projected in symbolic AI
2. One can generate intelligent behavior without explicit abstract reasoning like is being projected in symbolic AI
3. Intelligence is a property that is unique to certain complex system

Apart from Brooks other authors have also contributed to the development of reactive architecture, and their contributions are as follows (Michael & Nicholas, 1995): Agre and Chapman- PENGI, Rosenschein and Kaelbling-situated automata, Maes-Agent network architecture

Though Brooks architecture has been applauded by many authors it has some draw backs as outlined by Wooldridge (Wooldridg, 2002)

- The agents in Brooks architecture relies on the local information (i.e. the current state) instead of the models of the environment and for it to rely on the local information it is necessary that it has sufficient information therein, in order to determine an acceptable action.
- The decision made by purely reactive agent takes a short-term view as such decision relies only on the agent's current state (i.e. local information) rather than non-local information.
- Since purely reactive agents rely on local information it will be difficult for them to learn from experience and therefore improve in performance over time.
- Though it is easy to build effective agents with small number of layers, it is very difficult to build agents with many layers, as the interaction between the layers will be too complex to understand.

Since Brooks architecture also has some disadvantages researchers then decided to come up with another type of architecture called hybrid architecture. Details of this type of architecture will be discussed in the next section.

2.3 Hybrid (Layered) Architecture

This architecture is obtained by combining the advantages of logic-based architecture and reactive architecture. Since the agent is required to be capable of reactive as well as proactive behavior it is necessary to create different subsystems to handle different types of the agent's behaviors. This subsystem are arranged into hierarchy of interacting layers, and this leads to another class of architecture called layered architecture. There are two types of layered architecture, the horizontal layer and vertical layer. Each of the architectures is characterized by the information and control flow. Examples are TouringMachine and inteRRap.

Horizontal Layering: this type of architecture is composed of two or more layers with each layer having a sensory input and an action output. Based on this each layer acts as an agent thereby making suggestion as to what action to perform. The main advantage of this architecture is the conceptual simplicity. If the agent is required to exhibit n different behavior, then there is need to implement n different layers. However, there is a problem of coherency with the overall behavior of the agent since the layers are in competition over which layer is to generate the action suggestion. In order to correct this anomaly there is a mediator function that determines which layer has control of the agent at any point in time. This mediator function will in turn pose problem for the designer, as the designer will have to decide all possible interaction within the layers. That means that there will be m^n interaction if there are n layers and m suggestions per layer.

Vertical Layering: in this architecture a maximum of one layer is required to handle each of the sensory input and action output. This architecture addressed some of the problems associated with horizontal architecture. The vertically layered architecture comprises of one pass and two pass layered architecture. In the one pass architecture control flows sequentially from the lower layer to the topmost layer where action output is generated. In the two-pass architecture information flows sequentially up while control output flows down the layer. In vertically layered architecture there is n-1 interfaces that exists within the n layers such that if each layer has m actions to suggest then there is a maximum of $m^2(n-1)$ interactions between the layers. The problem with this type of architecture is that it is not fault tolerant any problem with one layer will affect the agent's performance.

2.4 Belief Desire Intention (BDI)

Belief Desire and Intension model developed by Michael Bratman in 1987 is one of the best known and widely studied practical reasoning system within the ATAL community. The main reason why this model was famous is because of the philosophical human practical reasoning. This popularity however have started diminishing since nothing tangible has been done to improve the architecture since it was developed in the mid 80's. Arguably so many architectures have been developed to tackle some of the issues associated with the BDI architecture. The inspiration of this architecture comes from logic and psychology and it is built using the symbolic representation of the agent's belief, desire, and intention.

From our discussion so far, we have seen agent's that cannot take a decision except it has reasoned about what action to take based on the percepts it has received from the environment. In some cases, this action may not yield the desired result as the environment may have changed before the agent finish reasoning. This was the case of Symbolic AI. After that was a case where the agents do not have a symbolic representation of the environment but rather respond to event as it occurs. In this case we say that the agent is highly reactive. That was the case of reactive architecture. The hybrid architecture took the behaviors (proactive and reactive) that are common to two of the architectures (logic based and reactive architecture) and put them into layered hierarchy of subsystems, and transfer control to the layers according to priority. The BDI architecture did not eliminate the need for reasoning or planning nor reactivity, but rather used intention to constrain the amount of reasoning.

The Belief, Desire and Intention can be explained as follows: Belief represent the information the agent holds about the environment which might be false. It might be false if the agent bases its belief on internal reference or faulty equipment like (sensor or camera). Cameras or sensors are some of the means through which the agent obtains information from the environment. Desire are those goals the agent would want accomplished. Desires may be many and may conflict with each other. Intention is the agent's commitment to a plan it want to use to achieve a desire (goal) it is committed to execute.

3. AGENT DEVELOPMENT APPROACHES. A PROGRAMMING LANGUAGE PERSPECTIVE

A programming language is an essential component for the implementation of agent-based system. The language that enables developers build agent-based systems based on the concepts (beliefs, goals, actions, plans, communication, etc) developed by the agent's theorists is called agent-oriented programming.

Though a lot of research has been done on the development of agent-oriented programming, some developers are probably building agents-based system using object-oriented programming language such as Java, C++ and Smalltalk. The reason why it is easy to build agent-based system using OOP is because it shares some propertied (such as encapsulation, inheritance, and message passing) with OOP. They however differ in the use of polymorphism as is commonly obtained in OOP. Apart from using OOP for the implementation of agent-based system, one can still realize agent-based system using some other languages like Pascal, C, Lisp or Prolog. Theses standard languages are the most dominant as the core agent-oriented programming languages are still on the experimental stage.

In other to classify and analyze agent programming language, it is necessary to put into consideration the computational model, programming paradigm, formal semantics, usage of mental attitude, architectural design choices, tools availability, platform integration, application areas etc. in other words we have to take a light top-level classification based on those aspects we consider relevant for agent system. In accordance with this we have our classifications based on the following headings (Bădică, et al, 2011): agent-oriented programming (AOP) languages, belief-desire-intention (BDI) languages, hybrid languages (which is built on a model that combines AOP and BDI) and other (prevalently declarative) language.

3.1 Agent Oriented Programming

The term agent-oriented programming (AOP) was coined by Shoham in 1989 to define a new programming paradigm, based on cognitive and societal view of computation (Shoham, 1995). Though agent-oriented programming is viewed as a specialization of object-oriented programming (OOP) they both differ in some ways. Object and agents have variable degree of autonomy. For an object to execute an action it directly invokes other objects directly. The decision of whether an action should be executed or not lies on the calling object while in AOP an agent expresses a desire that it wants an action to be executed. This request may be granted or turned down depending on the disposition of the receiving agent. The receiving agent has the right over its behavior whether to honor a request to execute an action or to reject the request. An agent is pro-active and adaptive to the environment that means they can set a goal and follow it up to conclusion or on the other hand they can learn from the environment and improve their performance over time. On a final note AOP and OOP share variable thread of control. OOP has single thread of control while AOP is multithreaded.

Before concluding the discussion on AOP it is important to reserve some sections to discuss a very vital concept called programming language. Agent oriented programming language according to Costin Bădică etal "is a tool that provides a high-level of abstraction directed towards developing agents and incorporates constructs for representing all the features defined by the framework."

3.2 AGENT0

AGENT0 was among the pioneers of agent-oriented programming language that was developed for direct implementation of agent-oriented paradigm. The language is still an experimental language, though it gives a feeling of how AOP concept could be used for the development of large scale system. The implementation of agents in AGENT0 consists of definition which are in four parts: capabilities set (that describes what the agent can do), belief set, commitments or intention set, and commitment rules set that contains condition for the message, condition for mental and an action (Wooldridge, 1999). The communication that exists between agents is done through exchange of messages, that can be achieved in one of three different styles: 1) a request to perform an action, 2) an "unrequest" to desist from an action 3) an informative message for information delivery.

The modification of agent's commitment can be achieved through request and "unrequest" while a change in agent's beliefs can be achieved through inform message. In addition, a message can be private or public depending on the context, if the message corresponds to internally generated subroutine then it is private otherwise if the communication exists between other agents in the environment then it is public. The messages can affect the agent's mental state, so it is important to put the agent's mental state in a consistent state by doing one of the following as proposed by Shoham (1995): 1) the use of formal methods and mathematical methods 2) Heuristic methods; 3) simplifying the language for mental state description and so permitting trivial verification (Shoham, 1995). The model for agent execution proposed by Shoham (1995) uses simple loop which regularly iterates by updating set of beliefs and commitments after reading the current message if there is need for that. Secondly executing all current cycle commitments, this in order words result in further modification of beliefs.

3.3 PLACA

PLACA is an immediate successor of AGENT0 with an improvement on the planning facilities which reduces the rate at which communication is generated (Badjonski, 2003). In PLACA an agent conserve its resources by limiting the number of request it sends to other agents in order for the receiving agent to execute an action. The sending agent can limit the message to one request signifying the desired state of affair. When the request is sent, and the agent receives it, it checks to see that the conditions of the rules are satisfied so it can respond with its plan to achieve the desired state of affair. PLACA has introduced plan which differentiates its mental categories from those in AGENT0.

The main disadvantage with PLACA is that it has not been deployed for production purposes rather it is still on prototype stage just like AGENT0.

3.4 Agent-K

Agent-K extends AGENT0 by including standardized KQML which serves as a replacement to custom communication messages (i.e. request, unrequest and inform). This enables interaction between different types of agents by enhancing the general interoperability of agents that employs KQML. There are some improvements brought by PLACA which are not part of Agent-K. if Agent-K is to be unified with PLACA then there is need to modify AGENT0 interpreter to handle KQML messages. The modification of the interpreter has enhanced the possibility of executing multiple actions at the same time i.e. matching the incoming message with multiple commitment rules. Furthermore, in Agent-K KQML messages can be transported via TCP/IP and email to remote server using KAPI library. The unification of KQML with Agent-K has enhanced interoperability of agents, however this is not fully realized as Agent-K encodes agent's belief and commitment using Prolog (thereby enabling communication to exist within Prolog-based agents only).

3.5 April and MAIL

Agent Process Interaction Language-April is a symbolic language and a development tool used for general purpose multiprocessing system (Costin Bădică, 2011). April allows the development of simple agents as it is equipped with all the infrastructures needed for the development and the employment. Agent in the multi-agent paradigm can be represented using "Process", which is a key entity in April. Every agent in April is associated with a private and public handle which helps in the identification of agents. The private handle has its scope limited to agents in the system only, while the public handle maintains an access scope available to agents residing outside the systems. Public handle enhances the building of a global April application through the interconnection of systems to the name server.

April allows access to non-April based applications using TCP/IP based communication infrastructure. The main disadvantage of April is that it does not have a global synchronization clock that means April system cannot guarantee the other of arrival when three agents are involved in communication. April system cannot be relied on when time critical real-time application is involved, this is because even though the time of arrival of messages involving one agent can be determined it is difficult to determine the execution time for an operation.

April gives developers the flexibility of defining new language constructs, based on existing one using "macro". Macro enhances the extension of April by allowing the development of new, richer and agent-oriented languages. This extension was intended for the creation of a new agent-oriented programming language called MAIL. The intention of MAIL as a high-level language was to realize a couple of MAS. April and MAIL was first introduced as part of the ESPIRIT project. MAIL though did not go beyond the prototype level, relied on April as the implementation language.

3.6 Concurrent MetateM

Concurrent MetateM is among the first programming language developed for the implementation of multi-agent system. In this kind of programming language each agent is programmed by given it the kind of behavior it should have through the provided temporal logic specification. It has some logical formulas associated with it which forms the basis for its execution. There is a strong connection that exists between logic and languages in concurrent metateM, just like the connection that exists between AGENT0 and PLACA which is made possible through the logic that illustrates their metal category.

In concurrent MetateM there is an associated interface and computational engine from which the definition of every agent lies. The agent identifier, sets of symbols the agent can recorganise and send are the constituents of the agent's interface that helps in the description of the agent. Computational engine on the other hand is made up of rules that helps to define an agent.

3.7 BDI Based Language

The success recorded in practical reasoning agent architecture has contributed to the trend in the development of agent programming language. PRS (Procedural Reasoning System) is about the first system to be developed that is embodying belief, desire and intention. Based on this concept several programming languages have been developed for the BDI based agent architecture.
Some of the programming languages that support BDI architecture and belong to hybrid paradigm will be presented in the following subsections.

3.8 AgentSpeak

AgentSpeak is used to enhance the design of agent programming language by using belief, desire and intention of the BDI architecture (Costin Bădică, 2011). The main reason for the development of AgentSpeak was to get a language that captures the essential features from both the BDI architecture for agents and for object-based concurrent

programming. AgentSpeak has an agent family which behaves like a class in object oriented programming. Every agent in the agent family (instance of the agent family) has a private and public areas. The private area is accessible to agents withing the family while the public area is accessible to agents outside the faimily. The private service of an agent helps it to fulfil its desire while the public services help other agents to invoke it.

There is a communication among agents in AgentSpeak. This communication is initiated through synchronous or asynchronous message exchange. An agent or the agent family is intended to exchange message, every message inteded for agent family is distributed to every instance of the family. A message can be an inform speech act or a request depending on whether there is or no obligation upon the receiving agent.

3.9 Jason

Jason is an interpreter used for the implementation of AgentSpeak(L). it uses Java programming language. It belong to hybrid agent paradigm. Though Jason has syntax similar to Prolog it has a different semantics based on AgentSpeak. The integration of jason with Java is a strength though it has some consequences like: 1) Java is used for tailoring the behavior of Jason interpreter 2)In other to build a situated agent Jason provides Java API that will be used to integrate with Java based environment model. 3) Some agent framework like JADE (Fabio Bellifemine, Giovanni Caire, Dominic Greenwood, 2007), AgentScape (Rogier C. van het Schip, 2009), and Agent Factory has been integrated with Jason.

3.10 AF-APL

AF-APL was extended from Agent-Oriented Programming to include some concepts in BDI (hybride paradigm). AF-APL is defined to be a “practical rule based language” which relies on commitment rules. A commitment rules unifies the following mental attitudes: Beliefs, plans and commitment. AF-APL language derives its syntax and semantics from a logical model which is based on how agent commits itself to a course of action (Robert RossRem CollierG. M. P. O’Hare, 2005). The formalization of AF-APL semantics by Rem Collier is based on multi-modal first-order branching-time logic.

AF-APL language has included in its development environment the flexibility to enable the developer to explicitly declare each agent to have a set of sensors(situated agents) also called perceptors and a set of effectors(actuators). Both perceptors and actuators are instances of Java class. Perceptors is responsible for the conversion of raw sensor data into beliefs while an actuator has the following responsibilites: 1) to identify an action using the defined action identifer. 2) to implement the actoion using its code.
AF-APL programming language and Agent Factory Framework have a strong relationship in their approach to the agent system development and deployment.

3.11 3APL

3APL has neitheer come from AgentSpeak(L) nor from Agent0, however AOP and BDI language family according to Costin Badica etal contributed to the developemtn of the language (Costin Bădică, etal, 2011). This idea is from the fact that both language families (AOP and BDI) emernated as a result of the influence of general setting of intentional stance in the direction of the understanding and development of a software system. Though 3APL in theory is a s expressive as AgentSpeak(L) the development of 3APL seem not to be supported any more. That notwithstanding 3APL has led to the development of GOAL oriented agent-programming language, and has integrated ideas from AOP, BDI and Logic within a single programming model with declarative goals (hybrid paradigm). 3APL has been succeeded by 2APL which is still being developed.

3.12 2APL

2APL is a direct offspring of 3APL and has enhanced it in the aspect of disintegrating the multi-agent concern from individual agents. The multi-agent aspect handles the specification of agent set, the external environment set and their relationship. The individual agent concept shares some similarities with programming notion of BDI and AOP language through the covering of beliefs, goals, plans, events, messages and rules. The major difference between 2APL and GOAL is that 2APL is an amalgamation of declarative and imperative programming styles, it is equally said to be a "practical programming language" while GOAL is a declarative language to the core. 2APL also extends 3APL by allowing better testing and debugging (Dastani, 2008).

3.13 JACK Agent Language

JACK Intelligent Agent also referred to as JACK is used for commercial development of intelligent agent system. It was developed by Autonomous Decision-Making software – AOS. It comprises of the following components: JACK Agent Language (JAL), JACK Compiler, JACK Kernel and JACK Development Environment. JAL enhances the development of agent-oriented programming in accordance with BDI model using the constructs provided by Java language. The JACK compiler helps in translating JAL syntax into Java source code. JACK Kernel is the JACK runtime engine used to run the resulting Java source code.

Because the development of distributed agent application is to be supported in JACK, it allows agents to be developed such that they can be deployed in separate processes, possibly running on different machines in a network. With JACK agents there is the ability for a peer-to-peer exchange of messages.

4. OTHER AGENTS LANGUAGES

Some of the agent's languages which do not use mental attitudes for putting the languages into shape are categorized into the generic class of "other languages". Agent languages in this class rather employ other useful constructs for building intelligent software agents. In other words, they support reasoning tasks that relies on formal logic, methods and calculi which are part of the main characteristics attributed to agents. Among the agent-oriented programming languages, there is a class that relies and can be characterized as declarative paradigm. Such class will be discussed in the following subsections.

4.1 GOAL

The reason for the development of Goal-Oriented Agent Language was to bridge the gap that exist between agent logic and agent programming model (BDI and AOP). This language unifies different concepts from three main languages like the commitment from Agent0, intentions from AgentSpeak (L) and goals from 3APL and by so doing it introduces the declarative perspective of goals in agent programming language. Though GOAL has been used for planning applications in the area of transportation and logistics, the bulk of the development is still a prototype used for educational purposes.

4.2 Golog

Some agent programming languages belong to the family of logic languages (declarative paradigm) and Golog (alGOL in LOGic) is one of them. For dynamic system specification John McCarthy of AI developed a situation calculus. Situation calculus according to Lin (2008) "is a logical language for representing changes" (Lin, 2008). It was on the bases of situation calculus formalism that GOLOG was developed. Situation calculus is made up of the following major concepts situation, action and fluent. The action concept is used to model changes in the world, situation concept is used to model histories of the world while relation and functions which depends on situation is represented by fluent. The fluent is composed of relational and functional fluent.

GOLOG has other languages included in its family. This language are: GOLOG which is the core language (Reiter, 2001), conGOLOG i.e. concurrent GOLOG (Giuseppe De Giacomo, etal, 2009), indiGOLOG i.e. incremental deterministic GOLOG (Giuseppe De Giacomo, etal, 2009). Since GOLOG can be used to achieve BDI-style of agent programming it is now clear that GOLOG can be used to bridge the gap that exist between BDI and action logic agent programming style. GOLOG exerts some influence in programming physical robots with cognitive capabilities endowment. Even though GOLOG was modeled for single robotic agents, there has been several proposals on the extension of GOLOG to be used for multi-agent systems in a game-theoretic setting, called GTGOLOG (Alberto Finzi, etal, 2004).

4.3 FLUX

Fluent executor – FLUX belong to the family of logic programming language (declarative paradigm) with its implementation based on fluent calculus. Fluent calculus is an improvement on situation calculus, however they have some differences which is based on what the situation represents. In fluent calculus situation is a representation of state description while in situation calculus, situation is a representation of histories of occurencies of action. Being a logic-oriented programming language FLUX shares some relationship with other agent oriented programming language though their difference is that FLUX has its main focus on developing a program that implements single agent that exhibits logical actions in a dynamic environment rather than implementing a program for multi-agent system development. Several works have been conducted to see how FLUX interpreter could be used in multi agent system to solve complex problems, through the cooperation of multiple agents.

5. A WSN BASED SURVEILLANCE AND INTRUSION DETECTION

The main aim of Wireless Sensor Network (WSN) is to connect different types of sensors, such as ultrasonic, thermal, seismic, camera, and motion detectors with the intention of gathering information from them and reporting it to the control station which in turn aggregate the information, process them and take appropriate action. Several research has been conducted on how to use wireless sensor network for surveillance and intrusion detection and we are going to review such works in this section.

5.1 Energy efficient and stealthy tracking of moving target

He et al. (2013) have proposed a stealthy and energy efficient detection of the position of moving object using WSN. Figure 2.2 shows the actual deployment of the sensor devices in the test field and the high-level diagram of the topology. In this project they deployed 70 MICA2 motes along 280 feet long perimeter along the passage way so they can detect and report the presence of a moving vehicle. Each mote has 433 MHZ Chipcon radio to enable communication. This radio is not capable of long distance (>1000 ft) transmission so the sensors, in other to transmit over a long distance, may in real system be supported with repeaters that will help to relay back information from the motes in the field to the base station. Each mote is accompanied with a magnetic, acoustic, and a photo sensor. The detection of moving target is captured by the magnetic sensor while the stealth capability is achieved by the minimization of the RF transmission and minimal exposure.

For effective power utilization and collaborative detection and tracking of events the system uses two components the sentry service component and group management component respectively. The sentry service component divides the motes into two subsets called sentries, the first subset is responsible for monitoring events while the second subset is put in low-power state waiting for an event to occur before being awaken by other sentries. In case of an event occurrence the motes in low-power state are awaken by the sentries in the region while the group management component arranges the motes into dynamic groups, so they can collaborate in tracking the event.

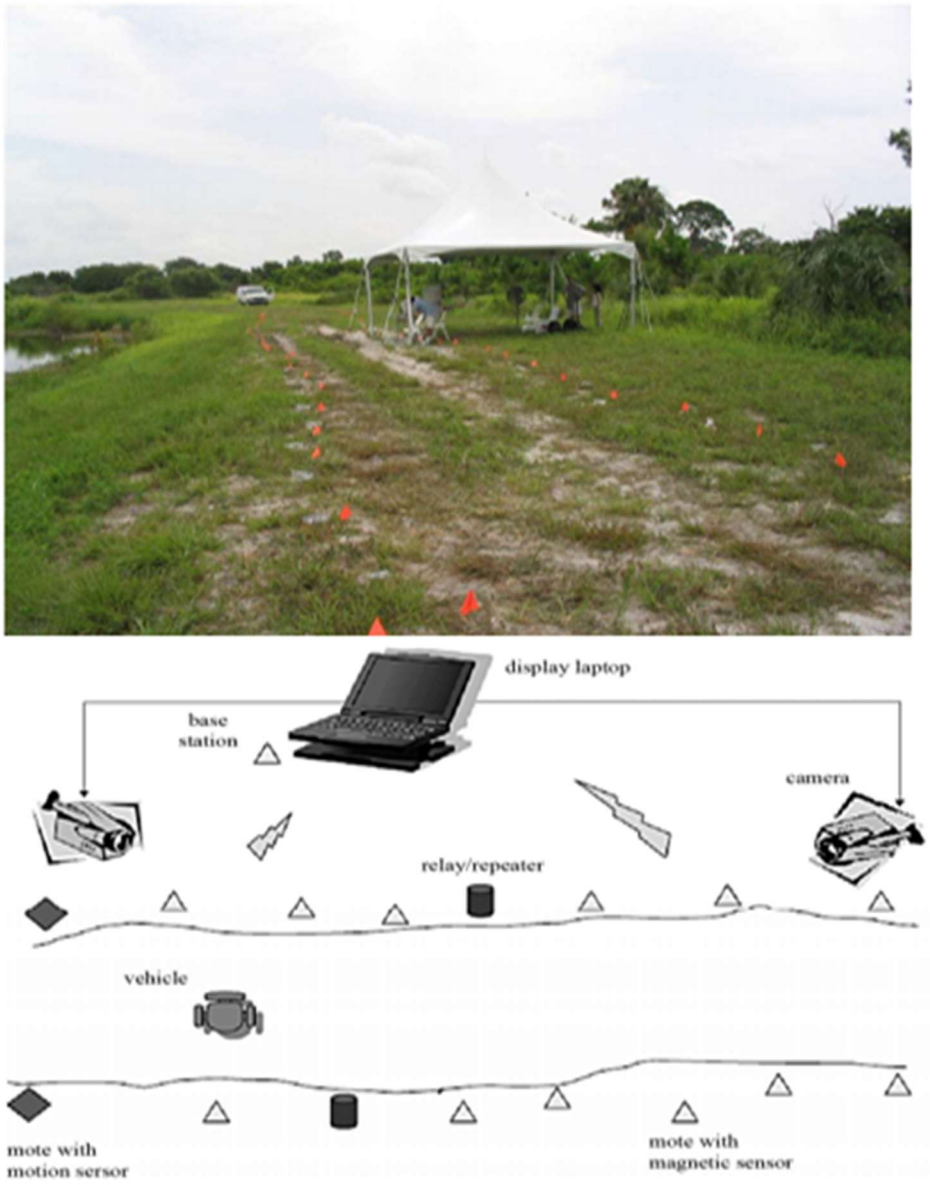

Figure 1: Actual deployment in the test field and a high-level diagram of the topology (Tian He, et al, 2004)

5.2 Evader tracking and detection using WSN

Sinopoli et al (2003) have proposed an evader tracking and detection based on WSN. The concept is used in pursuer evasion game where two teams (the pursuers and evaders) are involved in competition. The WSN is used to help the pursuer team detect and track the evader team. In this system the game is played in an environment where the sensor network is deployed so as to enable it to cooperate with the pursuer in other to track the evader. The SN is equipped with the capabilities of multi-vehicle tracking in other to distinguish between evader's vehicles from the pursuer's vehicle.

For optimum performance the network is provided with a dynamic routing structure in other to reduce the interval used to deliver information to the pursuer. The network has a security features that enables it to protect its information from the evader team. The control algorithm provided by the network helps to keep it alive even when any one of the nodes fail.

Furthermore, with the help of WSN there is a great improvement in the performance of PEG as the visibility of the field is greatly enhanced and also there is equally an increase in the communication distance. This is in contrast to what is obtained without WSN where the pursuers employed limited capability sensing techniques, such as computer vision or ultrasonic sensing which in turn make them to have relatively small detection range.

5.2.1 Line in the Sand

Arora, et al (2005), have proposed a system for the detection, tracking and classification of moving object into metallic and non-metallic object. To achieve this the authors deployed 90 sensor motes which have a combination of magnetometer and micro-power impulse radar sensors, in the target area. The sensor nodes collaborate in a network to detect and classify a moving object as metallic or non-metallic. A sample of enclosed mote used in the project is shown in figure 2.3. The project uses “influence field” which is the number of sensors within the range of the object under consideration. In this project the shape of the object is equally detected, classified and tracked.

Figure 2: A sample of mote enclosure for Line in the Sand project

5.2.2 Border Sense

Zhi, et al. (2011) proposed a hybrid wireless sensor network architecture to improve the detection accuracy of current border patrol system while bringing human involvement in surveillance and detection to the barest minimum. Border Sense according to Zhi, et al. (2011) “is a coherent system that coordinates various technologies, including unmanned aerial vehicles, unattended ground/underground sensors, and surveillance towers equipped with camera sensors”. The traditional border patrol system has some advantages which include provision of accurate detection result and large detection range, as is obtained in camera sensors; when the intrusion is not within the region of cameral sensor line-of-sight, the ground/underground sensors provides a detection functionality. Once the intruders have been detected, the mobile sensor keep track of them. All these advantages are combined together in Border Sense to enhance the traditional WSN border surveillance technique.

5.2.3 Wireless Sensor Networks/Underwater approach for Marin Surveillance

Marin surveillance and border protection uses the same application used for border surveillance of ground sensor networks. Mahdy et al, have shown how acoustic sensor deployed on a shallow water surface could help in keeping surveillance and detecting the enemy watercraft. In addition Luo, et al. equally proposed a wireless intrusion detection for detecting intrusion caused by ships (Felemban, 2013). Their approach bases its surveillance on the V-Shaped wave generated on the water surface by the ship movement. They deployed three-axis accelerometer sensors with iMote2 on buoys on the sea surface. The deployment can be shown in Figure 2.4 below.

The deployment is done in a predefined location using grid topology. Before the deployment the nodes are synchronized to reflect time of events. Accelerometers are used in the network to detect the speed of the passing ships. The major issues with underwater sensors is that they have slow bandwidth, noisy channel, unreliable links and insufficient energy.

Figure 3: An experimental view of sensor network deployed for detection of ship movement (Felemban, 2013)

5.2.4 Application of Nerual Network on WSN for Border Protection

Michael et al. (1995) proposed the training of Neural Network for intrusion detection based on known pattern of intrusion (Felemban, 2013). The proposed system will have information transmitted to the base station from a set of 32 MicaZ sensor nodes with the accompanying microphone and light sensors. The system can detect a single and group intrusion from nodes distributed along a perimeter. The experiment proves that the time of detecting intrusion was minimized and the probability was increased when both light and sound sensors are used.

5.2.5 FleGSens: A Wireless Sensor Network based Surveillance

Dudek, et al., (2009) proposed a system for trespasser detection using only passive infrared sensors (PIR Sensors). The system consists of two protocols: the trespasser detection protocol and node failure detection protocol. The trespasser detection protocol detects a trespasser and sends a signal to a dedicated gateway while the node failure detection protocol sends a signal to the gateway to inform it that a node has failed to respond. This is to ensure that the integrity of the network is maintained throughout the life time of the network. The project was implemented using 200 wireless sensor nodes called iSense. The iSense consist of Passive InfraRed (PIR) sensor covering a 500 m-long land strip.

5.3 Categorical review of Wireless Sensor Network Environments

Wireless Sensor Network faces different challenges and constraints depending on the environment where they are deployed. Currently sensor networks can be deployed on land, underground and underwater. The WSNs are categorized in accordance with the environment where they are deployed. There are five types of WSNs: The terrestrial WSN, underground WSN, multi-media WSN and mobile WSN (Jennifer Yick, Biswanath Mukherjee, Dipak Ghosal, 2008) .

5.3.1 The Terrestrial WSN

Terrestrial WSN is made up of inexpensive wireless sensor nodes ranging from hundreds to thousands of them deployed in a specified region. The deployment can be in ad hoc or in pre-planned manner. In ad hoc deployment there is no specific arrangement of the sensor nodes. The sensor nodes can be deployed from an air plane or placed in a random manner in the area under consideration. In the pre-planned deployment, the nodes has the following placement models (Jennifer Yick, Biswanath Mukherjee, Dipak Ghosal, 2008): grid placement, optimal placement, 2-d and 3-d placement models. In a terrestrial WSN it is important to have a reliable communication no matter how densely populated (i.e. with sensor nodes) the environment might be. That means, the sensor nodes should be able to effectively communicate back information to the base station.

In terrestrial WSN the network remains available so long as the battery remains charged. Thus, there is need for secondary power supply (i.e. equipping the sensor nodes with solar cells) since it might be difficulty to replace or recharge the battery. That notwithstanding the sensor node should be able to optimize the energy usage by applying the following approaches: multi-hop optimal routing, short transmission range, in-network data aggregation, eliminating data redundancy, minimizing delays, and using low duty-cycle operations.

5.3.2 Underground WSNs

Underground WSNs is a SN buried underground, or in a cave or mine, that communicate with the associated nodes underground so as to monitor underground conditions. The buried sensor nodes communicate with the sink node above ground to relay information to the base station. When compared with the terrestrial network the underground network is more expensive to equip, maintain, and deploy. The reason why underground WSN is expensive is because the nodes communicate over the soil, rocks, water and other mineral contents and they need to be equipped appropriately in other to ensure reliable communication. In order words the expense is due to the communication medium. Due to the environment where Underground WSN operate it usually possess some challenges; like signal loses and high levels of attenuation. Since replacement or recharging of the battery is usually an issue it is important to consider energy and cost during deployment. The deployment of underground WSN is done in a planned manner and not in ad hoc manner like in terrestrial WSN. Just like in terrestrial network the key objective is to plan the deployment (by implementing efficient communication protocol) such that energy can be conserved, and the life time of the network increased.

5.3.3 Underwater WSNs

Underwater WSNs are sensor network established by deploying sensor nodes and vehicles underwater. When compared to terrestrial WSN, underwater sensors are more expensive and sparsely deployed. The autonomous water vehicles are used for exploration and gathering of information from the underwater sensor nodes. Communication is established in underwater WSN through the transmission of acoustic waves. Due to the communication environment underwater sensor network is associated with the following issues: limited bandwidth, long propagation delay, signal fading and sensor node failures. When deployed the sensor nodes configure itself so as to be able to adapt to the harsh ocean environment. Just like terrestrial and underground network energy is an issue so efficient communication and networking technique has to be adapted in node deployment as battery cannot be recharged or replaced after deployment.

5.3.4 Multimedia WSNs

Multimedia WSNs consist of low cost multimedia nodes deployed in an environment with the intension of tracking and monitoring multimedia events such as video, audio and images. This node are wirelessly connected to each other so that they can retrieve, process, correlate, and compress data. The deployment of nodes in multimedia WSN is done in a pre-planned manner such that it effectively covers the area of interest. The main disadvantage of multimedia WSN

is as follows: high bandwidth demand, high energy consumption, quality of service (QoS) provisioning, data processing and compression technique and cross-layer design. The bandwidth required to deliver a video content is very high, this leads to high data rate which grossly affect the energy consumption in the network. In that case there is need to deploy a transmission technique that supports high bandwidth and low energy consumption. The major issue with QoS provisioning is that the variable channel capacity and delay affects the integrity of the data. In WSN time is of essence, the variation in the time of data capturing and delivery should be as small as possible, otherwise the transmitted data will be useless. Performance in terms of filtering, extraction of redundant information and merging contents within the multimedia WSN can be improved if processing, filtering, and compression of data can be done within the network.

5.3.5 Mobile WSN

Mobile WSN is made up of nodes that have the ability to move without help in other to interact with the physical environment. The mobile nodes can reposition and organize itself in other to sense, compute, and communicate information from the environment. Mobile nodes differ from static node in their ability to reposition and organize themselves in the network. Mobile nodes are dynamic they don't remain the way they are after deployment, they spread around the environment to gather information. If two mobile nodes are in the communication range of each other they can exchange information. In mobile WSN data distribution is done using dynamic routing, this is in contrast to static WSN which uses fixed routing or flooding. The main issues associated with mobile WSN are localization, deployment, self-organization, coverage, energy, navigation and control, maintenance, and data processing. Mobile WSN can have diverse application ranging from environmental monitoring, target tracking, real-time monitoring of hazardous material, search and rescue, etc. The main advantage of mobile WSN is that it can function well in area where manual deployment is not feasible may be because of the risk associated with the deployment. In such cases once the nodes are deployed through an air plane or in a random manner they can move to the direction of event and provide appropriate coverage.

6. CONCLUSION

Wireless agents remain an essential component used in our society. To realize the full potential of the agent system it is vital to choose the best architecture that will optimize the intelligent of the agent. This paper reviewed the agent's architectural trends with emphases on the logic based, reactive, hybrid and belief desire intention architecture.

REFERENCES

Aaron Hector, V.L. Narasimhan. (2005). A New Classification Scheme for Software Agents. *Proceedings of the Third International Conference on Information Technology and Applications.* IEEE.

Allen Newell, Herbert A. Simon . (1976). Computer Science as Empirical Inquiry: Symbols and Search. *Communication of the ACM*, 113-126.

Badjonski, M. (2003). *Adaptable Java Agents (AJA) – a Tool for Programming of Multi-Agent System.* Ph. D Thesis.

Brooks., R. (1991). How to Build Complete Creatures Rather than Isolated Cognitive Simulators. *CiteSeer*, 225--239.

Brustoloni, J. C. (1991). Autonomous Agents: Characterization and Requirements.

Dastani, M. (2008). 2APL: A Practical Agent Programming Language. *International Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)*, 214-248.

Fabio Bellifemine, Giovanni Caire, Dominic Greenwood. (2007). *Developing Multi-Agent Systems with JADE.* Chichester: John Wiley & Sons Ltd.

Felemban, E. (2013). Advanced Border Intrusion Detection and Surveillance Using Wireless Sensor Network Technology. *International Journal of Communications, Network and System Science*, 251-259 .

Jennifer Yick, Biswanath Mukherjee, Dipak Ghosal. (2008). Wireless sensor network survey. *Elsevier*, 2292-2330.

Lin, F. (2008). *Handbook of Knowledge Representation.* Elsevier B.V.

Maes, P. (1995). Artificial Life Meets Entertainment: Lifelike Autonomous Agents. *COMMUNICATIONS OF THE ACM*, 108-114.

Mark Burgin, Gordana Dodig-Crnkovic. (2009). A Systematic Approach to Artificial Agents.

Michael Winikoff, Lin Padgham . (2004). *Developing Intelligent Agent Systems A practical guide.* Liverpool: John Wiley and Sons Ltd.

Michael Wooldridge, Nicholas R Jennings. (1995). Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 115-152.

Nilsson, N. J. (2007). The Physical Symbol System Hypothesis: Status and Prospects. *Springer*, 9-17.

Reiter, R. (2001). *Logical foundations for specifying and implementing dynamical systems.* MIT Press.

Robert RossRem CollierG. M. P. O'Hare. (2005). AF-APL – Bridging Principles & Practice in Agent Oriented Languages. *Springer*, 66-88.

Rogier C. van het Schip, e. a. (2009). Deploying BDI agents in open, insecure environments. *Proceedings of the 7th European workshop on multi-agents systems.*

Schermer, B. W. (2007). *Software agents, surveillance, and the right to privacy: a legislative framework for agent-enabled surveillance.* Leiden University Press.

Shoham, Y. (1995). Agent-oriented programming. *Elsevier* , 51-92.

Stan Franklin, Art Graesser. (1996). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*.

Stuart J. Russell, Peter Norvig. (2010). *Artificial Intelligence A Modern Approach.* New Jersey: y Pearson Education.

Virdhagriswaran, S. (n.d.). Retrieved from t [http://www.crystaliz.com/logicware/mubot.html]

Wooldridg, M. (2002). *An Introduction to MultiAgent Systems.* West Sussex: JOHN WILEY & SONS, LTD.

Wooldridge, M. (1999). *Multiagent systems: a modern approach to distributed artificial intelligence.* Cambridge: MIT Press.

Zhi Sun, Pu Wanga, Mehmet C.Vuran. (2011). BorderSense: Border patrol through advanced wireless sensor networks. *Elsevier*, 469-477.