International Journal of Engineering Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org ||Volume 6 Issue 2|| February 2017 || PP. 15-20

The Evaluation of Generic Architecture for Information Availability (GAIA) and Multi agent System Engineering (MaSE)

Ugwa Chioma; Chizoba Ezeme, & Obinnaya. C. Omankwu;

Abstract: *Along with the growing interest in agent applications, there has been an increasing number of agent-oriented software engineering methodologies proposed in recent years. These methodologies were developed and specially tailored to the characteristics of agents. The roles of these methodologies can provide methods, models, techniques, and tools so that the development of agent based system can be carried out in a former and systematic way. The goal of this paper is to understand the relationship between two key agent-oriented methodologies: Gaia, and MaSE. More specially, we evaluate and compare these three methodologies by performing a feature analysis, on them, which is carried out by evaluating the strengths and weaknesses of each participating methodology using an attribute-based evaluation framework. This evaluation framework addresses some areas of an agent-oriented methodology: concepts, modeling language, process and pragmatics.*

I. Introduction

The role of software engineering is to provide methodologies which is a set of methods, models and techniques that make it easier to handle the complexity of the software development process increasing the quality of the resulting systems. Thus, the role of agent-oriented methodologies is to assist an agent-based application in all of its life cycle phases [Ghezzi etal,1991].

Agent Oriented (AO) methodology aims to prescribe all the elements necessary for the development of a software system, especially in the context of commercial applications. Most AO methodologies are in an early stage and still in the first context of mostly "academic" methodologies for agent-oriented systems development, although many of these methodologies have been tested in small, industrial applications (Martin etal,1995).

II. Agent Background Study

An Agent is an entity that is not only designed to run routine tasks commanded by its users, but also to achieve a proposed setting or goal within the context of a specific environment. The difference between an Agent and a traditional software entity is that the latter just follows its designed functions, procedures or macros to run deterministic codes. The former incorporates the ability to practice intelligence by making autonomous/semiautonomous Decisions based on dynamic runtime situations [Jennings etal, 1998].

III. Agent-Oriented Software Engineering (AOSE)

Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software [Jennings etal, 1998]. By definition, AOSE is the application of agents to software engineering in terms of providing a means of analyzing, designing, and building software systems [Nicholas,1999].

IV. Agent-Oriented (AO) Methodologies

Methodologies are the means provided by software engineering to facilitate the process of developing software and, as a result, to increase the quality of Software products. By definition, a software engineering methodology is a structured set of concepts, Guidelines or activities to assist people in undertaking software development [Wood etal,2000]. Object-oriented methodologies generally do not provide techniques and model to the intelligent behavior of agents [Jennings etal, 1998]. Therefore, there need to be software engineering methodologies, which are specially tailored to the development of agent-based systems. In answering that demand, there have been an increasing number of agent-oriented methodologies proposed in recent years.

We tend to focus on several prominent agent-oriented methodologies to examine them in depth in order to identify their strengths, weaknesses, domains of applicability as well as commonalities and differences between them. The selection of AO methodologies were based on several factors such as the methodology's significance and relevance with respect to the field of agents, and its available resource such as documentation, tool support, etc.
The two methodologies which were chosen in this research are: Gaia, and MaSE.

V. Gaia (Generic Architecture for Information Availability)

Gaia is one of the first methodologies which is specially tailored to the analysis and design of agent-based systems. Its main purpose is to provide the designers with a modeling framework and several associated techniques to design agent-oriented systems [Wooldridge etal,2000]. Gaia, proposed originally by M. Wooldridge et al. (2012), where the foundation of analysis is based on an Object Oriented design method called Fusion, from which it borrows terminology and notations.

Gaia is rooted in conceptual organizational modeling (Zambonelli, Jennings, & Wooldridge, 2010) and suggests that developers think about building Agent-based systems as a process of organizational design.

The Agent computational organization is viewed similarly to human organization consisting of interacting roles and functions.Gaia is one of the rst methodologies which is specifically tailored to the analysis and design of agent-based systems [Wooldridge et al. 2012]. Its main purpose is to provide the designers with a modelling framework and several associated techniques to design agent-oriented systems. Gaia separates the process of designing software into two di erent stages: **analysis** and **design**. Analysis involves building the conceptual models of the target system, whereas the design stage transforms those abstract constructs to concrete entities which have direct mapping to implementation code. Figure 2.1 depicts the main artifacts of each stage: **Role Model** and **Interaction Model** (Analysis), and **Agent Model**, **Services Model**, and **Acquaintance Model** (Design). A detailed description of the process steps which the developers need to follow to build these models is described below.

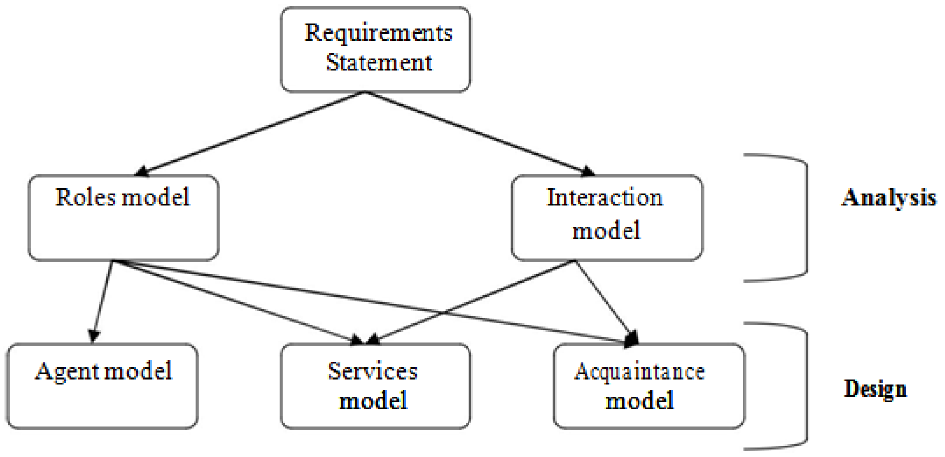

Figure 2.1: Relationship between Gaia's models, source from Wooldridge & Jennings

VI. Analysis

It is noted that, Gaia assumes the availability of a requirements specification. It means that before beginning the Gaia's development process, the analysts need to have a reasonable under-standing of what the system should do. These requirements form the overall objectives of the system which influence the analysis and design phase.

Gaia encourages the developers to view an agent-based system as an organisation. The software system organisation is similar to a real world organisation. It has a certain number of entities playing different roles. For instance, a university organisation has several key roles such as administration, teaching, research, students, etc. These roles are played by different people in the university such as managers, lecturers, students, etc. Inspired by that analogy, Gaia guides the designers to the direction of building agent-based system as a process of organisational design.

At the first step of the analysis phase, Gaia requires the analysts to de ne key roles in the system. At this step, these roles only need to be listed and described in an informal manner. The main purpose of this step understands what roles exist in the system and roughly what they do. Gaia calls the artifact of this step a prototypical roles model. Different roles in an organisation interact with each other to achieve their own goals and also to contribute toward the overall goals of the organisation. These interactions need to be defined in the next step of the Gaia analysis phase. The product of this step is the interaction model. This model consists of a set of protocol definitions for each role. Each protocol definition defines the purpose, the initiator, the responder, the inputs, the outputs and the processing during the course of the interaction. The valid sequence of messages involved in a conversation, however, is not required at this stage. Instead, Gaia focuses on the basic nature and purpose of the interaction and abstracts from exact instantiation details.

The final step of Gaia analysis phase involves elaborating the key roles identified in the first step. This process includes the identification of permissions of roles, their responsibilities as well as the protocols and activities in which they participate. This detailed description of a role is depicted by a Role Schemata. A set of Role Schemata forms the Role Model, which is considered as the key artifact of this analysis phase.

Responsibility defines the functionality of a role and is divided into two types: liveness properties (something good happens") and safety properties [Arazy, 2012]. Permissions define which resources the agents playing that role can and cannot use when performing a particular action. Activities are private" actions which do not involve interactions with other roles.

Protocols are actions that involve interactions with other roles and are derived from the protocol model built in the previous step. It is noted that the Gaia analysis process is not purely linear as described. In contrast, the analysts are encouraged to go back to add a new role, new protocols or move forward to add new permissions, activities, etc.

VII. Design

Having finished the analysis phase, the analysts basically complete building the conceptual model of the system with abstract entities. These entities do not necessarily have any direct realization within the system. They can now move to the second phase (i.e. the Design phase) where those abstract entities are transformed into concrete entities which typically have direct counter-parts in the run-time system.

The design stage requires the developers to build three models. First, an agent model which includes various agent types is constructed. Agent types are the counterparts of objects in object-oriented approaches. They are basic design units of an agent-based system and their realizations at run-time are agent instances. Agent types in the system under development are defined on the basis of the roles that they play. Therefore, the important feature of this step is to map roles identified in the analysis phase to agent types. A role can be mapped to one or more agent types and vice versa. Some general guidelines are proposed to help this process. For instance, a close relationship between several different roles indicates that they can be grouped together in a single agent type.

The second artifact developed in Gaia's design phase is a service model which depicts the services that each role provides. A service is a coherent, single block of activity in which an agent will engage". Each service should be represented by its properties: inputs, outputs, pre-conditions and post-conditions. Inputs and outputs are derived from the protocol model. Pre-conditions and post-conditions which define the constraints on services are derived from the safety properties of a role.

The final model which the designers need to complete is the acquaintance model. It depicts the communication links existing between agent types. It is in fact a directed graph in which nodes represent agent types and arcs show communication pathways. Gaia does not address implementation and there is no tool support that we, or Michael Wooldridge, one of the authors of Gaia, are aware of.

Critique of Gaia Methodology:

1. Goals implicitly coincide with subdivisions of the system, which potentially increase the modeling complexity. There is also no clear guideline on how to derive roles from the organizational model.
2. It is difficult to model Agents entering and exiting sub-organizations; or, adapting to the evolution of organizational structure. There is a lack of dynamic reasoning (Juan, Pearce, & Sterling, 2002)
3. Organizational metaphor is a strongly embedded abstraction coded in the Gaia methodology.

Multiagent Systems Engineering (MaSE)

Multiagent Systems Engineering (MaSE) [DeLoach,2001] is an agent-oriented software engineering methodology which is an extension of the object-oriented approach.

As a software engineering methodology, the main goal of MaSE is to provide a complete-lifecycle methodology to assist system developers to design and develop a multi-agent system. Similar to Gaia, it also assumes the availability of an initial requirements prior specification to the start of software development under the methodology process [Caire etal,2001]. The MaSE methodology is a specialization of more traditional software engineering methodologies. The general operation of MaSE follows the phases and steps shown below and uses the associated models.

Phases	**Models**
1. **Analysis Phase**	
a. Capturing Goals	Goal Hierarchy
b. Applying Use Cases	Use Cases, Sequence Diagrams
c. Refining Roles	Concurrent Tasks, Role Model

2. **Design Phase**

a. Creating Agent Classes	Agent Class Diagrams
b. Constructing Conversations	Conversation Diagrams
c. Assembling Agent Classes	Agent Architecture Diagrams
d. System Design	Deployment Diagrams

Similar to Gaia, it also assumes the availability of an initial requirements prior specification to the start of software development under the methodology process. The process consists of seven steps, divided into two phases. The **Analysis** phase consists of three steps: **Capturing Goals**, **Applying Use Cases**, and **Refining Roles**.
The remaining four process steps, **Creating Agent Classes**, **Constructing Conversations**, **Assembling Agent Classes**, and **System Design**, form the **Design** phase (Figure 2.4).

Analysis Phase
Once the concurrent tasks of each role are defined, the Analysis phase is complete. The MaSE Analysis phase is summarized as follows:
1. Identify goals and structure them into a Goal Hierarchy Diagram.
2. Identify Use Cases and create Sequence Diagrams to help identify roles and communications paths.
3. Transform goals into a set of roles.
(a) Create a Role Model to capture roles and their tasks.
(b) Define role behavior using Concurrent Task Models for each task.

Design Phase
There are four steps to the designing a system with MaSE. The first step is Creating Agent Classes, in which the designer assigns roles to specific agent types. In the second step, Constructing Conversations, the conversations between agent classes are defined while in the third step, Assembling Agents Classes, the internal architecture and reasoning processes of the agent classes are designed.
Finally, in the last step, System Design, the designer defines the number and location of agents in the deployed system.
Once the Deployment Diagrams are finished, the Design phase is complete.
The MaSE Design Phase can be summarized as follows:
1. Assign roles to agent classes and identify conversations.
2. Construct conversations, adding messages/states for robustness.
3. Define internal agent architectures.
4. Define the final system structure using Deployment Diagrams.

Agent Tool
The agentTool system (DeLoach and Wood, 2001) has been developed to support and enforce MaSE. Currently agentTool implements all seven steps of MaSE as well as automated design support.

Applications
MaSE has been successfully applied in many graduate-level projects as well as several research projects. The Multiagent Distributed Goal Satisfaction project used MaSE to design the collaborative agent framework to integrate different constraint satisfaction and planning systems. The Agent-Based Mixed-Initiative Collaboration project also used MaSE to design MAS focused on distributed human and machine planning. MaSE has been used successfully to design an agent-based heterogeneous database system as well as a multiagent approach to a biologically based computer virus immune system. More recently, we applied MaSE to a team of autonomous, heterogeneous search and rescue robots (DeLoach et al., 2003). The MaSE approach and models worked very well. The concurrent tasks mapped nicely to the typical behaviors in robot architectures. MaSE also provided the high-level, top-down approach missing in many cooperative robot applications.

Critique of MaSE Methodology:
1. Goal analysis, conducted at the beginning of a MaSE process, reinforces goal preservation through analysis and design phases. It facilitates role and Agent class modeling to focus on clear goal delegation, where every role is responsible for a particular goal to be accomplished. There are tasks that belong to the dedicated goals of roles.
2. In a role refinement step, it is crucial to match goals with roles. Every goal has to be associated with a role. With these roles defined, the design of communication between roles and their corresponding tasks become fixed, lacking dynamic adaptability of goals (and hence roles).

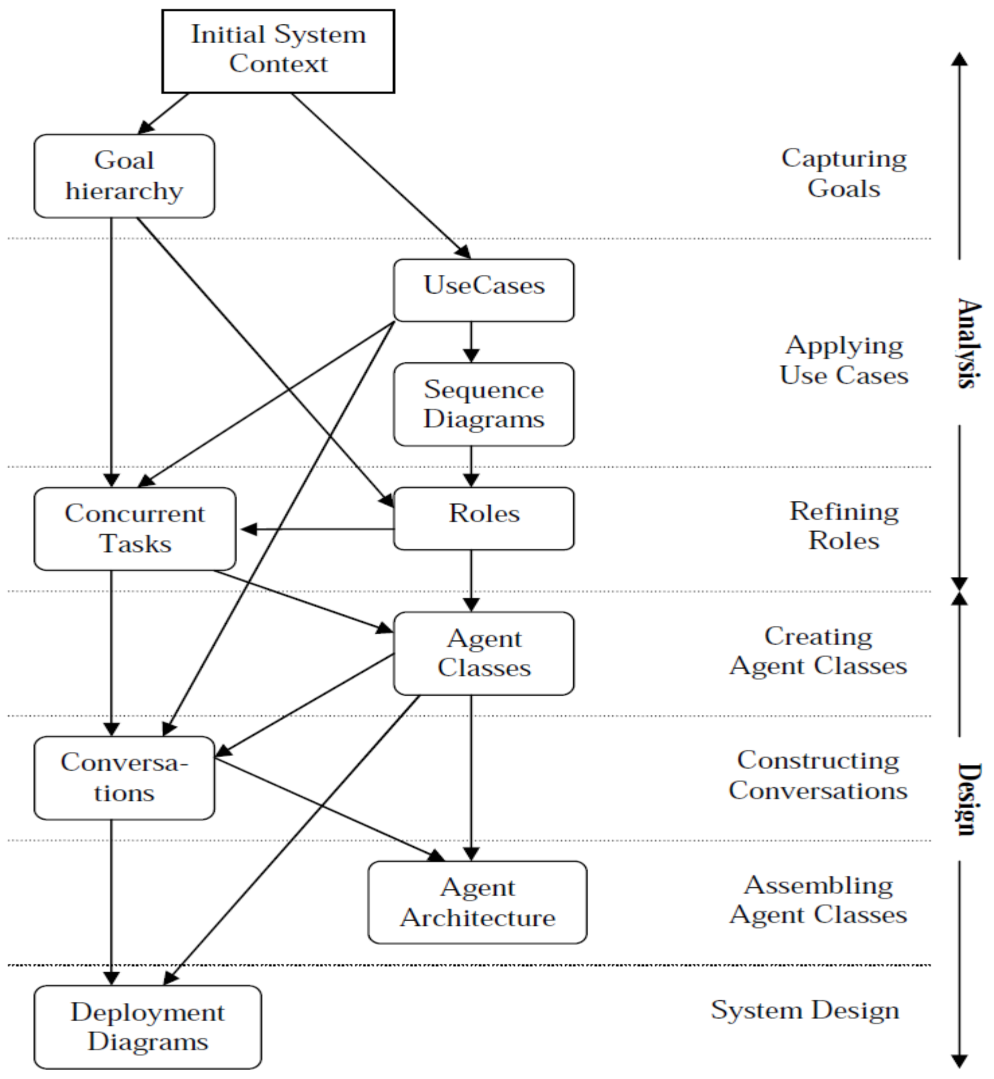

Figure 2.2 MaSE's process steps and artifacts (Source from DeLoach, 2012)

Software Engineering Methodology Evaluation

Several approaches have been applied to review and classify a large range of agent-oriented methodologies evaluation or to perform comparisons on a small number of methodologies [Chia-En Linetal.1996]. Unfortunately, such evaluations or comparisons are mostly subjective and are solely based on inputs from a single assessor (or group of assessors).

This research compares and evaluates two AO methodologies (GAIA, MaSE). This section briefly discusses various key methods, techniques and frameworks of the evaluated methodologies. Since object-oriented methodologies are considered as the "predecessor" of agent-oriented methodologies, we also look back at work on evaluating and comparing a number of object-oriented methodologies.

Methods for Evaluating Methodologies

We classify the major approaches to methodology valuation into four main groups: Feature-based evaluation, Quantitative evaluation, NIMSAD (**Normative Information Model-based System Analysis and Design**) framework and other evaluation approaches.

Feature-based evaluation

Feature-based evaluation (also often called Feature Analysis) is the most prominent and popular comparison approach (it has been chosen in this work). It is regarded as a qualitative method [Law, 2005]. It involves building an evaluation framework that can be represented in terms of a set of properties, qualities, attributes or characteristics [Law, 2005].

Quantitative evaluation approaches
Quantitative evaluations assess a methodology according to some measurable results produced by its use. These can be the software applications produced or the changes in the development process [Kruchten, 2000].

The NIMSAD (Normative Information Model-based System Analysis and Design) framework).
Being an evaluation framework, NIMSAD is not in fact a method for practically and efficiently comparing methodologies. It provides an alternative way of understanding and evaluating methodologies on the basis of the models and epistemology of systems thinking perspective [Jayaratna, 2000].

Comparisons of Object-Oriented Methodologies
Significant work in the area of OO methodology comparisons can be classified into three styles: comparison against a framework (feature analysis), comparison by meta-modeling and comparison by outcome (quantitative evaluation) [DeLoach,2001].

Evaluation Methods
Before proceeding with assessment, one needs to decide what evaluation methods should be used.

The Purpose of Evaluation
1. Getting a better understanding of the nature of AOSE methodologies, including their philosophies, objectives, features, etc.
2. Identify the strengths and weaknesses of the methodologies as well as their commonalities and differences in order to perform classifications and to improve future agent-oriented software system development.

Furthermore, it is emphasized that we are not trying to search for, in an isolation, for a best methodology. We believe that it is not always the case that all the AOSE methodologies are mutually exclusive. In fact, different methodologies may be appropriate to different situations, thus a methodology should be selected on the basis of considering different issues. These influencing factors can be the context of the problem being addressed, the domain, and the organization and its culture. However, we also expect that the evaluation would help in practical choices such as identifying the domains of applicability of each evaluated methodology.

The Evaluation Type and Procedure
There are several factors that may affect the decision of choosing an appropriate type of evaluation and procedure to carry out the evaluation. These are the available time, the level of confidence we need to obtain in the results of the evaluation, and the cost of the evaluation.

The Evaluation Framework
In this section, we describe a methodology evaluation framework within which the feature-based comparison is conducted. The framework consists of a set of criteria which addresses not only classical software engineering attributes but also properties which are uniquely found in AOSE.

In order to avoid using an inappropriate comparison framework, the properties in our framework were derived from a survey of work on comparing AOSE methodologies and on comparing OOSE methodologies. The evaluation framework covers four major aspects of each AOSE methodology: Concepts, Modeling language, Process, and Pragmatics. This framework is adapted from various frameworks [17] for comparing Object-Oriented methodologies.