IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-7, July 2018 E-ISSN: 2347-2693

Implementation of Validation of Requirements in Agent Development by Means of Ontology

G. K. Shankhdhar[1*], M. Darbari[2]

[1]Dep. of Computer Application, Babu Banarasi Das University, Lucknow, India
[2]Dep. of IT, Babu Banarasi Das National Institute of Technology and Management, Lucknow, India

[*]*Corresponding Author: g.kant.82@gmail.com , Tel.: +91-9792681898*

Available online at: www.ijcseonline.org

Accepted: 10/Jul/2018, Published: 31/Jul/2018

***Abstract*—** One significant limitation in Multi-Agent Systems development methodologies is lack of proper requirements validation. The authors have tried to implement requirement validation in Multi-Agent Systems using ontologies. Organizational Multi-Agent Systems Engineering is used as the agent development methodology. The aim of this paper is to include an appropriate method for validation of the requirements in Multi-Agent System development. In addition to working as a knowledge base, the authors in this paper have used ontologies to support requirements validation. Requirements validation is performed through rules that ascend from requirements and enforcement of these rules is done through a formal language, Semantic Web Rule Language. Genomic Information Retrieval is taken as case study. The Java Agent Development Environment (JADE) framework is used along with the Protégé 5.2.0 for ontology development. Apache Jena API, OWLAPI and SWRLAPI are used for implementation of the Multi-Agent System.

***Keywords*—** Multi- Agent System, Protégé, Jena, JADE, OWLAPI, SWRLAPI, agent communication, OWL, RDF

I. INTRODUCTION

The requirements must be put formally as properties in ontology. These formalized requirements called 'Rules' are executed by the reasoner to allow ontology to be debugged and unsatisfiable concepts and axioms are highlighted. The ontology is designed in Protégé 5.2.0. Jena API along with OWL API and SWRL API is used to interact with the ontology, named 'DNAONT'. The whole process is maneuvered programmatically in Java with Eclipse 4.5.2. SWRL, Semantic Web Rule Language is used for framing requirements in a formal specification, called Rules. These rules form the basis of ontology debugging by the reasoner.

The paper is organized as follows, Section I contains the introduction of requirements validation, Section II contains the related work concerned with development of MAS done recently, Section III contains the detailed MAS development process for GIRS, Genomic Information Retrieval System, Section IV contains the detailed discussion on requirement validation and section V brings the conclusion of research work with future directions.

II. RELATED WORK

A. Requirements Validation in MAS

During requirement specification process of software development activities, many existing systems or business process requirements are captured using natural language or specialized tools such as UML (Unified Modelling Language) or MAS (Multi-Agent System) Methodology specific tools like AT^3 (Agent Toolkit 3)[1]. However, the capturing of informal requirements into formalized properties using an ontology as a knowledge base has not been taken into attention by software developers due to time and budget constraints. It is critical for the informally captured requirements to be formally specified as Rules or Policies in order to perform requirement validation.
Reference [2] advocates unification of best of breed activities from existing MAS methodologies. It proposes an alternative approach that focusses on the use of domain knowledge through ontologies as offering the best potential for unifying access to them.

B O-MaSE and Recent Agent Development Methodologies:

O-MaSE (Organizational Multi-Agent Systems Engineering) is selected as MAS Methodology due to its 'organizational meta model' using method fragments and guidelines. These

three elements form the layers that can be used and developed independent of each other. The method fragments can be reused in a different scenario posed by an organization and in concomitance with the guidelines. The Requirement Analysis and Design phase are diagrammatically shown in figures 1 and 2.

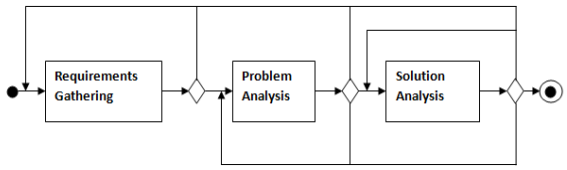

Figure 1. Requirement Analysis Phase in O-MaSE

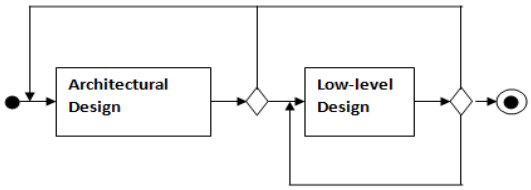

Figure 2. Design Phase in O-MaSE

C Multi-Agent Development Frameworks

Tools for Ontology development and integration with MAS are centered on APIs built for agent communication. Full-fledged architecture frameworks are distributed that had the capability for complete design of multi-agent systems. DECAF (Distributed, Environment Centered Agent Framework)[3] and JADE (Java Agent Development Environment)[4] are two of the best known architectures. These frameworks provide services for inter agent communication, planning, monitoring and co-ordination. The architecture simulates operating system type services with addition of learning and self diagnosis. The revelation of ad hoc versions of JADE through the LEAP project enables deploying of JADE agents seamlessly on various Java-based environments such as Android devices and J2ME-CLDC MIDP 1.0 devices and even partially connected NAT, IP address changes and firewalls. Other contributions in MAS architectures are TAEMS**,** RETSINA[5] and EMERALD[6].

D Ontological Support to MAS

The word ontology was taken from philosophy where it means "study of the nature of being". The most common definitions state that an ontology is a specification of a conceptualization [7] or that an ontology is the shared understanding of some domain of interest. Ontologies provide domain representation for multi-agent systems. It defines everything comprehensively in the domain. An ontology contains classification, properties, objects, literals and most importantly relationships between individual elements. Ontology provides the vocabulary for the messages passed between communicating agents. It specifies meaning to agent communication. This makes it easy to combine and add heterogeneous agents at runtime in order to function together even if they are unknown to their peers.

The ontological support in the multi agent system proffers reasoning. XML provides syntax. RDF(S), Resource Description Framework provides basic relational language and simple ontological primitives. OWL, web Ontology Language offers powerful decidability in an ontology language. But SWRL, Semantic Web Rule Language combining OWL and RuleML extends OWL. SQWRL (Semantic Query-Enhanced Web Rule Language) is used to query the Ontology. As SPARQL works over RDF, SQWRL works over OWL Ontologies.

In this implementation of MAS, Protégé 5.2.0 is used as an ontology design toolkit.

E DNA Sequencing

As a case study for implementing inter-agent communication, DNA pattern search in existing varied and heterogeneous Genome Repositories is chosen.

From the viewpoint of a computer science researcher the important concerns regarding sequencing a DNA are:

The Genome contains the DNA and the whole genetic structure. This genetic structure keeps the complete information necessary for an organism to live its life. This genetic material is similar in many organisms. Biologists and Life Science's experts sequence DNA in the form of sequences of four characters, A, C, T and G. This is done in order to represent a DNA programmatically. A DNA structure is made up of the combinations of these four elements:

1. Thymine (T)
2. Cytosine (C)
3. Guanine (G)
4. Adenine (A)

Since the DNA of an organism is similar to other organisms, conditions arise when Biologists look for similarity in DNAs like in areas like Pharmacy. There are requirements when a particular extract of a DNA has to be searched in disparate and heterogeneous data sources ranging from plain text files to plethora of databases acting as repositories of fully sequenced DNAs. The full discussion on DNA Sequencing is out of the scope of this paper. As a researcher, my quest deals only with inter-agent communication focusing validated policies and conflict detection and resolution in agents.

III. MULTI-AGENT SYSTEM DEVELOPMENT THROUGH O-MASE

A Requirements Gathering and Problem Analysis

The various phases of MAS development are necessary to understand in order to know the various artifacts produced during these phases. These artifacts are used to build ontology. The GIRS analysis diagrams like Goal diagram, Role Diagram, Agent diagram, Interaction diagram showing message exchange between agents and the State or Plan diagram are comprehensively discussed in [7]. Goal diagram being the most elementary is shown below. In the first phase of agent development the requirements of the end user are thoroughly investigated and studied. The requirements are the basis on which the organization policies are framed. They provide the exaction of operation by the agents. These policies have to be compulsorily adhered by the agents. In later sections we will see how these policies are validated through SWRL rules and ontological reasoning. This phase involves three basic activities: Model Goals, Refine Goals and Model Domain. The requirements are first compartmentalized into goals that the system will achieve with the most general goal at the top and successive sub goals down the goal hierarchy. This process is completed with a goal model for dynamic systems (GMoDS). Modeling a domain captures the object types, relationships and agent behaviors in the environment the agents will perceive and act. The GMoDS for the GIRS is shown in figure 3. Agent Tool3 (AT^3) distributed as a Java plug in is used for the construction of all work products created in O-MaSE.

The GIRS requirements can be quantized and enumerated as:

R1. Accept a DNA Pattern of a fixed character maximum length (usually 1024) that has to be searched over the internet.
R2. The DNA Pattern that has to be searched must only contain ‘A’, ’C’, ‘T’ and ‘G’ as character set.
R3. The break-up size of a DNA Pattern should be 20 characters.
R4. No special characters and white spaces are allowed.
R5. The searched results along with the annotation and web links to resources should be returned to the user.
R6 Total time from query submission to result display can be maximum 3.2 seconds

These requirements are transformed into GMoDS as depicted in the figure below.

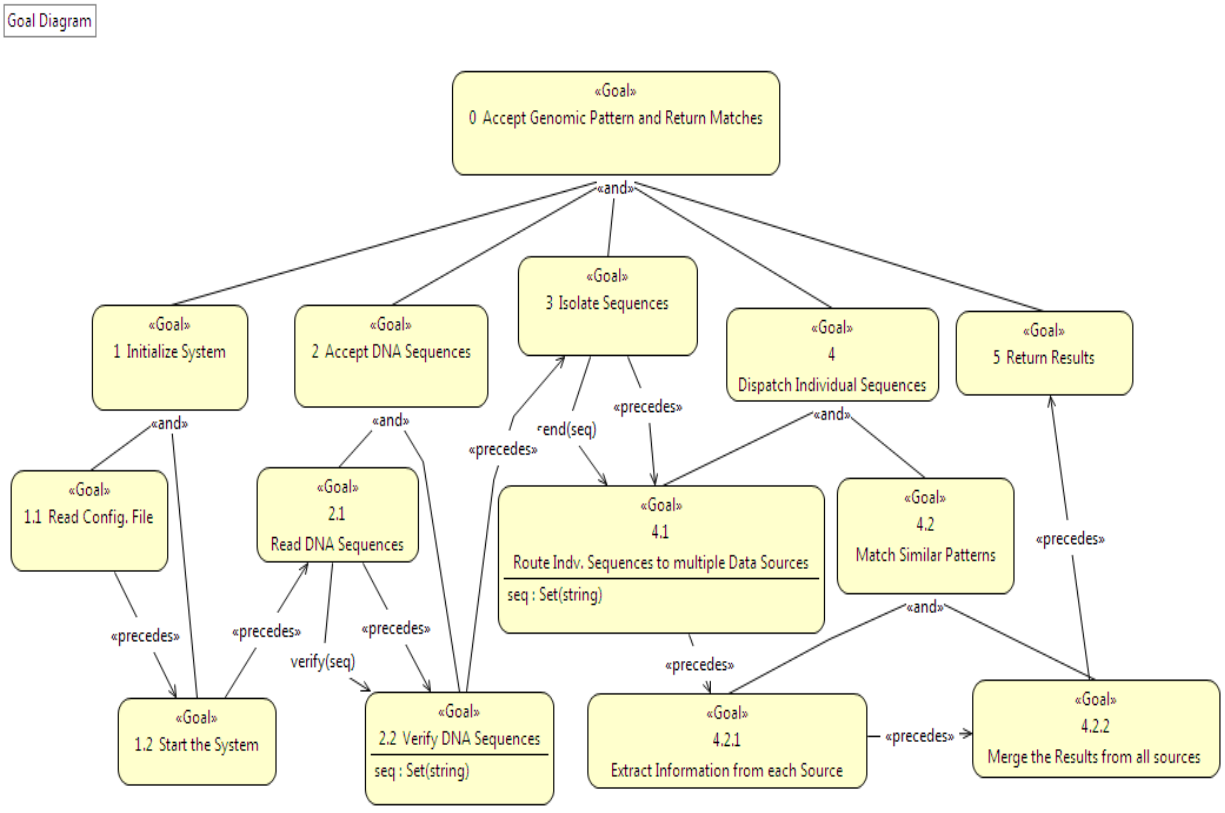

Figure 3. Goal Model for GIRS in O-MaSE

B Solution Analysis

This phase comprises of Modeling Roles and Defining Roles. In this, Roles on the basis of the Goals are identified. The GMoDS acts as input to this phase, resulting into well defined Roles.

C Architecture Design

Architecture Design deals with modeling agent classes, protocols and policies. Agent classes are obtained by Role Model. Roles, if necessary are combined for similar functionality to Agent capabilities. The agents in GIRS are derived from the role model to produce Agent Diagram. The sequence of inter agent communication through messages results in the Protocol Model[7]. Administrator launches the Initiator Agent that starts the UI Agent and the Wrapper Agent by reading the startup configuration file. The Facilitator Agent and the Router Agent as part of the FIPA (Foundation for Intelligent Physical Agents) are handled by the JADE Architecture for orchestrating the agent communication process. The user wanting to search a particular DNA pattern over the internet has to supply the search pattern to the UI Agent. The UI Agent with the help of DROOLS reasoner through JAVA's Jena Package checks and validates all the policies derived from the requirements, discussed shortly.

The search pattern received by the UI Agent is in the form of 'A', 'C', 'T' and 'G' characters and has maximum size of 1024 characters e.g., "ACTTTTGTGTCAAAC". The UI Agent forwards the search pattern to the Wrapper Agent. The Wrapper Agent checks if a similar search pattern has recently been searched, otherwise utilizing the Mapping Information routes the search pattern to different subscribed data sources[8]. The Source Agents are responsible for maintaining XML files of their respective databases. The XML files hold the annotation details of the organisms like origin, classification, version and description. When a match occurs, the annotation details are returned to the UI Agent and the Mapping Information is updated. GIRS Architecture is shown in figure 4.

As part of the Architecture Design, policies stating the rules that the agents must adhere to, are also derived. Some policies or rules through requirements stated in GIRS can be stated as:

P1: ONLY A, C, T, or G characters can be used to represent a DNA Pattern.

P2: The search string cannot contain white spaces, hyphens or any special character or numbers.

P3: Max Length of a search pattern can be set but assumed to be 1024 chars.

P4: Break-up size of search pattern can be set but assumed to be 20 chars.

P5: Patterns not conforming to P1, P2, P3 and P4 will be immediately discarded.

P6: Total time from query submission to result display can be maximum 3.2 seconds.

P7: Queries failing to meet P6 will be held for resubmission.

D. Low Level Design

Low level design deals with the states through which the agent undergoes. This is pictorially represented by Finite State Machine (FSM) also called a Plan Model [9]. This automaton logically proofs the existence and functionality of the agents.

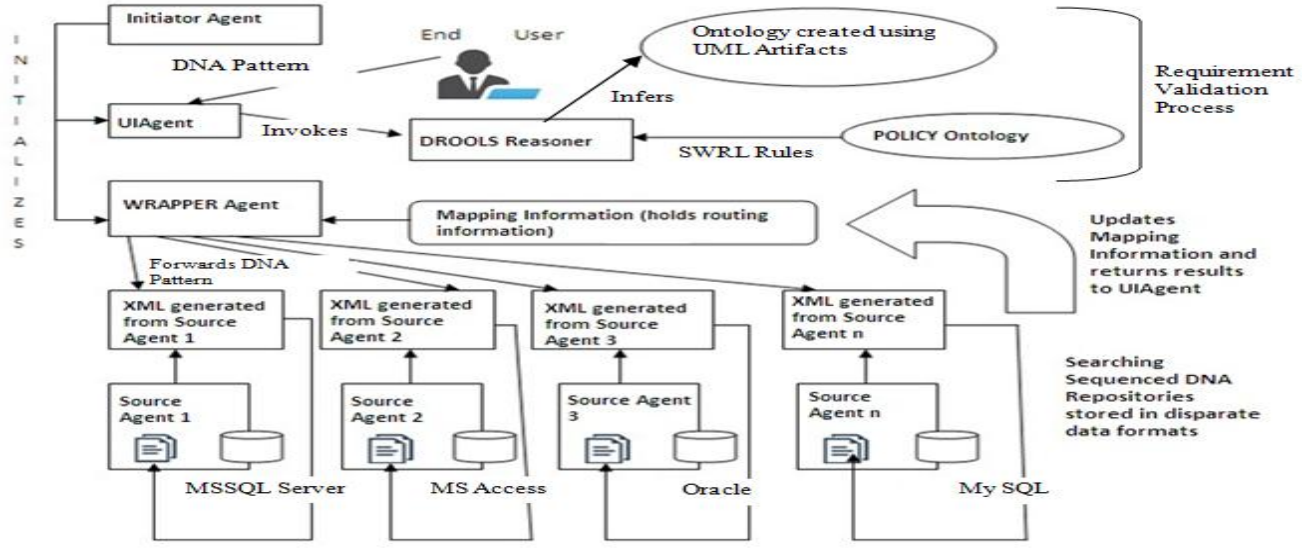

Figure 4. The GIRS architecture with Requirement Validation

IV. VALIDATION OF REQUIREMENTS

The implementation of the GIRS is developed in Java with Eclipse 4.5.2 incorporating Jena API, JADE, OWLAPI and SWRLAPI. Specifically OWLAPI and SWRLAPI are used to define rules and reason the ontology named DNAONT[10] shown in figure 5 which along with the knowledge base holds the constraints in the form of rules that are associated with agents in order to work within the domain of the organizations and abide to the policies framed by the organization. At the beginning of the MAS development process, the DNAONT ontology has a basic set of Classes, Properties, Individuals and SWRL Rules [11, 12].

As the architectural and low level design of a MAS progresses, UML/AT^3 diagrams are created carrying formal information of the fulfilled requirements by the system in design. These requirements need to be validated [13].

Now an ontology will be generated based upon the diagrams using UML to OWL generator. Since the basic DNAONT and the ontology generated by UML/AT^3 toolkit have the same structure and knowledge-base architecture, they are merged.

After merging the two ontologies, the DROOLS reasoner infers the Axioms in Protege which will then list all possible errors and we can use this information to make corrections to the model [14]. This is done programmatically using SWRLAPI with minimal user intervention.

The policies P1 and P2, discussed here, are specified and enforced by the formal language, SWRL. The rule can be stated as:

"PatternContent(?p) ^ has_dna_pattern(?p, ?pt) ^ swrlb:matches(?pt, "[ACTG]") ^ Length(?pt) ^ swrlb:lessThan(?pt, 20) -> ver_pat(?p, ?pt) ^ VerifiedDNAPattern(?p)"*

After execution of the rules, the verified DNA patterns become instances of the class *VerifiedDNAPattern* with the property *ver_pat* set. This scenario is shown in tables 1 and 2. Initially all the four individuals belong to the class PatternContent and have has_dna_pattern property set. Individual named 'Platypus' and 'Red_Ant' do not qualify for validation because the former is not 20 characters in length and the later contains 'S' character which does not occur in the set {'A','C','T','G'} [15, 16]. But when the SWRL rule in our example is executed by the Drools Rule Engine, the patterns conforming to the rule then become objects of VerifiedDNAPattern class and also have their ver_pat property set. An extract of the DNAONT ontology representing the above scenario is shown in Figure 5.

The code for the inference by the Rule Engine is given below:

```
OWLOntologyManager ontologyManager =
OWLManager.createOWLOntologyManager();
        org.semanticweb.owlapi.model.OWLOntology ontology
= ontologyManager.loadOntologyFromOntologyDocument (new
File("D:/ONT/DNA_ONT/DNAONT.owl"));

// Create a SWRL rule engine using the SWRLAPI

org.swrlapi.core.SWRLRuleEngine swrlRuleEngine =
SWRLAPIFactory.createSWRLRuleEngine(ontology);

Set<SWRLAPIRule> sets = swrlRuleEngine.getSWRLRules();
for(SWRLAPIRule item : sets){
            System.out.println(item.toString());
          }
swrlRuleEngine.infer();
swrlRuleEngine.exportInferredOWLAxioms();
ontology.saveOntology();
System.out.println("DNAONT Ontology Saved To Disk after
successful DROOL Reasoner's Inference. Now all Rules are
validated and only the validated rules have their ver_pat property
set and they become the member of VerifiedDNAPattern Class");
```

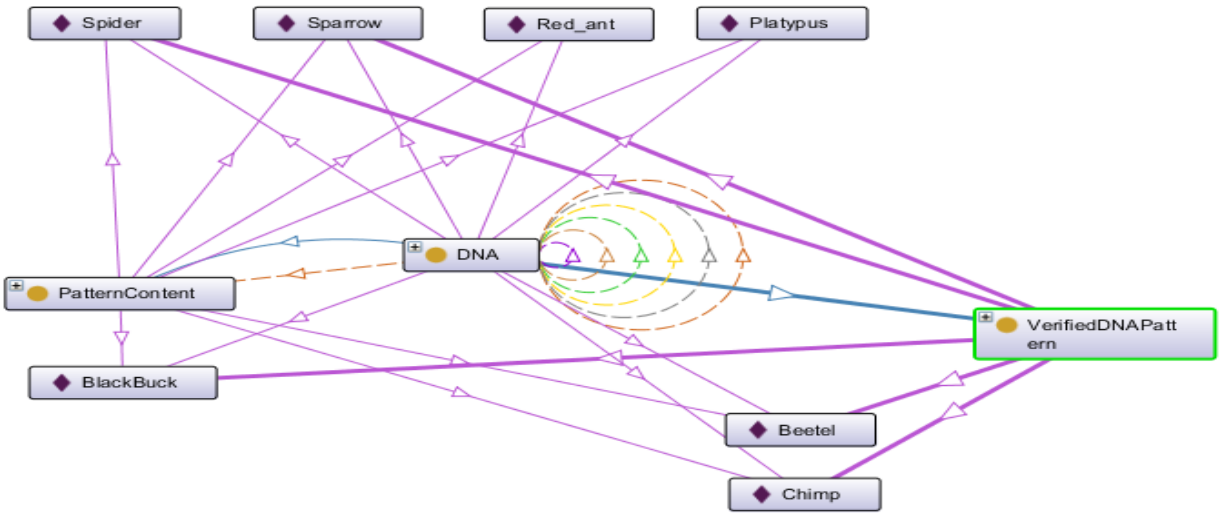

Figure 5. DNAONT Ontology built in Protégé 5.2.0

Table 1. Axioms before Rule Engine Inference

Class	Property	Individual	Value
PatternContent	has_dna_pattern	Chimp	CGCGTACTGTACTGCCGAAT
PatternContent	has_dna_pattern	Beetel	AACCTTGGGGAACCTTCCTT
PatternContent	has_dna_pattern	Platypus	CGCGTACTGTACTGCCGAA
PatternContent	has_dna_pattern	Red_ant	CGCGTACTGTACTGCCGASA

Table 2. Axioms after Rule Engine Inference with validated Individuals having ver_pat property and VerifiedDNAPattern Class set

Class	Property	Individual	Value
PatternContent, *VerifiedDNAPattern*	has_dna_pattern, *ver_pat*	Chimp	CGCGTACTGTACTGCCGAAT
PatternContent, *VerifiedDNAPattern*	has_dna_pattern, *ver_pat*	Beetel	AACCTTGGGGAACCTTCCTT
PatternContent	has_dna_pattern	Platypus	CGCGTACTGTACTGCCGAA
PatternContent	has_dna_pattern	Red_ant	CGCGTACTGTACTGCCGASA

V. CONCLUSION AND FUTURE SCOPE

The major limitation of current Multi-Agent System Development methodologies is centered on improper validation of requirements. The authors have proposed an implementation that leverages the strength of Ontology designed for this purpose in Protégé 5.2.0. In order to validate the requirements, they are framed as formal properties that can be evaluated against a formal language [17]. In this paper, SWRL is used as a formal language. The framed requirements are called rules or policies. These policies, designed in SWRL are executed on the DROOLS reasoner and rule engine to act in accord. The agent development is done in JADE and the semantic programming is dealt with Apache Jena Package together with OWLAPI and SWRLAPI. The Java IDE used is Eclipse 4.5.2. As future work, context-sensitive policies can be implemented in MAS in scenarios where the policies change with time, particularly after MAS implementation.

REFERENCES

[1] Scott Deloach, O-MaSE: A customisable approach to designing and building complex, adaptive multi-agent systems, https://www.researchgate.net, 2010.

[2] Beydoun, G. & Low, G. Complex Intell. Syst. (2016) 2: 235. https://doi.org/10.1007/s40747-016-0025-5

[3] Kieth Decker, DECAF, A Multi-Agent System for Automated Genomic Annotation, Kluwer Academic Publishers, 7,7-27, 2003.

[4] Fabio, Bellifemine, Giovanni Caire, Dominic Greenwood, Developing Multi Agent Systems with JADE, Wiley, 2007.

[5] Katia P. Sycara, Martin van Velsen, Massimo Paolucci, Joseph Andrew Giampapa, The RETSINA MAS infrastructure, Research Gate, July 2003.

[6] Kalliopi Kravari, Efstratios Kontopoulos and Nick Bassiliades, EMERALD: A Multi-Agent System for Knowledge-based Reasoning Interoperability in the Semantic Web, Research Gate, May 4-7, 2010.

[7] Gaurav Kant Shankhdhar, Manuj Darbari, Building Custom, Adaptive and Heterogeneous Multi-Agent Systems for Semantic Information Retrieval Using Organizational-Multi-Agent Systems Engineering, O-MaSE, IEEE Explore, ISBN: 978-1-5090-3480-2, 2016.

[8] Atul Verma , Narendra Jha, Verified Message Exchange in Providing Security for Cloud Computing in Heterogeneous and Dynamic Environment, International Journal of Applied Information Systems, 2017.

[9] Gaurav Kant, Manuj Darbari, Change Management in Semantic Web Services in Legal Domain using FSM & XXM Publication: IJAIS volume9/number1/751-1359, 2015.

[10] Sumit Kumar Mishra, V.K. Singh, Ontology Development For Wheat Information System Description: Publication: IJRET-International Journal of Research in Engineering and Technology 2015, V0l4/I05.

[11] Gaurav Kant, Manuj Darbari, Introducing Two Level Verification Model for Reduction of Uncertainty of Message Exchange in Inter Agent Communication in Organizational-Multi-Agent Systems Engineering, O-MaSE, IOSR Journal of Computer Engineering (IOSR-JCE), http://www.iosrjournals.org/iosr-jce/pages/19(4)Version-2.html, 2017

[12] Nasserine Hamrouni, Verification and validation for MAS APN, 6th International Conference on Sciences of Electronics Technologies of Information and Telecommunications (SETIT), 2012.

[13] Gaurav Kant Shankhdhar and M Darbari. Article: Legal Semantic Web- A Recommendation System. International Journal of Applied Information Systems 7(3):21-27, May 2014. Published by Foundation of Computer Science, New York, USA.

[14] Regulated Open Multi-Agent Systems (ROMAS), A Multi-Agent Approach for Designing Normative Open Systems, Springer, 2015.

[15] Jogannagari M.R., Kothari P.R, "The complexity of Validation Testing in Component Based Software Engineering", International Journal of Computer Science and Engineering, IJCSE,Volume 5, Issue 12, 2017.

[16] Yagyasen, Diwakar, and Manuj Darbari. (2014) "Application of Semantic Web and Petri Calculus in Changing Business Scenario." Modern Trends and Techniques in Computer Science. Springer International Publishing, 2014. 517-528.

[17] K. Laxmi Pradeep, K. Madhavi, "Approaches for Efficient Learning Software Models: A Survey", International Journal of Computer Sciences and Engineering, Vol.6, Issue.1, pp.108-113, 2018.

Authors Profile

Mr. G.K. Shankhdhar, having more than 7 years of experience in teaching software engineering subjects. He is the author of numerous research papers in MAS and IOT. Currently pursuing PhD from Babu Banarasi Das University is working on the post of Assistant Professor. Has industry experience in software development and training with Microsoft Certifications in Dot Net.

Dr. Manuj darbari, an Experienced Associate Professor with a demonstrated history of working in the telecommunications industry.

Skilled in Mathematical Modeling, Analytical Skills, Computer Science, Research Design, and Entrepreneurship. Have been awarded a PhD focused in Business Administration from University of Lucknow. Associate Professor at Babu Banarasi Das National Institute of Technology. He is a member of IEEE and author of many valuable research papers in the fields of Business Analytics, MAS, Cloud and many more.

