IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 4, Ver. II (Jul.-Aug. 2017), PP 08-18
www.iosrjournals.org

Introducing Two Level Verification Model for Reduction of Uncertainty of Message Exchange in Inter Agent Communication in Organizational-Multi-Agent Systems Engineering, O-MaSE

*Gaurav Kant Shankhdhar[1], Manuj Darbari[2]
[1] Department of Computer Applications Babu Banarasi Das University, Lucknow, Uttar Pradesh, India
[2] Department of IT, Babu Banarasi Das National Institute of Technology and Management, Lucknow, India, Member, IEEE
Corresponding Author: Gaurav Kant Shankhdhar

Abstract: *The last decade has seen tremendous advancements in the field of Multi-Agent Systems. Inter-agent communication is the most integral part of any MAS. The problem of impreciseness and obscurity in agent communication has been swelled with increasing complexity of MAS and deepened with the rise of heterogeneity and diversity in varied platforms of data storage. Another technology with parallel growth is the semantic theory, significantly contributing in providing meaning to expressions thereby reducing ambiguity. The authors here provide two layers of verification of messages. At first, a vocabulary of terms used in messages is constructed with the help of an ontology in OWL to diminish vagueness. Then at a higher level business organization rules contribute to another ontology defining the policies that orchestrate the MAS establishing the domain of organization. The O-MaSE is used as MAS design methodology. In this paper, extension to O-MaSE is provided through dual verification system.*

Keywords: *KQML, SPARQL, agent communication, OWL / RDF*

Date of Submission: 28-06-2017 Date of acceptance: 15-07-2017

I. Introduction

In order to demonstrate the dual verification of messages in Agent Communication[4] and providing as an extension to the O-MaSE Methodology[1][2] for MAS development, authors have used a model, namely, Genomic Information Retrieval System (GIRS)[1]. This model envisioned by the authors is used to illustrate minimizing vagueness and incompleteness in inter-agent communication and is in its development stage. During the development process of GIRS, some principal diagrams were constructed using the O-MaSE AT3 tool like Goal Model, Role Model, Agent Model, Protocol Model and Plan Model[2][3]. Details behind these diagrams can be found in [1].The proposed GIRS Multi-Agent System Model is shown in figure 1. The ambiguity in communicating agents[3][4] leads to uncertainty in results. The primary aim of this paper is to reduce the contravening consequences of obscurity and vagueness by using semantic[3] representation of the knowledge used by the agents. The vocabulary of the messages is a list of terms and needs to be organized with relationships, axioms and reasoning. This is carried out by use of Common Ontology, here, DNA_ONT. So, in the process of agent communication unknown or ambiguous terms that can lead agent to dilemma are comprehensively defined and the message is absolute and unmitigated. This comprises the first part of the verification process.

Policies in O-MaSE comprises of all the formally specified rules which describe the behavior of any organization in some situation. Policies lay down what the agents are supposed to do and from what to refrain. The agents work in accordance with the specifications of the policies. The actions by agents that work on sensitive data or resources have first to be located, substantiated and checked through another ontology, here, O-MaSE_POLICY_ONT, then the task is achieved. The second ontology used here is the policy ontology containing all the policies of the organization as well as their relationship with agents. This forms the second level of verification.

1.1 Genomic Information Retrieval System (GIRS)[1]: The authors have proposed a model to build custom, adaptive and heterogeneous Multi-Agent Systems for semantic information retrieval using Organizational-Multi-Agent Systems Engineering, O-MaSE. Genomic or DNA pattern searching is taken to be as a case Study.

1.2 Analysis

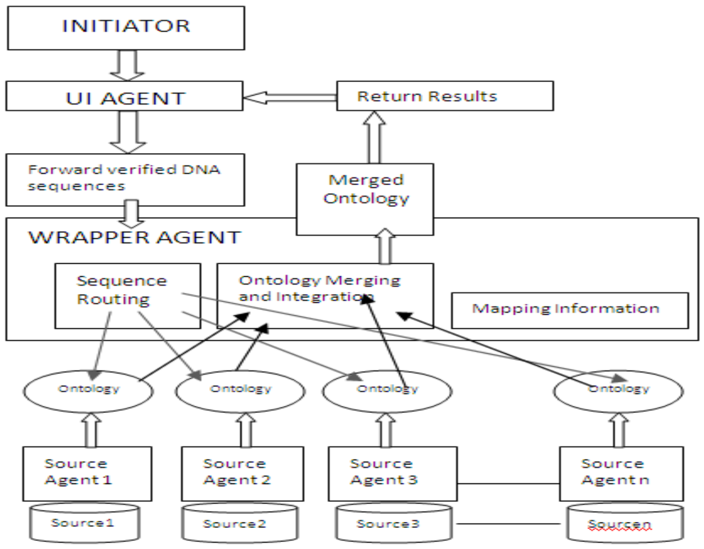

Fig 1. GIRS Multi Agent System Model

In fig1 is shown the GIRS MAS where the Initialization of the system is carried out by the Initiator Agent that further gives control to the UIAgent. Valid DNA sequences to be searched are forwarded to the Wrapper Agent that with coordination with the Facilitator Agent routes the DNA Search Pattern queries to successive Source Agents.

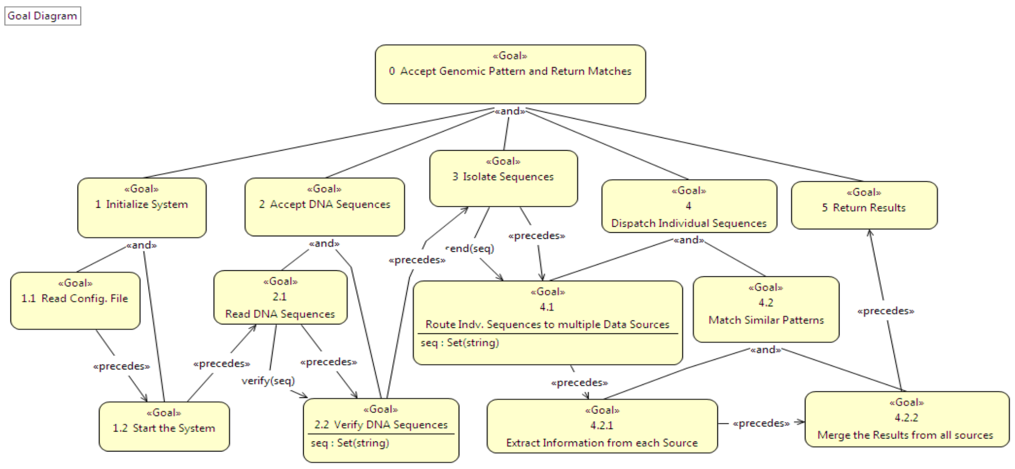
Goal Diagram
«Goal» 0 Accept Genomic Pattern and Return Matches
«and»
«Goal» 1 Initialize System
«Goal» 2 Accept DNA Sequences
«Goal» 3 Isolate Sequences
«Goal» 4 Dispatch Individual Sequences
«Goal» 5 Return Results
«and»
«and»
«and»
«Goal» 1.1 Read Config. File
«Goal» 2.1 Read DNA Sequences
«Goal» 4.1 Route Indv. Sequences to multiple Data Sources seq : Set(string)
«Goal» 4.2 Match Similar Patterns
send(seq)
verify(seq)
«precedes»
«Goal» 1.2 Start the System
«Goal» 2.2 Verify DNA Sequences seq : Set(string)
«Goal» 4.2.1 Extract Information from each Source
«Goal» 4.2.2 Merge the Results from all sources

Fig 2. GIRS GOAL Diagram

Fig. 2 shows the various goals that are to be achieved through the GIRS Model.

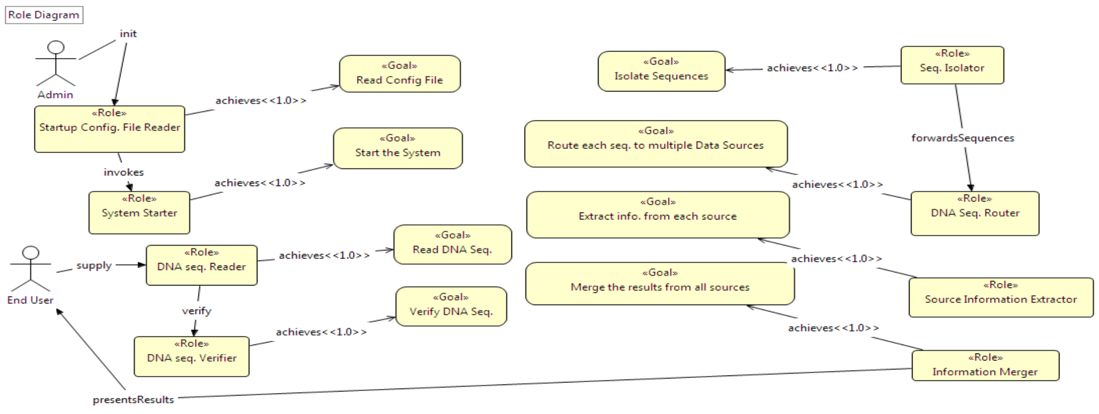

Fig 3. GIRS ROLE Diagram

After the Goals are analyzed, the next step is to derive the Roles which are graphically depicted by the Role Diagram.

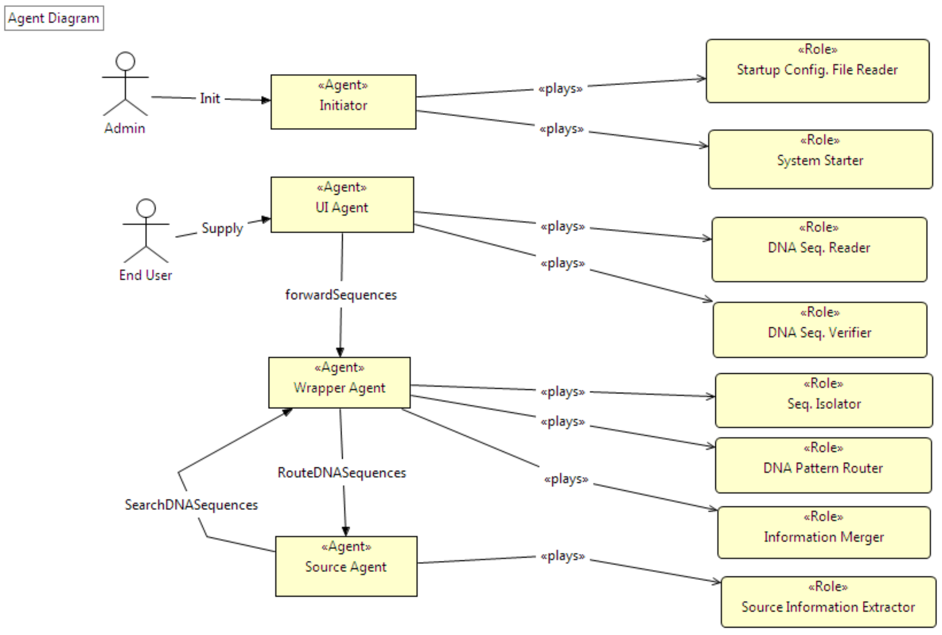

Fig 4. GIRS AGENT Diagram

The Roles that are derived give rise to the identification of Agents. Agents are formed by grouping Roles on the basis of related goals.
The Agents used in GIRS as derived from the Goals are as below:

- Initiator: used to initialize the system by reading a startup configuration file.
- UIAgent to read and display user queries and results.
- Wrapper: with the help of another agent called the Facilitator Agent routes and selects the most appropriate Source Agent to direct the DNA Search Pattern for querying over the source ontology.
- Source Agent: converts the underlying heterogeneous data-store to source ontology.

Another agent, called the Facilitator Agent[5] is also included and is discussed shortly.

II. Literature Survey

2.1 DNA Sequencing

From the perspective of a computer science researcher the concepts needed to be known can be stated as:DNA and the whole genetic structure are contained inside a Genome. For an organism to live its life this genetic structure holds the complete information for the organism's quest for being. This genetic material is monolithically similar from organism to organism. Biologists and Life Science's experts sequence DNA in the form of sequences of four characters, A, C, T and G. Actually, the DNA sequencing is carried out by the fact that DNA is composed of 4 different elements:
1. Thymine (T)
2. Cytosine (C)
3. Guanine (G)
4. Adenine (A)

A DNA Sequence comprises of the combinations of these four substances. Since the DNA of an organism is quite common to other organisms, situations arise when Biologists look for similarity in DNAs like in Pharmacy or Drugs development, then it becomes evident to look for similar DNA Patterns. The full discussion on DNA Sequencing is out of the scope of this paper. As a researcher, I have taken this as case study to demonstrate the process of verifying messages exchanged between software agents collaborating for extracting meaningful information from disparate environment.

2.2 O-MaSE:

The choice of O-MaSE over other MAS Methodologies is comprehensively discussed in [1]. The limitations of Prometheus, Tropos, GAIA and MaSE, leads to the selection of O-MaSE. As an addition to this methodology, the authors have added dual verification of messages of agents for reducing obscurity among agents. The composition of O-MaSE has three preliminary elements, namely a meta model, method fragments and few guidelines. the meta model defines the basic concepts required for the design and implementation of MAS. The method fragments are the processes that are carried out to produce work products like models and code. And the interrelatedness of the method fragments are defined by the guidelines. The O-MaSE Requirements Analysis phase is shown as under:

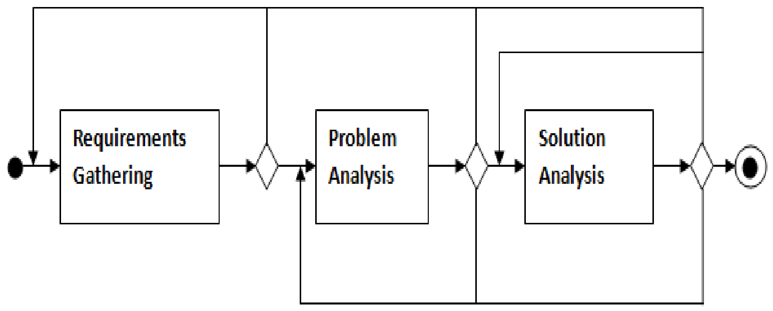

Fig 5. Activity Flow in Requirement Analysis Phase

The current comparison of the various parameters of the work product of the MAS Methodologies is shown in table. The *bold and italicized parameters* show few of the limitations of the present O-MaSE methodology. The authors have tried to remove these shortcomings by providing a Dual Verification Model through Common Ontology and Policy Ontology.

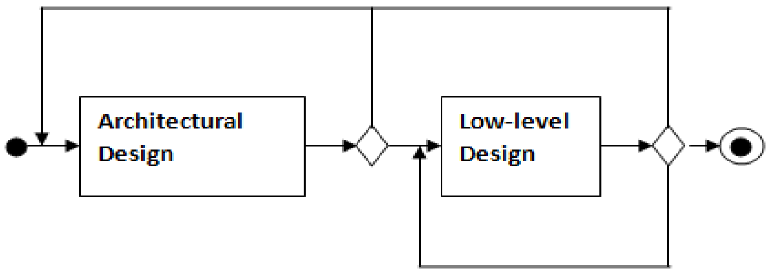

Fig 6. Activity flow in Design Phase

Validation of Requirements can be understood by enforcement of Policies. Policies, in fact, are derived through requirements. By providing an ontology for the policies, we not only make sure that all agents share the same copy of the policies, but also afford reasoning such that the 'actions' of the agents are made within policy rules defined by the organization. The table below[6] represents a comparison of O-MaSE with other contemporary multi-agent system development methodologies.

Table 1. Comparison of few important MAS Methodologies

Parameter	**O-MASE**	**OPERA**	**TROPOS**	**GORMAS**	**ROMAS**
Modeling Tool	Provided	Provided	Partially Provided	Provided	Provided
Code Generation	Partially Provided	Partially Provided	Not Provided	Partially Provided	Partially Provided
Validation of the Requirements	***Not Supported***	Not Supported	Partially Supported	Not Supported	Not Supported
Verification of Inconsistencies	***Not Supported***	Not Supported	Not Supported	Not Supported	Not Supported
Tests	Not Supported	Not Supported	Not Supported	Not Supported	Not Supported
Coherence of the Normative Context	Partial verification in the case tool	Partial verification in the case tool	Not Supported	Not Supported	Not Supported
Traceability of the Normative Context	Not Supported	Not Supported	Not Supported	Not Supported	Supported

III. Inter-Agent Communication And Kqml

The interaction diagram shown below in figure 5 depicts the communication between agents based on the sequence of messages exchanged between them.

There are many ways how the agents can communicate including RPC, RMI, Message Sharing, Pipes but the hierarchical communication demands the use of Knowledge Query and Manipulation Language, KQML[5]. A problem with KQML is lack of definite semantics. FIPA[7] developed Agent Communication Language, ACL, holds better in semantic support but is hard to use and understand. The KQML language is divided into three layers:

- Content Layer
- Message Layer
- Communication Layer

The content layer holds the actual content of the message in application's representation language. Message Layer is the core of KQML and tells how the agents communicate and send speech act or performatives. The Communication Layer deals with lower level communication parameters like identity of the sender agent and the receiver.

3.1 Communication Between Agents In GIRS

In figure 5 is shown the sequence of messages exchanged between agents. The Initiator starts the UIAgents. The DNA search pattern comprising of A, C, T and G, like "ATTGCTGCCT" is supplied by the end user to the UIAgent. The UIAgent *forwards* this pattern to the Wrapper Agent. The Wrapper recommends it to the Facilitator Agent. The Source Agents *advertize* to the Facilitator Agent. Then the Facilitator Agent *Asks* the Source Agents for DNA Pattern Matches. The Source Agent if successful in finding similar patterns *tells* the Facilitator Agent about its findings. Then Facilitator *tells* the successful pattern matched to the Wrapper and then the Wrapper to the UIAgent.

Here, the terms italicized are actually KQML Performatives that are special keywords of the KQML Language. The semantic deficiency of KQML can be dealt with providing a common ontology for message vocabulary and using a query language that can inspect, modify and extract the ontology itself. A query language that can reason on the shared ontology and draw inferences is needed.

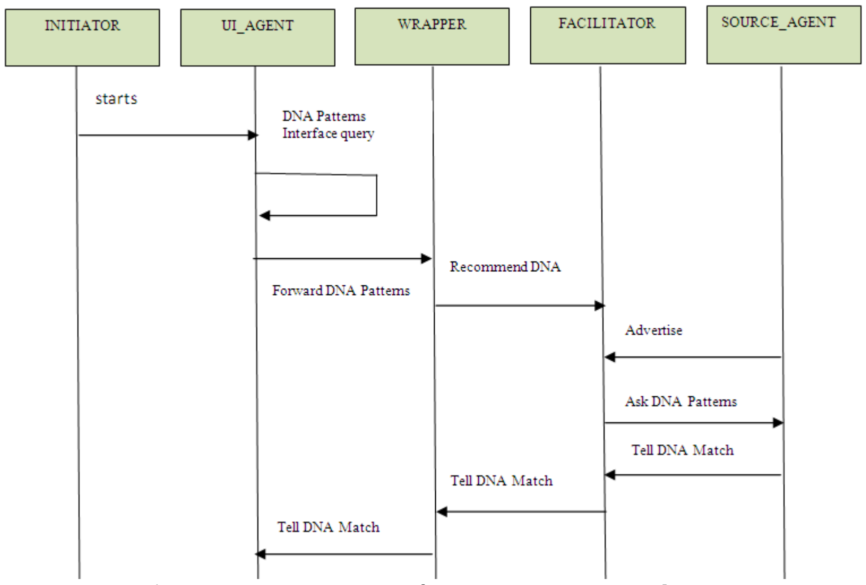

Fig 7. Sequence Diagram for GIRS Message Exchange

IV. KQML With SPARQL

KQML was developed by a group of representatives from varied projects. They were concerned with managing distributed applications. These projects included expert systems and systems for CAD/CAM.

KQML consists of two dedicated programs, a router and a facilitator and an interface routine library, called a KRIL.

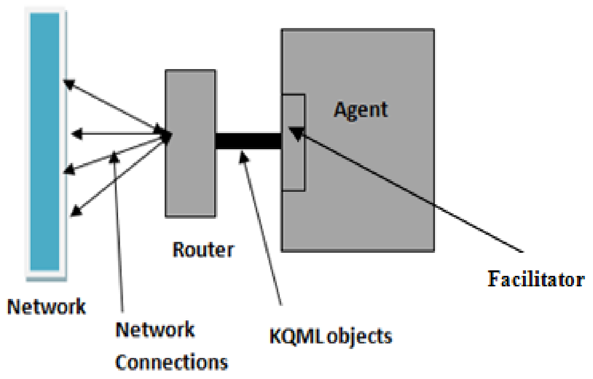

Fig. 8 - Router provides single interface to the network. It acts both as a client and a server and handles multiple simultaneous connections.

4.1 KQML Routers.

Routers have nothing to deal with the content of the message. Each KQML bound software agent is related with its own copy of router process. The process of each router is identical. A router is responsible for all the KQML messages passing from its related agent. Router relies extensively on KQML performatives. The router directs the outgoing message with a particular internet address and even locates the service if requested.

4.2 KQML Facilitators.

Facilitators deal with the messages that are incompletely addressed. Among the many services that the facilitator provides one is to maintain a registry that contains the service names. Facilitators help routers in finding hosts to route messages. Every local group of agents has one facilitator

4.3 KQML KRILs.

KRIL, KQML Router Interface Library, provides a programming interface between the application and the router. Since the Router has no knowledge about the content field of the message, KRIL uses application tools to extract the message content.

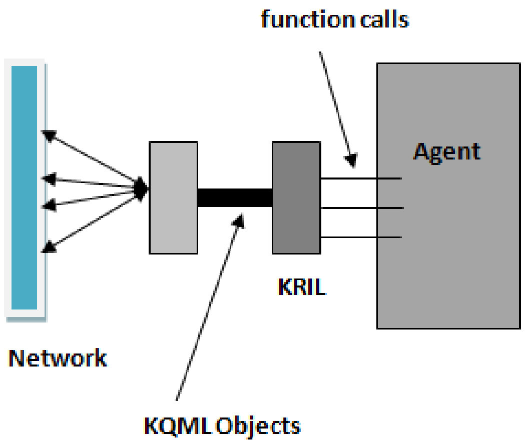

Fig. 9 - KRIL providing programming interface between the application and the router.

The Semantic Web needs a semantic query language. In this work the authors provide Ontological verification of the messages exchanged between agents, held by DNA_ONT ontology and another Global Ontology, OMaSE_POLICY_ONT for enforcement of policies. The traditional query languages used with KQML cannot suffice for the semantic web. Other promising query languages like Prolog and SPARQL[8] are made for the purpose of Semantic inter agent communication where the messages between agents have a meaning understood by the machines. The benefits of the Semantic Web are outside the scope of this work. SPARQL (pronounced "sparkle", a recursive acronym for SPARQL Protocol and RDF Query Language) is an RDF query language, that is, a semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework (RDF) format. In this paper, KQML is used with SPARQL to provide semantic message exchange between agents. SPARQL and Prolog queries can access and modify RDF data sources encompassed with reasoning. It is through SPARQL that the messages exchanged through KQML can be verified against the Common Ontology. A subset of KQML messages passed between agents which are also shown in the interaction diagram in figure 7. These messages form a part of the first level of verification. In order to clarify the messages it is pertinent to mention over here the syntax of KQML messages. The KQML messages begin with a ‘Performative’ that actually conveys the intent of the dialog. This forms a part of the “speech act theory”. Common performatives are: tell, ask-one, recommend-one, ask-all, broadcast, tell, advertize and even additions are allowed.

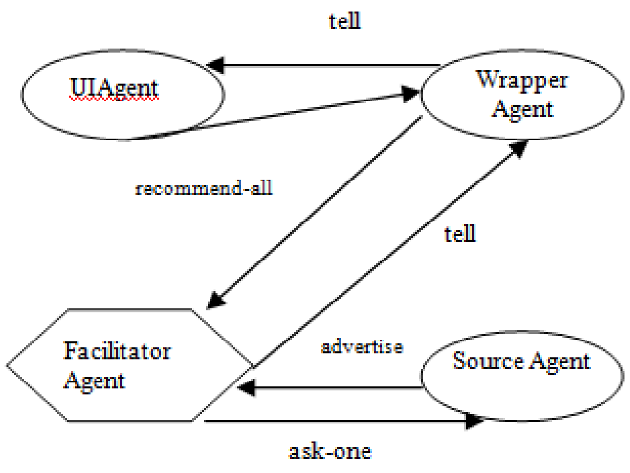

Fig. 10 - Diagrammatic representation of Performatives exchanged between agents.

V. KQML Message Exchange

KQML message format is as given below:

sender: is the agent that sends the message.
content: makeup of the message.
receiver: agent that is intended to receive message
language: the query language like prolog, but the authors have selected SPARQL.
ontology: the underlying common ontology for vocabulary of the organization of agents.

```
(tell
:sender UIAgent
:content (DNA_PATTERN    DNA pat)
:receiver WRAPPERAgent
:language SPARQL
:ontology DNA_ONT)

(recommend-all
:sender WRAPPERAgent
:content (DNA_PATTERN    DNA pat)
:receiver FACILITATORAgent
:language SPARQL
:ontology DNA_ONT)

(advertise
:sender SOURCEAgent
:receiver FACILITATORAgent
:language SPARQL
:ontology DNA_ONT)
:content    (stream-all :content (DNA_PATTERN    DNA ?pat)))

(ask-one
:sender FACILITATORAgent
:content (DNA_PATTERN    DNA ?pat)
:receiver SOURCEAgent
:language SPARQL
:ontology DNA_ONT)
```

```
(tell
:sender SOURCEAgent
:content (DNA_PATTERN    DNA res)
:receiver FACILITATORAgent
:language SPARQL
:ontology DNA_ONT)

 (tell
:sender FACILITATORAgent
:content (DNA_PATTERN    DNA res)
:receiver WRAPPERAgent
:language SPARQL
:ontology DNA_ONT)

(tell
:sender WRAPPERAgent
:content (DNA_PATTERN    DNA res)
:receiver UIAgent
:language SPARQL
:ontology DNA_ONT)
```

The syntax of KQML and the implementation details are taken from ARPA Knowledge Sharing Effort (KSE)[9]

5.1 SPARQL and DNA_ONT Ontology

SPARQL which is used as a query language for KQML, queries the DNA_ONT ontology during first level of verification. Additionally, reasoning can be applied in knowledge extraction[9][10]. A sample SPARQL code is as shown below:

```
SELECT ?pat
WHERE {
   ?DNA_PATTERN pat ?pat .
}
```

The query fragment selects the DNA Pattern belonging to the Class DNA_PATTERN held by the pat variable.
An extract of the common owl ontology (DNA_ONT) made in Protégé 4 for inter-agent messaging is shown below.
Major classes of this ontology as can be seen in the diagram are Organism, Features, Pattern, Pattern_Length, Accession and Origin. These belong to every organism and are useful information for annotation[11][12][19]. In the figure below a Pattern ID, here Pat_ID is dnas67.24.19 that has a pattern length of 44 and also contains a DNA Pattern consisting of sequence of A, C, T and Gs.

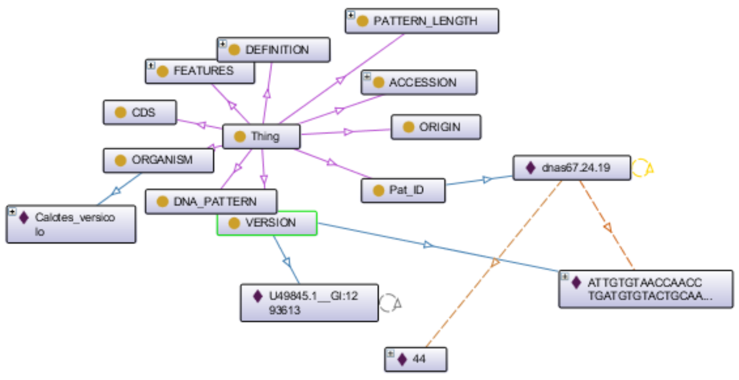

Fig 11. DNA_ONT ontology extract

```
<ObjectPropertyAssertion>
       <ObjectProperty IRI="#has_FEATURES"/>
```

```
        <NamedIndividual
IRI="#1..5028_organism=Saccharomyces_cerevisiae_db_xref=taxon:4932___chromosome=IX__map=9"/>
    </ObjectPropertyAssertion>
    <ObjectPropertyAssertion>
        <ObjectProperty IRI="#has_DNA_PATTERN_LENGTH"/>
        <NamedIndividual IRI="#44"/>
    </ObjectPropertyAssertion>
    <ObjectPropertyAssertion>
        <ObjectProperty IRI="#has_DNA_PATTERN"/>
        <NamedIndividual IRI="#ATTGTGTAACCAACCTGATGTGTACTGCAACCTGATGTGTACTG"/>
    </ObjectPropertyAssertion>
    <ObjectPropertyAssertion>
        <ObjectProperty IRI="#of_ORGANISM"/>
        <NamedIndividual IRI="#Calotes_versicolo"/>
    </ObjectPropertyAssertion>
```

The Web Ontology Language OWL is used to develop the DNA_ONT ontology. The scrap of owl above depicts a small segment of the DNA_ONT ontology.

VI. Policies

Policies frame the rules that the agents in any case have to abide. The infrastructure of the MAS lays on the Policies. The policies define the constraints and a functional boundary within which the agents act.
Some stringent policies enforced on the agents in GIR are as below:

1. ONLY A, C, T, or G characters can be used to represent a DNA Pattern.
2. The search string cannot contain white spaces, hyphens or any special character or numbers.
3. Max Length of a search pattern can be set but assumed to be 1024 chars.
4. Break-up size of search pattern can be set but assumed to be 11 chars.
5. Only the Initiator Agent through Human Agent can modify the startup file.
6. The UIAgent can be resident on the same machine as the Initiator or on a different machine.
7. The DNA Pattern is accepted by the UIAgent only if it is syntactically correct.
8. The DNA Pattern approved by UIAgent is passed to the WRAPPER Agent.
9. Wrapper Agent recommends the DNA Pattern to the Facilitator Agent.
10. Eligible SOURCE Agents Advertize to the Facilitator Agent
11. Facilitator Agent direct the query to the Source Agents.
12. Source Agent replies back to the Facilitator Agent
13. Facilitator Agent replies back to WRAPPER Agent
14. Wrapper Agent returns query results to UIAgent
15. The duty of the UIAgent is to give the output to the user.
16. The Facilitator agent is responsible for routing incompletely addressed messages. It maintains a registry of service names

6.1 Policy Ontology with KQML

A subset of KQML messages passed between agents concerning policy rules are as below. These messages form a part of the second level of verification.

```
(broadcast
:sender INITIATOR
:content (DNA_PATTERN_PERM_LENGTH    DNA ?p_len)
:receiver ALL
:language SPARQL
:ontology OMaSE_POLICY_ONT)

(broadcast
:sender INITIATOR
:content (Divided_DNA_PATTERN_PERM_LENGTH    DNA ?d_p_len)
:receiver ALL
:language SPARQL
:ontology OMaSE_POLICY_ONT)
```

```
(ask-one
:sender INITIATOR
:content (STATUS    DNA ?status)
:receiver UIAgent
:language SPARQL
:ontology OMaSE_POLICY_ONT)

(tell
:sender UIAgent
:content (STATUS    DNA status_value)
:receiver INITIATOR
:language SPARQL
:ontology OMaSE_POLICY_ONT)
```

These policies form the basis on which the agent communication[13][14][15] is validated. Policies provide the constraints and the duties the agent has to answer. Policies form a domain for the agent[16] which it is not allowed to trespass. In the beginning of the paper it was discussed as to how the policies are framed into ontologies[17][18] to provide verification. SPARQL is used to query the OMaSE_POLICY_ONT ontology to retrieve and confirm that agent communication is going according to the norms. An extract from the OMaSE_POLICY_ONT developed in PROTÉGÉ 4 is as shown below.

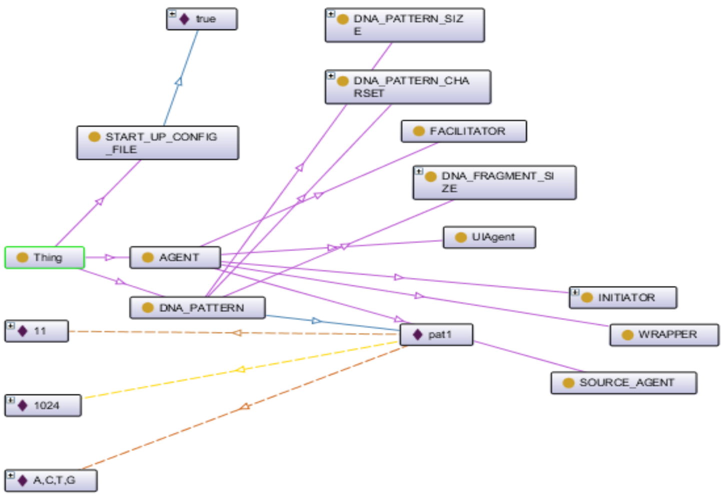

Fig. 12. OMaSE_POLICY_ONT ontology extract

In the above ontology the classes are identified as Agent, its sub classes are INITIATOR, UIAGENT, WRAPPER, FACILITATOR and SOURCE_AGENT. There is a class called DNA_PATTERN that has an instance called pat1 with allowable character set {A,C,T,G}, has maximum pattern size of 1024 and maximum segmented DNA pattern size of 11 characters.

VII. Conclusion

In this work, the authors have remedied the problem of vagueness, incompleteness and ambiguity in inter agent communication. Messages without thorough support of underlying ontology to the communication language and protocol may tend to create uncertainties and obscurity in message exchange. Here, a two level verification of the overall communication has been conducted. Firstly, a common ontology, DNA_ONT is devised that is used by KQML in managing the routine message exchange. Then, another ontology, OMASE_POLICY_ONT is built that is used by KQML for Policy Verification. The two of the major shortcomings in O-MaSE Methodology namely, Validation of the Requirements and verification of inconsistencies are dealt with. Additionally, SPARQL is used as a query language for KQML providing better information retrieval from RDF/OWL resources. As a future scope Context Sensitive Policy Handling is proposed to be another addition in O-MaSE. A fully implemented Multi-Agent Framework incorporating the Dual Verification System has been started to be worked upon.

References

[1] 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, Gaurav Kant Shankhdhar, Manuj Darbari.2016.

[2] O-MaSE: A customisable approach to designing and building complex, adaptive multi-agent systems, Scott Deloach, https://www.researchgate.net, 2016.

[3] Gaurav Kant, Atul Verma , Verified Message Exchange in Providing Security for Cloud Computing in Heterogeneous and Dynamic Environment, International Journal of Applied Information Systems, 2017

[4] Gaurav Kant Shankhdhar, V K Singh 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.

[5] Tim FININ "KQML as an agent communication language" University of Maryland Baltimore County Baltimore MD USA, 1995

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

[7] Foundation For Intelligent Physical Agents, FIPA ACL Message Structure Specification, fab@fipa.org

[8] Yi Zhang,2016 "Research on Efficient SPARQL Query Processing for RDF Data" DOI: 10.2991/iwmecs-15.2015.94

[9] R. Neches, www-ksl.stanford.edu/knowledge-sharing/papers/kse-overview.html.

[10] Dhanda, Namrata, Manuj Darbari, and Neelu J. Ahuja. (2012) "Development of Multi Agent Activity Theory e-Learning(MATeL) Framework focusing on Indian Scenario." International Review on Computers and Software 7.4 (2012).

[11] Ankit Jagga, Aarti Singh (2016) "Assessment of KQML Improved ", Int. J. on Adv Networking and Applications. Volume 07 Issue 6.

[12] 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.

[13] Sahai, Priya, and Manuj Darbari.(2014) "Adaptive e-learning using Granulerised Agent Framework." International Journal of Scientific & Engineering Research 5.2 (2014).

[14] Yagyasen, Diwakar, Manuj Darbari, and Hasan Ahmed (2014). "Transforming non-living to living: a case on changing business environment." IERI Procedia 5 (2013): 87-94.

[15] Darbari, Manuj, Sanjay Medhavi, and Abhay Kumar Srivastava.(2007) "Application of UML for modeling urban traffic system using producer consumer theory to generate process algebra model." Journal of International Technology and Information Management 16.4: 75-82.

[16] Darbari, Manuj, and Bhaskar Karn.(2008) "Enterprise Modelling using Unified Framework supporting Distributed Object Computing." Journal of International Technology and Information Management 17.3 (2008): 3.

[17] Darbari, Manuj, and Hasan Ahmed. (2011) "Fuzzy Logic Application in Modeling Bioinformatics Sequence Markup Language (BSML)." International Journal of Database Theory and Application 4.1 (2011): 1-6.

[18] Yagyasen, Diwakar, and Manuj Darbari.(2015) "Quantification of Dynamic Business Environment by Development of Ontology using Task Reduction." International Journal of Scientific & Engineering Research, Volume 6, Issue 1, January-2015

[19] Dhanda, N., Ahuja, N. J., Darbari, M., & Siddiqui, I. A. (2013). An Adaptive Normative Multi Agent System using Web 3.0 for e-Learning Platform. AJIT-e, 4(12), 7.

[20] K. Bryson, M. Luck, M. Joy, and D. Jones 2000. Applying agents to bioinformatics in geneweaver. In Proceedings of the Fourth International Workshop on Collaborative Information Agents.

[21] Cossentino, M., Gaud, N., Hilaire, V. Galland, S. Koukam, A. ASPECS: an agent-oriented software process for engineering complex systems. Journal of Autonomous Agents and Multiagent Systems Systems. 20, 260–304 (2009)

[22] DeLoach, (2008) S.A., Oyenan, W., Matson, E.T. A capabilities based model for artificial organizations. Autonomous Agents and Multiagent Systems. 16, 13–56

[23] DeLoach, (2010) S.A., Miller, M. A goal model for adaptive complex systems. International Journal of Computational Intelligence: Theory and Practice. 5, 83–92

[24] DeLoach, S.A., Valenzuela Jorge, L. (2006) An agent environment interaction model. In: Padgham, L., Zambonelli, F. (eds.) Agent-Oriented Software Engineering VII: 7th International Workshop, AOSE 2006. LNCS Vol. 4405, pp. 1-18. Springer: Heidelberg

[25] Garcia-Ojeda, J.C., DeLoach, S.A. Robby. (2009) agentTool process editor: supporting the design of tailored agent-based processes. In: Proceedings of the 2009 ACM symposium on Applied Computing (SAC '09), pp. 707-714, ACM: New York

[26] Sapna MaharP 1 P, Pradeep Kumar Bhatia, Measuring the Intelligence of Software Agent, P 2 P P 1,2PComputer science and Engineering, Guru Jambheshwer University of Science & Technology, Hisar, Haryana, INDIA, IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 6, August 2014.

[27] Gaurav Kant Shankhdhar, Narendra Jha and Atul Verma, Change Management in Semantic Web Services in legal domain using FSM and XXM, International Journal of Applied Information Systems, 2015.

[28] J.V. Diggelen, et al., Efficient Semantic Information Exchange for Ambient Intelligence. Comput. J. Vol 53, No. 8, 2010, pp. 1138-1151.

[29] M. Cochez, Semantic agent programming language: use and formalization, in Department of Mathematical Information Technology, 2012, University of Jyväskylä.

[30] G. Stoilos, G. Stamou, and J.Z. Pan, Fuzzy Extensions of OWL, International Journal of Approximate Reasoning, Vol 51, No. 6, 2010, pp. 656 - 679.

[31] P. Sutton, Vagueness, Communication and Semantic Information, 2013 King's College.

IOSR Journal of Computer Engineering (IOSR-JCE) is UGC approved Journal with Sl. No. 5019, Journal no. 49102.

Gaurav Kant Shankhdhar. "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)IOSR Journal of Computer Engineering (IOSR-JCE) 19.4 (2017): 08-18.