Chapter 4

AN INTELLIGENT MULTIAGENT INTERFACE FOR SEMANTIC WEB

4.1 Introduction

Although Context based information retrieval is the main motivation behind the semantic web however, in order to improve the performance of semantic web, research efforts have been made in varying directions. In fact, an analytical study of literature indicates that researchers have made attempts to lay a smooth floor for the working components (agents and ontologies) of semantic web but the same careful investigation also reveals the fact that the floor laid is not yet smooth as we could identify the following potential areas which still needs attention:

- Standardization of Ontology Mapping Interface
- Secure Communication Strategy for communication among MAS
- Ensuring Security at all phases of MAS Interaction
- Handling Uncertainty involved in user queries
- Ensuring Robustness of agents in MAS

In order to address the above highlighted issues, this work aims to propose a Multi Agent Framework for Semantic Web (MAFSW).

4.2 High Level View of MAFSW

MAFSW is designed with an objective to provide complete communication solution to MAS operating in semantic cyberspace. It aims to not only facilitate standard communication interface for MAS capable of providing ontology mapping in both homogeneous and heterogeneous domains but also intends to establish trust among
communicating MAS. In addition, security of communicated messages and fault-tolerant features has also been paid attention. It mainly comprises of following components as depicted in Fig. 4.1.

1. Intelligent and Adaptive Mapping Mechanism for Multiagent Interface (Ontology Mapping Layer)
2. Contract Net Trust Establishment Protocol (Trust Establishment Layer)
3. Elliptical Curve Cryptography based security engine (Security Layer)

This framework consists of various agents working individually and collaboratively as well, at various layers shown in Fig. 4.1. Each layer performs a specific functionality and contributes towards overall goal of efficient communication among multi-agent systems. Various agents in each layer are arranged in hierarchical manner so that parent agent can communicate with other connected agents (sociological agents) who may adopt the goal of parent agent which in turn communicate with other descendents and so on.
This type of topology reduces communication load. The agents in all places are assumed to be communicating through shared memory communication method, which is not explicitly shown, to reduce complexity. Upcoming sections provide the description of various layers mentioned above.

4.3 Ontology Mapping Layer

Communication among various MASs’ in semantic web solely relies on ontology mapping. But before focusing on the mapping of ontologies, the design structure of ontology database must be paid attention. Ontology database development is comparatively an ignored area, but in order to work on mapping mechanism for ontologies used by different MAS, intrinsic details of ontology designing must be clearly understood.

4.3.1 Ontology Database

Ontology is an explicit specification of a conceptualization where, conceptualization is an abstract and simplified view of the world that describes the objects, concepts and other entities, existing in a domain along with their relationships [144].

In Artificial Intelligence ontologies are categorized according to three key dimensions namely, Formality, Purpose and Subject [89] where Formality refers to the design structure and meaning of vocabulary and Purpose and Subject defines the intended use of the ontology and the nature of the subject that the ontology is characterizing respectively. The detailed classification of these dimensions is illustrated in Fig. 4.2.

Ontologies [111] have emerged as a significant component of Semantic Web [98],[137]. Use of semantics can not be materialized without use of ontology. Due to this reason, development of ontology needs proper attention. Studies [30],[58],[89],[154] indicated that many techniques have been proposed and are in use, but most of them are application specific. At the time of citation no one general technique is available for the ontology development, which is flexible enough to be applied to any application.
For the purpose of constructing ontology the research group at Edinburgh university proposed the first ontology construction method in 1996 [87] also known as the ‘Skeletal method’ as shown in Fig. 4.3.

Fig. 4.3 High Level View of Skeletal Method Proposed by Uschold [87],[89]
Most ontology construction methods concentrate on modeling aspects, rather than how domain concepts and relationships are to be elicited, which is also equally important. This research work proposes a technique for ontology development based on Language Extended Lexicon (LEL) technique [75]. The underlying principle of the lexicon is contextualism, according to which context of use of a system must be clearly understood before requirements can be derived and also the motivation of deploying LEL in developing ontological database is the systematization for the elicitation, modeling and analysis of ontology concepts.

LEL is a representation of the symbols in the application language anchored on the simple idea of understanding the language of the problem without understanding the problem to be solved, i.e. major emphasis is on understanding the semantics of the problem in hand. Central to LEL are two principles, principle of closure and principle of minimal vocabulary where:

- **Principle of closure**: is to maximize the use of other lexicon terms when describing the notion and behavior response of a new term.
- **Principle of minimal vocabulary**: is to minimize the use of terms external to the Universe of Discourse (UoD). If unavoidable, we must ensure that they belong to the basic vocabulary of the natural language in use.

Next section throws light on the model proposed for development of ontological database.

### 4.3.2 The Proposed Design Structure

The proposed ontology is expressed as a set of six tuple \((C, R, t, CH, rel\_terms, rule\_set)\) where:

- \(C\) and \(R\) are disjoint sets, called the set of concepts & set of relations respectively.
- \(CH \subseteq C \times C\) is a concept hierarchy or taxonomy.
- \(t\) is the set of lexicon terms/tokens.
rel_terms: $R \rightarrow t \times t$ is a function that defines the relation between different terms in the UoD.

rule_set: is a set of axioms for ontology, expressed in appropriate logical language.

The process of developing ontology comprises of three major steps:

- Identify the UoD for which ontology is required and the sources of information for that.
- Identify and prepare a list of relevant terms for the UoD.
- Classify the terms in the domain. The terms can be classified as: subject(s), object (o), verb (v) and state (st). Basically each term in the ontology is composed of function/notion and behavior /connotations. At this stage the major emphasis is on extracting the meaning of each term clearly. When describing the terms, we should enforce the principle of closure and minimal vocabulary.

The classified terms are than mapped to define a specific ontology as depicted in Fig.4.4

Next section discusses the mapping procedure of term set to ontology set.

**The mapping procedure**

This section proposes the mapping procedure (MPo) which performs the translation of terms to ontology. MPo is a three step process described as below:
1) **Maintaining the term list (term_list):** Chronological list of terms is maintained according to their type (subject, object, verb, state), called as term_list for reference here.

2) **Maintaining the behavior list (behavior_list):** A list of behavior is maintained for each term where;

   - Behavior is defined as temporal changes of parameter values. The behavior model of a component is ideally independent from the context which the component is used in [154].

3) **Maintaining the function list:** A list of functions is maintained for each term where;

   - Function: A function is defined as a result of interpretation of a behavior under an intended goal [154].

Every term is first matched against the term_list for a match. If true, context-based behavior is chosen form the behavior_list and then mapping from behavior to function list is performed. Here it is essential to identify mapping primitives between behavior and function that enable the reasonable and effective interpretation of the behavior. Algorithm for the ontology development is depicted in Fig. 4.5 given below.
4.3.3 An Example

Following example illustrates the proposed algorithm. Consider a UoD say animal. Let the animal to be searched is mouse. This term can have two behaviors for instance, either an animal i.e. biological being or input component of computers i.e. mechanical thing. Now on the basis of the context in which above term appears appropriate behavior can be opted. Fig. 4.6 given below illustrates this example. Depending on the behavior of the term, appropriate function can be chosen to satisfy the context.

Fig. 4.5 Algorithm for Ontology Development
This example can be expanded and generalized using bottom up approach and a broad ontological database can be formed.

![Diagram of term mapping in ontology knowledge base]

Fig. 4.6 Mapping of Term in Ontology Knowledge Base

Now, with the growing size of web, more and more applications are employing MAS for performing various tasks and providing the desired services to the users. Sometimes MAS are expected to provide services which are beyond their domain or capabilities. Thus in order to facilitate those tasks for the users, MAS needs to communicate with other MAS. With large number of ontologies on the web, mapping of ontology from among heterogeneous MASs’ becomes a complex issue as compared to the MAS operating in homogeneous domains. Although lot of efforts for mapping of ontologies in homogenous domains have already been done, but there is no consensus on one common platform. Also, none has addressed ontology mapping in both homogenous as well as heterogeneous domains. Also, ontology mapping frameworks developed so far require human intervention for mapping, which is practically not suitable in web based applications. Thus it was observed that there is need of a common layer in SW, through which various ontologies can be accessed and information can be exchanged semantically. In this work a common layer referred to as *Intelligent and Adaptive Mapping Mechanism for Multiagent Interface (IAM3I)* [13] shown in Fig. 4.7 has been designed that attempts to rectify the above listed shortcomings.
4.3.4. Intelligent and Adaptive Ontological Interface (IAM3I)

The complete framework mainly comprises of three prime components *Source_Domain*, *Destination_Domain* and *IAM3I* wherein *Source_Domain* and *Destination_Domain* comprises of Personal task agents (PTAs), Ontology Database (ODB) and an Interface Agent (IFA) whereas *IAM3I* contains three agents namely, Information Agent (*IA*), Message Handler Agent (*MHA*) and Mapping Master Agent (*MMA*). Following are the few key assumptions accounting to simplicity and reusability:

(a) Key Assumptions

- **Support for IA:** It gathers information from *Source_Domain* and passes mapped message to *Destination_Domain*. IA has access to ontologies of all participating domains. It can not only fetch RDF communication tags but also, can extract the definition of various terms from their respective URIs as and when required.

- **Support for MMA:** It is associated with two data structures namely, *Thesaurus_Based_Dictionary (TBD)* where TBD has been devised using available
semantic networks such as wordnet\(^1\) and a *Temporary Log (TL)* that holds the definitions of the most recently accessed keywords along with their URIs. TL has been implemented as two-dimensional array where rows represent \(C_s\) as concept names and the URI\(_s\) containing definitions for that \(C_s\), form the columns. Thus there are multiple columns for each \(C_s\). Initially matrix will be populated with null values but since intelligence and adaptivity have been incorporated as two important parameters, it is assumed that as the matrix will become more and more populated with \(C_s\) and URI\(_s\), it would behave intelligently and would definitely provide the desired definition.

**Support for IAM3I:** The proposed framework makes use of two concepts namely Extension and Intension as discussed below:

- **Extension:** Extension of a concept is the set of instances that can be classified under that concept. The extension of the universal concept (everything) \(C_u\) is the set of all instances while the extension of the absurd concept (nothing) \(\bot\) is the empty set (Koes, Nourbakhsh and Katia (2004)). Mathematically, it can be explained considering: (\(C_s\) set of concepts in source ontology \(O_s\) (\(C_T\) is set of concepts in target ontology \(O_T\)).

\[
\begin{align*}
C_s &\in O_s \land C_T \in O_T \\
C_s' &\supseteq C_s \land C_T' \supseteq C_T \text{ then} \\
\text{if } (\left( C_s \rightarrow \neg C_T \right) \lor \left( \neg C_s \rightarrow C_T \right)) &\text{ then} \quad \ldots \ldots \text{(1)} \\
\text{extension relation holds.}
\end{align*}
\]

- **Intension:** It can be defined as the subsumption of sets of instances classified under the concept reflecting the \(\subset_c\) relation (Koes, Nourbakhsh and Katia (2004)). Mathematically it can be formulated as

\(^1\) http://www.cogsci.princeton.edu/wn/
\[
\text{if} \left\{ \left( \left. \begin{array}{c}
(O_s \rightarrow O_T \forall O_s \supseteq O_T) \\
\lor (O_s \rightarrow O_T \forall O_s \subseteq O_T)
\end{array} \right) \lor (O_s \rightarrow O_T \forall O_s \subseteq O_T) \right\}
\text{then}
\]

intension relation holds

The explanation of concepts of extension and intension is provided in the upcoming sections.

(b) The High Level View

This section presents the ecology of agents included in the proposed framework. Fig. 4.8 given below illustrates the coordination among the source domain, destination domain and IAM3I.

![Proposed Intelligent Ontological Interface](image)

Fig. 4.8 Proposed Intelligent Ontological Interface

The high level view of Fig. 4.9 is explained as follows and the working algorithms are presented in the subsequent sections:

1. PTA_{source\_domain} communicates the input phrase to IFA_{source\_domain} which provides interface necessary for communication with other agent based applications.
2. $I_{FA_{source\_domain}}$ extracts the ontology $O_s$ from $O_{DB_{source\_domain}}$ and forwards the clubbed message $(input\_phrase \ and \ O_s)$ to $IA_{IAM3I}$.

3. $IA_{IAM3I}$ hands over the message to $MHA_{IAM3I}$ for converting the message in standard format by eliminating stop words and grammatical words. $IA_{IAM3I}$ has access to both $O_s$ and $O_T$ and it passes them to $MMA_{IAM3I}$ for usage.

4. $MHA_{IAM3I}$ passes the message to $MMA_{IAM3I}$.

5. Mapping algorithm firstly checks the TBD to check if the keywords in the source message are synonymous with the keywords in $O_T$.

6. If keywords are synonymous then $MMA_{IAM3I}$ replaces the source keywords with the keywords of $O_T$ and the mapped message is sent to $I_{FA_{destination\_domain}}$ through $IA_{IAM3I}$.

7. In case if keywords are not synonymous then $MMA_{IAM3I}$ will explore definitions of the keywords, so as to be sure that same things are not referred using different names.

8. For increasing the recall rate and reducing the mapping time, $MMA_{IAM3I}$ is provided with TL, which keeps the definitions of most recently accessed keywords along with their tags and URIs. $MMA_{IAM3I}$ will first explore the TL before requesting $IA_{IAM3I}$ for fetching definitions of the keywords, if definitions containing same tags are found they can be used, leading to amplified recall rate.

9. In case definition is not available in the TL, $MMA_{IAM3I}$ will send request to $IA_{IAM3I}$ for fetching the required definitions from their respective URIs. $IA_{IAM3I}$ fetches and passes definitions to $MMA_{IAM3I}$, which explores definitions to decide whether intension holds with the classification given in $O_T$, so that mapping can be performed with some loss of information. If intension relation holds, message is mapped and sent to $IA_{IAM3I}$.

10. If the source definitions are not subset of the classification used in $O_T$ then $O_S$ needs to be extended in order to carry on the communication, thus $MMA_{IAM3I}$ sends ontology extension request to $IA_{IAM3I}$ along with the class that needs to be added in $O_S$.

11. $IA_{IAM3I}$ in turn passes the extension request to $I_{FA_{source\_domain}}$ along with the class needed for the extension of the $O_S$.

12. $I_{FA_{source\_domain}}$ extends the $O_S$ by adding the class at appropriate place. This is possible because of the learning ability of the agents and is required due to the fact
that agent doesn’t always need to communicate with other frameworks containing subset of their ontologies or working on similar kind of ontologies only. In case of frameworks working on similar domains but using different classification or working on entirely different domains, communication can be ensured only due to the extension feature of source_domain ontology, in our proposed framework. With the passage of time extension of ontologies will make them richer and easy to communicate with other frameworks.

Now mapping becomes feasible and can be performed either through exact or intension relation of keywords mapping as discussed above.

13. Finally the mapped message is delivered by IA_{IAM3I} to the IFA_{destination_domain}, which finally passes it to the desired PTA_{destination_domain}.
Algorithms for various agents are depicted in Fig. 4.10(a), 4.10(b), 4.10(c) and 4.10(d), along with these, algorithm for updating TL is also provided in Fig. 4.10(e). Efficiency of TL will depend on its implementation, if implemented with appropriate amount of memory, it can lead to increase in precision and recall rate of the proposed system.
\begin{strip}
\begin{flalign*}
\text{Interface agent()}
\text{Input: message, extension\_request(Cs)}
\text{Output: message to IA}
\text{Action: activate, sleep, extension(Os)}
\{
&\quad on(input) \rightarrow \text{activates;}
&\quad \text{if(message)}
&\quad \{\text{pass to IA;}\}
&\quad \text{if (extension\_request(Cs))}
&\quad \{\text{extend(Os) s.t. } (Cs \subset Os' \\text{ s.t. } \text{∀ Os' } \supset eq Os )\}
&\quad \text{if (message\_from\_IA)}
&\quad \{\text{pass to desired agent in domain;}\}
&\quad \text{call(sleep)}
\}
\end{flalign*}
\end{strip}

\text{Fig. 4.10(a) Interface Agent}

\begin{strip}
\begin{flalign*}
\text{Information agent()}
\text{Input: message,Os,Ot, fetch(Dc), extension\_request(Cs);}
\text{Output: to\_MHA(message, Os,Ot);}
\text{Action: activate, sleep;}
\{
&\quad on(input) \rightarrow \text{activates;}
&\quad \text{if(message)}
&\quad \{\text{pass to MHA(message ,Os,Ot);}\}
&\quad \text{else}
&\quad \text{if(fetch(Dc))}
&\quad \{\text{fetches(Dc,URI);}
&\quad \text{Pass Dc to MMA;}
&\quad \}
&\quad \text{if(extension\_request(Cs))}
&\quad \{\text{pass extend(Cs) to Interface Agent();}\}
&\quad \text{call(sleep);}
\}
\end{flalign*}
\end{strip}

\text{Fig. 4.10(b) Information Agent}
Mapping Master Agent()
Input: message, Os, Ot;
Output: message mapped in Ot, extend(Cs);
Action: activate, sleep, extension;
{
  on(input) → activate;

Case 1: if (∀ Cs ∈ Os) ∃ (Ot ∈ Ot) s.t. Cs ≡ Ot)
  { return( mapped message);}

Case 2: if((Os → Ot) s.t. Os ⊇ Ot) ∨ (Os → Ot’’) s.t. (Ot’’ ⊆ Ot))
  { return(mapped message);}

Case 3: if (Os → ¬ Ot) ∨ (¬Os → Ot))
  { return (call(extension));}
  call (sleep);}

Message handler Agent()
Input: message, Os, Ot;
Output: message to MMA;
Action: activate, sleep;
{
  on(input) → activate;
  If(message)
  {
    transform(message) → standard_form;
    pass(message,Os,Ot) to MMA
  }
  call(sleep);
}

Temporary_Log()
TL is a two dimensional matrix of Cs and URIs;
{ for each (cs ∈ Cs: cs ⊇ Cs,a)
  {
    if (not(TL[Cs,URI] = Null) )then
      return (Dc);
    else
      create TL[Cs,URI] = Dc;
  }
return TL;
}

Fig. 4.10(c) Mapping Master Agent
Fig. 4.10(d) Message Handler Agent
Fig. 4.10(e) Updation algorithm for Temporary Log
A case study is presented in the next section demonstrating the working of proposed framework.

(c) Case study

Suppose communication is desired between Source_domain: Student and Destination_domain: Hospital (see Fig. 4.11 (a)).

PTA\_{source\_domain} Input: A student needs medical services.

IFA\_{source\_domain} : \langle phrase, O_S, O_T \rangle

IA\_IAM3I : \langle phrase, O_S, O_T \rangle

MMA\_IAM3I : (i) student \neq Person

(ii) \{student\} \subset \{Person\} so intension relation holds

IA\_IAM3I output: A person needs medical services.

IFA\_{destination\_domain} Input: A person needs medical services.

If communication is required in the reverse direction then \{Person\} \supseteq \{Student\} extension is required. Thus IA\_IAM3I will send ontology extension request to IFA\_{source\_domain} with concept Person with all its attributes and its relationship with student (student \subset person). Only in this way agent working in source\_domain can understand what a person is and can participate in communication.

Extension of ontology is possible due to learning ability of agents and it will lead to richness in the domain ontology with passage of time.
The next example illustrates the usage of ontology mapping interface for homogenous domains. Both ontologies include vocabulary of animals but use different classifications.

Fig. 4.11(a) Ontology Mapping Between Student and Hospital

Fig. 4.11(b) Ontology Mapping Between Living Being and Animals
source_domain: classification of living beings

destination_domain: classification of animals

Case 1:

PTA_{source\_domain} Input: A snack is in the farm.

IFA_{source\_domain} : <phrase, O_S >

IA_{IAM3I}: <phrase, O_S, O_T >

MMA_{IAM3I} : (i) snack ≠ animal

(ii) \{snack\} ⊆ \{Reptile\} so intension relation holds

IA_{IAM3I} output: A reptile is in the farm.

IFA_{destination\_domain} Input: A reptile is in the farm.

Case 2:

PTA_{source\_domain} Input: Lizard eats mosquitoes.

IFA_{source\_domain} : <phrase, O_S >

IA_{IAM3I}: <phrase, O_S, O_T >

MMA_{IAM3I} : (i) Lizard ≠ animal

(ii) \{lizard\} ⊆ \{Reptile\} so intension relation holds

IA_{IAM3I} output: reptile eats mosquitoes.

IFA_{destination\_domain} Input: reptile eats mosquitoes.
Thus in both cases, keywords Snack and Lizard will be mapped to reptile class in $O_T$ resulting in the loss of their specific attributes. However this kind of loss of information has to be tolerated due to the large and distributed nature of ontologies. Since it can never be assumed that similar classification will be used and similar attributes will be included while creating ontologies. Also, difference in origin, application area and the thought process of the ontology developer leads to differences in ontologies defined even for similar domains.

\textit{(d) Evaluation}

This section evaluates the proposed framework, along with some existing known ontology mapping mechanisms. Table 4.1 depicts the comparison of IAM3I with four other mapping mechanisms on criterion such as input received, output provided, user interaction, mapping strategy/algorithm employed, scalability, recall rate and precision.
Table 4.1 Evaluating IAM3I with other Mapping Mechanisms

<table>
<thead>
<tr>
<th>Criterion</th>
<th>CTXMATCH$^{[183]}$</th>
<th>GLUE$^{[43]}$</th>
<th>ONION$^{[86]}$</th>
<th>PROMPT$^{[99]}$</th>
<th>IAM3I</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>Concepts in concept hierarchy</td>
<td>Two taxonomies with their data instances in ontologies</td>
<td>Terms in two ontologies</td>
<td>Two input ontologies</td>
<td>Communication phrase with Source &amp; target ontologies</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>Semantic relation between concepts</td>
<td>A set of pairs of similar concepts</td>
<td>Set of articulation rules between two ontologies</td>
<td>A merged ontology</td>
<td>Mapped communication phrase from source to target ontology</td>
</tr>
<tr>
<td><strong>User Interaction</strong></td>
<td>No, being an algorithm</td>
<td>User provides data for training and also provides similarity measures</td>
<td>A human expert chooses or discards or modifies suggested matches using a GUI tool</td>
<td>The user accepts, rejects or adjusts system’s suggestion</td>
<td>No interaction is required as it is a layer of service and is hidden from user.</td>
</tr>
<tr>
<td><strong>Mapping strategy/Algorithm</strong></td>
<td>Logical deduction</td>
<td>Multi-strategy learning approach: Machine learning technique</td>
<td>Linguistic matcher, structure and inference based heuristics</td>
<td>Heuristic based analyzer</td>
<td>Lexical similarity, whole term, word constituent and type matching</td>
</tr>
<tr>
<td><strong>Scalability</strong></td>
<td>Poor, since it works more effectively when data is less</td>
<td>Good, works effectively when amount of data is large</td>
<td>Poor, for larger ontologies the algorithm doesn’t scale well</td>
<td>Poor</td>
<td>Scalability is very Good due to composition being agent based</td>
</tr>
<tr>
<td><strong>Recall rate</strong></td>
<td>Poor</td>
<td>Poor</td>
<td>Good (nearly 70%)</td>
<td>Good</td>
<td>Very Good, due to the use of temporary log</td>
</tr>
<tr>
<td><strong>Precision</strong></td>
<td>Good</td>
<td>Good (66%-97%)</td>
<td>Good</td>
<td>Good</td>
<td>Very Good, since no suggestions are required from outside</td>
</tr>
</tbody>
</table>

For evaluation purpose, measures provided in literature for ontology mapping tools are used [95]. For evaluating the proposed framework, the datasets from machine learning repository (ML, 2007) and USGS Geographic Names Information System (GNIS, 2009) datasets have been chosen. The experiment was carried out using data sets of size 200 to
1000 words, for fixed time and number of words mapped were recorded. The graph given in Fig. 4.12 below illustrates the performance of the proposed system.

![Mapping trend in fixed time](image)

Fig. 4.12 Graph for Performance of Proposed System

It is clear from the graph that with increase in size of data set shown on X axis, number of matches (shown on Y axis) increase rapidly due to the learning ability of the proposed system.

*The unique contribution of the ontology mapping layer is that it completely delegates ontology mapping functionality to agents, thus automating the mapping process. Further, it can provide mapping in both homogeneous & heterogeneous domains with same efficiency. Extension mechanism employed in this layer allows it to extend the ontologies whenever a new concept arrives initially for mapping, thus with passage of time ontology becomes richer and can provide mapping without delay.*

### 4.4 Secure Communication Strategy

Communication of MAS in SW follows Contract Net Protocol (CNP) which is a standard provided by Foundation for Intelligent Physical Agents (FIPA), a non-profit organization working for standardization of protocols for MAS. CNP facilitates interaction between
agents for fully automated negotiations in the form of contracts. Although CNP addresses most of the issues related to agent communication in distributed environment, still shortcomings such as lack of formal coordination procedure for participants to follow is existing. Many variations in conventional CNP have been proposed by researchers to incorporate mechanisms for handling specialized interactions. However present variations of the communication protocol don’t ensure the truthfulness of agents which is very essential for open, dynamic and heterogeneous MAS. Establishing trust amongst agents in MAS would help them in choosing appropriate communication partners. Since, CNP do not consider the trust as a prime factor therefore, the necessity of Trust Establishment Protocol was apparent. Thus a new version of CNP incorporating trust establishment capability called as Contract Net Trust Establishment Protocol (CNTEP) [7] has been proposed in this research work.

4.4.1 CNTEP Architecture

Trust and reputation are crucial for effective communication in open MASs as agents are owned by a variety of stakeholders and they can participate or leave a system dynamically. It may be noted that participating agents are likely to be unreliable, self-interested and possessed with incomplete knowledge. Moreover, since agents are designed to behave intelligently and work in team therefore their intentions might change with time.

The available literature as presented above strongly recommends the need of a protocol that could establish a level of trust among interacting agents. In order to meet the above stated need, a trust establishment protocol is being proposed and hence incorporated in existing CNP to help monitoring and selecting their interaction partners.

Before, proposing the main TEP, following are some basic terms that must be understood:
- **Trust**: Trust can be defined as an agent’s belief in another agent that its (later’s) behavior will support formers plan of action. Further, Trust has been categorized in two ways [33]:
  - **Interaction Trust**: It generates from past experience of direct interactions among agents.
  - **Role-based Trust**: It arises from various role-based relationships between the agents.

- **Reputation**: Reputation can be considered as observation of an agent’s past behavior by a society of agents. Reputation can be categorized in the following two ways [33]:
  - **Witness Reputation**: It results from reports of witnesses about an agent’s behavior by other agents.
  - **Certified Reputation**: It emerges from references provided by organizations dealing with/creating the agent. For example, The agent created by IBM shall have certificate from IBM itself. The agents get registered with FIPA in order to prove their authenticity to others whenever required.

Although there are four ways of establishing trust and reputation but certified reputation is very crucial for agents entering in a new system as other solutions can face some drawbacks in certain cases. For instance, if agent A has not interacted with agent B ever before, neither A nor B has any information to calculate the interaction trust. Moreover, A may not be able to find relevant witness ratings about B, or the search process may be too complicated. Also the role-based relationship will not exist as they had not interacted ever before. Also, all these conditions may prevail at the same time if agent A has just joined the environment. Thus it may not be able to assess agent B’s trustworthiness at all. Traditionally, if agent B presents its certificate, its trustworthiness can be assessed to an extent. However, the proposed approach is different than the traditional as it considers both trust and reputation methods at the same time. The
proposed architecture comprises of a Trust Matrix, which will be discussed in the subsequent sections.

4.4.2 Working of CNTEP Architecture

The proposed approach focuses on incorporating trust establishment mechanism with in the existing Contract Net Protocol. The high level view of CNTEP is shown in Fig. 4.13.

CNTEP mainly comprises of an Initiator Agent (IA), Bid Evaluation Agent (BEA), Contractor Agent (CA) and Trust Establishment Protocol (TEP), wherein TEP further comprises of Trust Verification Agent (TVA), Trust Matrix (TM) & Agent Registration List(ARL). The IA sends the list of keywords to be searched in the form of Call for Proposal (CFP) to the perspective CAs. CAs are not allowed to directly revert back to IA unless or until they posses Trust Certificate (TC). Therefore instead of reverting back to the respective IA, the CA executes TEP.
Now, when a CA calls for authentication to TEP the TVA gets activated and in first instance it demands for certificate that authenticates the agents as registered agents. In turn CA presents all the certificates, it is possessed with. The TVA verifies the same and consults ARL if the same CA is a registered agent and had delivered the reliable results in past. If an entry for the same exists, the TM is consulted to compute trust percentile.

Table 4.2 represents the TM where the columns represent two trust and two reputation parameters respectively. Here presence of 1 in a cell indicates existence of that trust parameter whereas a 0 in a cell indicates absence of the same.

Table 4.2 Trust Matrix

<table>
<thead>
<tr>
<th>Trust Parameters</th>
<th>TRUST</th>
<th>REPUTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contractors</td>
<td>I.T.</td>
<td>R.B.T.</td>
</tr>
<tr>
<td>CA1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CA2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CA3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CA4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

It shall be noted that for the agents participating in for the first time, certificate of authentication is used as certificate of reputation (CR). For instance, as evident from the TM, CA1 is only possessed with CR as indicated by value 1, Then TVA grants a TC with a percentile of 0.25. A CA meeting all four criterions will be given a maximum percentile of 1.0. A CA having 0.0 percentile implies that it is neither FIPA nor registered to any other organization. Such a CA is never granted a TC and is actually rejected. The detailed criterion for evaluating Trust Percentile from TM is as follows:

\[
\text{Trust Percentile} = \begin{cases} 
1.0 & \forall \text{CAi} \exists < \text{IT, RBT, WR, CR} > = \{1,1,1,1\} \\
0.75 & \forall \text{CAi} \exists < \text{IT, RBT, WR, CR} > = \{1,1,0,1\} \cup \{1,0,1,1\} \cup \{0,1,1,1\} \\
0.50 & \forall \text{CAi} \exists < \text{IT, RBT, WR, CR} > = \{1,0,0,1\} \cup \{0,1,0,1\} \cup \{0,0,1,1\} \\
0.25 & \forall \text{CAi} \exists < \text{IT, RBT, WR, CR} > = \{0,0,0,1\} \\
\end{cases}
\]

\[\text{Range of Trust Percentile is given as:} \quad \{0.25 \leq \text{TrustPercentile} \leq 1.0\}\]
Trust percentile will be computed as follows:

\[
\text{Trust Percentile} = 0.25 \times \text{IT} + 0.25 \times \text{WR} + 0.25 \times \text{RBT} + 0.25 \times \text{CR} \quad \ldots \ldots (4)
\]

The CA’s which are able to prove their identity and gain some value of the percentile are granted TC with the respective percentile value mentioned on it. The percentile clearly indicates the degree of trust of a particular agent. In contrast, CA’s with 0 percentile are considered to be malicious and are sent Trust Not Established (T.N.E) message. Now CA’s with a TC send their bid (response + TC) to IA, where the proposed bids are added to the Bid_List, a data structure comprising of all bids. Moment, the submission deadline expires, the final Bid_List is sent to BEA for task assignment. BEA evaluates the bids and past performances of each CA. The past performances can be evaluated from the History Buffer (HB). On completion of a task, BEA updates TM in accordance to HB which is updated periodically on the basis of previous events. The HB maintains a record of each contract including contractor id, contract status whether successfully completed, pending or unsuccessful. This status helps BEA in updating TM. BEA after evaluating Bid_List sends the list of most Feasible Contractors Agents (FCA) to IA, which in turn sends the proposal approved message to them. The contractor accepting the invitation initiates the job and finally reports as Task Successful, Task Pending or Task Unsuccessful to the IA in due time, which is essential in CNP. The IA keeps record of the responses in the HB through BEA and later utilizes them in policy making. A successfully completed job increases the credit of the FCA and it becomes eligible for a witness reputation certificate. Similarly a failure decreases its social reputation and credibility. At the completion of a job, TM is updated to reflect new trust value for that particular contractor, which will be helpful in future interactions with same contractor. Fig. 4.14(a) and Fig. 4.14(b) below provide the flow diagram of complete CNTEP architecture and its implementation in IAM3I.
Fig. 4.14(a) Flow Diagram for CNTEP

Fig. 4.14(b) Flow Diagram for CNTEP in IAM3I
Algorithms for the various agents involved in the proposed framework are given in Fig. 4.15(a)-4.15(d) respectively.

**Figure 4.15(a) Initiator Agent**

```plaintext
Initiator agent()
Input: KFU: keyword from user, BCA: bid from CA, FCA: FeasibleContractorAgents; Report;
Output: CFP: CallforProposal, BidAcceptedMessage; BidRejectedMessage;
Action: activate, sleep;
{ Input (KFU);
 Activate IA;
 If (KFU)
 Set timer; Generate CFP; Activate CA;
 CA ← CFP; IA ← CA(Bid);
 If (bid & timer timed out)
 BEA ← IA(bid); Activate BEA;
 IA ← BEA(FCA)
 Generate Bid_accepted_Message;
 Activate FCAi;
 ∀ activated FCAi
 (FCAi ← IA( Bid accepted message);
 IA ← FCAi(Report);
 TM ← IA(Report);
 Update(TM);
 sleep;
 }
```

**Figure 4.15(b) Trust Establishment Protocol**

```plaintext
Trust_Establishment_Protocol()
Input: Response, Report from BEA;
Output: TC, TNE, Message;
Action: Activate, sleep;
{ Activate TVA;
 TVA ← CA(certificate);
 If (certificate==FIPA)
 { CR==I; Trust Percentile= 0.25*CR;
 check ARL; TVA ← ARL(Registered_CA);
 TM ← TVA(Registered_CA);
 rustPercentile=0.25*IT+0.25*WR+0.25*RBT+0.25*CR
 TVA ← TM(Trust Percentile);
 TC ← TVA;
 CA ← TVA(TC)
 sleep; }
 else
 set Trust Percentile=0.0; CA ← TVA(TNE);
 sleep; }
```

**Figure 4.15(c) Contractor Agent**

```plaintext
Contractor agent ()
Input: CFP: CallforProposal, ProposalAcceptedMessage;
Output: Report: Task_Successful, Task_Pending, Task_Unsuccessful;
Action: Activate, Successful, Pending, Unsuccessful;
{ Input(CFP);
 Activate(CA);
 if(CFP)
 { list_of_proposals ← CA;
 if (interested)
 { Activate TEP;
 TEP ← CA(Certificate); CA ← TEP(TC||TNE)
 If(TC)
 Bid ← Response+TC;
 Activate IA; IA ← CA(Bid);
 CAi ← IA(Bid accepted) v IA(Bid not accepted);
 If(Bid acceptable)
 Activate FCAi; Execute Task; Generate report;
 IA ← FCAi(Report);
 Sleep; }
```

**Figure 4.15(d) Bid Evaluation Agent**

```plaintext
Bid Evaluation agent()
HB: HistoryBuffer;
Input: Bids, Reports;
Output: FCA: FeasibleContractorAgents;
Action: activate, sleep;
{ If(Bid ← CA)
 Bid_list ← Bid;
 If (timer timed out)
 { Max(TC) ← Bid_list; FCA_List ← BEA;
 Activate IA; IA ← FCA_List;
 If (Report ← CA)
 }
 { Update HB;
 if (task successful message)
 update TM for CA(s);
 }
 sleep;
 }
```

90
The unique contribution of this layer is that it provides a new protocol for identifying trust among agents. The proposed TEP mainly relies on the trust and reputation parameters which can be earned more and more as an agent completes a task successfully. A CA is assigned a task iff it has earned a minimum value of trust. Therefore this work greatly adds to more secure communication among agents and hence more reliable output is generated.

4.5 Secure Communication

MAS involve interactions of autonomous agents to provide desired services to their users. MAS provide new services and functionalities to the users by co-operating with other agents which makes them versatile and appealing solutions. This flexibility of providing services to users raises significant security challenges for the agents participating in communications. Concerns about security are actually two fold: First, checking the authenticity of the agents prior to communication i.e. check whether they are what they claim to be. Secondly, even after their authenticity has been proved, ensuring security of communicated messages among communicating MAS.

Encryption of mobile agents and communicated messages is one of the solutions for ensuring security. However traditional encryption algorithm such as DH, DSA and RSA, employ key sizes which are very large resulting in high time and space complexity. In contrast to this Elliptical Curve Cryptography (ECC) technique is a public key cryptosystem that besides using much smaller key sizes is able to provide a competitive security edge as that of other strong encryption algorithms. Most attractive feature of ECC is its relatively short operand length compared to that of RSA and also it is based on discrete logarithm in finite fields. ECC can provide various security services in the form of key exchange, communication privacy through encryption, authentication of sender and digital signatures to ensure message integrity.

Research efforts have been made to elaborate applicability of ECC in wireless security, WLAN authentication and privacy infrastructure (WAPI), Wireless Transport Layer
Security (WTLS) and many more. However, an in depth review of literature indicates that researchers have remained quite towards applying ECC into semantic cyber space. Thus an ECC based security engine [45] has been proposed which amalgamate security and trust, as these two dimensions are intertwined and the complete security cannot be assumed without focusing on both of them simultaneously.

4.5.1 Elliptic Curve Cryptography: An Overview

The elliptic curve discrete logarithm problem can be defined as follows. Choose a prime p and an elliptic curve:

\[ Q = xP \] ..................................(5)

Here xP denotes the point P on elliptic curve added to itself ‘x’ number of times. Now the elliptic curve discrete logarithm problem is to determine ‘x’ given P and Q. It is relatively easy to compute Q given x and P, but it is very difficult to determine x given Q and P.

For elliptic curve cryptography we are interested in a restricted form of elliptic curve defined over a finite field. For cryptographic utility elliptic group ‘mod p’ is of particular interest, where p is a prime number. It is defined using two nonnegative integers ‘a’ and ‘b’, less than ‘p’, satisfying equation:

\[ 4a^3 + 27b^2 \pmod{p} \neq 0 \] ............(6)

Then \( \text{Ep}(a,b) \) represents the elliptic group mod p whose elements \((x, y)\) are pairs of nonnegative integers less than p satisfying

\[ y^2 = x^3 + ax + b \pmod{p} \] ............(7)

along with the point at infinity \( O \).
4.5.2 Encryption / Decryption using Elliptic Curve Keys

The first task in this process is to encode the plaintext message \( m \) to be sent as point \( P_m(x,y) \). Actually it is the point \( P_m \) which will be encrypted as cipher text and decrypted subsequently. It is important to note here that we can’t simply encode the message as \( x \) or \( y \) coordinates of a point since all such point may not satisfy \( E_p(a, b) \). Also both key exchange system and encryption/decryption process require a common point \( G \) and an elliptic curve \( E_p(a,b) \) as parameters. Once these two factors are decided among communicating parties, user A selects a private key \( P_{RA} \) and generates a public key as

\[
P_A = P_{RA} \times G
\]  \hspace{1cm} (8)

Here \( P_A \) and \( P_{RA} \) denote public & private key of A respectively.

To encrypt and send a message \( P_m \) to B, A selects a random positive integer \( x \) and produces the cipher text \( C_m \) consisting of the pair of points:

\[
C_m = \{ xG, P_m + xP_B \}
\]  \hspace{1cm} (9)

It can be seen that A uses B’s public key \( P_B \) while encrypting message for it. For decryption of the ciphertext \( C_m \), user B multiplies the first point of the pair by its own secret key ( \( P_{RB} \) ) and subtracts the result from the second point i.e.

\[
P_m + xP_B - P_{RB}(xG)
\]  \hspace{1cm} (10)

\[
P_m + x(P_{RB}G) - P_{RB}(xG)
\]  \hspace{1cm} (10)

\[
P_m
\]

It means that A masks the message \( P_m \) by adding \( xP_B \) to it. Now none except A knows the value of \( x \) thus even though \( P_B \) is a public key, no one can remove the mask \( xP_B \). On the other hand A includes a clue which is sufficient to remove the mask if one knows the private key \( P_{RB} \). For an intruder to recover the message, the intruder would have to compute \( G \) and \( xG \) which is quite hard.
Next we discuss Elliptical Curve Digital Signature Algorithm (ECDSA) [115]:

ECDSA Signature Generation Algorithm: ECDSA_sign_gen( )

The user A signs the message m using three steps:

1. Select a random integer \( k \in [2,n-2] \)
2. Compute \( k.d=(x_1,y_1) \) and \( r=x_1 \mod n \).
   
   If \( x_1 \in \text{GF}(2^k) \), it is assumed that \( x_1 \) is represented as a binary number.

   If \( r=0 \) then goto step 1.

3. Compute \( k^{-1} \mod n \).
4. Compute \( s=k^{-1}(H(m)-dr) \mod n \). [Here H is the secure hash algorithm SHA]

   If \( s=0 \) goto step 1.

5. The signature for the message m is the pair of integers \((r,s)\).

ECDSA Signature Verification: ECDSA_sign_veri( )

The user B verifies A’s signature \((r,s)\) on the message m by applying the following steps:

1. Compute \( c=s^{-1} \mod n \) and \( H(m) \).
2. Compute \( u_1=H(m)c \mod n \) and \( u_2=r \mod n \).
3. Compute \( u_1.P+ u_2.Q=(x_0, y_0) \) and \( v=x_0 \mod n \).
4. Accept the signature if \( v=r \).

4.5.3. Proposed Work: ECC-based Security Engine

The proposed framework initially establishes trust among the communicating agents and later ensures the confidentiality and integrity of the messages that are being communicated. Primarily, Trust Establishment layer (TEL) and Secure communication
layer (SCL) are the two components. Since, the working of TEL has already been discussed in the previous sections; this section discusses working of SCL only.

Every agent is registered to a Central Certificate Authority (CCA). On authentication of the agent, CCA issues a trust certificate (TC) containing agent-id, range of its elliptical curve public keys, its elliptical curve private keys and validity period of the keys. CCA grants a digitally signed TC to every authenticated agent. CCA makes use of EC key generation algorithm for generating public and private keys of the agents. Fig. 4.16 given below illustrates the process of handshaking and encrypted message transmission.

![Fig. 4.16 Process of Handshaking and Message Transmission](image)

Now, whenever two agents want to communicate or an agent want to access a resource from a platform, initially it will go through the process of handshaking, in which agent A will send communication request to agent B along with its TC and other Reputation certificates (RC). Security layer of agent B verifies DS of CCA to ensure validity of the TC. Once verified, it forwards the same to TEL for computation of Trust Percentile which is calculated using table 4.2 and equation (3) & (4) as already discussed.

The agent which is able to prove its identity and gain at least minimum value of the mentioned percentile is chosen for handshaking. TP clearly indicates the degree of trust
or trustworthiness of a particular agent and different values of TP may provide different privileges to the requesting agent. It may vary according to the policy of the supporting organization, or policy of the MAS that which rights should be provided at a particular TP. In contrast, agents with 0 percentile are considered to be illegal and are sent Trust Not Established (T.N.E.) message. Table 4.3 illustrates the level of privileges assigned at different trust percentiles in this work.

<table>
<thead>
<tr>
<th>Trust Percentile</th>
<th>Privilege Assigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Access Denied</td>
</tr>
<tr>
<td>0.25</td>
<td>Request for providing services can be entertained</td>
</tr>
<tr>
<td>0.50</td>
<td>Request for providing services can be entertained as well as agent can be considered for taking some services, but access of resources not permitted</td>
</tr>
<tr>
<td>0.75</td>
<td>Request of resources permitted</td>
</tr>
<tr>
<td>1.0</td>
<td>All requests can be entertained</td>
</tr>
</tbody>
</table>

TEL after calculating TP returns back to B for further decision. If agent A has TP value greater than or equal to 0.25, B sends handshaking message back to it, indicating that trust has been established and it is ready for communication. In the second phase communicated messages will be encrypted by the security layer. Encryption/Decryption will make use of ECDSA as earlier discussed. The Fig. 4.17 given below provides a high-level view of the security and trust framework.

![Fig. 4.17 High-Level View of the Framework](image-url)
The flow diagram given in Fig. 4.18 illustrates the working of the proposed framework.

1. Agent A in MAS1 initiates a communication request for agent B in MAS2, by sending a communication request (c_request) to security layer at its end.

2. The trust component of the security layer, i.e. Trust Interface agent (TIA) performs two tasks:
   - From a requesting agent it receives communication request and sends (c_request, <TC>,<RC>) tuple to the recipient.
   - On receiving agent side, it receives (c_request, <TC>, <RC>) tuple, verifies <TC> in terms of DS of CCA using its public key. Sends <TC> to TEL and gets TP computed by it.

3. TEL returns the TP for submitted TC. If TIA of agent B finds TP acceptable, it sends handshaking message back to A, and also decides the level of privileges for it. Otherwise it sends T.N.E. message.

4. On receiving handshaking message, agent A starts actual communication. Now the messages sent by A are handled by Endecryption (Encryption-Decryption) module of the security layer which encrypts the message before transmission using the public keys of the receiver, referring it from the Certificate and Keys Buffer (CKB). It also adds digital signatures of the sender on the message for authentication after encrypting the message. On the receiving end encrypted messages are received by the same layer, which first verifies the digital signature of the sender by using its public key and then decrpyts the message using receiver’s private key and plaintext message is then passed to recipient B.
Algorithms for various agents involved in the proposed framework are given in Fig. 4.19(a), 4.19(b) and 4.19(c) respectively.
Trust_Verification_Agent()
Input: TC, RC;
Output: TP, TNE, Message;
Action: Activate, sleep;
{ on TC Activate()
    if (certificate==CCA)
    {
        CR==1;
        compute Trust Percentile= 0.25*CR;
        check ARL;
        TVA ← ARL(Registered_CA);
        TM ← TVA(Registered_CA);
        Compute Trust Percentile=(0.25*IT+0.25*WR +0.25*RBT+0.25*CR)
        TVA ← TM(Trust Percentile);
        return( TP);
        sleep;
    }
    else
    {
        set Trust Percentile=0.0;
        return(TNE);
        sleep;
    }
}  

Encryption_Decryption_Agent()
Input: P_m:Plaintext_message,C_m:encrypted_message;
Output: C_m, P_m;
Action: Activate, sleep;
{ on input activate()
    if ( input== P_m )
    {
        check receiver_id;
        check Certificate_Key_Buffer;
        EDA ← P_B (Public_key(receiver_id));
        C_m ← { xG, P_m+x P_B };
        Digitalsig(C_m);
        return(C_m);
    }
    if(input== C_m )
    {
        check sender_id;
        check certificate_key_buffer;
        EDA ← P_A (Public_key(sender_id));
        ECDSA_sign_veri(P_A );
        if (verified)
        { EDA ← PR_B(private_key(receiver));
            decrypt(C_m , PR_B )
            { P_m + x P_B - PR_B (xG)
            P_m + x(PR_B (xG)PR_B (xG)P_m }
            return(P_m);
        }
        else
        return(failure_message);
    }
    sleep();
}
The Fig. 4.20(a) and 4.20(b) depicts the class diagram and interaction diagram for the proposed work respectively.

```
Trust_Interface_Agentl()
Input: c_request, TC, RC;
Output: Handshaking, TNE, (c_request,<TC>);
Action: Activate, sleep;
{
    on input activate():
        if (input==c_request)
            {send(c_request,<TC>);
             on receiving(Handshaking)
                 return(communication allowed);
             on receiving(TNE)
                 return( permission error);
             sleep();}
    if(input==<c_request,TC>)
        { verify DS(CCA);
            if (true)
                { TVA ← <TC,RC>
                 TVA → ARL(registered(TC));
                    If(true)
                        { TM ← agent_id;}
                    else
                    {ARL ← register_request(<TC>);
                 TM ← <TC>}
            Compute TrustPercentile≡=(0.25*IT+0.25*WR +0.25*RBT+0.25*CR)
                 TVA ← TM(Trust Percentile);
                    return( TP);}
            return(Handshaking);
            sleep;}
    else
        {set Trust Percentile=0.0;
         return(TNE);
         sleep; }
}
```

Fig. 4.19 (c) Trust Interface Agent
Fig. 4.20(a) Class Diagram for the Proposed Framework
Unique contribution of this layer is that it provides an ECC based security engine to MASs operating in semantic cyberspace ensuring two dimensional securities. In addition to computing trust percentile, the proposed framework makes use of elliptical curve keys for encryption/decryption purpose, which increases message security to a larger extent. Thus, this work adds greatly towards improvement of communication security of Multi-Agent Systems. Major advantage of this framework is that existing application are not required to be written from scratch, instead it can be implemented in the security layer of existing model of semantic web.

While extending our works towards providing context sensitive information to web users, it was found that user queries are vague and uncertain and obviously, the search engines that are making use of ontology databases containing crisp knowledge only, are not able to handle imprecise or uncertain inputs. This research work has made an attempt to extend
conventional ontology with fuzzy logic thus providing **Fuzzy Integrated Ontology Model** (FIOM).

### 4.6 Fuzzy Integrated Ontology Model

Fuzzy logic has already been applied in wide range of applications and this technique has returned impressive results across large variety of domains where human like reasoning and behavior is required. This technology promises capacity of implementing machine intelligence. Thus it makes sense to embed this crucial technology into ontology which bridges the gap between human understandable soft logic and machine understandable hard logic in SW. The next section provides the basics of Fuzzy set theory.

#### 4.6.1 Introduction of Fuzzy Set Theory

Fuzzy logic mainly focuses on quantifying vague or uncertain terms that appear in our natural language conversations. These terms are known as linguistic variables [81], or fuzzy variables. Zedeh [81],[82], “Father of fuzzy logic theory” highlighted that these linguistic variable which are building blocks of Computing with words(CW) methodology, might allow machines to work with perception based rational decision making in environments of imprecision, uncertainty and partial truth. Imprecision and uncertainty are key features of user exploitation of WWW. For example query ‘Biodata of persons who are experienced’ is a fuzzy statement. As who can be considered as experienced is not clear. Here biodata is a linguistic variable having value experienced. The range of all values that a linguistic variable can acquire makes its Universe of Discourse (UoD). Major thrust of fuzzy logic is use of linguistic variable in contrast to quantitative variables for representing imprecise concepts.

**Fuzzy set and membership values**

Conventional set theory allows the members of sets to have only one of two possible values either true or false, i.e. either the member belongs to a set or they don’t, but there may not be partial membership. This can be mathematically stated by using a membership function \( m_A(x) = 1 \) if \( x \in A \) and \( m_A(x) = 0 \) otherwise.
Fuzzy set theory extends conventional set theory by allowing the members to have degrees of membership or truth. A fuzzy set is characterized by a membership function which defines range of possible values for a variable. Capability of associating fuzzy variables with membership values is the major thrust of fuzzy logic which makes it suitable for handling uncertainty in SW.

4.6.2 The Proposed FIO M

This work extended the general design structure for ontology database (already discussed in previous sections) by providing general ontology with the capability of processing linguistic variables, new sets of fuzzy concepts as well as fuzzy qualifiers have to be included. The extended ontology called as FIO is expressed as a six tuple \((C, R, CH, CF, Q, U)\) where:

- \(C\) is set of concepts
- \(R\) is set of relations among the concepts defined in set \(C\) like \(\subseteq, \supseteq, \leq, \geq, \neq, =\)
- \(CH \subseteq C \times C\) is concept hierarchy or taxonomy for the domain of interest.
- \(CF\) is the set of fuzzy concepts over the concepts defined in \(C\). For e.g. In Ontology for employees, set \(C\) will contain descriptions for concepts like employee, manager, salary, designation etc. whereas \(CF\) will contain description for terms like experienced, fresher, young, old, teenager etc.
- \(Q\) is set of qualifiers for the fuzzy terms which help in mapping fuzzy terms with membership values. For example, The employee ontology stated above will have \(Q\) values as \{less, more, high, very, not very …..\}
- \(U\) denotes the UoD for the domain under consideration.

One important step in FIO is to associate \(Q\) terms with membership values. UoD can be defined with different values for different terms.

Reconsidering the example taken earlier ‘Biodata of all experienced persons’ can be defined over UoD= \([0, 50]\) i.e. range of experience value is taken between 0 to 50 years. Fig. 4.21 illustrates the same.
Fuzzy qualifiers like less, more etc. have been defined taking same UoD, as shown in Fig. 4.22 given below:

Now for implementing this ontology, the domain concepts, fuzzy concepts and qualifiers are to be defined using RDF. Values of the qualifiers will be associated with numbers using membership functions. For the example taken earlier i.e. the biodata of experienced persons, attributes of persons are defined using RDF as shown in Fig. 4.23 below.
Value of experience tag can be fetched and then associated with corresponding membership value from the fuzzy ontology, which will result in whether the person is fresher or less experienced, experienced or much experienced etc.

Thus using Fiom uncertainty in user queries and requests can be handled, leading to user satisfaction and better exploitation of knowledge available on web.

In order to accommodate uncertainty and vagueness in domain knowledge one possible solution is to incorporate ability of computation with words (i.e. fuzzy logic) in semantic web at ontology level. Next section provides the details of mechanism proposed for fault management.
4.7 Fault Management Layer

Multi Agent Systems (MAS) are prone to failures such as processor failure, communication link failure, software bugs and attacks by malicious agents leading to faults in MAS. However continuous service provision regardless of any failure is the most desired feature of any MAS. Fault management in such a system spans across fault detection, fault avoidance, fault tolerance and fault recovery [122]. Fault tolerance is an approach to increase dependability of an application or system avoiding system failures in the presence of faults [73]. Inherent modularity of MAS produces some extent of fault tolerance in itself but dynamic nature of working environment, non-deterministic nature of agents and lack of central controlling authority in MAS makes it difficult to predict and handle errors, in advance. It requires improving existing architecture of agents as well as agent platforms to make these systems more effective and fault tolerant. Thus need of a strategy for providing fault tolerance in MASs is very apparent.

Replicating system components, either in hardware or software is most widely used technique for improving fault tolerance of such systems. Replication is basically creating one or more duplicate agents in the system. Each of these duplicates is capable of performing the same task as the original agent. The aim is whenever a component fails; another component shall immediately take the charge without affecting the stability of system and also, continues to provide services to the users with an acceptable delay. However, this process increases the overall complexity of the system and incurs overhead. Literature indicates that replication can be classified [37] as Active replication and Passive replication. Active replication processes the input with the help of all replicas simultaneously. It means that state of the replica is updated dynamically where as, in passive replication, only original agent processes the input messages and its state is transmitted to all its replicas periodically, in order to maintain consistency. Though active replication provides fast recovery in case of failures but it requires more cost and system resources. Thus it is preferred for hard real time applications. Passive replication on the other hand imposes less cost and resource overhead on the system but results in delays while recovering in case of any failure [37].
In order to have an effective fault tolerance strategy, system should have an appropriate mix of active and passive replication. Since all the components (agents) in a system, may not have same importance for the overall working of the system, thus instead of having active replication for all the agents, system shall have active replication for more critical agents and passive replication for lesser ones. This strategy not only, shall provide optimal utilization of system resources and reduce system complexity but also should lead to optimal fault tolerance to the system. It has been observed from the literature [159] that absence of centralized controlling authority in MASs makes it difficult to detect and handle failures of individual agents which leads to fault propagation and serious performance degradation of the MAS. This provided the inspiration for the proposed work.

Literature review highlighted the fact that although many techniques for providing fault tolerance in MAS have been proposed, there is no universal consensus on any one scheme. As mentioned above, various frameworks provide flexibility of applying active and passive replication adaptively but do not consider criticality of agents. Therefore there exists a gap which demands adaptive replication strategy that dynamically assesses agent criticality based on its cardinality, thereby increasing the degree of fault tolerance of MAS under consideration.

4.7.1 The Proposed Fault Tolerance Framework

This section throws light on intrinsic details of the proposed fault tolerance framework. Although it makes use of replication technique in order to provide fault tolerance but major focus of the work is to provide equilibrium in replication strategy based on criticality of an agent. The replication strategy will change as the criticality of agent changes thus adapting to the changes in the environment.

The proposed framework named as Central Fault Management Unit (CFMU) consists of three components namely Fault Management Agent (FMA), Event Monitor agent (EMA) and Replica stores (Active & Passive replica stores).
FMA controls the fault management and keeps track of the kind of replication i.e. either active or passive, to be provided to an agent, along with the status of its replica. The type of replication depends upon the criticality of the agent. Criticality assessment can be carried out using emergent graph structure of agents in MAS, which highlights their interdependence and thus can be useful in assessing positional importance of an agent in MAS. Emergent graph of communicating MAS is created by EMA and provided to FMA.

Also, an EMA is responsible for identifying any crashed agent so as to substitute a replica in place of failed agent. To determine fault in communicating agents EMA timestamps every communicated message and waits for response from the agent for a predefined threshold time, after which an agent can be assumed to crash and same is reported to FMA for further actions.

The Replica store maintained by CFMU is divided in two parts, Passive Replica Store (PRS) & Active Replica Store (ARS). PRS is in turn divided into two parts, code section and state section, for keeping the code and state of the passively replicated agents. State of such agents is updated periodically. ARS on the other hand contains working replicas for the critical agents. Details of active and passive replication are provided in the next section.

Fig. 4.24 given below provides the high level view of the proposed framework. The diagram contained in the highlighted box labeled as IAM3I, is taken from ontology mapping framework [13]. However any MAS can be substituted in its place, since here the focus is on fault tolerance of MASs, no matter what job they perform.
4.7.2 Criticality Assessment

Dynamic replication (active or passive) is the key contribution of this work which is achieved by computing the criticality of MAS under consideration. The criticality in turn is computed using agent’s cardinality which can be defined as the number of communication links associated with that agent. This implies that the cardinality of an agent is directly proportional to the importance or the activity of any agent. Further, cardinality can be accessed from emergent graph structure of MAS. Failure of critical
agents can result in much more loss as compared to those having less criticality. Thus FMA can provide active replication to agents having severe criticality whereas less critical agents can be provided passive replication.

**Emergent Graph Structure**

An emergent graph can be defined as a set:

$$G = \{N, E\}, \text{ where } N \text{ is the set of nodes and } E \text{ is the set of edges}$$

In the emergent graph structure of MAS, agents form the nodes and communication links between the agents provide the set of edges. Now for all agents (N) varying from 1 to n and associated edges (E), varying from 1 to m. Cardinality of an agent ($n_i$) is given as:

$$\text{Cardinality} (n_i) = \begin{cases} 0 & \forall \ i=1 \ & e=0 \\ 2 & \forall \ i=2 \ & e=2 \\ e_{ij} \cup e_{ji} & \forall i \leq n-1 \ & 1 \leq j \leq m \end{cases} \quad \ldots \ldots \ (11)$$

Emergent graph of MASs changes dynamically as agents can enter or leave a system at any time, thus variation in the cardinality of agents is a continuous process. Consider an emergent graph as shown in Fig. 4.25.

The graph comprises of 6 agents and 12 communication links. Now, the cardinality of agent N5 can be computed as:
Card(N5) = \{ e(N3,N5) \cup e(N4,N5) \cup e(N5,N3) \cup e(N5,N4) \cup e(N5,N6) \cup e(N6,N5) \} = 6

Hereon an agent having minimum cardinality of four or more will be referred to as Critical Agent (CA). Critical agents are provided active replication considering them important for the working of the overall MAS. Importance of an agent for the system is also influenced by factors like the nature and capability of its task performance. For active replication of CAs, whenever a task starts, a replica is created for that agent and every input processed and resultant state change of the agent is reflected in its replica by the FMA. Once the task pending with CA is terminated successfully, its replica can be washed out, as intermediate results are no longer required. Final state change for the CA is maintained in the memory. This way dynamic replication can be provided to different sensitive agents using same memory block.

Coding of all other agents (non-critical) will be kept in the code area of the CFMU and their state will be maintained in the passive replicas, which are updated after execution of every task, so as to update their experience and learning.

FMA maintains a table as shown below (refer table 4.4) for keeping record of the replication status of each agent in the MAS. Here, A-11 represents an actual agent where as R-11 represents the replica of same.

<table>
<thead>
<tr>
<th>Agent-id</th>
<th>Cardinality</th>
<th>Type of replication</th>
<th>Replica-id</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-11</td>
<td>2</td>
<td>P</td>
<td>R-11</td>
</tr>
<tr>
<td>R-11</td>
<td>2</td>
<td>P</td>
<td>A-11</td>
</tr>
</tbody>
</table>

Although literature highlights that use of more than one replica for a single component is common, to provide more robust fault tolerance, but it obviously consume more
resources of the host system. Thus this framework keeps only one replica (active or passive) for a component which will take charge when the working component fails. The actual agent and its replica exchanges their roles whenever actual agent fails in to carry the task further Unless and until the application is very hard real time, use of more than one replica is not recommended.

Second row entry in the above table is indicating that when agent A-11 crashes, its replica i.e. R-11 takes its charge and plays the role of active node, in the mean time former is recovered and then substituted as replica of R-11. This way keeping only one replica for each node in the system can make it work.

Algorithms of agents involved in the proposed framework are given in Fig. 4.26(a) and 4.26(b) respectively.

Event Monitor Agent ( )
Input: CR: communication request among MASs.
Output: EG(N,E), crashed(agent_id);
Action: activate, sleep;
Input (CR)
{
activate(EMA);
create EG(N,E);
FMA ← EG(N,E);
For every (message)
message ← (message , timestamp);
if (response_time>threshold_time)
{ if (replication_status==passive )
{ FMA ← crashed(agent_id);
FMA ← update(replication_status_table);
}
substitute(agent ← replica);
}
sleep();
}

Fig. 4.26(a) Event Monitor agent
Thus the proposed framework can significantly increase the fault tolerance of scalable multi-agent systems by making use of proper mix of active and passive replication strategies. Fig. 4.27 provides the flow diagram for the above framework.

Fig. 4.26(b) Fault Management agent
This layer proposed and hence employed hybrid replication strategy for providing fault tolerance to distributed agent systems. The proposed framework exploits the agents’ cardinality and hence criticality to maintain the balance between static and dynamic replication, providing effective fault tolerance. The results obtained are consistent to the stated objectives as it was found that the only fifty percent of the agents were required to
be actively replicated and rest remained in passive state, thus reducing the overall system complexity.

The Fig. 4.28 given below provides the detailed view of the proposed MAFSW. In this diagram the agents which are critical are shown in grey color, with application of the proposed fault-tolerance strategy the dynamic replication requirement reduced to half. Thereby improving overall efficiency of the system.

---

**Fig. 4.28 Detailed View of Multi-Agent Framework for Semantic Web (MAFSW)**
4.7 Conclusions

In this chapter some of the issues in the field of SW are undertaken and their solutions are proposed. The algorithms of various agents involved in different layers are also presented. Proposed frameworks have also been evaluated independently; however the evaluation of the integrated framework is left and will be provided in the next chapter.