Multi-Agent and Knowledge-Based System for Power Transformer Fault Diagnosis

Thesis submitted in accordance with the requirements of
the University of Liverpool for the degree of Doctor in Philosophy by

Farhad Davoodi Samirmi

September 2013

To my family

Acknowledgements

It would not have been possible to write this doctoral thesis without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here.

My first debt of gratitude must go to my primary supervisor, Prof. Henry Wu, for his guidance, caring, patience, and providing me with an excellent atmosphere for doing research. His wisdom, knowledge and commitment to the highest standards inspired and motivated me.

I would like to express my gratitude to my secondary supervisor, Prof. Wenhu Tang, for his invaluable advise and unsurpassed knowledge, time and attention.

I am very thankful to Prof. Peter McBurney from King's College, London, for his scientific advice and knowledge and many insightful discussions and suggestions during first year of my study.

Special thanks to Prof. Michael Wooldridge, the University of Oxford, for his invaluable help in multi-agent programming and help in development of my background in this area.

A would like to acknowledge my good friends, Jonathan and Waqar, for their support, friendship and assistance has meant more to me than I could ever express.

A very special thanks to my loving and caring family for their support and understanding. I owe them everything and wish I could show them just how much I love and appreciate them.

Last, but by no means least, I would like to thank my wife, Marina, for her love, care and support. She was always there cheering me up and stood by me through the good times and bad.

Abstract

Transformer reliability and stability are the key concerns. In order to increase their efficiency, an automatic monitoring and fault diagnosing of the power transformers are required. Dissolved Gas Analysis (DGA) is one of the most important tools to diagnose the condition of oil-immersed transformer. Agents technology as a new, robust and helpful technique, successfully applied for various applications. Integration of the Multi-Agent System (MAS) with knowledge base provides a robust system for various applications, such as fault diagnosis and automated actions performing, etc. For this purpose, the present study was conducted in the field of MAS based on Gaia methodology and knowledge base. The developed MAS followed by Gaia methodology represents a generic framework that is capable to manage agents executions and message delivery. Real-time data is sampled from a power transformer and saved into a database, and it is also available to the user on request. Three types of knowledge-based systems, namely the rule-based reasoning, ontology and fuzzy ontology, were applied for the MAS.

Therefore, the developed MAS is shown to be successfully applied for condition monitoring of power transformer using the real-time data. The Roger's method was used with all of the knowledge-based systems named above, and the accuracy of the results was compared and discussed. Of the knowledge-based systems studied, fuzzy ontology is found to be the best performing one in terms of results accuracy, compared to the rule-based reasoning and ontology. The application of the developed fuzzy ontology allowed to improve the accuracy by over 22%. Unlike the previous works in this field, that were not capable of dealing with the uncertainty situations, the present work based on fuzzy ontology has a clear advantage of successfully solving the problem with some degree of uncertainty. This is especially important, as the most of the real-world situations involve some uncertainty.

Overall, the work contributes the use of the knowledge base and the multi-agent system for the fault diagnosis of the power transformer, including the novel application of fuzzy ontology for dealing with the uncertain situations. The advantages of the proposed method are the ease of the upgrade, flexibility, efficient fault diagnosis and reliability. The application of the proposed technique would benefit the power system reliability, as it would result in reduction of the number of engineering experts required, lower maintenance expenses and extended lifetime of power transformer.

Contents

Illustrations

List of Figures

List of Tables

List of Symbols and Abbreviations

The following symbols and abbreviations are found throughout this thesis:

Abbreviations

ACC	Agent Communication Channel
ACL	Agent Communication Language
AI	Artificial Intelligence
AID	Agent Identity
AMS	Agent Management System
ANN	Artificial Neural Network
AOP	Agent-Oriented Programming
AP	Agent Platform
API	Application Programming Interface
BDI	Beliefs Desires Intentions
BOT	Bottom Oil Temperature
CA	Communication Act
DAU	Data Acquisition Unit
DETC	De-Energised Tap Changer
DF	Directory Facilitator
DGA	Dissolved Gas Analysis
DL	Description Logic
DNP	Distributed Network Protocol
DSS	Decision Support System

EA	Evolution Algorithms
EMS	Energy Management System
FIPA	Foundation for Intelligent Physical Agents
FOL	First Order Logic
HLIM	High-Level and Intermediate Models
HMI	Human-Machine Interface
HST	Hot Spot Temperature
HTTP	Hyper Text Transfer Protocol
IE	Interface Engine
IEC	International Electrotechnical Commission
IED	Intelligent Electronic Device
IIOP	Internet Inter-Orb Protocol
IP	Internet Protocol
IT	Information Technology
JADE	Java Agent Development framework
JDBC	Java Database Connectivity
JESS	Java Expert System Shell
KA	Knowledge Acquisition
KBS	Knowledge-Based System
KLD	Kullback-Leibler Divergence
K**NN**	K-Nearest Neighbour
KP	Knowledge Processing
KR	Knowledge Representation
LGPL	Library Gnu Public License
LTC	Load Tap Changer
MAS	Multi-Agent System
MASIF	Mobile Agent System Interoperability Facility
MILP	Mixed Integer Linear Programming
MTS	Message Transport System
NGT	National Grid Transco
OMG	Object Management Group

OWL	Ontology Web Language
PD	Partial Discharge
ppm	Parts Per Million
PR	Pattern Recognition
RDF	Resource Description Framework
RTU	Remote Terminal Unit
SAS	Substation Automation System
SCADA	Supervisory Control and Data Acquisition
SVM	Support Vector Machine
SW	Semantic Web
TOT	Top Oil Temperature
UHF	Ultra-High Frequency
UML	Unified Modelling Language
WWW	World Wide Web

Symbols

$\mathcal{A}_{CB}$	Circuit Breaker Agent
$\mathcal{A}_{CNT}$	Controller Agent
$\mathcal{A}_{DS}$	Data Sender Agent
$\mathcal{A}_{KB}$	Knowledge Base Agent
$\mathcal{A}_{ONT}$	Ontology Agent
$\mathcal{A}_{SGT1}$	Transformer Agent
$\mathcal{A}_{USER}$	User Agent

Chapter 1

Introduction

1.1 Introduction

Power systems generate, transmit and distribute electrical energy to consumers. Electric power distribution system is an important part of electrical power systems in delivery of electricity to consumers. Electric power utilities worldwide are increasingly adopting the computer aided monitoring, control and management of electric power distribution system in order to provide better services to consumers. Therefore, recent research and development are conducted in area of Information Technology (IT) and data communication for automation purpose.

Supervisory Control And Data Acquisition (SCADA) systems are powerful and successful technology, that has been applied in industrial automation. The main features of the SCADA systems are data acquisition, event processing, condition monitoring, controlling components and user interaction. In power system, it has been applied in various area, such as power generation, power transmission and power distribution. Providing real-time information from large scale distributed network, to the user in the control center, is the main application of SCADA. This has been recently carried out with the help of agent technology [1].

In computer science, an agent is a programming software, situated in some environment, that can capture real-time status of this environment, think and select appropriate actions, based on its goal and then apply with its actuators [2]. The MAS in power system find their applications in condition monitoring, control, automation, etc. [1, 3–5].

This also applies to ontology for the purpose of knowledge representation [6]. The majority of applied MAS provide the real-time information of the components to the user, and user need to decide which action to be performed.

Power transformer is an important component of the power system, whose correct and reliable functioning is vital to the system operation. Therefore, condition monitoring provides the opportunity to protect the system from serious problem, to control power transformer on-line and in an appropriate time. However, in case of large distributed network, large amount of information is collected and presented to the user. This makes engineers operate more data which requires an appropriate managing and monitoring techniques. Besides that, the power transformer has been designed for the long term operation, around 30 years. During its operation various types of stresses related to the loads, ambient temperature, operation time, etc., influence its operation and the outcome end up with losses [7]. Therefore, it is important to diagnose the fault as early as possible when the fault appears. Automatic actions can be performed even in some critical situation on behalf of the user.

This thesis is dedicated to the development of knowledge base and MAS for power transformer fault diagnosis. The developed MAS is aimed to provide a condition monitoring, perform various types of actions and fault diagnosis of the power transformer. This chapter provides an overview of the above approaches based on applying knowledge base for developed MAS. Furthermore, the motivations and objectives of the thesis are given, as well as contribution of this project. The outline of the thesis is given in Section 1.4.

1.2 Motivation

MAS has been applied for the purpose of condition monitoring of the power system. The system is designed to present the real-time information of the plant to the user for an appropriate action to be performed [3–6, 8]. Various research works have been conducted earlier in the field of power transformer fault diagnosis [9–12], however several problems still remained unsolved. Therefore, this work was motivated by finding the solutions for the following problems:

- Large amount of data obtained from each component of the power system requires efforts and knowledge of experts in various fields;
- On-line power transformer monitoring prevents a forehead diagnosis of the fault. This consequently results in energy loss and shortens the lifetime of power transformer.
- Lack of intelligence and flexibility of existing methods for power transformer fault diagnosis. The traditional methods of diagnosis usually have fixed topology, and they can not be altered easily according to the situation requirements.
- The knowledge representation applied to date is only capable of providing the real-time information of power system and not able to select an appropriate action in critical situations.

The objectives of this work aimed to find solution for the problems described above are highlighted in the following section of the thesis.

1.3 Objectives of Thesis

The objective of this work is to design a framework for power transformer fault diagnosis by utilising various types of knowledge-based systems. The framework is capable of collecting the real-time data from the component (in our case power transformer), providing the real-time status of the component, fault diagnosis and performing an automated action corresponding to the fault types. The following objectives were set during this work:

- Development of a MAS based on a formal methodology for power transformer on-line monitoring and fault diagnosis;
- Applying rule-based reasoning for MAS development for purpose of performing automatic actions;
- Applying several types of knowledge-based systems, namely rule-based reasoning, ontology, fuzzy ontology, for power transformer fault diagnosis;

The more detailed information on the conducted project structure is presented in the following section.

1.4 Thesis Overview

This thesis is structured as follows:

Chapter 1 defines the background for the research carried out as well as provides an overview of research motivation, aims and contributions.

Chapter 2 gives an overview of industrial power system automation. Agents, MAS and their applications are introduced, followed by reviewing different methodologies of MAS design. The chapter presents agent standard, languages and platforms, in particular, the elements applied in power system are highlighted. Components of power system, including the power transformer, are reviewed, followed by some traditional methods of transformer fault diagnosis. Finally, the current applications of MAS in power system are discussed.

Chapter 3 presents the developed multi-agent framework for power transformer monitoring and fault diagnosis based on Gaia methodology. The analysis part and design part of the Gaia methodology are reviewed in details. The specifications of applied agents are introduced and categorized. Three types of knowledge-based system are developed for the proposed MAS for the purpose of condition monitoring, controlling and performing automatic actions in power system components. Various type of tasks are applied, and the agent collaborations are presented. An agent called *Analyser* is developed to interact with MATLAB; it is used to load the data to MATLAB for fault diagnosis. 191 of DGA samples are tested with the KNN classification to show agent performance. Finally, an experimental system is proposed to test and perform the case of study.

Chapter 4 proposes a novel knowledge-based system for the purpose of automation, condition monitoring and fault diagnosis of power transformer. First, expert system and knowledge-based system are introduced, as well as their application. The knowledge-based system for power transformer fault diagnosis is developed in terms of rules. Roger's method as an instance of Dissolved Gas Analysis (DGA)-based fault diagnosis is applied. The performance of the proposed system is evaluated via experimental gas data

(70 DGA samples), and the results are discussed. Finally, the issues related to the knowledge representation in form of rules are discussed, followed by their advantages and disadvantages.

Chapter 5 presents the knowledge-based system in form of logic, which leads to the ontology-based knowledge representation. First, a brief introduction to the Description Logic (DL) is given followed by its applications. Such an important application of DL, as ontology, is discussed, including the languages, platforms and design status. Finally, an ontology and *Ontology* agent are developed for power transformer fault diagnosis. The summarised agent system architecture and ontology based fault diagnosis are described. Traditional fault diagnosis based on Roger's method is investigated in form of ontology. Experimental results for 70 DGA samples are presented, the conclusion about the method accuracy is presented followed by discussion.

Chapter 6 describes an overview of fuzzy system and its application in power transformer fault diagnosis. The issues of ontology-based fault diagnosis are discussed. Fuzzy ontology, including that for Protégé plug-in, are introduced. A fuzzy ontology for Roger's method is developed. The case study is applied for 70 DGA samples; the accuracy of the proposed method is discussed and compared with the other methods.

Chapter 7 draws a conclusion for the thesis, based on the outcomes of this study, followed by the discussion of the challenges of this work. Suggestions for future work are also presented in this chapter.

1.5 Contribution of Research

The main contribution of this work lays in the field of using a multi-agent and knowledge-based system for power transformer condition monitoring and fault diagnosis. The work successfully met the objectives set earlier, and introduces several novel features and applications.

- The multi-agent system for power transformer fault diagnosis followed by Gaia methodology. Gaia methodology enables the convenient design of MAS, that can be easily expanded according to the system requirements. Therefore, the flexibility of the developed MAS is increased, making it potentially suitable to be applied to the other types of equipment than power transformer.

- Use of knowledge representation to perform automated action on behalf of user. This results in lower number of expert engineers required to maintain the control, and lower negative impact of the human factor.
- MAS combined with ontology for fault diagnosis. This enables the real-time on-line fault diagnosis, thus minimising the chance of error, compared to the traditional methods currently applied in industry.
- Application of fuzzy ontology for dealing with the fault situations involving some uncertainty. One of the key problems of the ontology-based fault diagnosis is inability of dealing with the situations involving some degree of uncertainty. This work introduces the novel use of fuzzy ontology for solving this problem, and thus significantly increases the accuracy of the fault diagnosis.

Overall, this work contributes the solution of the common problems of using the power transformers and provides generic user-friendly and expandable technique of industrial fault diagnosis.

1.6 Autobibliography

List of publications produced during this work:

1. **Davoodi Samirmi, F.**; Tang, W.H.; Wu, Q.H., "Implementation of Gaia methodology for multi-agent based transformer condition monitoring", Innovative Smart Grid Technologies (ISGT Europe), *2012 3rd IEEE PES International Conference and Exhibition*, pp. 1-8, 14-17 Oct. 2012.
2. **Davoodi Samirmi, F.**; Tang, W.H.; Wu, Q.H., "Feature Selection in Power Transformer Fault Diagnosis based on Dissolved Gas Analysis", Innovative Smart Grid Technologies (ISGT Europe), *2013 4th IEEE PES International Conference and Exhibition*, in press.

3. **Davoodi Samirmi, F.**; Tang, W.H.; Wu, Q.H., "Fuzzy Ontology Reasoning for Power Transformer Fault Diagnosis", Submitted to *IET Software*, September, 2013.

4. **Davoodi Samirmi, F.**; Tang, W.H.; Wu, Q.H., "Power Transformer Fault Diagnosis with Multi-agent System based on Ontology Reasoning", 5th IEEE PES Asia-Pacific Power and Energy Engineering Conference 2013 (*IEEE PES APPEEC 2013*), in press.

Further publications based on Chapter 4 results are in preparation.

Chapter 2

Multi-Agent System and Power Transformer Condition Monitoring

2.1 Background of Power System Automation

Power systems generate, transmit and distribute electrical energy to consumers. Computers are applied for efficient monitoring, controlling and automation of the power system and its components. Three types of automation systems used in the transmission industry are itemized as follows [1]:

- Energy Management Systems (EMS): A system that is run with the aid of computers to control, monitor and optimize the performance of power system;
- Supervisory Control And Data Acquisition (SCADA) systems: Responsible for data acquisition, event process and Human-Machine Interface (HMI);
- Substation Automation Systems (SAS): Reducing the operation in the plants with the aid of components condition monitoring in a single substation.

Figure 2.1 shows the structure of power automation system. The hierarchical structure is formed by an individual SAS of each substation at the lowest level. The SCADA system is superior to the SAS, and the EMS is at the top level.

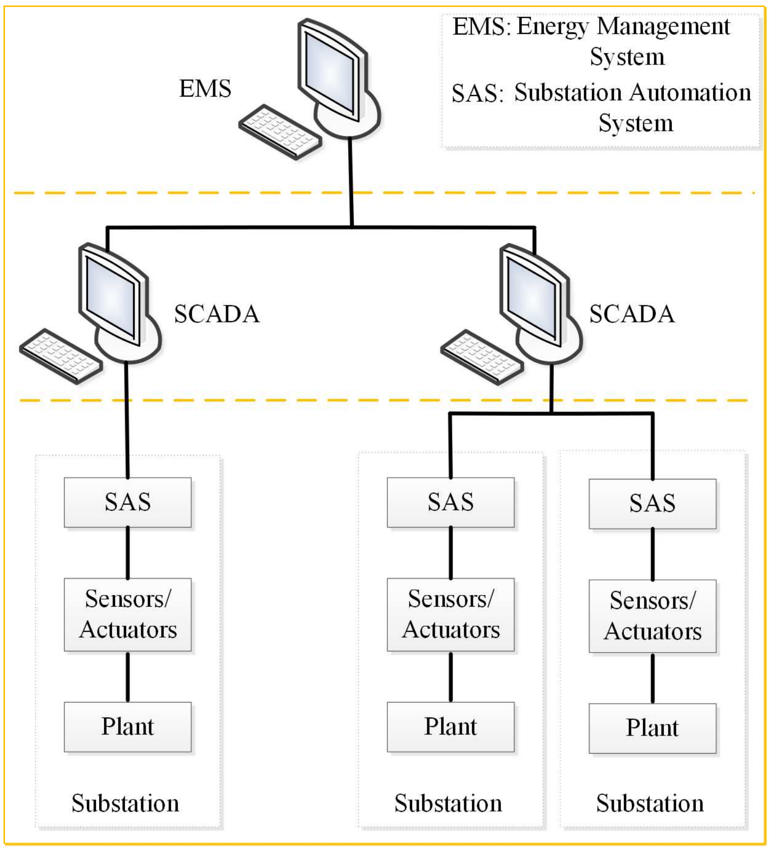


FIGURE 2.1: Power system automation

The SAS uses a number of devices which are integrated into a function package by communication technology for the purpose of monitoring and controlling the substation. In fact, the SAS components are classified into three levels: power system equipment, such as transformers and circuit breakers, is at the lowest level, Intelligent Electronic Devices (IEDs) with various application abilities are at the second level, and the utility enterprise is on the top level [13]. The IEDs are the microprocessor systems with capability of sending and receiving data, which are designed and integrated for performing various tasks, such as protection relays, load survey, operator indicating meter, etc., often via a network. The fact that they are designed by different vendors with various protocols may cause the problems in communication stage. In this case open systems, based on the use of non-proprietary, standard software and hardware interface, are recommended. Open systems use standard protocols, such as Modbus, Modbus Plus, Distributed Network Protocol 3 (DNP3), etc., to enable future upgrades from multiple suppliers at lower cost and easy integration with relative ease and low risk. Another recommendation is to apply the IEC 61850, an international standard for communications within the substation [14]. This standard improves multi-vendor interoperability and establishes high speed communication between IEDs. Moreover, it results in lower installation, extension, engineering costs and provides easier extension of functions and implementation of new functions [13].

Monitoring is a main level above the equipment control in power system automation. This level helps to monitor the state of the system, operate switching and control devices remotely, and protect the system from disturbance. At this stage various methods are applied for on-line condition monitoring in order to provide their functionalities in real-time, according to the requirements. Information is captured and monitored in control center to be further analysed by experts. One of the recent applied techniques of monitoring and controlling electric power components is multi-agent system. The following section provides a detailed description of the multi-agent system basics.

2.2 Agent and Multi-Agent Systems

Artificial Intelligence (AI) is the area of computer science that focuses on creating machines that can be engaged in behaviour that human consider to be intelligent. Reasoning, knowledge, planning, communication, perception and the ability to move, learning and manipulate objects are the central goals of AI research. A systematic approach to software engineering simplifies the process and results in a software that is understandable, verifiable and reliable. Object-oriented programming is one of the examples of this methodology approach, presented by Booch in 1982 [15]. The key element of the Object-oriented methodology is an object. An object is some real world entity that programmer wishes to model. Agent-Oriented Programming (AOP) is another type of programming paradigm with capabilities of interfaces and messaging. The internal structures are the major difference between objects and agents. Objects encapsulate methods and attributes, while agents encapsulate goals, plans, beliefs, and commitments. The main goal of an agent is to discover an autonomous entity, to perform the task autonomously. The main problem of objects is that they are passive service providers and do not possess this ability directly.

Agents and agent programming represent a new, robust and helpful technique based on computing. There is no single definition of an agent, but the most common description states that an agent is a computer system situated in an environment and it is able to act autonomously on behalf of a user or an owner [2]. Wooldridge also noted that the agent takes the information from its environment with its sensors, and decides what action to perform. The action can be performed with agent's actuators to make some changes in its environment. Figure 2.2 shows an agent with its effectors/actuators that are able to change the environment. In complex systems the agent on its own would not be able to act autonomously and control its environment, so in this case several agents can be used. These agents have different abilities and cooperate to achieve their goals. Thus multi-agent systems are used to achieve results in this case.

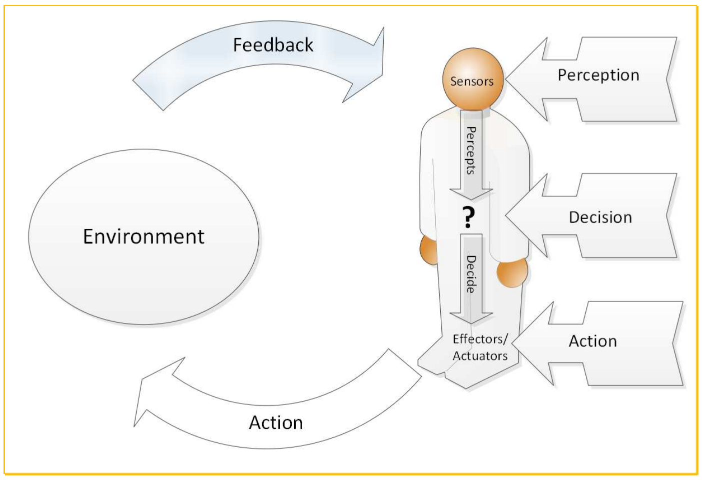


FIGURE 2.2: An agent in its environment

Agents' properties encourage researchers to apply this technology in a variety of domains. Some of these properties are itemized as follows:

- Autonomous: agent is able to control its actions without any direct human intervention;
- Reactive: agent is able to perceive environment by its sensors and respond to it by changing the environment;
- Pro-active: agent's behaviour and its responses are directed by exhibited goals;

- Social ability: communication language between agents that provides the interaction with each other.

According to the agent properties and ability to use various techniques in decision making, agents act quite similar to the human beings [2]. A standard for multi-agent technology provides a comprehensive performance in MASs collaboration.

2.3 Agent Standards and Communication

There are various standards for agent technology, such as Object Management Group (OMG) [16], Agentcities [17], Foundation for Intelligent Physical Agents (FIPA) [18], etc. The OMG Mobile Agent System Interoperability Facility (MASIF) was formed in 1989. It aims to satisfy a demand in a component-based software marketplace, which followed an introduction of standardized object software. Thus, the organization is responsible for establishing the industry guidelines and setting detailed object management specifications, which results in creating the common framework for application development.

FIPA is a collection of standards related to agent software; it was established in 1996. It aims to standardize the interoperation of heterogeneous software agents. The details of FIPA specification are given in the following subsection. A new initiative standard is called Agentcities, which aims to build a worldwide, publicly accessible test bed for the deployment of FIPA agent-based services.

2.3.1 FIPA standard for agent development

FIPA is a standards organization accepted by the IEEE in 2005 [18]. It promotes agent technology and the interoperability of its standards with the other technologies. The core principle of this standard is to provide a new paradigm for solving old and new problems followed by standardization mechanics. The FIPA specification consists of four main parts: abstract architecture, agent communication, agent management and agent message transport, as shown in Figure 2.3.

Abstract architecture is aimed to deal with the abstract entities for developing agent services and its environment. Semantic meaning message exchange between agents, including multiple message transport management and defining agent and servers location by using the directory services, are the key points of this architecture.

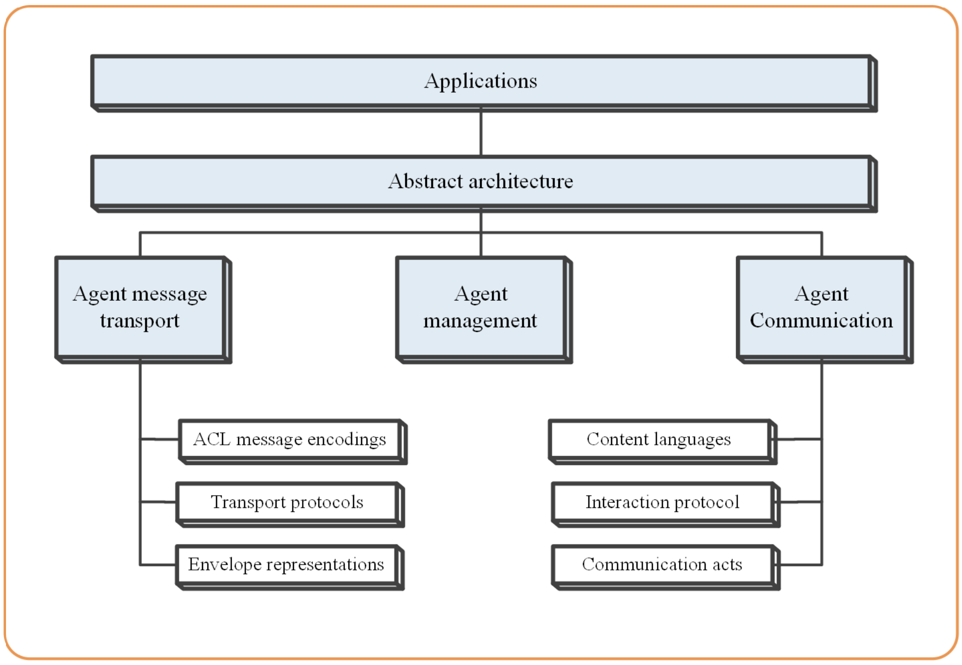


Figure 2.3: FIPA standard

A set of interaction protocols was defined by FIPA for coordinating the multi-message actions. Agent Communication Language (ACL) messages, communication acts, exchange messages protocols and language for representing the content, are the main specifications of agent communication. An agent must be able to receive any FIPA-ACL communicative act message; in the case if the message cannot be processed, the agent should respond with a “not-understood” message. More details of the agent communication will be given in the following section.

Agent management deals with operating and establishing the logical reference model. The logical reference model is applied for creating agents, migration, location, communication, registration and retirement [18]. The components of agent management are presented in Figure 2.4. The scope of these components includes the following items:

- An Agent Platform (AP);
- An Agent;

- A Directory Facilitator (DF);
- Agent Management System (AMS);
- A Message Transport Service (MTS).

The AP provides a physical infrastructure to deploy agents. Several APs are developed for agent technology, such as JADE, Jason, BDI, JACK, etc. An agent is a form of distributed code process, which inhabits in the AP and offers computational services. Agent Identity (AID) is the agent identification distinguished unambiguously. DF provides a yellow pages service, that can be accessible to the other agents. Agents publish their most current services in DF, so the other agent can easily find them. In AP, Agent Management System (AMS) is a necessary agent responsible for managing their operation, that also provides white pages service. Agents are able to de-register from AMS or register in it, which can be available to other agents. ACL messages are transported between agents by Message Transport Services (MTS). On any given AP, the MTS is provided by an Agent Communication Channel (ACC). Hypertext Transfer Protocol (HTTP) and Internet Inter-Orb Protocol (IIOP) are two transport protocols specified by FIPA Message Transport Protocol.

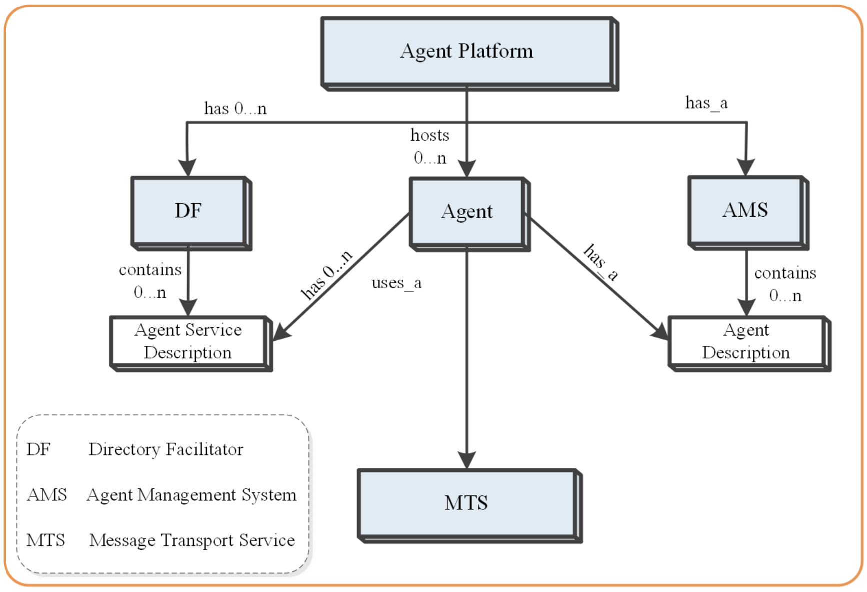


FIGURE 2.4: Depiction of the agent management ontology

2.3.2 Communication languages for agent development and platforms

FIPA communication language history started from creation of ARCOL by Sadex in 1991 from France Telecom, which soon after became known as FIPA-ACL or just ACL [19]. For the purpose of message content expression and dealing with several cooperation protocols, France Telecom recommends to use Semantic Language (SL). The main points of any communication act are the facts that the message is performative, includes sender, receiver and content. Several parameters of FIPA-ACL messages are given in Table 2.1 [18]. Sub-layers of the FIPA communication are detailed below:

- Transport: IIOP, WAP and HTTP are the message transports defined by FIPA;
- Encoding: messages can be represented in high level, such as String, XML and Bit-Efficient;
- Message: independent message;
- Ontology: can be used for expression of message content;
- Content expression: provides different types of expression, such as logic, algebra, etc., e.g. FIPA-SL;
- Communication Act (CA): performing or acting messages, e.g. request and inform;
- Interaction Protocol: used for message exchange, e.g. agree or refuse to the request message.

As mentioned previously, FIPA represents the CA standard for action purpose. Some set of CA performatives are “Inform”, “Inform If”, “Agree”, “Refuse”, etc. An example of FIPA-ACL message with an “Inform” performative is given below. An agent called *TransformerOilSensor* sends a message containing three gas ratios, to the receiver agent called *DataCollector*.

```
( INFORM
   :sender (agent-identifier
        :name TransformerOilSensor)
   :receiver (set (agent-identifier
                :name DataCollector))
        :content ‘‘(( set (Ratio1 0.00815)
                          (Ratio2 6.7)
                           (Ratio3 6.9)))’’
        :language FIPA-SL
        :ontology PartPerMillion
        :protocol fipa-request
    )
```

TABLE 2.1: FIPA-ACL message parameters

Parameter	Description
Performative	Type of communicative acts
sender	Identity of the sender of the message
receiver	Identity of the intended recipients of the message
reply-to	Participant in communication
content	Content of the message
language	Language content are expressed
encoding	Description of content
ontology	Description of content
protocol	Conversation of interaction protocol
conversation-id	Control of conversation
reply-with	Control of conversation
in-reply-to	Control of conversation
reply-by	Control of conversation

The logic of mental attitudes and actions in CAs is based on first-order modal languages, which represent the intentional semantics for FIPA-SL. There are three subclasses of SL (SL0, SL1, SL2) extended by FIPA to support various operations. The message content expression in FIPA-SL can be used in three cases:

- An action, for performing action. It is used as a content expression when the act is requested, and other CAs are derived from it;
- A proposition, for assigning a truth value. It is used in the “Inform” CA, and other CAs are derived from it;

- An identifying reference expression, for identification of object in the domain. It is used in the inform-ref macro act, and other CAs are derived from it.

Clarification given above may be described using an example of two agents, A and B, that make use of the *iota* operator. The *iota* operator is a constructor for giving an expression. In this case, agent A has the following knowledge base: KB=P(A), Q(1,A), Q(1,B). The interaction between agents A and B is:

```
( QUERY-REF
   :sender (agent-identifier
        :name B)
   :receiver (set (agent-identifier
        :name A))
   :content ``((iota ?x (p ?x)))''
   :language FIPA-SL
   :reply-with query1)
)


( INFORM
   :sender (agent-identifier
        :name A)
   :receiver (set (agent-identifier
        :name B))
   :content ``((= (iota ?x (p ?x)) alpha))''
   :language FIPA-SL
   :in-reply-to query1)
)
```

The expression (*iota* x (p x)), where x is term and (p x) is a formula, can be read as “the x such that p [is true] of x”. The query-ref message is replied with alpha which is the only object can satisfy the proposition p(x) [20].

In terms of a series of messages among agents, interaction protocol is given by FIPA which allows an agent, called *Initiator*, to request other agent, called *Participant*, to perform an action. The *Participant* processes the request and decides whether the request should be accepted or refused. The interaction protocol *Initiator* and *Participant* is shown in Figure 2.5.

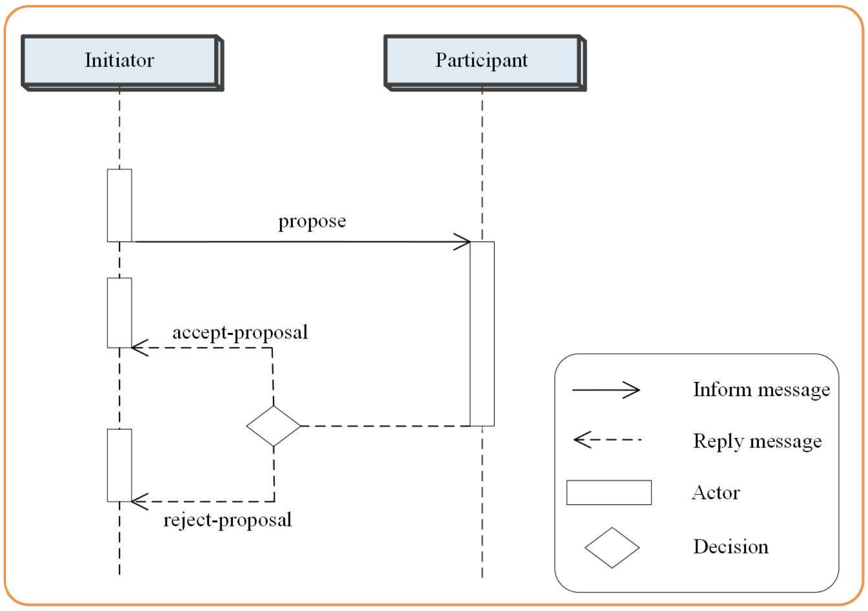


FIGURE 2.5: The FIPA Interaction Protocol

Java Agent Development framework (JADE), developed by the University of Parma, is one of the agent platforms compliant with FIPA standard [20]. First JADE platform was developed in late 1998, by Telecom Italy under LGPL (Library Gnu Public License). The key idea of JADE is in implementation of an abstraction over a well-known Java object-oriented language. Thus, JADE programmers must develop their agents in full Java programming. In JADE platform agents live in containers, that provide the JADE run-time and all the services needed for hosting and executing agents. According to FIPA standards requirement, JADE platform utilizes the complete agent management specification, including the key services of ACC, AMS, MTS and DF. Main container is the first launch container with hosting for two main agents, AMS and DF. AMS supervises the entire platform and provides white pages service, while the DF agent implements yellow pages service. Furthermore, JADE platform provides a graphical user interface with various tools for helping developers in their design.

2.3.3 Agent architectures

Various definitions of agents' architectures are provided by researchers. One definition provided by Michael Luck et al. [21] is that:

"Architectures provide information about essential data structures, relationships between these data structures, the process or functions that operate on these data structures, and the operation or execution cycle of an agent."

The notion of agent's architecture is the way how agents work together to achieve the complex tasks by using different paradigms. Wooldridge and Jennings classified the agent's architecture into three categories [22] as follows:

- Deliberative/symbolic reasoning architectures;
- Reactive architectures;
- Hybrid architectures.

The basic idea of symbolic reasoning architecture is that the symbols represent the environment. In fact, all the information of the environment is written in form of symbols. In this case the main point is the way to describe the current state of this environment. Therefore, these types of architecture faced two key problems; the first problem is the difficulty to translate the environment into the symbols. The second problem is a difficulty of the real world representation in a complex. Several models have been provided by researchers to overcome these problems [2].

Reactive architecture, developed by Brooks and Maes [22], is another type of agent's architecture. In this architecture there is no symbolic or logical model of environment. The key idea of this architecture is that an intelligent behaviour can be generated without explicit representations and abstract reasoning provided by symbolic artificial intelligence techniques; intelligence is an emergent property of certain complex systems. Subsumption architecture was developed by Brooks in 1991; it states that the sensors can transmit real-time information into the layers of finite state machines. Layers with lower level have less control than the higher level of the stack in a hierarchy of behaviour, so decision-making is achieved through goal-directed behaviour. The advantage of this architecture is faster response, however it only works for certain environments [2].

Hybrid architecture consists of reactive and deliberative architectures, which act according to their needs. The reactive architecture is useful in situations when the agents ability to react on time is required. For the situations that require agent's ability to act reasonably, deliberative/symbolic architecture can be useful. Touring Machine is one type of hybrid architecture provided by Ferguson [23]. The main point of Touring Machine is to coordinate and control the actions of autonomous agents situated in dynamic worlds.

2.4 Agent Design Methodologies

The key point of designing MAS is to define the agents cooperation. Usually methodology covers the whole life-cycle of system development, such as analysis, design, implementation and validation. In analysis step, agents are associated with the entities of the analysed scenarios. The capabilities and responsibilities of each agent are identified. Finally, the interactions between agents are applied [24].

Different types of methodologies have been introduced by researchers for various applications. Some of the proposed methodologies are: High-Level and Intermediate Models for Agent-oriented Methodology, MASE and Gaia methodology, etc. Based on their abilities, each methodology has some advantages and disadvantages. There is a framework to compare these methodologies and their suitability for specific applications [25].

2.4.1 High-Level and Intermediate Models (HLIM) methodology

A general-purpose multi-agent methodology model is called HLIM [26]. This method represents the development of agent systems through a series of abstraction levels where human, with machine assistance, can manipulate abstractions at one level into abstractions at the next lower level. Overall, the methodology has two phases, namely, discovery phase and definition phase. The discovery phase guides the discovery of agents and their high-level behaviour, while the definition phase produces implementable definitions. The HLIM contains five models, as itemized below:

- A high-level model;

- An internal agent model for the internal structure of agents;
- A relationship model;
- A conversational model;
- A contract model.

A high-level model provides the high-level view of the system and initial point for development. The application scenarios are tracked to describe functional behaviour, discovering agents and behaviour patterns along the way. Goals, plans and beliefs aspects of agents are described by internal model, which are derived directly from the high-level model. The relationship model describes agent relationships, such as dependency and jurisdiction. The coordination among the agents is described by conversational model. The structure of agents commitments are defined by the contract model. Contracts are created during the agents instantiation or execution as required. This methodology has a lack a detailed description of the implementation and testing phases [25].

2.4.2 MaSE methodology

MaSE methodology focused mainly on robotics; it captures goals and continues through the conceptual phase to design the system [27]. Number of graphically-based models are used to describe the agent types and their interface with the other agents, as well as an architecture-independent definition of the internal agent design. The MaSE methodology consists of analysis and design parts. The analysis part provides an overview of required system, while the design part models it into useful construction for implementing the MAS.

The analysis part is captured in three steps: capturing goals, applying use cases, and refining roles. In first step the users requirements are taken, goals are captured and turned to the system level goals. From this level, the use cases and sequence charts are extracted to initialize the set of roles and their communications paths. Finally, the third step is to refine and extend the initial set of roles and define tasks to accomplish each goal in the refining roles step. Creating agent classes, constructing conversations, assembling agent classes and system design are four steps of the design stage, that are captured from analysis stage.

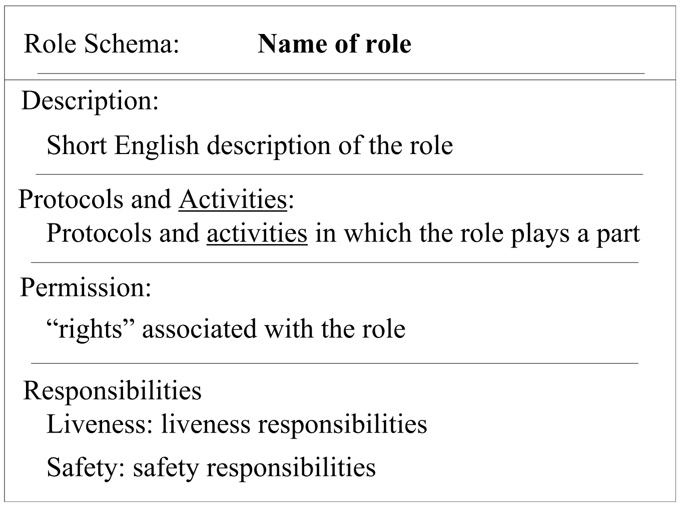
In MaSE methodology, communication between agents is applied through Finite S-tate Machines which leads to an algebraic description of conversations. For this reason the method is considered to be successful and acceptable in describing interactions between agents [25]. On the other hand, it does not support adaptability and mental attitudes, such as Beliefs Desires Intentions (BDI).

2.4.3 Gaia methodology

The Gaia methodology [28] is anticipated to facilitate an analyst to gradually shift from an initial ambiguous state to a more concise and methodical design which can be implemented directly. According to the Gaia methodology, the process of building MAS is similar to a process of organisational design. Hence, the relationship and interaction between hierarchical roles in one organisation would be defined as the analysis category of this Gaia methodology, which, in turn, consists of two other sub-categories, namely the role model and interaction model. For instance, the role model in this case consists of a role called *Data_Collector*, and its responsibility is to collect data from any messages received from devices or equipment in a plant and store them in appropriate databases and tables.

According to Gaia, each role has different permissions and responsibilities related to their tasks. Each role should be responsible of its tasks to be completed in a correct way. Permission defines information resources that are allocated to each role. These resources can be the knowledge or information related to the agent. Therefore, some resources can be carried out by a role, identified as liveness. On the other hand, sometimes an agent does not change the existing condition. It is required to preserve it for some purpose, and this is defined as safety.

In the case when roles should complete their tasks according to their responsibilities, the roles may use some other activities in their own way to improve the task. This is identified as activities. The way the role interacts with other roles is called role's protocols. The role model template is provided in Figure 2.6. The second part of analysis is to find a link between roles defined as the interaction model. This model consists of a set of protocols' definitions, including such attributes, as its purpose, the initiator, responder, input, output information and processing.


Role Schema: **Name of role**

Description:
Short English description of the role

Protocols and Activities:
Protocols and activities in which the role plays a part

Permission:
"rights" associated with the role

Responsibilities
Liveness: liveness responsibilities
Safety: safety responsibilities

FIGURE 2.6: Template for MAS role schema

The analysis model is transformed into an implementation model during a design phase. Gaia identifies three models for the design part. The first model identifies the agent types in the system by mapping one to one from the role model in the analysis part. The second model, called services model, shows the main services required to assign agents' roles. Finally the lines of agents' communication are documented in the acquaintance model. More detailed explanation of Gaia methodology is provided in Section 3.2.

As the agent-based programming is becoming more and more popular in recent years, the multi-agent systems now find their application in various areas; this will be discussed in details in following section.

2.5 Agent Applications

Agent technologies have been successfully used in many industrial applications. The first applications of MAS appeared in 1980 and expanded to various areas. The main applications of MAS are the electronic business [29], monitoring [30] and control [31] [32], information management [33], automation intelligence behaviour [34] and so on.

One of industrial applications of MAS is energy management, a process of monitoring and controlling the cycle of generation, transportation and distribution of energy to industrial and domestic customers. The operators' work efficiency in critical situations can be improved by applying a set of decision support systems (DSS). ARCHON a decentralised software platform that provides technology for connecting DSS with each other, thus extending their use. ARCHON helps the components work together, offers them control and a level of integration. Thus, the agent consists of an ARCHON layer and application program [35] [8].

Extensibility and flexibility are the key advantages of MAS; it had been applied in many power engineering fields. Various functionalities with different abilities are able to be implemented by MAS by designing appropriate agent. In power systems MAS is described to find application in automation [4] [5], diagnosis [36], monitoring [37] and control [38], which helps the asset management of power system. These applications recommend various frameworks with different abilities, to represent the real-time information to the users for the purpose of decision making.

2.6 Electric Power Systems and Their Components

Electricity can be generated in several ways, such as hydroelectric, nuclear, solar, wind, etc. Transmission lines are required to transmit the power from the plant to the substations. The key point of substation is to change the voltage level and perform several other important operations, such as protection, control, etc. These operations can be improved by on-line monitoring and control of the components, which may consequently results in the maintenance's cost reduction. Substations also can be interconnected to create the electricity transmission networks. The central nerve of power system is the control center which senses the pulse of the power system, adjusts its condition, coordinates its movement, and provides defence against exogenous events [39].

The main components of substation are transformers, switch-gears and other items of plant. An electric circuit can be protected from short-circuit with automatically-operated switches, called circuit breakers. Relays are another type of switches; they are used to control, isolate and protect the power system from high current flow. Transformer is a static electric device, the most expensive component of substation. The transformers may differ significantly in terms of size and application. Such an expensive equipment as power transformer is always in a high risk of damage in high voltage working state. In the UK there are over 5000 substations; 377 substations at 275 kV or

400 kV, and 4849 substations at 132 kV and below; they are operated by National Grid Transco (NGT) [40].

2.6.1 Power transformer

As mentioned earlier, power transformer is one of the most important parts of electric power system; it converts voltage to higher or lower levels. Depending on size, they can be divided into three groups, as follows [41]:

- Small power transformers, 500 to 7500 kVA;
- Medium power transformers, 7500 to 100 MVA;
- Large power transformers, 100 MVA and above.

Operated within the ratings, the lifetime of power transformer is expected to be about 30 years of operation, however operating beyond its rating values may lead to a significant life shortening. On the other hand, all industries require a reliable and correctly operated power system at all times. Failure of power transformer would result in power supply loss to the industry, supplied by these transformers.

Heat is one of the most common transformer destroyers. During transformer operation, a temperature increase by 10 °C above its rating will result in shorter life time. Depending on the transformer, various classes of cooling systems are applied. In dry type transformers air forced devices (fans) are implemented to reduce the temperature. Oil-immersed transformer is another type of transformer which uses oil-forced pumps for reducing the temperature.

Mathematical modelling of the power transformers is widely used for research of electromagnetic transient processes and can be simulated in a range of computer programs. These mathematical models are intended to describe work of power transformer using mathematical equations, and can be used by researchers to solve the problems arising in actual power transformers [42, 43]. The mathematical modelling of power transformer can be quite complex process, as it is require representation of equivalent circuits with the help of modern computer programs. In particular, thermal model [11] is capable of predicting the transformer temperature and evaluation suggestions. This model can be used by engineers for better understanding of transformer working conditions. Another important feature of the thermal model is the transformer load ability analysis that allows an engineer to simulate the possible conditions before switching loading between transformers [11].

The basic construction parts of power transformers are essential throughout the industry to a certain degree. Some these types, such as core, winding, taps-turns ratio adjustment are described below, as well as accessory equipment.

Core

Magnetic field path in power transformer is guided by core, it consists of thin strips of high-grade steel with thin coating insulation. Depending on the transformer rating, the type of steel and the core size can be varied. During working state of power transformer, heat is generated in the core and must be transferred. One way of cooling the core down is to immerse it in the tank of oil. Oil temperature can vary depending on position of sensors in the tank. The Top Oil Temperature (TOT) and Bottom Oil Temperature (BOT) are the examples of the temperature in various parts of the tank. In larger units, cooling ducts are used inside the core for reducing heat and avoiding the hotspots.

Winding

Windings are the conductors for carrying the current; they are made of aluminium or copper materials by wounding around the section of the core. Their types depend on the transformer rating, as well as the core construction. In the case of high electrical and mechanical stress in power transformer, the winding is made of disc coils. The flow of liquid (oil) through the windings can reduce the heat during of transformer operation time.

The taps-turns ratio adjustment

Voltage in secondary winding of transformer can be adjusted by the ratio of the number of turns in primary. It is possible to be operated manually and/or automatically. In terms of manual operation, De-Energized Tap Changer (DETC) switches mechanism and provides the external accessibility to change the tap position. There is an on-line capability of monitoring and changing the ratio of transformer, referred to Load Tap Changing [44] [45].

Accessories

There are many types of accessories used for monitoring and protecting power transformer. Some of accessories are: liquid level indicator, pressure relief devices, liquid temperature indicator, winding temperature indicator, sudden pressure relay, etc. For instance, the liquid level indicator in oil-immersed transformer is used to indicate and control the oil level.

These parameters of components can be captured every minute to present the power transformer status. Data is collected by Remote Terminal Units (RTUs) situated in substations [46]. In order to increase the reliability and efficiency of component, an on-line monitoring is required.

2.6.2 Power transformer on-line monitoring

On-line monitoring of the power transformer is a process of accessing the data while transformer is operational. The characteristics of component on-line monitoring can be varied, it depends on the number of parameters monitored and the accessibility of the data required. For an on-line monitoring system normally data, reports and alarms are recorded periodically. The following major components are required in order to monitor the devices and equipment functions of power station: [41].

- Sensors: capture information or data about the equipment;
- Data Acquisition Units (DAU): measure and collect signals from different sensors;
- Communications Line between DAU and Computer: various types of communication networks to transfer data from DAU and sensors to the control room;
- Computer: software platform for monitoring of the components and communication with controlling facilities.

On-line monitoring significantly improves the efficiency of operation and maintenance procedures of power transformer [47]. The characteristics of on-line monitoring process can vary; they depend on the number of parameters monitored and the accessibility of the data required. There are over ten parameters that can be monitored to prevent the acceleration of deterioration processes during long term operation [48].

Power transformer components status can be captured with various types of electronic sensors, including Ultra-High Frequency (UHF) ones [10].

The IEEE Guide [49] covers such aspects of on-line monitoring, as the monitoring systems and their equipment, various configurations of the system with their benefits and application. The key parameters of on-line monitoring of power transformer are itemized as follows:

- Dissolved Gas in Oil Analysis: It is one of the most efficient diagnostic tools for problems determination in transformer operation. Overheating, partial discharge or local breakdown cause the presence of several gases dissolved in oil. Thus, the identification of these gases presence helps to indicate the fault.
- Moisture in Oil: The presence of moisture in oil causes decrease in dielectric strength followed by reduction of the insulation strength. It is important to measure the moisture level before any failure occurs, that can be done by regularly taking oil samples.
- Partial Discharge (PD): PD is a type of fault in power transformer, that often occurs in the case of dielectric breakdown. The level of PD can be measured with various methods, such as electrical and acoustical, providing information about the changes in power transformer.
- Oil temperature: Oil temperature is one of the key parameters in the overall temperature conditions of power transformer, which includes ambient temperature, top oil temperature, fan operation and load. These factors are important to determine the condition of transformer during its operation.
- Winding temperature: One of the limiting factors for the loading capability is hot-spot temperature of the winding. The mechanical strength of paper insulation in power transformer can be reduced by prolonged exposure to excessive heat.
- Load current and voltage: Automatic tracking of load current and voltage of power transformer will increase their life-time, by restricting their maximum load.

- Insulation power factor: All electrical insulation has a measurable quantity of dielectric loss, regardless of condition. Chemical substances and moisture may increase losses, more then normal stage.

- Pump/Fan operation: Fan operation is designed to control the temperature of transformer under various conditions. Its abnormal operation may cause failure of the cooling system. Fans and pumps status can be captured by measuring their current drown and its correlation with the measured temperature.

- Load Tap Changer (LTC) operation: LTD failures are either mechanical or electrical in nature. Failures can be caused by poor design or misalignment of the contacts, high loads, excessive number of tap changers etc. Various parameters of LTD, such as initial peak torque or current, average torque, motor current index etc., can be monitored to avoid its failures.

Thus on-line monitoring systems provide detailed information about the power transformer components and help to minimise the probability of an unexpected outage.

There are some requirements applied to transformer on-line monitoring system functionality:

- Cost efficiency: The installation and maintenance cost of on-line monitoring system should be balanced with the benefit of having this system installed.

- Long-time operation: The reliability of the monitoring system should be maintained for the lifetime of power transformer (30 years and over).

- Selectivity: The on-line monitoring system should record the parameters that could be used for the interpretation of the power transformer current condition.

- Data accessibility: All the data recorded in the monitoring process should be easily accessible by the user, and stored for the sufficient period of time.

However, the monitored data can not be used without their correct interpretation. Thus, the next step after monitoring is a fault diagnosis which provides interpretation of on-line captured data.

2.6.3 Power transformer fault diagnosis based on gas analysis

Problems in transformer can arise from defects/deficiencies and develop into incipient faults of deterioration processes. Increasing temperature, moisture, oxygen and other contaminants during transformer operation can significantly contribute in insulation degradation. According to [50], the typical failures due to high voltage in transformer are shown in Figure 2.7. As shown, the bushings and the windings are at the highest risk of failure, because these regions operate under the highest electrical filed. In this case, fault diagnosis techniques are required to identify the failure component of power transformers. Various diagnosis methods, e.g. chemical, electrical, thermal, optical and mechanical can be applied on-line and/or off-line to diagnose the transformer failure. These methods can be applied in terms of various techniques, such as artificial intelligence [51], fuzzy logic [52], machine learning, such as Support Vector Machine (SVM) [53], etc. For instance, in AI techniques, data from specific tool (like gas samples) are collected in the expert system to facilitate the decision making, in terms of fault detection. For the case of fault diagnosis based on temperature data, thermal model with capability of predicting transformers' temperature at different locations is applied [11].

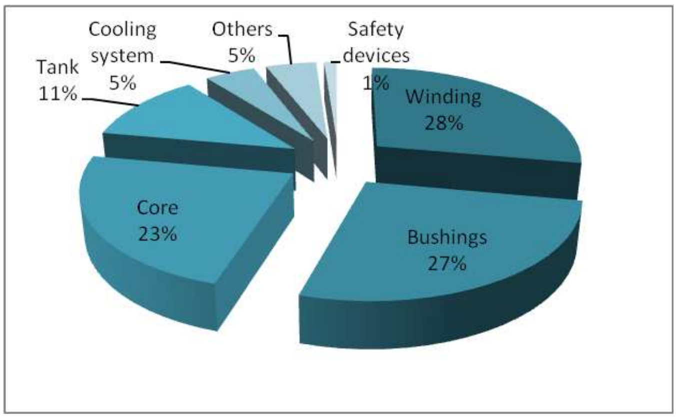


FIGURE 2.7: High voltage transformer failure distribution

DGA is a common diagnostic technique that had been used for several decades to diagnose the condition of oil-immersed transformers. Oil samples are taken from a transformer and sent for fault diagnosis, such as hot-spots, overheating, partial discharge, arcing, etc. Hydrocarbon fragments and hydrogen can be formed as a result of mineral oil hydrocarbon molecules decomposition under electrical and thermal stress. Key gases, such as acetylene (C_2H_2), methane (CH_4), hydrogen (H_2), ethylene (C_2H_4), ethane (C_2H_6), carbon mono oxide (CO), etc., may be formed by combination of fragments. For instance, overheating fault may generate the following gases:

$H_2 \Rightarrow CH_4 \Rightarrow C_2H_6 \Rightarrow C_2H_4 \Rightarrow C_2H_2$.

DGA fault diagnosis is based on key gases and/or gas ratios (such as $R_1 = CH_4/H_2$, $R_2 = C_2H_2/C_2H_4$, $R_3 = C_2H_2/CH_4$, $R_4 = C_2H_6/C_2H_2$, $R_5 = C_2H_4/C_2H_6$) by applying different methods for fault detection. Each method uses different gas ratios (or key gases) for fault classifications. Obviously, more categories of classification give better diagnosis results. In power transformer status evaluation, the rapid increase of key gases should be paid more attention, rather than the total amount of gas. However, the acetylene makes an exception, as any amount of it over few part per million (ppm) is generated as a result of the high energy arcing, while trace amount (several ppm) can be a result of thermal fault over 500 °C. DGA based fault diagnosis methods in power transformer, including such methods as Duval Triangle, Doernenburg, IEC ratio, Roger's ratios, are briefly summarised below [54].

Doernenburg method

The Doernenburg method [55] utilizes four types of gas ratios, $R_1 = CH_4/H_2$, $R_2 = C_2H_2/C_2H_4$, $R_3 = C_2H_2/CH_4$, $R_4 = C_2H_6/C_2H_2$ to diagnose three general fault types, such as thermal fault, partial discharge and arcing. At first, the values of gases are compared with the given table, to ascertain whether there is a problem with the unit. In the case of existing problem, the ratios are applied in order to obtain the suggested faults, based on Table 2.2.

TABLE 2.2: Diagnosis with Doernenburg key gas ratios method

Fault suggestion	R_1	R_1	R_2	R_2	R_3	R_3	R_4	R_4
	Oil	Gas space	Oil	Gas space	Oil	Gas space	Oil	Gas space
Thermal Fault	>1.0	>0.1	<0.75	<1.0	<0.3	<0.1	>0.4	>0.2
Partial discharge (low-intensity)	<0.1	<0.01	NS*	NS*	<0.3	<0.1	>0.4	>0.2
Arcing (high-intensity)	>0.1 to <1.0	>0.01 to <0.1	>0.75	>1.0	>0.3	>0.1	<0.4	<0.2

* NS = Not Significant

This method is a part of IEEE Standard C57.104-2008 [55], based on thermal degradation principles. In the case of missing gas value and gas ratios, the implementation of this method may result in significant number of “no interpretation” or undefined faults.

Roger’s and IEC ratios

Roger’s ratio method [56] of DGA fault diagnosis is one of the tools to be used for analysing at gases dissolved in transformer oil. Roger’s method is based on the earlier work of Doernenburg, but unlike it uses four key gas ratios. Later on, International Electrotechnical Commission (IEC) introduced the fault diagnosis based on Roger’s method using only three gas ratios. According to Roger’s method, the quantities of different gases are compared by dividing them one to the other and representing the result as a ratio of the key gases. Thus, the Roger’s method assumes that the certain gas ratio indicates that the specific temperature has been reached. The presence of certain faults was proven by comparing a large number of power transformers with similar gas ratios and examining the data according to Roger’s method. However, Roger’s ratio method is only an additional technique in analysing the problems in power transformer. Like the other ratio methods, it is only valid in the case of significant gas presence. The fault classification according to IEC ratio method is shown in Table 2.3. The faults are classified into eight types, allowing to make an assumption on the fault of the power transformer.

Table 2.3: Diagnosis of IEC ratio method

Case	Fault Type	R_2	R_1	R_5
0	No fault	$R_2 < 0.1$	$0.1 \leq R_1 \leq 1$	$R_5 \leq 1$
1	Low energy partial discharge	$0.1 \leq R_2 \leq 3$	$R_1 < 0.1$	$R_5 \leq 1$
2	High energy partial discharge	$0.1 \leq R_2 \leq 3$	$R_1 < 0.1$	$R_5 \leq 1$
3	Low energy discharge, sparking, arcing	$0.1 \leq R_2$	$0.1 \leq R_1 \leq 1$	$1 \leq R_5$
4	High energy discharges, arcing	$0.1 \leq R_2 \leq 3$	$0.1 \leq R_1 \leq 1$	$3 < R_5$
5	Thermal fault less than 150 °C	$R_2 < 0.1$	$0.1 \leq R_1 \leq 1$	$1 \leq R_5 \leq 3$
6	Thermal fault temperature range 150-300 °C	$R_2 < 0.1$	$1 \leq R_1$	$R_5 \leq 1$
7	Thermal fault temperature range 300-700 °C	$R_2 < 0.1$	$1 \leq R_1$	$1 \leq R_5 \leq 3$
8	Thermal fault temperature range over 700 °C	$R_2 < 0.1$	$1 \leq R_1$	$3 < R_5$

For instance, assume the gas dissolved in the oil sample are (in ppm): $C_2H_2 = 2$, $CH_4 = 170$, $H_2 = 26$, $C_2H_4 = 25$, $C_2H_6 = 278$, $CO = 92$ and $CO_2 = 3125$. The ratios can be calculated as follows:

$R_1 = CH_4/H_2 = 170/26 = 6.54$

$R_2 = C_2H_2/C_2H_4 = 2/25 = 0.08$

$R_5 = C_2H_4/C_2H_6 = 25/278 = 0.09$

By referring to Table 2.3, the gas ratios indicate that transformer has a thermal fault in the temperature range of 150 °C to 300 °C, which is case 6.

Duval triangle

Duval Triangle method [57] uses three gas values (CH_4,C_2H_4, C_2H_2) and their location in a triangular map to diagnose the fault. The triangle graphical method is used to visualize the different cases and facilitate their comparison. Faults are categorised into seven types, as given below:

- PD = Partial Discharge;
- T1 = Thermal Fault Less than 300 °C;
- T2 = Thermal Fault between 300 °C and 700 °C;
- T3 = Thermal Fault greater than 700 °C;
- D1 = Low Energy Discharge (Sparking);
- D2 = High Energy Discharge (Arcing);
- DT = Mix of Thermal and Electrical Faults;

These types of faults can be identified by visual inspection of the equipment after the fault has occurred in service. The Duval Triangle with its fault's region is shown in Figure 2.8. As can be seen in the Figure 2.8, there is no region of the triangle designated to the normal ageing.

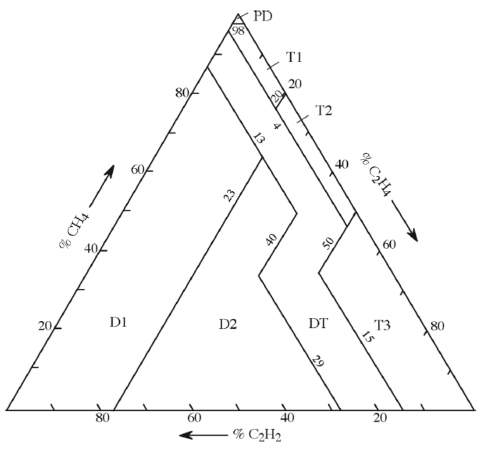


Figure 2.8: The Duval triangle

2.7 Current Applications of MAS in Power System

There are various techniques have been applied for monitoring and diagnosis of power transformers, but the complexity of their structure and the limitations of each technique force the researchers to look for new methods with better outcome. For instance, MAS is used for transformer condition monitoring based on the data of PD activity captured and measured by UHF [10]. Then captured data are analysed and diagnosed to be available to the engineers only.

A MAS with ontology has been used for power system automation in [3, 6]. The system has the ability to take a user's order from a user interface console to perform certain actions. The real-time status of equipment is captured by an ontology agent. For instance, if a user requests to open or close a specific circuit breaker, then the real-time status (open or closed) of the circuit breaker is provided by an ontology agent. The system is only able to provide the real-time information of a component to a user, and user is the one who performs specific actions. In terms of complex distributed network system with more components, more experts are required. The MAS only monitors the components of the system and does not diagnose the system faults.

Ontology-based fault diagnosis was applied for the power transformer in [9, 12]. The proposed system in [9] consists of classes and subclasses, based on category of fault types. The ontology was only able to derive the subclasses or individuals, as defined in the time of building ontology. The ontology in the system is built for fault diagnosis with four subclasses of partial discharge, thermal fault, frequency response and dissolved gas analysis. For instance, in the built ontology there is a class called “Fault phenomenon” with subclasses of different gases ratios. The ontology could get gas ratios through the user interface, and by matching it with the ontology fault diagnosis, it can capture the related fault and report it to the expert users. The system is not able to deduce the new information or undefined subclasses, and only can provide the predefined knowledge. Furthermore, the fault diagnosis method is able to provide the fault types to the user, and there is no automatic action and agent system were performed in proposed system. The ontologies proposed in [12] did not considered any situations that involve some degree of uncertainty. However uncertain situations are a common problem in real-world fault diagnosis.

The agent-based system is also applied in substation power system for the purpose of monitoring and controlling its components [1, 10, 37]. The outcome of the system is similar to the SCADA system, which is able to present the information of the substation to the user situated in higher level. Based on monitored component and agents who provide services, the user is able to request performing some actions towards the agent system. However, none of the examples described above includes dealing with situations involving an uncertainty. This feature is essential for the real-world situations, as the majority of them include some uncertainty.

According to the agent definition and its properties for providing automation in the system, the most of the proposed systems are only able to provide the information to the user, and user can inform the relevant agent to perform the necessary action. In this case, the decision maker is only user, who needs to decide which actions need to be performed in critical situation. In terms of large scale distributed network system, a group of users is required to deal with. One of the large scale distributed network system (over 5000) is power transformers on-line monitoring and fault diagnosis. It is hardly possible for the user to diagnose the fault types and performs appropriate actions on-time individually before serious problem appear. The lifetime of power transformer is designed to be around 30 years, in terms of normal operation. According to its long lifetime and maintenance cost, their maintenance services are carried out less often than supposed to be, and the transformer lifetime may shorten [7]. On the the other hand, some types of substations may have a spare transformer for the case of fault situation. For this purpose, there are circuit breakers which provide a possibility of excluding the faulty transformer (for the purpose of servicing) from the bus line and switching to the spare one. For instance, Figure 2.9 shows the parallel transformers applied to supply the 11KV low voltage bus (parts A and B in figure). In case of correct operation of two transformers SGT1 and SGT2, the bus-tie breaker remains open, and the 11KV low voltage bus is supplied. In the case of some fault appears in the power transformer SGT1, part A of the 11KV bus remains without power. In this case, MAS for on-line transformer fault diagnosis informs the CB1 (the agent that wrapped CB1) to open, and similarly the bus-tie breaker to close. Finally, the part A of the system is supplied by power, therefore faulty transformer operates abnormally for a shorter period of time.

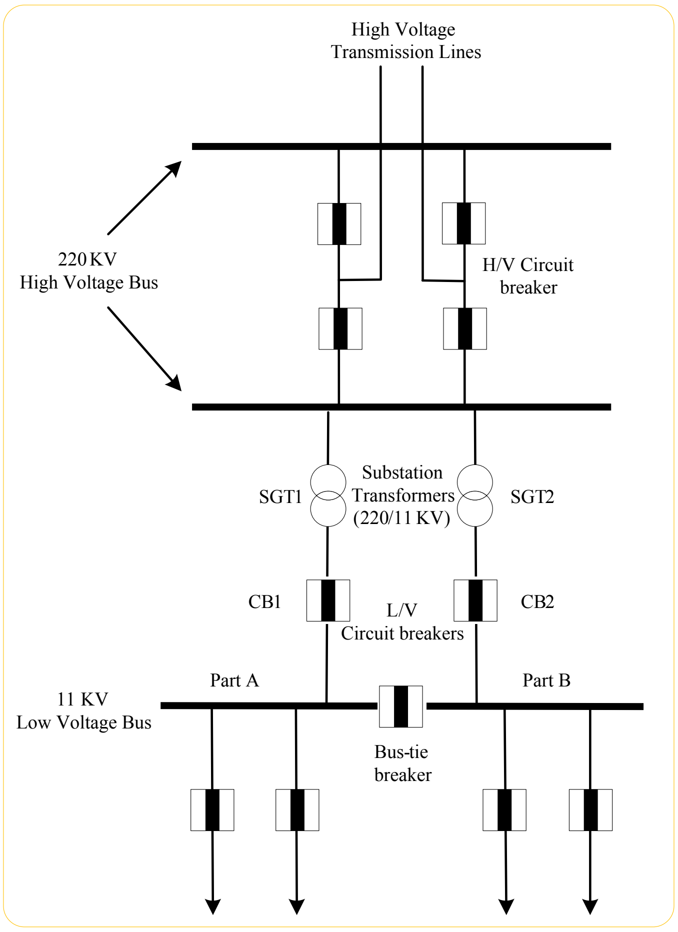


FIGURE 2.9: Distribution substation schematic diagram

On-line monitoring and fault diagnosis of distributed power transformer are the key points of this work. One of the research projects in the Department of Electrical

Engineering and Electronics at the University of Liverpool is focused in e-Automation [58]. The e-Automation architecture contains a real-time simulator, data acquisition devices, real time automation platforms and IP networks, including a wireless local area network, that can be used to undertake research in the area of network-based industrial automation. The key point of the laboratory is to focus on improving the real-time condition monitoring, information management, automation and fault diagnosis in a power system.

Previous MAS works in power system automation are developed in this e-Automation laboratory. For instance, agents are designed to collect the data from different components of power system simulator and save them in database for the future analysis. The collected data are accessible by the user through the relevant agents. The agent system has the ability to take a user's order from a user interface console to perform certain actions. The system is able to collect and present the real-time information of the components to the user on the top level. In the case of large scale, the participation of more experts for the purpose of control is necessary, which obviously leads to increases of cost.

2.8 Summary

The background of agents, multi-agent system and condition monitoring of power transformer are introduced in this chapter. Agents definition and their standard for communication are reviewed. FIPA as the main standard is explained clearly. Some methodologies for design of MAS are summarised, and their characteristics and specifications are discussed. MAS application is introduced, and the recent research projects is reviewed. Power system components are briefly discussed. Power transformer and its components for on-line monitoring are introduced. Some method of fault diagnosis based on DGA are also reviewed. Regarding to agent properties, the key point of using them in such application is agent ability to capture information from its environment, think and perform the appropriate actions. Based on this objective, the MAS should be able to capture the real time information of equipment and by using some techniques of decision making, an appropriate actions are selected and applied. This can be solved by providing an knowledge-based system for MAS.

Chapter 3

Multi-Agent Framework for Power Transformer Monitoring and Fault Diagnosis

3.1 Introduction

The aim of this chapter is to present a new framework for power system automation based on a formal methodology called Gaia; this methodology was introduced by Wooldridge and Jennings [28]. The analysis and design parts of the Gaia methodology are implemented step by step. Following this method, a design process is clarified, simplified and standardized while creating optimal MAS for transformer condition monitoring. Agents developed in this system are capable of receiving real-time data from a transformer relevant sensors (like DGA sensor) in a substation, and perform such tasks as saving, monitoring, reporting and reacting autonomously by determining the most suitable solution. In addition to reducing the user activities, knowledge-based systems in forms of rule based reasoning, ontology and fuzzy-ontology are developed for power transformer on-line monitoring and fault diagnosis.

The overall structure of the developed multi-agent framework is presented in Figure 3.1. As can be seen in the Figure 3.1, the hierarchy structure consists of three levels; components at the lowest level, agents are situated at the second level, and finally the top level consists of the user and knowledge-based systems for monitoring and decision

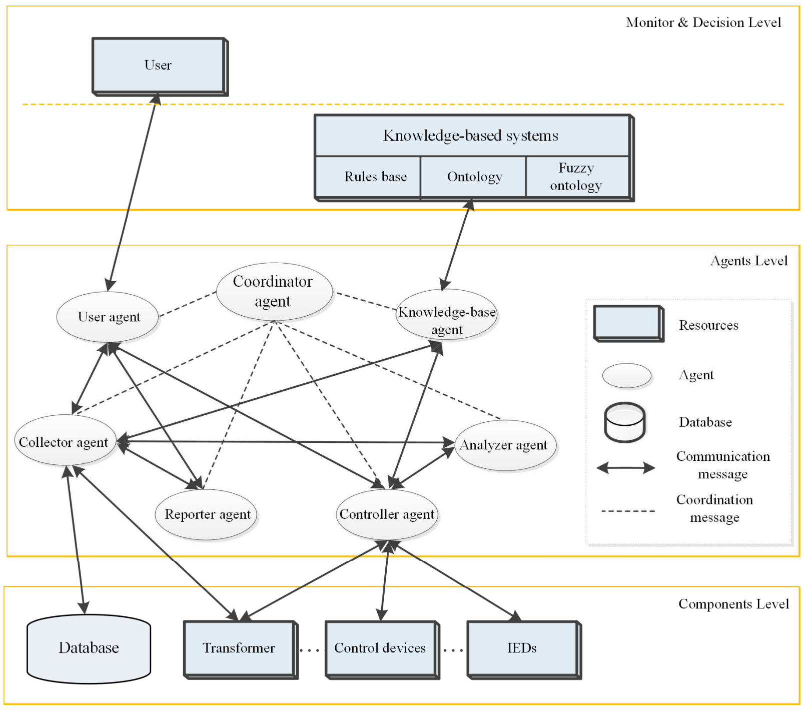


FIGURE 3.1: A hierarchy of the multi-agent framework

processes, noted as monitor and decision level. However, user priority is the key concern for influencing the overall system. Components level contains equipment, such as transformer, circuit breakers, DAU, databases, IEDs, etc., that are combined with different agents for data extraction, operation, condition monitoring, fault detection and etc. The agents level represents various types of agents, which are analysed and designed according to Gaia methodology. Usually in decision level, the monitoring process and decision performance are implemented by the user only. The developed system introduces three types of knowledge-based systems for this level, in order to reduce the users' efforts.

In this work, the developed multi-agent framework is designed and implemented based on the e-Automation architecture, later on extended with application of the rule-based and ontology-based reasoning for the purpose of automation and fault diagnosis. The software agents development design based on Gaia methodology and the evaluation of the performance of system are described in details in the following section.

3.2 Gaia Methodology for Agent Development

As mentioned previously, Gaia is a methodology for agent-oriented analysis and design in the macro and micro levels of systems. Its design structure allows the developers to choose the most suitable organisational structure for solving a problem, and enables re-use of agent-oriented organisational patterns, such as efficiency, robustness, degree of openness and ease of enactment of organisational structures [59]. The Gaia methodology is meant to be suitable for developing such systems as ADEPT [60] and ARCHON [8]. Building an agent-based system following this methodology is similar to the process of organisational design. An organization is a collection of roles with certain relationships with each other in systematic patterns and interaction.

The aim of this methodology is to define a road map from statement of requirements, denoted as analysis part, to the implementation of the system, called design part. The main models of Gaia are presented in Figure 3.2 [28].

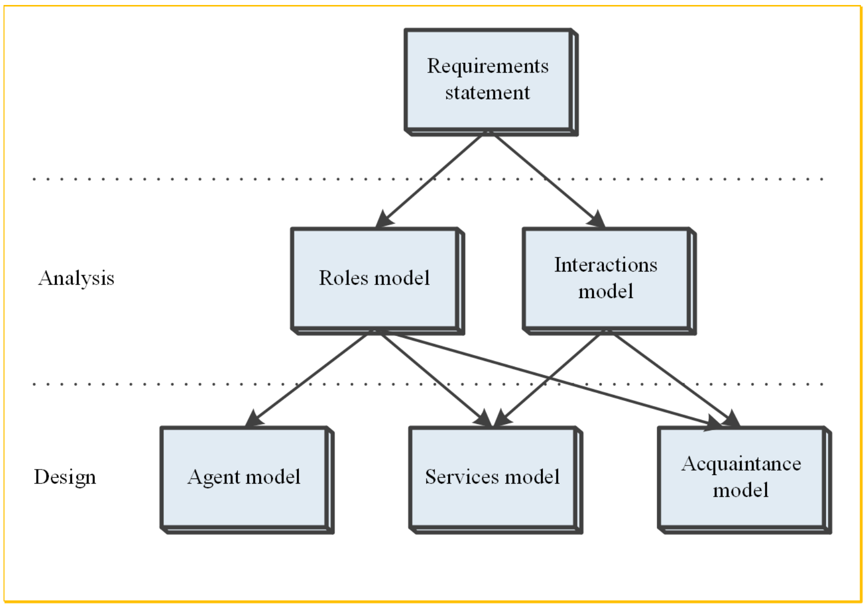


Figure 3.2: The summary of Gaia methodology models

As shown in the Figure 3.2, the first requirement of Gaia is the analysis part, that can be done by initializing the role model. The role model is the key feature of an agent system design. Each role consists of schema to represent its responsibility, permission, protocol and activity. The key role attributes, defined as responsibilities, determine the role functionality. There are two types of responsibilities: liveness properties and

Table 3.1: Formal notation to express the properties in Gaia

Operator	**Interpretation**
$x \cdot y$	x followed by y
$x \mid y$	x or y occurs
x^*	x occurs 0 or more times
x^+	x occurs 1 or more times
x^ω	x occurs infinitely often
$[x]$	x is optional
$x \| y$	x and y interleaved

safety properties. Liveness properties state that "something good happens", while safety properties states that "nothing bad happens" [28]. Activities or protocols are atomic components of a liveness expression. Table 3.1 shows the formal notation to express these properties presented in Gaia.

In order to draw the responsibilities of each role, some information resources are available, defined as permissions. Figure 2.6 in Section 2.4.3 represents the template for MAS role schema. Following that, the second part of analysis involves the interaction model between roles. The interaction model consists of a set of protocol definitions, one for each type of inter-role interaction. The protocol definition consists of six attributes itemized as follows:

- Protocol name: brief textual description of interaction;
- Initiator: the starter role for interaction;
- Responder: the initiator role interacts with;
- Inputs: information used by initiator;
- Outputs: information supplied by responder;
- Processing: brief description of processing protocol initiator.

The schema for one protocol is given in the Figure 3.3.

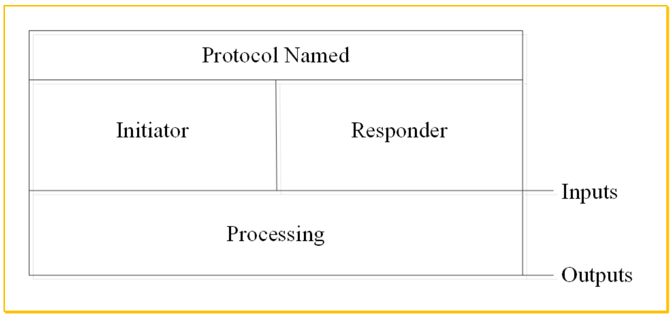


FIGURE 3.3: Schema of protocol definition

Design part of the Gaia methodology is the next step after analysis of the system requirements; it requires sufficiently low abstraction level, thus easily can be used in traditional design techniques, such as Object-Oriented (OO) techniques, for agents implementation. This process involves use of three models: agent model, services model and acquaintance model, as shown in Figure 3.2. For instance, the role called "Data_Collector" in role model can appear as an agent denoted as *Data_Collector* agent. A service in object orientation can correspond to a method available for another object. An agent also engages a single, coherent block of activity, denoted as service in service model. Four attributes are required for each service: inputs, outputs, pre-conditions and post-conditions. The protocol model from analysis part helps to find the inputs and outputs of services, while the safety properties of the role derive pre- and post-conditions. To put it another way, the list of protocols, activities, responsibilities and liveness properties of the role derives the services that an agent performs. Finally, the existing communication pathways between agents are defined by acquaintance model. This model can be represented with graph, consisting of nodes (corresponding to agents) and arcs (corresponding to communication pathways). It might be necessary during the design part of system to revise the analysis stage to recover the problems.

3.2.1 Gaia implementation for transformer condition monitoring

Several different standardisations are accepted in multi-agent technology, one of them is the FIPA. The principle aim of FIPA is to provide a general standard for multi-agent technology with different paradigm in a variety of domains. JADE platform is one of the agent platforms compliant with FIPA standards. JADE provides important services, such as agent management system, directory facilitator, agent communication channel etc. According to these specifications, the developed system was built in JADE platform and followed the FIPA specification standard.

Two key aspects are required to design a MAS for power transformer, condition monitoring and fault diagnosis. In terms of power transformer condition monitoring, various parameters, such as voltages, currents, temperatures, key gas ratios, etc., are required to be captured, maintained and precised to increase the reliability. In terms of fault diagnosis, the captured data can be diagnosed, based on applied techniques in knowledge-based system. In this case, top level (user) is able to access to the required data and reports in a form of request. Moreover, user can request to perform actions at the components level; for instance ask to close or open the specific circuit breaker. In parallel user performance, the knowledge-based system can also perform actions in critical situation.

The proposed system is designed to meet the following requirements:

- **Data_Sender** can read the actual data from given text file, then create a message every minute and send it to the "Data_Collector";
- **Data_Collector** waits to receive the real-time data from the "Data_Sender", save them in the database as raw data and send them to the "Knowledge base" for fault diagnosis. The result of fault diagnosis will be saved in the database. It can also get requests from "User" for an access to data and, thus, can respond to "User";
- **Reporter** is capable of being informed by "User" about requested data, then obtains the data from "Data_Collector" and plots them in a graphical window;
- **User** is able to utilise a user interface, which can send requests about chosen reports or data and get replies. It is also able to send request to perform actions in components level;
- **Controller** is an equipment situated in the components level that is designed to perform action at the time required. For instance, equipment such as alarms, circuit breakers, IEDs, etc., are wrapped by this role, for being informed to operate in critical time, subject to receiving an appropriate message;

- **Knowledge base** is able to utilise the knowledge-based system. It is able to receive data and use the knowledge-based system for diagnosing the data status. It is also can inform the "Controller" with an appropriate actions;
- **Analyser** receives data, connects to the MATLAB program for fault diagnosis, uses KNN method. The result will be sent to the "Data_Collector" to be saved in database.
- **Coordinator** provides the coordination between roles to manage their actions correctly and accurately. This can be done by roles that are able to present their performed reports in certain time;

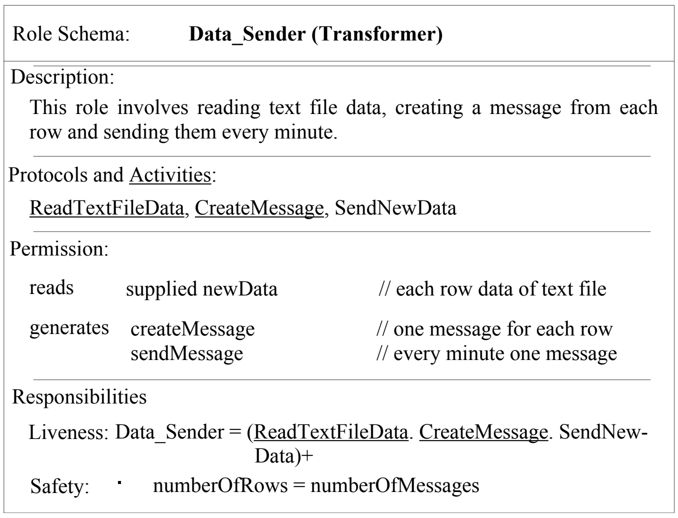
According to these requirements, the analysis part of the developed system is presented in the following section.

3.2.2 Gaia system analysis

The analysis phase identifies eight roles: "Data_Sender" (instead of "Equipment") is the first role which sends the data in form of message. The remaining roles are outlined as follows:

"Data_Collector" wraps a database while being responsible for data query and data saving in a database. "Reporter" handles a report. "Controller" wraps the control equipment, such as alarm, circuit breaker, etc., for performing an appropriate action requested with messages. "User" can handle the user requirements, and is able to interact with "Controller" in order to perform an action. The "Analyser" role is designed for the purpose of connecting to the MATLAB for fault diagnosis based on using different methods (such as machine learning). The role called "Coordinator" provides coordination between roles by being informed with report of tasks completion. Finally, the "Knowledge_Base" role wraps the knowledge-based system, consisting of rule-based system, ontology and fuzzy ontology.

Figure 3.4 shows the schema for "Data_Sender" ("Transformer") role. This role is able to read data in form of text file and send them in form of message every minute. The brief description of the role is given in the description part of the role schema in the Figure 3.4. The role's activity is to read the text file containing data every minute and create a message containing data. The message with new data, defined as protocol, needs to be sent to the role “Data_Collector”. Following these definitions, the activities for this role are “ReadTextFileData” and “CreateMessage”, while its protocol is “SendNewData”, as given in Figure 3.4. The role needs to access some information, in this case the text file data, to be able to perform its action. This is given as a permission of this role to read the new data from the text file. This role also needs to create a message and send it to the specific role. These three permissions for role “Data_Sender” are given in the role schema. Finally, the role responsibilities consist of liveness and safety. The liveness are protocols and activities required to occur timely. This means that the role needs to read the data, create a message and send it every minute. The safety in this case is to check whether the number of rows in text file data is equal to the number of the messages. Similarly, the rest of the role schemas are defined and presented in Appendix A.



Role Schema: **Data_Sender (Transformer)**

Description:
This role involves reading text file data, creating a message from each row and sending them every minute.

Protocols and Activities:
ReadTextFileData, CreateMessage, SendNewData

Permission:

reads	supplied newData	// each row data of text file
generates	createMessage	// one message for each row
	sendMessage	// every minute one message

Responsibilities
Liveness: Data_Sender = (ReadTextFileData. CreateMessage. SendNewData)+
Safety: • numberOfRows = numberOfMessages

FIGURE 3.4: Schema for the role “Data_Sender” (Transformer)

Second part of the Gaia analysis is the interaction model, which defines the dependencies and relationships between various roles. The definition of protocol associated with “Data_Sender” role is illustrated in Figure 3.5. The protocol is called “SendNewData”; and its purpose is to get new data and create a message containing these data. The data is required to be delivered to the “Data_Collector” role to be saved in the database. In this case, the initiator is the “Data_Sender” role, who starts the interaction with the responder. The responder is the “Data_Collector”, who receives the message. The input information used by initiator is “newData”, while the output information supplied during interaction is “messageWithNewData”. It is also obvious, that more details of protocol definition can be added at any time to complement and clarify its conceptual definition. The rest of the roles are designed in the same way, as given in Appendix A.

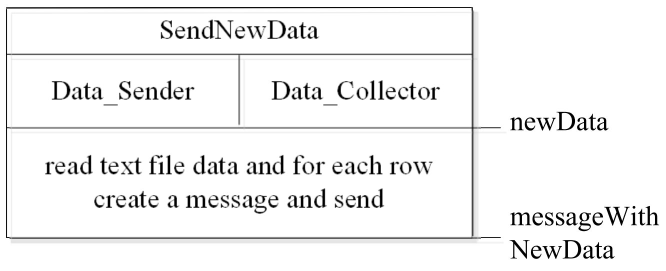

SendNewData
Data_Sender
Data_Collector
read text file data and for each row create a message and send
newData
messageWith NewData

FIGURE 3.5: Definition of protocol associated with the “Data_Sender”

The analysis part of Gaia methodology provides all the functional characteristics required for the design part. These structured specifications can be used in architectural design of MAS in the following section.

3.2.3 Gaia system design

The agent model can be represented with aggregated roles of the analysis model for the system. The analysis phase is aimed to define the general properties of MAS, while design phase focuses on actual characteristics of MAS. The first step of Gaia design is to define the agent’s model. This model for the developed system consists of eight types of agents. The *Data_Sender* agent is capable of “Data_Sender” role. This agent reads text file data and sends them every minute. The *Data_Collector* agent with capability of “Data_Collector” role deals with the data. The *Reporter* agent gets request about particular report and plots them in form of graphical reports. The *Controller* agent gets request to perform an action and informs user that the action is completed. The *Coordinator* agent provides coordination and cooperation between agents, while the *Knowledge_Based* agent provides accessibility to the knowledge-based system. Finally, *Analyser* agent handles the analysis of role attributes. The developed agent model is presented in Figure 3.6. For instance, the agent called *Data_Sender* is expressed with arrow and annotation "+" to play the role "Data_Sender". The annotation "+" means that one or more instances at the run time will be created in the MAS.

$$\text{“Data\_Sender”}^{+} \xrightarrow{play} \textit{Data\_Sender}$$

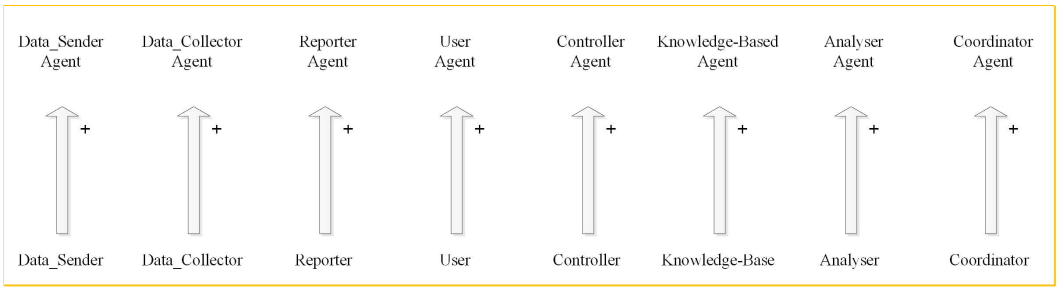


FIGURE 3.6: The agent model for developed system

The second part of design process is to develop a service model representing the main services required to assign the agents' roles. This model is simply taken from the roles' protocols of the analysis part. For instance, the "Data_Sender" role with its "SendNewData" protocol derives a service called "present new data". The input of this service is new data, while the output is the message containing new data. The pre- and post-conditions, associated to this protocol, is the sent message with new data. Table 3.2 represents some type of services given in the service model. According to the Gaia methodology, the service model does not require a particular type of implementation for the services it documents, thus the designer is free to decide what kind of services are appropriate for the framework implementation [28].

The acquaintance model is the final part of the design, as illustrated in Figure 3.1, which represents the multi-agent framework. This model represents the interaction between agents for the developed system.

This analysis and design of the Gaia methodology provide a kind of roadmap for implementing the MAS. Another step required before developing the MAS is to select a platform compatible with FIPA standard. The next section presents the development of MAS in JADE platform.

Table 3.2: The service model for developed system

Service	Inputs	Outputs	Pre-condition	Post-condition
present new data	newData	messageContain-NewData	messageWithNewDataSend = true	NewDataSend = true
obtain new data	messageWithNewData	newData	data saved = true	data saved = true
find fault type	messageWithNewData	faultType	fault diagnosed = true	fault diagnosed = true
obtain data	messageWithRequestData	time&date	replied request data = true	replied request data = true
obtain report	messageWithRequestReport	report	repliedRequestReport = true	repliedRequestReport = true
required data	messageWithRequiredData	data	dataReceived = true	dataReceived = true
request to perform action	messageWithRequestAction	action	true	true
inform action done	messageWithActionConfirmation	actionDone	actionPerformed = true	actionPerformed = true

3.3 Developed MAS

In this section the MAS and a generic agent platform are presented. The section provides wider information about the agent platform and its relation with the FIPA standard of proposed system. Agent platform is aimed for implementation of the MAS and agents communication, according to the FIPA standards. JADE was selected as an agent platform, as it is compliant with the FIPA standards. The Gaia methodology applied covers the analysis and design phases of the software development cycle. Also, the current section contains information on combining Gaia methodology with JADE platform.

3.3.1 JADE platform

JADE is aimed to provide a framework for developing MAS according to the FIPA s-tandards. JADE platform have been selected as a platform in previous research works conducted in power system monitoring and control [3] [30] [61]. Distributed platform for building agents, AMS, DF, ACC and ACL, are the most important of the features presented in JADE. The language called FIPA-ACL is also used for presenting the communication between agents through their messages. In terms of JADE communication performance, the message can be sent by the *Initiator* agent to the *Responder* agent. In case if the message is not understood, or action requested cannot be performed, the *Responder* agent replies with “not understood” or refuse message, respectively. Otherwise, the *Responder* agent confirms that the action can be completed.

The agent’s performance is based on the different behaviours interaction. Thus, building an agent in JADE assumes the use of behaviours for the implementation of the agents tasks. Behaviours are defined as logical execution threads [62] with “setup” and “action” methods, and classified into three types [20], explained as follows:

1. One-shot behaviours are able to complete an action in one execution phase; their “action” method is thus executed only once;
2. Cyclic behaviours can be never completed, “action” method in this case performs the same task every time it is asked to;
3. Generic behaviours perform different operations depending on some status value, set as a status trigger. They can be completed subject to conditions met.

Moreover, JADE provides an opportunity of creating more complex behaviour by combining several types of them. Depending on the MAS requirements, the necessary behaviours are selected and applied for the developed system.

3.3.2 Combination of JADE with Gaia

The roadmap of switching from Gaia to the JADE has been developed according to an algorithm provided in [63]. Thus, the following steps were completed during this process:

1. The ACL messages were defined by using Gaia protocols and interaction models;
2. Found the required software modules and data structures to be used by agents in terms of using Gaia roles;
3. Figured out the safety conditions and their implementation in the case of each role;
4. The behaviour classes provided by JADE were used for defining the JADE behaviours starting from the lowest level;
5. Initialised all the structures of the agent data;
6. All the behaviours are added at the lowest level of the agent scheduler.

Any ACL message contains some information that can be accessed both by sender and receiver agents. The message usually consists of the following fields: message sender, list of message receivers, FIPA performative, protocol, language and content. FIPA performative contains some information that depends on the sender's intentions towards the message receiver, such as request, query, or inform. The content of the message provides additional information on the action to be performed. The language helps to express the message content in the way to be understandable both by message sender, and receiver. Additionally, the ACL message may contain other information, such as ontology, timeouts, etc. For instance, according to the FIPA ACL Message Structure Specification [18], the JADE ACL messages for “RequestPerformAction” and “InformActionDone” are presented in Table 3.3.

As shown in Table 3.3, the *User* agent can send a request to the *Controller* agent to turn the alarm system on. The user can be informed that the alarm system is on. The

Table 3.3: The ACL message definition for user to perform an action

ACL Message: RequestPerformAction	**ACL Message**: InformActionDone
Sender: *User*	**Sender**: *Controller*
Receiver: *Controller*	**Receiver**: *User*
FIPA Performative: REQUEST	**FIPA Performative**: INFORM
Protocol: RequestPerformAction	**Protocol**: InformActionDone
Language: SL	**Language**: SL
Content: AlarmSignal_ON	**Content**: AlarmSignal_ON

one-shot behaviour is selected to perform the action, subject to receiving a specific ACL message. This can be done with request messages given as follows:

```
( REQUEST
   :sender (agent-identifier
        :name User)
   :receiver (set (agent-identifier
        :name Controller))
   :content ‘‘(( action( ON AlarmSignal )))’’
   :language FIPA-SL
   :protocol fipa-request
)
```

The message informing that the alarm system is on, can be presented in the following way:

```
( INFORM
   :sender (agent-identifier
        :name Controller)
   :receiver (set (agent-identifier
        :name User))
   :content ‘‘(( done (action ( Alarm_Signal_is_ON ))))’’
   :language FIPA-SL
   :protocol fipa-request
)
```

In case of agent is located in the different host, it is required to have its Internet Protocol (IP) address. For instance, if the alarm system is located in the host with IP address "192.168.1.127", the sender should send a message with receiver name and its IP address, e.g. "Controller@192.168.1.127". The IP address of any agent can be captured through the DF agent, as applied previously in power system automation [6, 11]. Message sender also can be a knowledge-based agent which had been fired after meeting the data status. For instance, in case of rule-based reasoning, the knowledge about performing particular actions is written in form of "*IF ... THEN ...*" statements. The received data (facts) can be matched with the defined status, and necessary action will be performed.

3.4 Knowledge-Based Systems

Knowledge-based system is a software system that can mimic the performance of a human expert in a limited sense. The knowledge can be represented in different ways [64], such as semantic network, logic, procedure, production systems (rules), frames, etc. Each form provides different characteristics. The purpose of the knowledge representation is to solve the problems arising with the integration of some body of knowledge into the computer system. This results in automated and intelligent reasoning. In case of the fault diagnosis, three types of knowledge-based system with different abilities are developed. In terms of this work, the rule-based reasoning was applied as a first knowledge-based system to present the information about transformer fault diagnosis and perform some necessary actions. Ontology was chosen as a second way of knowledge representation to perform the same actions in terms of ontology. The improvement of the ontology use, in terms of proposed knowledge base, was achieved by using fuzzy ontology as a third step of transformer fault diagnosis, based on DGA.

3.4.1 Rule-based reasoning for transformer fault diagnosis

A power transformer fault diagnosis system with rule-based reasoning has been established in the proposed multi-agent framework to reduce the user effort. In this system, two applications of rule-based reasoning for power transformer fault diagnosis are investigated. First application is to diagnose the power transformer based on DGA data.

Transformer's status can be diagnosed based on gas ratios and Roger's method. The second application is to apply the rule-based reasoning on behalf of the user in critical situation. Furthermore, reducing the user's efforts may lead to the cost reduction. The various agents are implemented using the JADE platform; rules are written in Java Expert System Shell (JESS) [65]. Figure 3.7 demonstrates a UML use in case of this system.

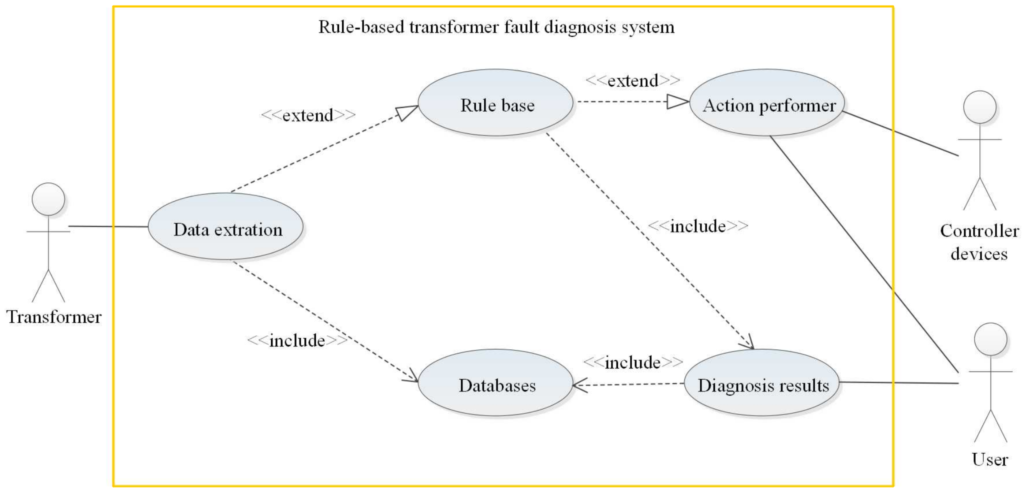


FIGURE 3.7: A UML use case diagram of the transformer fault diagnosis system based on rule-based reasoning

3.4.2 Ontology-based reasoning for transformer fault diagnosis

Knowledge in knowledge-based system can be represented in terms of logic. In this case, a set of concepts within a domain and their logical relationships between pairs of concepts are defined as a ontology. Ontology provides a shared and common understanding of data that exists within an application integration problem domain, and the way of facilitation of communication between people and information systems. Thus, the concept of ontology can be used to organise and share information, manage knowledge and improve interoperability of communication systems within the company. Based on the proposed multi-agent framework, ontology is able to represent the transformer and its components. Ontology is applied for transformer fault diagnosis. In this system an ontology is developed to represent the relationship between transformer's components, fault symptoms and fault types. The purpose of ontology is to enable the knowledge sharing and reuse. The developed ontology for transformer fault diagnosis is built in

Protégé platform [66] and wrapped with developed agent for interaction with multi-agent system. To demonstrate the requirements of the system, a UML use case diagram is shown in Figure 3.8.

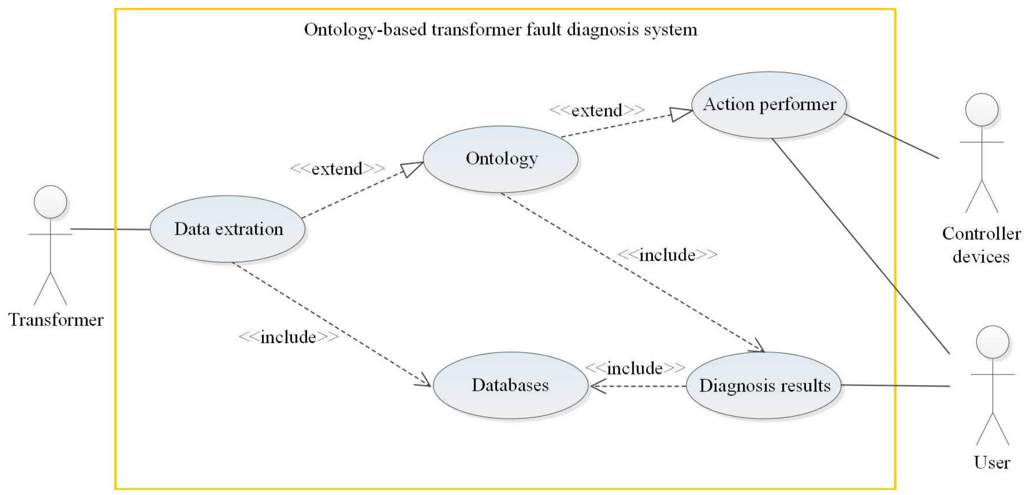


FIGURE 3.8: A UML use case diagram of the transformer fault diagnosis system based on ontology-based reasoning

The ontology development for transformer fault diagnosis can be improved further by using the fuzzy ontology. The advantage of the fuzzy ontology is that its elements can belong to a set to some degree. This helps to define with some certain degree whether the concepts belong to some category. In this case, the fuzzy ontology for transformer fault diagnosis is developed to improve the system performance. The software called Fuzzy OWL 2 Protégé plug-in [67] was applied to build a fuzzy ontology.

3.5 Agents Collaboration

The agent communication uses FIPA-ACL in the speech act theory that states communicative acts and messages representations. The chain process governing the system is organized by sending data in form of messages to the relevant agents. According to the messages exchange, various tasks can be performed using the agent system developed. The tasks for developed system are itemised as follows:

1. Data collection and fault diagnosis;

2. User interaction;

3. Automatic action performing.

The following subsection illustrates the examples of the tasks execution using the system proposed.

3.5.1 Agents collaboration for data collection and fault diagnosis tasks

The purpose of this task is information collection from the messages received, retrieval of this information and placing it to the correct table of database. The status of data also requires verification through interaction with the knowledge-based system. The sequential steps of collecting data and diagnosis task, as well as collaboration and coordination between agents are illustrated with a UML diagram in Figure 3.9.

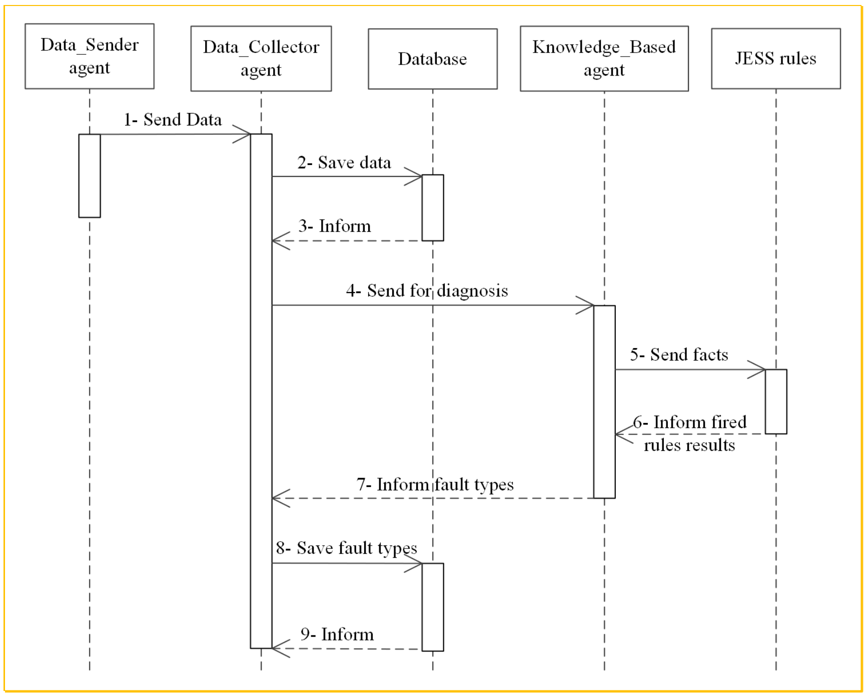


FIGURE 3.9: Collaboration between agents for data collection task

The aim of the *Data_Collector* agent is to collect the received data and save them into the table of database. This process is initiated by *Data_Sender* agent, who sends a messages containing data to the *Data_Collector* agent. The *Data_Collector* agent uses Java Database Connectivity (JDBC) API to connect to the database. In this case, the database is designed with MySQL and contains various tables. The data need to be sent for fault diagnosis by *Knowledge_Based* agent. Knowledge about specific information is written in JESS program. The data are send to the JESS for fault detection, and the results (fault types) are replied to the *Knowledge_Based* agent. The *Knowledge_Based* agent informs the *Data_Collector* agent about the fault types. Finally, the fault types are saved into the database by *Data_Collector* agent.

3.5.2 Agents collaboration for user interaction task

The tasks are aimed to provide an interaction between user, MAS and components. For this purpose, the user is able to request data, reports and action performing. The sequence of collaboration and coordination between agents is shown in the UML diagram, displayed in Figure 3.10. As can be seen from the figure, the *User* agent handles the user interface. This agent is able to send the requested data from *Data_Collector* agent and get reply data. The user is also able to request report from *Reporter* agent. The *Reporter* agent needs to request the data from *Data_Collector* agent, that accesses the database. The required data is sent to the *Reporter* agent for applying the drawn function and replied to the *User* agent.

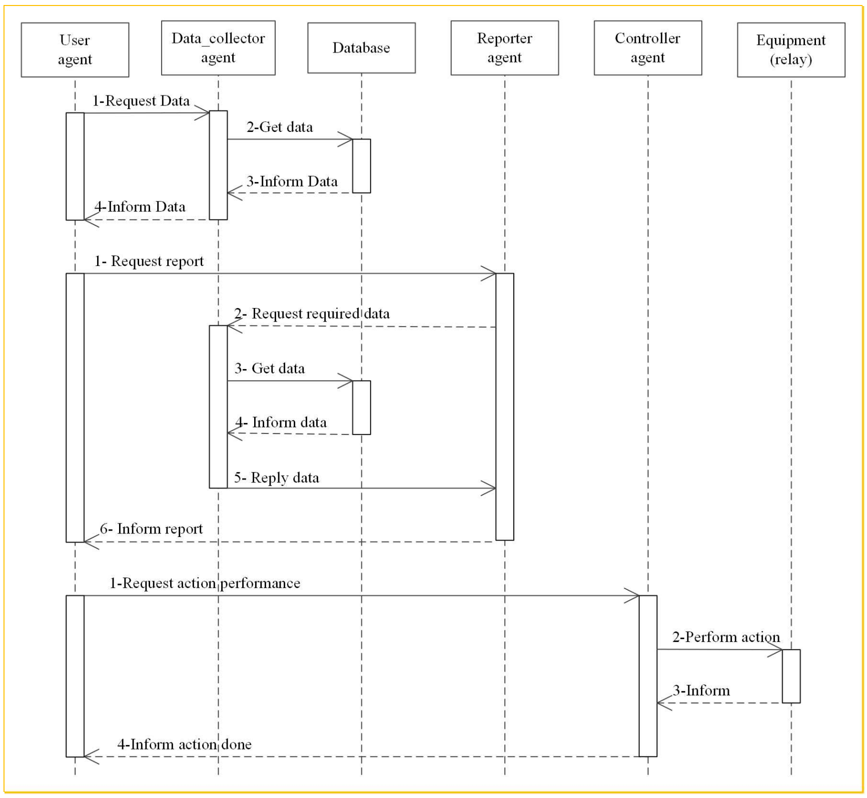


FIGURE 3.10: Collaboration between agents for user interaction task

The key task that can be applied by user is a request to perform a particular action. This can be done by *User* agent, who requests the action from the *Controller* agent and gets information whether the action is done. The more applicable this type of agent is previously applied in [1] by registering the controllers with DF and AMS agents. Then the *User* agent is able to identify the participating agent for performing the action. For instance, two relay agents $\mathcal{A}_{Rel1}$ and $\mathcal{A}_{Rel2}$ are able to perform actions, such as opening or closing the relay. These two agents are registered with DF and AMS agents. In case if user requires to perform action on the first relay (open or close), the requested action will be sent to *Controller* agent for particular service. The *Controller* agent sends a request of service provider AID agent from DF agent and gets reply. The *Controller* agent finally sends request to perform the action to the service provider (in this case $\mathcal{A}_{Rel1}$), and gets information that the action is completed.

3.5.3 Agents collaboration for automatic action performance task

The aim of this task is to perform appropriate action on behalf of user in critical situation. For this purpose all the agents should register their services with DF and AMS agents, in order to be accessible by the other agents. The real-time data is sent to the *Knowledge_Based* agent for the purpose of fault diagnosis and defining required actions. According to the knowledge-based system, the rules are fired, and the appropriate actions are defined. The search of the agents that are able to provide the required action can be performed through DF agent. The *Knowledge_Based* agent requests the services from the service provider (in this case *Controller* agent). The *Controller* agent requests the action to be performed by corresponding the controlling equipment and gets reply. Finally, the *Controller* agent informs the user of the applied actions. The sequence UML diagram of agents collaboration and coordination is provided in Figure 3.11.

3.6 Agent Analyser

The key point of this study is to investigate that the proposed multi-agent framework has ability of applying fault diagnosis methods based on MATLAB platform. For this purpose an agent called *Analyser* capable of carrying out the fault diagnosis based on the DGA samples has been developed. In this case key gases (or gas ratios) are sent to

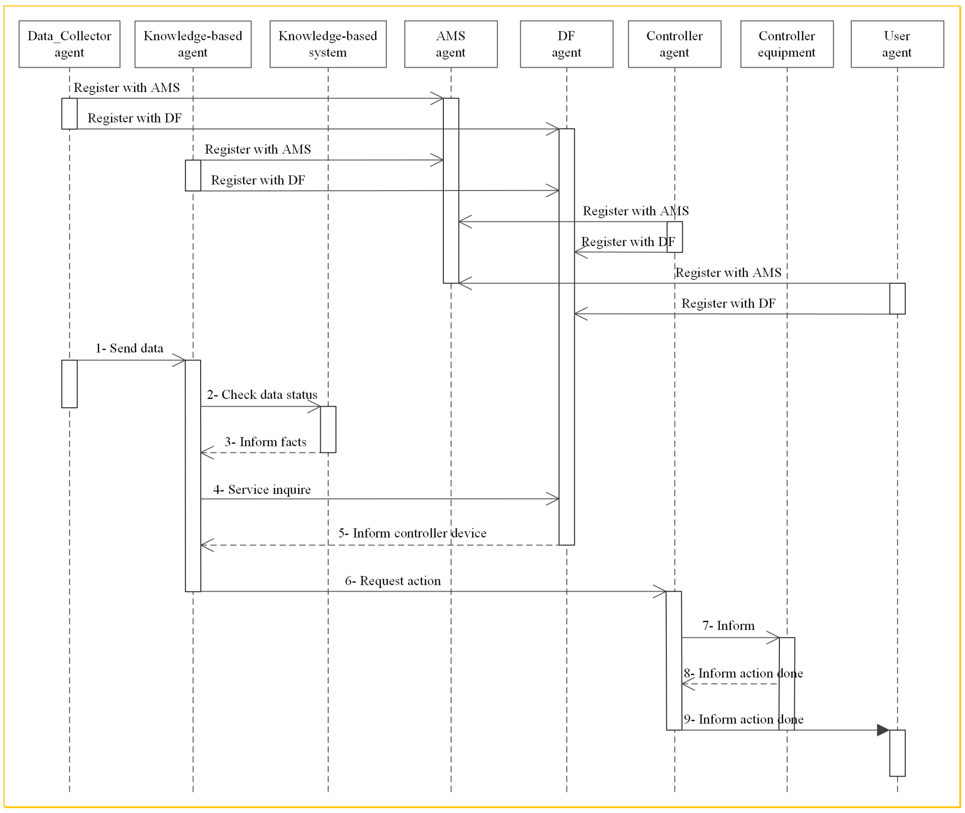


Figure 3.11: Collaboration between agents for automatics performs an action task

the *Analyser* agent. This agent is able to connect to the MATLAB and uses K-Nearest Neighbour (KNN) classification method for fault diagnosis. Finally, the information about the fault types is sent to the message sender.

3.6.1 Data and features

In electric power system real-time data are captured from equipment and available in DAU [46]. Depending on required information, sensors are designed and installed in equipment. In power transformer on-line monitoring, the DAUs are designed and situated in a cubicle at the transformer or in a control center. The collected original data are also defined as features, which saved into the database for future analysis. For instance, oil temperatures in oil-immersed transformer are different at the top and bottom of the oil. These two objects are captured separately and defined as two features.

The captured data are usually presented in the database as some numerical data, such as temperature, gas ratio, etc. In practice, database may sometimes have missing data. These incomplete data may be due to missing measurements, incorrect measurements or imperfect procedures of manual data entry, which is common in data collections. Missing data may be reflected as a noise in classification stage [68]. This issue should be solved before any action is performed on incomplete data. The action of completing the missing values in a set of data is called imputation. Three traditional treats are recommended for data imputation: (a) remove the samples or (b) fill them with zeros or (c) fill with mean computed from available values. First method can be used for the low number of missed data and it is not applicable for the large incomplete data. Filling with zero may not be useful if the new features have to use mathematical equation (like division) for generating new features. The third method takes the mean from available values in each feature and replaces missing data with mean. More significant and accurate imputation methods are recommended in [69, 70] to deal with this problem. Table 3.4 shows an example of thermal fault gas samples with some of the data missing. As can be seen from Table 3.4, the missing values (later in experiment) are replaced by mean of corresponding features in actual data.

Table 3.4: DGA samples of thermal fault containing missing values

H_2	CH_4	C_2H_2	C_2H_4	C_2H_6	CO	CO_2
12	18	-	4	4	559	1710
48	610	-	10	29	1900	970
150	22	11	60	9	-	-
1860	4980	1600	10700	-	158	1300
8800	64064	-	95650	72128	290	90300

The following section describes a linear classification used in this study for fault diagnosis. The K-Nearest Neighbour (KNN) classifier was used for classification in this study, as explained in the next section.

3.6.2 KNN Classification Algorithm

The KNN algorithm is a non-parametric algorithm used to classify the objects based on K closest samples. A majority vote of the nearest neighbours is used to find the class of any object. The class of the test sample is the most common class amongst its K nearest neighbours [71]. In the case of $K = 1$, known as the NN rule which is the simplest version of this method, the class of the test sample is the class of the nearest neighbour. There is no simple way to select the value of K, and its value depends on particular application. A high value of K increases the computing time, while a value too low can increase the noise effect on classification performance.

The KNN algorithm uses different methods, such as Euclidean, Manhattan or Hamming, to define the distance between two input vectors. In Euclidean method, let x_n and x_m be two input vectors with two dimensional space. The distance between $x_n = (x_{n1}, x_{n2})$ and $x_m = (x_{m1}, x_{m2})$ is denoted by the difference vector $x_n - x_m$:

$$D_E = |x_n - x_m| = \sqrt{(x_{n1} - x_{m1})^2 + (x_{n2} - x_{m2})^2} \tag{3.1}$$

where the $K = 1$.

In the case of $K > 1$, the distance is defined:

$$D_E = \sqrt{\sum_{i=1}^{K} (x_{ni} - x_{mi})^2} \tag{3.2}$$

This formula is only valid for continuous variables, and in the case of categorical variables the Hamming distance can be applied as follows:

$$D_H = \sum_{i=1}^{K} |x_{ni} - x_{mi}| \tag{3.3}$$

In this study Euclidean distance was used for distance measurement.

3.6.3 Experimental agents for collaboration with MATLAB

The architecture of agent-based fault diagnosis for power transformer and the software agent development is shown as a diagram in the Figure 3.12. The process of fault diagnosis starts from sending the data by *Initiator* agent (in this case *Data_Sender* agent) to the *Analyser* agent. The content of the message is 7 key gases, the actual data obtained from the database. The *Analyser* agent is able to connect to MATLAB (as described in [61]) in order to carry out the fault diagnosis based on the key gases samples data. The identified fault type is passed by *Analyser* agent to the *Data_Sender* agent.

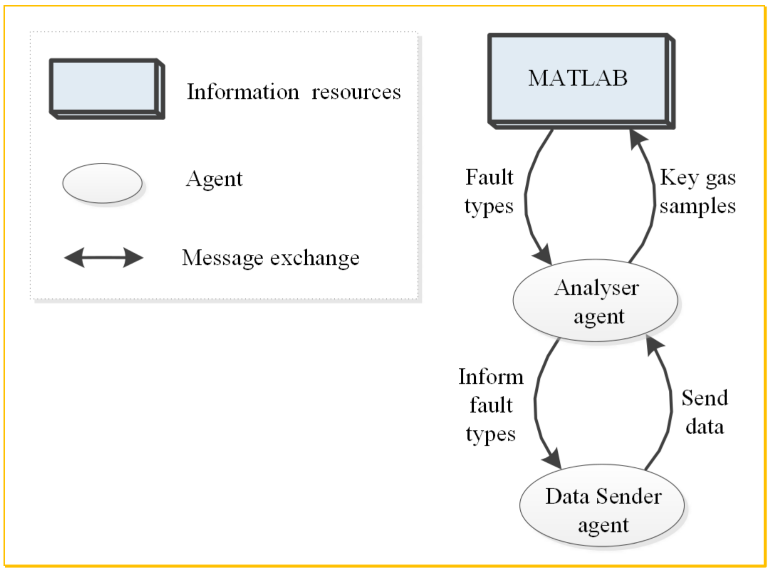


FIGURE 3.12: Collaboration of *Data_Sender* and *Analyser* agents in experimental system

3.6.4 Agent coordination model

Figure 3.13 illustrates the mechanism of the interaction, as well as the agent communication message sequence. The process of communication is initiated by *Data_Sender* agent, who sends the data on 7 key gases to the *Analyser* agent. This can be done by using following message:

```
( query-ref
  :sender (agent-identifier
     :name Data_Sender@192.168.1.187:1051/JADE)
  :receiver (set (Agent-Identifier
     :name Analyser@pc042385:1891/JADE))
  :content ‘‘((Key_Gases (H2 8266) (CH4 1061)
            (C2H2 2357.9) (C2H4 582.14) (C2H6 22)
            (CO 107) (CO2 498)))’’
  :protocol fipa-request
)
```

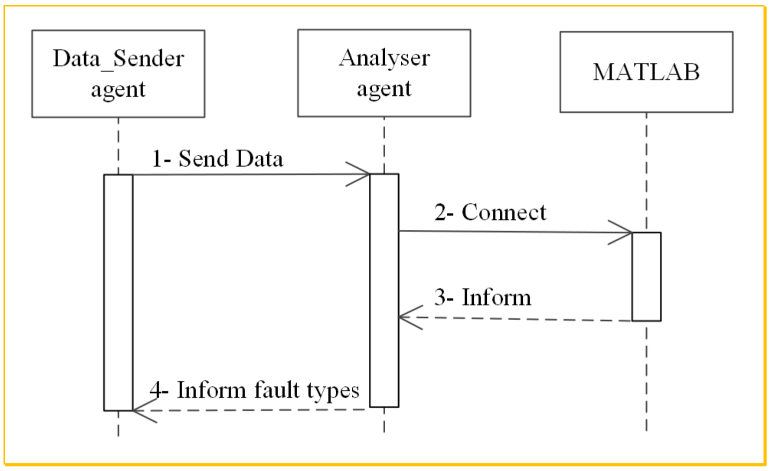


FIGURE 3.13: Collaboration between *Data_Sender* and *Analyser* agents

The *Analyser* agent consequently connects to the MATLAB and applies the KNN method to identify the fault types. The obtained information about the fault types is passed to the *Data_Sender* agent with following message:

```
( inform
  :sender (agent-identifier
     :name Analyser@pc042385:1891/JADE)
  :receiver (set (Agent-Identifier
     :name Data_Sender@192.168.1.187:1051/JADE))
  :content ‘‘((set Low_Energy_Discharge))’’
  :protocol fipa-request
)
```

3.6.5 Experimental result based on KNN classifier and agent *Analyser*

In this study, data were extracted from DGA, obtained from [54, 57, 72] for 191 samples. These samples contain 4 types of classes: “No Fault” for 49 samples, “Low Energy Discharge” for 48 samples, “High Energy Discharge” for 44 samples and “Thermal Fault” for 50 samples. Each sample consists of 7 types of gases, such as hydrogen (H_2), methane (CH_4), acetylene (C_2H_2), ethylene (C_2H_4), ethane (C_2H_6), carbon monoxide (CO) and carbon dioxide (CO_2).

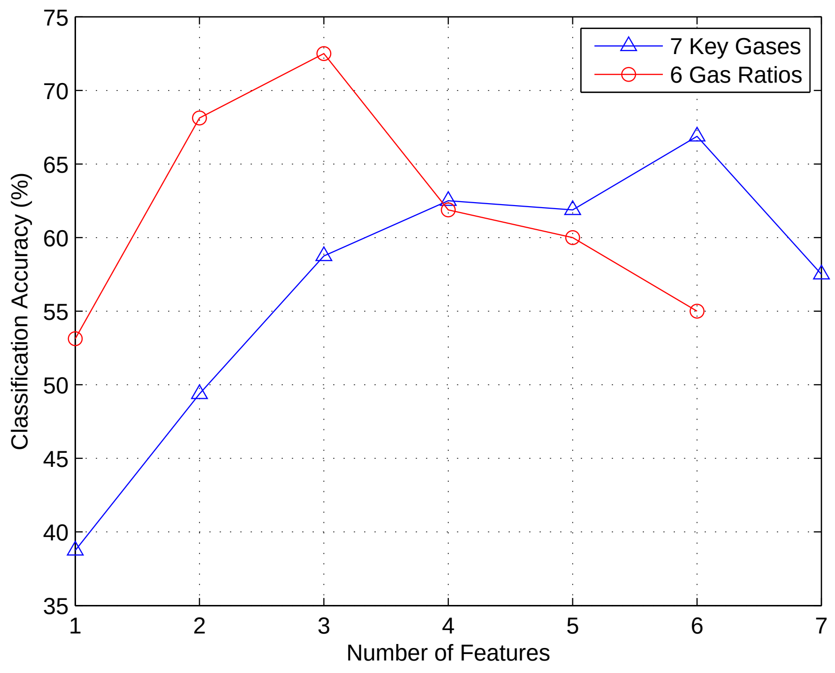


FIGURE 3.14: Accuracy with 7 key gases and 6 gas ratios

In this study the dataset was divided into five different training and test datasets with 80% and 20% training and test data partition, respectively. This 20% test class contains eight samples from each class fault. The division was done to carry out a 5-fold cross validation test ($5 \times f_c$) to be tested with KNN classifier.

Accuracy of the dataset is investigated for seven key gases and six gas ratios features, as shown in Figure 3.14. As can be seen, the maximum accuracy reached with seven key gases is 66.88%, and with six gas ratios is 72.25%.

3.7 The Experimental System for Rule-based Reasoning

The experimental systems were developed individually to evaluate the interaction between the *Data_Collector*, *Reporter*, *Controller* and *User* agents. Figure 3.15 shows the interaction between user and knowledge-based system. In this case, an agent called *Data_Sender* sends a message containing some data to the *Knowledge_Based* agent. This agent can connect to the knowledge-based system for the fault diagnosis and define the appropriate actions to be performed. Three types of knowledge-based systems individually are applied in this work, namely rule-based reasoning, ontology and fuzzy ontology.

The knowledge-based systems are able to perform action on behalf of user in critical situation. The rule-based reasoning was applied for performing an appropriate action. This choice was caused by the need of faster response, similarly to reactive architecture of MAS. All three types of knowledge-based reasoning were applied for fault diagnosis. The design and implementation of each type of knowledge-based system will be presented in the following chapters.

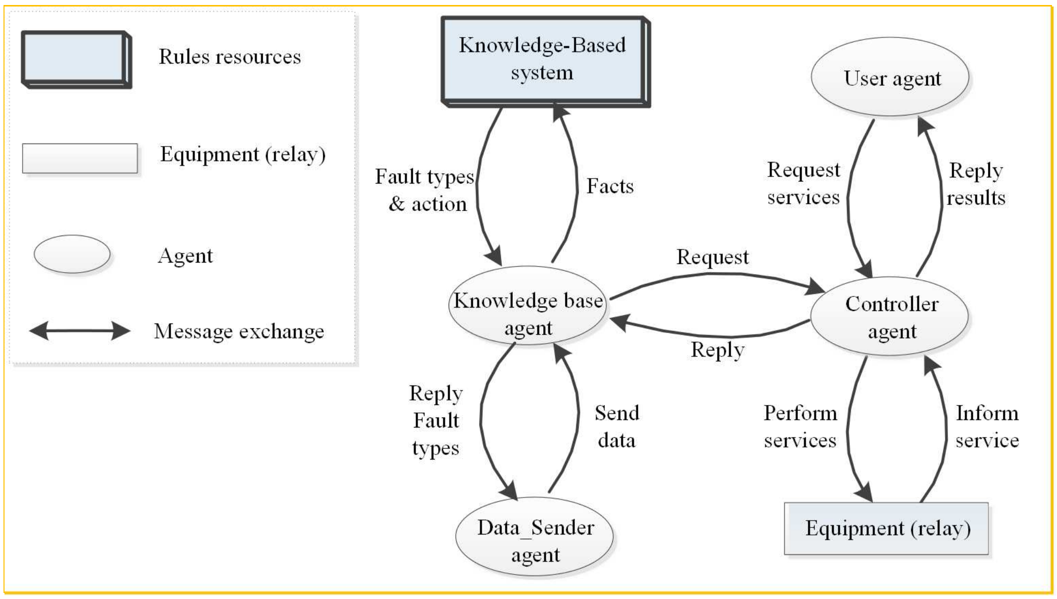


FIGURE 3.15: The collaboration of *Knowledge-Based*, *User* and *Controller* agents in the experimental system

The developed agent system uses MySQL database for data collection, with tables built for different types of information. The tables are built using the power transformer actual data available, such as ambient, top oil and bottom oil temperatures, key gases, etc. The *Data_Collector* agent is able to insert information to and retrieve it from the database, as applied in [73]. The software component JDBC was utilized for interaction of Java applications with a database. JDBC has the capability of designing a single Java program to manipulate the data in a variety of different SQL database servers (without modifying the program). Thus, the combination of JDBC and MySQL provides a powerful union to fulfil a variety of purposes. Appendix A illustrates a part of database content, filled by the *Data_Collector* agent with the given data.

JFreeChart [74] was used for production of graphical report (e.g. line graph). JFreeChart is capable of plotting lines, pies, bars, etc., depending on the requirements.

However, this work operates only the line plots for the process simplification. The data requires from the *User* agent are plotted and provided to the user. Figures 3.16 and 3.17 show the reports created for bottom oil temperature and ambient temperature of a power transformer by the *Reporter* agent, respectively.

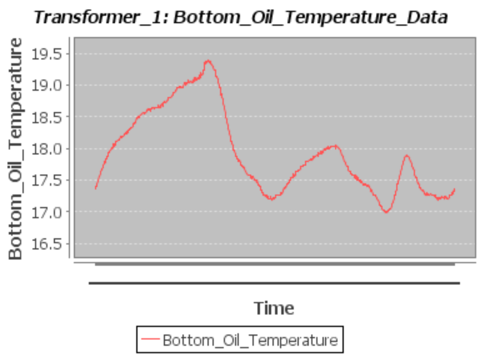


FIGURE 3.16: A sample report of Bottom-Oil temperature

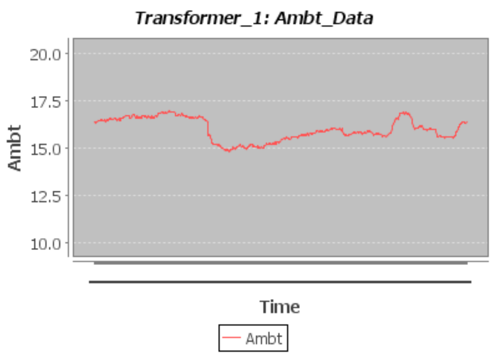


FIGURE 3.17: A sample report of Ambient temperature

3.8 Summary

A development of multi-agent architecture for power transformer monitoring and fault diagnosis is introduced in this chapter. The developed MAS is analysed in details and designed according to the Gaia methodology. Three types of knowledge-based systems were applied for the MAS to diagnose the fault and perform some actions. The proposed agent system solves the problem of on-line monitoring and fault diagnosis. Additionally, examples of agent collaboration for task performing were presented, such as interaction with user and performing some action using the information obtained from the knowledge-based system. The KNN classification algorithm is applied to find the classification accuracy for 191 DGA samples. The MAS was established based on the JADE platform, that is capable of agents execution and control over the message delivery. The chapter also presents detailed system design and gives examples of various agents performance.

Chapter 4

Power Transformer Fault Diagnosis with Rule-based Reasoning

4.1 Introduction

In this chapter rule-based reasoning is proposed for performing an automated action and power transformer fault diagnosis. In this case, an expert or Knowledge-Based System (KBS), a branch of AI with ability of utilizing computers to simulate the human intelligence in a limited way [75], are applied. Integration of MAS with KBS enables two applications of it. First, KBS is implemented to represent information about the particular component of the system and the actions required in critical situation. The second application of KBS and MAS combination is power transformer fault diagnosis. The fault diagnosis based on Roger's ratio method has been applied on DGA samples.

The chapter begins with introducing the knowledge-based system and rule-based reasoning, followed by the overview of the agent system architecture for two types of applications. Example of communication acts between various agents are provided. Finally, the actual data are applied practically with MAS for the purpose of fault diagnosis in power transformer.

4.2 Rule-based Reasoning or Knowledge-based System

A software system that can mimic the human performance expertise in a limited sense is denoted as a knowledge-based system [75]. Many researchers also denote the expert system as a KBS. In fact, the expert system is a classical example of a rule-based system that uses rules to make deductions in particular domain. It finds various applications, such as diagnosis, interpretation, prediction, monitoring, control, etc. The main components of KBS are Knowledge Acquisition (KA), Knowledge Representation (KR) and Knowledge Processing (KP). The KP is also known as a knowledge engineer; it collects the expertise in specific domain and arranges them in the form suitable for further use. The knowledge in KR can be in the form of logic, production systems (rule), direct (analogical), semantic network, procedure, frames, etc. The presented knowledge can be driven by program called Inference Engine (IE). The IE traverses the knowledge base to provide one or more outcomes, regarding its observations. Usually the outcome information is presented to the user, who represents the interface between the KBS and external world. Different types of inputs, such as transducers and sensors, are used to capture the environment. The output of the KBS can be stored in the databases or directly sent to actuators or controllers for appropriate actions to be taken.

The KR in form of production (rule), also called situation-action rules, consists of the rules written in an object-oriented programming language, i.e. Java. A rule system may consist of three components, itemized as follows:

- Rule base – consists of a set of rules;
- Fact base – consists of a set of facts;
- An interpreter for the rules.

In simple design, the rules are written in terms of "*IF - THEN - ACTION*", thus rules and actions can be performed if the clause is true. Therefore, an external text editor is often applied for KA facilitation. Shells are the most widely used expert systems, containing the software required for programming. User is responsible for the knowledge base building according to the system requirements; this can be taken as an advantage of the shell and finds various application. The independence of the rules represents their other advantage, thus the rules can be easily modified, added and deleted.

knowledge-based system can find several applications in power system, such as fault diagnosis [76], alarm processing for an energy management system [77], supervisory and control of voltage using fuzzy logic [78], etc. It also can be applied with MAS for purpose of automated management, SCADA analysis and fault recording [79].

In order to apply the KBS with Java compatible program, a use of tool called Java Expert System Shell is required.

4.2.1 Java Expert System Shell (JESS)

The JESS is based on CLIPS (a public domain software tool), and it is a rule engine and scripting environment written in Java language by Ernest Friedman-Hill [65]. Applying the JESS tool with built-in application in Java provides capability of reasoning by using the knowledge supplied in the form of declarative rules. For building an intelligent software system, a set of rules is applied to the collection of the facts about the world. There are three ways to represent them in JESS: rules, functions and object-oriented programming. The data captured from environment (facts) are presented, and the matching rules are fired. Rete algorithm [80] is the fastest applied algorithm; it is used in JESS to derive facts and rules. One of the obvious advantages of the JESS shell is that it provides the knowledge-base containing rules and IE, the basic elements of a KBS. The program written in JESS may consist of rules, facts and objects, where the executed rules are inferred by IE.

4.2.2 Facts and rules

Facts can be either ordered or unordered; they contain a “head” and “slots”. The advantages of ordered facts is that they can be accessed faster. Various functions, such as clear, assert, reset, etc., can be applied to utilize the facts in an appropriate manner. For instance, an example below shows the gas ratios asserted for the power transformer fault diagnosis.

```
( assert (Ratio2 0.00815)
         (Ratio1 6.7)
```

```
                (Ratio5 6.9)
      )
```

where the gas ratio1 is 6.7, the gas ratio2 is 0.00815, and the gas ratio5 is 6.9 ppm.

The left hand side of the rules written in JESS consists of facts, while sequences of function calls are placed in the right hand side. These two side are separated by the characters "=>". The right hand side functions are executed (fired) if the JESS engine matches with the left hand side of rules. In some cases, the fired rule can satisfy the right hand side of the other rules, and consequently these rules will fired. An example of rules written for power transformer fault diagnosis can be presented as follows:

```
( (Gases_Ratio {Ratio2 < 0.1}
               {Ratio1 >= 0.1 && Ratio1 < 1}
               {Ratio5 <= 1})
  =>
  (send "Report: No Fault" ?UserAgent)
  (assert (Transformer CoolerSystem_OFF))
 )
```

The presented rule provides the necessary actions to be performed in case if the facts received meet the rule's condition. In the presented rule two actions are performed: the cooler system should be switched off and the message should be sent in order to notify the user about the fault absence.

Based on JESS characteristic and agent system developed in JADE, the JESS tool can be a good choice of representing knowledge about specific system for interaction with agent system. MAS with KBS can be designed as a combination of the agent, capturing the real-time information from transformer, with the JESS, providing the KBS.

4.3 System Architecture Design

The agent-based architecture developed for automated action and transformer fault diagnosis is shown in Figure 4.1. The proposed system has two applications. First

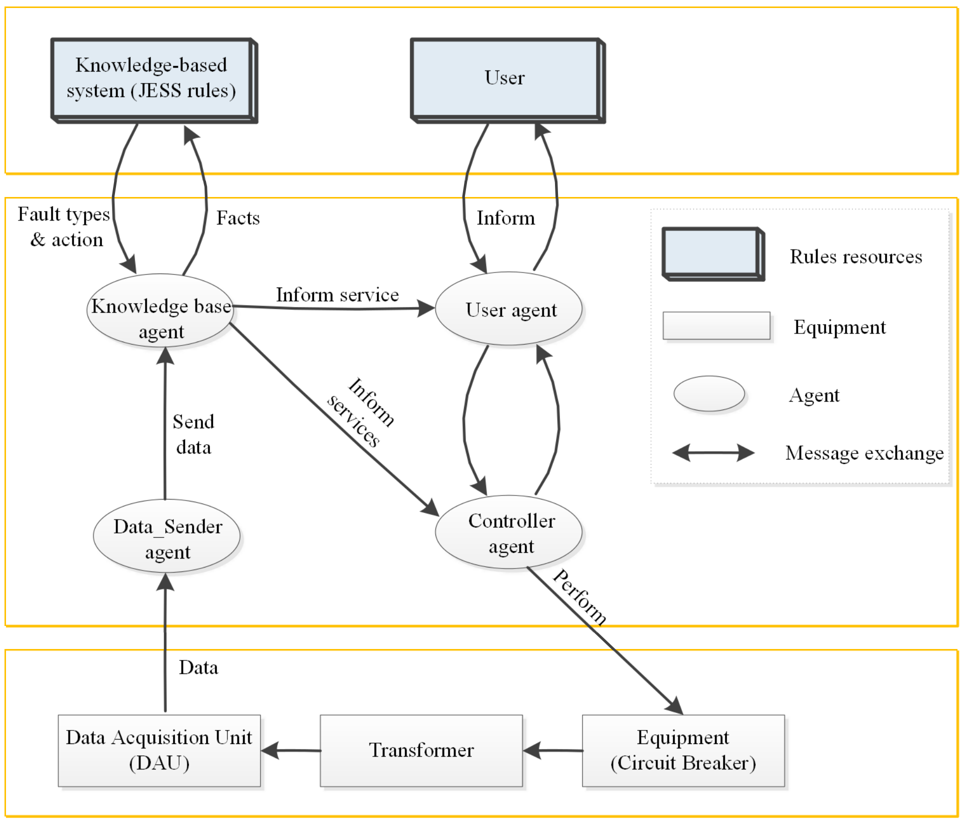


FIGURE 4.1: An agent architecture for transformer fault diagnosis using rule-based reasoning

application is to perform automated actions in case if the fault appears. The second application is to diagnose the power transformer fault, based on DGA samples. For both of these applications the data from power transformer are captured by designed sensors of transformer and collected in DAU. The data is sent to the *Knowledge_base* agent, that wraps the KBS for fault diagnosis. The *Knowledge_base* agent is able to connect to the JESS engine to share the content of the message received (samples gas ratios). JESS contains some information about the power transformer fault diagnosis and the protection components. The *Knowledge_base* agent is always ready to receive a message containing data (facts). The captured facts are passed to the JESS engine and checked for the matching rules to be fired. Based on real-time status of transformer, the required actions are performed by informing the agent responsible for the relevant equipment. One example of used *Knowledge_base* agent is given in the Appendix B.

To use the JESS for automated action, two experiments are applied. First experiment is carried out on power transformer components, such as air cooling system (fan), alarm signal, trip signal, etc. The second experiment informs the relevant circuit breaker to perform some action (open or close).

4.3.1 Automated action in power transformer

In order to implement the rule-based reasoning for performing automated action in power transformer, the IEC publications about a thermal system of power transformer was used. The IEC is an organization that provides the international standards in the domain of electric power system and relevant technologies. Different standards, such as IEC60354 and IEC60905, are published for different types of transformers, such as oil immersed transformer and dry transformer [81, 82]. Thermal model is one of the most essential issues and construction of modelling transformer's temperatures, such as Top Oil Temperature (TOT), Bottom Oil Temperature (BOT), Hot Spot Temperature (HST), etc. This is an important aspect of transformer condition monitoring, that represents the relation between TOT, BOT, HST, etc., and transformer status. According to standard [81], the captured data on temperatures (TOT, BOT, etc.) of power transformer can provide an information about the thermal fault. Figure 4.2 shows the loading and cooling conditions of the winding transformer indicator for recommended transformer rating [9]. As can be seen in the figure, there is a relation between the winding temperature of power transformer and the status of its components. Five possible conditions of the winding temperature must be followed with the relevant actions, which can be defines as five rules. For instance, the winding temperature below 50 °C does not require the cooler system to be turned on. However, for the winding temperature over 120 °C, such components as cooling system, alarm and trip signal must be turned on.

The relation between winding temperature and dependent components can be represented through five rules. The example of the rule for the case of winding temperature over 120 °C is presented here, the complete set rules is given in the Appendix B.

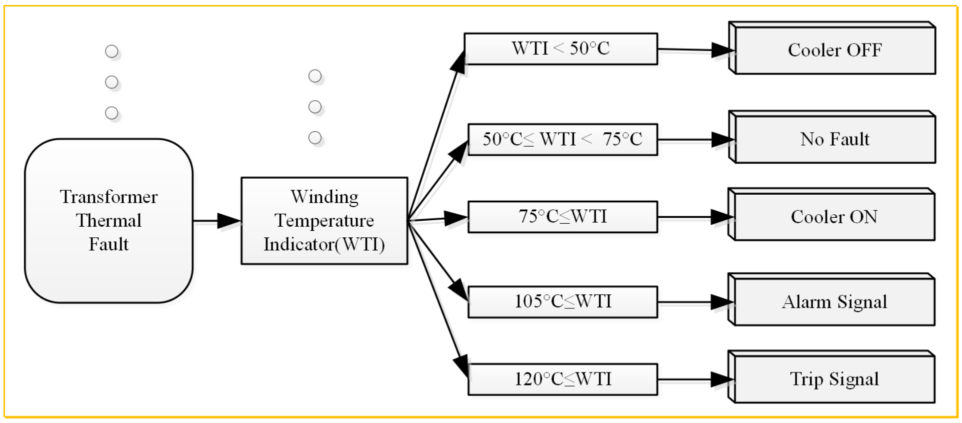


FIGURE 4.2: Loading and cooling conditions for power transformer

```
Rule5:
    (defrule Control5_Component_Temperature
         (Component_Temperature {Winding_Temperature >= 120} (Agent ?agent))
         =>
          (assert (Transformer TripSignal_ON))
         ;(send "ON TripSignal" ?Controller)
          (send "Report: Trip Signal = ON" ?User)
```

The rule requires the trip signal to be turned on if the captured temperature is equal or higher than 120 °C. The trip signal can be turned on by asserting it as a fact, or by sending a message to the relevant (*Controller*) agent to perform the action. It is also required to inform the user that the trip signal is on. It is interesting to note, that in JESS the rules can be fired by the other rules, subject to satisfaction of their conditions. This might be illustrated with an example when the winding oil temperature is 121 °C. As it over the condition provided in fifth rule, the trip signal is turned on. The temperature condition at the same time satisfies the rules 3 and 4, so the relevant rules (3 and 4) are fired and asserted with these two rules are performed (cooler system and alarm signal turn on).

4.3.2 Automated action in power system

In power system an appropriate action is performed by the engineers, based on various information provided, such as SCADA, digital fault recorder, microprocessor-based protection relays, travelling-wave fault locators, etc. [83, 84]. Similarly, the same approach can be used in agent system, with the agents assisting the decision making. The process of decision making and informing about the decisions made is presented in Figure 4.3. In this study it is considered that required action is to be performed subject to the fault presence in power transformer. The operation of power transformer in the case of thermal fault situation may result in solid insulation decomposition causing the loss. Therefore, faulty power transformer should be disconnected from the system and serviced in time to avoid more serious problems.

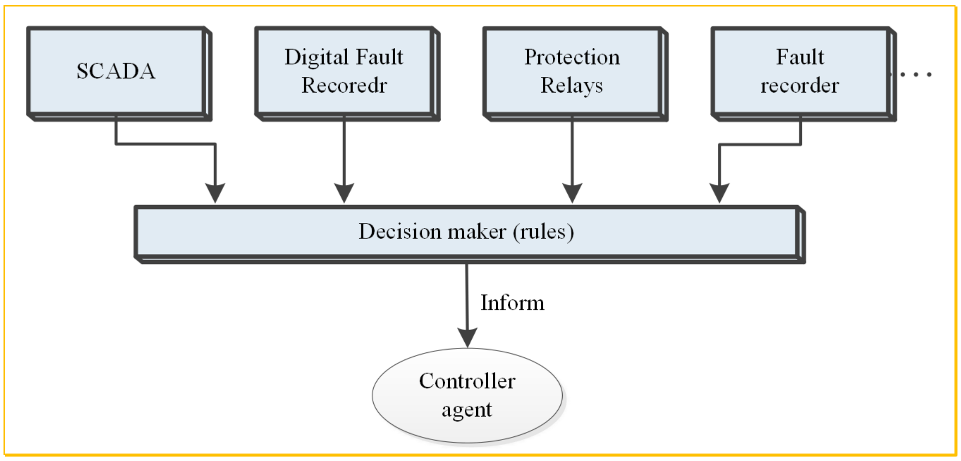


FIGURE 4.3: Process of decision making to perform an action in power system

Similarly to the automatic performance described earlier, the control equipment (e.g. circuit breaker) can be used to protect the system from loss; it can be informed to operate in case if the fault is diagnosed. A transformer fault diagnosis based on Roger's method is applied in JESS for this purpose. According to the information presented in Table 2.3, the status of fault situation can be covered with the help of nine rules. In the case if the gas ratios are not in the regions defined by Roger's method, an additional rule is required. The applied fault diagnosis method with nine types of faults is based on three gas ratios (R_2, R_1, R_5). For instance, in case if $R_2 < 0.1$, $1 \leq R_1$ and $3 < R_5$, the fault

"Thermal fault with temperature over 700 °C" is discovered. The rule representing this status is given below.

```
;Rule_Case9:
 (defrule Transformer_Fault_Diagnosis_C9_Rogers_Method
   (Gases_Ratio {Ratio2 < 0.1} {Ratio1 >= 1} {Ratio5 > 3} (Agent ?agent))
   =>
   (send "Open CB1" ?Controller)
   (send "Report: Thermal Fault(TF) TF>700 Celsius degrees" ?User)
```

Fault types can be diagnosed in case if DGA ratios samples satisfy the rule's conditions. To illustrate this application, the performance of the circuit breaker $CB1$ based on presence the fault in power transformer $SGT1$ is presented in Figure 2.9. The developed agent system senses the transformer $SGT1$ status, and in the case fault appears, the $CB1$ will be informed to operate, as shown in Figure 4.4.

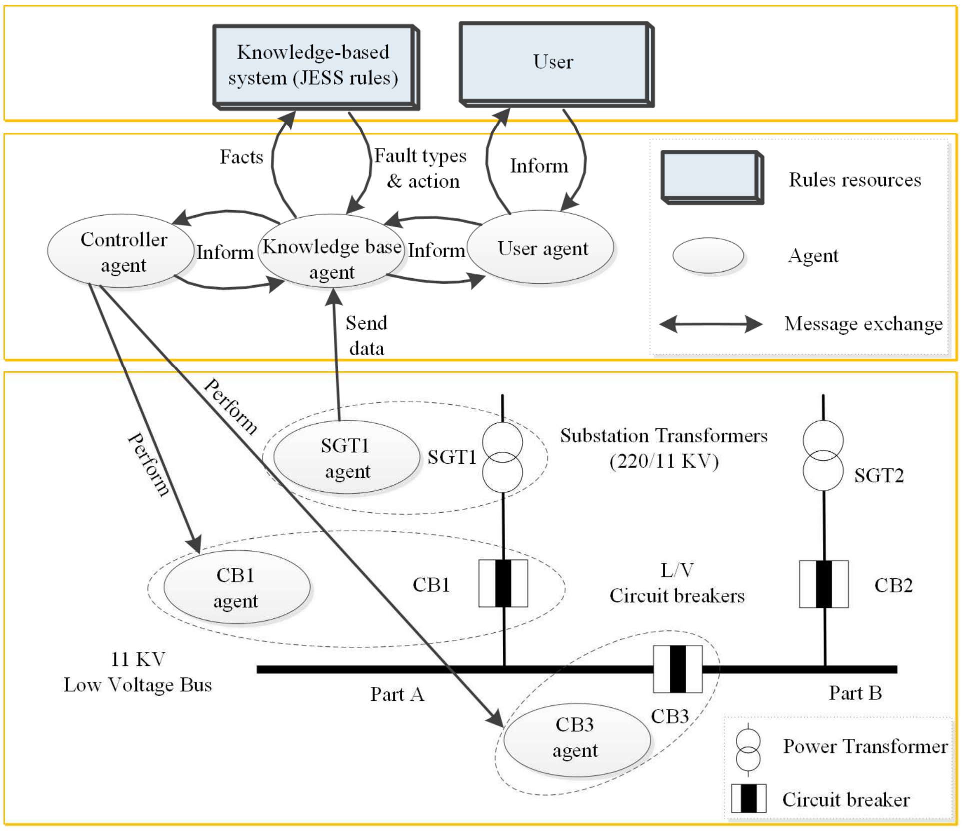


Figure 4.4: Automated operation in power system

Figure 4.4 shows that there is an agent denoted as $\mathcal{A}_{SGT1}$, that wraps the transformer $SGT1$ for the purpose of capturing real-time information (can use the data from DAU). Similarly, agents $\mathcal{A}_{CB1}$ and $\mathcal{A}_{CB3}$ wrap the circuit breakers $CB1$ and $CB3$ respectively. In our case, the real-time gas ratio samples are sent from the $\mathcal{A}_{SGT1}$ to the *Knowledge base* agent, denoted as $\mathcal{A}_{KB}$, for fault diagnosis. The $\mathcal{A}_{KB}$ utilizes the JESS engine and applies the gas ratios from received message to diagnose the fault. *Controller* agent, denoted as $\mathcal{A}_{CNT}$, receives request for performing an appropriate actions towards the messages. The $\mathcal{A}_{CNT}$ sends the request to the $\mathcal{A}_{CB1}$ to perform the action, and after the action is performed, the $\mathcal{A}_{CNT}$ receives a notification. The agent $\mathcal{A}_{CB3}$ acts similarly regarding circuit breaker $CB3$. The results of the fault types and the actions performed are supplied to the user.

4.3.3 Rule-based reasoning for transformer fault diagnosis

Application of MAS with JESS for purpose of fault diagnosis, based on the use of Roger's method, is investigated in this section of work. The developed agent architecture used for fault diagnosis task is presented in Figure 4.5. The data applied containing three gas ratios (R_1, R_2 and R_5) is written as a text file. *Data_Sender* agent, denoted as $\mathcal{A}_{DS}$, is able to read each row of text file data and create a message. The message containing the gas ratios is sent to the $\mathcal{A}_{KB}$. The $\mathcal{A}_{KB}$ utilizes the JESS engine and fires the matching rules. Thus, the fault types are diagnosed, and $\mathcal{A}_{KB}$ is informed. The information on the fault types is sent to the *Data_Collector* agent ($\mathcal{A}_{DC}$) to be saved in the database, and also sent to the *User* agent ($\mathcal{A}_{USER}$) to be reported to the user.

4.4 An Example of Agent Communication Act

In this section an example of implementing the KBS for performing an automated action is investigated. According to the distribution substation, presented in Figure 2.9, an automated action can be performed in terms of fault diagnosis. In this example we use the actual DGA samples captured on-line from the transformer $SGT1$. For this purpose, the real DGA samples obtained from [57] are used, as shown in the Table 4.1. The data contains two categories: "*No Fault*" and "*Thermal Fault*".

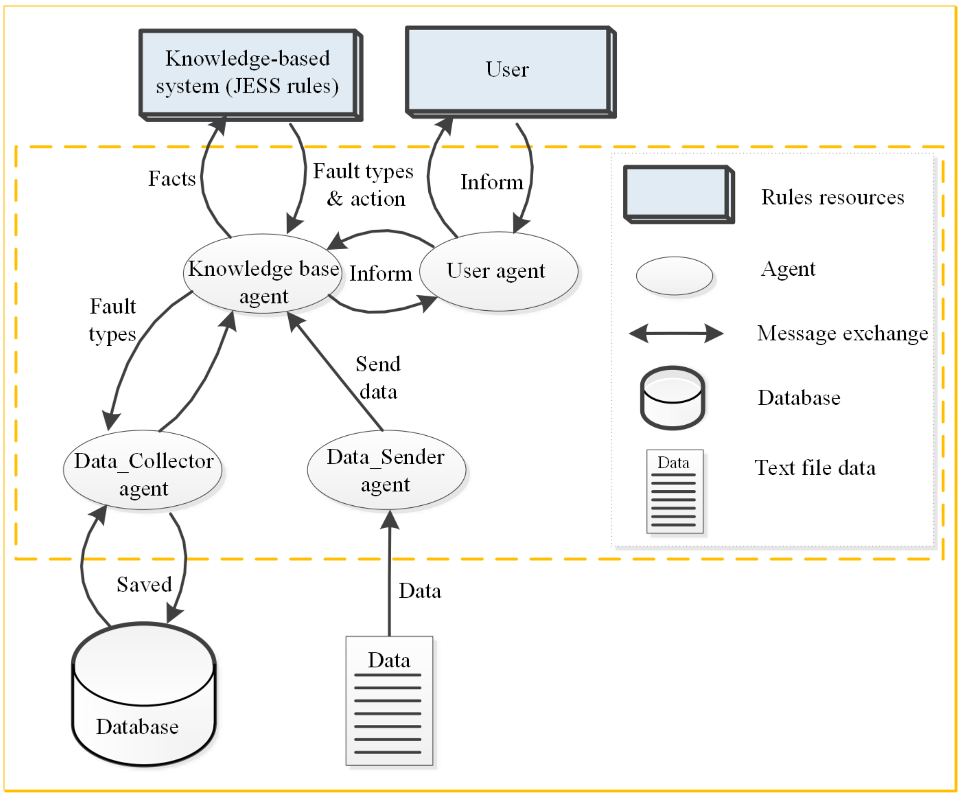


FIGURE 4.5: Experimental MAS for fault diagnosis, based on rule-based reasoning

As shown in overview of the system performance in Figure 4.4, the communication act is initiated by agent $\mathcal{A}_{SGT1}$. The $\mathcal{A}_{SGT1}$ sends messages containing the gas ratios, as given in Table 4.1. The agent $\mathcal{A}_{KB}$, as a receiver agent, is able to access the KBS. The Roger's method, as an example of fault diagnosis method is applied for this purpose. In real life situation user makes a decision to undertake an appropriate action based on the information obtained. In this case we assumed that in terms of "*No Fault*" status, the power transformer $SGT1$ should remain in its normal operation mode, while in case of "*Thermal Fault*" situation it should be disconnected from the power system, and user should get informed. In terms of JESS rules, it is also possible to send a message directly to the $\mathcal{A}_{CNT}$. The $\mathcal{A}_{CNT}$ is able to search the DF to find an appropriate agents to provide the services required, and inform them. To simplify the case of our study and reduce the agents' communication involved, we assumed that the $\mathcal{A}_{CNT}$ has already informed the service provider agent. The service provider is the agent $\mathcal{A}_{CB1}$, that wraps the circuit breaker $CB1$ and able to perform the action (open or close $CB1$). Thus, the action required in particular situation can be performed.

TABLE 4.1: The actual DGA samples applied for performing an automatic action

Data Samples	H_2	CH_4	C_2H_2	C_2H_4	C_2H_6	$Ratio_2$	$Ratio_1$	$Ratio_5$	**Actual Fault**
Sample 1	150	120	1	40	130	0.025	0.8	0.31	No Fault
Sample 2	1270	3450	8	1390	520	0.006	2.72	2.67	Thermal Fault
Sample 3	360	610	9	260	259	0.035	1.7	1.01	Thermal Fault
Sample 4	960	4000	6	1590	1290	0.004	4.17	1.23	Thermal Fault

The communication acts between relevant agents can be initiated by $\mathcal{A}_{SGT1}$ sending the gas ratios. The $\mathcal{A}_{SGT1}$ is presented as "$SGT1$@192.168.1.66 : 1099/$JADE$" with ability of reading the gas ratios from the Table 4.1 and send them to the $\mathcal{A}_{KB}$ shown at "$Knowledge - based$@192.168.1.181 : 6126/$JADE$". The following message is used for sending the first gas ratios:

```
( INFORM
   :sender (agent-identifier
        :name SGT1@192.168.1.66:1099/JADE)
   :receiver (set (agent-identifier
        :name Knowledge-based@192.168.1.181:6126/JADE))
   :content ‘‘(( set (Ratio2 0.025)
                      (Ratio1 0.8)
                       (Ratio5 0.31)))’’
   :protocol fipa-request
)
```

The $\mathcal{A}_{KB}$ is designed to await for the message to be received, get the message content and then run the JESS engine. The JESS engine contains rules corresponding to the Roger's method for fault diagnosis. Two related rules associated to the test conditions are given as follows:

```
;********Rules.clp**********
  ;Rule0:
    (defrule Transformer_Fault_Diagnosis_C0_Rogers_Method
      (Gases_Ratio {Ratio2 < 0.1} {Ratio1 >= 0.1 && Ratio1 <= 1}
      {Ratio5 <= 1} (Agent ?agent))
     =>
      (send (ACLMessage (communicative-act INFORM)
            (sender Knowledge-based@192.168.1.181:6126/JADE)
            (receiver User@pc2214:1099/JADE) (conversation-id ?cid)
            (content NO_Fault)))
       )
    ...
    ;Rule5:
      (defrule Transformer_Fault_Diagnosis_C5_Rogers_Method
      (Gases_Ratio {Ratio2 < 0.1} {Ratio1 >= 0.1 && Ratio1 <= 1}
      {Ratio5 >= 1 && Ratio5 <= 3} (Agent ?agent))
     =>
      (send (ACLMessage (communicative-act INFORM)
            (sender Knowledge-based@192.168.1.181:6126/JADE)
            (receiver User@pc2214:1099/JADE) (conversation-id ?cid)
            (content Thermal_Fault<150)))
      (send (ACLMessage (communicative-act REQUEST)
            (sender Knowledge-based@192.168.1.181:6126/JADE)
            (receiver Controller@192.168.1.184:1428/JADE)
            (conversation-id ?cid) (content open CB1)))
       )
    ...
```

The first DGA sample from the Table 4.1 correspond to *No Fault* condition, therefore there are no actions required to be performed. In this situation the user will be informed that there is no fault in the $SGT1$ transformer. For the case of the DGA samples indicating the presence of some *Thermal Fault* (samples 2, 3, 4), the situation is different, and an appropriate actions should be undertaken. Appropriate actions in this case are defined as opening the circuit breaker $CB1$ (and also can close the circuit breaker $CB3$)

at first and then informing the user about the fault type occurred. The system is aimed to disconnect the transformer from the circuit in the case of *Thermal Fault*.

Thus, the rule five will be fired, and the user will be informed about the fault type. A message will also be sent to the $\mathcal{A}_{CNT}$ to request the appropriate action to be performed. According to previous works, the agent system can use ontology agent for the purpose of knowledge representation in power system. This have been applied previously in [6], and can be applied to the present work in the same way. For the case of using the ontology agent, the real-time information of the power system (e.g. $CB1$ open or close) and its components is presented. To simplify the example, we assume that the $\mathcal{A}_{CNT}$ already knows that the $\mathcal{A}_{CB1}$ is able to perform the action requested. The request message for opening the $CB1$ will be sent to the $\mathcal{A}_{CB1}$, located at $CB1@192.168.1.194 : 1099/JADE$, with the following request message:

```
( REQUEST
   :sender (agent-identifier
        :name Controller@192.168.1.184:1428/JADE)
   :receiver (set (agent-identifier
        :name CB1@192.168.1.194:1099/JADE))
   :content ‘‘((action (agent-identifier
          :name CB1@192.168.1.194:1099/JADE)
          (open CB1)))’’
   :protocol fipa-request
)
```

The agreement will be returned to the *Controller* agent in the case of consulting the $\mathcal{A}_{CB1}$ with device in a form of the following message:

```
( AGREE
   :sender (agent-identifier
        :name CB1@192.168.1.194:1099/JADE)
   :receiver (set (agent-identifier
        :name Controller@192.168.1.184:1428/JADE))
```

```
    :protocol fipa-request
)
```

Finally, the action permission is granted and the circuit breaker $CB1$ switches off. An informing message is sent to the $\mathcal{A}_{KB}$ to confirm the status of circuit breaker $CB1$ (opened):

```
( INFORM
    :sender (agent-identifier
         :name CB1@192.168.1.194:1099/JADE)
    :receiver (set (agent-identifier
         :name Knowledge-based@192.168.1.181:6126/JADE))
    :content ‘‘((done(action (agent-identifier
           :name CB1@192.168.1.194:1099/JADE)
           (open CB1))))’’
    :protocol fipa-request
)
```

This a simple example of communication messages for performing an automated action in MAS to reduce the human efforts.

4.5 The Experiment Results for Fault Diagnosis

The case study described in this section was developed to evaluate the performance of the KBS for fault diagnosis. For this purpose, the practically obtained data and their actual faults are used, as listed in Table 4.2. All of these data and results of actual inspection were published in [57, 85, 86]. The actual data contain one case of no fault, 2 cases of partial discharge, 14 cases of arcing, 25 cases of overheating and 28 cases of low energy discharge, giving the total of 70 DGA samples investigated.

In this study, the Roger’s method is chosen for the purpose of fault diagnosis to interact with agent system, as shown in Figure 4.5. The data are sent to the $\mathcal{A}_{KB}$ by

Table 4.2: Actual gas ratio samples from power transformer

Fault types	**Number of samples**
No Fault	1
Partial Discharge	2
Arcing	14
Overheating	25
Low Energy Discharge	28
Total	**70**

Data_Sender agent, denoted as $\mathcal{A}_{DS}$. The $\mathcal{A}_{KB}$ uses the JESS rules to diagnose the fault. The results of fault diagnosis are sent to the *Data_Collector* agent to be saved in the database. According to the actual data applied and the fault diagnosis method, the results corresponding to the situation where fault type could not be defined are shown in Table 4.3 as "ND". This happens in the case where ratios of the samples do not match with any of the regions of the Roger's method. In this case using different methods of knowledge-based system (fuzzy ontology) can improve the accuracy of the fault diagnosis (as investigated in Chapter 6). In practical situation for the case of undefined the fault types, an expert needs to apply his experience to diagnose the faults. Table 4.3 presents 20 data samples diagnosed with the help of rule-based reasoning.

The results of rule-based reasoning for fault diagnosis with developed agent system are summarised in Table 4.4. This way of presenting the results makes them more convenient for discussion and comparison with further work based on using of identical DGA samples (the results can be improved by use of fuzzy ontology).

As shown in Table 4.4, such fault cases as "No Fault", "Overheating" and "Low Energy Discharge", are diagnosed correctly, while in the case of "Arcing" and "Partial Discharge", the fault type could not be identified. In this study, the data applied consisted of more then 70 samples, but in some cases the fault types were not identified correctly. 70 data samples where the fault types could be identified correctly were used for various methods application, such as ontology and fuzzy ontology.

TABLE 4.3: The actual DGA samples applied with rule-based reasoning (JESS)

R_2	R_1	R_5	**Actual Fault**	**JESS Agent Results**	**Results**
1.16	0.46	5.2	Arcing	Low Energy Partial Discharge, Arcing, Sparking	Correct
0.07	5.43	5.26	Overheating	Thermal Fault(TF) $TF > 700°$C	Correct
1.65	0.17	3.13	Arcing	Low Energy Partial Discharge, Arcing, Sparking	Correct
1.06	1.74	9.26	Arcing	Undefined Fault	ND*
0.04	3.86	6.94	Overheating	Thermal Fault $TF > 700°$C	Correct
0.97	1.79	7.06	Arcing	Undefined Fault	ND*
0.01	40.99	5.07	Overheating	Thermal Fault $TF > 700°$C	Correct
3.25	0.08	17.75	Partial Discharges	Undefined Fault	ND*
0.02	3.09	7.44	Overheating	Thermal Fault $TF > 700°$C	Correct
0.01	1.42	10.02	Overheating	Thermal Fault $TF > 700°$C	Correct
2.74	1.54	13.42	Arcing	Undefined Fault	ND*
0.01	2.69	8.62	Overheating	Thermal Fault $TF > 700°$C	Correct
2.93	0.09	6.6	Arcing	Undefined Fault	ND*
2.26	0.29	10.82	Arcing	Low Energy Partial Discharge, Arcing, Sparking	Correct
3.42	0.08	5.6	Partial Discharges	Undefined Fault	ND*
0.02	2.39	7.16	Overheating	Thermal Fault $TF > 700°$C	Correct
3.3	0.07	16.5	Arcing	Undefined Fault	ND*
0.02	2.4	6.7	Overheating	Thermal Fault $TF > 700°$C	Correct
0	4.85	1.85	Overheating	Thermal Fault $300 < TF < 700°$C	Correct
1.45	0.84	14	Arcing	Low Energy Partial Discharge, Arcing, Sparking	Correct

* ND = Not Defined

TABLE 4.4: Summary of rule-based reasoning for fault diagnosis

Fault Types	**Total Samples**	**Diagnosed Correctly**	**Not Defined Fault**	**Accuracy**
No Fault	1	1	0	100%
Partial Discharge	2	0	**2**	**0%**
Arcing	14	9	**5**	**64.3%**
Overheating	25	25	0	100%
Low Energy Discharge	28	28	0	100%
Average Total Accuracy	—	—	—	**72.86%**

The rule-based reasoning finds various applications, such as detecting power system failure based on fuzzy rule [87], fuzzy algorithm for power transformer diagnosis [88], fuzzy rule set for fault diagnosis [89], expert system for transformer fault diagnosis based on DGA [90], etc. For instance, in [90] the defined fuzzy membership functions are applied in the expert system to handle uncertain norm threshold. Further information on application of fuzzy system is given in the Section 6.1.1. However the difficulty of rules tracking for the large knowledge base may be named as one of the disadvantages of the rule-based reasoning. Structural knowledge (knowledge about cause and effect) cannot be easily handled with the rule-based reasoning [75]. On the contrary, different method of knowledge representation, such as ontology-based reasoning, can provide a comprehensive knowledge of the system, as will be discussed in the following chapter.

4.6 Summary

A MAS with rule-based reasoning for performing action and fault diagnosis based on DGA samples is presented in this chapter. Two types of applications are investigated, such as performing an automatic action for individual component of power system (e.g. power transformer) and fault diagnosis of power transformer. The rule-based reasoning is applied in the JESS program, which is able to interact with the agent system. The JESS engine provides the specific knowledge about system, performs automated action and capable of fault diagnosis. Various type of agents are designed, and their communication messages are detailed. Furthermore, practical DGA data samples are utilized for investigation of the fault diagnosis, based on use of MAS. The results of fault diagnosis are presented and possible future improvements are discussed.

Chapter 5

Ontology-based Fault Diagnosis for Power Transformer

5.1 Introduction

The ability to be accessible to the other applications of key importance for the knowledge-based systems. Extensible Markup Language (XML) is the basic type of knowledge representation, which provides a syntax designed to be readable by both machines and humans. Its drawback is that the represented knowledge in particular domain is not understandable to the other applications (e.g. software agent). Semantic Web (SW) provides the meaning layer to the World Wide Web (WWW) to make it machine understandable. Based on SW and knowledge representation, ontology provides the mechanism, that assigns the semantics to the web. Ontology also can be used by different software applications [91]. Therefore, the Description Logic (DL) as the main feature of ontology is discussed here first.

5.1.1 Description Logic

Knowledge can be represented in the form of logic. In AI various types of formal logic are proposed for knowledge base representation. One of the formal deductive systems, called First Order Logic (FOL), permits to predict and quantify the propositional logic. For instance, FOL uses some variables to describe the notation of “all the Mercedes-Benz are cars" as follows:

$$\forall n(Mercedes - Benz(n) \rightarrow Car(n))$$

Being a suffocated logic is one of the FOL advantages, as well as an ability to capture the most of the natural language. Its drawback is that the higher order functions are not directly expressible in FOL.

The DL is a subset of the FOL, another family of knowledge representation, which can be used to represent the knowledge of an application domain in a structured and formally well-understood way [92]. Many types of DL, such as $\mathcal{ALC}$, $\mathcal{SHIQ}$, $\mathcal{ALCNIO}$, $\mathcal{SHOIN}$, etc., can describe their operation with different attributes. A huge number of shared properties and logic-based knowledge representation formalisms are formed (in DL) to precise the DL definition. The main components of DL are itemized below:

- **Concept (C)** represents an abbreviation of the objects in the world;
- **Roles (R)** are binary relationships between set of concepts;
- **Functions (F)** are defined over the concepts to return a concept;
- **Axioms (A)** are true assertions, that impose the definition of concepts, rules, etc.;
- **Individuals (I)** are instances of the concepts, which also correspond to the concepts subset.

In summary, the DL describes a domain of interest in terms of concepts, roles and individuals. Concepts (or classes) and roles (or properties) in DL are the building constructors, such as conjunction, disjunction, negation, etc., that can be varied depending on DL types. An overview of the most important constructors is given in Table 5.1, where the first column represents the name of constructor, and its syntax is provided in the second column.

The DL architecture consists of two parts, namely terminological box (TBox) and assertion box (ABox), which are represented as a reasoner system. The TBox contains intentional knowledge, which means the abbreviation (name) and schema for a complex description. For example,

$$Car \equiv Vehicle \sqcap Engine$$

TABLE 5.1: Summary of constructors in description logic

Constructor Name	Syntax	Semantic	Symbol
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	$\mathcal{AL}$
Disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	$\mathcal{U}$
Negation	$\neg C$	$\Delta^{\mathcal{I}} - C^{\mathcal{I}}$	$\mathcal{C}$
Value restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \in xR^{\mathcal{I}} : y \in C^{\mathcal{I}}\}$	$\mathcal{A}$
Exists restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in xR^{\mathcal{I}} : y \in C^{\mathcal{I}}\}$	$\mathcal{E}$
Nominals	$\{O_1, ..., O_n\}$	$\{O_1^{\mathcal{I}}, ..., O_n^{\mathcal{I}}\}$	$\mathcal{O}$
Unqualified number restriction	$\geq nR$	$\{x \in \Delta^{\mathcal{I}} \Vert xR \mid \geq n\}$	$\mathcal{N}$
Unqualified number restriction	$\leq nR$	$\{x \in \Delta^{\mathcal{I}} \Vert xR \mid \leq n\}$	$\mathcal{N}$
Qualified number restriction	$\geq nR.C$	$\{x \in \Delta^{\mathcal{I}} \Vert xR \cap C^{\mathcal{I}} \mid \geq n\}$	$\mathcal{Q}$
Qualified number restriction	$\leq nR.C$	$\{x \in \Delta^{\mathcal{I}} \Vert xR \cap C^{\mathcal{I}} \mid \leq n\}$	$\mathcal{Q}$

defines that car is a type of vehicle, and it has an engine. The ABox contains extensional knowledge, which represents the data of complex description. For example,

$$BMW : Car$$

$$(BMW, X5) : hasModel$$

defines that BMW is a car, and BMW has a model called $X5$.

The DL semantics is defined in a model-theoretic way, one central notion is that of an interpretation. The interpretation is a structure $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where the $\Delta^{\mathcal{I}}$ is the domain and $\cdot^{\mathcal{I}}$ is an interpretation function. Every concept name A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ with semantic ($A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$), and every role name r to a binary relation $r^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$ with semantic ($r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$) [92]. For instance, the interpretation can be described with following example, where the TBox contains two axioms:

$$Mercedes - Benz \sqsubseteq Car$$

$$BMW \sqsubseteq Car$$

The semantics are interpreted as follows:

$$Mercedes - Benz^{I} \subseteq Car^{I}$$

$$BMW^I \subseteq Car^I$$

Individuals, such as *Mercedes-Benz* and *BMW*, are the subsets of *Car*, this can be represented with Venn diagram, as shown in Figure 5.1.

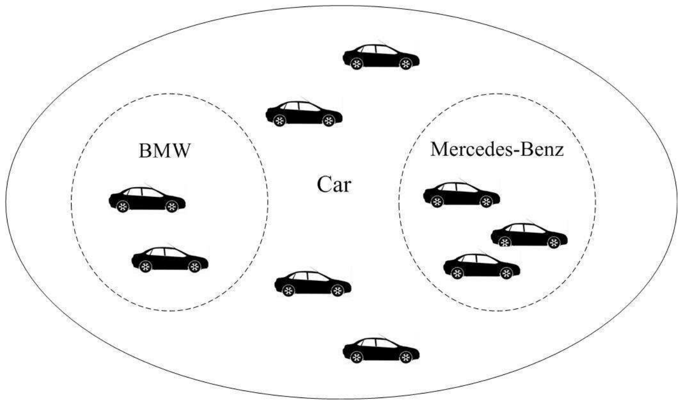


FIGURE 5.1: Description logic interpretations represented with Venn diagram

The reasoning services in DL provide the automatic deduction of implicit knowledge from the explicit represented knowledge. For this purpose various algorithms can be implemented, some examples of them are presented below [92]:

- *Subsumption* algorithm determines the subconcepts and superconcept relationships: C is subsumed by D if all instances of C are necessarily instances of D. For example, $BMW \sqsubseteq Car$ where the BMW is a subclass of the class Car;
- *Instance* algorithm determines instance relationships: the individual i is an instance of the concept description C, if i is always interpreted as an element of the interpretation of C. For example, $X5$ is an instance of class *BMW*, as presented with *BMW*(X5);
- *Consistency* algorithm determines whether a knowledge base, consisting of the TBox and ABox, is non-contradictory. For example, if all the cars can be defined as either sport or four wheel drive, and the $X5$ is an instance of *BMW* class and it is not sport car, then we can not say that $X5$ is not four wheel drive, which represents its inconsistency.

DL finds various applications, such as software information and documentation, databases, query answering, ontology languages, etc. [91]. The Ontology Web Language (OWL) is one of the most important applications of DL, where various tools and reasoning techniques are widely been used. This will be discussed in further sections. This application of the DL is perhaps one of the most prominent applications used [92].

5.1.2 Ontology

There are various definitions given for ontology, one of the commonly used in computer science is [93]:

"Ontology is a formal, explicit specification of a shared conceptualization."

In this definition, the specification is a formal description of how something could be constructed to meet certain criteria, while the conceptualization is required to use the ontology language. It is involves the computer symbols with the individuals and relations in the world. It provides a particular abstraction of the world and notation for that abstraction. For example, the propositional logic formula

$$X(n) \rightarrow Y(n)$$

is an abstract, and can be grounded into a domain as

$$Mercedes - Benz(n) \rightarrow Car(n)$$

where the interpretation of it can be shared and utilized in particular procedure.

The main components of ontology are itemized as follows:

- **Classes** or **Concepts**;
- **Properties** or **Roles**;
- **Axioms**.

Many programming languages are developed for building an ontology. Resource Description Framework (RDF) is the one of the basic ontology languages, which allows to build a simple hierarchy of the concepts and properties. OWL is extended from

RDF; it is one of the standard ontology languages recommended by W3C [94]. OWL is a powerful ontology with RDF's abilities, and additionally capable to describe the concepts in complex situations. The key feature of OWL is that it can be used not only for presenting the information, but also to process the presented information and to extract the new information. This point makes it possible to use the OWL for various applications, such as knowledge sharing and representation, semantic web, information system, ontology-based reasoning, etc. An example of OWL representation and its explanation are given below. Let us consider the two axioms represented in the following TBox:

$$Mercedes - Benz^I \subseteq Car^I$$

$$BMW^I \subseteq Car^I$$

The OWL file can be represented in the following way:

```
<owl:Class rdf:about="#Car">
  </owl:Class>
    <owl:Class rdf:about="#Mercedes-Benz">
<rdfs:subClassOf rdf:resource="#Car"/>
  </owl:Class>
    <owl:Class rdf:about="#BMW">
  <rdfs:subClassOf rdf:resource="#Car"/>
</owl:Class>
```

which represents the class *Car* has two subclasses *Mercedes-Benz* and *BMW*. The graph representing this ontology is given in Figure 5.2, where the various subclasses of class *Thing* are shown.

OWL uses a DL expression called $\mathcal{SHIQ}$ to express the ontology language and reasoner such as FaCT++, RACER and Pellet [95]. The $\mathcal{SHIQ}$ provides various features, with ability of more expression compared to the other types of DL (e.g. $\mathcal{ALC}$). The expressions provide several features and attributes, such as qualified number restrictions, inverse roles, transitive roles, sub-roles, etc., which easily formulate the complex terminological axioms. For instance, the qualified number restrictions in the $\mathcal{SHIQ}$ help to

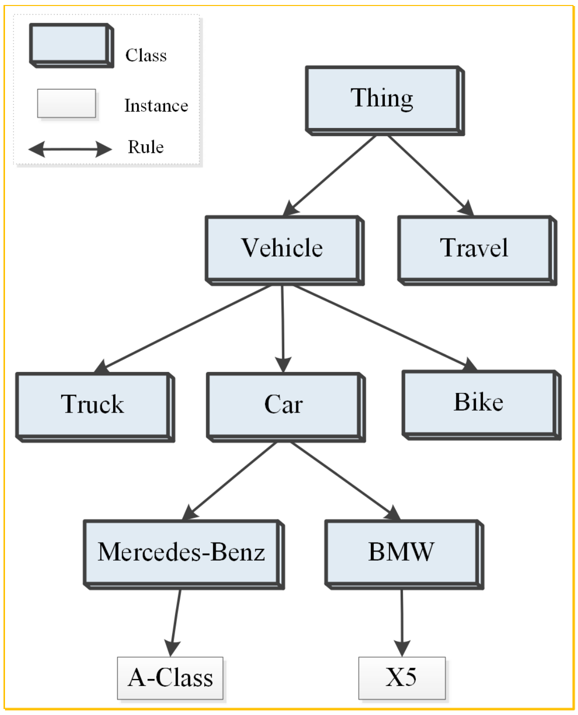


FIGURE 5.2: Graphical representation of an ontology

present the statement that a car most often has five doors:

$$\leq 5 \; hasDoor.$$

OWL-Lite, OWL-DL and OWL-Full are three categories of OWL with different features. OWL-Lite gives simple constraint features, while OWL-Full provides the maximum expressiveness. OWL-DL corresponds to DL and supports the maximum expressiveness without losing computational completeness [92]. A summary of constructors supported by the OWL is given in Table 5.2 [96].

The syntax language for concepts are defined as follows: *concept* names (C_0, C_1, ...), *property* names (P_0, P_1, ...), concept constructor "$\sqcup$" called *unionOf*, *disjunction* or *or*, concept constructor "$\sqcap$" called *intersectionOf*, *conjunction* or *and*, concept constructor "$\exists$" called *existential restriction* constructor, "$\forall$" called *value restriction* constructor and so on. For instance, the concept of fault diagnosis for power transformer is defined as: "*A transformer has a thermal fault in its component, and the symptoms are either temperature or gases*". The description of this concept can be defined using the following

Table 5.2: Summary of OWL constructors

Constructor	DL Syntax	Example
intersectionOf	$C_1 \sqcap ... \sqcap C_n$	Vehicle $\sqcap$ Car
unionOf	$C_1 \sqcup ... \sqcup C_n$	Car $\sqcup$ Bike
complementOf	$\neg C$	$\neg$ Car
oneOf	$\{x_1 ... x_n\}$	{BMW, Mercedes}
allValuesFrom	$\forall r.C$	$\forall$ hasEngine.BMW
someValuesFrom	$\exists r.C$	$\exists$ hasEngine.Vehicle
hasValue	$\exists r.\{x\}$	$\exists$ madeIn.{Germany}
minCardinality	$\geq nr$	($\geq$3 hasDoor)
maxCardinality	$\leq nr$	($\leq$ 1 hasEngine)
inverseOf	r'	hasEngine$'$

DL:

$$TransformerComponent \sqcap (\exists hasFault.ThermalFault) \sqcap$$

$$(\forall hasSymptom.(Temperature \sqcup GasRatios))$$

OWL2 language is newer version of OWL, equivalent to DL $\mathcal{SROIQ}$(D) [96]. Compared to OWL, OWL2 provides more functionality including property chains, keys, richer datatypes, data ranges, disjoint properties, etc. [94].

5.1.3 Reasoners

The OWL ontology consists of classes, properties and individuals. Hierarchy classes in ontology are formed with superclasses and subclasses. Different types of properties, such as inverse, functional, inverse functional, transitive, symmetric, asymmetric, reflexive and irreflexive properties, provides restriction to the built ontology. Inference problems could be performed by some reasoning algorithms. Tableau algorithm is one of the most widely used techniques for reasoning in DL [92]. It is used to prove a decidability or computational complexity result. A reasoner allows the inference to be made, based on the construction of compositional concepts and roles. Precisely the concept D is subsumed by concept C, when all instances of the D are also instances of C. Various reasoners can be used, such as Racer [97], FaCT [98], FaCT++ [99] and Pellet [100]. The differences between reasoners are the types of algorithms used and the way they are implemented in reasoning tasks.

One powerful approach of building ontology-based reasoning is to use such reasoning techniques [101]. An ontology reasoner deduces the queried results from the ontology knowledge base. Therefore, an Application Programming Interface (API) is required, for accessing the reasoner to the ontology. OWL APIs with various interfaces provide accessibility to OWL reasoners [102].

5.2 System Architecture Design

Seven steps required to design an appropriate ontology are given in [94]. These steps are recommended for gradual design of the hierarchy classes, properties, individuals, etc., to form an ontology. Based on these recommendation, the ontology-based transformer fault diagnosis is developed for the agent framework. The proposed system consists of two parts. An ontology for power transformer fault diagnosis is developed at the first step; and then the agent system is designed to wrap the ontology and interact with it. The developed ontology is able to query the fault types, using the information of given symptoms. The ontology is also able to define the relevant information of the faulty components of power transformer. The interaction between developed ontology and power transformer is carried out through the developed agent, who sends the real-time symptoms (for the case study using the DGA gas ratios) to the ontology. The overall architecture of transformer fault diagnosis is shown in the Figure 5.3.

As can be seen in the Figure 5.3, an agent called *Ontology*, wraps the ontology for transformer fault diagnosis. *Ontology* agent is able to communicate with other agent (in this case *Data_Sender* agent), in order to establish an interaction with the other agents.

5.2.1 Ontology for power transformer fault diagnosis

As the transformer is one of the most important units in power system, its reliability is a prime concern in power system operation. Real-time condition monitoring and fault diagnosis of the power transformer help to improve its reliability and prevent more serious problems. A fault diagnosis system with ontology-based reasoning provides a comprehensive knowledge base, which can be utilized by other application. For this purpose, an on-line fault diagnosis system based on ontology reasoning is developed.

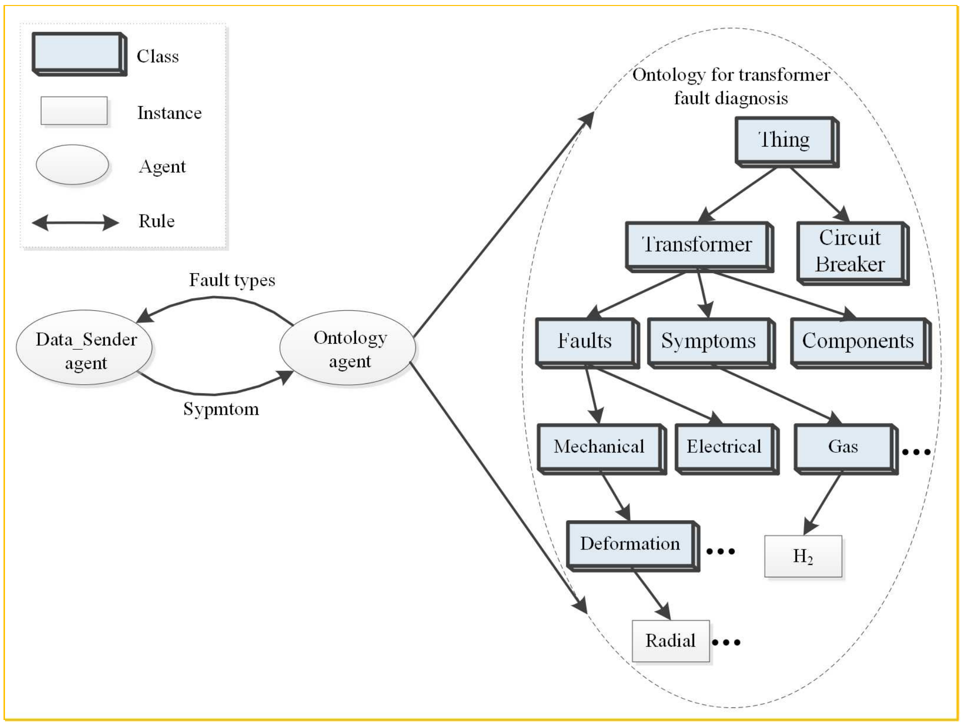


FIGURE 5.3: MAS for power transformer fault diagnosis based on ontology reasoning

A fault typically appears in a power transformer during its operations, however, more likely, any type of faults could change the working status of transformer, which is obviously reflected in some symptoms related to the fault. This is similar to the concepts of cause and effect, where the cause is the event that has relation with its effect, known as phenomenon. In other word, any type of fault has some relevant symptoms. Knowing symptoms enables the identification of relevant fault types. For instance, a cooling system (fan) in a power transformer is used to dissipate heat to its external surrounding. A fault can affect the working status of a cooling system, which may lead to malfunction of its correct performance (e.g. fan stops working). This results in abnormal oil temperature increase, presented as a symptom. Thus, the temperature increase may indicate a problem of a cooling system, and later cause arcing.

In reality, because of complexity of transformer fault mechanism, there are many types of symptoms and faults. To build an appropriate ontology for power transformer fault diagnosis, three different categories are defined, namely fault, symptom and component. A summary of faults is collected in the fault category, restricted by some types of properties to the symptom category. For instance, fault A has the symptom B, thus

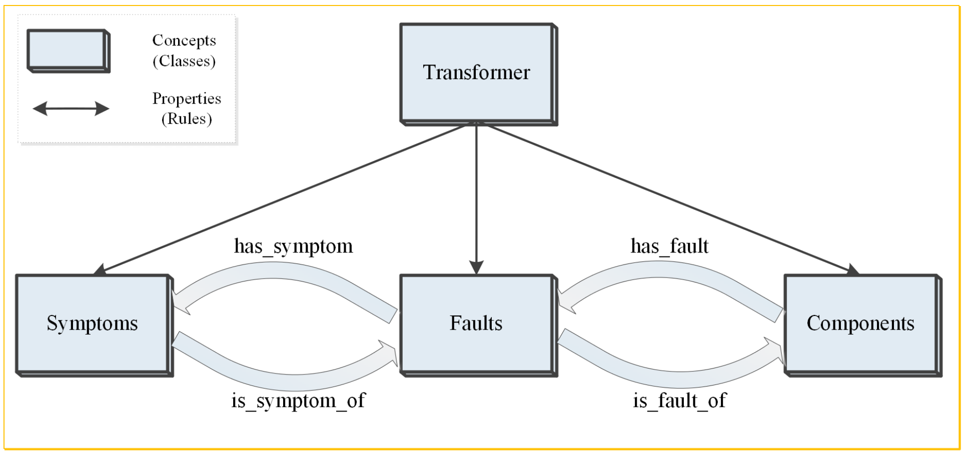


FIGURE 5.4: Main classes of proposed ontology for power transformer fault diagnosis

the fault A can be diagnosed by observing symptom B. The component category is also linked via some properties to the fault category, which reflects the relationship between faults and components. These three categories and their relations are employed as the basic concepts of ontology reasoning for transformer fault diagnosis. The elements of the developed ontology are shown in Figure 5.4.

Based on ontology structure, the components of developed ontology for power transformer fault diagnosis are defined as follows:

Class

The developed ontology consists of three main classes: *Components*, *Symptoms* and *Faults*, and consequently each of them is defined as subclass of class *Transformer*, as described in the following axioms:

$$Symptoms \sqsubseteq Transformer \tag{5.1}$$

$$Faults \sqsubseteq Transformer \tag{5.2}$$

$$Components \sqsubseteq Transformer \tag{5.3}$$

where these axioms are represented in OWL as:

```
<owl:Class rdf:about="#Transformer">
  </owl:Class>
```

```
    <owl:Class rdf:about="#Symptoms">
<rdfs:subClassOf rdf:resource="#Transformer"/>
  </owl:Class>
    <owl:Class rdf:about="#Faults">
<rdfs:subClassOf rdf:resource="#Transformer"/>
  </owl:Class>
    <owl:Class rdf:about="#Components">
<rdfs:subClassOf rdf:resource="#Transformer"/>
</owl:Class>
```

The syntax in the statement above defines that, there is a concept called *Transformer* with three subclasses, *Symptoms*, *Faults* and *Components*, built in OWL.

Faults in a power transformer can be classified into five types: *Electrical*, *Thermal*, *Mechanical*, *Degradation* and *Ageing*. These five types are defined as the subclasses of class *Faults*, described as follows:

$$Electrical\_Faults \sqsubseteq Faults \tag{5.4}$$

$$Thermal\_Faults \sqsubseteq Faults \tag{5.5}$$

$$Ageing\_Faults \sqsubseteq Faults \tag{5.6}$$

$$Degradation\_Faults \sqsubseteq Faults \tag{5.7}$$

$$Mechanical\_Faults \sqsubseteq Faults \tag{5.8}$$

Each types of faults can be further subdivided into different types of related faults, as shown below in the case of *Degradation* fault:

$$Degradation\_Of\_Insulation \sqsubseteq Degradation\_Faults \tag{5.9}$$

$$Degradation\_Of\_Iron \sqsubseteq Degradation\_Faults \tag{5.10}$$

$$Degradation\_Of\_Paper \sqsubseteq Degradation\_Of\_Insulation \tag{5.11}$$

The Roger's method, as an example of the existing fault diagnosis technique, has been applied as a class called *Rogers_Method_Faults* with eight types of faults, defined as subclasses (the ontology can be extended using diagnosis methods of fault diagnosis).

Class *Symptoms* consists of various symptoms types, which may appear in a power transformer. For instance, in the Roger's fault diagnosis method, the symptoms are three gas ratios R_1, R_2 and R_5, that help to distinguish the fault types. These types of gas ratios can be represented as the subclasses of class *Gas*. Other types of symptoms can be represented, such as *Acidity*, *Temperature*, *Electrical* and *Physical* symptoms, which are defined as the subclasses of class *Symptoms*. For the case of Roger's fault diagnosis the *Symptoms* contain a subclasses *GasRatios*, with five types of gas ratios *Ratio*1 to *Ratio*5. The axioms related to the class *Symptom* are given in equations 5.12 to 5.21.

$$Temperature \sqsubseteq Symptoms \tag{5.12}$$

$$Electrical \sqsubseteq Symptoms \tag{5.13}$$

$$Acidity \sqsubseteq Symptoms \tag{5.14}$$

$$Physical \sqsubseteq Symptoms \tag{5.15}$$

$$GasRatios \sqsubseteq Symptoms \tag{5.16}$$

$$Ratio1 \sqsubseteq GasRatios \tag{5.17}$$

$$Ratio2 \sqsubseteq GasRatios \tag{5.18}$$

$$Ratio3 \sqsubseteq GasRatios \tag{5.19}$$

$$Ratio4 \sqsubseteq GasRatios \tag{5.20}$$

$$Ratio5 \sqsubseteq GasRatios \tag{5.21}$$

Similarly, the class *Components* contains the power transformer components, such as *Winding*, *Cooling_system*, *Taps*, *Oil* etc., defined as its subclasses. Figure 5.5 shows the defined class of transformer fault diagnosis.

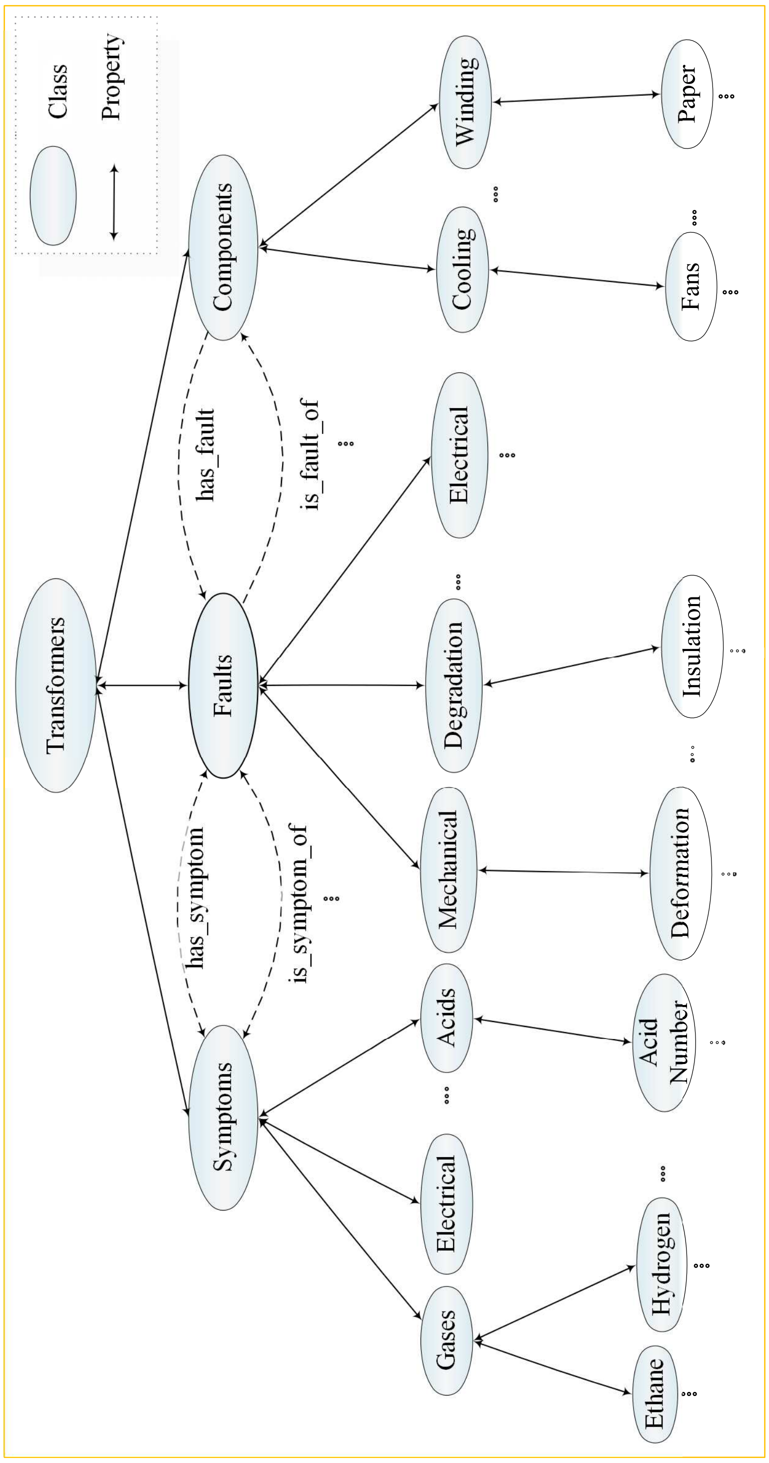


FIGURE 5.5: The class *Transformer* and its subclasses for power transformer fault diagnosis

The designed classes are restricted to various types of properties, (shown with dashed line between classes in Figure 5.5).

Properties

Properties provide the binary relations of classes or individuals. There are two main types of properties, namely the object properties and datatype properties. They provide different attributes to the classes. Two categories of properties, "*has_category*" and "*is_category_of*" with inverse characteristics to each other are applied. The inverse properties represent that, if a properties links individual x to individual y, then the inverse property links the individual y to individual x. In this study, the *has_category* property is the inverse of *is_category_of* property.

OWL can be used to define sub-properties of each property. In this work, each property has three sub-properties. For instance, the property *has_category* has three sub-properties as "*has_fault*", "*has_symptom*" and "*has_component*", with different characteristics, such as functional, inverse, etc. An individual with functional property represents that there can be at most one individual related to him via this property [103]. The functional property with an example in ontology for fault diagnosis is examplified here. It is assumed that the property called *is_symptom_of* is defined as functional. If individual *High_Temperature* is a symptom of *OverHeating*, and also that the individual *High_Temperature* is a symptom of *Thermal_Fault*, then because *is_symptom_of* is a functional property, it can be inferred that *Thermal_Fault* and *Overheating* must be the same individual. This is shown in the Figure 5.6.

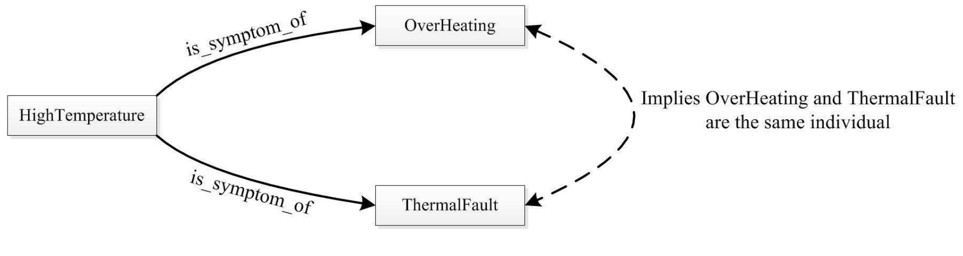


FIGURE 5.6: An example of a functional property *is_symptom_of*

Similarly for the property *is_category_of*, three sub properties are defined, including *is_fault_of*, *is_symptom_of* and *is_component_of*. Summary of applied properties is itemized below.

1. $has\_category = \left\{ \begin{array}{l} has\_fault \\ has\_symptom \\ has\_component \end{array} \right\};$

2. $is\_category\_of = \left\{ \begin{array}{l} is\_fault\_of \\ is\_symptom\_of \\ is\_component\_of \end{array} \right\}.$

To apply these properties for the defined classes in previous sections, the following statements are used:

A) $Faults\ has\_symptom$ some $Symptoms$;

B) $Symptoms\ is\_symptom\_of$ some $Faults$.

which means that all types of faults have some types of symptoms, defined in class *Symptoms* (item A). The word some represents *existential restriction* to describes the class *Faults*, which means all the faults has at least one symptom. The inverse statement (item B) indicates that the symptoms correspond to some types of fault. The following examples of power transformer fault diagnosis are given to illustrate the above statement. For instance, a fault of partial discharge may lead to the presence of hydrogen in the oil symptom (found from gas ratio values). The statements to describe this restriction are:
Example for A) $Partial\_Discharge\ has\_symptom$ some $Hydrogen$;
Example for B) $Hydrogen\ is\_symptom\_of$ some $Partial\_Discharge$.

Furthermore, the components of power transformer can be identified by the relevant fault types. For instance, the degradation of paper insulation in winding causes the fault called arcing. In the case of the fault type (here – arcing) has been diagnosed based on captured symptoms, and it is required to detect the faulty component of power transformer, where the paper insulation in winding caused arcing. These types of restrictions are defined in the following statements:

C) $Components\ has\_fault$ some $Faults$;

D) $Faults\ is\_fault\_of$ some $Components$.

An example is shown here for transformer fault diagnosis. Almost all types of transformers have a tank made of carbon steel. Acidity can be defined as the mass of potassium hydroxide in milligrams, which is required for neutralisation of acid in one gram of transformer oil [104]. Consequently, higher amount of acid in oil is represented as higher acid numbers. The acid number generally tends to increase with the ageing of power transformer due to oxidative processes in the insulation and acid formation. The acid attacks the metal inside of the tank and results in tank corrosion. Therefore, the presence of corrosion faults can be illustrated with the help of the statement as "$Corrosion\ is\_fault\_of$ only $Tank$", where the only is defined as universal restriction.

A concept C can be described with the set of necessary condition, if the values x_i in some properties p_i, $i = 1, 2, ..., n$, and p_i is denoted as a necessary condition. This can be denoted:

$$C \subseteq D \tag{5.22}$$

where "$\subseteq$" denoted the dependency of C on D.

For instance, the fault *Corrosion* can be defined as necessary condition of the fault class. This assertion can be expressed with following statement:

$$Corrosion \subseteq Faults \tag{5.23}$$

It is also possible to define the concept with sufficient condition, where the condition D is sufficient condition of the concept C. This can be denoted with:

$$C \equiv D \tag{5.24}$$

This can be illustrated with example of ontology for power transformer fault diagnosis represented with sufficient condition. Using an information of previous example, that the corrosion fault of the tank in power transformer component is only have a high acid number in the oil, this notation can be defined as follows:

$$Corrosion\_In\_Tank \equiv Fault \sqcap \forall has\_symptom.High\_Acid\_Number \tag{5.25}$$

In the case of having no common elements, the class can be defined as disjoint class. For instance, two classes, *Faults* and *Symptoms* are disjoint, because the faults can have different types of symptoms (like gas, temperature symptoms) but cannot be a symptom. This can be expressed with the following notation:

$$Faults \sqcup Symptoms \sqsubseteq \perp \tag{5.26}$$

Protégé [66] is a powerful tool to support OWL; it is based on a graphical editor. The Protégé ontology editor supports $\mathcal{SHIQ}$(D). Figure 5.7 represents the developed ontology for power transformer fault diagnosis.

Accordingly, by giving the symptoms, the fault types can be diagnosed; and the relevant components are identified by giving the fault types. The communication between the developed ontology and power transformer in the substation can be handled by an agent.

5.2.2 Ontology agent

As shown in the Figure 5.3, the built ontology is wrapped by an agent called *Ontology*, can be denoted as $\mathcal{A}_{ONT}$. The $\mathcal{A}_{ONT}$ receives a message containing some symptoms from the agent who captures the real-time information from power transformer. In this case we assumed that the agent *Data_Sender*, denoted as $\mathcal{A}_{DS}$, sends the symptoms (e.g. gas ratios). The $\mathcal{A}_{ONT}$ passes the symptoms as arguments to the ontology, and ontology uses its reasoner to verify inconsistency of the classes and also to diagnose the relevant fault. The FaCT++ reasoner is applied to this ontology. This information is also can be reported to the *User* agent or sent to the *Data_Collector* agent to be saved in the database. A sample of $\mathcal{A}_{ONT}$ are presented in the Appendix C.

5.2.3 Ontology reasoning

One of the key features of applying ontology is to extract hidden information from the explicit facts built in ontology. To consider this situation, two examples are presented. Corrosion fault in the tank can be diagnosed based on the high acid number of oil. The information related to high acid number can be delivered by the agent sensors connected inside of the oil tank, to the $\mathcal{A}_{ONT}$ who wraps the ontology for fault diagnosis.

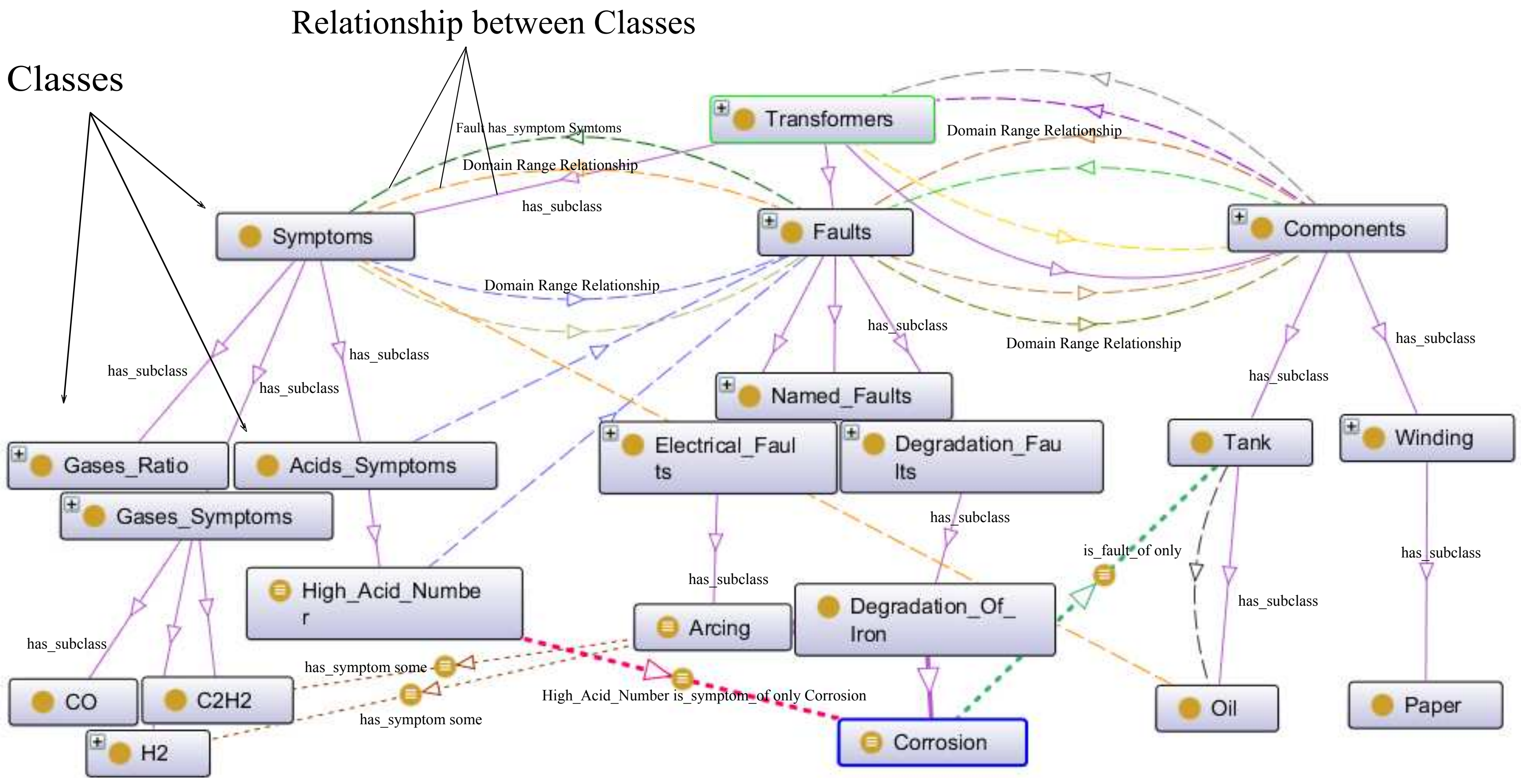


FIGURE 5.7: Ontology for power transformer fault diagnosis

The proposed ontology has a class *Faults*, containing a subclass *Corrosion*. The class *Symptoms* has a defined subclass, *High_Acid_Number*, and the class *Tank* is defined as the subclass of the class *Components*, with a subclass of *Oil*. The interesting part of the ontology reasoner is that the class *Faults* does not have any subclass called *Tank_Corrosion*, which can be deduced by this ontology. In this case, the applied restrictions for faults, symptoms and components are given below:

$$Corrosion\ has\_symptom\ only\ High\_Acid\_Number \tag{5.27}$$

which means corrosion is a type of fault, which can be detected only with symptom of high acid number.

$$High\_Acid\_Number\ is\_symptom\_of\ only\ Corrosion \tag{5.28}$$

which represent the inverse property of statement 5.27.

$$Corrosion\ is\_fault\_of\ only\ Tank \tag{5.29}$$

which represents the necessary and sufficient conditions of the corrosion with tank.

$$Tank\ has\_component\ some\ Oil \tag{5.30}$$

which means tank is filled with oil. The elements of this ontology reasoner are shown in Figure 5.8.

According to agent's observations, the acid number of oil is high. The properties, high acid number is the only symptom of the fault corrosion, from statement 5.28 and tank filled by this oil statement 5.30, the undefined knowledge can be deduced. In fact, the ontology reasoner is able to infer the new knowledge that the tank has a corrosion fault.

Another example of reasoning, based on DL description, can be presented. Degradation is a common type of fault in power transformer, happening due to the transformer ageing. Moreover, degradation itself speeds the ageing of the equipment up. There are several factors other than equipment ageing that can cause the degradation, such as

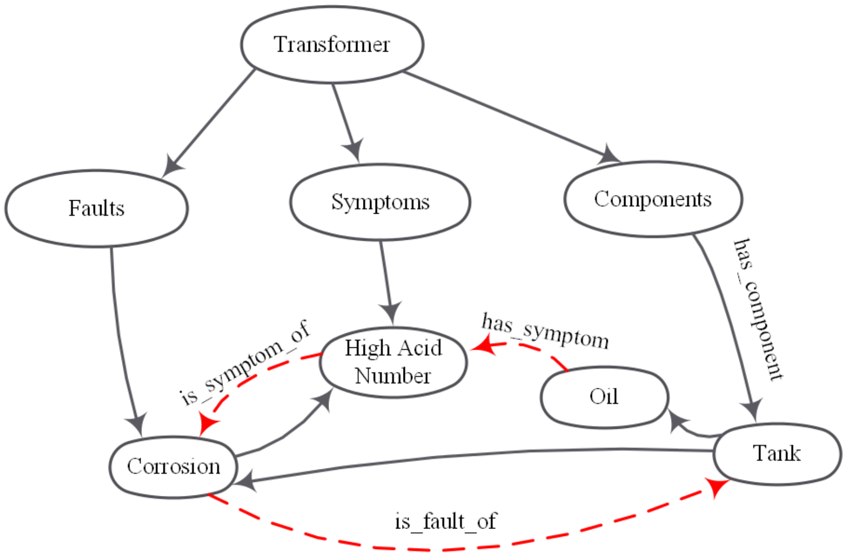


FIGURE 5.8: An example of ontology reasoner for extracting implicit information from the explicit facts

water, temperature, byproducts, etc. High temperature or presence of water are the key factors (or symptoms) for degradation of transformer component, e.g. paper insulation. *Degradation* is an abbreviation for the concept description, which can be defined in the TBOX as follows:

$$\begin{aligned} Degradation \equiv Faults \ \sqcap \ (\exists\ has\_symptom.Symptoms) \ \sqcap \\ (\forall\ has\_symptom.(water \sqcup temperature)) \end{aligned} \tag{5.31}$$

which means that the degradation is a type of fault and it has some symptoms, either water or temperature. The current situation is described in ABox stating the properties of individuals. The ABox contains:

$$\begin{aligned} Degradation(Paper\_Degradation), has\_fault(Paper, Degradation), \\ \neg water(Paper) \end{aligned} \tag{5.32}$$

It means that the instance *Paper_Degradation* belongs to the concept *Degradation*; *Paper* has a fault *Degradation*, and there is no water in *Paper* (the paper is not wet). Users receive the reasoning services from the modern DL, which can automatically deduce implicit knowledge from the explicitly represented knowledge, and it always yields a correct answer in finite time [92]. For the case presented above, the instance algorithm determines the instance relationships.

For the given ABox and the definition of *Degradation*, *Paper has_fault Degradation* because *Paper_Degradation* is an instance of *Degradation*, so all its symptoms are either *Water* or *Temperature*, and paper is not wet ($\neg water(paper)$), then concluding that the paper degradation is caused by temperature.

5.3 Case Study Using Ontology-based Reasoning for Fault Diagnosis

The routine work of power engineers includes such tasks, as condition assessment, fault diagnosis, maintenance and decision-making, which involves their knowledge and data analysis. This work presents a comprehensive knowledge base for power transformer fault diagnosis based on ontology reasoning. For this purpose, the Roger's method in from of ontology is investigated for transformer fault diagnosis. A summary of proposed ontology contains class of faults with eight subclasses, which represent the recommended cases of the fault types in the Roger's method. Three datatypes properties are applied for restricting the faults' classes. The datatype property called *has_ratio* contains three sub-properties, namely *has_ratio_R1*, *has_ratio_R2* and *has_ratio_R5*, applied for this ontology.

The first case of the Roger's method as illustrated in Table 2.3, corresponding to *No_Faults* can be defined in Protégé with the following statements:

$$\begin{aligned} &Faults\ and\ (has\_ratio\_R2\ some\ float[< 0.1])\ and\ (has\_ratio\_R1\ some \\ &\quad float[\geq 0.1,\ \ \leq 1.0])\ and\ (has\_ratio\_R5\ some\ float[\leq 1.0]) \end{aligned} \tag{5.33}$$

which means, if the received gas ratios are within the defined boundaries, then there is no fault in the power transformer. Ontology can be formalised in a TBox with DL in $\mathcal{SHIQ}$ for the "No_Fault" statement [95], as follows:

Case 0:

$$No\_Fault \equiv (\forall Faults.\top) \sqcap (< 0.1\ has\_ratio\_R2.\top) \sqcap ((\geq 0.1\ has\_ratio\_R1 .\top) \sqcap (\leq 1.0\ has\_ratio\_R1.\top)) \sqcap (\leq 1.0\ has\_ratio\_R5.\top) \tag{5.34}$$

which expresses the case of *No_Fault* with three conditions of gas ratios. This statement can be represented in OWL2 syntax, given in Appendix C. The cases 1 and 2 of the Roger's method present conditions for the partial discharge fault; they can be summarised in one case only. The rest of the Roger's method cases are formalized in the TBox in the same way, as given below:

Case 1&2:

$$Partial\_Discharge \equiv (\forall Faults.\top) \sqcap ((\geq 0.1\ has\_ratio\_R2.\top) \sqcap (\leq 3.0\ has\_ratio\_R2.\top)) \sqcap (< 0.1\ has\_ratio\_R1.\top) \sqcap (\leq 1.0\ has\_ratio\_R5.\top) \tag{5.35}$$

Case 3:

$$Low\_Energy\_Discharge, Sparking, Arcing \equiv (\forall Faults.\top) \sqcap (\geq 0.1\ has\_ratio\_R2.\top) \sqcap ((\geq 0.1\ has\_ratio\_R1.\top) \sqcap (\leq 1.0\ has\_ratio\_R1.\top)) \sqcap (\geq 1.0\ has\_ratio\_R5.\top) \tag{5.36}$$

Case 4:

$$High\_Energy\_Discharge, Arcing \equiv (\forall Faults.\top) \sqcap ((\geq 0.1\ has\_ratio\_R2.\top) \sqcap (\leq 3.0\ has\_ratio\_R2.\top)) \sqcap ((\geq 0.1\ has\_ratio\_R1.\top) \sqcap (\leq 1.0\ has\_ratio\_R1.\top)) \sqcap (> 3.0\ has\_ratio\_R5.\top) \tag{5.37}$$

Case 5:

$$Thermal\_Fault\_Temperature\_Less\_Than\_150\ ^{\circ}\mathrm{C} \equiv (\forall Faults.\top) \sqcap (< 0.1\ has\_ratio\_R2.\top) \sqcap ((\geq 0.1\ has\_ratio\_R1.\top) \sqcap (\leq 1.0\ has\_ratio\_R1.\top)) \sqcap ((\geq 1.0\ has\_ratio\_R5.\top) \sqcap (\leq 3.0\ has\_ratio\_R5.\top)) \tag{5.38}$$

Case 6:

$$Thermal\_Fault\_Temperature\_between\_150\_300\ ^{\circ}\mathrm{C} \equiv (\forall Faults.\top) \sqcap (< 0.1\ has\_ratio\_R2.\top) \sqcap (\geq 1.0\ has\_ratio\_R1.\top) \sqcap (\leq 1.0\ has\_ratio\_R5.\top) \tag{5.39}$$

Case 7:

$$Thermal\_Fault\_Temperature\_between\_300\_700\ ^{\circ}\mathrm{C} \equiv (\forall Faults.\top) \sqcap (< 0.1\ has\_ratio\_R2.\top) \sqcap (\geq 1.0\ has\_ratio\_R1.\top) \sqcap ((\geq 1.0\ has\_ratio\_R5.\top) \sqcap (\leq 3.0\ has\_ratio\_R5.\top)) \tag{5.40}$$

Case 8:

$$Thermal\_Fault\_Temperature\_Over\_700\ ^{\circ}\mathrm{C} \equiv (\forall Faults.\top) \sqcap (< 0.1\ has\_ratio\_R2.\top) \sqcap (\geq 1.0\ has\_ratio\_R1.\top) \sqcap (> 3.0\ has\_ratio\_R5.\top) \tag{5.41}$$

Some screenshots of the developed ontology are given in the Appendix C. The built ontology is tested with $\mathcal{A}_{DS}$, who sends the DGA gas ratio samples (identical to the used data in previous experiment), and diagnoses the fault types.

5.3.1 The experimental results of fault diagnosis with ontology

Ontology-based fault diagnosis for power transformer have been tested using the real data, identical to the data presented in Section 4.5. Obviously, the results of fault diagnosis for the applied ontology are the same as the results used with JESS programming. Ontology-based reasoning for power transformer fault diagnosis is a novel representation of the knowledge-based system in power system, which can be improved with other types of ontology, for example fuzzy ontology. Table 5.3 presents 20 data samples diagnosed with the ontology-based reasoning.

Table 5.4 contains the summary of results for 70 DGA gas samples applied. Similarly, three types of fault categories are diagnosed correctly, while the "Arcing" and "Partial Discharge" fault types could not be identified correctly in some cases. The situation with undefined faults can be improved further by applying the fuzzy ontology (as presented in the following chapter).

TABLE 5.3: The actual DGA samples applied with ontology-based reasoning

R_2	R_1	R_5	**Actual Fault**	**Ontology Agent Results**	**Results**
1.16	0.46	5.2	Arcing	Low Energy Partial Discharge, Arcing, Sparking	Correct
0.07	5.43	5.26	Overheating	Thermal Fault(TF) $TF > 700°$C	Correct
1.65	0.17	3.13	Arcing	Low Energy Partial Discharge, Arcing, Sparking	Correct
1.06	1.74	9.26	Arcing	Undefined Fault	ND*
0.04	3.86	6.94	Overheating	Thermal Fault $TF > 700°$C	Correct
0.97	1.79	7.06	Arcing	Undefined Fault	ND*
0.01	40.99	5.07	Overheating	Thermal Fault $TF > 700°$C	Correct
3.25	0.08	17.75	Partial Discharges	Undefined Fault	ND*
0.02	3.09	7.44	Overheating	Thermal Fault $TF > 700°$C	Correct
0.01	1.42	10.02	Overheating	Thermal Fault $TF > 700°$C	Correct
2.74	1.54	13.42	Arcing	Undefined Fault	ND*
0.01	2.69	8.62	Overheating	Thermal Fault $TF > 700°$C	Correct
2.93	0.09	6.6	Arcing	Undefined Fault	ND*
2.26	0.29	10.82	Arcing	Low Energy Partial Discharge, Arcing, Sparking	Correct
3.42	0.08	5.6	Partial Discharges	Undefined Fault	ND*
0.02	2.39	7.16	Overheating	Thermal Fault $TF > 700°$C	Correct
3.3	0.07	16.5	Arcing	Undefined Fault	ND*
0.02	2.4	6.7	Overheating	Thermal Fault $TF > 700°$C	Correct
0	4.85	1.85	Overheating	Thermal Fault $300 < TF < 700°$C	Correct
1.45	0.84	14	Arcing	Low Energy Partial Discharge, Arcing, Sparking	Correct

* ND = Not Defined

5.3.2 Discussion and conclusion

It is important to share and use the knowledge-based systems in various domains. Earlier expert systems have a significant disadvantage of both domain knowledge and rules (how to use the domain) contained in the knowledge base. Thus, the success in one domain could hardly be replicated to another one, due to high degree of interconnections between domain knowledge and rules. For example, the knowledge-based system for transformer fault diagnosis with rule-based reasoning (described in previous chapter) is able to take the gas ratios and then present the appropriate fault types. For the case of properly

Table 5.4: Summary of ontology-based reasoning for fault diagnosis

Fault Types	**Total Samples**	**Diagnosed Correctly**	**Not Defined Fault**	**Accuracy**
No Fault	1	1	0	100%
Partial Discharge	2	0	**2**	**0%**
Arcing	14	9	**5**	**64.3%**
Overheating	25	25	0	100%
Low Energy Discharge	28	28	0	100%
Average Total Accuracy	—	—	—	**72.86%**

conceptualized knowledge it could be possible to separate the domain knowledge from the application, while the used inference rules are directed to the knowledge, which makes it difficult to separate them. In this case sharing them in different domain is a difficult task, and this can be taken as disadvantage. Furthermore, the represented knowledge-based system with the use of rule-based reasoning cannot be easily improved, as it is impossible to know all the conditions to build the rule-based reasoning.

The ontology-based DL provides the strong structured knowledge representation, which can be understood by other applications. Proving advance services, such as semantic search and automated reasoning, enables various applications of it. Ontology-based knowledge representation has been proposed for condition assessment of the power system components [3, 6]. The main scope of the developed ontology was to provide the real-time information of power system components to the agent system. The proposed ontology only can represent the hierarchy structure of components and cannot participate in the main feature of ontology for extracting hidden information. In case of fault diagnosis, ontology is applied for power transformer [9]. The ontology described in [9] has the weak point of applying object properties instead of datatype. This can result in need of more classes to represent all statuses of fault types. The ontology developed in this work, based on DL and applied datatype properties, provides more features.

5.4 Summary

To make the knowledge-based system useful for various applications, ontology-based DL is presented. For this purpose we developed an ontology for power transformer fault diagnosis. The developed ontology in OWL-DL is translatable into a DL representation, which can perform automated reasoning using a DL reasoner. The DL reasoners compute various inference services, such as computing the inferred hierarchy classes and super-classes, inferred inconsistency in classes, subsume of classes, etc. Ontology agents are designed to interact with the developed ontology for the purpose of providing the real-time information (e.g. symptoms) to diagnose the fault types. Finally, the actual DGA samples are tested with ontology, and the results are discussed.

Chapter 6

Fuzzy Ontology for Power Transformer Fault Diagnosis

6.1 Introduction

A knowledge base is not a static collection of information, but a dynamic resource that has a capability to be revised by itself or with the help of agents technology. Therefore, it is important to have the knowledge updated with the newer knowledge-based systems. Ontology-based knowledge representation provides an opportunity to upgrade a knowledge easily. For this purpose a new ontology for power transformer fault diagnosis based on the fuzzy ontology have been developed. The ontology presented earlier in the Chapter 5 can be improved. Thus, the developed fuzzy ontology is able to deal with uncertainty, which is a common requirement in the real world. The improved accuracy of the developed fuzzy ontology is another issue discussed in this chapter.

6.1.1 Fuzzy theory and fuzzy sets

Human reasoning is based on approximation and imprecision, which can be handled by fuzzy systems. The fuzzy set theory was proposed by Lotfi Zadeh in 1965 [105] for dealing with the approximate reasoning. It finds various applications in such fields as artificial intelligence, control theory, etc. Element of the fuzzy set belongs to a set to some degree, defined as a membership function, while in the classical set, the elements either belong to a set or not. Therefore, crisp thresholds used in the classical set are replaced with the fuzzy thresholds for the fuzzy set.

A fuzzy set A in the fuzzy subset of X is defined as a mapping:

$$A : X \rightarrow [0,1]\ ,\ x \in X$$

where the "$\mu_A(x)$" can be defined as the membership degree of x to the fuzzy set A [105].

Various logic operations provide connectives in fuzzy set theory and fuzzy logic. In fuzzy, the membership can vary between 0 and 1, while in the classical case, it can be either 0 or 1. These characteristics of the fuzzy sets theory provide an ability of representing the imprecise or vague types of knowledge. For instance, the concept "$Teenager\_Boys$" is defined as follows:

$$Teenager\_Boys \ = \ Boys \cap \ \exists \ has\_Age.Teenager$$

where the linguistic term *Teenager* may be defined by a trapezoidal function, as shown in Figure 6.1. The mathematical representation for *Teenager* is given in equation 6.1.

$$\mu_{Teenager}(Age) \ = \begin{cases} (Age-12)/(13-12) & if \quad 12 \leq Age \leq 13 \\ 1 & if \quad 13 \leq Age \leq 18 \\ (19-Age)/(19-18) & if \quad 18 \leq Age \leq 19 \\ 0 & if \quad Age < 12 \ or \ Age > 19 \end{cases} \tag{6.1}$$

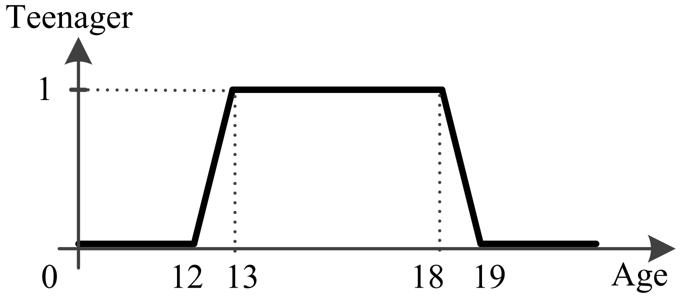


FIGURE 6.1: Example of trapezoidal membership function

The same example can be illustrated in classical set without using the fuzzy set, where only individual in the age from 13 to 18 years old can be called *Teenager*.

Many types of fuzzy logic apply various sets of operations, such as intersection, union, complement and implication, for presenting different types of properties. Lukasiewicz, Product and Godel are three main fuzzy logics, providing different properties [106, 107].

Fuzzy system has been applied in power system for the DGA fault diagnosis, as summarised in [108, 109]. The fuzzy set can be implemented in several ways, and the most common structure is based on the rule set, from which actions and conclusions can be suggested. The accuracy of fault diagnosis can be improved with the use of fuzzy logic in combination with Artificial Neural Network (ANN) or Evolution Algorithms (EA). The fuzzy system proposed in [110] delivers the accuracy up to 88%, based on 561 DGA gas samples.

6.1.2 Fuzzy ontology OWL2

The idea of expressing imprecise or vague objects is taken from the fuzzy logic and applied in semantic web ontologies, for representing the fuzzy ontology. The fuzzy ontology has been introduced by Straccia to apply the non crisp data within the ontology definition [106]. Some advantages of fuzzy ontology are itemized below:

- Fuzzy OWL ontology can be shared and reused easily;
- Easily extendable to the other types of fuzzy OWL2 statements;
- It can be easily translated into the syntax of other fuzzy-DL reasoner.

Three alphabets are assumed in fuzzy OWL2, namely fuzzy concepts, fuzzy roles and individuals [111]. The fuzzy concepts denote fuzzy sets of individuals, and fuzzy roles denote fuzzy binary relations. Fuzzy modifier is the function which can be applied to a fuzzy set to change its membership function. Two types of modifier, linear and triangular, can be applied in fuzzy ontology. The fuzzy modifiers have capability of using some expressions, such as *very*, *more* or *less*, to express their membership functions in fuzzy sets. For instance, the oil temperature can be *very* high, where the *very* is a linear modifier, which can be defined as *linear(0.9)*. Table 6.1 presents the syntax of the fuzzy OWL2 given in [111].

Table 6.1: Summary of fuzzy OWL2 syntax

Concept	**Syntax**	**Axiom**	**Syntax**
(C1)	A	(A1)	$\langle a{:}C \bowtie \alpha \rangle$
(C2)	$\top$	(A2)	$\langle (a,\ b){:}R \bowtie \alpha \rangle$
(C3)	$\perp$	(A3)	$\langle (a,\ b){:}\neg R \bowtie \alpha \rangle$
(C4)	$C \sqcap D$	(A4)	$\langle (a,\ \nu){:}T \bowtie \alpha \rangle$
(C5)	$C \sqcup D$	(A5)	$\langle (a,\ \nu){:}\neg T \bowtie \alpha \rangle$
(C6)	$\neg\ C$	(A6)	$\langle a \neq b \rangle$
(C7)	$\forall R.C$	(A7)	$\langle a = b \rangle$
(C8)	$\exists R.C$	(A8)	$\langle C \sqsubseteq D \triangleright \alpha \rangle$
(C9)	$\forall T.\ d$	(A9)	$C_1 \equiv ...\ C_m$
(C10)	$\exists T.\ d$	(A10)	dis(C_1, ..., C_m)
(C11)	$\{\alpha/a\}$	(A11)	disUnion(C_1, ..., C_m)
(C12)	$\geq m\ S.C.$	(A12)	$\langle R_1\ ...R_m \sqsubseteq R \triangleright \alpha \rangle$
(C13)	$\leq n\ S.C.$	(A13)	$\langle T_1 \sqsubseteq T_2 \triangleright \alpha \rangle$
(C14)	$\geq m\ T.\ d$	(A14)	$R_1 \equiv ...\ R_m$
(C15)	$\leq n\ T.\ d$	(A15)	$T_1 \equiv ...\ T_m$
(C16)	$\exists S$.self	(A16)	domain(R, C)
(C17)	*mod(C)*	(A17)	range(R, C)
(C18)	$\alpha \cdot C$	(A18)	range(T, **d**)
(C19)	$(\alpha_1 \cdot C_1) + ... + (\alpha_k \cdot C_k)$	(A19)	func(S)
Role	**Syntax**	(A20)	func(T)
(R1)	R_A	(A21)	$R \equiv R^-$
(R2)	T	(A22)	trans(R)
(R3)	R^-	(A23)	dis(S_1, ..., S_m)
(R4)	U	(A24)	dis(T_1, ..., T_m)
(R5)	*mod(R)*	(A25)	ref(R)
Datatype	**Syntax**	(A26)	irr(S)
(D1)	left$(k_1,\ k_2,\ a,\ b)$	(A27)	sym(R)
(D2)	right$(k_1,\ k_2,\ a,\ b)$	(A28)	asy(S)
(D3)	triangular$(k_1,\ k_2,\ a,\ b,\ c)$		
(D4)	trapezoidal$(k_1,\ k_2,\ a,\ b,\ c,\ d)$		
(D5)	*mod*(d)		

As can be seen in the Table 6.1, the fuzzy concepts represent various constructors, from C1 to C19. The fuzzy roles syntax from R1 to R5, and the datatypes syntax from D1 to D5, are provided for restriction. The applied axioms in fuzzy OWL2 based on logic, are given in the Table 6.1, and can be grouped into the ABox (A1 to A7), TBox (A8 to A11) and RBox (A12 to A28). Let us consider an example of fuzzy ontology for power transformer fault diagnosis. The following concept can represent the fault

Corrosion:

$$Corrosion \equiv Fault \sqcap \exists\ has\_symptom.High\_Acid\_Number.$$

The concept $High\_Acid\_Number$ can be easily defined with fuzzy concept "$Acid\_Number$" and a fuzzy modifier "*High*". The concept $\langle$ *Acid_Number*: *High* ≥ 0.9 $\rangle$ states that the acid number is high with at least degree of 0.9.

Annotation in fuzzy OWL2 will be delimited with a starting tag "<fuzzyOwl2>", and an ending tag "< /fuzzyOwl2>". Let us define the fuzzy modifier $High = linear(0.9)$ of the *Corrosion* fault presented previously. A fuzzy datatype *High* is annotated as follows:

```
<AnnotationAssertion >
   <AnnotationProperty IRI = #fuzzyLabel/>
   <IRI >#High </IRI >
   <Literal datatypeIRI = &rdf;PlainLiteral>
      <fuzzyOwl2 fuzzyType = "modifier">
         <Modifier type = "linear" c="0.9" />
      </fuzzyOwl2 >
   </Literal >
</AnnotationAssertion >
```

A fuzzy datatype D is a pair $\langle \triangle_D,\ \Phi_D \rangle$, where $\triangle_D$ is a concrete interpretation domain, and Φ_D is a set of fuzzy concrete predicates d with an arity n and an interpretation $d^{\mathcal{I}}\ : \Delta_D^n\ \to [0,\ 1]$, which is an n-ary fuzzy relation over $\triangle_D$ [111]. Various functions, such as trapezoidal, triangular, left-shoulder function (L-function), right-shoulder function (R-function) etc., can be used to specify the membership function in fuzzy modified datatypes, as shown in Figure 6.2.

Depending on the fuzzy datatypes, various parameters, K_1, K_2, a, b, c, d can be applied, as shown in the Table 6.1. For instance, the gas ratio R_2 from the Roger's method represents the condition $R_2 < 0.1$. According to [110, 112], the undefined fault cases usually appear when the value of gas ratios are close to the crisp boundaries, given in

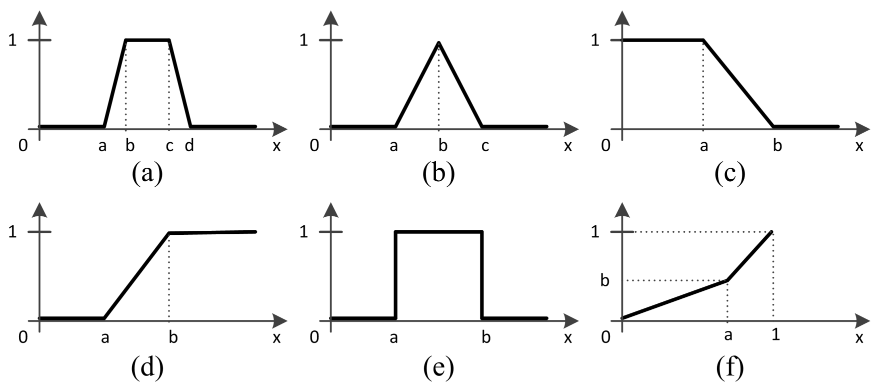


FIGURE 6.2: (a) Trapezoidal function, (b) Triangular function, (c) Left-shoulder function, (d) Right-shoulder function, (e) Crisp function, (f) Linear function

the Roger's method. However, the use of non crisp threshold based on fuzzy membership function is recommended in order to solve this situation [110, 112]. In this case, the fuzzy ontology can be applied in the form of left-shoulder function, with parameters $a = 0.05$ and $b = 0.15$, for the datatype R_2, as shown in Figure 6.3.

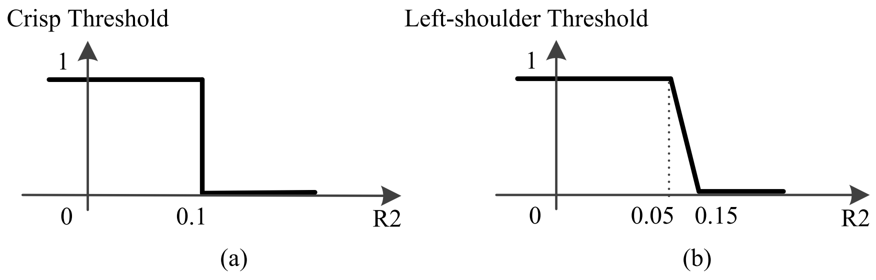


FIGURE 6.3: Example of replaced fuzzy left-shoulder function with crisp function (a) Crisp threshold function, (b) Left-shoulder threshold function

As shown in the Figure 6.3, the ratio R_2 is represented with the left-shoulder function, allowed by fuzzy modified datatype given in the Table 6.1.

6.1.3 Fuzzy DL reasoner

Several fuzzy reasoners, such as DELOREAN, FuzzyDL, FIRE, etc., are developed to support the fuzzy ontology [106]. Fuzzy DL reasoner have been proposed as an extension to the classical DLs, which aims to deal with fuzzy/vague/imprecise concepts. It supports fuzzy logic reasoning and based on the fuzzy DL $\mathcal{SHIF}(\mathcal{D})$ [113]. The tableaux algorithm and Mixed Integer Linear Programming (MILP) optimization problem have been applied in the fuzzy DL reasoner [114]. Query language is one of the useful features of the reasoner; it allows to compute several different types of queries. Table 6.2 represents the concrete syntax for the queries.

For instance, the concept query ($max - subs$? C D) determines the maximal degree of the concept C subsuming the concept D. Another feature, called "Show Expressions", can be used to show the values in an optimal solution, as presented in the Table 6.3 [114].

6.2 Fuzzy Ontology for Transformer Fault Diagnosis

As mentioned earlier, the main components of building a fuzzy ontology are fuzzy datatypes, fuzzy modifiers, fuzzy concepts and fuzzy roles. To use the fuzzy ontology for power transformer fault diagnosis, the Roger's method is applied. Previously in this work, the developed ontology-based fault diagnosis used three crisp threshold datatypes, R_1, R_2 and R_5, for restriction of defined fault classes. However, there were some situations where the ontology could not diagnose the fault types. Consequently, the undefined faults were displayed in the results. The present chapter illustrates how the situation of the undefined fault can be solved with the use of the proposed fuzzy ontology. The case five of the Roger's method "*Thermal fault less than 150* °C" with three datatypes $R_2 < 0.1$, $0.1 \leq R_1 \leq 1$ and $1 \leq R_5 \leq 3$ is investigated in this case. Assuming the presented gas ratios, R_1 and R_2 satisfy their conditions, while R_5 has the value 0.99, which is slightly below the crisp threshold value ($1 \leq R_5 \leq 3$). It is also known that, the actual fault is the thermal fault less than 150 °C. In this case, the Roger's method identifies the fault as case 0, which shows that there is no fault. There are also some other conditions that could not be defined in terms of Roger's method, which consequently results in the presence of undefined faults. These can be solved by applying the fuzzy datatypes instead of the crisp threshold, as presented in [110, 112].

The developed fuzzy ontology applies ten datatypes with different functionalities and boundaries. The first ratio R_2 from the Table 2.3 is modified as three ratios, $R21$, $R22$ and $R23$, with fuzzy modified datatypes. The other two ratios, R_1 and R_5, are modified in the similar way. The summary of proposed ratios is given in equation 6.2.

Table 6.2: Concrete syntax for the queries

Query	Semantics	Expression
(max-instance? a C)	$\sup\{n \mid \mathcal{K} \models \langle\ a:\ C, n\ \rangle\}$	maximal degree to which individual a is an instance of concept C
(min-instance? a C)	$\inf\{n \mid \mathcal{K} \models \langle\ a:\ C, n\ \rangle\}$	minimal degree to which individual a is an instance of concept C
(max-related? a b R)	$\sup\{n \mid \mathcal{K} \models \langle\ (a,b):\ R, n\ \rangle\}$	maximal degree to which individual pair (a,b) is an instance of role name R
(min-related? a b R)	$\inf\{n \mid \mathcal{K} \models \langle\ (a,b):\ R, n\ \rangle\}$	maximal degree to which individual pair (a,b) is an instance of role name R
(max-subs? C D)	$\sup\{n \mid \mathcal{K} \models \langle\ D \sqsubseteq C,\ n\ \rangle\}$	maximal degree to which concept C subsumes concept D
(min-subs? C D)	$\inf\{n \mid \mathcal{K} \models \langle\ D \sqsubseteq C,\ n\ \rangle\}$	minimal degree to which concept C subsumes concept D
(max-sat? C [a])	$\sup_{\mathcal{I}} \sup_{a} \epsilon \Delta^{\mathcal{I}} C^{\mathcal{I}}\ (a)$	maximal degree to which concept C is satisfiable
(min-sat? C[a])	$\inf_{\mathcal{I}} \sup_{a} \epsilon \Delta^{\mathcal{I}} C^{\mathcal{I}}\ (a)$	minimal degree to which concept C is satisfiable
(max-var? var)	$\sup\{\text{var} \mid \mathcal{K} \text{ is consistent}\}$	maximal value for variable var in [0,1], taking into account the constraints in KB
(min-var? var)	$\inf\{\text{var} \mid \mathcal{K} \text{ is consistent}\}$	minimal value for variable var in [0,1], taking into account the constraints in KB
(defuzzify-lom? C a t)	defuzzify t using LOM	Defuzzify the value of feature t with the largest of the maxima method.
(defuzzify-som? C a t)	defuzzify t using SOM	Defuzzify the value of feature t with the smallest of the maxima method.
(defuzzify-mom? C a t)	defuzzify t using MOM	Defuzzify the value of feature t with the middle of the maxima method.

TABLE 6.3: Show expression for fuzzy OWL2 statement

Statements	Expression
(show-concrete-fillers $f_1 ... f_n$)	show all fillers of concrete feature $f_1 ... f_n$ in an optimal solution to a query
(show-concrete-fillers-for a $f_1 ... f_n$)	for individual a, show all fillers of concrete feature $f_1 ... f_n$ in an optimal solution to a query
(show-variables $x_1 ... x_n$)	show the value of the variables $x_1 ... x_n$ in an optimal solution to a query
(show-instances $A_1 ... A_n$)	show all instances of atomic concepts $A_1 ... A_n$ and their degree in an optimal solution to a query
(show-concepts $a_1 ... a_n$)	show all atoms to which a_i is instance and their degree in an optimal solution to a query
(show-language)	show the description logic language of the $\mathcal{KB}$, from $\mathcal{ALC}$ to $\mathcal{SHIF}$
(show-abstract-fillers R $A_1 ... A_n$)	show the membership to atomic concepts $A_1 ... A_n$ of the fillers of R
(show-abstract-fillers-for a R $A_1 ... A_n$)	show the membership to atomic concepts $A_1 ... A_n$ of the fillers of R for individual a
(show-concrete–instance-for a f $A_1 ... A_n$)	show degrees of being the f filler of a an instance of concept A_i
(show-abstract-fillers $R_1 ... R_n$)	show fillers of $R_1 ... R_n$ and membership to any concept
(show-abstract-fillers-for a $R_1 ... R_n$)	show fillers of $R_1 ... R_n$ for a and membership to any concept

$$R_2 = \begin{Bmatrix} R21 \\ R22 \\ R23 \end{Bmatrix}, \quad R_1 = \begin{Bmatrix} R11 \\ R12 \\ R13 \end{Bmatrix}, \; and \;\; R_5 = \begin{Bmatrix} R51 \\ R52 \\ R53 \\ R54 \end{Bmatrix} \tag{6.2}$$

where the values of membership functions for each gas ratios are shown in the Figure 6.4.

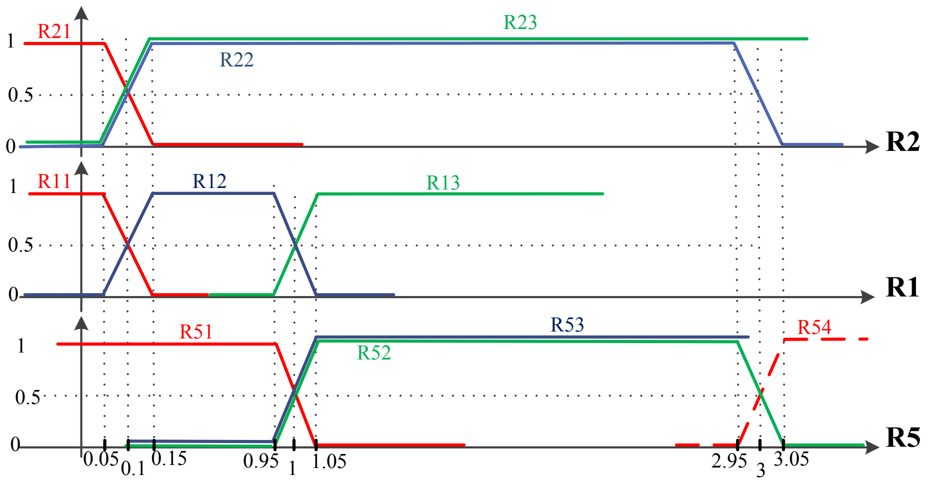


FIGURE 6.4: Proposed fuzzy datatypes for three gas ratios of the Roger's method

For instance, the datatype R21 is defined as a left shoulder function with parameters "a=0.05" and "b=0.15" (from the Figure 6.2 part C), instead of being crisp values of "a=b=0.1". This can be defined in fuzzy OWL2 with the following statement:

```
$<fuzzyOwl2 fuzzyType = "datatype">$
  $<Datatype type = "leftshoulder" a="0.05"  b="0.15" />$
$</fuzzyOwl2>$
```

The case study is applied for the Roger's method with actual DGA samples, to investigate how fuzzy ontology can improve the ontology developed earlier for transformer fault diagnosis. For this reason, the required classes and properties are designed, similar to the ontology described in the previous chapter.

TABLE 6.4: Revised Roger's method for fault diagnosis

Case	Fault Type	R_2	R_1	R_5
0	No fault	$R21$	$R12$	$R51$
1	Low energy partial discharge	$R22$	$R11$	$R51$
2	High energy partial discharge	$R22$	$R11$	$R51$
3	Low energy discharge, sparking, arcing	$R23$	$R12$	$R53$
4	High energy discharges, arcing	$R22$	$R12$	$R54$
5	Thermal fault temperature less than 150 °C	$R21$	$R12$	$R52$
6	Thermal fault temperature range 150-300 °C	$R21$	$R13$	$R51$
7	Thermal fault temperature range 300-700 °C	$R21$	$R13$	$R52$
8	Thermal fault temperature range over 700 °C	$R21$	$R13$	$R54$

6.2.1 Classes and properties

The software Protégé plug-in helps to make the syntax of the fuzzy ontology annotation [111]. Similarly to the developed ontology described in Chapter 5, the proposed fuzzy ontology for power transformer fault diagnosis contains various classes. Three subclasses, *Faults*, *Symptoms* and *Components* are chosen for the class *Transformer*. The subclass of faults, called *Rogers_Method_Faults*, represents nine cases of the Roger's method faults. These nine subclasses are restricted with fuzzy modified datatypes properties. Depending on the type of fault, various relevant classes and subclasses are applied for the fuzzy ontology. Screenshots provided in Appendix D show the classes, subclasses and applied restrictions for the developed fuzzy ontology.

To provide the restrictions on the selected classes, various types of properties are applied. One of the applied object properties is called *has_ratio*. The modified datatypes properties presented in equation 6.2 are also asserted, instead of using the crisp threshold properties. Therefore, the Roger's method can be revised using the fuzzy membership functions, as shown in the Table 6.4.

This can be illustrated with the first case of Roger's method in the form of fuzzy ontology. According to Table 6.4, the case 0 represents *No_Fault* situation with $R21$, $R12$ and $R51$ gas ratios conditions corresponding to it (their values are defined in the Figure 6.4). This can be presented in Protégé with the following statement:

$$\begin{split} No\_Faults\ EquivalentTo\ Faults\ and\ (has\_ratio\ some\ R21)\ and\ (has\_ratio \\ some\ R12)\ and\ (has\_ratio\ some\ R51) \end{split} \tag{6.3}$$

while in the fuzzyDL form, this statement is defined as:

Case 0:

$$\begin{split} No\_Fault \equiv Faults\ \sqcap\ (\exists\ has\_ratio.R21)\ \sqcap\ (\exists\ has\_ratio.R12)\ \sqcap \\ (\exists\ has\_ratio.R51) \end{split} \tag{6.4}$$

which expresses the case of having no fault with corresponding three conditions of gas ratios. The OWL2 syntax expressing this statement is given in Appendix D. The cases 1 and 2 of the Roger's method present similar conditions for the *Partial Discharge* fault (low and high energy); and they can be merged into one case. The rest of the Roger's method cases are formalized in the TBox as given below:

Case 1&2:

$$\begin{split} Partial\_Discharge \equiv Faults\ \sqcap\ (\exists\ has\_ratio.R22)\ \sqcap\ (\exists\ has\_ratio.R11)\ \sqcap \\ (\exists\ has\_ratio.R51) \end{split} \tag{6.5}$$

Case 3:

$$\begin{split} Low\_Energy\_Discharge, Sparking, Arcing \equiv Faults\ \sqcap\ (\exists\ has\_ratio.R23)\ \sqcap \\ (\exists\ has\_ratio.R12)\ \sqcap\ (\exists\ has\_ratio.R53) \end{split} \tag{6.6}$$

Case 4:

$$\begin{split} High\_Energy\_Discharge, Arcing \equiv Faults\ \sqcap\ (\exists\ has\_ratio.R22)\ \sqcap\ (\exists \\ has\_ratio.R12) \sqcap\ (\exists\ has\_ratio.R54) \end{split} \tag{6.7}$$

Case 5:

$$\begin{split} Thermal\_Fault\_Temperature\_Less\_Than\_150\ ^{\circ}\mathrm{C} \equiv Faults\ \sqcap\ (\exists\ has\_ratio \\ .R21) \sqcap\ (\exists\ has\_ratio.R12)\ \sqcap\ (\exists\ has\_ratio.R52) \end{split} \tag{6.8}$$

Case 6:

$$Thermal\_Fault\_Temperature\_between\_150\_300\ ^{\circ}\mathrm{C} \equiv\ Faults\ \sqcap\ (\exists\ has\_ratio .R21)\ \sqcap\ (\exists\ has\_ratio.R13)\ \sqcap\ (\exists\ has\_ratio.R51) \tag{6.9}$$

Case 7:

$$Thermal\_Fault\_Temperature\_between\_300\_700\ ^{\circ}\mathrm{C} \equiv Faults\ \sqcap\ (\exists\ has\_ratio .R21)\ \sqcap\ (\exists\ has\_ratio.R13)\ \sqcap\ (\exists\ has\_ratio.R52) \tag{6.10}$$

Case 8:

$$Thermal\_Fault\_Temperature\_Over\_700\ ^{\circ}\mathrm{C} \equiv Faults\ \sqcap\ (\exists\ has\_ratio.R21) \sqcap (\exists\ has\_ratio.R13)\ \sqcap\ (\exists\ has\_ratio.R54) \tag{6.11}$$

The relevant screenshots for the developed fuzzy ontology, including classes, properties, fuzzy datatype, etc., are provided in the Appendix D.

6.2.2 The experimental results for fault diagnosis with fuzzy ontology

Various types of programming software, such as Protégé, Java, Gurobi, etc., were used in this work to apply and test the fuzzy ontology. The actual data and the corresponding faults from Table 4.2 are investigated with the help of fuzzy ontology. The accuracy of fault diagnosis is improved by applying the fuzzy membership functions instead of crisp thresholds. For this purpose, the values of membership functions of the gas ratios are found and investigated. In the case of membership function value is equivalent to one, the developed method performs similarly to the ontology described in the previous chapter. For the membership function value below one, the conclusion on the fault type will be made considering whether the gas ratio combination belongs to the one (or more) of the defined classes. This can be illustrated with an example of actual gas ratios (as provided in Table 6.5). A fault type for the set of gas ratio values (R_2=3.25, R_1=0.08, R_5=17.75) could not be defined with the ontology, while using the developed fuzzy ontology can solve this problem. With the help of fuzzy ontology the following membership functions were obtained:

$\mu(R_2 = 3.25) = \mu(R23) = 1$

$$\mu(R_1 = 0.08) = \left\{ \begin{array}{c} \mu(R11) = 0.7 \\ \mu(R12) = 0.3 \end{array} \right\}$$

$$\mu(R_5 = 17.75) = \left\{ \begin{array}{c} \mu(R53) = 1 \\ \mu(R54) = 1 \end{array} \right\}$$

Therefore, the conclusions on the several combinations of the participating datatypes are made as follows:

1. $R23 \sqcap R11 \sqcap R53 \Rightarrow$ There is no fault defined with this condition.
2. $R23 \sqcap R11 \sqcap R54 \Rightarrow$ There is no fault defined with this condition.
3. $R23 \sqcap R12 \sqcap R54 \Rightarrow$ There is no fault defined with this condition.
4. $R23 \sqcap R12 \sqcap R53 \Rightarrow$ The conditions match with the fault in case 3.

Verifying the combination of membership functions with the table of defined classes (Table 6.4) allows to identify the fault type as “Low energy partial discharge, sparking, arcing” (case 3).

The overall accuracy can be increased in the case of applying more DGA samples in the threshold areas. As the same actual data were used in all cases, it can be easily compared to the results of proposed rule-based reasoning and ontology methods. Table 6.5 shows 20 results of the developed fuzzy ontology for power transformer fault diagnosis.

As shown in Table 6.5, the number of undefined faults is reduced to three cases only. Summary of the results for the 70 DGA samples is given in Table 6.6. As shown in Table 6.6, two cases of partial discharge faults were diagnosed correctly, and two cases of undefined faults were verified as the arcing faults. The overall accuracy increased from 72.86% to 95.71% by applying fuzzy ontology.

6.2.3 Discussion

Complex problems can be solved with knowledge-based systems using the reasoning techniques. It is also important for the knowledge-base system to be conceptualised. As mentioned earlier, the significant disadvantage of early expert system was having both domain knowledge and rules within the same knowledge base. The knowledge-based systems utilizing rule-based reasoning were facing this problem. Having a proper

TABLE 6.5: The actual DGA samples applied with fuzzy ontology-based reasoning

R_2	R_1	R_5	**Actual Fault**	**Fuzzy Ontology Results**	**Results**
1.16	0.46	5.2	Arcing	Low Energy Partial Discharge, Arcing, Sparking	Correct
0.07	5.43	5.26	Overheating	Thermal Fault(TF) $TF > 700°C$	Correct
1.65	0.17	3.13	Arcing	Low Energy Partial Discharge, Arcing, Sparking	Correct
1.06	1.74	9.26	Arcing	Undefined Fault	ND*
0.04	3.86	6.94	Overheating	Thermal Fault $TF > 700°C$	Correct
0.97	1.79	7.06	Arcing	Undefined Fault	ND*
0.01	40.99	5.07	Overheating	Thermal Fault $TF > 700°C$	Correct
3.25	0.08	17.75	Partial Discharges	Low Energy Partial Discharge, Arcing, Sparking	Correct
0.02	3.09	7.44	Overheating	Thermal Fault $TF > 700°C$	Correct
0.01	1.42	10.02	Overheating	Thermal Fault $TF > 700°C$	Correct
2.74	1.54	13.42	Arcing	Undefined Fault	ND*
0.01	2.69	8.62	Overheating	Thermal Fault $TF > 700°C$	Correct
2.93	0.09	6.6	Arcing	Low Energy Partial Discharge, Arcing, Sparking	Correct
2.26	0.29	10.82	Arcing	Low Energy Partial Discharge, Arcing, Sparking	Correct
3.42	0.08	5.6	Partial Discharges	Low Energy Partial Discharge, Arcing, Sparking	Correct
0.02	2.39	7.16	Overheating	Thermal Fault $TF > 700°C$	Correct
3.3	0.07	16.5	Arcing	Low Energy Partial Discharge, Arcing, Sparking	Correct
0.02	2.4	6.7	Overheating	Thermal Fault $TF > 700°C$	Correct
0	4.85	1.85	Overheating	Thermal Fault $300 < TF < 700°C$	Correct
1.45	0.84	14	Arcing	Low Energy Partial Discharge, Arcing, Sparking	Correct

* ND = Not Defined

concept of the knowledge enables the separation of domain and application of knowledge, however the problem of separation of rules remains unsolved. This can be done with the help of ontology. For this reason, an ontology-based knowledge representation was developed for the purpose of power transformer fault diagnosis.

The interaction between agent technology and the developed ontology for transformer fault diagnosis is able to facilitate and automate the actions. The key advantage of proposed system is the ability to revise the knowledge-based system with newer ontology with better performance. This also can be done with help of agents technology. The fuzzy ontology has never been applied to date for power transformer fault diagnosis.

Table 6.6: Summary of fuzzy ontology-based reasoning for fault diagnosis

Fault Types	**Total Samples**	**Diagnosed Correctly**	**Not Defined Fault**	**Accuracy**
No Fault	1	1	0	100%
Partial Discharge	2	2	**0**	**100%**
Arcing	14	11	**3**	**78.57%**
Overheating	25	25	0	100%
Low Energy Discharge	28	28	0	100%
Average Total Accuracy	—	—	—	**95.71%**

In this study, the simple fuzzy system is investigated, while on the latter stages it can be used with more significant threshold boundaries of the fuzzy system for DGA fault diagnosis.

There are various techniques of power transformer fault diagnosis based on DGA samples developed by researchers. Ontology-based knowledge representation for power system is applied in [3, 6], where it aims to provide the real-time information to the user. Therefore, the system described in [3, 6] has no ability of fault diagnosis, while the proposed system is able to do so.

Ontology was applied for power transformer fault diagnosis in some of the literature sources found, such as [9, 12]. In both cases, the proposed ontologies were not able to handle the uncertain conditions. The ontology proposed in [9] uses the object properties only, and therefore requires an additional programming for fault classification. However, agent does not participate in this work, and experts are required to carry out the fault diagnosis. Application of datatype properties, similarly to this work, would reduce the number of classes representing the fault types. The system developed in this work does not require any additional programming for fault diagnosis. The overall accuracy for the developed system have been increased by applying different methods. A summary of results for three types of developed systems is given in Table 6.7.

As shown in the Table 6.7, the identical values of the accuracy were obtained with the use of rule-based system and ontology, while for the fuzzy ontology this value was higher. The total accuracy of the fault diagnosis with the fuzzy ontology applied has increased by 22.85% compared to the other two methods. As the real time power transformer

Table 6.7: Results summary of developed systems for fault diagnosis

Fault Types	Rule-based	Ontology	Fuzzy Ontology	Increased Accuracy
No Fault	100%	100%	100%	—
Partial Discharge	0%	0%	100%	100%
Arcing	64.30%	64.30%	78.57%	14.27%
Overheating	100%	100%	100%	—
Low Energy Discharge	100%	100%	100%	—
Average Total Accuracy	**72.86%**	**72.86%**	**95.71%**	**22.85%**

fault diagnosis always involves some degree of uncertainty, therefore the use of ontology in this case might be limited. Fuzzy ontology is capable of dealing with the situations involving some uncertainty better, and the applications of it to power transformer fault diagnosis provides better results.

6.3 Summary

A fuzzy ontology developed for power transformer fault diagnosis, based on the Roger's method, was presented in this chapter. Three types of fuzzy modified datatypes, namely left-shoulder, right-shoulder and trapezoidal functions, were applied to revise the Roger's method. The developed fuzzy ontology applies various classes and subclasses, restricted by different types of properties. Finally, the actual DGA samples were tested; the results were compared to the work described previously, such as rule-based reasoning and ontology. The results showed that the use of fuzzy ontology allowed to increase the fault diagnosis total accuracy by 22.85% compared to two other methods.

Chapter 7

Conclusion and Future work

7.1 Conclusion

This thesis has described a multi-agent system using a knowledge base for the purpose of performing automatic actions and fault diagnosis of the power transformer. MAS is the main component of this architecture; it consists of a FIPA standard agent platform and number of various types of agents. MAS provides condition monitoring, performs various actions and fault diagnosis in power system. Known conditions of the power system component, here – power transformer, are collected into the knowledge-based system for the purpose of fault diagnosis. Three types of knowledge representation are applied in this study for power transformer fault diagnosis and performing automatic actions in case of fault situation. The developed knowledge-based systems are rule-based reasoning, ontology-based reasoning and fuzzy ontology.

Initially, the agent architecture is developed based on the Gaia methodology to clarify, simplify and standardize the design of the MAS. The Gaia methodology has been applied for first time in power transformer condition monitoring and fault diagnosis in combination with knowledge-based systems. The developed agent system is able to monitor the condition of the component (power transformer), diagnose its fault, and perform an action, if required. Furthermore, several types of knowledge-based systems are developed in order to improve the system versatility. Accuracy of the fault diagnosis based on the use of fuzzy ontology was shown to be increased by 22.85%, for the identical data samples used. The agent architecture for power transformer condition monitoring and real-time fault diagnosis can be applied to different industrial situations that require SCADA system to reduce the human efforts required, as well as equipment maintenance cost reduction.

7.1.1 Summary of thesis

Chapter 2 described various types of applied automated systems in industry and the background of power system automation. Some aspects of agents and multi-agent systems, including definition, architectures, design methodologies, standards and their applications, have been reviewed. Power system components, power transformers and the components required for condition monitoring were introduced. DGA, as a common type of fault diagnosis technique, with various methods, was described. Finally, the current applications of MAS in power system automation, ontology, condition monitoring, etc. were reviewed.

Chapter 3 described the overall hierarchy of the developed multi-agent framework for the power transformer condition monitoring and fault diagnosis. First, the Gaia methodology was chosen to analyse and design the MAS step by step. Various types of agents for the purpose of condition monitoring, controlling and performing automatic actions were developed. Three types of knowledge based systems were proposed to provide information to the MAS.

Agent collaboration for three types of tasks, real-time data collection and fault diagnosis, user interaction and automatic action performing, were investigated. Various types of software, such as MySQL database, JDBC, JFreeChart, etc., were applied in order to develop the system. Agent *Analyser* was developed to interact with MATLAB program for the case of fault diagnosis. A classification method based on machine learning, KNN, was applied for the fault classification. A total number of 191 DGA samples was tested using key gases and gas ratios individually. An experimental system was applied for the purpose of agents collaboration with the knowledge based system. The experimental system used actual data, such as bottom oil temperature and ambient temperature, etc., for purpose of data collection, monitored the information for the user and replied to the request of report.

Chapter 4 described the developed knowledge-based system for power transformer fault diagnosis. Rule-based reasoning has been successfully applied to present information of the power transformer status. Two applications of rule-based reasoning in cooperation with MAS were proposed. For the first application rule-based reasoning was implemented to perform automatic action in power system and its component. For this purpose two types of experimental tests were investigated. First test used performing of automatic actions in power transformer. For this purpose, the information of thermal fault, given by IEC standard publication, was loaded into the knowledge-based system. Based on receiving the real-time data, such as transformer winding temperature, an appropriate action could be performed. Second test was applied for performing an automatic action in power system based on fault diagnosis.

The second application of the rule-based reasoning was applied for power transformer fault diagnosis based on Roger's method. Actual data samples were investigated to show the fault diagnosis performance. Various types of agents, such as data sender, knowledge base agent, controller and user agents, were utilised to perform the actions. The knowledge-based system used JESS program for the interaction of the required information of the transformer and power system with agent system. The accuracy of the fault diagnosis for 70 DGA samples was evaluated as 72.88%.

Chapter 5 described the ontology-based reasoning for power transformer fault diagnosis. The OWL-DL language was applied to develop the ontology. Various classes and subclasses were asserted and restricted by different types of properties. The developed ontology based on OWL-DL with DL reasoner had an ability to infer the hierarchy classes, subclasses, inconsistency, etc. The way developed ontology could extract implicit information from the explicit facts built in, was also reviewed in this chapter. An agent was designed to wrap the ontology for the purpose of interaction. In the study conducted, an ontology agent was able to receive the DGA samples and pass them to the ontology for fault diagnosis. In this case study, the Roger's method was applied in form of OWL-DL ontology. Finally, the applied 70 DGA sample were investigated, and the accuracy of discussed fault diagnosis method was assessed.

Chapter 6 described the novel application of the fuzzy ontology in power system, that was capable of dealing with the uncertainty. The advantages of fuzzy ontology and some examples of its application for power transformer fault diagnosis were discussed. To investigate the improvement of the fuzzy ontology compared to the previously developed ontology, the fault diagnosis based on Roger's method was applied. The accuracy of developed fuzzy ontology was evaluated using the same DGA samples as previously; then the overall accuracy was assessed. It was shown that the use of fuzzy ontology allowed to improve accuracy by over 22 %, compared to the other types of knowledge-based systems discussed earlier.

7.2 Future Research

Due to the time limitation and the broadness of the research field the work was carried out at, it was not possible to investigate all the possible methods in terms of this project. This section provides several suggestions that might be used for future work conducted in the relevant area.

- **MAS decision maker based on fuzzy ontology in power system**

  Fuzzy ontology provides the semantic annotations based on logic for dealing with uncertain knowledge. The present thesis describes a limited number of the agents developed. Various types of behaviour, such as cyclic, parallel, etc., can be implemented for improving the agent system. However, one feature of the fuzzy ontology is to implement the relative importance of every criterion with decision alternatives, by assigning a weight to it. This feature can be applied for the power transformer fault diagnosis for the case of diagnosed various types of faults. In case of x decision alternatives and a set of y criteria according to which the desirability of the fault type can be judged. The use of developed agent framework with knowledge-based system combined with the features of fuzzy ontology described above can provide an additional abilities to be used in power system.

- **BDI-agent software for power system automation**

Agent technology is an impressive technique that can be applied in power system for condition monitoring and automation of the components. Depending on the application, the agent architectures may vary to be able to cooperate for achieving the goals. In this work the developed agent system for transformer fault diagnosis uses the reactive architecture. This type of architecture was chosen based on its ability to respond actively to the fault situation. However, in case of electricity marketing and utilities, long term decision makers based on agent system observation and goal direction are required. The BDI agent architecture is a useful architecture can be applied for those type of applications in power system.

- **Fault diagnosis with non-linear classification method**

  The interaction of the proposed multi-agent framework with KNN classifier were successful. A linear classifier method, KNN, was applied to evaluate the accuracy. For the purpose of increasing the accuracy of the fault diagnosis, various other methods can be applied. For instance, using non-liner classification, such as SVM, can improve the overall accuracy.

- **Knowledge-based learning system**

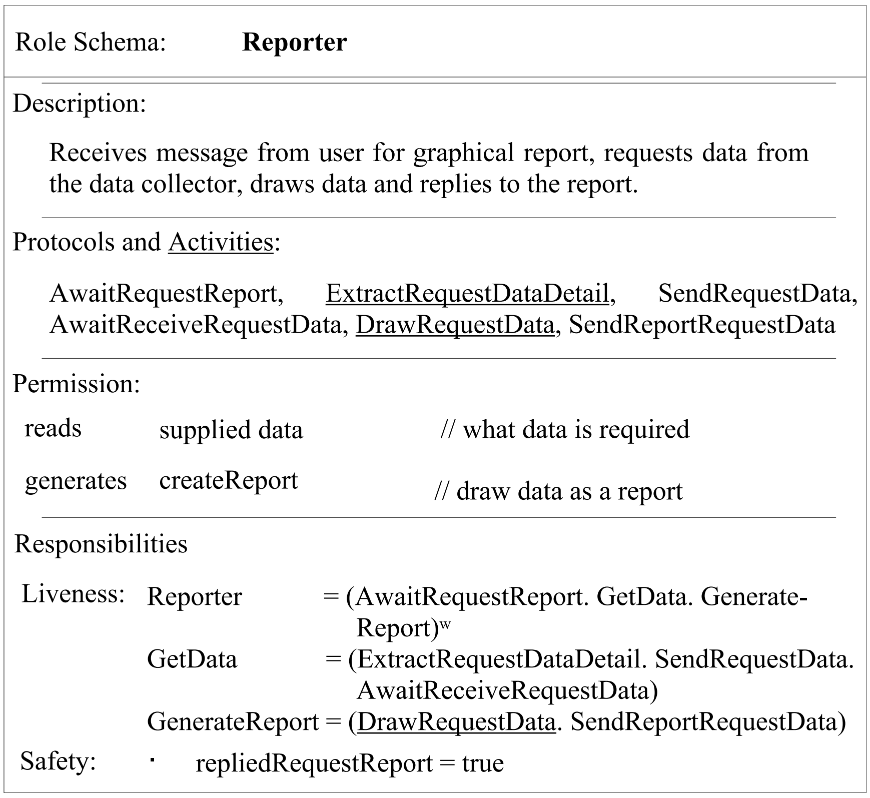
  The complexity of real-world problems often require complicated methods and tools for building on-line, knowledge-based intelligent systems. The multi-agent framework described in this thesis does not possess the learning method. Therefore the proposed system can be potentially improved with introducing the learning ability to it. Thus, the learning ability of the knowledge base means that the system attempts to complete the missing knowledge and reduce non-reliability of the communication process between man and machine [115].

Appendix A

The Gaia Methodology Design for the System

A.1 Role Schema of Gaia Methodology

Figures A.1 to A.7 represent the role schemas of Gaia methodology for the remain roles.



Role Schema: **Reporter**

Description:

Receives message from user for graphical report, requests data from the data collector, draws data and replies to the report.

Protocols and Activities:

AwaitRequestReport, ExtractRequestDataDetail, SendRequestData, AwaitReceiveRequestData, DrawRequestData, SendReportRequestData

Permission:

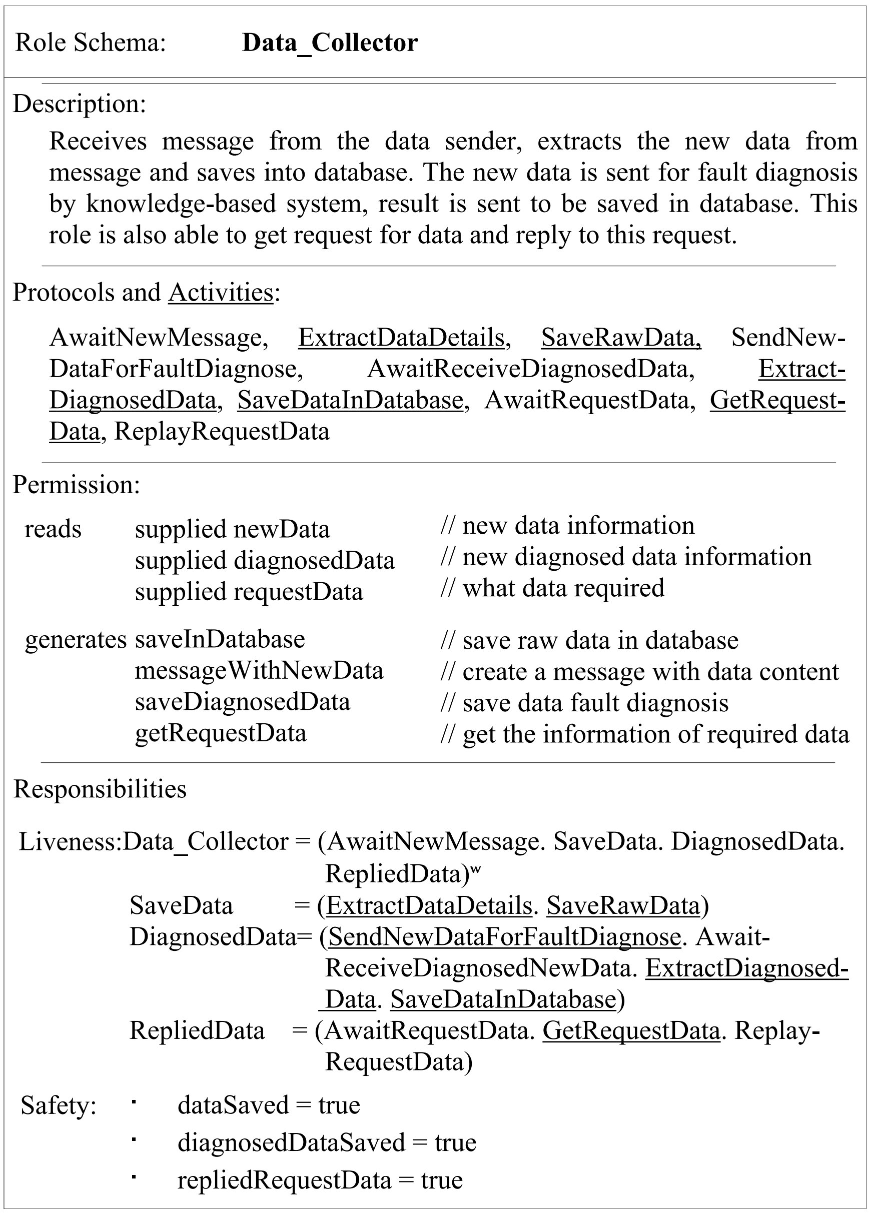
reads	supplied data	// what data is required
generates	createReport	// draw data as a report

Responsibilities

Liveness: Reporter = (AwaitRequestReport. GetData. Generate-Report)^{w}
GetData = (ExtractRequestDataDetail. SendRequestData. AwaitReceiveRequestData)
GenerateReport = (DrawRequestData. SendReportRequestData)

Safety: · repliedRequestReport = true

Figure A.1: The “Reporter” role schema



Role Schema: **Data_Collector**

Description:

Receives message from the data sender, extracts the new data from message and saves into database. The new data is sent for fault diagnosis by knowledge-based system, result is sent to be saved in database. This role is also able to get request for data and reply to this request.

Protocols and Activities:

AwaitNewMessage, ExtractDataDetails, SaveRawData, SendNewDataForFaultDiagnose, AwaitReceiveDiagnosedData, ExtractDiagnosedData, SaveDataInDatabase, AwaitRequestData, GetRequestData, ReplayRequestData

Permission:

reads	supplied newData	// new data information
	supplied diagnosedData	// new diagnosed data information
	supplied requestData	// what data required
generates	saveInDatabase	// save raw data in database
	messageWithNewData	// create a message with data content
	saveDiagnosedData	// save data fault diagnosis
	getRequestData	// get the information of required data

Responsibilities

Liveness:Data_Collector = (AwaitNewMessage. SaveData. DiagnosedData. RepliedData)^{w}

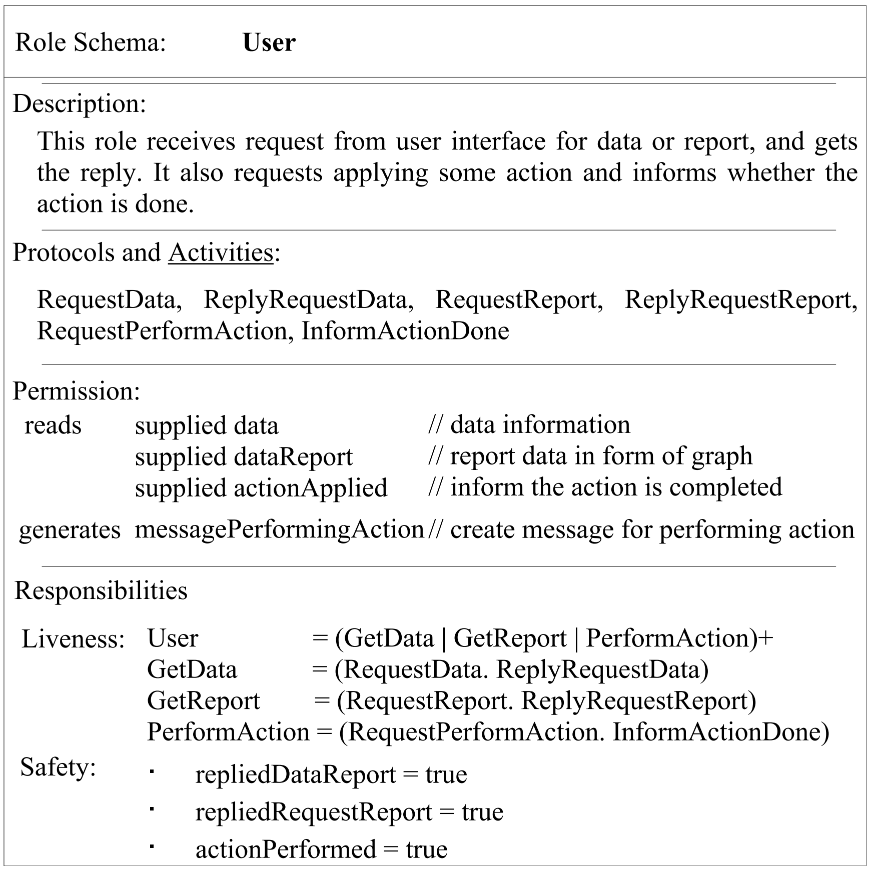
SaveData = (ExtractDataDetails. SaveRawData)

DiagnosedData= (SendNewDataForFaultDiagnose. AwaitReceiveDiagnosedNewData. ExtractDiagnosedData. SaveDataInDatabase)

RepliedData = (AwaitRequestData. GetRequestData. ReplayRequestData)

Safety:
- dataSaved = true
- diagnosedDataSaved = true
- repliedRequestData = true

Figure A.2: The “Data_Collector” role schema



Role Schema: **User**

Description:

This role receives request from user interface for data or report, and gets the reply. It also requests applying some action and informs whether the action is done.

Protocols and Activities:

RequestData, ReplyRequestData, RequestReport, ReplyRequestReport, RequestPerformAction, InformActionDone

Permission:

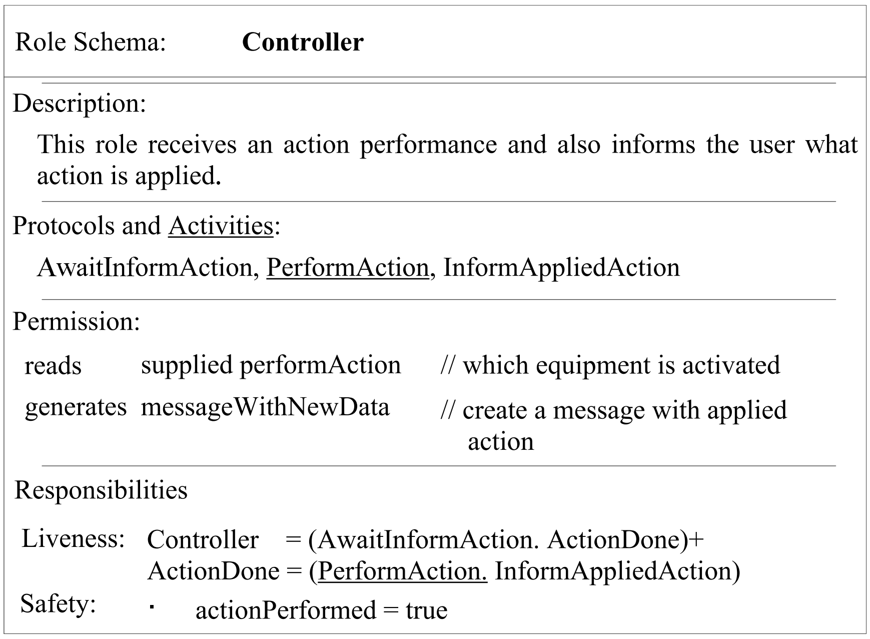
reads	supplied data	// data information
	supplied dataReport	// report data in form of graph
	supplied actionApplied	// inform the action is completed
generates	messagePerformingAction	// create message for performing action

Responsibilities

Liveness: User = (GetData | GetReport | PerformAction)+
GetData = (RequestData. ReplyRequestData)
GetReport = (RequestReport. ReplyRequestReport)
PerformAction = (RequestPerformAction. InformActionDone)

Safety:
- repliedDataReport = true
- repliedRequestReport = true
- actionPerformed = true

Figure A.3: The "User" role schema



Role Schema: **Controller**

Description:

This role receives an action performance and also informs the user what action is applied.

Protocols and Activities:

AwaitInformAction, PerformAction, InformAppliedAction

Permission:

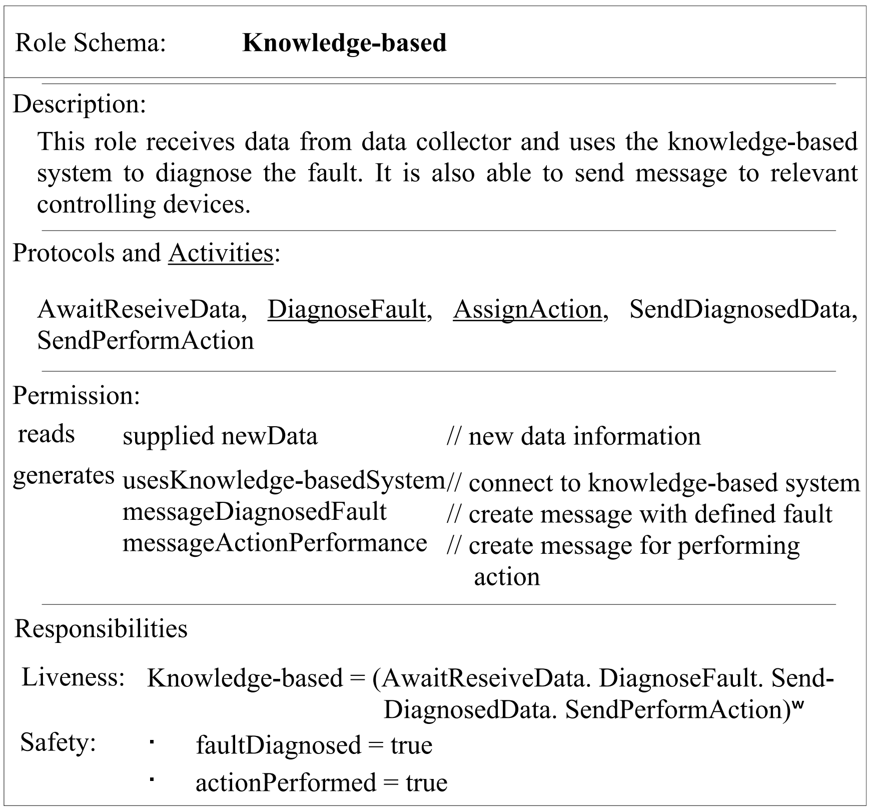
reads	supplied performAction	// which equipment is activated
generates	messageWithNewData	// create a message with applied action

Responsibilities

Liveness: Controller = (AwaitInformAction. ActionDone)+
ActionDone = (PerformAction. InformAppliedAction)

Safety:
- actionPerformed = true

Figure A.4: The "Controller" role schema



Role Schema: **Knowledge-based**

Description:

This role receives data from data collector and uses the knowledge-based system to diagnose the fault. It is also able to send message to relevant controlling devices.

Protocols and <u>Activities</u>:

AwaitReseiveData, <u>DiagnoseFault</u>, <u>AssignAction</u>, SendDiagnosedData, SendPerformAction

Permission:

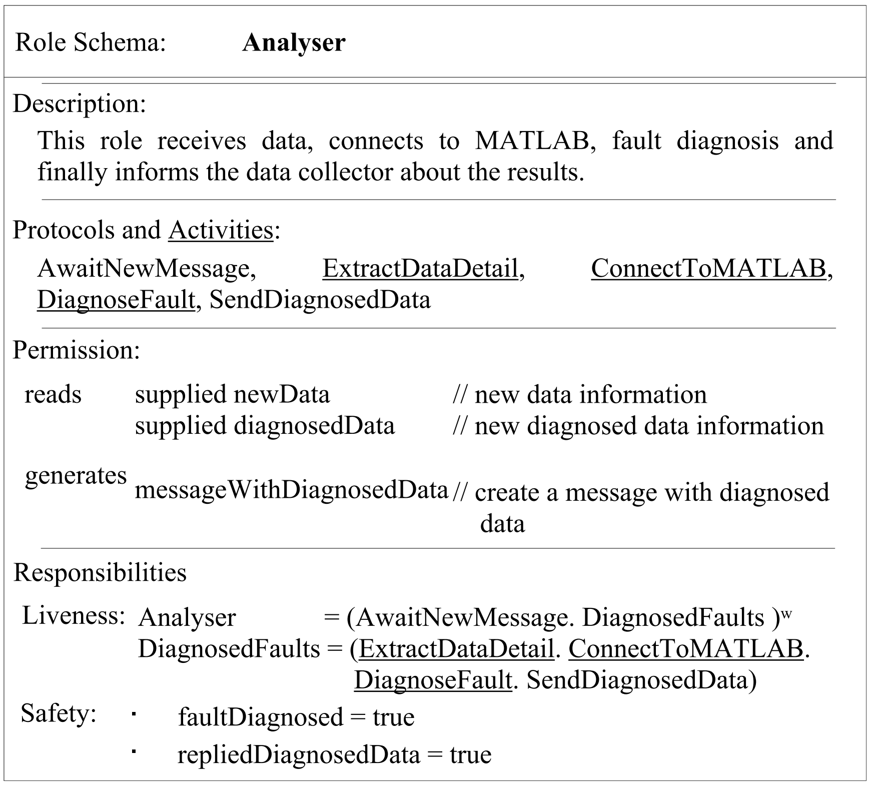
reads	supplied newData	// new data information
generates	usesKnowledge-basedSystem	// connect to knowledge-based system
	messageDiagnosedFault	// create message with defined fault
	messageActionPerformance	// create message for performing action

Responsibilities

Liveness: Knowledge-based = (AwaitReseiveData. DiagnoseFault. Send-DiagnosedData. SendPerformAction)^{w}

Safety:
- faultDiagnosed = true
- actionPerformed = true

FIGURE A.5: The "Knowledge_Base" role schema



Role Schema: **Analyser**

Description:

This role receives data, connects to MATLAB, fault diagnosis and finally informs the data collector about the results.

Protocols and Activities:

AwaitNewMessage, ExtractDataDetail, ConnectToMATLAB, DiagnoseFault, SendDiagnosedData

Permission:

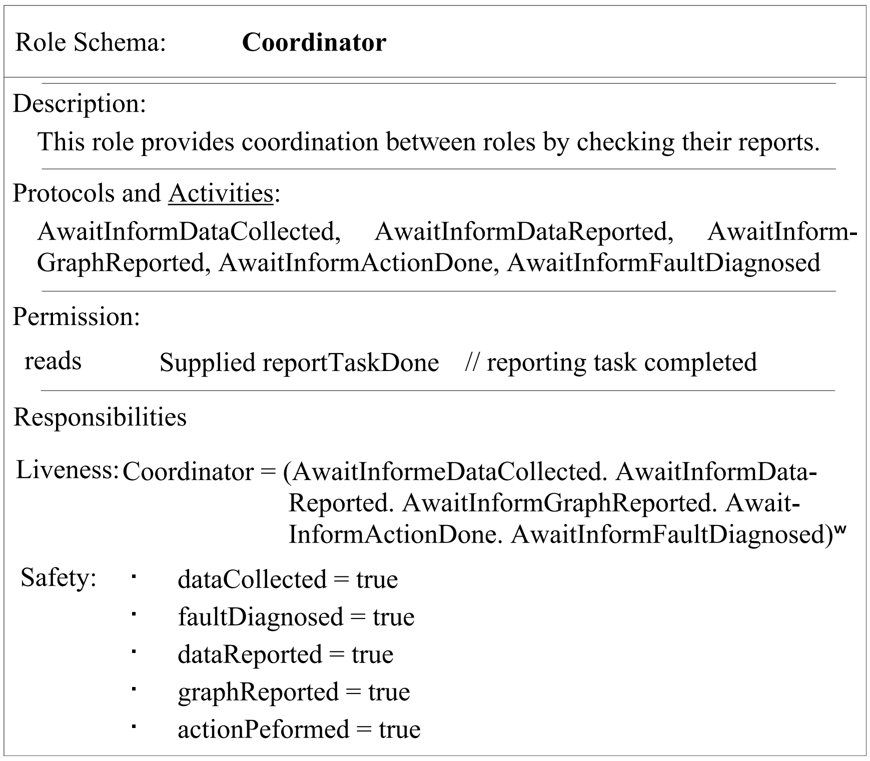
reads	supplied newData	// new data information
	supplied diagnosedData	// new diagnosed data information
generates	messageWithDiagnosedData	// create a message with diagnosed data

Responsibilities

Liveness: Analyser = (AwaitNewMessage. DiagnosedFaults)^{w}
DiagnosedFaults = (ExtractDataDetail. ConnectToMATLAB. DiagnoseFault. SendDiagnosedData)

Safety:
- faultDiagnosed = true
- repliedDiagnosedData = true

FIGURE A.6: The "Analyser" role schema



Role Schema: **Coordinator**

Description:

This role provides coordination between roles by checking their reports.

Protocols and Activities:

AwaitInformDataCollected, AwaitInformDataReported, AwaitInform-GraphReported, AwaitInformActionDone, AwaitInformFaultDiagnosed

Permission:

reads Supplied reportTaskDone // reporting task completed

Responsibilities

Liveness: Coordinator = (AwaitInformeDataCollected. AwaitInformData-Reported. AwaitInformGraphReported. Await-InformActionDone. AwaitInformFaultDiagnosed)^{w}

Safety:
- dataCollected = true
- faultDiagnosed = true
- dataReported = true
- graphReported = true
- actionPeformed = true

FIGURE A.7: The "Coordinator" role schema

A.2 Interaction Model for Developed MAS

Figures A.8 to A.11 represent the definition of protocol associated with the roles.

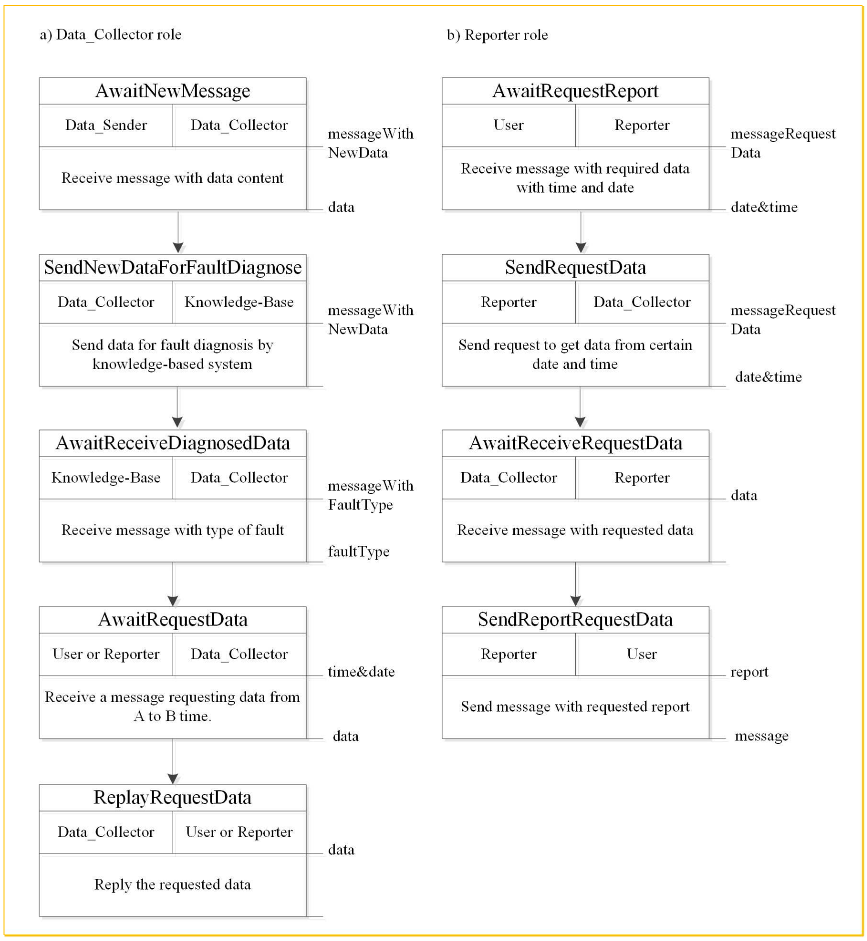


FIGURE A.8: Protocol definition associated with roles a) "Data_Collector" b) "Reporter"

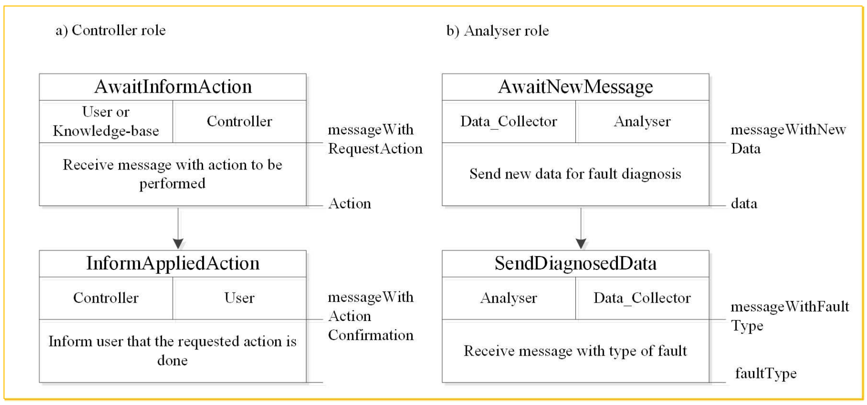


FIGURE A.9: Protocol definition associated with roles a) "Controller" b) "Analyser"

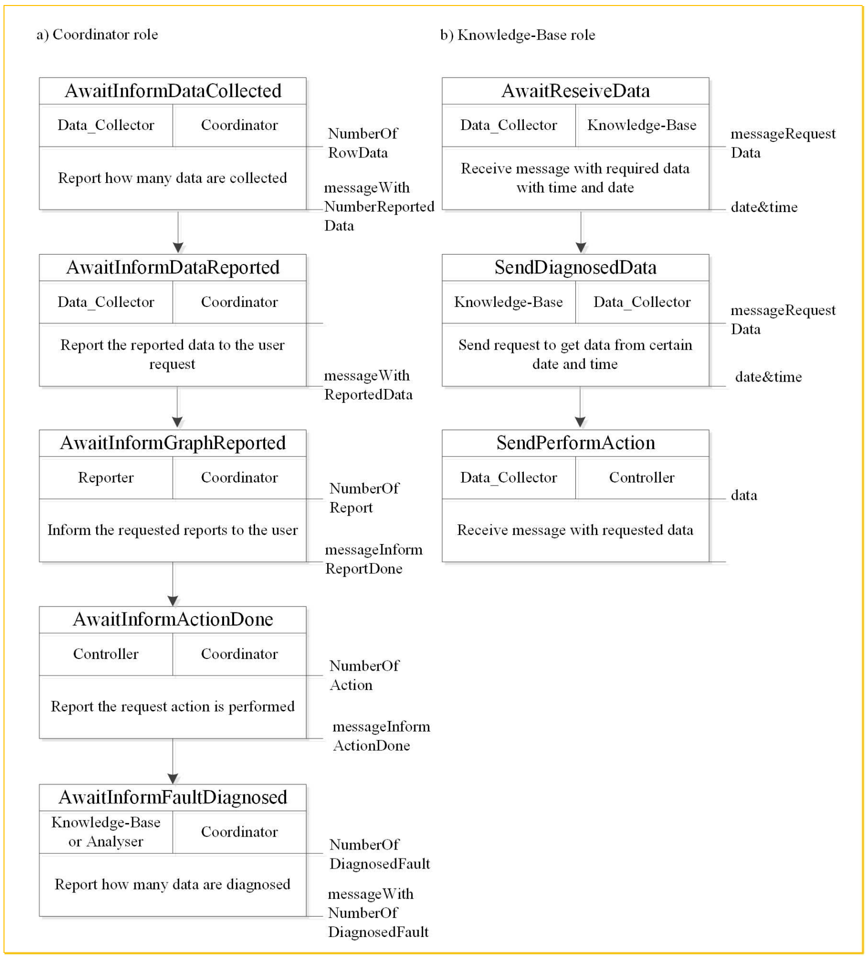


Figure A.10: Protocol definition associated with roles a) "Coordinator" b) "Knowledge_Base"

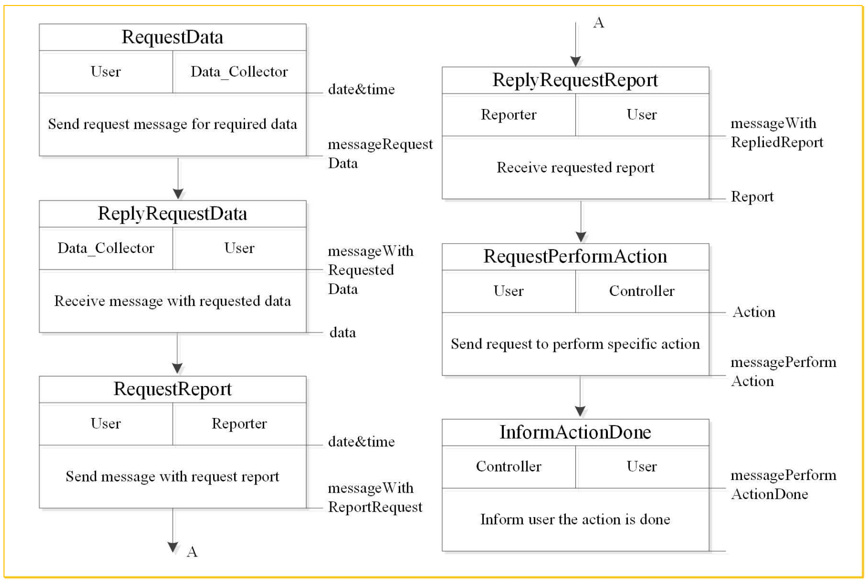
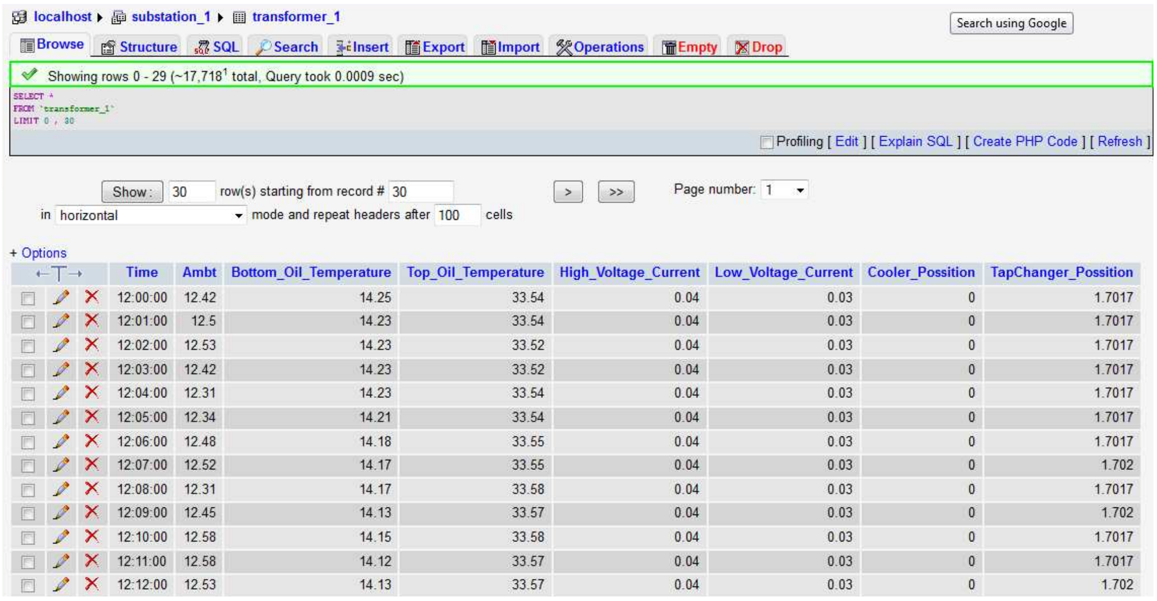


Figure A.11: Protocol definition associated with "User" role

A.3 Database Contents

Figure A.12 provides an example of database content, filled by the *Data_Collector* agent then stored the data in the related database and table.



Showing rows 0 - 29 (~17,718[1] total, Query took 0.0009 sec)

```
SELECT *
FROM `transformer_1`
LIMIT 0 , 30
```

Show : 30 row(s) starting from record # 30 > >> Page number: 1

in horizontal mode and repeat headers after 100 cells

Time	Ambt	Bottom_Oil_Temperature	Top_Oil_Temperature	High_Voltage_Current	Low_Voltage_Current	Cooler_Possition	TapChanger_Possition
12:00:00	12.42	14.25	33.54	0.04	0.03	0	1.7017
12:01:00	12.5	14.23	33.54	0.04	0.03	0	1.7017
12:02:00	12.53	14.23	33.52	0.04	0.03	0	1.7017
12:03:00	12.42	14.23	33.52	0.04	0.03	0	1.7017
12:04:00	12.31	14.23	33.54	0.04	0.03	0	1.7017
12:05:00	12.34	14.21	33.54	0.04	0.03	0	1.7017
12:06:00	12.48	14.18	33.55	0.04	0.03	0	1.7017
12:07:00	12.52	14.17	33.55	0.04	0.03	0	1.702
12:08:00	12.31	14.17	33.58	0.04	0.03	0	1.7017
12:09:00	12.45	14.13	33.57	0.04	0.03	0	1.702
12:10:00	12.58	14.15	33.58	0.04	0.03	0	1.7017
12:11:00	12.58	14.12	33.57	0.04	0.03	0	1.7017
12:12:00	12.53	14.13	33.57	0.04	0.03	0	1.702

FIGURE A.12: Contents of database

Appendix B

Rule-based Reasoning

B.1 Sample Code of JESS Rules

The following JESS rules are written for the loading and cooling conditions of power transformer based on IEC standard recommendations.

```
/*
 * IEC_Standard.clp
 * Create on 27 May 2011, 14:15
 * author F. Davoodi Samirmi
 */
********Template*******
(deftemplate Component_Temperature
    (slot Winding_Temperature)
    (slot  Agent)
)

********Rules**********
Rule0:
  (defrule st
      (initial-fact)
       =>
      (printout t "Jess engine started!" crlf))
```

```
Rule1:

  (defrule Control1_Component_Temperature

      (Component_Temperature{Winding_Temperature < 50}(Agent ?agent))

       =>

      (send "Report: Cooler System = OFF" ?User)

      (assert (Transformer CoolerSystem_OFF))

  )


Rule2:

  (defrule Control2_Component_Temperature

      (Component_Temperature {Winding_Temperature >= 50 &&

        Winding_Temperature < 75}

      (Agent ?agent))

       =>

      (send "Report: Cooler System = OFF" ?User)

      (send "Report: No Fault" ?agent)

      (assert (Transformer CoolerSystem_OFF))

  )


Rule3:

  (defrule Control3_Component_Temperature

      (Component_Temperature{Winding_Temperature >= 75}(Agent ?agent))

       =>

      (send "Report: Cooler System = ON" ?User)

      (assert (Transformer CoolerSystem_ON))

  )


Rule4:

  (defrule Control4_Component_Temperature

      (Component_Temperature{Winding_Temperature >= 105}(Agent ?agent))

       =>
      (send "Report: Alarm Signal = ON" ?User)
      (assert (Transformer AlarmSignal_ON))
  )


Rule5:
  (defrule Control5_Component_Temperature
      (Component_Temperature{Winding_Temperature >= 120}(Agent ?agent))
       =>
      (send "Report: Trip Signal = ON" ?User)
      (assert (Transformer TripSignal_ON))
  )
```

B.2 Sample Code of *Knowledge_base* Agent

The following example shows a simple agent called *Knowledge_Based* agent designed according to FIPA specification. The agent is able to receive a message containing data from *Data_Sender*, connect to the JESS for the purpose of fault diagnosis.

```
/*
 * Knowledge_Based_Agent.java
 * Create on 5 June 2011, 10:56
 * author F. Davoodi Samirmi
 */
 import jade.core.Agent;
 import jade.core.AID;
 import jade.core.behaviours.CyclicBehaviour;
 import jade.lang.acl.ACLMessage;
 import jade.lang.acl.MessageTemplate;
 import java.io.FileReader;
 import java.io.IOException;
 import jess.*;
 public class Knowledge_Based extends Agent
   {protected void setup()
    {
      JessBehaviour jessBeh=new JessBehaviour(this, "JessFile.clp");
      addBehaviour(jessBeh);
      addBehaviour(new MsgListening(this, jessBeh));
      MyACLMessage myMsg = new MyACLMessage(ACLMessage.INFORM);
      myMsg.setContent("My content");
      myMsg.addReceiver(new AID("Data_Sender", false));
      send(myMsg);
    }

   public class MyACLMessage extends ACLMessage
{
      private String content;
      MyACLMessage(int perf) {
        super(perf);
     }

   public String getContent() {
      return content;
     }

   public void setContent(String content) {
      this.content = content;
     }
   }
  class JessBehaviour extends CyclicBehaviour {
  private jess.Rete jess;
  private static final int MAX_JESS_PASSES = 1;
  JessBehaviour(Agent agent, String jessFile) {
     super(agent);
     jess = new jess.Rete();
     jess.addUserfunction(new JessSend2(myAgent));
      try {
       FileReader fr = new FileReader("C:\\~\\jessFile.clp");
       jess.Jesp j = new jess.Jesp(fr, jess);
         try {
          j.parse(false);
           } catch (jess.JessException je) {
           je.printStackTrace();
           }
           fr.close();
        } catch (IOException ioe) {
        System.err.println("Error loading JessFile, engine is empty");
              }
             }
      public void action() {
        int executedPasses = -1;
        try {
          executedPasses = jess.run(MAX_JESS_PASSES);
              } catch (JessException je) {
          je.printStackTrace();
          }
       if(executedPasses < MAX_JESS_PASSES)
         block();
 }
     boolean addFact(String jessFact) {
     try {
     jess.reset();
     jess.assertString(jessFact);
          System.out.println(jessFact);
```

```
              } catch(JessException je) {
                  return false;
           }
     if(!isRunnable()) restart();
     return true;
     }
       boolean newMsg(ACLMessage msg) {
        String jf=msg.getContent();
        return addFact(jf);
   class MsgListening extends CyclicBehaviour {
     private JessBehaviour jessBeh;
      MsgListening(Agent agent, JessBehaviour jessBeh) {
       super(agent);
       this.jessBeh = jessBeh;
      }

public void action() {
MessageTemplate mt=MessageTemplate.MatchPerformative(ACLMessage.INFORM);
   ACLMessage msg=myAgent.receive(mt);
    if (msg != null) {
      if(jessBeh.newMsg(msg))
       System.out.println("New fact asserted. ");
        else
          System.out.println("Fact assertion failed. ");
      } else
     block();
    }
   }
 }
```

Appendix C

Ontology-based Reasoning

C.1 Sample Code of an *Ontology* Agent

According to FIPA specification, an *Ontology* agent is implemented. The life-cycle operation of this agent is initiated by creating the agent in the platform. The agent starts performing after receiving a message containing of gas ratios.

```
/*
 * OntologyAgent.java
 * Create on 3 December 2011, 17:10
 * author F. Davoodi Samirmi
 */

import java.io.File;
import java.util.Set;
import org.semanticweb.owlapi.apibinding.OWLManager;
import org.semanticweb.owlapi.model.IRI;
import org.semanticweb.owlapi.model.OWLClass;
import org.semanticweb.owlapi.model.OWLClassExpression;
import org.semanticweb.owlapi.model.OWLDataFactory;
import org.semanticweb.owlapi.model.OWLObjectProperty;
import org.semanticweb.owlapi.model.OWLOntology;
import org.semanticweb.owlapi.model.OWLOntologyCreationException;
import org.semanticweb.owlapi.model.OWLOntologyManager;

import org.semanticweb.owlapi.reasoner.ConsoleProgressMonitor;

import org.semanticweb.owlapi.reasoner.Node;

import org.semanticweb.owlapi.reasoner.NodeSet;

import org.semanticweb.owlapi.reasoner.OWLReasoner;

import org.semanticweb.owlapi.reasoner.OWLReasonerConfiguration;

import org.semanticweb.owlapi.reasoner.OWLReasonerFactory;

import org.semanticweb.owlapi.reasoner.SimpleConfiguration;

import org.semanticweb.owlapi.util.DefaultPrefixManager;

import uk.ac.manchester.cs.factplusplus.owlapiv3.

                                   FaCTPlusPlusReasonerFactory;

import jade.core.Agent;

import jade.core.behaviours.*;

import jade.lang.acl.*;


public class OntologyAgent extends Agent {

 protected void setup() {

  addBehaviour(new CyclicBehaviour(this) {

   public void action() {

    ACLMessage msg=receive();

    if (msg!=null) {

    System.out.println( " - " + myAgent.getLocalName() +

     " received Symptoms: " +  msg.getContent() );

  try {


  //Create our ontology manager in the usual way.

   OWLOntologyManager manager=OWLManager.createOWLOntologyManager();


  //Load a copy of the PowerTransformerOntology.

    File file=new File("C:/Ontology/~.owl");
```

```
//Now load the local copy
 OWLOntology ont=manager.loadOntologyFromOntologyDocument(file);
 System.out.println("The Ontology Loaded: " + ont.getOntologyID());
 DefaultPrefixManager pm=new DefaultPrefixManager("http://~.owl#");

//Create an instance of OWLReasoner. (Using factPP.OWL reasoner)
ReasonerFactory reasonerFactory=new FaCTPlusPlusReasonerFactory();

//We'll now create an instance of an OWLReasoner.
ConsoleProgressMonitor progressMonitor=new ConsoleProgressMonitor();

//Specify the progress monitor via a configuration.
 OWLReasonerConfiguration config=new SimpleConfiguration(
                                                 progressMonitor);

//Create a reasoner that will reason over ontology.
 OWLReasoner reasoner=reasonerFactory.createReasoner(ont, config);

 //Ask reasoner to determine the consistent ontology.
   reasoner.precomputeInferences();
   boolean consistent=reasoner.isConsistent();
   System.out.println("Consistent: " + consistent);
   System.out.println("\n");

 //We can easily get a list of unsatisfiable classes.
   Node<OWLClass> bottomNode=reasoner.getUnsatisfiableClasses();
   Set<OWLClass> unsatisfiable=bottomNode.getEntitiesMinusBottom();
   if (!unsatisfiable.isEmpty()) {
   System.out.println("The following classes are unsatisfiable: ");
   for(OWLClass cls : unsatisfiable) {
   System.out.println("    " + cls);
```

```
      }
     }
     else {
      System.out.println("There are no unsatisfiable classes.");
      }
     System.out.println("\n");

    //Now we want to query the reasoner for all types of faults.
     OWLDataFactory fac=manager.getOWLDataFactory();
     OWLClass koh=fac.getOWLClass(IRI.create(ont.getOntologyID().
      getOntologyIRI()+ "#" + msg.getContent()));
     OWLObjectProperty has_symp=fac.getOWLObjectProperty(IRI.create
     ("http://www.liv.co.uk/~owl#has_symptom"));
     OWLClassExpression has_symp_koh=fac.getOWLObjectSomeValuesFrom(
                                                   has_symp, koh);

   //Now ask reasoner for the has_symptom property values for koh
   NodeSet<OWLClass>faults=reasoner.getSubClasses(has_symp_koh,false);
     Set<OWLClass> faultsb=faults.getFlattened();
      System.out.println("faults: ");
       for(OWLClass cls : faultsb) {
        System.out.println("    " + cls);
        }
         for(OWLClass cls : faultsb) {
          if(!cls.isBottomEntity()) {
          System.out.println("Faults are   " + pm.getShortForm(cls));

         // Send message back
           ACLMessage reply=msg.createReply();
           reply.setPerformative( ACLMessage.INFORM );
           reply.setContent("faults are: " + pm.getShortForm(cls));
           send(reply);
            }
           }
           System.out.println("\n");
           }
           catch(UnsupportedOperationException exception) {
            System.out.println("Unsupported reasoner operation.");
           }
           catch (OWLOntologyCreationException e) {
            System.out.println("Could not load the ontology:
                " + e.getMessage());}
           }
           block();
         }
        });
    }
}
```

C.2 The Developed Ontology for Fault Diagnosis

Ontology for power transformer fault diagnosis is built in Protégé. Screenshots of three classes, *Faults*, *Symptoms* and *Components*, and their subclasses are given in the Figures C.1 – C.4. Figure C.5 presents the applied datatype properties for the case 8 of the Roger's method.

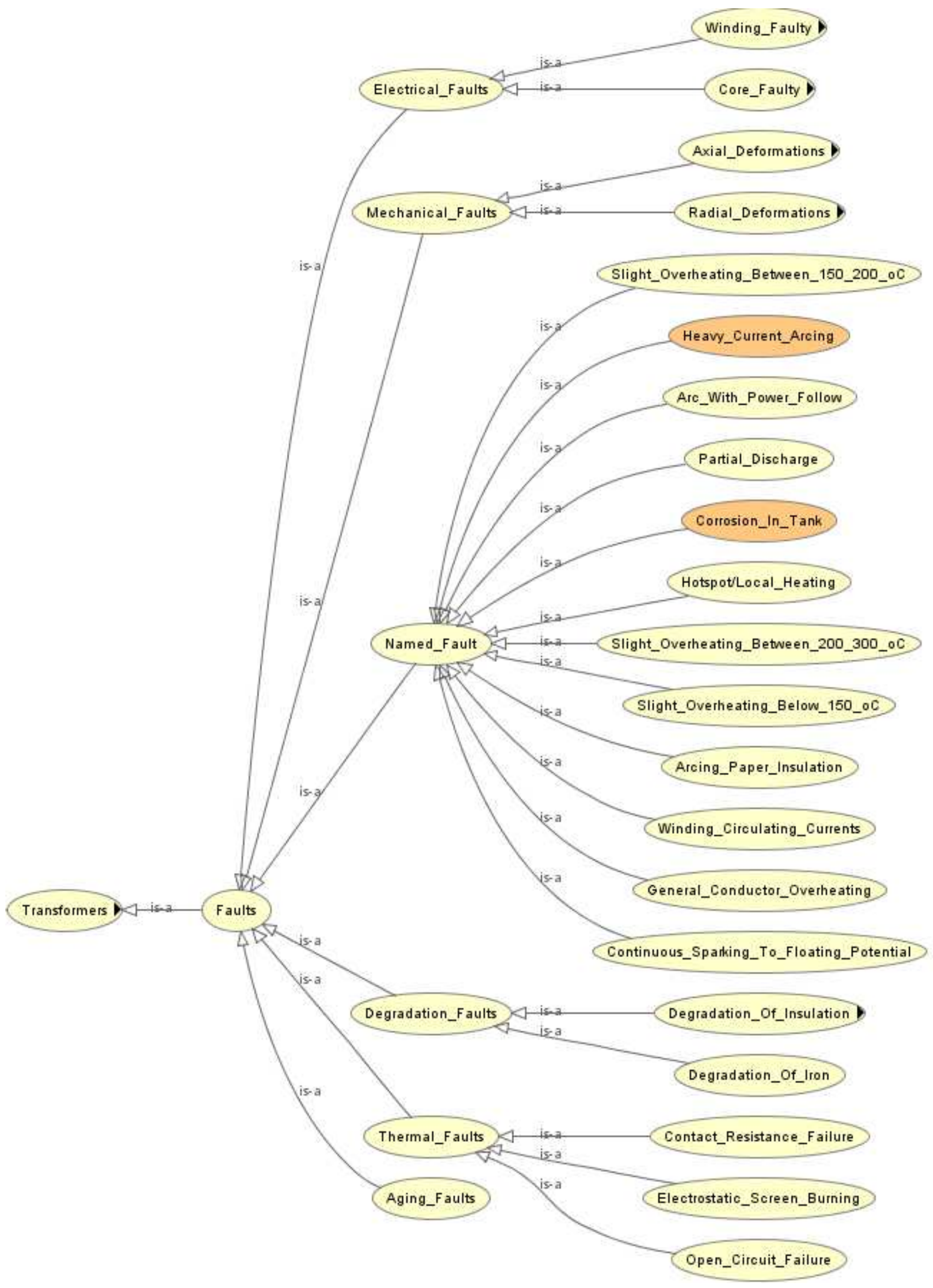


FIGURE C.1: Class *Faults* and its subclasses for developed ontology

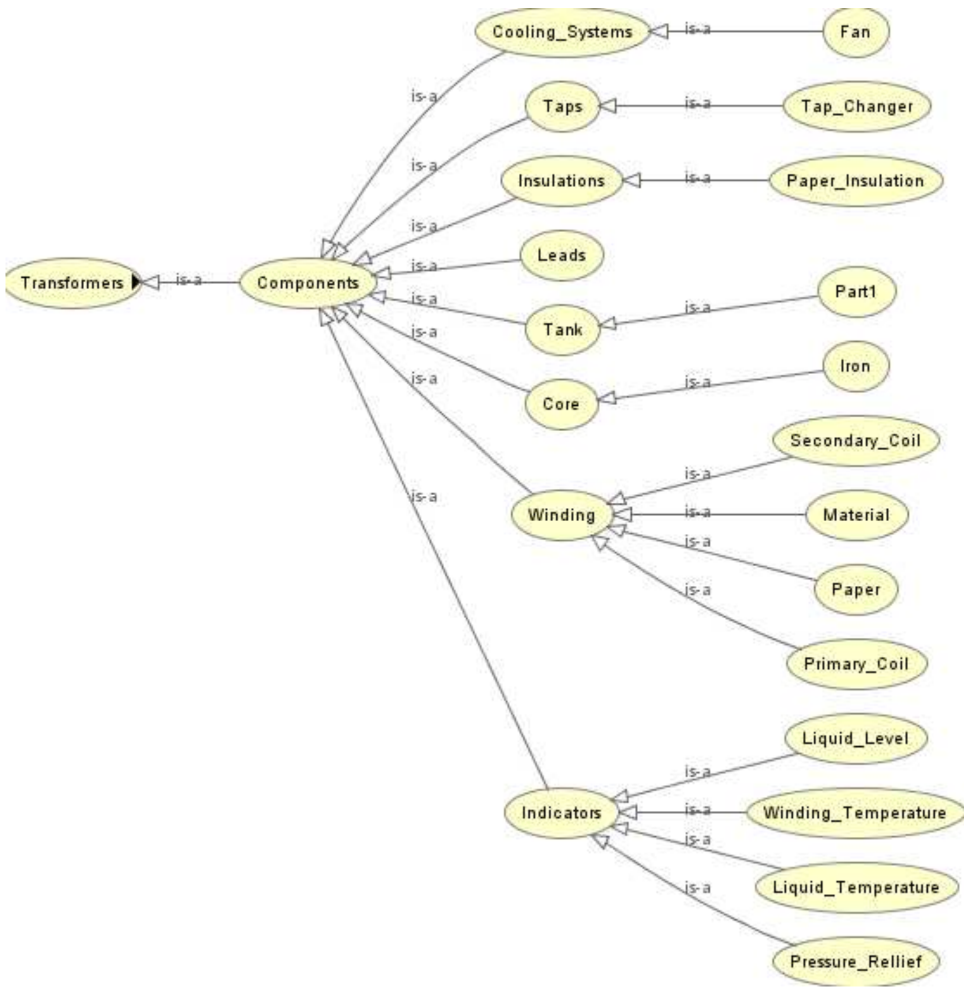


FIGURE C.2: Class *Components* and its subclasses for developed ontology

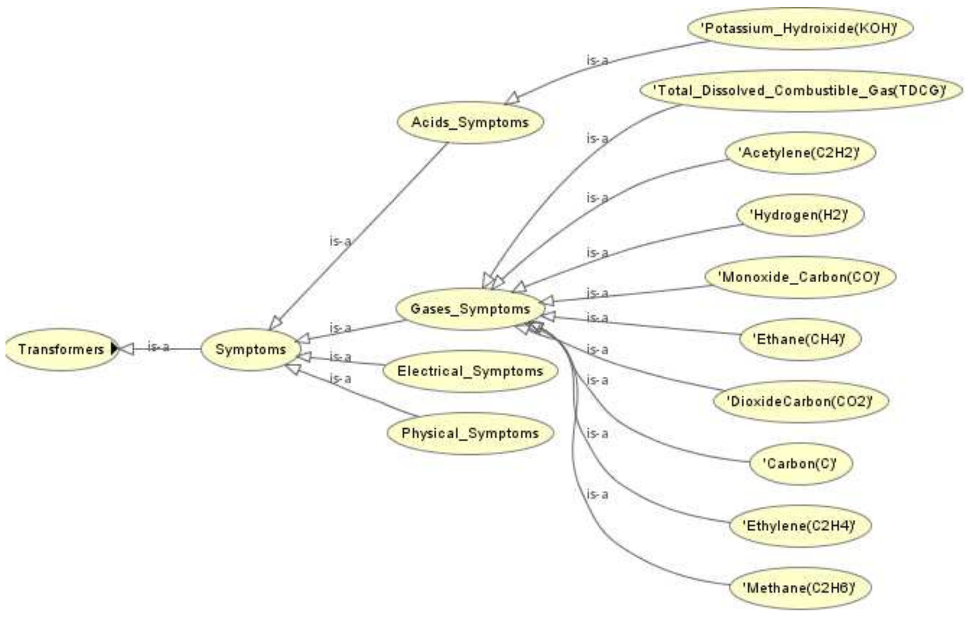


FIGURE C.3: Class *Symptoms* and its subclasses for developed ontology

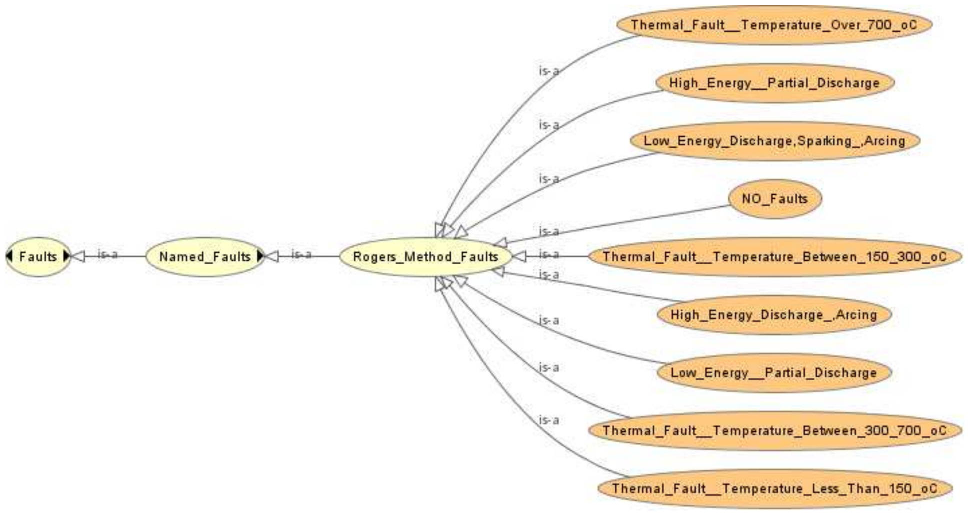
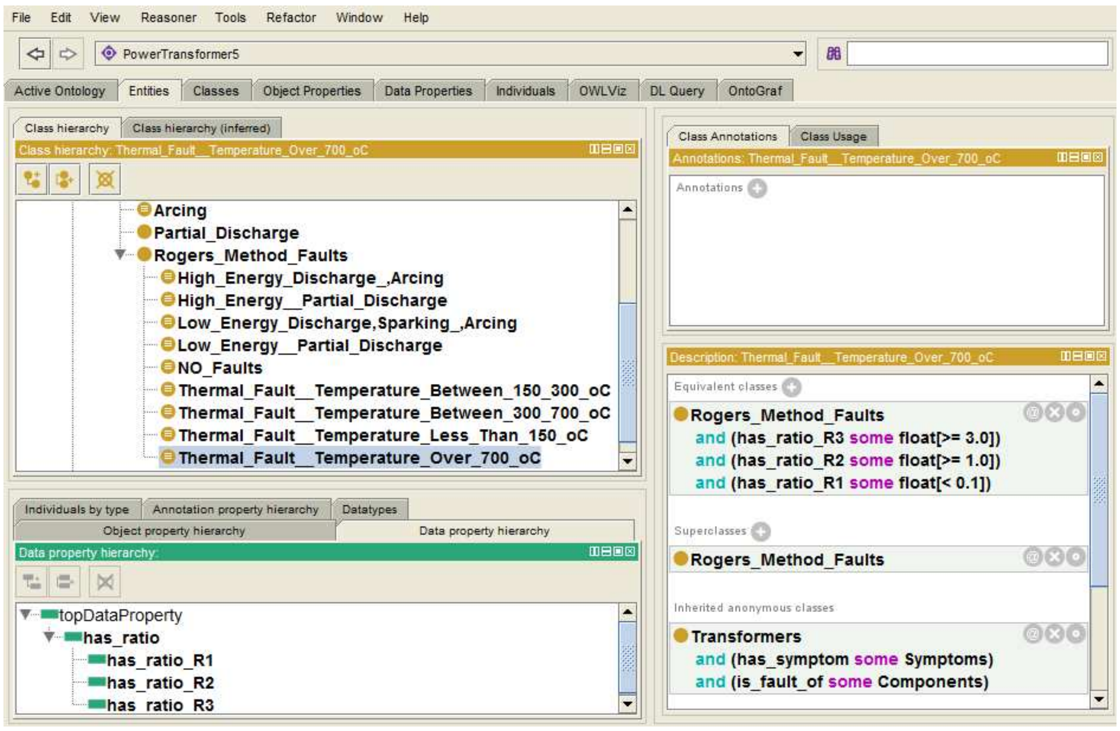

Faults
Named_Faults
Rogers_Method_Faults
Thermal_Fault__Temperature_Over_700_oC
High_Energy__Partial_Discharge
Low_Energy_Discharge,Sparking_,Arcing
NO_Faults
Thermal_Fault__Temperature_Between_150_300_oC
High_Energy_Discharge_,Arcing
Low_Energy__Partial_Discharge
Thermal_Fault__Temperature_Between_300_700_oC
Thermal_Fault__Temperature_Less_Than_150_oC
is-a

FIGURE C.4: Class *Rogers_Method_Faults* and its subclasses for developed ontology



Class hierarchy: Thermal_Fault__Temperature_Over_700_oC

```
Arcing
Partial_Discharge
Rogers_Method_Faults
    High_Energy_Discharge_,Arcing
    High_Energy__Partial_Discharge
    Low_Energy_Discharge,Sparking_,Arcing
    Low_Energy__Partial_Discharge
    NO_Faults
    Thermal_Fault__Temperature_Between_150_300_oC
    Thermal_Fault__Temperature_Between_300_700_oC
    Thermal_Fault__Temperature_Less_Than_150_oC
    Thermal_Fault__Temperature_Over_700_oC
```

Data property hierarchy:

```
topDataProperty
  has_ratio
    has_ratio_R1
    has_ratio_R2
    has_ratio_R3
```

Annotations: Thermal_Fault__Temperature_Over_700_oC

Annotations

Description: Thermal_Fault__Temperature_Over_700_oC

Equivalent classes

```
Rogers_Method_Faults
  and (has_ratio_R3 some float[>= 3.0])
  and (has_ratio_R2 some float[>= 1.0])
  and (has_ratio_R1 some float[< 0.1])
```

Superclasses

```
Rogers_Method_Faults
```

Inherited anonymous classes

```
Transformers
  and (has_symptom some Symptoms)
  and (is_fault_of some Components)
```

FIGURE C.5: Datatype properties applied for Roger's method fault diagnosis

C.3 The OWL Ontology Annotation

The developed ontology for power transformer fault diagnosis built in Protégé 4.1, contains hierarchy classes and subclasses restricted by various types of properties. An example of applied datatype in OWL2 syntax for the case 0 of the Roger's method is given below:

```
<EquivalentClasses>
 <Class IRI="http://www.liv.co.uk/~.owl#No_Fault"/>
  <ObjectIntersectionOf>
   <DataSomeValuesFrom>
    <DataProperty IRI="http://www.liv.co.uk/~.owl#has_ratio_R2"/>
     <DatatypeRestriction>
      <Datatype abbreviatedIRI="xsd:float"/>
       <FacetRestriction facet="&xsd;maxInclusive">
        <Literal datatypeIRI="&xsd;double">0.1</Literal>
        </FacetRestriction>
       </DatatypeRestriction>
      </DataSomeValuesFrom>
     <DataSomeValuesFrom>
    <DataProperty IRI="http://www.liv.co.uk/~.owl#has_ratio_R1"/>
     <DatatypeRestriction>
       <Datatype abbreviatedIRI="xsd:float"/>
        <FacetRestriction facet="&xsd;minInclusive">
          <Literal datatypeIRI="&xsd;double">0.1</Literal>
        </FacetRestriction>
       <FacetRestriction facet="&xsd;maxInclusive">
          <Literal datatypeIRI="&xsd;double">1.0</Literal>
        </FacetRestriction>
       </DatatypeRestriction>
      </DataSomeValuesFrom>
     <DataSomeValuesFrom>
    <DataProperty IRI="http://www.liv.co.uk/~.owl#has_ratio_R5"/>
     <DatatypeRestriction>
        <Datatype abbreviatedIRI="xsd:float"/>
         <FacetRestriction facet="&xsd;maxExclusive">
          <Literal datatypeIRI="&xsd;double">1.0</Literal>
        </FacetRestriction>
      </DatatypeRestriction>
    </DataSomeValuesFrom>
  </ObjectIntersectionOf>
</EquivalentClasses>
```

Appendix D

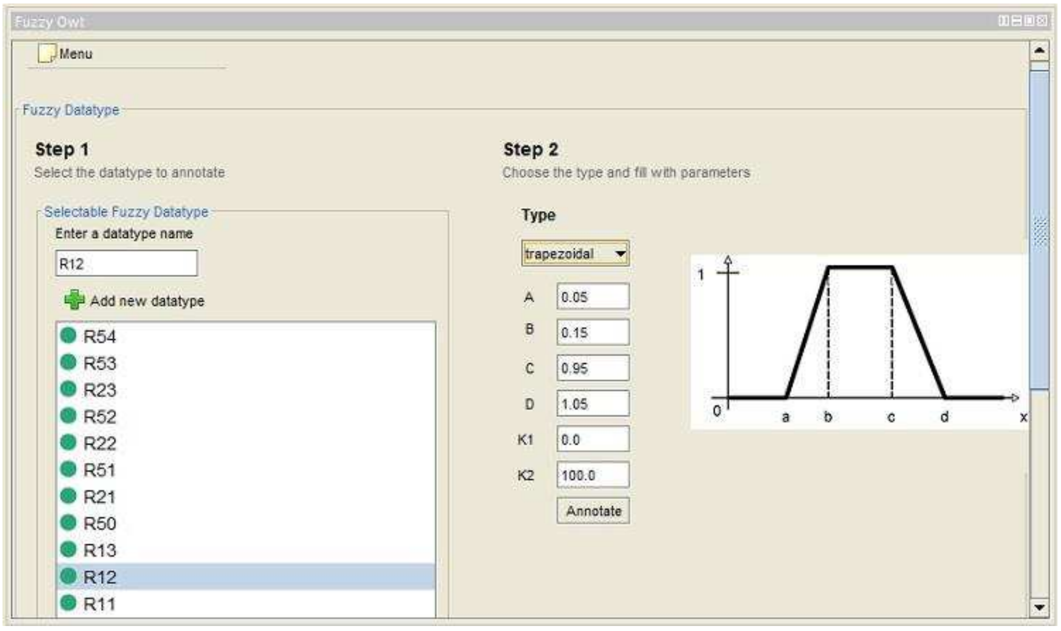
Fuzzy Ontology

D.1 Fuzzy Ontology-based Knowledge Representation

The OWL2 syntax expresses the case 0 of the Roger's method; it can be written as follows:

```
      <EquivalentClasses>
        <Class IRI="http:/~FuzzyOntology1.owl#No_Fault"/>
        <ObjectIntersectionOf>
            <DataSomeValuesFrom>
                <DataProperty IRI="http:/~FuzzyOntology1.owl#has_ratio"/>
                <Datatype IRI="#R21"/>
            </DataSomeValuesFrom>
            <DataSomeValuesFrom>
                <DataProperty IRI="http:/~FuzzyOntology1.owl#has_ratio"/>
                <Datatype IRI="#R12"/>
            </DataSomeValuesFrom>
            <DataSomeValuesFrom>
                <DataProperty IRI="http:/~FuzzyOntology1.owl#has_ratio"/>
                <Datatype IRI="#R51"/>
            </DataSomeValuesFrom>
        </ObjectIntersectionOf>
    </EquivalentClasses>
```

Figures D.1 to D.4 show the screenshots of the developed fuzzy ontology for power transformer fault diagnosis, based on Roger's method.



Step 1
Select the datatype to annotate

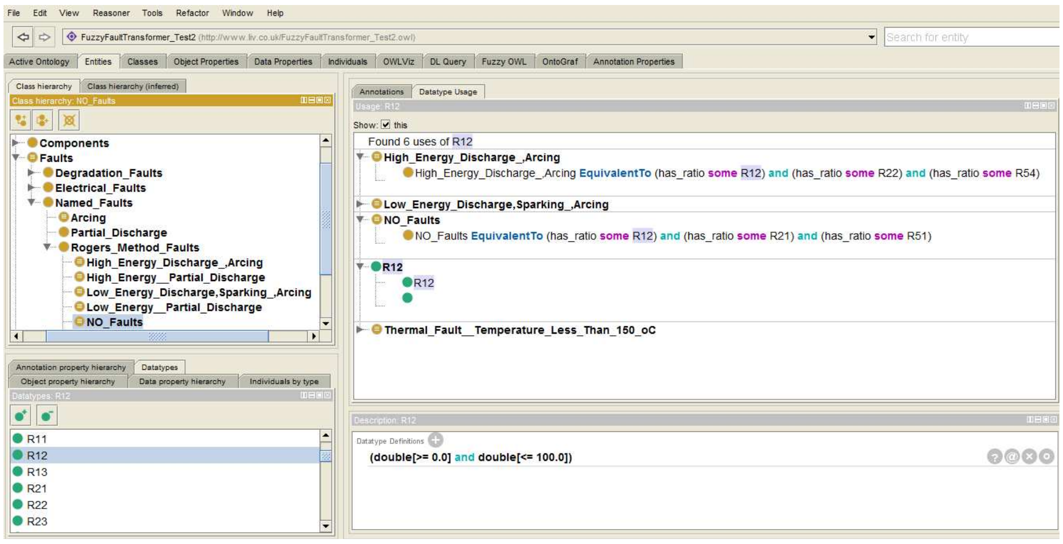
Enter a datatype name: R12

R54
R53
R23
R52
R22
R51
R21
R50
R13
R12
R11

Step 2
Choose the type and fill with parameters

Type	trapezoidal
A	0.05
B	0.15
C	0.95
D	1.05
K1	0.0
K2	100.0

FIGURE D.1: Fuzzy modified datatype representing the R12 condition



Components
Faults
Degradation_Faults
Electrical_Faults
Named_Faults
Arcing
Partial_Discharge
Rogers_Method_Faults
High_Energy_Discharge_,Arcing
High_Energy__Partial_Discharge
Low_Energy_Discharge,Sparking_,Arcing
Low_Energy__Partial_Discharge
NO_Faults

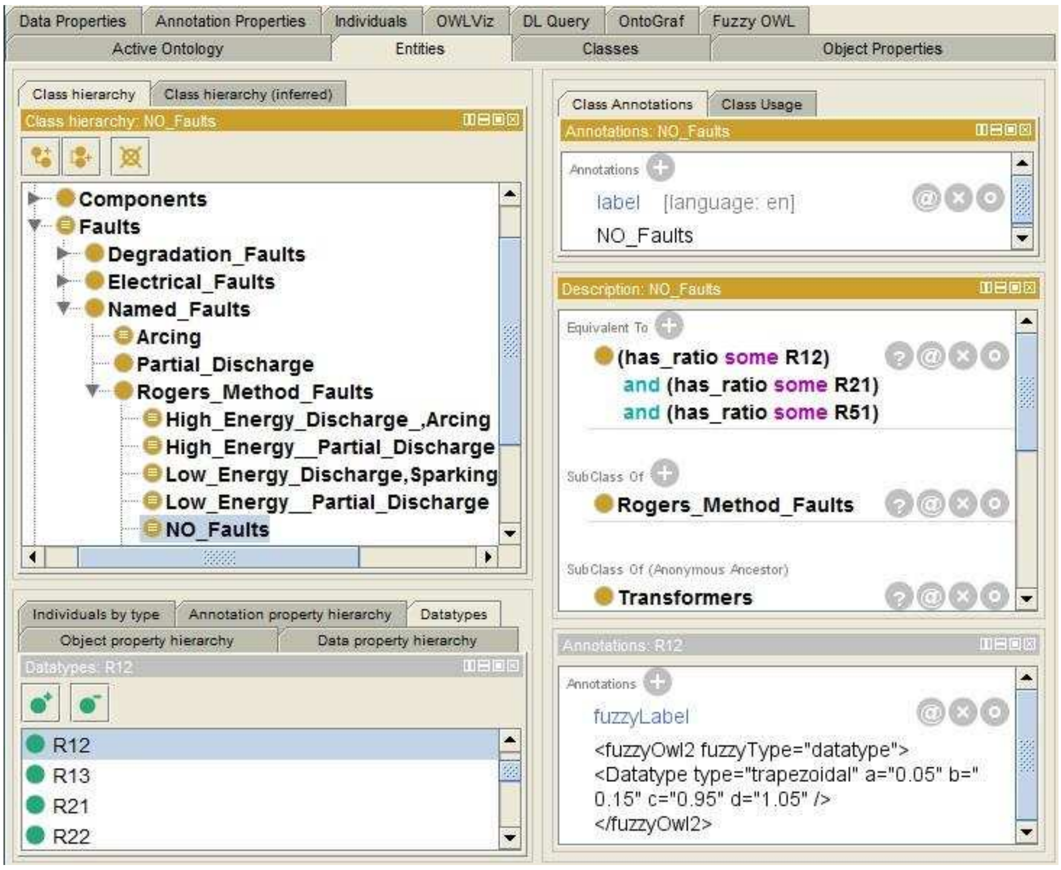
Found 6 uses of R12

High_Energy_Discharge_,Arcing
High_Energy_Discharge_,Arcing EquivalentTo (has_ratio some R12) and (has_ratio some R22) and (has_ratio some R54)
Low_Energy_Discharge,Sparking_,Arcing
NO_Faults
NO_Faults EquivalentTo (has_ratio some R12) and (has_ratio some R21) and (has_ratio some R51)
R12
R12
Thermal_Fault__Temperature_Less_Than_150_oC

R11
R12
R13
R21
R22
R23

(double[>= 0.0] and double[<= 100.0])

FIGURE D.2: The fuzzy datatype R12 applied to the classes in Protégé



(has_ratio some R12)
and (has_ratio some R21)
and (has_ratio some R51)

```
<fuzzyOwl2 fuzzyType="datatype">
<Datatype type="trapezoidal" a="0.05" b="
0.15" c="0.95" d="1.05" />
</fuzzyOwl2>
```

FIGURE D.3: Protégé plug-in classes and properties

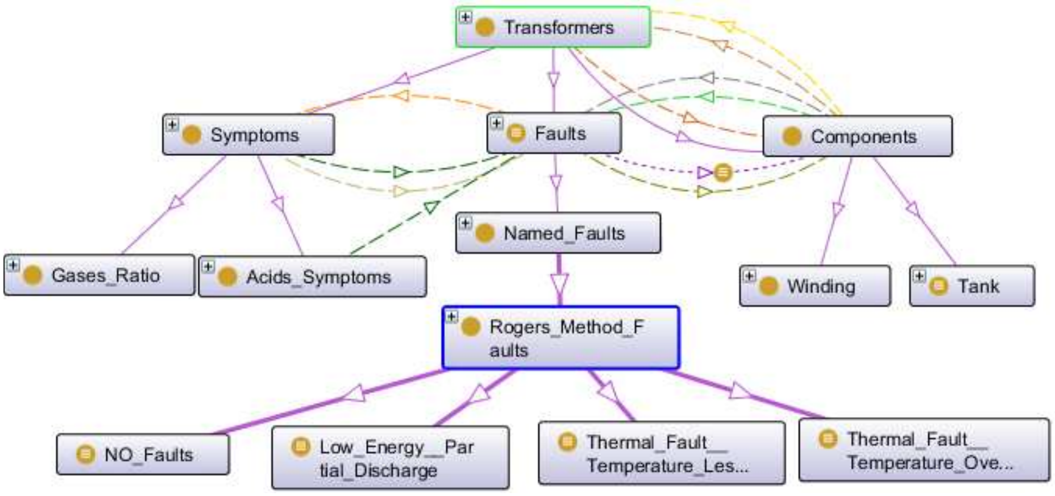


FIGURE D.4: The fuzzy ontology designed in Protégé

References

[1] D. P. Buse and Q. H. Wu, *IP Network-Based Multi-Agent Systems for Industrial Automation*. Springer, 2006.

[2] M. Wooldridge and N. R. Jennings, “Intelligent agents: Theory and practice,” *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, 1995.

[3] Z. Yang, C. Ma, J. Q. Feng, Q. H. Wu, S. Mann, and J. Fitch, “A multi-agent framework for power system automation,” *International Journal of Innovations in Energy Systems and Power*, vol. 1, 2006.

[4] S. D. J. McArthur, E. M. Davidson, V. M. Catterson, A. L. Dimeas, N. D. Hatziargyriou, F. Ponci, and T. Funabashi, “Multi-agent systems for power engineering applicationspart I concepts, approaches, and technical challenges,” *Power Systems, IEEE Transactions*, vol. 22, no. 4, pp. 1743–1752, 2007.

[5] S. D. J. McArthur, E. M. Davidson, V. M. Catterson, A. L. Dimeas, N. D. Hatziargyriou, F. Ponci, and T. Funabashi, “Multi-agent systems for power engineering applications part II technologies, standards, and tools for building multi-agent systems,” *Power Systems, IEEE Transactions*, vol. 22, no. 4, pp. 1753–1759, 2007.

[6] J. Q. Feng, J. S. Smith, Q. H. Wu, and J. Fitch, “Condition assessment of power system apparatuses using ontology systems,” in *Transmission and Distribution Conference and Exhibition: Asia and Pacific, 2005 IEEE/PES*, pp. 1–6, 2005.

[7] A. Purwadi, N. Heryana, D. Nurafiat, C. Sosetyo, A. Setiana, and A. Mustaqim, “Testing and diagnostics of power transformer in pt. Indonesia power kamojang geothermal power plant unit 1,” in *Electrical Engineering and Informatics (ICEEI), 2011 International Conference*, pp. 1–5, 2011.

[8] N. R. Jennings, E. H. Mamdani, J. M. Corera, I. Laresgoiti, F. Perriolat, P. Skarek, and L. Z. Varga, “Using Archon to develop real-world DAI applications,” *IEEE Expert*, vol. 11, pp. 64–70, 1996.

[9] D. Wang, W. H. Tang, and Q. H. Wu, “Ontology-based fault diagnosis for power transformers,” in *Power and Energy Society General Meeting, 2010 IEEE*, pp. 1–8, 2010.

[10] S. D. J. McArthur, S. M. Strachan, and G. Jahn, “The design of a multi-agent transformer condition monitoring system,” *Power Systems, IEEE Transactions*, vol. 19, no. 4, pp. 1845–1852, 2004.

[11] J. Q. Feng, D. P. Buse, Q. H. Wu, and J. Fitch, “A multi-agent based intelligent monitoring system for power transformers in distributed substations,” in *Power System Technology, 2002. Proceedings. PowerCon 2002. International Conference*, vol. 3, pp. 1962–1965 vol.3, 2002.

[12] A. Akbari, A. Setayeshmehr, H. Borsi, E. Gockenbach, and I. Fofana, “Intelligent agent-based system using dissolved gas analysis to detect incipient faults in power transformers,” *Electrical Insulation Magazine, IEEE*, vol. 26, no. 6, pp. 27–40, 2010.

[13] J. D. McDonald, “Substation automation. IED integration and availability of information,” *Power and Energy Magazine, IEEE*, vol. 1, no. 2, pp. 22–31, 2003.

[14] J. cheng Tan and W. Luan, “IEC 61850 based substation automation system architecture design,” in *Power and Energy Society General Meeting, 2011 IEEE*, pp. 1–6, 2011.

[15] G. Booch, *Object-oriented analysis and design with applications.* Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc., 1994.

[16] *Object Management Group. OMG home page. Available at http://www.omg.org.*

[17] *Agentcities. Agentcities home page. Available at http://www.agentcities.org.*

[18] *Foundation for Intelligent Physical Agents. FIPA home page. Available at http://www.fipa.org.*

[19] B. Chaib-draa and F. Dignum, “Trends in agent communication language,” *Computational Intelligence*, pp. 89–101, May 2002.

[20] F. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*. Wiley Series in Agent Technology, 2004.

[21] M. d. Michael Luck, Ronald Ashri, *Agent-Based Software Development*. British Library Cataloguing in Publication Data, 2004.

[22] M. Wooldridge, *An Introduction to MultiAgent Systems - Second Edition*. John Wiley & Sons, May 2009.

[23] I. A. Ferguson, *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents (1992)*. PhD thesis, University of Cambridge, 1992.

[24] F. Toulouse, *Information Technology: Selected Tutorials*. Springer, 2012.

[25] A. Sturm and O. Shehory, “A framework for evaluating agent-oriented methodologies,” in *Proc. of the Int. Bi-Conference Workshop on Agent-Oriented Information Systems, AOIS 2003, volume 3030 of LNCS*, pp. 94–109, Springer, 2003.

[26] M. Elammari and W. Lalonde, “An agent-oriented methodology: High-level and intermediate models,” in *Proc. of the 1st Int. Workshop. on Agent-Oriented Information Systems*, 1999.

[27] S. A. Deloach, “Analysis and design using mase and agentTool,” in *In 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS) 2001*, p. 2001.

[28] M. Wooldridge, N. R. Jennings, and D. Kinny, “The gaia methodology for agent-oriented analysis and design,” *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 3, pp. 285–312, 2000.

[29] W. Tsaur and H.-C. Tsai, “Secure electronic business applications in mobile agent based networks using elliptic curve cryptosystems,” in *Computer Symposium (ICS), 2010 International*, pp. 204–209, 2010.

[30] D. P. Buse, P. Sun, Q. H. Wu, and J. Fitch, “Agent-based substation automation,” *IEEE power and energy magazine*, vol. 1540-7977, 2003.

[31] K. Manickavasagam, M. Nithya, K. Priya, J. Shruthi, S. Krishnan, S. Misra, and S. Manikandan, "Control of distributed generator and smart grid using multi-agent system," in *Electrical Energy Systems (ICEES), 2011 1st International Conference*, pp. 212–217, 2011.

[32] A. L. Dimeas and N. D. Hatziargyriou, "Operation of a multiagent system for microgrid control," *Power Systems, IEEE Transactions*, vol. 20, no. 3, pp. 1447–1455, 2005.

[33] S. D. J. McArthur, E. M. Davidson, and V. M. Catterson, "Building multi-agent systems for power engineering applications," in *Power Engineering Society General Meeting, 2006. IEEE*, p. 7, 2006.

[34] V. K. Singh and A. K. Gupta, "Agent based models of social systems and collective intelligence," in *Intelligent Agent Multi-Agent Systems, 2009. IAMA 2009. International Conference*, pp. 1–7, 2009.

[35] N. Avouris and L. Gasser, *ARCHON: Theory and Practice.* Kluwer Academic Press, 1992.

[36] J. A. Hossack, J. Menal, S. D. J. McArthur, and J. R. McDonald, "A multi-agent architecture for protection engineering diagnostic assistance," *Power Systems, IEEE Transactions*, vol. 18, no. 2, pp. 639–647, 2003.

[37] S. D. McArthur, C. D. Booth, J. R. McDonald, and I. T. McFadyen, "An agent-based anomaly detection architecture for condition monitoring," *Power Systems, IEEE Transactions*, vol. 20, no. 4, pp. 1675–1682, 2005.

[38] D. P. Buse and Q. H. Wu, "Mobile agents for remote control of distributed systems," *Industrial Electronics, IEEE Transactions*, vol. 51, no. 6, pp. 1142–1149, 2004.

[39] F. F. Wu, K. Moslehi, and A. Bose, "Power system control centers: Past, present, and future," *Proceedings of the IEEE*, vol. 93, no. 11, pp. 1890–1908, 2005.

[40] *Numbers of substations in the UK, home page. Available at http://www.emfs.info/Sources+of+EMFs/Substations/substationnumbers.*

[41] L. L. Grigsby, *The Electric Power Engineering Handbook.* Taylor & Francis Group, 2007.

[42] G. Diaz, A. Barbon, and J. Gomez-Aleixandre, “Mathematical model of a distribution transformer as applied to turn-to-turn faults analysis,” in *EuroPES*, pp. 557–562, June 2002.

[43] P. Stakhiv, O. Hoholyuk, L. Byczkowska-Lipi, and N. Ska, “Mathematical models and macro models of electric power transformers,” *Przeglad Dermatologicznyd Elektrotechniczny*, vol. ISSN 0033-2097, R. 87 NR, pp. 163–165, 2011, May.

[44] P. Kang, D. Birtwhistle, J. Daley, and D. McCulloch, “Non-invasive on-line condition monitoring of on load tap changers,” in *Power Engineering Society Winter Meeting, IEEE*, vol. 3, pp. 2223–2228, 2000.

[45] M. Foata, R. Beauchemin, and C. Rajotte, “On-line testing of on-load tap changers with a portable acoustic system,” in *Transmission and Distribution Construction, Operation and Live-Line Maintenance Proceedings. 2000 IEEE ESMO - 2000 IEEE 9th International Conference*, pp. 293–298, 2000.

[46] T. Leibfried, “Online monitors keep transformers in service,” *Computer Applications in Power, IEEE*, vol. 11, pp. 36–42, Jul 1998.

[47] S. Tenbohlen, T. Stirl, G. Bastos, J. Baldauf, P. Mayer, M. Stach, B. Breitenbauch, and R. Huber, “Experienced based evaluation of economic benefits of online monitoring systems for power transformers,” in *21, rue dArtois, F-75008 Paris, Session 2002, CIGRE, www.cigre.org.*

[48] D. Chu and A. Lux, “On-line monitoring of power transformers and components: a review of key parameters,” in *Electrical Insulation Conference and Electrical Manufacturing amp; Coil Winding Conference*, 1999.

[49] “IEEE guide for application for monitoring equipment to liquid-immersed transformers and components,” *IEEE Std C57.143-2012*, pp. 1–83, 2012.

[50] G. J. Pukel, H. M. Muhr, and W. Lick, “Transformer diagnostics: Common used and new methods,” in *International Conference on condition Monitoring and Diagnosis*, 2006.

[51] Y. Zhang, X. Ding, Y. Liu, and P. J. Griffin, “An artificial neural network approach to transformer fault diagnosis,” *Power Delivery, IEEE Transactions*, vol. 11, pp. 1836–1841, Oct 1996.

[52] S. Mofizul Islam, T. Wu, and G. Ledwich, “A novel fuzzy logic approach to transformer fault diagnosis,” *Dielectrics and Electrical Insulation, IEEE Transactions*, vol. 7, pp. 177–186, Apr 2000.

[53] H. B. Zheng, R. J. Liao, S. Grzybowski, and L. J. Yang, “Fault diagnosis of power transformers using multi-class least square support vector machines classifiers with particle swarm optimisation,” *IET Electric Power Applications, Issue 9*, vol. 5, pp. 691–696, 2011.

[54] M. Duval, “A review of faults detectable by gas-in-oil analysis in transformers,” *IEEE in Electrical Insulation Magazine*, vol. 18, pp. 8–17, May-June 2002.

[55] “IEEE guide for the interpretation of gases generated in oil-immersed transformers,” *IEEE Std C57.104-2008 (Revision of IEEE Std C57.104-1991)*, pp. C1–27, 2009.

[56] R. R. Rogers, “IEEE and IEC codes to interpret incipient faults in transformers, using gas in oil analysis,” *Electrical Insulation, IEEE Transactions*, vol. EI-13, no. 5, pp. 349–354, 1978.

[57] M. Duval and A. dePabla, “Interpretation of gas-in-oil analysis using new IEC publication 60599 and IEC TC 10 databases,” *Electrical Insulation Magazine, IEEE*, vol. 17, pp. 31–41, March-April 2001.

[58] Q. H. Wu, D. P. Buse, J. Q. Feng, and P. Sun, “E-automation, an architecture for distributed industrial automation systems,” *International Journal of Automation and Computing*, pp. 17–25, 2004.

[59] F. Zambonelli, N. R. Jennings, and M. Wooldridge, "Developing multiagent systems: The Gaia methodology," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 12 Issue 3, pp. 317–370, July 2003.

[60] N. R. Jennings, P. Faratin, M. J. Johnson, T. J. Norman, P. OBrien, and M. E. Wiegand, "Agent-based business process management," *International Journal of Cooperative Information Systems*, vol. 5(2-3), pp. 105–130, 1996.

[61] J. Q. Feng, C. Ma, W. H. Tang, J. S. Smith, and Q. H. Wu, "A transformer predictive maintenance system based on agent-oriented programming," in *Transmission and Distribution Conference and Exhibition: Asia and Pacific, 2005 IEEE/PES*, pp. 1–6, 2005.

[62] P. Moraitis, "Combining Gaia and JADE for multi-agent systems development," in *In Proceedings of the 17th European Meeting on Cybernetics and Systems Research (EMCSR) 2004*, 2004.

[63] P. Moraïtis, E. Petraki, and N. I. Spanoudakis, "Engineering JADE agents with the Gaia methodology," in *Conference on Agent technologies, infrastructures, tools, and applications for E-services*, NODe'02, (Berlin, Heidelberg), pp. 77–91, Springer-Verlag, 2003.

[64] R. Akerkar and P. Sajja, *Knowledge-Based Systems.* USA: Jones and Bartlett Publishers, Inc., 1st ed., 2009.

[65] *Java Expert System Shell (JESS) home page. Available at http://www.jessrules.com.*

[66] *Protege home page. Available at http://protege.stanford.edu.*

[67] *Fuzzy ontology plug-in home page. Available at http://gaia.isti.cnr.it/straccia /-software /FuzzyOWL/index.html.*

[68] P. Chen and D. Suter, "Recovering the missing components in a large noisy low-rank matrix: application to SFM," *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol. 26, pp. 1051–1063, Aug. 2004.

[69] P. Vateekul and K. Sarinnapakorn, “Tree-based approach to missing data imputation,” in *Data Mining Workshops, 2009. ICDMW ’09. IEEE International Conference*, pp. 70–75, Dec. 2009.

[70] A. Farhangfar, L. A. Kurgan, and W. Pedrycz, “A novel framework for imputation of missing values in databases,” *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions*, vol. 37, pp. 692–709, Sept. 2007.

[71] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, 4th Edition.* Academic Press, October 20, 2008.

[72] N. Gao, G. J. Zhang, Z. Qian, Z. Yan, and D. H. Zhu, “Diagnosis of DGA based on fuzzy and ANN methods,” in *Electrical Insulating Materials, 1998. Proceedings of 1998 International Symposium*, pp. 767–770, Sep 1998.

[73] D. P. Buse, *Information Management, Condition Monitoring and Control of Power Systems Over Internet Protocol Networks.* PhD thesis, The University of Liverpool, Department of Electrical Engineering and Electronics, 2003.

[74] *JFreeChart. Agentcities home page. Available at http://www.jfree.org/jfreechart.*

[75] “Introduction to knowledge-based systems,” in *Electronic Technology Directions to the Year 2000, 1995. Proceedings.*, pp. 18–27, 1995.

[76] J. R. McDonald, G. M. Burt, and D. J. Young, “Alarm processing and fault diagnosis using knowledge based systems for transmission and distribution network control,” *Power Systems, IEEE Transactions*, vol. 7, no. 3, pp. 1292–1298, 1992.

[77] D. B. Tesch, D. C. Yu, L. M. Fu, and K. Vairavan, “A knowledge-based alarm processor for an energy management system,” *Power Systems, IEEE Transactions*, vol. 5, no. 1, pp. 268–275, 1990.

[78] A. B. Marques, G. N. Taranto, and D. M. Falcao, “A knowledge-based system for supervision and control of regional voltage profile and security,” *Power Systems, IEEE Transactions*, vol. 20, no. 1, pp. 400–407, 2005.

[79] E. M. Davidson, S. D. J. McArthur, J. R. McDonald, T. Cumming, and I. Watt, “Applying multi-agent system technology in practice: automated management

and analysis of SCADA and digital fault recorder data," *Power Systems, IEEE Transactions*, vol. 21, no. 2, pp. 559–567, 2006.

[80] C. L. Forgy, "Rete: A fast algorithm for the many pattern/many object pattern match problem," *Artificial Intelligence*, vol. 19, no. 1, pp. 17–37, 1982.

[81] "IEC 60354 oil immersed transformers."

[82] "IEC 60076-11 dry type transformers."

[83] S. D. J. McArthur and E. M. Davidson, "Automated post-fault diagnosis of power system disturbances," in *Power Engineering Society General Meeting, 2006. IEEE*, p. 6, 2006.

[84] S. D. J. McArthur, E. M. Davidson, J. A. Hossack, and J. R. McDonald, "Automating power system fault diagnosis through multi-agent system technology," in *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference*, p. 8, 2004.

[85] N. Gao, G. J. Zhang, Z. Qian, Z. Yan, and D. H. Zhu, "Diagnosis of DGA based on fuzzy and ANN methods," in *Electrical Insulating Materials, 1998. Proceedings of 1998 International Symposium*, pp. 767–770, 1998.

[86] D. Siva Sarma and G. N. S. Kalyani, "ANN approach for condition monitoring of power transformers using DGA," in *TENCON 2004. 2004 IEEE Region 10 Conference*, vol. C, pp. 444–447 Vol. 3, 2004.

[87] S. DRAN and P. VARTHINI, "Detecting power systems failure based on fuzzy rule in power grid," *Przeglad Dermatologicznyd Elektrotechniczny*, pp. 172–177, 2013. SSN 0033-2097, R. 89 NR.

[88] N. K. Dhote and J. B. Helonde, "Fuzzy algorithm for power transformer diagnostics," *Adv. Fuzzy Sys.*, pp. 1–4, Jan 2013.

[89] Z. Fan and M. Huang, "Fuzzy rule set based engine fault diagnosis," in *Power and Energy Engineering Conference, APPEEC Asia-Pacific*, pp. 1–5, 2009.

[90] C. E. Lin, J.-M. Ling, and C. Huang, "An expert system for transformer fault diagnosis using dissolved gas analysis," *Power Delivery, IEEE Transactions*, vol. 8, no. 1, pp. 231–238, 1993.

[91] P. Doran, *Ontology modularization : principles and practice.* PhD thesis, The University of Liverpool , Department of Computer Science, 2009.

[92] F. van Harmelen, F. van Harmelen, V. Lifschitz, and B. Porter, *Handbook of Knowledge Representation.* San Diego, USA: Elsevier Science, 2007.

[93] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199 – 220, 1993.

[94] N. F. Noy and D. L. McGuinness, "Ontology development 101: A guide to creating your first ontology." Online, 2001.

[95] U. S. Franz Baader, Ian Horrocks, *Handbook on Ontologies.* International Handbooks on Information Systems, 2004.

[96] *Ontology Web Language (OWL) OWL home page. Available at http://www.w3.org.*

[97] V. Haarslev and R. Möller, "RACER system description," in *Proceedings of the First International Joint Conference on Automated Reasoning*, IJCAR '01, (London, UK), pp. 701–706, Springer-Verlag, 2001.

[98] I. Horrocks, "The FaCT system," in *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, TABLEAUX '98, (London, UK), pp. 307–312, Springer-Verlag, 1998.

[99] D. Tsarkov and I. Horrocks, "FaCT++ description logic reasoner: System description," in *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, vol. 4130 of *Lecture Notes in Artificial Intelligence*, pp. 292–297, Springer, 2006.

[100] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical OWL-DL reasoner," *Web Semant.*, vol. 5, pp. 51–53, Jun 2007.

[101] A.-Y. Turhan, "Description logic reasoning for semantic web ontologies," in *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*, WIMS '11, (New York, NY, USA), pp. 6:1–6:5, ACM, 2011.

[102] *OWL Application Programming Interface (API) home page. Available at http://owlapi.sourceforge.net.*

[103] M. Horridge, H. Knublauch, A. Rector, R. Stevens, and C. Wroe, *A Practical Guide To Building OWL Ontologies Using The Protege-OWL Plugin and CO-ODE Tools Edition 1.0*, August 2004.

[104] United States Department of Commerce National Technical Information Service, *Facilities instructions, standards and techniques (FIST) 3-30 transformer maintanance*, October 2000.

[105] L. A. Zadeh, *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A. Zadeh.* River Edge, NJ, USA: World Scientific Publishing Co., Inc., 1996.

[106] H. Ghorbel, A. Bahri, and R. Bouaziz, "Fuzzy Protégé for Fuzzy Ontology Models," tech. rep., Protégé, 2009.

[107] B. Bede, *Mathematics of Fuzzy Sets and Fuzzy Logic.* Springer-Verlag Berlin Heidelberg, 2013.

[108] Y.-C. Huang and H.-C. Sun, "Dissolved gas analysis of mineral oil for power transformer fault diagnosis using fuzzy logic," *Dielectrics and Electrical Insulation, IEEE Transactions*, vol. 20, no. 3, 2013.

[109] H.-C. Sun, Y.-C. Huang, and C.-M. Huang, "Fault diagnosis of power transformers using computational intelligence: A review," *Energy Procedia*, vol. 14, pp. 1226 – 1231, 2012.

[110] Y.-C. Huang, H.-T. Yang, and C.-L. Huang, "Developing a new transformer fault diagnosis system through evolutionary fuzzy logic," *Power Delivery, IEEE Transactions*, vol. 12, no. 2, pp. 761–767, 1997.

[111] F. Bobillo and U. Straccia, "Fuzzy ontology representation using OWL2," *International Journal of Approximate Reasoning*, vol. 52, no. 7, pp. 1073–1094, 2011.

[112] H. Ma, Z. Li, P. Ju, J. Han, and L. Zhang, "Diagnosis of power transformer faults on fuzzy three-ratio method," in *Power Engineering Conference, 2005. IPEC 2005. The 7th International*, pp. 1–456, 2005.

[113] F. Bobillo and U. Straccia, “Fuzzy ontology representation using OWL 2,” *CoRR*, vol. abs/1009.3391, 2010.

[114] F. Bobillo and U. Straccia, “fuzzyDL: An expressive fuzzy description logic reasoner,” in *Fuzzy Systems, 2008. FUZZ-IEEE 2008. (IEEE World Congress on Computational Intelligence). IEEE International Conference*, pp. 923–930, 2008.

[115] W. Rodder and G. Kern-Isberner, “Self learning or how to make a knowledge base curious about itself,” in *Advances in Artificial Intelligence* (A. Gnter, R. Kruse, and B. Neumann, eds.), vol. 2821 of *Lecture Notes in Computer Science*, pp. 465–474, Springer Berlin Heidelberg, 2003.