PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministère de Higher Education and Scientific Research
University Mohamed Khider – BISKRA
Faculty of Exact Sciences, Science of Nature and Life
Computer Science department

N° order : /M2/2018

Memory

presented to obtain the diploma of academic master in

COMPUTER SCIENCE

option : **Artificial intelligence**

Modeling of Prognostic by SMA in the PHM

By:

ZERARI DJAMEL EDDINE

Defended on June 25, 2018, before the jury composed of

GUERROUF FAYCAL	M.A.A	President
TERRISSA SADEK LABIB	M.C.A	Supervisor
ALOUI AHMED	M.A.A	Examiner

Abstract

prognostics and health management (PHM) systems have been studied by many researchers from many different engineering fields to increase system reliability, availability, safety and to reduce the maintenance cost of engineering assets. Many works conducted in PHM research concentrate on designing robust and accurate models of Multi Agent System to assess the health state of components for particular applications to support prognostics . Models which involve mathematical interpretations, assumptions and approximations make PHM hard to understand and implement in real world applications, especially by Predictive maintenance . Prior knowledge to implement PHM in Multi Agent System is crucial to building highly reliable systems. To fill this gap and motivate industry practitioners, this work attempts to model this process using MAS in PHM domain and discusses important issues on, implementation aspects next to prognostics feature and tool evaluation.

Les systèmes de pronostics et de gestion de la santé (PHM) ont été étudiés par de nombreux chercheurs issus de nombreux domaines d'ingénierie différents afin d'améliorer la fiabilité, la disponibilité et la sécurité du système et de réduire les coûts de maintenance des ressources d'ingénierie. De nombreux travaux menés dans le cadre de la recherche PHM se concentrent sur la conception de modèles robustes et précis de Multi Agent System pour évaluer l'état de santé des composants pour des applications particulières afin de soutenir les pronostics. Les modèles qui impliquent des interprétations mathématiques, des hypothèses et des approximations rendent difficile la compréhension et la mise en œuvre de PHM dans des applications réelles, notamment par la maintenance prédictive. La connaissance préalable de la mise en œuvre de PHM dans un système multi-agent est cruciale pour la construction de systèmes hautement fiables. Pour combler cette lacune et motiver les praticiens de l'industrie, ce travail tente de modéliser ce processus en utilisant MAS dans le domaine PHM et discute des questions importantes sur les aspects de mise en œuvre à côté de la fonctionnalité de pronostic et l'évaluation des outils.

التنبؤ وصحة الإدارة لقد قام العديد من الباحثين بتطوير و بحث في هذا المجال لسعي للوصول إلى نضام ثابت و مرن من متعدد الوكلاء ليضمن   اقل قيمة من تكليف الصيانة و كبر قدر من وقاية من اعطاب ولكن واجهوه تحديات كبيرة وهذا يرجع إلى طبيعة المجال الذي يحتوي قدر كبير من متغيرات ومنه نحاول في هذه المذكرة تقديم عمل يدعم هذه الجهود و يوضح طبيعة المجال وشرح أساليب متبعة لي تطبيقه في صيانة التنبؤية للقطارات

1-What is maintenance and why is it performed?

Past and current maintenance practices in both the private and government sectors would imply that maintenance is the actions associated with equipment repair after it is broken. The dictionary defines maintenance as follows: “the work of keeping something in proper condition upkeep .This would imply that maintenance should be actions taken to prevent a device or component from failing or to repair normal equipment degradation experienced with the operation of the device to keep it in proper working order. Unfortunately, data obtained in many studies over the past decade indicates that most private and government facilities do not expend the necessary resources to maintain equipment in proper working order. Rather, they wait for equipment failure to occur and then take whatever actions are necessary to repair or replace the equipment. Nothing lasts forever and all equipment has associated with it some predefined life expectancy or operational life. [1] For example, equipment may be designed to operate at full design load for 5,000 hours and may be designed to go through 15,000 starts and stop cycles.

The need for maintenance is predicated on actual or impending failure – ideally, maintenance is performed to keep equipment and systems running efficiently for at least design life of the component(s). As such, the practical operation of a component is time-based function. If one were to graph the failure rate a component population versus time, it is likely the graph would take the “bathtub” shape shown in Figure 1 In the figure the Y axis represents the failure rate and the X axis is time. From its shape, the curve can be divided into three distinct: infant mortality, useful life, and wear-out periods.

The initial infant mortality period of bathtub curve is characterized by high failure rate followed by a period of decreasing failure. Many of the failures associated with this region are linked to poor design, poor installation, or misapplication. The infant mortality period is followed by a nearly constant failure rate period known as useful life. There are many theories on why components fail in this region, most acknowledge that poor O&M often plays significant role. It is also generally agreed that exceptional maintenance practices encompassing preventive and predictive elements can extend this period. The wear-out period is characterized by a rapid increasing failure rate with time. In most cases this period encompasses the normal distribution of design life failures.

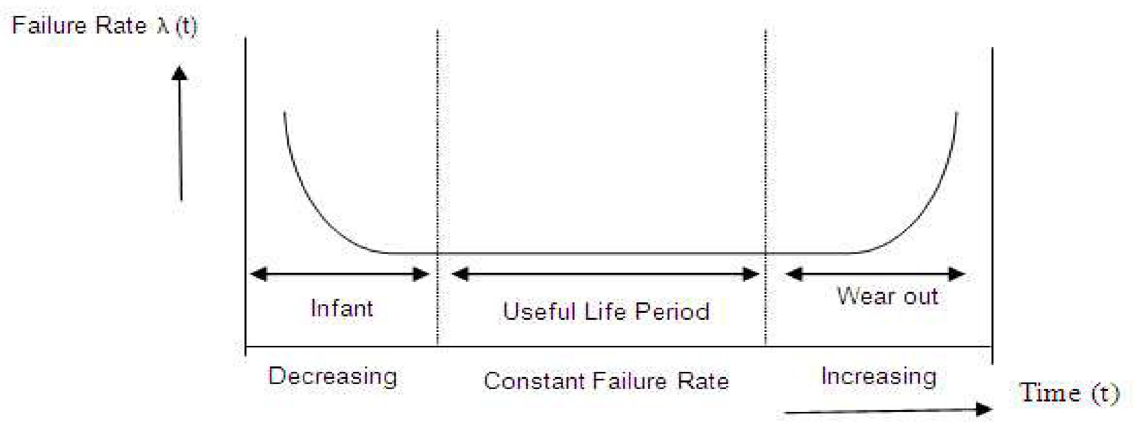


Figure 1. Component failure rate over time for component population.

2-Types of maintenance

Traditionally, 5 types of maintenance have been distinguished, which are differentiated by the nature of the tasks that they include [2]:

2-1-Corrective maintenance:

The set of tasks is destined to correct the defects to be found in the different equipment and that are communicated to the maintenance department by users of the same equipment.

2-2-Preventive Maintenance:

Its mission is to maintain a level of certain service on equipment, programming the interventions of their vulnerabilities in the most opportune time. It is used to be a systematic character, that is, the equipment is inspected even if it has not given any symptoms of having a problem.

2-3- Predictive Maintenance:

It pursues constantly know and report the status and operational capacity of the installations by knowing the values of certain variables, which represent such state and operational ability. To apply this maintenance, it is necessary to identify physical variables (temperature, vibration, power consumption, etc.). Which variation is indicative of problems that may be appearing on the equipment. This maintenance it is the most technical, since it requires advanced technical resources, and at times of strong mathematical, physical and / or technical knowledge.

2-4-Zero Hours Maintenance (Overhaul):

The set of tasks whose goal is to review the equipment at scheduled intervals before appearing any failure, either when the reliability of the equipment has decreased considerably so it is risky to make forecasts of production capacity . This review is based on leaving the equipment to zero hours of operation, that is, as if the equipment were new. These reviews will replace or repair all items subject to wear. The aim is to ensure, with high probability, a good working time fixed in advance.

2-5-Periodic maintenance (Time Based Maintenance):

the basic maintenance of equipment made by the users of it. It consists of a series of elementary tasks (data collections, visual inspections, cleaning, lubrication, retightening screws,…) for which no extensive training is necessary, but perhaps only a brief training. This type of maintenance is the based on TPM (Total Productive Maintenance).

3-Predictive Maintenance

3-1-Definition

Predictive maintenance can be defined as follows: Measurements that detect the onset of system degradation (lower functional state), thereby allowing causal stressors to be eliminated or controlled prior to any significant deterioration in the component physical state. Results indicate current and future functional capability.

Basically, predictive maintenance differs from preventive maintenance by basing maintenance need on the actual condition of the machine rather than on some preset schedule. You will recall that preventive maintenance is time-based. Activities such as changing lubricant are based on time, like calendar time or equipment run time. For example, most people change the oil in their vehicles every 3,000 to 5,000 miles traveled. This is effectively basing the oil change needs on equipment run time [3]. No concern is given to the actual condition and performance capability of the oil. It is changed because it is time.

This methodology would be analogous to a preventive maintenance task. If, on the other hand, the operator of the car discounted the vehicle run time and had the oil analyzed at some periodicity to determine its actual condition and lubrication properties, he/she may be able to extend the oil change until the vehicle had traveled 10,000 miles. This is the fundamental difference between predictive maintenance and preventive maintenance, whereby predictive maintenance is used to define needed maintenance task based on quantified material/equipment condition.

3-2-Advantages

The advantages of predictive maintenance are many. A well-orchestrated predictive maintenance program will all but eliminate catastrophic equipment failures. We will be able to schedule maintenance activities to minimize or delete overtime cost. We will be able to minimize inventory and order parts, as required, well ahead of time to support the downstream maintenance needs. We can optimize the operation of the equipment, saving energy cost and increasing plant reliability [4] Past studies have estimated that a properly functioning predictive maintenance program can provide a savings of 8% to 12% over a program utilizing preventive maintenance alone. Depending on a facility's reliance on reactive maintenance and material condition, it could easily recognize savings opportunities exceeding 30% to 40%. In fact, independent surveys indicate the following industrial average savings resultant from initiation of a functional predictive maintenance program:

- Return on investment: 10 times
- Reduction in maintenance costs: 25% to 30%
- Elimination of breakdowns: 70% to 75%
- Reduction in downtime: 35% to 45%
- Increase in production: 20% to 25%.

3-3-Disadvantages

On the down side, to initially start into the predictive maintenance world is not inexpensive. Much of the equipment requires cost in excess of $50,000. Training of in-plant personnel to effectively utilize predictive maintenance technologies will require considerable funding. Program development will require an understanding of predictive maintenance and a firm commitment to make the program work by all facility organizations and management.

4-Prognostics and health management

Prognostics and health management (PHM) is an engineering discipline that aims at minimizing maintenance cost by the assessment, prognosis, diagnosis, and health management of engineered systems. With an increasing prevalence of smart sensing and with more powerful computing[5] PHM has been gaining popularity across a growing spectrum of industry such as aerospace, smart manufacturing, transportation, and energy at breakneck speed. Regardless of application, one common expectation of PHM is its capability to translate raw data into actionable information to facilitate maintenance decision making. This practice in industry is often referred to as Predictive Maintenance, which, as estimated by Accenture , could possibly save up to 12% over scheduled repairs, reduce overall maintenance costs by up to 30% and eliminate asset failures up to 70%. For example, a study performed by National Science Foundation (NSF) indicates that Center for Intelligent Maintenance Systems (IMS), which is a leading research center in the field of PHM, has created more than $855 M of economic impact to the industry with a benefit cost ratio of through the development and deployment of PHM technologies to achieve near-zero unplanned downtime and a more optimized maintenance practice. However, the value of PHM does not stop at maintenance alone. By performing smart analytics to asset usage data, users would be able to gain knowledge about how to achieve optimized performance of the asset. For instance, the State of Health and Remaining Useful Life (RUL) of batteries on electric vehicles are highly dependent on the driving behavior. By analyzing the relationship between driving behavior and battery condition, a customized solution could be provided for the improvement of user's driving [6] behavior and thus prolong battery life. Also, by relating process data with product quality metrics, predictive error compensation can be realized for increased product quality assurance. Additionally, asset usage and failure analysis could be fed back to the designers and manufacturers of the asset to nurture customer co-creation for an improved product design.

5-PHM MAIN TASKS

In this section, PHM implementation steps are discussed and explained in detail to instruct practitioners and make them familiar with PHM infrastructure. Steps involve data acquisition, data preprocessing, detection, diagnostics and prognostics, decision making and finally human-machine interface [7] PHM steps are depicted in Fig. 2. Each step will be explained in the following subsections.

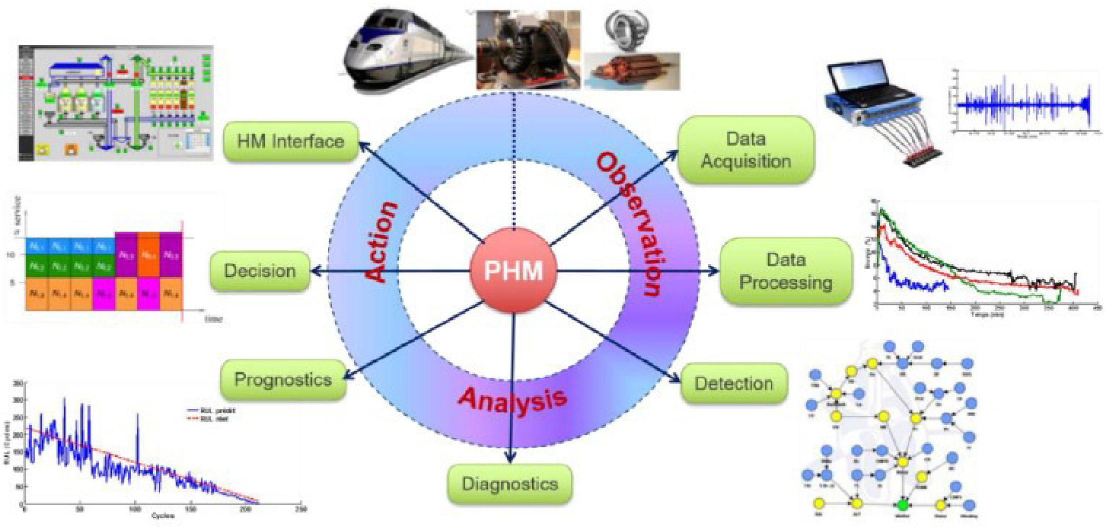


Fig. 2. PHM steps.

5-1-Data Acquisition

Data acquisition is an initial and essential step of PHM which is known as a process of data collection and storage from physical component/system under investigation for further diagnostics and prognostics purposes. Collected data could be either sensory data or event data (ED)[9]. The ED include the information of maintenance actions (e.g. oil change, repairs etc.) taken on the events (e.g. failure, breakdown, installation etc.) that happened to the physical component. CM or sensory data are measurements tracked via installed sensors from asset under investigation, such as; acoustic emission data, vibration data, temperature, pressure, humidity, resistance, voltage, etc. Where ED includes events performed by maintenance technician, such as corrective maintenance, asset repairs, installation, breakdown, cleaning and oiling on the component/system. Data acquisition process is depicted in Fig. 3 for a railway point machine example. ISO definition for data acquisition can be found in (ISO 13374-1:2003, n.d.).

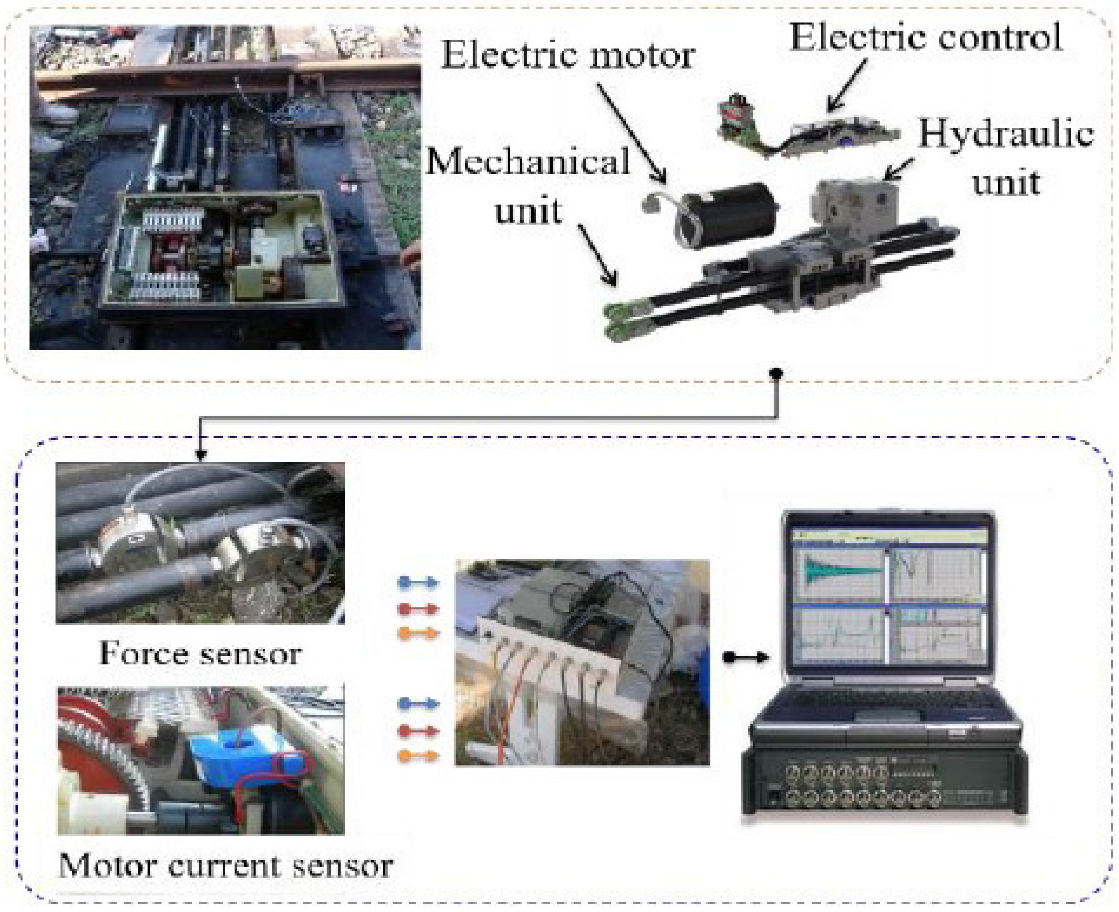


Fig. 3. Data acquisition process for railway point machine.

5-2-Data Preprocessing

Data preprocessing involves data cleaning and data analysis steps. Cleaning errors/noise from raw data increases the chance of getting error-free data for further investigations. Data analysis, which is the second step of data preprocessing, involves feature extraction, feature evaluation, and selection processes. Cleaned sensory time series should undergo a feature extraction process to extract only the important and useful features that reflect system health state being monitored. Extracted features should indicate the failure progression of the system. The feature extraction techniques are categorized as time-domain based, frequency-based and time-frequency based techniques in the literature The time-domain based feature extraction techniques are used to analyze the global characteristics of data and to extract the features in time domain [10]. The frequency-domain based feature extraction techniques (e.g. Fourier transform, envelop analysis etc.) transform the data into frequency domain and are used to detect and identify a faults which are not possible by time-domain based techniques. The time-frequency domain based techniques (e.g. Fourier transform, envelop analysis etc. Hilbert-Huang transforms Wigner-Ville distribution etc.) analyze the data in both time and frequency domains. Feature evaluation and selection process is the second important step of data analysis after extraction. A feature evaluation can be defined as a feature goodness quantification process in feature selection. There are different techniques used to quantify the feature goodness (i.e. degradation trend) such as monotonicity, prognosability and trendability The best features, which have clear degradation trend, are further selected in a feature selection process after evaluation (Kimotho & Sextro, 2014). More information on feature extraction techniques .

An accurate prediction of remaining-useful-life (RUL) of assets depends on well evaluated and selected prognostic features. Overall data preprocessing procedure is depicted in Fig. 4. ISO definition for data preprocessing can be found in (ISO 13374-1:2003, n.d.).

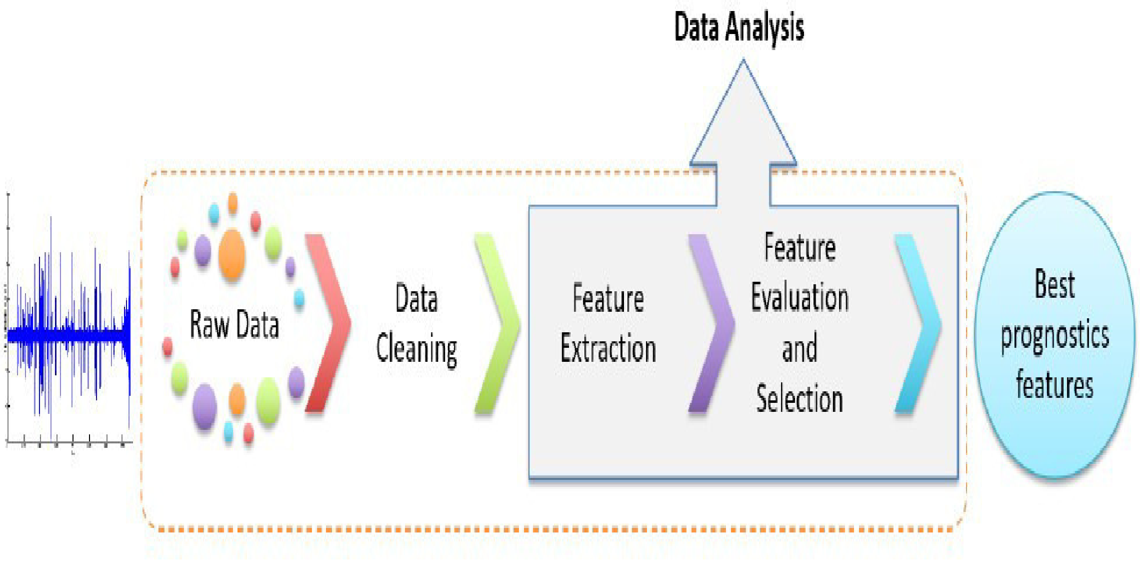


Fig. 4. Data preprocessing procedure

5-3-Detection

There are many factors that cause system components to degrade over time, losing their initial performance, and which therefore need to be considered in detection modeling. Health state detection is the process of detecting and recognizing incipient failures and/or anomalies from CM data[11] A fault detection is typically based on the quantification of the inconsistencies between the actual and the expected behavior of the system in nominal conditions. Fig. 5 illustrates failure propagation of component based on CM data.

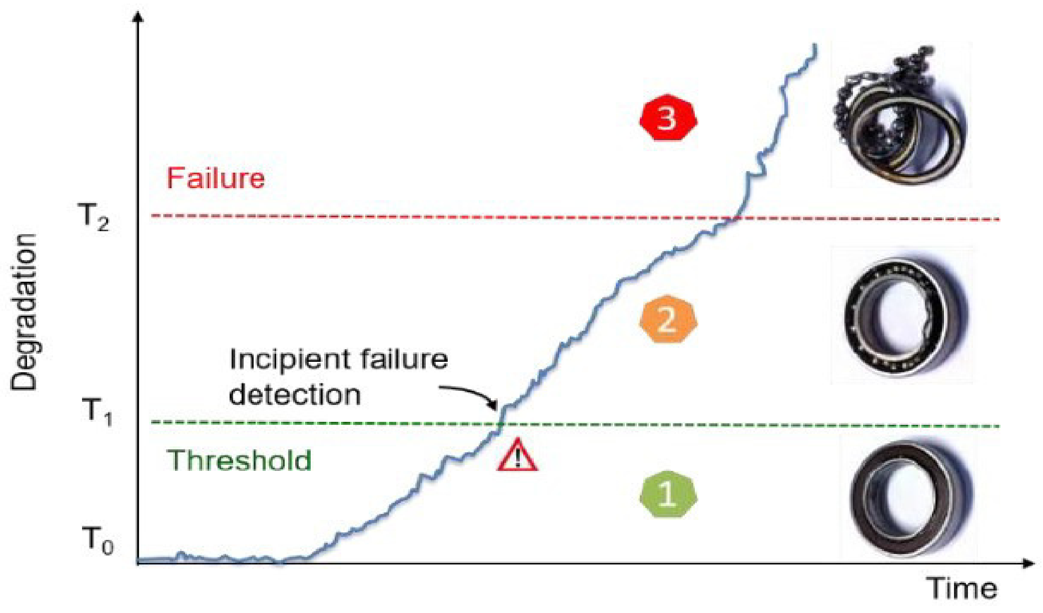


Fig. 5. Component failure propagation.

In Fig. 5, a component CM indicator increases with time as the component degrades. As illustrated in the figure, the evolution of the health state of the component can be divided into 3 phases; Phase-1 (T0 <T1) where the component is in a healthy state, Phase-2 (T1< T2) where the component is in a faulty state, Phase-3 (T2<) where the component is in the completely failed state. Predefining thresholds (T1 and T2) is very challenging, which needs serious experience for practitioners[11]Historical CM data can be also used to compare and set new thresholds for the same type of components. Time-to-failure estimation is performed in Phase-2 after the detection process where maintenance activities are planned based on estimated time-to-failure. Therefore, early detection of component failure is important. Fig. 5 illustrates features' health state transitions for slowly propagating failures (healthy-faulty-failed) and for sudden failures (healthy-failed). ISO definition for fault detection can be found in (ISO 13374-1:2003, n.d.).

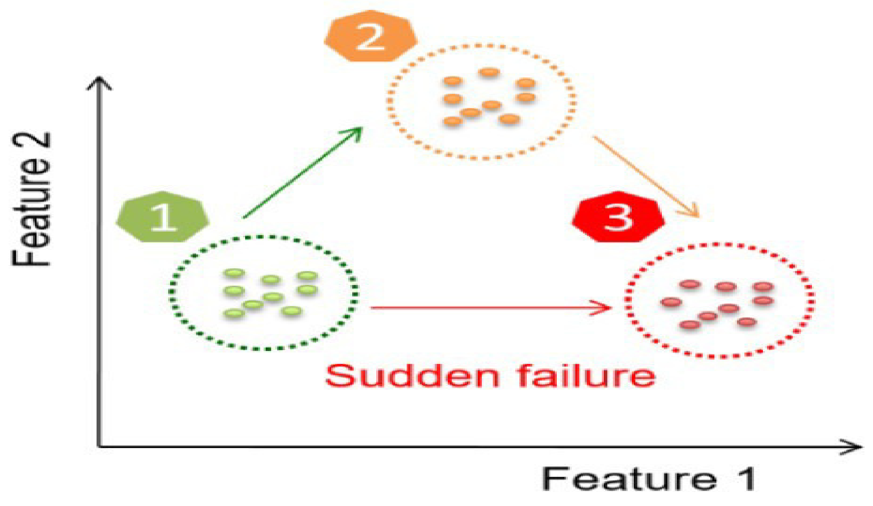


Fig. 6. Feature health state transitions.

5-4-Diagnostics

Fault diagnostics is a process of fault detection, isolation (i.e. which component is failed), failure mode identification (i.e. what is the cause of failure or fault) and degradation level assessment (i.e. quantification of the failure severity) in condition monitoring[12] Diagnostics can be conducted when a machine is either in complete failure state or in faulty state. Fig. 7 illustrates the post-mortem fault diagnostics for failed component. Diagnostics results can be used for reactive as well as proactive decision making (the latter when diagnosing a degraded condition, as opposed to a complete failure). ISO definition for fault diagnostics can be found in (ISO 13372:2012, n.d.).

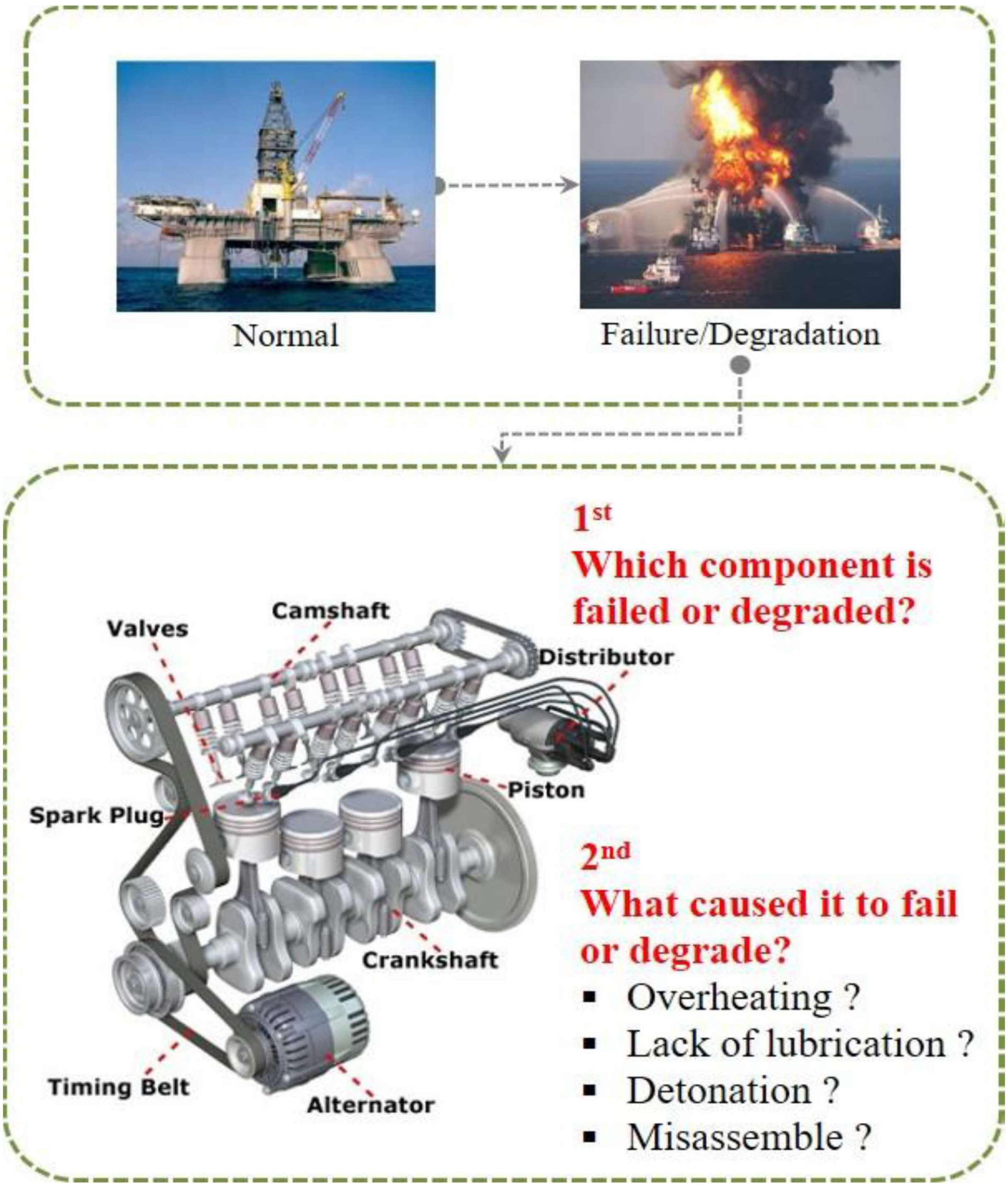


Fig. 7. Diagnostics.

5-5-Prognostics

Prognostics is defined as the process of predicting the time (RUL) at which a component will no longer perform a particular function and it is illustrated in Fig. 8. Prognostics results are used to support proactive decision making. ISO definition for fault prognostics can be found in (ISO 13381-1:2005, n.d.).

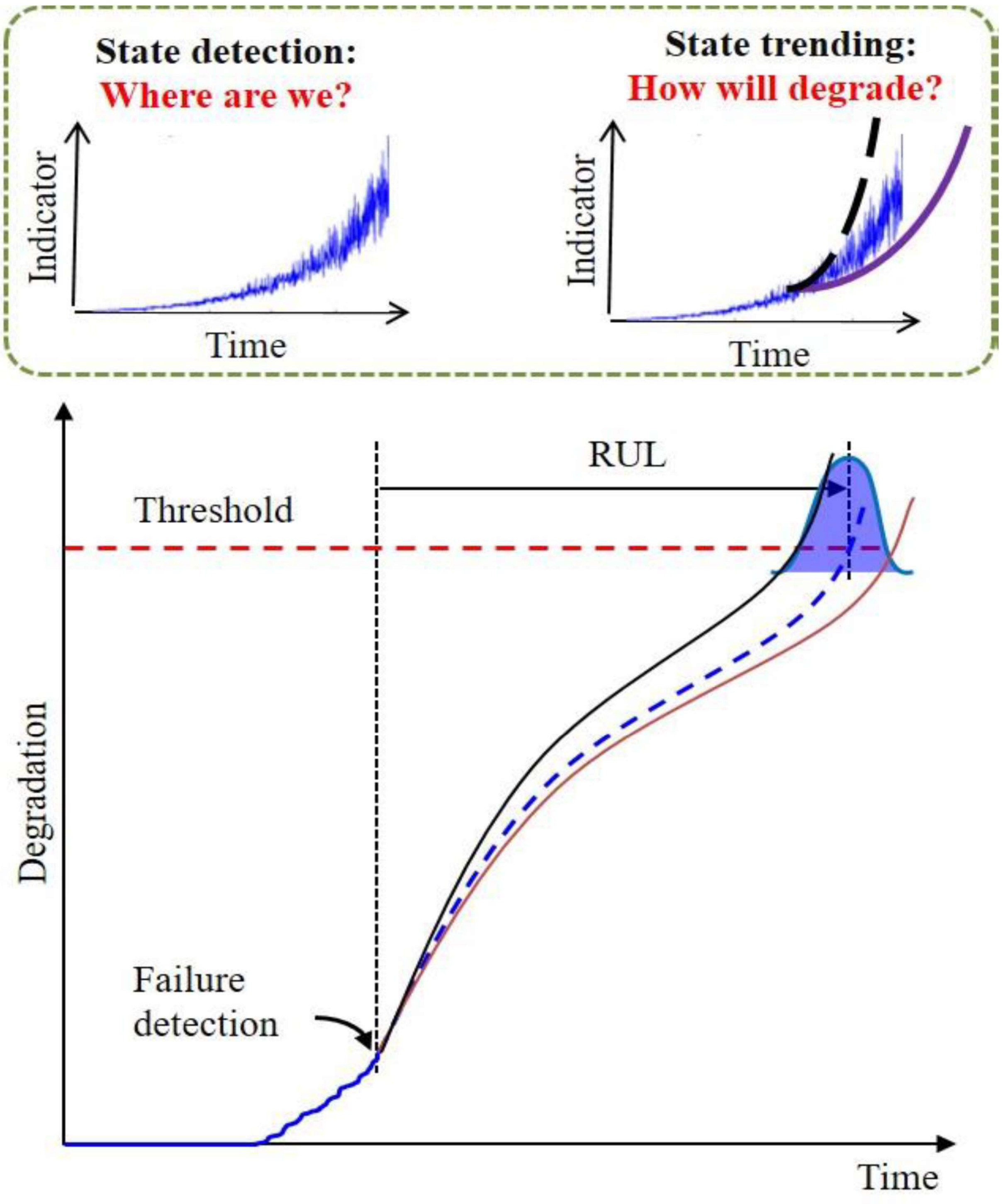


Fig. 8. Prognostics.

As illustrated in two examples (Fig. 7 and Fig. 8), prognostics serves as prevention of system from possible failures by predicting future states while diagnostics is concerned with fault isolation and classification process. Prognostics and diagnostics difference is depicted in Fig. 9.

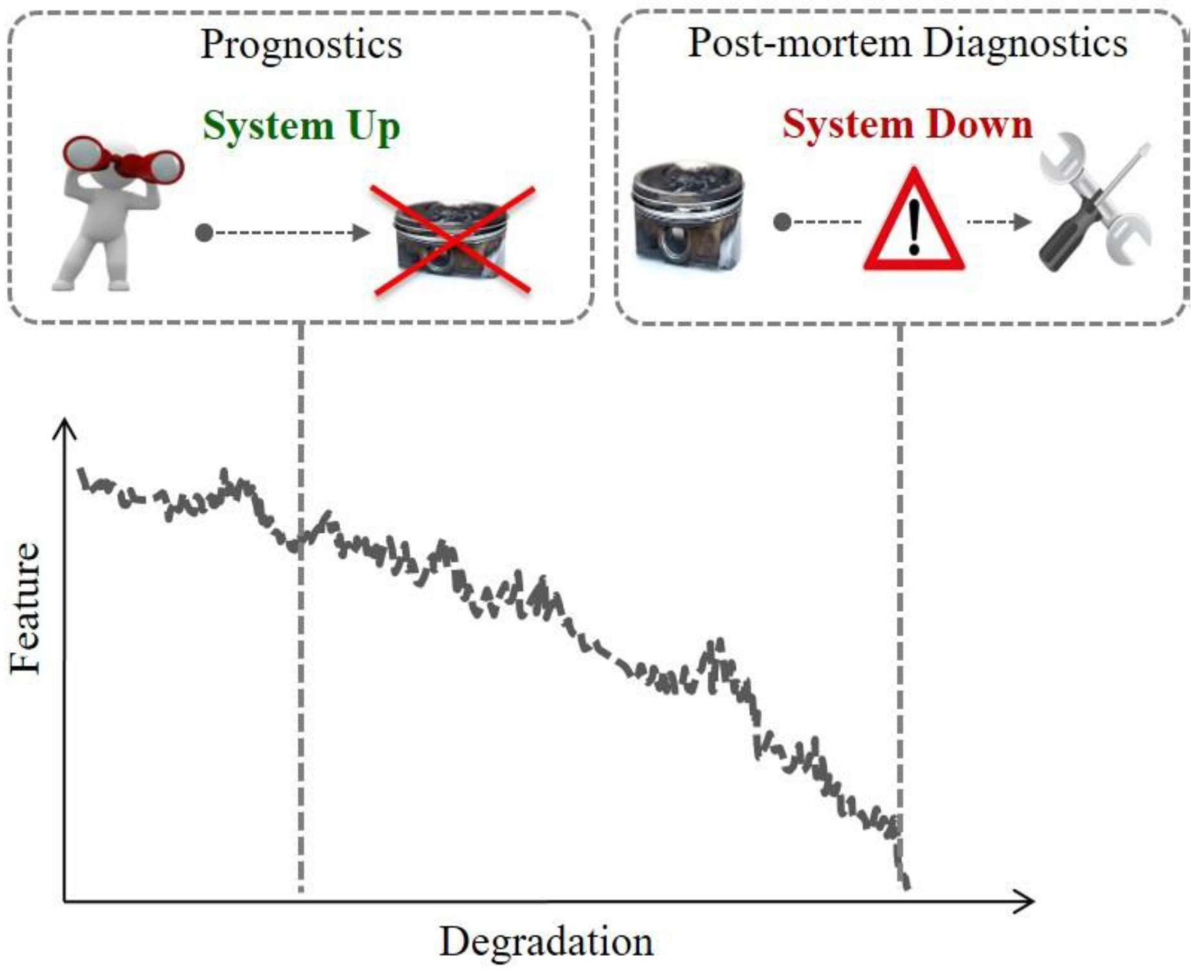
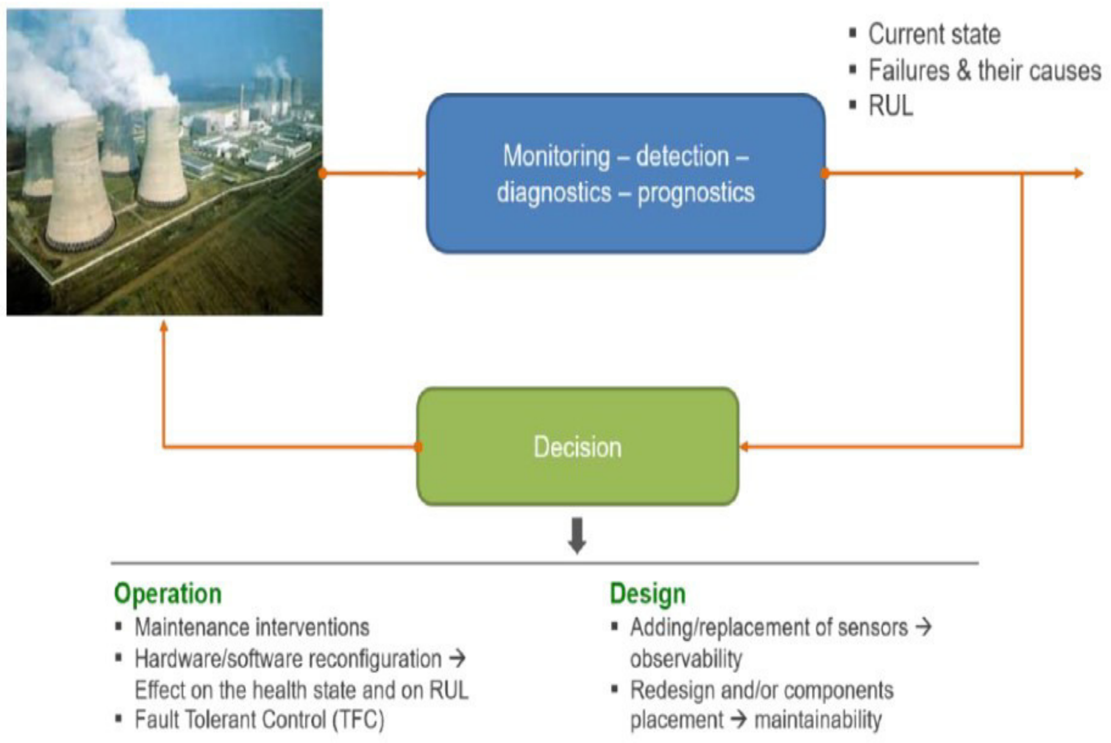


Fig. 9. Prognostics vs. Diagnostics.

5-6-Decision Making

Decision making is a process resulting in the selection of logical and/or right maintenance action among several alternatives. Maintenance technician must evaluate the negatives and positives of each action based on the diagnostics or prognostics results. To make effective decisions, the technician also should be able to estimate the outcomes of each alternative as well. ISO definition for decision making can be found in (ISO 13374-1:2003, n.d.)

Outcomes of decisions could be either operational or design based. Decisions made on operational actions could be maintenance interventions, hardware/software reconfigurations and fault tolerant control (FTC). Design based outcomes could be adding and/or replacing sensors observability and redesign and/or components placement. Decision-making process is illustrated in Fig. 10.



▪ Current state
▪ Failures & their causes
▪ RUL
Monitoring – detection – diagnostics – prognostics
Decision
Operation
▪ Maintenance interventions
▪ Hardware/software reconfiguration → Effect on the health state and on RUL
▪ Fault Tolerant Control (TFC)
Design
▪ Adding/replacement of sensors → observability
▪ Redesign and/or components placement → maintainability

Fig. 10. Decision making.

5-7-Human-Machine Interface

Human-machine interface is Graphical User Interface (GUI) which is used to visualize component health status, to execute tasks, to analyze data and to control the maintenance operations.

6-CONCLUSION

PHM technology is employed widely to enhance system availability and safety and to analyze the system performance based on time series data acquired from different sensors depending on component functionality. We have presented a general view of PHM and its steps to provide prior knowledge for users, reviewed different PHM approaches under model-based, data-driven and hybrid models, and discussed their merits and drawbacks. We have also reviewed previous and on-going research in bogie components PHM to highlight problems faced in the railway industry. As a result of PHM literature review on bogie components, we noticed that nearly all research conducted in bogie health assessment is mostly limited to diagnostics rather than prognostics tasks. Since railway vehicle bogies are critical components, research on prognostics for asset health management is also crucial to provide a safe and comfortable ride for customers. Consequently, if somebody wants to implement PHM technology at system or component level, the first step is to identify critical components in the system that have a great impact on system functionality when they fail. The secondly step is to select and install right and robust sensors on the system for accurate CM. Collected CM data should be properly processed for good prognostics feature extraction, as a third step. Finally, a suitable PHM approach should be carefully selected based on the user requirements as well as a prognostics tools selection process. Since accurate prognostics results are based on the tools used in prediction, the tools evaluation and selection process is an important task in PHM implementation. Two separate prognostics tools evaluation matrices were presented, for both model-based and data-driven approaches. Prognostics tools were evaluated and ranked based on a combination of user knowledge and tool performance metrics. As a result of our investigation, we can conclude that further investigations and improvements should be carried out in following areas for bogie prognostics;

- High-speed train bogies have dynamic loads and environmental conditions which might affect component failures differently. To detect and monitor component behaviors under different loads, intelligent component criticality identification systems should be developed.
- Intelligent on-board or off-board data preprocessing systems which analyze CM data efficiently and extract better health indicators for prognostics should be designed.

Efficient and intelligent prognostics tool evaluation and selection systems that integrate user requirements with tool performance metrics, tool applicability, CM data characteristics and suggest a suitable tool for better component RUL prediction with minimum uncertainties should be developed.

Chapter II : Multi-Agent System

1.Introduction

More than twenty years of research in the field of agents and multi-agent systems have offered remarkable results from both theoretical and practical point of view. However, after so many years of the research there is no general consensus on some basic notions such as “what is an agent?”, “what is a multi-agent systems”, agreement on the terminology used, etc.

Indeed, this fact makes confused those interested in applying agent based or multi-agent based technology to solve practical problems. This short note is intended to serve as a “gentle” introduction to the field of agents and multi-agent systems particularly for those interested in using these technologies in solving practical engineering problems[13] The main purpose of the note is to familiarize readers with basic terminology and definitions. This note will not enumerate (many) and elaborate different views on the subject but rather presents a view on
the subject that (according to authors of the note) may help engineering people to get feeling on this important subject.[14]Throughout of this note, when deemed necessary, simple examples are provided to illustrate different concepts. All the examples are based on the idea to use agent or multi-agent technology to design effective load shedding scheme. For readers interested for a deeper inside in the subject, from different points of view, this note offers at the end a list of useful references and links.

2. What is an agent?

There is no strict definition about what an agent is. Literature offers a variety of definitions ranging from the simple to the lengthy and demanding ones. All available definitions are strongly biased by the background field interested in agent technology (main being: artificial intelligence, software engineering, cognitive science, computer science, engineering in general, etc.).
Instead to list and elaborate different definitions, two definitions of the agent coming from [15] and [16] are given here since they seem to be rather general, widely accepted by different research communities, and closest to power engineering people perception of agents. In [15], an agent is defined as follows,
“An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors”.
According to this definition an agent is any entity (**physical** or **virtual** one) that senses its environment and acting over it.

Physical entities that could be considered as agents are, in the case of a power system, simple protection relay or any controller that controls directly particular power system component or part of the system.

Virtual entity that can be considered as an agent is a piece of software that receives inputs from an environment and produces outputs that initiate acting over it.

Often an agent is a combination of physical (computation architecture) and virtual one (a piece of software running on the computational architecture).

A more precise definition of the agent, that is in fact extension of the definition mentioned above, is given in [16],

"Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are deigned".

The key extensions in this definition with respect to first one are words: **computational**,**autonomy**, and **goals**. Word **computational** makes difference in agents that we are interested in engineering (**computational agents**) from biological agents (humans, animals, bacteria) since from first definition this distinction is not obvious. **Autonomy** means that computational agents operate without the direct intervention of some other entities and have some kind of control over their actions. Assigning the **goals** to the agent means that acting upon environment should be done in order to achieve some specified objective (goal) and that the agents expose a sort of **rational** (an agent that minimizes or maximizes its performance measure) behavior in the environment. **Behavior** here means the action that is performed after receiving sensory inputs (or any sequence of sensory inputs) [15]. A general single-agent framework is illustrated in Figure 11.

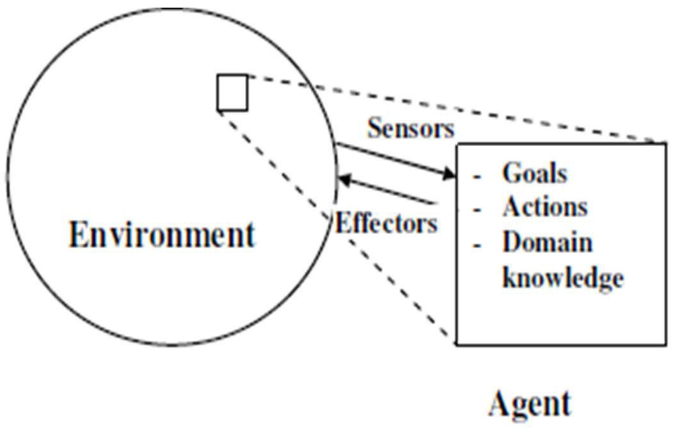


Figure 11. A general single-agent framework.

In addition to the sensory inputs, actions, and goals an agent may include domain knowledge(the knowledge about particular environment or problem to be solved). This knowledge can be algorithmic, artificial intelligence (AI) technique based (rule-based, fuzzy, neural networks, machine learning), heuristics, etc. In AI case an agent is often termed as **intelligent** one.

The notion of environment that an agent inhabits (is situated in or simply said placed in) include physical systems (as in engineering), operating system, the Internet, or perhaps some of these systems combined.

Observe also that physical entities and virtual ones can be combined to constitute an agent.

If an agent simply responds, in timely fashion, to changes in the environment and reactively converts its sensory inputs to actions, this agent is known as **reactive** (sometimes termed **reflex** agent). Reactive agents usually do not maintain internal state (a simple example of internal state is keeping a set of previous sensory inputs) of the agent and do not predict the effect of the actions. For can be said is that if an agent predicts the effect of the actions then it is not reactive (meaning that reactive agent may also maintain its internal state).

On the other hand, if an agent maintains internal state predicts the effects of its actions, or more generally includes a sort of reasoning (behave more like they are " thinking"), that agent behaves deliberatively and is termed as **deliberative** agent (or agent that keeps the track of environment).

Figure 12 illustrates reactive and deliberative agents. Some of the authors do not assign any goals to reactive agents but the goals can be embodied in the sensory inputs processing or condition-action rules.

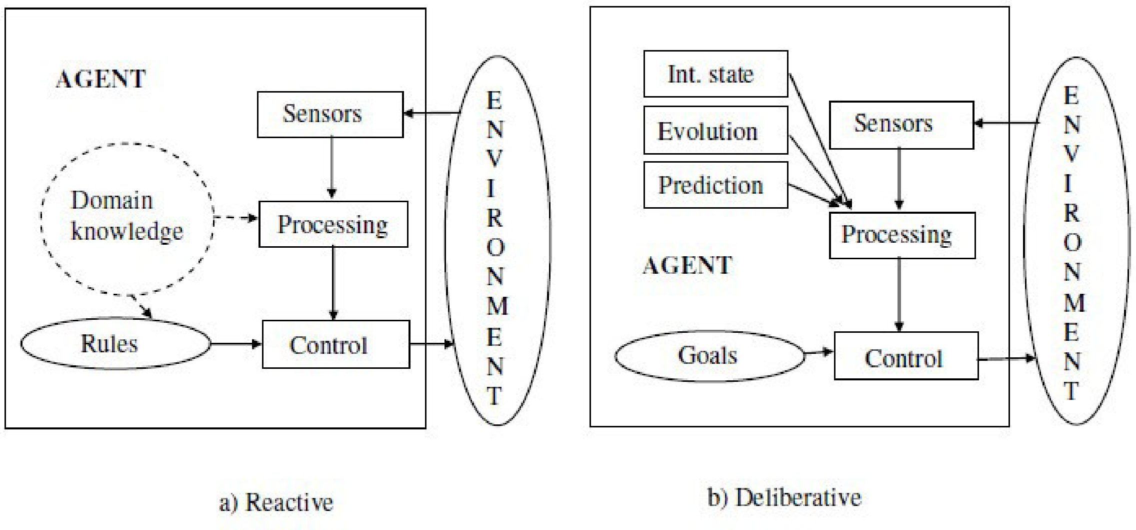


Figure 12. Reactive (reflex) and deliberative agents

The various definitions of agents involve a host of properties of an agent and agents are usually classified based on those properties.
Table I lists several properties that can be encountered in literature and are given here for the reader just to have an idea what the particular term (often considered as a type of agent) means.

Table I. Some properties of agents

Property	Other names	Meaning
Reactive	Reflex, sensing and acting	Responds in a timely fashion to changes in the environment
Autonomous		Exercises control over its own actions
Goal-oriented	Pro-active, purposeful	Does not simply act in response to the environment
Temporally continuous		Is a continuously running process
Communicative	Socially able	Communicates with other agents
Learning	Adaptive	Changes its behavior based on its previous experience
Mobile		Able to transport itself from one machine to another (this is associated manly with software agents)
Flexible		Actions are not scripted

3. What is a Multi-agent system (MAS)?

As for the definition of an agent the same holds for definition of a multi-agent system various definitions have been proposed depending on research discipline it is coming from again, the aim here is not to list and elaborate different definitions and rather a single definition is chosen that seems to be general and close to power engineering research community. In [17], a multi-agent system is defined as,

" A multi-agent system is a loosely coupled network of problem-solving entities (agents) that work together to find answers to problems that are beyond the individual capabilities or knowledge of each entity (agent)" .

The fact that the agents within a MAS work together implies that a sort of cooperation among individual agents is to be involved. However, the concept of cooperation in MAS is at best unclear and at worst highly inconsistent, so that the terminology, possible classifications, etc., are even more problematic then in the case of agents what makes any attempt to present MAS a hard problem.

A typology of cooperation from [18] seems the simplest and here we start with this typology as the basis for MAS classification. The typology is given in
Figure1 4.

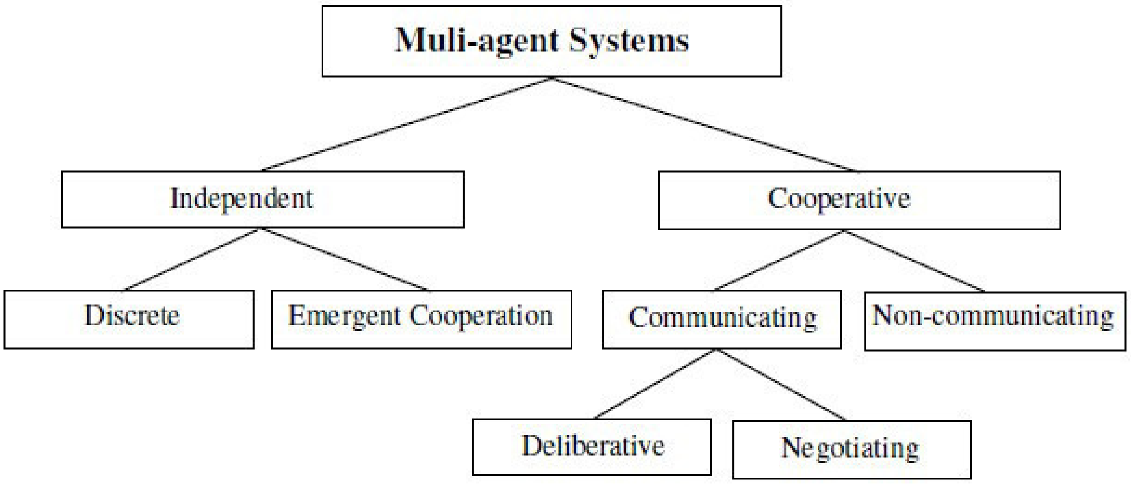


Figure 13. Cooperation typology

A MAS is independent if each individual agent pursues its own goals independently of the others. A MAS is discrete if it is independent, and if the goals of the agents bear no relation to one another. Discrete MAS involve no cooperation. However agents can cooperate with no intention of doing so and if this is the case then the cooperation is emergent. The term deliberative in Figure 13 is not to be mixed with the term of deliberative agent, and in the figure it refers to the idea that agents jointly plan their actions so as to cooperate with each other. The cooperation within a MAS can be realized in three ways:

- by explicit design (the designer of the agents purposely designs the agent behaviors so that cooperation occurs),
- by adaptation (individual agents learn to cooperate),
- by evolution (individual agents cooperation evolves through some kind of evolutionary process).

4. What are advantages of MAS?

Main advantages of MAS are robustness and scalability. Robustness refers to the ability that if control and responsibilities are sufficiently shared among agents within MAS, the system can tolerate failures of one or more agents.
Scalability of MAS originates from its modularity. It should be easier to add new agents to a MAS than to add new capabilities to a monolithic system.

5. Independent discrete MAS

This type of MAS is encountered in the environments that permit decoupling or decomposition (spatial, temporal). As an example of this type of MAS we use two agents (controllers) controlling synchronous generator in a power system: automatic voltage regulator (AVR) and speed governor. They have different goals that bear no relation to one another. AVR goal is to keep terminal voltage at predefined value and speed governor goal is to keep angular speed at synchronous value. AVR exercises control over the excitation system while speed governor exercises control by controlling inflow of primary medium (steam,water). They are designed independently and based on engineering observation of temporal decomposition (AVR acts much faster) and coupling between active power and speed as well reactive power and voltage magnitude). Of course one might argue that these agents are not completely independent but here serve to illustrate an example that looks like independent discrete and to point out prerequisite of decomposition and decoupling to realize this type of MAS.

6. Independent MAS with emergent cooperation

In this MAS agents are designed independently and each individual agent pursues its own goals independently of the others. It is more important to stress that in this MAS the individual agents are not aware of existence of other agents and each agent considers others as a part of the environment. Since agents exist in the same environment they can affect each other indirectly and the cooperation can emerge with no intention of doing so. The cooperation among independent agents can emerge in two ways:

- individual agents receive as sensory inputs (directly or they estimate them) the inputs or control of one or more other agents in the environment,
- individual agents, by their actions, change sensory inputs of another agent in a cooperative way without explicit intention of doing so.

Let us use again simple example to illustrate how cooperation can emerge in individual MAS.

7. Cooperative MAS

Since the aim of this note is to present the subject as clearly as possible to be understandable for power system researchers, the choice in the remaining of this text is to stick with presenting cooperative MAS from two dimensions: agent heterogeneity and amount of communication among individual agents within a MAS. With respect to these two dimensions cooperative MAS can be classified in four groups:

1. **Homogeneous non-communicating MAS,**
2. **Homogeneous communicating MAS,**
3. **Heterogeneous non-communicating MAS, and**
4. **Heterogeneous communicating MAS.**

8. Homogeneous non-communicating MAS

In this MAS there are several different agents with identical structure. All individual agents have the same goals, domain knowledge, and possible actions. They also have the same procedure for selecting among their actions. The only differences among individual agents are their sensory inputs and the actual actions they take, or in other words they are situated (placed) differently in the environment.

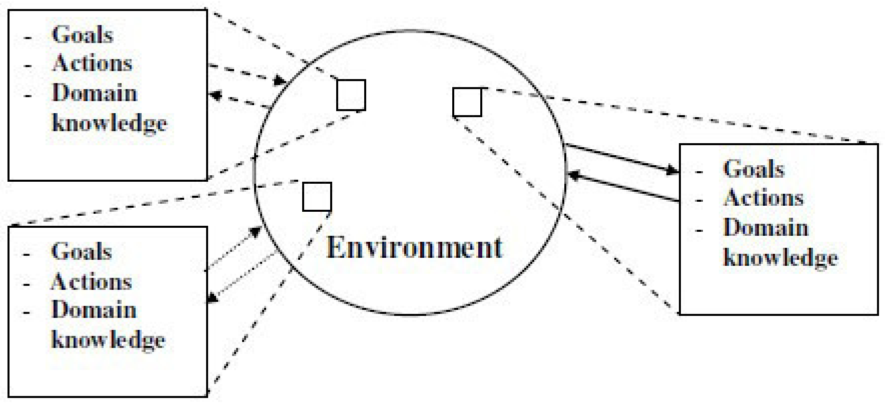


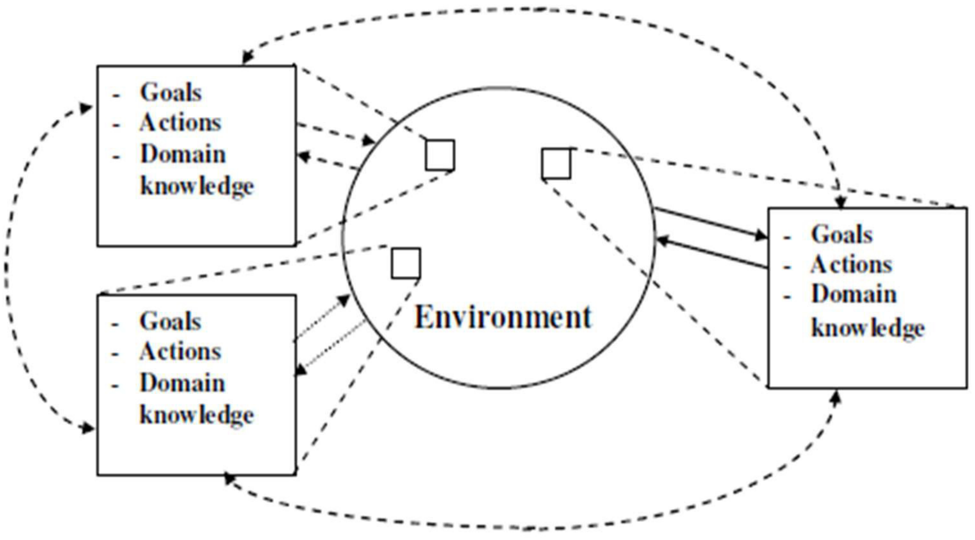
Figure 14. Homogeneous non-communicating MAS

A MAS with homogeneous non-communicating agents is illustrated in Figure 14 (the fact that individual agents have the same goals, actions, and domain knowledge, is indicated in the figure by representing them with identical fonts).
Individual agents in this MAS make their own decisions regarding which actions to take.

Having different actions (outputs) is a necessary condition for MAS since if individual agents all act as a unit, then they are essentially a single agent.
In order to realize this difference in output, homogeneous agents must have different sensory inputs as well. Otherwise they will act identically.

9. Homogeneous communicating MAS

In this MAS individual agents can communicate with each other directly. With the aid of communication, agents can coordinate more effectively. A MAS with homogeneous communicating agents is shown in Figure 15



.

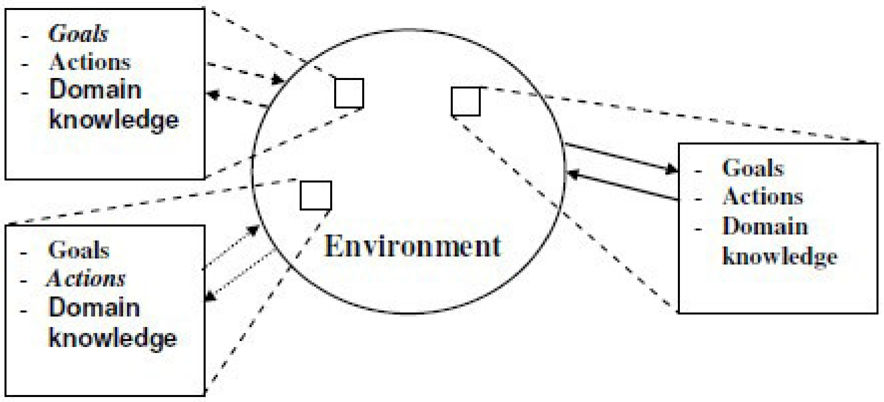
Figure 15. Homogeneous communicating MAS.

Communication raises several issues to be addressed in MAS. One of the most important issues is what the individual agents should communicate. Some possibilities are:
communicate only sensing to each other, share information regarding individual goals, etc. also it is possible for individual agents to learn what and when to communicate with other agents within a MAS based on observed effects of the performances of MAS.

10. Heterogeneous non-communicating MAS

Individual agents within a MAS might be heterogeneous in a number of ways, from having different goals to having different domain knowledge and actions. Adding the possibility of heterogeneous agents in a MAS adds a great deal of potential power at the price of added complexity.

A MAS with heterogeneous agents is illustrated in Figure 16 (the heterogeneity is indicated by different fonts for goals, actions, or domain knowledge)



.

Figure 16. The heterogeneous non-communicating MAS.

Here, the individual agents are also situated differently in the environment which causes them to have different sensory inputs and necessitates their taking different actions. However, in this case the agents have much more significant differences. They may have different goals, actions, and/or domain knowledge.

One of the most important issues to consider when designing a MAS of this type is whether the different agents will be **benevolent** (willing to cooperate) or **competitive**.

Even if the individual agents have different goals, the agents can be benevolent if they are willing to help each other achieve their respective goals. On the other hands, the agents may be selfish and only consider their own goals when acting (competitive).

11. Heterogeneous communicating MAS

This type of MAS is shown in Figure 17. Individual agents (with different goals, actions, and/or domain knowledge) can communicate with one another. This MAS inherits the issues of communicating from homogeneous communicating MAS but heterogeneity brings additional issues to the communication.

Two most important issues are: communication protocols and theories of commitment. Also, the issue of benevolence vs. competitiveness becomes more complicated for this type of MAS.

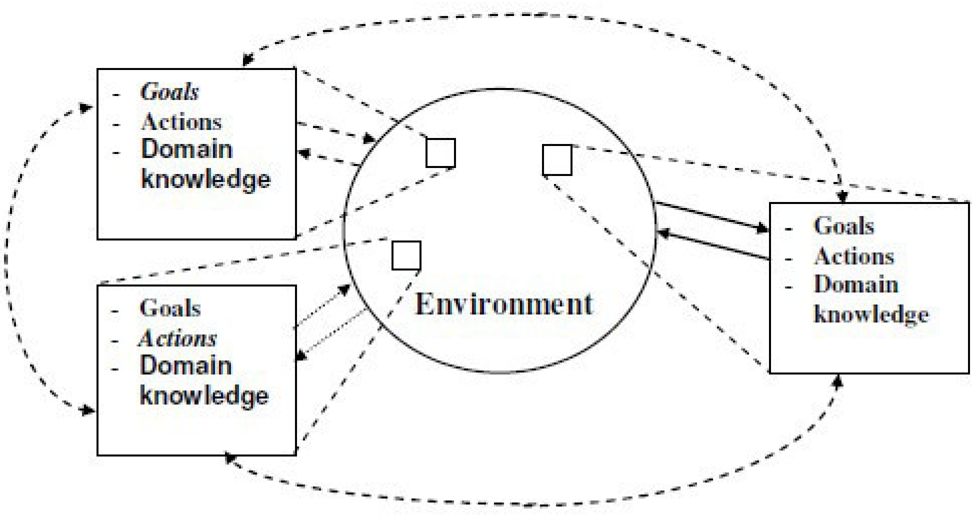


Figure 17. The general heterogeneous communicating MAS.

In all communicating MAS, and particularly in the case of heterogeneous individual agents, there must be some set language and protocol for individual agents to use when communicating.
Independent aspects of protocols are information content, message format, and coordination conventions. Most popular (and widely accepted) existing protocols for these three levels are: KIF (acronym for Knowledge Interchange Format) for content, KQML (acronym for Knowledge Query and Manipulation Language), for message format, and COOL (acronym for COOrdination Language) for coordination.

When agents communicate, they may decide to cooperate on a given task or for a given amount of time. In so doing, they make commitments to each other. Committing to another agent involves agreeing to pursue a given goal, possible in a given manner, regardless of how much it serves individual agent's own interests. There are, in general, three types of commitment: internal commitment – an agent binds itself to do something; social commitment – an agent commits to another agent (term **social agent** is often used); and collective commitment – an agent agrees to fill a certain role (term **team agent** is often in use).

12. Software agents

Software agents can be as simple as subroutines, but typically they are larger entities with some sort of persistent control. Usually, in software engineering, what makes agent-based software different with respect to " ordinary" software is its ability to interoperate and software agents are virtual (software) entities that communicate with their peers by exchanging messages in an expressive agent communication language. Agent-based software engineering was invented to facilitate the creation of this type of software.

This paradigm is often compared to object-oriented programming. Like an " object" , an agent provides a message-based interface independent of its internal data structures and algorithms.

The primary difference between the two approaches lies in the language of the interface. In general object-oriented programming, the meaning of a message can vary from one object to another. In agent-based software engineering, agents use a common language with an agentindependent semantics.

The concept of agent-based software raises a number of questions and most important ones are,

1. What is an appropriate agent communication language?
2. What communication " architectures" are conductive to cooperation?

Before addressing above questions let us first address very important issue, particularly from power engineering (but also other engineering disciplines) standpoint and that is the issue of existing (legacy) software. There is a vast panel of developed software capable to help solve many engineering problems. These tools are developed using procedural or object-oriented paradigms, they implement best algorithms and were (or still are) programmed using best programming skills in these paradigms. Their validity and usefulness is confirmed through many years of their use in practice. No one can discard any software tool just because it is not programmed using agent-oriented paradigm, and legacy software is to be agentified if one is interested in using them as a part of a MAS. Three possible approaches to agentification of legacy software are illustrated in Figure 18.

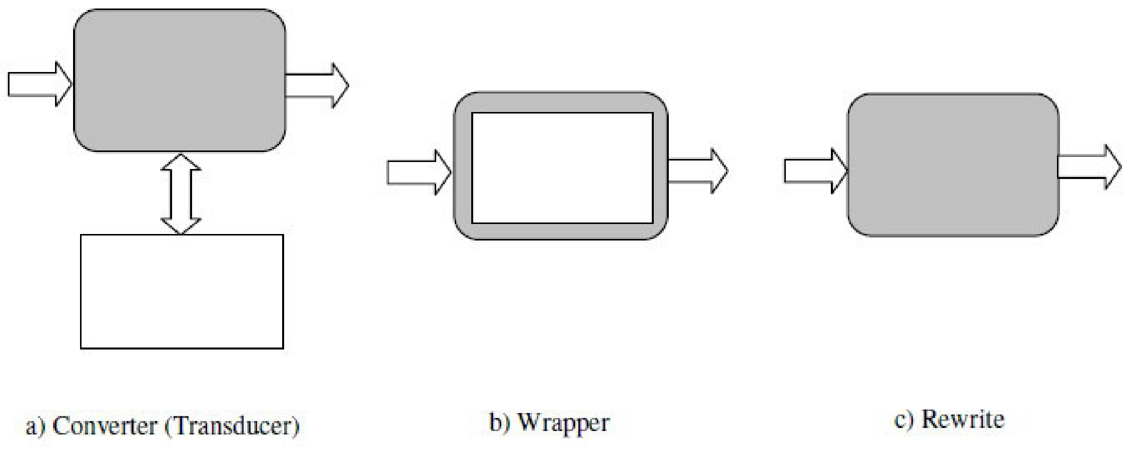


Figure 18. Three approaches to agentification

One approach (Figure 18a) is to implement a converter (sometimes termed transducer) that mediates between an existing program and other agents within a MAS. The converter accepts messages from other agents, translates them into the program's " understandable"
variables or values, and passes those to the program, and of course vise versa, it accepts the program's response, translates into agent communication language and sends the resulting messages to other agents. This approach has the advantage that it requires no knowledge of the program itself, only its communication behavior (inputs and outputs). It is, therefore, especially useful for the situations in which the code for the program is unavailable or too
delicate to modify.

A second approach to dealing with legacy software (Figure 18b) is to implement a wrapper, i.e. inject a new piece of the code into a program to allow it to communicate in agent communication language. The wrapper can directly examine the data structures of the program and can modify those data structures. Furthermore, it may be possible to inject calls out of the program so that it can take advantage of externally available information and services.

This approach has the advantage of greater efficiency than the converter approach since there is less serial communication. It also works for cases where there is no interprocess communication ability in the original program. However, it requires that the code for the program be available.

The third approach (Figure 18c) is a drastic one and requires the original program to be rewritten in order to comply with new requirements. From engineering standpoint this is probably last resort and will not be further elaborated in this note.

13. Agent communication language

Communication language standards facilitate the creation of interoperable software by decoupling implementation from interface. As long as programs are committed to the details of the standards, it does not matter how they are implemented. Today, standards exist for a wide variety of domains. For example, electronic mail programs from different vendors manage to interoperate through the use of mail standards like SMTP.
Agent communication language (as any other standard) should assure consistency (the same words or expressions must have same meaning for all programs) and compatibility (all programs must use same words and expressions to “ say” the same things).

There are two popular approaches to the design of agent communication language: the procedural approach and the declarative approach. The procedural approach is based on the idea that communication can be best modeled as the exchange of procedural directives. Scripting languages (such as Perl and TCL) are based on this approach. They are both simple and powerful. They allow programs to transmit not only individual commands but entire programs. They are also (usually) directly and efficiently executable.

Unfortunately, there are disadvantages to purely procedural languages and they are: sometimes the procedures require information about the recipient that may not be available to the sender, procedures are unidirectional (for agents the information share usually should be usable in both directions), scripts are difficult to merge.

The declarative approach to language design is based on the idea that communication can be best modeled as the exchange of declarative statements (definitions, assumptions, etc). To be maximally useful, a declarative language must be sufficiently expressive to communicate information of widely varying sorts (including procedures).

At the same time, the language must be reasonably compact, it must ensure that communication is possible without excessive growth over specialized languages. In agent-based systems the use of declarative communication languages is paradigm to prevail but in short-term procedural approach will still be in rather wide use.

A declarative agent communication language 9sometimes called ACL) can best be thought of as consisting of three parts: its vocabulary (vocabulary implements what is known in agent literature as ontology, but the reader should be careful since one vocabulary can implement more than one ontology), an “ inner” language called KIF, and an “ outer” language called

KQML. An agent communication language message is a KQML expression in which the " arguments" are terms or sentences in KIF formed from words in the vocabulary.

The vocabulary is listed in a large and open-ended dictionary of words appropriate to common application areas . Each word in the dictionary has an English description for use by humans in understanding the meaning of the word, and each word has formal annotations(written in KIF) fro use by programs.

Note that the existence of such a dictionary does not imply that there is only one way of describing an application area. Indeed, the dictionary can contain multiple ontologies for any given area. An example of multiple ontologies is description three-dimensional geometry in terms of polar coordinates, rectangular coordinates, cylindrical coordinates, etc.

14. Architectures for software MAS

In a software MAS one of the issues is how individual agents should be organized to enhance cooperation Two very different approaches have been explored: direct communication, in which agents handle their own coordination) and assisted coordination, in which agents rely on special system programs to achieve coordination.

The advantage of direct communication is that it does not rely on the existence, capabilities, or biases of any other programs. Two popular architectures for direct communication are the contract-net approach and specification sharing[19]In the contract-net approach (also termed market-based organization), agents in need of service distribute requests for proposals to other agents.

The recipients of these messages evaluate those requests and submit bids to the originating agents. The originators use these bids to decide which agents to task and then award contract to those agents.

In the specification sharing approach, agents supply other agents with information about their capabilities and needs, and these agents can then use this information to coordinate their activities. The specification sharing approach is often more efficient than contract-net approach because it decreases the amount of communication that must take place.

One disadvantage of direct communication is cost. As long as the number of agents is small, this is not a problem. But, in a setting with huge number of programs the cost of broadcasting the bids or specifications and consequential processing of those messages is prohibitive. In this case, the only alternative is to organize the agents in some way that avoids such broadcast. Another disadvantage is implementation complexity. In the direct communication schemes, each agent is responsible for negotiating with other agents and must contain all of the code necessary to support this negotiation. If only these capabilities could be provided by the system, this would lessen the complexity of application programs[20] A popular alternative to direct communication that eliminates both above mentioned disadvantages is to organize agents into what is often called a federated system. A simple structure of such a system is illustrated in Figure 19.

As suggested by the diagram, agents do not communicate directly but they communicate only with system programs called facilitators (or sometimes termed mediator, but it could be claimed that the facilitator is generalization of the concept of mediator).

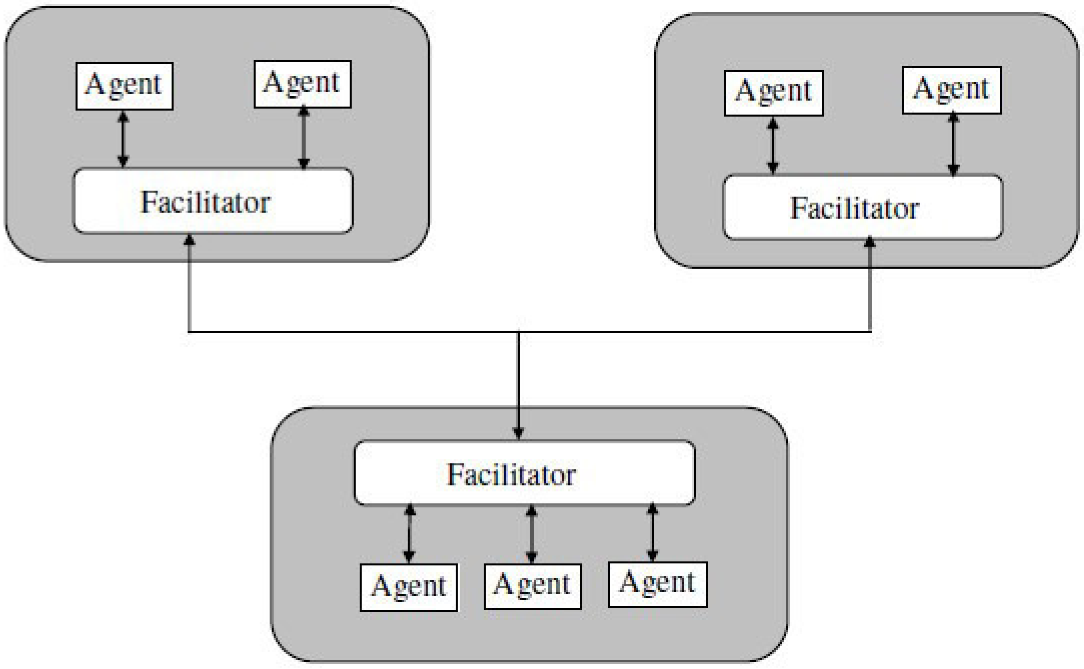


Figure 19. Federated system

In this system, agents use agent communication language (in practice, a restricted set of this language) to document their needs and abilities for their local facilitators. In addition to this they also send application-level information and request to their facilitators and accept application-level information and requests in return. Facilitators use the documentation provided by these agents to transform these application-level messages and route them to the appropriate place. In effect, the agents form a " federation" in which they surrender their autonomy to their facilitators and the facilitators take the responsibility for fulfilling their needs.

15. Other organizational paradigms

In addition to the organizational paradigms considered in previous section, several other paradigms emerged for MAS such as:

- Hierarchical organization,
- Coalitions,
- Teams,
- Congregations,
- Societies, and
- Compound organizations.

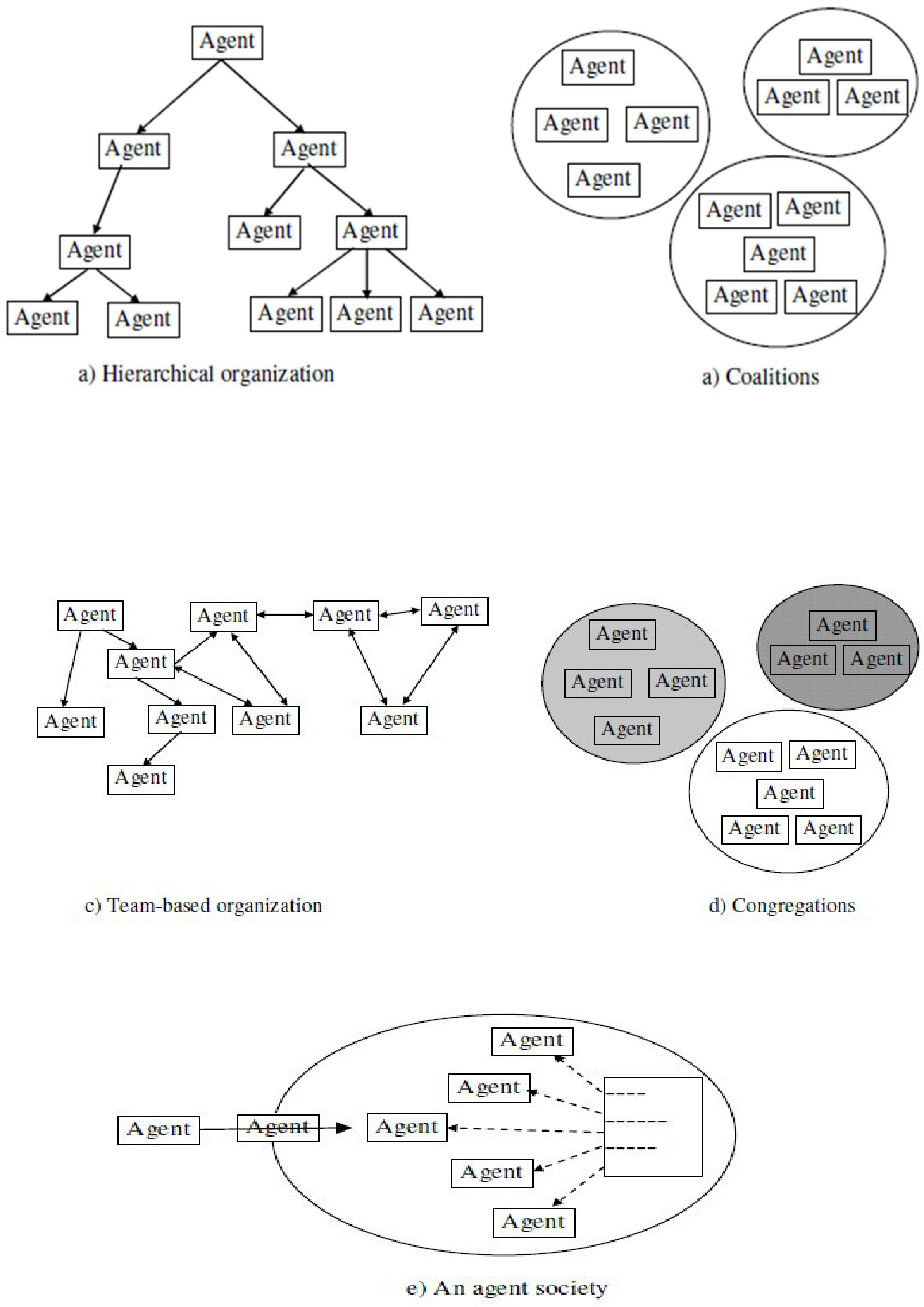


Figure 20. Some other organizational paradigms of agents within a MAS

In hierarchical organization individual agents are organized in a tree-like structure, where agents higher in the tree have a more global view than those below them. Interactions do not take place across the tree. The data produced by lower-level agents in a hierarchy typically travels upwards to provide a broader view, while control flows downward. In coalition type of organization the agents are grouped according to their goals into the groups of the agents having similar goals. Within a coalition (Figure 20b), the organizational structure is typically flat, although there may be a distinguished " leading agent" which acts as a representative for the group.

Once formed, coalitions may be treated as single entity. The agents in one group (coalition) are expected to coordinate their activities in a manner appropriate to the coalition' s purpose.

Coordination does not take place among agents in separate coalitions, except to the degree that their individual goals interact.

A team of agents (Figure 20c) consists of a number of cooperative agents which have " agreed" to work together toward a common goal. In comparison with coalitions, teams attempt to maximize the utility of the team itself, rather than that of the individual members.

Agents are expected to coordinate in some fashion such that their individual actions are consistent with and supportive of the team' s goal. Usually individual a=gents takes a role in achieving common goal.

Similar to coalitions and teams, agent congregations are groups of individual agents banded together into a typically flat organization in order to derive additional benefits. Congregations (Figure 20d) are formed among agents with similar or complementary characteristics. Different shadings in Figure 19d represent the potentially heterogeneous purpose behind each grouping, in comparison to typically homogeneous coalitions. Here, individual agents do not necessarily have a single or fixed goal. Analogous everyday examples of this type are: clubs, support groups, academic departments, etc.

Unlike other organizational paradigms, agent societies are inherently open systems.

Agents have different goals and heterogeneous capabilities. In this paradigm there is a set of constraints that are imposed over the individual agents (Figure 21e), and these constraints are known as social laws, norms or conventions. The norms are in fact the set of rules or guidelines by which the agents must act, which provides a level of consistency of behaviors.

For example the rules might constrain the type of protocols the agents use to communicate. Compound organizational structures combine some of the above mentioned paradigms making the use of best characteristics of different organizational paradigms.

16.Conclusion

As with many other engineering fields, the power system research communities and practitioners are getting more and more interested in using these technologies to solve different kind of problems. This note will not elaborate individually all considerations and rather points out some publications where power system problems are tackled or solved using agent or multi-agent technologies.

1- Introduction

Process modeling has many notions in common with agents-oriented programming: It serves as a high-level abstraction for distributed systems composed of many cooperative or competing actors, communicating via messages and services, and reacting to events. Thus, it is not surprising that process modeling has been adopted for the modeling of multi-agent systems in a number of works.

In this chapter highlighted the importance of combining performance engineering with agent oriented design methodologies in order to develop large agent based applications to derive process performance measures, we need a quantitative process analysis technique. Process simulation appears to be a prominent technique that allows us to derive such measures (cycle time) given data about the activities (processing times) and data about the resources involved in the process through process simulation an engineer can forecast the process execution time, identify possible bottlenecks and perform tests regarding the response of the process to increasing demand. Process simulation is a versatile technique supported by a range of process modeling and analysis tools.

2- Motivation

While not as common as train accidents caused by track malfunctions and human error, mechanical failure does play a large role in train accidents and derailments. Train accidents caused by mechanical failure account for over 10 percent of the total number of accidents. Trains must be regularly serviced to stay in working condition, and the failure to notice an obvious issue with the train performance could lead to a devastating accident in the future

The motivation behind this study is to overcome the problem associated with current human-based practices for creation of a plan. Making a reasonable and quick decision in such information-rich, dynamic environments is essential to this kind of problem. Recently, a novel approach called the multi-agent system (MAS) is being increasingly used in transportation applications. Literature has shown that the MAS are suitable to solve a communication and coordination problem in a distributed environment. Presently, the MAS is often combined with a well-designed ontology model capturing the domain knowledge to strengthen its reasoning capability, in order to help project stakeholders collaborate in the decision-making process

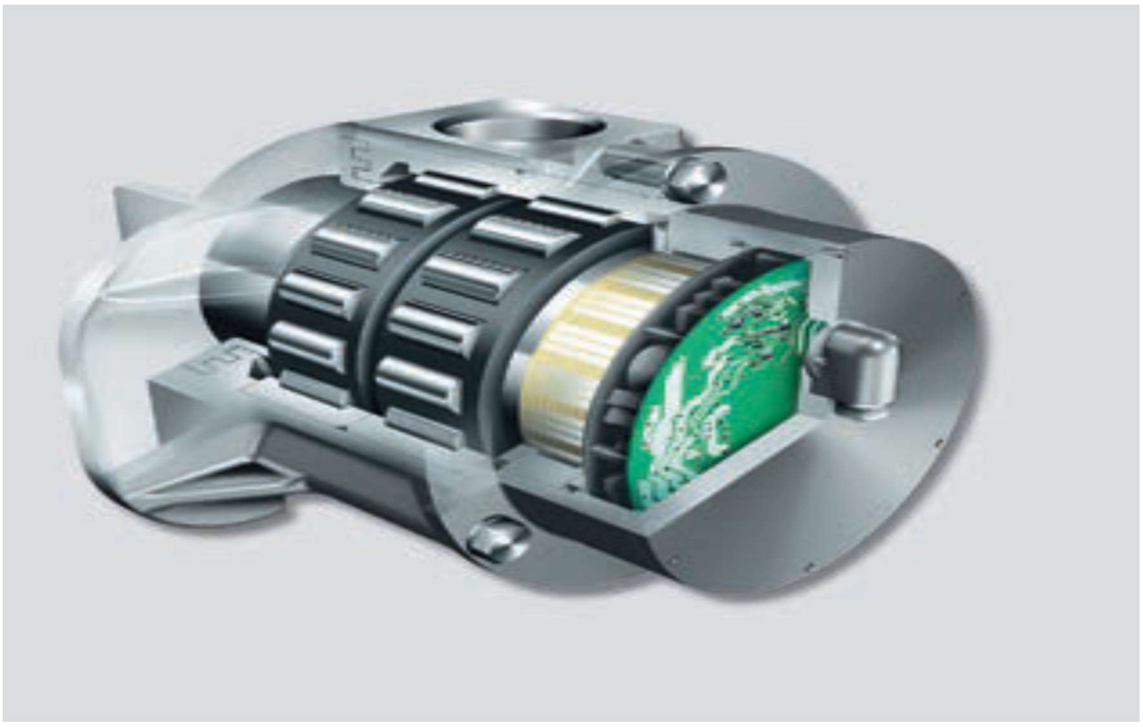


Figure 21: One of the most important parts of the train

3- Overall design

This MAS model consists of four intelligent agents: Train Agent, Communication Agent, Analysis Agent, and RUL Agent. Contains a set of inference rules to represent the knowledge pertaining to RUL

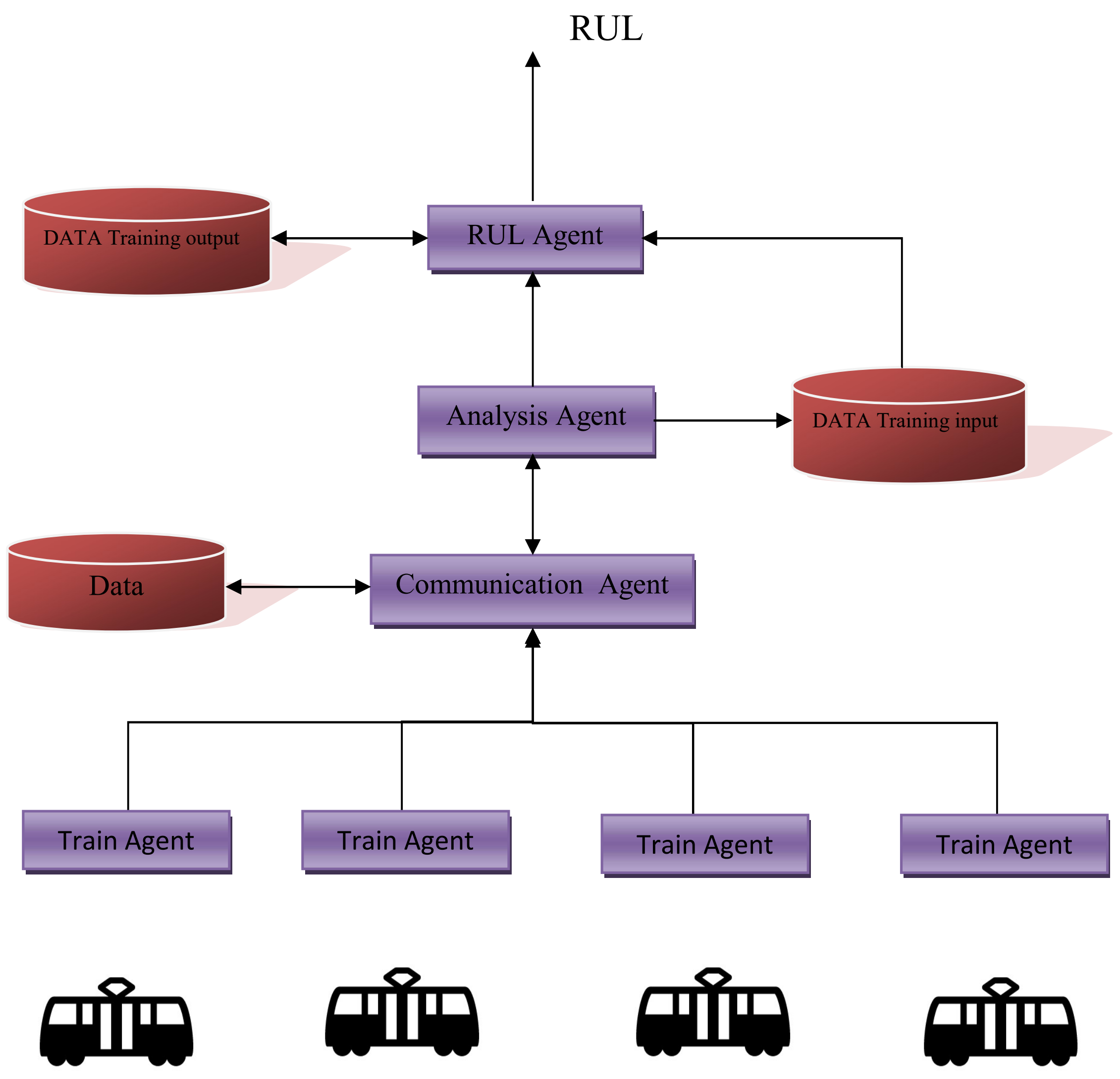


Figure 22: BDI architecture

As shown in the form the train agent sends the values of the sensors on the train the Communication Agent receives all that information and stores it in databases to be later graduated by Analysis Agent the RUL agent receives an order to calculate the RUL value that was previously trained we note the sequence in which the functions of the Cole agent are performed where tasks that have the same attributes are distributed to the agent to be a consistent action

4- Partial design

4-1- Definition

It is important to determine how sophisticated the agents' reasoning will be. Reactive agents simply retrieve pre-set behaviors similar to reflexes without maintaining any internal state. On the other hand, deliberative agents behave more like they are thinking, by searching through a space of behaviors, maintaining internal state, and predicting the effects of actions. Although the line between reactive and deliberative agents can be somewhat blurry, an agent with no internal state is certainly reactive, and one which bases its actions on the predicted actions of other agents is deliberative. Here we describe one system at each extreme as well as two others that mix reactive and deliberative reasoning

Train Agent: reactive agent architecture based on the following theses: Intelligent behavior can be created without clarification in the train agent Representations of the kind that Amnesty International's proposals are symbolic Intelligent behavior can be generated without a clear summary I send sensor values such as (heat and vibration) The number of agents of this type is the same number of trains and has actual income

Communication Agent: cognitive agent includes infrastructure features and services for enabling inter-server communication. Communication Agent provides the connection management, reliable routing services and software compatibility management, and supports mechanisms for exchange of Stack Events between stacks hosted on different Message Processors. Communication Agent successfully routes messages between layers across processes and servers.

Analysis Agent: cognitive agent Existing methodologies for sensitivity analysis focus on end-point behavior, which is insufficient for Analysis Agent, as they are explicitly time-dynamic. Other methodologies that do consider time-dependence are tailored for the analysis of data display several features, in particular the adaptability of agents and a system openness, that need to be taken into account in model analysis

RUL Agent: cognitive agent when we expand our environments we get a larger and larger amount of tasks, eventually we are going to have a very large number of actions to pre-define. Another way of going about creating an agent is to get it to learn new actions as it goes about its business, this still requires some initial knowledge but cuts down on the programming greatly. This will allow the agent to crating new RUL

4-2- System communication

Agents can interact either by performing language actions (communicating with each other) or by performing non-linguistic actions that modify their environment by communicating, agents can exchange information and coordinate their activities in MAS two main strategies have been used to support agent-to-agent communication: agents can exchange messages directly or they can access a shared database (called a blackboard or "blackboard") in which information is posted. Communications are at the root of the interactions and social organization of MAS:

4-2-1-Direct method:

There are many auxiliary parts to a message in addition to the content, for example: the intended recipients, the sender and the message type it is essential for the message as a whole to respect a common format, in this MAS, messages adhere strictly to the ACL (Agent Communication Language) standard which allows several possibilities for the encoding of the actual content.

4-2-2-Indirect method:

In artificial intelligence technology, Blackboard is widely used to identify shared memory by different systems. But we turned the idea to a different idea, so we put the file to be an indirect means of sharing information. Be tween Analysais Agent and RUL Agent

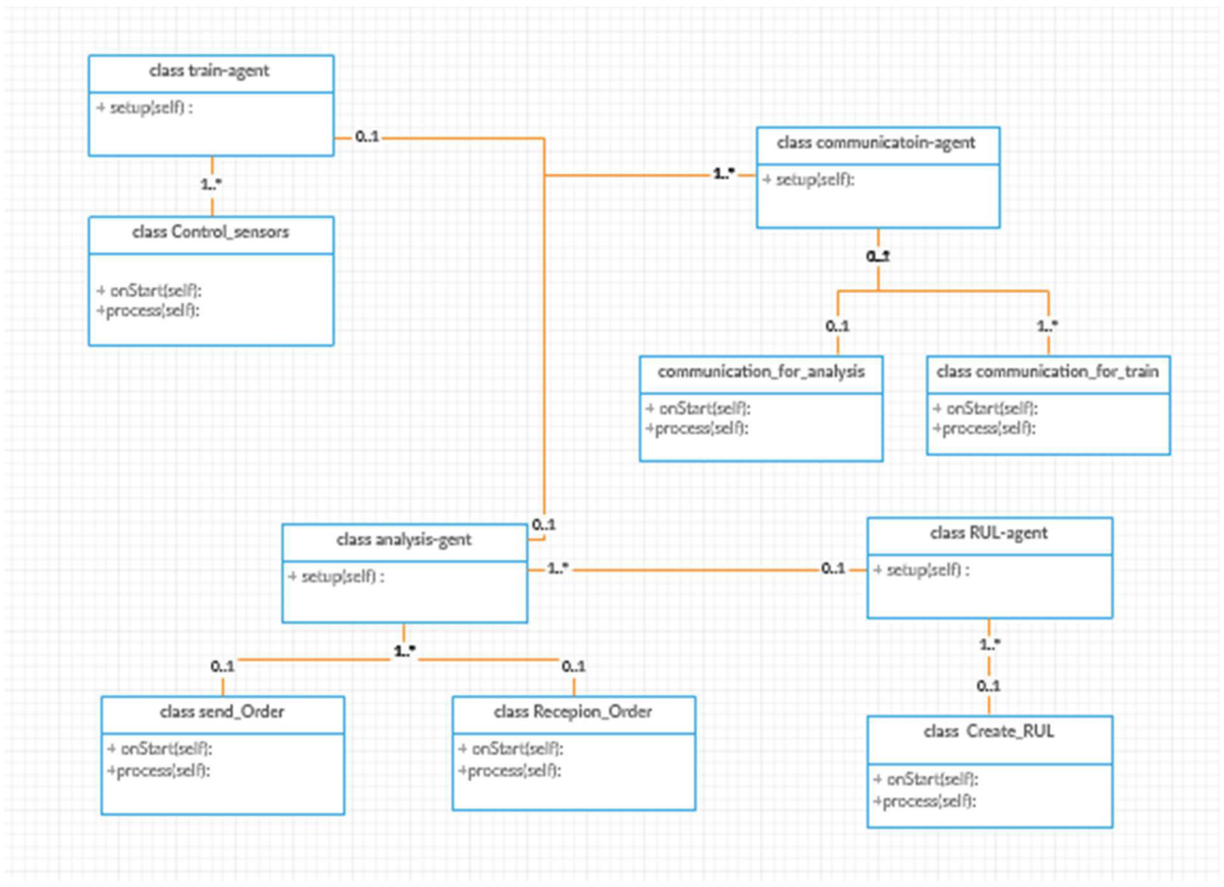


Figure 23: class diagram

The ability of our agent modeling technique and architecture to support physical agents that are themselves multi-agent systems highlights a particular property of the architecture its ability to straightforwardly implement more specialized architectures for example, consider a layered agent architecture, Suppose the state and behavior of each of these layers is specified as a separate logical agent class. There are then two distinct ways to combine them in our framework, as illustrated in Figure 23

Conclusion

Today's PHM solutions are still very personalized and limited to a single application. Different applications would have different data acquisition and storage systems, different characteristics depending on domain knowledge and different monitoring objectives. It is very difficult to create a platform that could cover all kinds of applications in the next chapter we will talk about tools and methods followed in programming

1- Introduction

Computer science has been and, more than ever, it is still a discipline characterized by a rapid evolution, which enforces software developers and software houses to carry out radical changes, involving mainly the conceptual and methodological point of view. The agent-oriented paradigm can be considered as the result of this evolving process, due to the ever-increasing software complexity. Nowadays, software agents are undoubtedly more than a promising approach to complex software development, during the last twenty years a lot of research work has been undertaken, ranging from theoretical foundations to concrete tools and technologies and application case studies But in spite of this effort the proposed modeling languages and methodologies still remain incomplete or at least they are not able to fully capture and/or communicate the underlying meaning and complexity of multi-agent systems (MAS) . In particular, up to now, the study of issues connected with the PHM implementation phase has not been stressed enough.

Over the past few Months we have been directing our research towards the development of a theoretical and practical methodology for designing, implementing and testing MAS as part of this effort, which aims at simplifying the PHM designer’s work, increasing the reuse of code through a database of agents/tasks patterns, and a testing framework, which provides a coordinated and automated approach to the testing of MAS, have been implemented. The aim of this chapter is to give an overview of the work done so far, focusing in particular on the latest results achieved.

2-What is Python?

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid

Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

3-What is spade ?

Smart Python multi-Agent Development Simply put, SPADE is an agent platform based on the XMPP/Jabber technology. This technology offers by itself many features and facilities that ease the construction of MAS, such as an existing communication channel, the concepts of users (agents) and servers (platforms) and an extensible communication protocol based on XML, just like FIPA-ACL. Many other agent platforms exist, but SPADE is the first to base its roots on the XMPP technology.

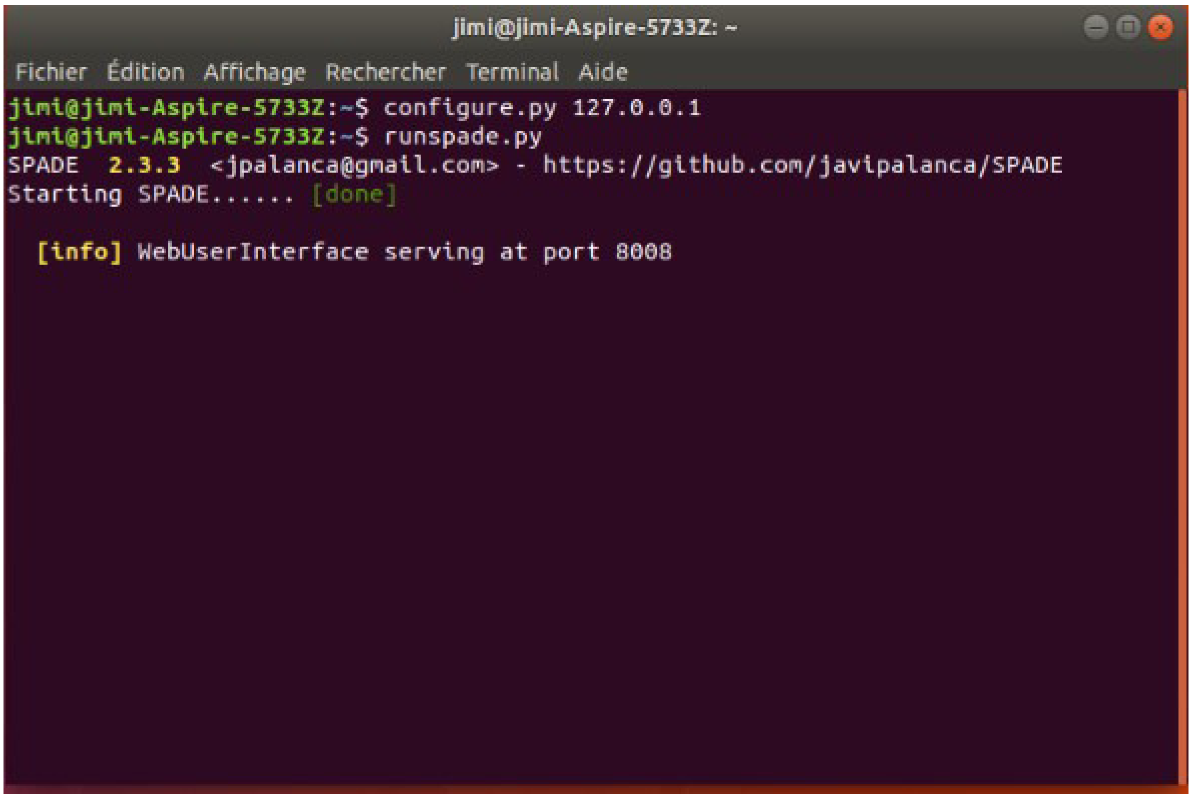
The SPADE Agent Platform does not require (but strongly recommends) the operation of agents made with the SPADE Agent Library (see next section). The platform itself uses the library to empower its internals, but aside from that, you can develop your own agents in the programming language of your choice and use them with SPADE. The only requirement those agents must fulfill is to be able to communicate through the XMPP protocol. The FIPA-ACL messages will be embedded in XMPP messages. Be warned, however, that some features of the whole SPADE experience may not be available if you do not use the SPADE Agent Library to build your agents SPADE is written in the Python programming language. In order to fully understand and use SPADE, a bit of knowledge about Python is required,

The SPADE Agent Library is a module for the Python programming language for building SPADE agents. It is a collection of classes, functions and tools for creating new SPADE agents that can work with the SPADE Agent Platform. Using it is the best way to start developing your own SPADE agents. In the future, we would like to offer bindings of the SPADE Agent Library for more programming languages, like Java or C#, but for now only Python is supported.

4- Why SPADE ?

- Developed using Python
- Covers the FIPA standard
- Implements four MTPs: XMPP, P2P, HTTP and SIMBA
- Supports two content languages: FIPA-SL and RDF
- SPADE agents do reach their goals by running deliberative and reactive tasks
- Has a web interface to manage the platform
- Allows its execution in several platforms and operating systems
- Presence notification allows the system to determine the current state of the agents in real-time
- Multi-user conference allows agents to create organizations and groups of agents

Once you have succesfully installed SPADE run the **configure.py** script in order to configure the platform. You can provide a parameter to this script, which is the machine's hostname and that will be used as the platform's name. Finally, launch the **runspade.py** script and see your platform rise before your eyes. After a few seconds you should see the green word Done on the screen confirming that the platform was succesfully launched. In a nutshell:


jimi@jimi-Aspire-5733Z: ~
Fichier Édition Affichage Rechercher Terminal Aide

```
jimi@jimi-Aspire-5733Z:~$ configure.py 127.0.0.1
jimi@jimi-Aspire-5733Z:~$ runspade.py
SPADE  2.3.3  <jpalanca@gmail.com> - https://github.com/javipalanca/SPADE
Starting SPADE...... [done]

  [info] WebUserInterface serving at port 8008
```

Figure 24 :SPADE

5-Tools

5-1-Bearing Data

Experiments on bearings' accelerated life tests provided by FEMTO-ST Institute, Besançon, France.

In order to test and verify the prognostic methods developed and published in the literature, dedicated test beds and platforms have been designed and realized by several laboratories over the world. Most of these experimental systems concern specific physical components, such as bearings, gears, pumps, etc. The following paragraphs summarize the experimental platforms which are yet published. Note that not all of the published test beds provide experiment data for external users, but only some of them

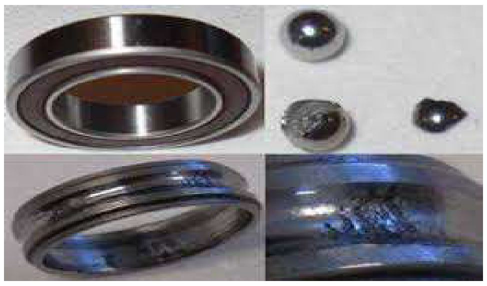


Figure 25 :Normal and degraded bearing

5-2-Overview of test beds

The following paragraphs present an overview of experimental platforms reported in the literature and related to critical physical components such as gearboxes, pumps, pinions ,etc.

A test bed related to a gearbox and a pinion gear has been used in In this application, a spiral bevel pinion was seeded with two electrical discharge machine notches (heel and toe) on the drive side of one of the pinion gear teeth to artificially accelerate tooth root cracking. Several accelerometers were then placed on the gearbox with a health and usage monitoring system used to generate the vibration features In the case of machining tools, a milling data set experiments related to milling machine for different speeds, feeds and depth of cut can be found In the same way, an experimental platform has been developed by SIMTech of Singapore to provide data during the PHM challenge organized in 2010 by the PHM society.

In the authors used an experimental setup related to drilling life tests to verify their method. The tests were conducted on a MAHO 700S machine, which is a computer numerical controlled five axis machining center, with movement in three perpendicular axes and a rotary/tilt table. Finally, in a method is proposed to estimate the tool wear of a turning process over a wide range of cutting conditions. The developments were validated thanks to experiments conducted on a conventional lathe TUD-50 Concerning the pumps, an experimental setup has been used to evaluate the performance of a developed Hidden Semi-Markov Model method for equipment health prognostic. The experimental setup consisted in a real hydraulic pump During the experiments, long-term wear test experiments were conducted at a research laboratory facility. Three pumps were then worn by running them using oil containing dust. Finally, data set experiments related to charging and discharging of Li-Ion batteries can be found The records concern the impedance as the damage criterion and the data set was provided by the Prognostics Center of Excellence at NASA. The same center proposed preliminary data from thermal overstress accelerated aging for six devices. Note that this overview is obviously not exhaustive, but enables to see that real systems are required to test and verify PHM algorithms. Also, the variety of presented applications reveals that most PHM tools are application-based. Thus, further developments to face this aspect are still required. Among the works of the PHM field, bearings failures analysis benefits

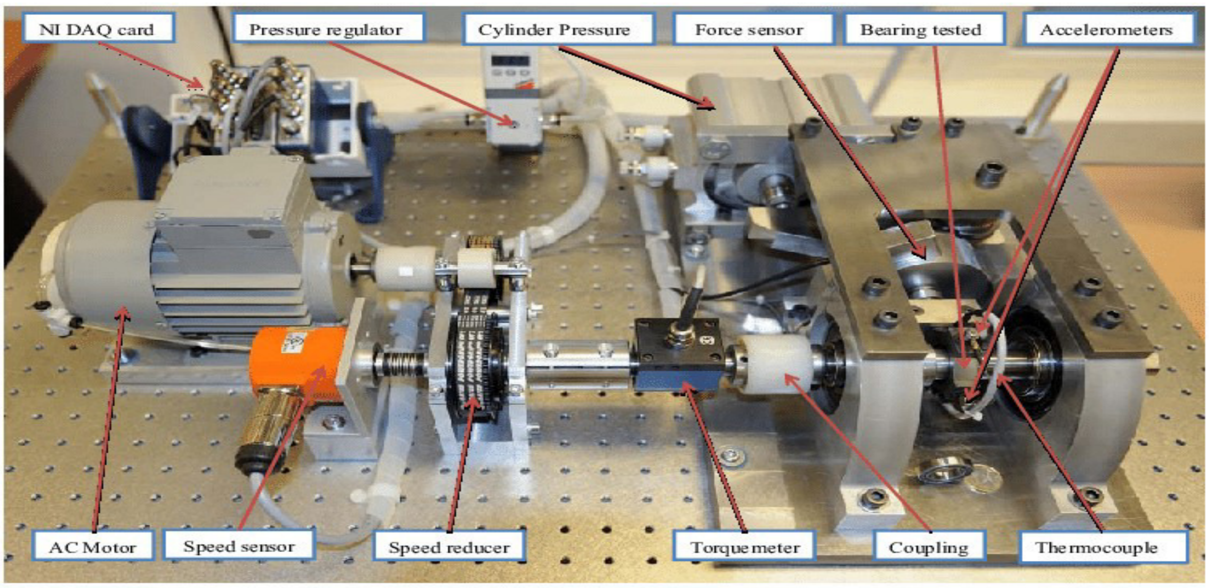
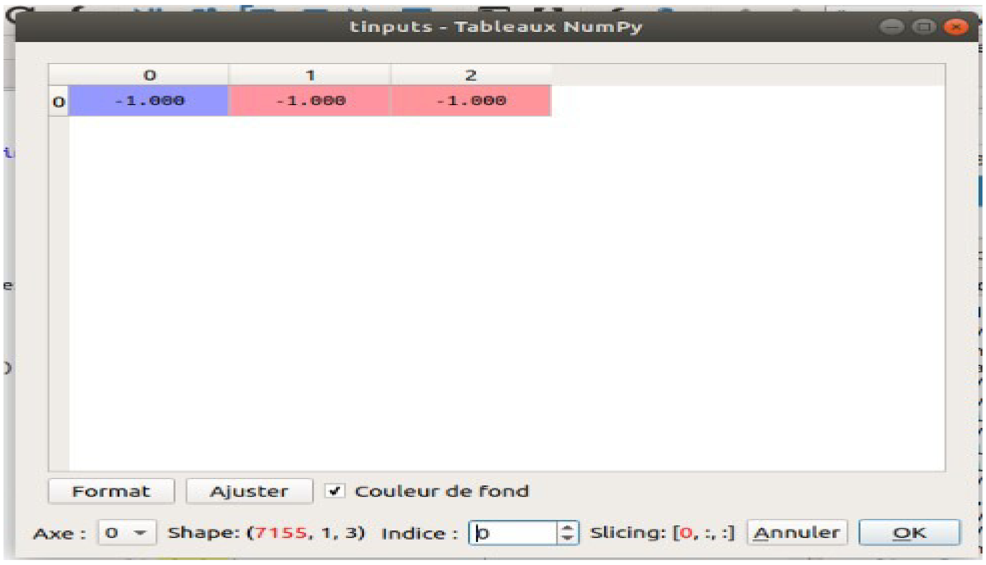


Figure 26 TUD-50

It is these experiences we have provided two types of databases,

- Vertical and horizontal vibration, gives us 2560 value in seconds and runs every ten seconds
- 600 value of the temperature every minute

After the consolidation of the databases on a consolidated database in the language of seven million valuable, but we have to reduce them, to become in the following form



	0	1	2
0	-1.000	-1.000	-1.000

Axe : 0 Shape: (7155, 1, 3) Indice : 0 Slicing: [0, :, :]

Figure 27 data in applicatoin

5-3-What is SQLite ?

SQLite is a software library that provides a relational database management system. The lite in SQLite means light weight in terms of setup, database administration, and required resource.

5-4-What is Keras?

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

5-5-What is Spyder?

Spyder is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates NumPy, SciPy, Matplotlib and IPython, as well as other open source software It is released under the MIT license

6-The parts in the code

With this sentence, the SPADE Agent Library is imported and all its features become available. Let's start defining the base class for the agent:

```
class MyAgent(spade.Agent.Agent):
        def _setup(self):
                print "MyAgent starting . . ."
```

As you can see, we have derived a class from spade.Agent.Agent, which is the base class for all SPADE normal agents. Also, note that we have defined a method called _setup() . Every SPADE agent should override this method. It is where the initialization (or setup) code of the agent must be placed.

```
if __name__ == "__main__":
        a = MyAgent("agent@127.0.0.1", "secret")
        a.start()
```

one that uses an actual behavior to perform a task. As we stated earlier, the real programming of the SPADE agents is done mostly in the behaviors.

```
b=self.Control_sensors()
self.addBehaviour(b, None)
```

7-Artificial neural networks

Artificial neural networks (ANNs) or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the results to identify cats in other images. They do this without any prior knowledge about cats, e.g., that they have fur, tails, whiskers and cat-like faces. Instead, they automatically generate identifying characteristics from the learning material that they process.

An ANN is based on a collection of connected units or nodes called artificial neurons which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it.

In common ANN implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The connections between artificial neurons are called 'edges'. Artificial neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

The original goal of the ANN approach was to solve problems in the same way that a human brain would. However, over time, attention moved to performing specific tasks, leading to deviations from biology. ANNs have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, We use it in this program to value RUL

8-Application

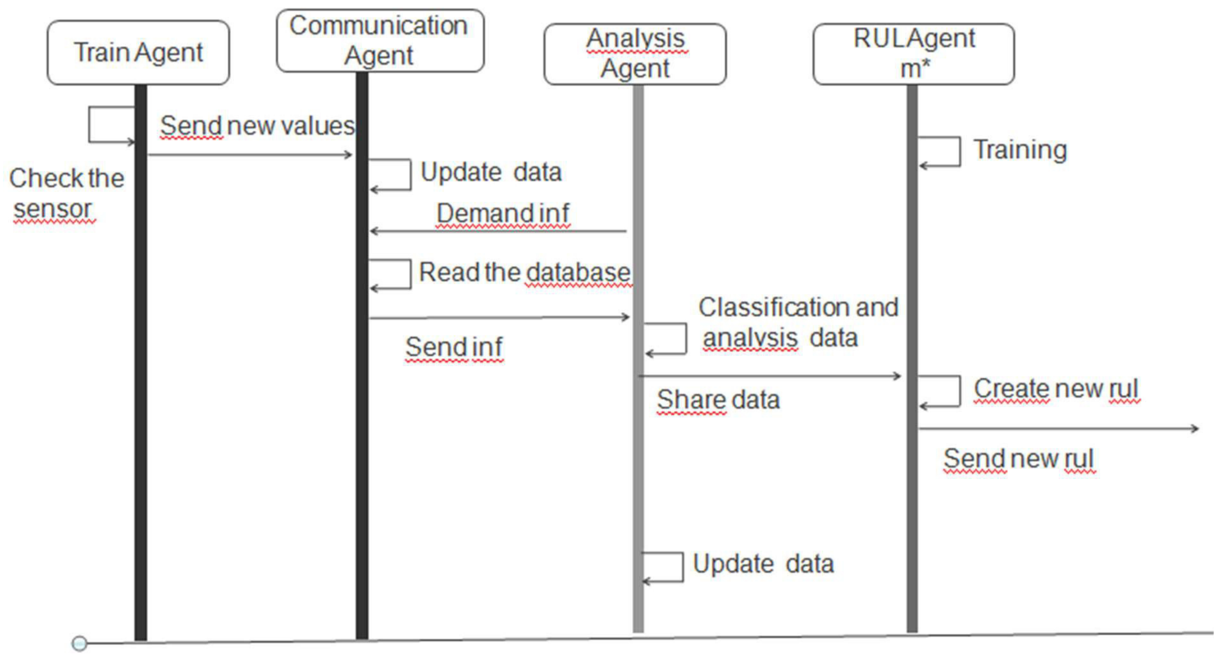


Figure 28: sequence diagram

This representation is a way of communication and the work of the program in general note that the first agent is the one who gives the data followed by a communication agent who updates the data after each information. In exchange for an analysis agent, information is periodically requested to be sent to an agent specialized in calculating the value of the RUL

The result shows the time remaining for this piece

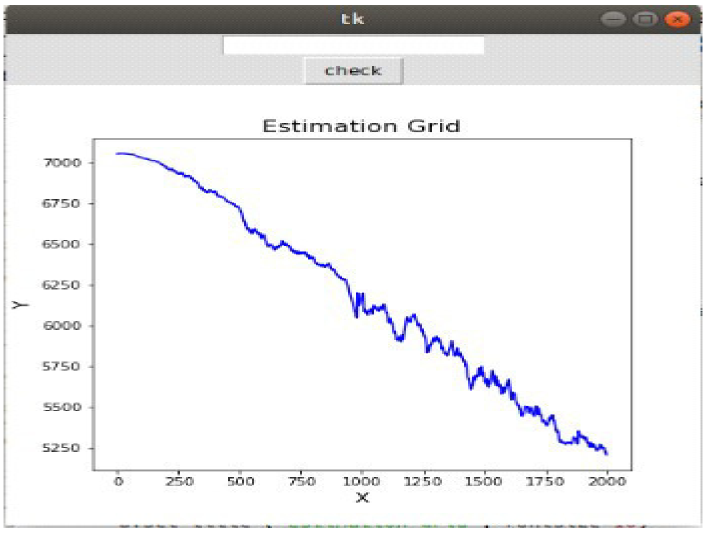


Figure 29 : RUL

The axis of the hierarchy represents a change in the value of RUL The comma axis represents a time value change From which we observe each passing period of time less than RUL

9-Conclusion

The aim of the chapter is to cope with important issues connected to the transition from the design phase to the implementation phase, with particular attention to testing and diagrammatic notations supporting the development process when this is very close to the actual implementation of MAS.

At first glance the work presented in this chapter and the previous works, of which the present paper represents an evolution, may seem composed of loosely connected parts in reality this is due to the nature of the problem. As a matter of fact moving inside the implementation phase means solving several issues connected with the refinement of the design models in order to facilitate code production, with the deployment of the system, with the necessity of reducing the prototype implementation time and finally with the testing activities.

We think that the robust and easy-to-use test tool and the diagrammatic notations proposed, together with a SPADE tool supporting them, a library of re-usable code component, and a notation supporting the representation of the deployment of MAS, are a significant step towards the widespread acceptance of the agent software paradigm. But, we are well aware that there is more. As a matter of fact more work needs to be done; other issues should be tackled.

General Conclusion

Prognostics is quickly evolving, but still needs more attention from governments, industries, and academia to become less of an art and more of a science. This could be done if all efforts in this field are integrated together to obtain a clear and definite steps for prognostics system design, development, validation, and verification.

In this work , we tried to present a complete vision about prognostics as a major component part of maintenance. We gathered a lot of sparse information about prognostics and combined all of these information together to present an integrated work that shows the importance of prognostics and its influencing rule in maintenance.

The primary contribution of this work has been to provide the elements of a rigorous framework for modeling and specifying complex multi-agent systems. We have presented modeling techniques to describe the external and internal perspective of multi-agent systems, based on a PHM architecture, which build upon and adapt existing, well understood object-oriented models our agent methodology, with its emphasis on roles, responsibilities, services, and goals, permits a engrained analysis that allows agent boundaries to be chosen exile and results in system designs that are robust, modular, and extensible.

We have given a semantics for inheritance, aggregation and instantiation relationships amongst agent classes and instances which provides powerful and exile mechanisms for enforcing modularity of state and behavior within agents, and for sharing them between agents. Related beliefs, goals, and plans may be encapsulated in separate classes which may then be grouped together, by aggregation or inheritance.

The ability to take an agent class and renew it by the addition of further beliefs, goals, or plans provides a compositional framework for system design and encourages re-use. Encapsulation makes more tractable the task of analyzing interactions between plans, which is crucial to the process of design variation.

By building upon and adapting existing, well-understood techniques, we aim to take advantage of their maturity to develop models and a methodology which will be easily learnt and understood by those familiar with the PHM paradigm. This is important if the design, implementation, and maintenance of multi-agent systems is to be carried out by software analysts and engineers rather than research scientists, and if they are to be successfully applied on a sign cant scale to commercial and industrial applications.

References

1-NASA. 2000. *Reliability Centered Maintenance Guide for Facilities and Collateral Equipment*. National Aeronautics and Space Administration, Washington, D.C.

2-Piotrowski, J. April 2, 2001. *Pro-Active Maintenance for Pumps, Archives, February 2001,* **Pump-Zone.com** [Report online]. Available URL: **http://www.pump-zone.com**. Reprinted with permission of Pump-Zone.com.

3-Alfi, S., Bionda, S., Bruni, S., & Gasparetto, L. (2011). Condition monitoring of suspension components in railway bogies. *5th IET Conference on Railway Condition Monitoring and Non-Destructive Testing (RCM 2011)*, 1–6.
4-An, D., Choi, J.-H., & Kim, N. H. (2013). Prognostics 101: A tutorial for particle filter-based prognostics algorithm using Matlab. *Reliability Engineering & System Safety*, *115*, 161–169.

5-An, D., Kim, N. H., & Choi, J. H. (2015). Practical options for selecting data-driven or physics-based prognostics algorithms with reviews. *Reliability Engineering and System Safety*, *133*, 223–236.

6-Andre, D., Appel, C., Soczka-Guth, T., & Sauer, D. U. (2013). Advanced mathematical methods of SOC and SOH estimation for lithium-ion batteries. *Journal of Power Sources*.

7-Atamuradov, V., & Camci, F. (2016). Evaluation of Features with Changing Effectiveness for Prognostics. *Annual Conference of the Prognostics and Health Management Society 2016*.

8-Baraldi, P., Compare, M., Sauco, S., & Zio, E. (2013). Ensemble neural network-based particle filtering for prognostics. *Mechanical Systems and Signal Processing*, *41*(1–2), 288–300.

9-Baraldi, P., Mangili, F., & Zio, E. (2015). A prognostics approach to nuclear component degradation modeling based on Gaussian Process Regression. *Progress in Nuclear Energy*, *78*, 141–154.

10-Boukra, T., & Lebaroud, A. (2014). Identifying New Prognostic Features for Remaining Useful Life Prediction. *Power Electronics and Motion Control Conference and Exposition (PEMC), 2014 16th International*, 1216–1221.
Boutsidis, C., & Garber, D. (2015). Online Principal Component Analysis. *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms.*, *887–901.*

11-Bressel, M., Hilairet, M., Hissel, D., & Ould Bouamama, B. (2016). Extended Kalman Filter for prognostic of Proton Exchange Membrane Fuel Cell. *Applied Energy*, *164*, 220–227.

12-Burgess, W. L. (2009). Valve Regulated Lead Acid battery float service life estimation using a Kalman filter. *Journal of Power Sources*, *191*(1), 16–21.

13-Camci, F., Eker, O. F., Baskan, S., & Konur, S. (2016). Comparison of sensors and methodologies for effective prognostics on railway turnout systems. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit.*

14-Javed, K., Gouriveau, R., Zemouri, R., & Zerhouni, N. (2011). Improving data-driven prognostics by assessing predictability of features. *Annual Conference of the Prognostics and Health Management Society*, 1–6.

15-S. Russel, P. Norvig, *"Artificial intelligence – A modern approach",* Prentice Hall, 1995.

16-P. Maess, *"Artificial life meets entertainment: Life like autonomous agents",* Communications of the ACM, vol. 38, no. 11, pp. 108-114, 1995.
17- P. Stone, M. Veloso, "Multiagent systems: A survey from a machine learning perspective" , *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, 2000.
18- J. Doran, S. Franklin, N. R. Jenkins, T. J. Norman, "On cooperation in multi-agent systems" , In *UK Workshop on Foundations of Multi-agent Systems*, Warwick, 1996.
19-A. L. Dimeas, N. D. Hatziargyriou, " Operation of a Multiagent System for Microgrid Control" , *IEEE Transactions on Power Systems*, vol. 20, no. 3, pp. 1447-1455, 2006.

20-J. M. Bradshow, " An introduction to software agents" , [Online] Available: www.cs.umbc.edu/**agents**/introduction/01-Bradshaw.pdf

Contents

Figure Table