République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université AMO de Bouira

Faculté des Sciences et des Sciences Appliquées

Département d'Informatique

Mémoire de Master

en Informatique

Spécialité *: ingénierie des systèmes d'information et de logiciels*

Thème

Conception et implémentation d'un système expert d'aide au diagnostic de pannes d'un PC

Encadré par

- Mr Demmouche Mouloud

Réalisé par

- Mr Hamiche Farid
- Mr Chikh kamel

2018/2019

Remerciement

Avant tout, nous remercions Allah qui nous a donné le Courage et la volonté pour achever ce modeste travail.

Nous remercions tous ceux qui nous ont soutenus et aidés à réaliser ce projet.

Comme nous remercions tous les professeurs de département informatique et en particulier notre encadreur « Demmouche Mouloud » .

Farid & kamel

Dédicaces

A mes chères parents pour leurs sacrifices et leur soutien. Et a toute la famille « Hamiche ». A a tout mes amis je dédie ce travail.

Farid

A mes chères parents pour leurs sacrifices et leur soutien. Et a toute la famille « Chikh ». A a tout mes amis je dédie ce travail.

kamel

Table de matières

Chapitre I :Généralités sur l'IA et les systèmes experts

Chapitre II : Conception de système expert

Chapitre III : Réalisation de système expert

Liste des figures

Page

Liste des tableaux

Page

Liste des abréviations :

Abréviations	désignations
AGP	**A**ccelerated **G**raphics **P**ort
ATA	**A**dvanced **T**echnology **A**ttachment
BDD	**B**ase **D**e **D**onnées
BC	**B**ase de **C**onnaissances
BF	**B**ase de **F**aits
BR	**B**ase de **R**ègles
CD/DVD	**C**ompact **D**isk/ **Di**gital **V**ersatile **D**isc
CISEADPO	Conception et implémentation d'un système expert d'aide au diagnostique des pannes d'un micro ordinateur.
XAMPP	Appache MariaDB Php Perl
SB	**F**ront **S**ide **B**us
IA	**I**ntelligence **A**rtificielle
LED	**L**ight **E**mitting **D**iode
MI	**M**oteur d'**I**nférence
NIC	**N**etwork **I**nterface **C**ard
PCI	**P**eripheral **C**omponent **I**nterconnect
PCI Express	**P**eripheral Component Interconnect Express
PHP	**H**yper**T**ext **P**re**P**rocessor
RAM	**R**andom **A**ccess **M**emory
ROM	**R**ead **O**nly **M**emory
SATA	**S**erial **A**dvanced **T**echnology **A**ttachment)
SE	**S**ystème **E**xpert
JDBC	**Java**
UML	**U**nified **M**odeling **L**anguage
UP	**U**nified **P**rocess
USB	**(**Universal Serial Bus, en français Bus série universel)
VGA	video graphical
UC	**U**se **C**ase
Ventirad	**Venti**lateur + **Rad**iateur
DD	**Disque dur**
CPU	**Central Processeur Unit**

Introduction générale

L'être humain voulait toujours savoir et comprendre ce mécanisme mental qui lui permet de prendre des décisions et d'interagir avec le monde extérieure, ce qui engendre a la moitié de vingtième siècle un nouveau concept « l'intelligence artificielle » maintenant il est devenu très récurent dans les masses medias.
Parmi les domaines de l'intelligence artificielle figure les systèmes expert qui envahissent les différent activités de la vie : médecine, informatiques, jeux…

A cet effet les micro-ordinateurs sont devenus de plus en plus indispensables dans tous les champs de travail allant de l'agriculture à l'industrie. Cependant l'utilisation abusive de cet outil entraine des pannes et des dysfonctionnements ce qui freine les utilisateurs d'accomplir leurs tâches et fonctions.

Dans le chapitre1 nous décrirons quelques notions et définitions de l'intelligence artificielle et les systèmes experts, puis nous citerons les composants des systèmes experts, en suite nous présenterons notre domaine d'expertise (symptômes et pannes d'un PC).

Dans le chapitre 2 nous utiliserons UML pour faire la conception de notre système expert (diagramme de cas d'utilisation, diagramme de séquence, diagramme de classe …).

Dans le chapitre 3 nous réaliserons l'application de système expert en utilisant java netbeans IDE soutenu par un serveur locale XAMPP. En commençant par la création des interfaces utilisateurs afin que ces derniers puissent interagir avec le système, et en finissant par l'implémentation des programmes (code d'authentification, code chainage avant).

En fin dans la conclusion générale nous citerons les bénéfices et le profit que nous avons acquis par ce modeste travail, ainsi nous souhaiterons que les attentes des utilisateurs aient été atteintes.

CHAPITRE I

Généralités sur l'intelligence artificielle et les systèmes experts

Introduction générale :

L'être humain voulait toujours savoir et comprendre ce mécanisme mental qui lui permet de prendre des décisions et d'interagir avec le monde extérieure, ce qui engendre a la moitié de vingtième siècle un nouveau concept « l'intelligence artificielle » maintenant il est devenu très récurent dans les masses medias.
Parmi les domaines de l'intelligence artificielle figure les systèmes expert qui envahissent les différent activités de la vie : médecine, informatiques, jeux…

A cet effet les micro-ordinateurs sont devenus de plus en plus indispensables dans tous les champs de travail allant de l'agriculture à l'industrie. Cependant l'utilisation abusive de cet outil entraine des pannes et des dysfonctionnements ce qui freine les utilisateurs d'accomplir leurs tâches et fonctions.

Dans le chapitre1 nous décrirons quelques notions et définitions de l'intelligence artificielle et les systèmes experts, puis nous citerons les composants des systèmes experts, en suite nous présenterons notre domaine d'expertise (symptômes et pannes d'un PC).

Dans le chapitre 2 nous utiliserons UML pour faire la conception de notre système expert (diagramme de cas d'utilisation, diagramme de séquence, diagramme de classe …).

Dans le chapitre 3 nous réaliserons l'application de système expert en utilisant java netbeans IDE soutenu par un serveur locale XAMPP. En commençant par la création des interfaces utilisateurs afin que ces derniers puissent interagir avec le système, et en finissant par l'implémentation des programmes (code d'authentification, code chainage avant).

En fin dans la conclusion générale nous citerons les bénéfices et le profit que nous avons acquis par ce modeste travail, ainsi nous souhaiterons que les attentes des utilisateurs aient été atteintes.

Problématique :

Notre problématique concerne plus Particulièrement la modélisation de savoir-faire d'un expert dans le domaine maintenance micro-ordinateur (PC).
Y a- t-il un moyen technique pour diagnostiquer une défaillance d'un PC à travers d'un ensemble de symptômes observés ou matériels visibles. Ainsi éviter à chaque fois la répétition des tests fatidiques et gagner du temps.

Ce problème ne peut pas être résolu en utilisant des méthodes traditionnelles de calcul. Alors que les applications systèmes expert peuvent répondre à cette problématique.

Introduction

Durant ce chapitre nous allons en premier lieu décrire quelques notions sur l'intelligence artificielle puis nous définirons les systèmes experts allant de la base connaissances au moteur d'inférence.

En deuxième lieu nous citerons les différentes présentations de connaissances et base de données orientées objets.

1. L'intelligence Artificielle(IA)

1.1 Définitions de l'IA

L'intelligence artificielle (IA, ou AI en anglais pour *Artificial Intelligence*) consiste à mettre en œuvre un certain nombre de techniques visant à permettre aux machines d'imiter une forme d'intelligence réelle. L'IA se retrouve implémentée dans un nombre grandissant de domaines d'application.

La notion voit le jour dans les années 1950 grâce au mathématicien Alan Turing. dans son livre *Computing Machinery and Intelligence*, ce dernier soulève la question d'apporter aux machines une forme d'intelligence. Il décrit alors un test aujourd'hui connu sous le nom « Test de Turing » dans lequel un sujet interagit à l'aveugle avec un autre humain, puis avec une machine programmée pour formuler des réponses sensées. Si le sujet n'est pas capable de faire la différence, alors la machine a réussi le test et, selon l'auteur, peut véritablement être considérée comme « intelligente ».

De Google à Microsoft en passant par Apple, IBM ou Facebook, toutes les grandes entreprises dans le monde de l'informatique planchent aujourd'hui sur les problématiques de l'intelligence artificielle en tentant de l'appliquer à quelques domaines précis. Chacun a ainsi mis en place des réseaux de neurones artificiels constitués de serveurs et permettant de traiter de lourds calculs au sein de gigantesques bases de données.[1]

1.1.1 Quelques exemples d'usage d'IA

La vision artificielle, par exemple, permet à la machine de déterminer précisément le contenu d'une image pour ensuite la classer automatiquement selon l'objet, la couleur ou le visage repéré.

Les algorithmes sont en mesure d'optimiser leurs calculs au fur et à mesure qu'ils effectuent des traitements. C'est ainsi que les filtres anti spam deviennent de plus en plus efficaces au fur et à mesure que l'utilisateur identifie un message indésirable ou au contraire traite les faux-positifs.

La reconnaissance vocale a le vent en poupe avec des assistants virtuels capables de transcrire les propos formulés en langage naturel puis de traiter les requêtes soit en répondant directement via une synthèse vocale, soit avec une traduction instantanée ou encore en effectuant une requête relative à la commande.

1.2 Les avantages et les inconvénients de l'intelligence artificielle

➢ Les avantages

- Optimisation et mise à jour des systèmes d'exploitation et les logiciels d'applications.
- la limitation des erreurs de calculs et la précision dans le travail.
- Amélioration de la qualité des jeux (jeux d'échec, scrable…)

➢ Les inconvénients

- Au lieu de compter sur son cerveau l'homme compte sur des agents intelligents

1.3 domaine d'application de l'IA

Aujourd'hui l'intelligence artificielle regorge de nombreuses capacités et de tâches possibles comme :

1.3.1 Le calcul formel

Cela permet d'exécuter de nombreuses formules symboliques (ex : mathématico). [3]

1.3.2 Santé

Lien patient-médecin, recherche, prévention, diagnostic, traitement : l'IA pénètre tous les segments de l'industrie médicale. Elle est déjà meilleure que le médecin pour détecter le caractère cancéreux d'un mélanome ou analyser une radio des poumons. Le système Watson d'IBM pratique l'aide au diagnostic et à la prescription, notamment en fonction du séquençage ADN des tumeurs cancéreuses. D'autres IA jouent les psys virtuels… [2]

1.3.3 La reconnaissance

Qu'elle soit de la parole, de l'écriture ou du visage, l'IA a fait de nombreux progrès dans ces actions. [3]

1.3.4 L'apprentissage automatique

Pour qu'une intelligence artificielle puisse faire son devoir elle doit apprendre les notions données par l'homme.

1.3.5 Les systèmes experts

L'IA peut réaliser la même tâche qu'une personne humaine, notamment dans la robotique (Ex : logiciel MYCIN en 1974 qui donne des diagnostiques) [3]

2. systèmes experts (SE)

2.1 Définition d'un SE

Un système expert est un programme conçu pour simuler le comportement d'un humain qui est un spécialiste ou un expert dans un domaine très restreint" P. Denning (1986).

Système expert est un programme :

- Incorporant à la fois
  - Des connaissances formelles
  - Des jugements personnels de l'expert (sous forme d'heuristiques)
- Expliquant à la fois
  - Le raisonnement conduisant aux réponses aux questions
  - Le savoir qu'il contient
- Capable d'incorporer du nouveau savoir dans le savoir existant de manière incrémentale
- Séparant
  - Les connaissances,
  - Le contrôle (moyen d'utiliser les connaissances)
- "Dialoguant" (systèmes évolués) avec l'utilisateur dans un langage proche du sien (langue naturelle) [4]

2.2 Structure d'un système expert

L'architecture d'un système expert typique est constituée de plusieurs modules, comme le montre la figure suivante :

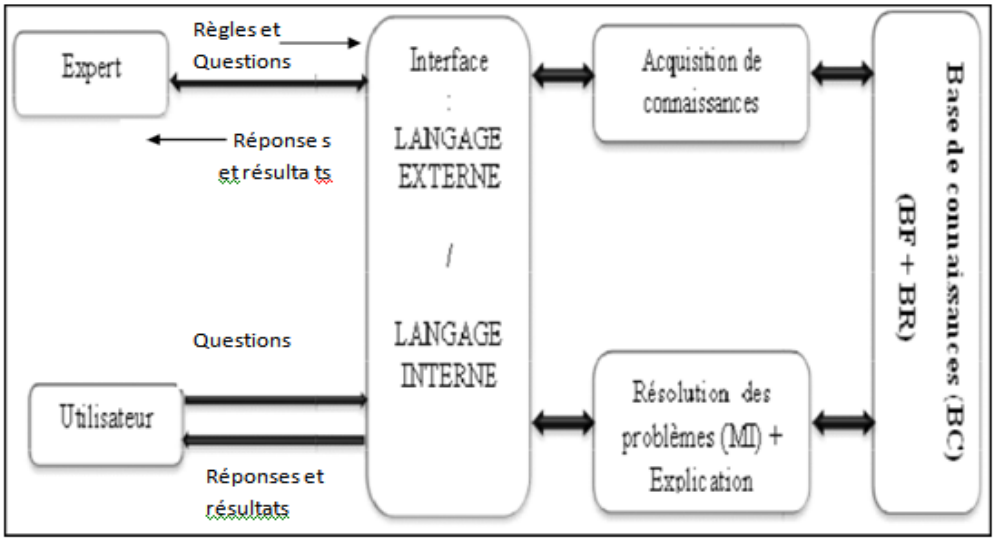


Figure 1. Architecture d'un Système Expert **[5]**

Un système expert est composé de :

- **L'interface utilisateur**

Elle permet aux utilisateurs de consulter le système pour résoudre un problème donné du domaine d'expertise. [6]

- **la base de connaissances**

contient les deux bases suivantes :

– **base de faits** : code la connaissance sur l'étude en cours
Son état évolue en cours d'expertise (*mémoire de travail*)
– **base de règles** : code la connaissance sur le domaine.
Fixe pour plusieurs expertises.
Règle : SI condition ALORS action

2.2.1 Le moteur d'inférences

Composé des algorithmes utilisés pour la déduction :
• chaînage avant
• chaînage arrière
• chaînage mixte.

2.2.1.1 le chaînage avant

Dans le mode de chaînage avant, le moteur d'inférence part des faits pour arriver au but, c'est-à-dire qu'il ne sélectionne que les règles dont les conditions de la partie gauche sont vérifiées (∈ base de faits), puis applique la première de ces règles, ce qui ajoute la conclusion à la base. Ce processus est réitéré jusqu'à ce qu'il n'y ait plus de règles applicables ou que le but soit atteint.
Voici l'algorithme du chaînage avant :

- **Algorithme du chainage avant**

Entrée : BF (base de faits), BR (base de règles), F (proposition à vérifier)

- **DEBUT**
- **TANT QUE** F n'est pas dans BF **ET QU**'il existe dans BR une règle applicable
- **FAIRE**
- Prendre la première règle applicable R
- BR = BR - R (désactivation de R)
  BF = BF union conclusion(R)
  (déclenchement de la règle R, sa Conclusion
- est rajoutée à la base de faits)
- **FIN TANT QUE**
- **SI** F appartient à BF **ALORS**
- F est établi (succès)
- **SINON**
  F n'est pas établi (échec) **FIN**

Voici un exemple d'exécution :

Soit la base de connaissances suivante :

Base de faits initiale

BF= {H, K} ET le but {C}

Base des règles

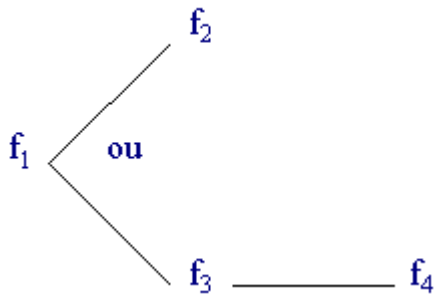
- **R1 :** A --> E
- **R2 :** B --> D
- **R3 :** H --> A
- **R4 :** E et G --> C
- **R5 :** E et K --> B
- **R6 :** D et E et K --> C
- **R7 :** G et K et F --> A

- H --> A (R3) et la base de faits BF = {A, H, K}
- A --> E (R1) et BF = {A, E, H, K}
- E et K --> B (R5) et BF = {A, B, E, H, K}
- B --> D (R2) et BF = {A, B, D, E, H, K}
- D et E et K --> C (R6) et BF = {A, B, C, D, E, H, K}

Base finale : {A, B, C, D, E, H, K}. C figure dans la base finale, réussite [7]

2 .2.1.2 le chaînage arrière

Le chaînage arrière est un mécanisme d'exploitation guidé par les buts.
Si q est non vrai et si p implique q alors p est non vrai.
Le chaînage arrière traduit un raisonnement déductif :



- de f4 est déduit f3
- de f3 est déduit f1

- **Algorithme du chainage arrière**

Fonction **chainage Arrière**
Paramètres : in BR, in BF, in listeButs.
SI est Vide (listeButs) **ALORS**
Résultat ← SUCCES

SI demBut (premier (listeButs)) **ALORS**
Résultat ←chaînage Arriere (suite (listeButs))
SI
Résultat ← ECHEC
FIN SI
FIN SI
Retourner Résultat. [8]

- **Exemple :**

Soit la base de faits **BF= {A, M}**, on cherche à démontrer le fait **C**, et prenons comme critère de résolution de conflits l'ordre de survenance des règles dans la base de connaissances.

Soit la base de règles suivante :

1. **SI** E et F **alors** C
2. **SI** M et F **alors** E
3. **SI** H et A **alors** F
4. **SI** F et H **alors** M
5. **SI** A et M **alors** H

Raisonnement

Numéro cycle	Evaluation de la base des faits	Numéro règles
1	A, M, F, E	1
2	A, M, H, F	2
3	A, M, H	3
4	A, M	

Base finale : {A, M }

2.2.1.3 Le chaînage mixte

Le chaînage mixte est une extension du chaînage arrière qui supplante ce dernier car il pallie à un problème qu'il pose. En effet le chaînage arrière pose le problème du retard à l'évaluation. Lorsque l'on évalue un but par chaînage arrière, l'évaluation peut conduire à des conclusions sur d'autres attributs, qui ne sont pas faites et pour lesquelles d'autres procédures de chaînage arrière sont inutilement invoquées.

De là l'idée de combiner le chaînage arrière, inefficace à lui seul, à du chaînage avant. Ainsi après chaque évaluation d'une prémisse de règle, une propagation en avant de cette évaluation est faite pour tirer l'ensemble des conclusions qu'il est possible d'en tirer. [9]

2.3 Domaine d'application des systèmes experts

Informatique, Médecine, mathématiques, chimie, biologie, géologie…

2.3.1 Quelque systèmes experts

MYCIN : En 1972-73 fut créé Mycin (en), un système expert de diagnostic de maladies du sang et de prescription de médicaments, avec un vrai moteur et une vraie base de règles.

POSPECTOR, 1978 : aide le géologue à évaluer l'intérêt d'un site en vue d'une prospection minière. (1600 règles).

MUSCADAT, 1984 : mathématique, démonstration de théorèmes

3. Représentation des connaissances

La représentation des connaissances désigne un ensemble d'outils et de procédés destinés d'une part à représenter et d'autre part à organiser le savoir humain pour l'utiliser et le partager. les connaissances n'ont jamais été, et ne sont toujours pas, systématiquement représentées par des mots et des phrases. Les systèmes d'informations utilisent notamment :

- Des dessins et des schémas ; utilisés pour noter les connaissances concernant les animaux, les plantes, des objets, etc. dans les études de sciences naturelles (anatomie du corps humain par exemple) ;
- Des plans ont permis de coordonner le travail des architectes et des maçons ou des tailleurs de pierre, depuis l'Antiquité ;
- Le dessin industriel, après le dessin cavalier (cf. ceux de Léonard de Vinci), est venu préciser la logique de fabrication des machines depuis le XIX siècle ;
- La cartographie, puis les systèmes d'information géographique (SIG) géolocalisent un nombre croissant d'éléments de connaissance.[11]

- **Classement des connaissances**

Les outils classiques (non électroniques) de représentation des connaissances sont les taxonomies ou classifications, qui permettent d'organiser les connaissances sur les objets du monde, et les thesaurus utilisés en indexation documentaire.

- **Formalisation des connaissances**

Des outils plus formels et permettant de représenter des connaissances complexes sont par exemple les graphes conceptuels ou les réseaux sémantiques.

Dans le domaine des nouvelles technologies, la représentation formelle des connaissances s'est développée dans le domaine de l'intelligence artificielle. Dans une représentation formelle, les connaissances sont représentées par des objets logiques reliés par des propriétés, axiomes et règles. Ce type de représentation est utilisé dans les systèmes experts.

Le développement du Web, et en particulier la perspective du Web sémantique a renouvelé le domaine en introduisant le terme controversé d'*ontologie*. Un certain nombre de langages ont été développés dans cette perspective, comme les standards RDFS SKOS et OWL .

- **liens entre connaissances et raisonnements**

La représentation formelle des connaissances (ou des croyances) permet d'automatiser divers traitements sur ces informations. C'est un des domaines de recherche de l'intelligence artificielle symbolique : la simulation de raisonnements « intelligents » à partir d'informations.

Un des cadres formels les plus utilisés est la logique propositionnelle (ordre 0 ou 1 ou 0+). En effet, un grand nombre de problèmes peuvent se résoudre via un codage en logique propositionnelle.

On peut par exemple coder divers jeux sous forme de formule propositionnelle

D'autres cadres formels permettent de représenter des informations présentant une structure particulière, comme les systèmes d'argumentation, les réseaux bayésiens ou la logique possibiliste.

3.1. Les réseaux sémantiques

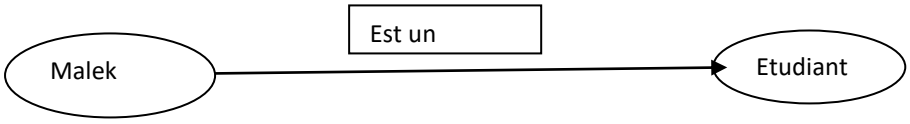
Le réseau sémantique est un outil qui simule notre représentation de la mémoire. C'est un modèle qui montre comment,
1) l'information pourrait être représentée en mémoire ?
2) comment on pourrait accéder à ces informations ?

- **Explication**

Notre mémoire est représentée comme un bassin de donnée contenant des concepts, des événements, des sensations, ...Tous ces éléments forment un immense réseau en interrelation. couché sur le papier, un réseau sémantique est composé de *nœuds* dont les interrelations sont établies par des *pointeurs* étiquetés. Les nœuds sont les différents types d'information en mémoire. A ces nœuds peuvent être associées des propositions, énoncés qui caractérisent les propriétés s'appliquant au nœuds du réseau. L'étiquette associée au pointeur indique quel est le type de relation entre deux nœuds.

- **Structure générale des réseaux sémantiques**

Tout d'abord, les réseaux sémantiques peuvent représenter des objets individuels, des catégories d'objets, et des relations entre objets ou catégories. Les objets sont représentés dans des boîtes tandis que les relations étiquettent les nœuds qui relient ces objets : **[12]**



Ceci correspond à l'assertion en logique à l'application d'un prédicat sur un terme :

Etudiant (Malek)

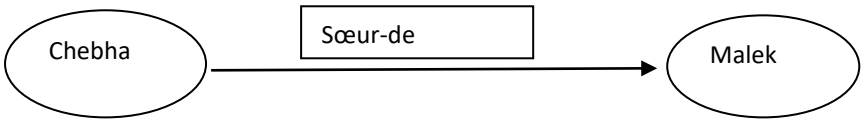
Ou bien, en théorie des ensembles :

malek∈ Etudiant.

De la même manière, l'instance de la relation :

sœur − de(X,Y), sœur − de(chebha, malek),

s'écrit en réseaux sémantiques par :



3.2. Les représentations logiques

- **Logique d'ordre 0 :**

C'est la logique booléenne dont les formules appelées des propositions.

- Le calcul propositionnel et le calcul arithmétique
- Les formules sont construites à l'aide de variables d'un ensemble dénombrable et des symboles {}.

Exemple :

Si a et c alors b

- **Logique d'ordre 0+ :**

Cela revient à dire que l'on considère les fonctions et prédicats comme des objets à part entière.

Un prédicat d'ordre supérieur est un prédicat qui prend comme argument un ou plusieurs autres prédicats. De manière générale, un prédicat d'ordre *n* prend comme argument un ou plusieurs prédicats d'ordre *n*-1, avec *n* > 1. Les mêmes définitions s'appliquent aux fonctions d'ordre supérieur. [13]

Exemple :

Si âge = 9 **Alors** **période** = « enfance »

Prémisse conclusion

- **Logique d'ordre 1 (logique des prédicats)**

Un prédicat est une expression logique qui prend la valeur 0 ou 1 selon la valeur des arguments.

Logique d'ordre 1 constitué de :

* Des variables (x, y ,z …)
* Des constantes (a, b ,c …)
* Quantificateurs logique (∀, ∃)
* Connecteurs (~, →, ↔, ^).
* Des fonctions (f, g ,h ...) [13]

Exemple :

Etudiant (x) → x est un Etudiant .

∀ (x) oiseau(x) → avoir_ailles (x)

3.3. Les réseaux de neurones

Un réseau de neurones artificiels, ou réseau neuronal artificiel, est un système dont la conception est à l'origine schématiquement inspirée du fonctionnement des neurones biologiques, et qui par la suite s'est rapproché des méthodes statistiques.

Les réseaux de neurones sont généralement optimisés par des méthodes d'apprentissage de type probabiliste, en particulier bayésien. Ils sont placés d'une part dans la famille des applications statistiques, qu'ils enrichissent avec un ensemble de paradigmes [2] permettant de créer des classifications rapides (réseaux de Kohonen en particulier), et d'autre part dans la famille des méthodes de l'intelligence artificielle auxquelles ils fournissent un mécanisme perceptif

indépendant des idées propres de l'implémenter et fournissant des informations d'entrée au raisonnement logique formel (voir *Deep Learning*).

En modélisation des circuits biologiques, ils permettent de tester quelques hypothèses fonctionnelles issues de la neurophysiologie, ou encore les conséquences de ces hypothèses pour les comparer au réel.

- **Structure du réseau**

Un réseau de neurones est en général composé d'une succession de couches dont chacune prend ses entrées sur les sorties de la précédente. Chaque couche (i) est composée de N_i neurones, prenant leurs entrées sur les N_{i-1} neurones de la couche précédente. À chaque synapse est associé un poids synaptique, de sorte que les N_{i-1} sont multipliés par ce poids, puis additionnés par les neurones de niveau i, ce qui est équivalent à multiplier le vecteur d'entrée par une matrice de transformation. Mettre l'une derrière l'autre les différentes couches d'un réseau de neurones reviendrait à mettre en cascade plusieurs matrices de transformation et pourrait se ramener à une seule matrice, produit des autres, s'il n'y avait à chaque couche, la fonction de sortie qui introduit une non linéarité à chaque étape. Ceci montre l'importance du choix judicieux d'une bonne fonction de sortie : un réseau de neurones dont les sorties seraient linéaires n'aurait aucun intérêt.

Au-delà de cette structure simple, le réseau de neurones peut également contenir des boucles qui en changent radicalement les possibilités mais aussi la complexité. De la même façon que des boucles peuvent transformer une logique combinatoire en logique séquentielle, les boucles dans un réseau de neurones transforment un simple dispositif de reconnaissance d'entrées en une machine complexe capable de toutes sortes de comportements.

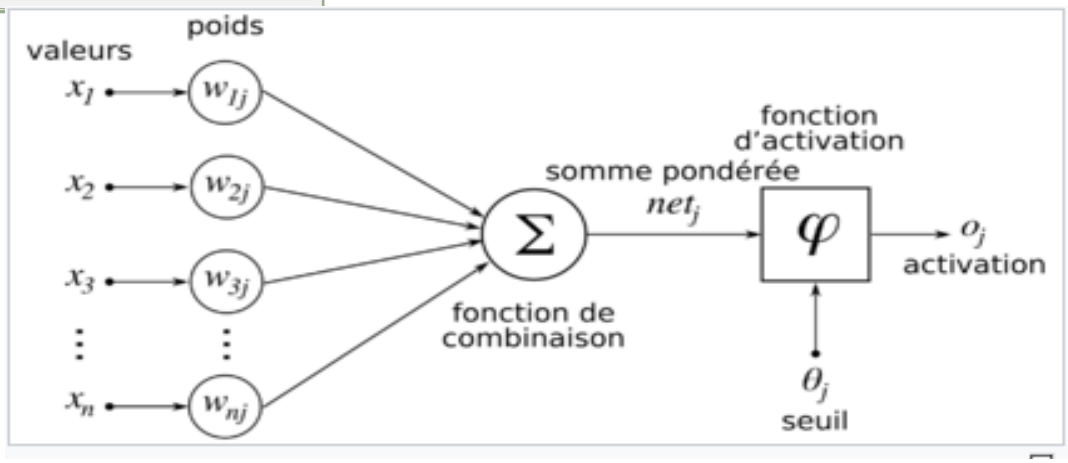


Figure 2 : Représentation d'un réseau neurone artificiel [14]

3.4. Systèmes à base de règles de productions

Les systèmes à base de règles de production "emmagasinent" la connaissance sous forme de règles : SI...ALORS…afin d'être le plus proche de la façon naturelle de la représentation des connaissances. La partie entre le SI et le ALORS s'appelle la partie prémisses de la règle. Elle correspond à une conjonction d'expressions qui doivent être vérifiée pour que la règle se Déclenche. La partie qui suit le ALORS s'appelle la partie conclusion. Elle correspond à une conjonction d'actions.

En logique de propositions ou de premier ordre, les règles sont décrites via des clauses contenant des propositions ou des prédicats. **[15]**

Exemple :

SI X possède un micro-ordinateur et X maitrise java ALORS il pourrait créer des applications informatiques.

3.5. Les objets

Il est impossible de parler de la Programmation Orientée Objet sans parler d'objet, bien entendu. Tâchons donc de donner une définition aussi complète que possible d'un objet.
Un objet est avant tout une structure de données. Autrement, il s'agit d'une entité chargée de gérer des données, de les classer, et de les stocker sous une certaine forme. En cela, rien ne distingue un objet d'une quelconque autre structure de données. La principale différence vient du fait que l'objet regroupe les données et les moyens de traitement de ces données.

Un objet rassemble de fait deux éléments de la programmation procédurale.

- **Les champs**

  Les champs sont à l'objet ce que les variables sont à un programme : ce sont eux qui ont en charge les données à gérer. Tout comme n'importe quelle autre variable, un champ peut posséder un type quelconque défini au préalable : nombre, caractère... ou même un type objet. [16]
- **Les méthodes**

  Les méthodes sont les éléments d'un objet qui servent d'interface entre les données et le programme. Sous ce nom obscur se cachent simplement des procédures ou fonctions destinées à traiter les données.

- Les champs et les méthodes d'un objet sont ses membres.

Si nous résumons, un objet est donc un type servant à stocker des données dans des champs et à les gérer au travers des méthodes.
Si on se rapproche du Pascal, un objet n'est donc qu'une extension évoluée des enregistrements (type record) disposant de procédures et fonctions pour gérer les champs qu'il contient. [16]

Exemple :

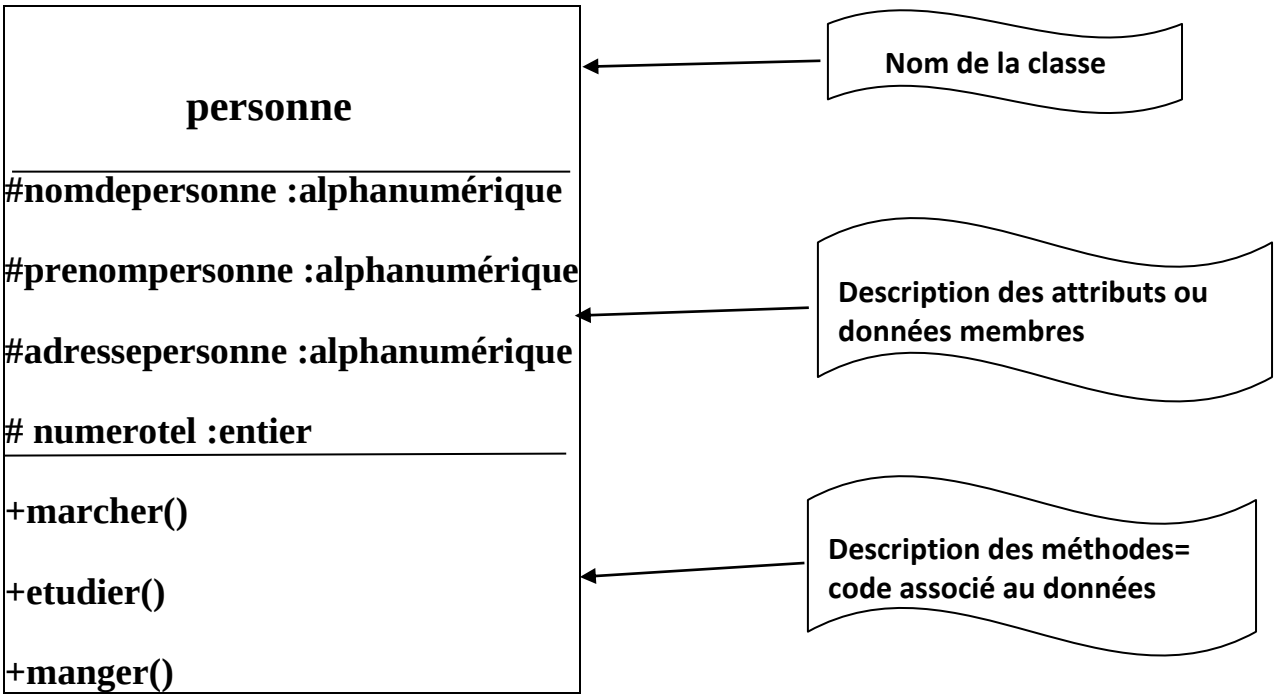


Figure 3 : représentation de la classe personne.

Les trois fondamentaux dans la structure d'objet sont :

- **Encapsulation**

Est déjà une réalité dans les langages procéduraux (comme le Pascal non objet par exemple) au travers des unités et autres librairies, il prend une toute nouvelle dimension avec l'objet.

En effet, sous ce nouveau concept se cache également un autre élément à prendre en compte : pouvoir masquer aux yeux d'un programmeur extérieur tous les rouages d'un objet et donc l'ensemble des procédures et fonctions destinées à la gestion interne de l'objet, auxquelles le programmeur final n'aura pas à avoir accès.

L'encapsulation permet donc de masquer un certain nombre de champs et méthodes tout en laissant visibles d'autres champs et méthodes.[16]

- **Héritage**

  Est un concept puissant de la programmation orientée objet, permettant entre autres la réutilisabilité (décomposition du système en composants) et l'adaptabilité des objets grâce au polymorphisme. Elle se nomme ainsi car le principe est en quelque sorte le même que celui d'un arbre généalogique. Ce principe est fondé sur des classes dont les « filles » héritent des caractéristiques de leur(s) « mère(s) ».

  Chaque classe possède des caractéristiques (attributs et méthodes) qui lui sont propres. Lorsqu'une classe fille hérite d'une classe mère, elle peut alors utiliser ses caractéristiques.

- Si la classe mère est abstraite (retardée) il y a de forte chance qu'elle présente des caractéristiques abstraites (retardées). Pour être effective (instanciable), la classe fille doit alors les définir, sinon elle sera elle même abstraite.
- Si la classe mère est effective (toutes les caractéristiques sont définies) alors la classe fille est également effective. Il est possible d'y ajouter des caractéristiques, d'utiliser les caractéristiques héritées (appartenant aux parents) et de redéfinir les méthodes héritées. Généralement cette redéfinition se fait par surcharge sémantique (on déclare de nouveau la méthode avec le même nom et la même signature).
- Pour qu'une classe fille puisse hériter des propriétés de la classe mère, il faut que les propriétés de la classe mère possèdent des attributs de visibilité compatibles. Il existe dans la plupart des langages trois niveaux de visibilité (dont la définition peut changer légèrement) :
- Public est la visibilité la plus large (accessible directement en dehors de la classe)
- Private est la plus restrictive (accessible uniquement depuis la classe courante)
- Protected est intermédiaire et visible par la classe courante, toutes les classes filles, mais inaccessible en dehors des classes mère/filles

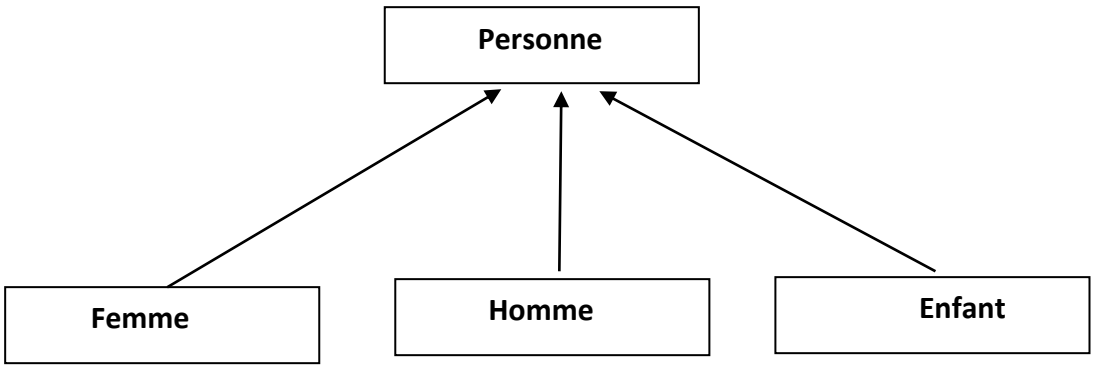


Figure 4 : Exemple d'héritage

- **Polymorphisme**

le polymorphisme en informatique est l'idée d'autoriser le même code à être utilisé avec différents types, ce qui permet des implémentations plus abstraites et générales.[18]

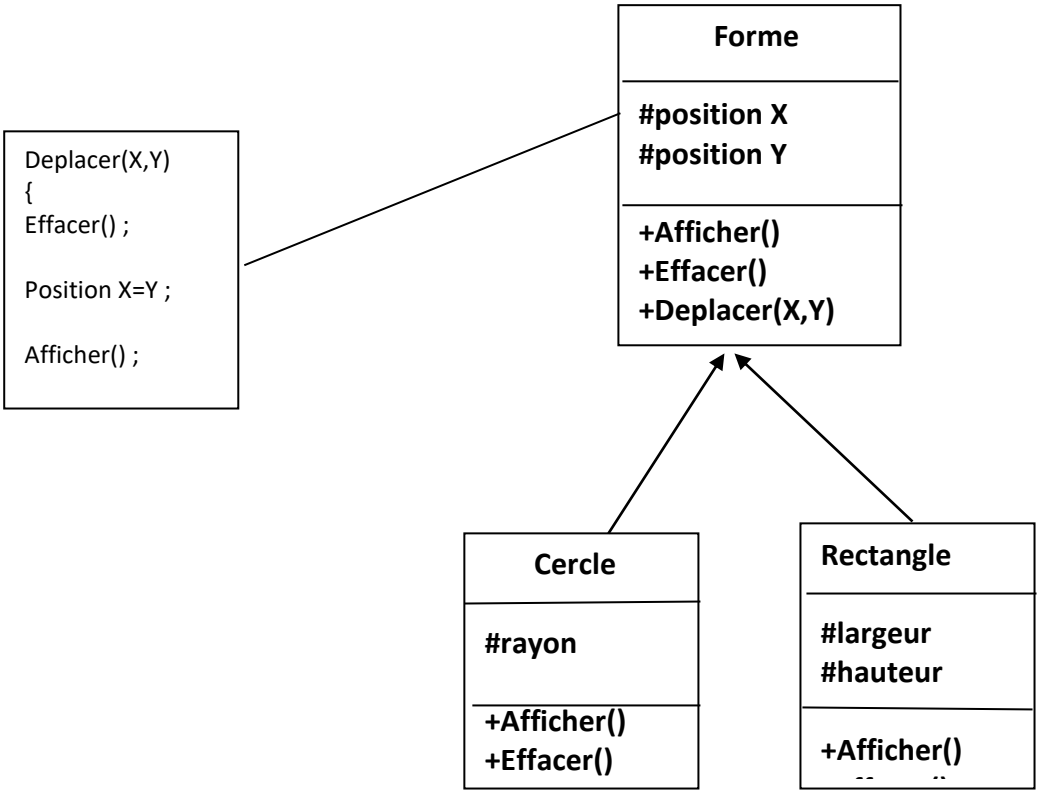


Figure 5 : exemple de polymorphisme [19]

4. Le cycle de base d'un moteur d'inférence

Au cours de son raisonnement, le moteur d'inférence passe par une série de cycles. Un cycle est constitué de deux phases : évaluation et exécution.

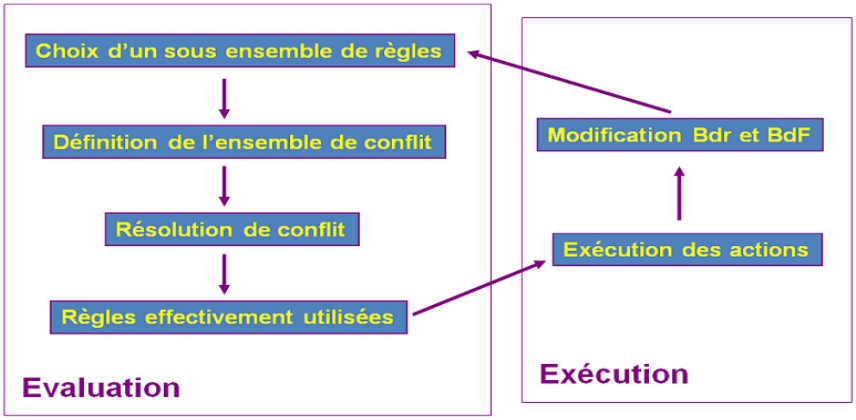


Figure 6 : Schéma du fonctionnement simplifié d'un moteur d'inférence. **[20]**

4.1 La phase d'évaluation

Cette phase consiste en un choix de la règle qui va effectivement être utilisée à un moment donné. En effet, lors de la rédaction du programme, on ne définit jamais explicitement dans quel ordre vont être déclenchées les différentes règles. Le déclenchement va se faire en fonction de l'adéquation entre une situation donnée et les conditions des règles. Et c'est ce qui va différencier une IA d'un algorithme.

Le choix de la règle utilisée à un moment donné se fait en 3 étapes :

- **La sélection**

  Le moteur d'inférence va choisir dans l'ensemble de toutes les règles un sous-ensemble qui, a priori, mérite d'être utilisé (cas d'une famille de règles se rapportant à une situation en cours), cette étape ne figure pas dans notre programme puisque le nombre de règles est assez limité dans celui-ci.

- **Le filtrage**

  Parmi le sous-ensemble de règles choisies, on ne garde que les règles dont les conditions sont satisfaites et qui formeront ce que l'on appelle l'ensemble de conflit.

- **La résolution de conflits**

  La résolution de conflit ou choix des règles qui vont être effectivement déclenchées et utilisées est souvent arbitraire.

4.2 La phase d'exécution

Cette phase consiste simplement à exécuter la partie « action » des règles déclenchées et à modifier en conséquence la base de connaissances qui pourra ensuite être réutilisée dans la phase d'évaluation.

Le fonctionnement n'est, au final, pas aussi complexe que l'on pourrait se l'imaginer.

La difficulté réside plutôt dans l'acquisition de la connaissance par le système. On ne peut pas être expert, programmeur et usager en même temps ! De plus, il faut adapter sa réflexion au langage du programme.

Conclusion

A la moitié de vingtième siècle tout le monde parlait de l'intelligence artificielle dont les bénéfices sont apparus dans tous les domaines d'activité maintenant. Plusieurs agents intelligents ont été mis au service de l'être humain (robot aspirateur, voiture autonome, avion sans pilote, la cuisine intelligente …)

Parmi les domaines de l'intelligence artificielle figure les systèmes expert, ces derniers ont été implantés presque dans tous les domaines et en particulier en médicine et en informatique pour résoudre les problèmes clés.

Les systèmes experts ont prouvé leurs efficacités et leurs exactitudes dans la maintenance informatique (matériels et logiciel) dont nous allons présenter les composants et les pannes d'un PC dans le chapitre qui suit.

CHAPITRE II

Conception de système expert

Introduction

Dans ce chapitre nous allons utiliser UML pour concevoir et modéliser notre système.

Ce langage pourrait être appliqué à toute science fondée sur la description d'un système.

Car il offre plusieurs diagrammes : diagrammes de cas utilisation, séquence, activités, classe…

UML est utilisé pour spécifier, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet. UML offre un standard de modélisation, pour représenter l'architecture logicielle.

Grâce aux outils de modélisation UML, il est également possible de générer automatiquement tout ou partie du code d'une application logicielle, par exemple en langage Java, à partir des divers documents réalisés.

1. Spécification des besoins :

Pour concevoir notre système expert ona suivis une démarche basée sur :

- Interviewer l'expert du domaine.
- Choisir la façon de la présentation de connaissances.
- Déterminer le type de raisonnement et la stratégie utilisée (chainage avant ou arrière).
- Concevoir l'interface utilisateur.
- Le prototype doit être testé et affiné par l'ingénieur de connaissances et l'expert de domaine en même temps.
- Le prototype peut être complété au fur et à mesure en ajoutant des nouveaux éléments dans la base de connaissance.
- Utilisation l'UML pour la conception.
- Les pannes détaillées d'un micro-ordinateur.

2. présentation des composants d'un ordinateur

L'ordinateur (PC) se compose de plusieurs éléments.

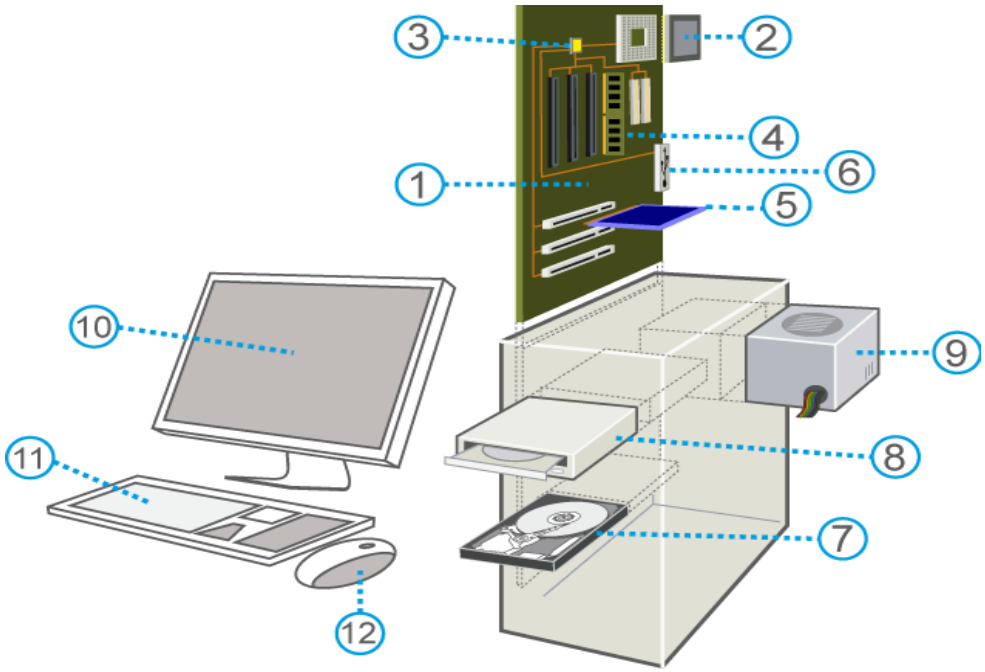


Figure 7 : Constitution d'un ordinateur personnel (P.C. : personal computer)

1	La carte mère	7	Le disque dur
2	Le processeur ou microprocesseur	8	Le lecteur de disque
3	Le(s) bus	9	L'alimentation électrique
4	La mémoire vive	10	L'écran
5	la carte graphique	11	Le clavier
6	Les entrées-sorties	12	La souris

Tableau 1 : Liste des principaux composants d'un ordinateur personnel. **[21]**

3. Etude des pannes d'un micro-ordinateur

3.1. Pannes des composants d'un micro-ordinateur

Composant	Visibilité de la panne	Symptôme
Souris	**Aucune**	Le curseur ne réagit pas au mouvement de la souris
Clavier	**Aucune**	Aucune réaction aux actions du clavier
		Les lettres tapées ne s'affichent pas sur l'écran
		Quand on tape sur une lettre une autre s'affiche
Prise USB (Universal Sérial Bus)	**Aucune**	Le système d'exploitation n'affiche pas les périphériques connectés à la prise USB
Prise jack (Output)	**Aucune**	Absence de son
Prise jack (Input)	**Aucune**	Insensibilité à la parole
Lecteur de cartes mémoires	**La lampe témoin ne s'allume pas**	Le système d'exploitation n'affiche pas les cartes mémoire insérées
Bouton d'allumage	**Aucune**	L'ordinateur ne s'allume pas
Bouton « Reset»	**Aucune**	L'ordinateur ne redémarre Pas
Prise HDMI	**Aucune**	Absence de l'image
		Absence du son
Prise Séries	**Aucune**	Le curseur ne réagit pas au mouvement de la souris
Prise VGA	**Aucune**	Absence d'image
Prise DVI	**Aucune**	Absence d'image
Prise de courant	**Aucune**	L'ordinateur ne s'allume pas
Prise PS2 Verte	**Aucune**	Le Clavier ne réagit pas
Prise PS2 Violette	**Aucune**	Le curseur ne réagit pas au

		mouvement de la souris
Prise Ethernet	**Aucune**	Pas de connexion internet
Baffle/ connecteur baffle	**Aucune**	Absence de son
Lecteur CD/DVD	**Aucune**	Insensibilité à la lecture du CD/DVD
Imprimante	**Aucune**	Non réaction aux demandes d'impression
Flash Disc /carte mémoire	**Aucune**	Le système affiche le périphérique mais aucun Accès

Composant	**Visibilité de la panne**	**Symptôme**
Carte graphique	**Aucune**	Aucun affichage
		Ralentissement de l'image
	Ventilateur ne tourne pas	
	Déformation du connecteur (AGP/PC)	Déformation de l'image
	Gonflement des Condensateurs	Ecran Bleu
Carte son	**Aucune**	Absence de son
	Déformation du connecteur	Déformation du son
RAM	**Aucune**	Ecran bleu
	Déformation du connecteur	Ralentissement de l'ordinateur
		Plantage de l'ordinateur
Chipset	**Odeur flagrante de brulure**	L'ordinateur ne boot pas
Pile	**Aucune**	Configuration par défaut
Câble d'alimentation	**Aucune**	L'ordinateur ne s'allume pas
Boitier d'alimentation	**Aucune**	L'ordinateur ne s'allume pas
		Ralentissement du Pc
Ventilateur / Ventirad	**Bruit gênant**	Ralentissement du Pc
	Aucune	Au bout d'un moment le Pc

		s'éteint
Carte réseau (Ethernet /Wifi)	**Aucune**	Pas de connexion En Local, ni sur Internet
	Lampe témoin éteinte (pour certaines cartes réseaux)	
	Déformation du connecteur	

Tableau 6:pannes de composants d'un micro-ordinateur

3.2.Symptôme d'écran bleu

Le système d'exploitation parfois ne démarre plus ou bien redémarre chaque instant donnant le lieu à un écran bleu comme le montre le tableau qui suit :

Code d'écran bleu	Matériel
0x0000000A ou 0xA	Incompatibilité de Matériel
0x100000EA	Carte Graphique
0x00000024	Problème de Câblage
0x000000C2 ou Stop 0xC2	Barrettes Mémoire défaillante (RAM)
0x00000050	Mémoire Défaillante (ROM)
0x0000006B	Lecteur CD ou DVD
0x0000007E	Clavier
0x0000007F	Soucis de Microprocesseur
0x1000008E	Incompatibilité Carte mère et processeur
0x0000009C	Ventilateur de micro-processeur défectueux ou insuffisant
0x000000CE	Carte Graphique (vidéo)

Tableau 7 : code d'écran bleu **[22]**

3.3. Signification de combinaisons de bips sonores

Un grand nombre des pannes associées aux signaux sonores ou de voyants clignotants requiert une réparation ou de nouvelles pièces.

Combinaison de Bips sonores	Signification
1-1-2-1	Problème de carte mère. ou du processeur
1-1-2-3	Erreur d'initialisation du système matériel
1-1-3-1	Problème de Chipset
1-1-3-3	Disfonctionnement du CPU (Processeur)
1-3-3-1	Erreur mémoire volatile (RAM)
2-1-2-3	Erreur de ROM
2-1-3-2	Erreur du bus PCI
4-2-4-3	Clavier
1-2-1-1	Puissance insuffisante (Boitier d'alimentation)
2-1-3-3	Problème de Carte Graphique
4-2-4-1	carte mère défectueuse ou un de ses composants

Tableau 8 : Signification des combinaisons de Bips. **[23]**

4. UML

4.1 Définition

Le Langage de Modélisation Unifié, de l'anglais *UnifiedModelingLanguage* (UML), est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé en développement logiciel et en conception orientée objet.

L'UML est le résultat de la fusion de précédents langages de modélisation objet : Booch, OMT, OOSE. Principalement issu des travaux de GradyBooch, James Rumbaugh et Ivar Jacobson, UML est à présent un standard adopté par l'Object Management Group (OMG).

L'UML offre un standard de modélisation, pour représenter l'architecture logicielle. Les différents éléments représentables sont :

- Activité d'un objet/logiciel
- Acteurs
- Processus
- Schéma de base de données
- Composants logiciels
- Réutilisation de composants. **[22]**

4.2 Formalisme

UML 2.3 propose 14 types de diagrammes (25 en UML 1.3). UML n'étant pas une méthode, leur utilisation est laissée à l'appréciation de chacun, même si le diagramme de classes est généralement considéré comme l'élément central d'UML ; des méthodologies, telles que l'UnifiedProcess, axent l'analyse en tout premier lieu sur les diagrammes de cas d'utilisation (use case). De même, on peut se contenter de modéliser seulement partiellement un système, par exemple certaines parties critiques.

UML se décompose en plusieurs parties :

- **Les vues :** ce sont les observables du système. Elles décrivent le système d'un point de vue donné, qui peut être organisationnel, dynamique, temporel, architectural, géographique, logique, etc. En combinant toutes ces vues, il est possible de définir (ou retrouver) le système complet.

- **Les diagrammes :** ce sont des ensembles d'éléments graphiques. Ils décrivent le contenu des vues, qui sont des notions abstraites. Ils peuvent faire partie de plusieurs vues.
- **Les modèles d'élément :** ce sont les éléments graphiques des diagrammes.

4.2.1 Les vues UML

Une façon de mettre en œuvre UML est de considérer différentes vues qui peuvent se superposer pour collaborer à la définition du système :

- Vue des cas d'utilisation (use-case view) : c'est la description du modèle vu par les acteurs du système. Elle correspond aux besoins attendus par chaque acteur (c'est le quoi et le qui).
- Vue logique (logicalview): c'est la définition du système vu de l'intérieur. Elle explique comment peuvent être satisfaits les besoins des acteurs (c'est le comment).
- Vue d'implémentation (implementationview) : cette vue définit les dépendances entre les modules.
- Vue des processus (processview) : c'est la vue temporelle et technique, qui met en œuvre les notions de tâches concurrentes, stimuli, contrôle, synchronisation…
- Vue de déploiement (deploymentview) : cette vue décrit la position géographique et l'architecture physique de chaque élément du système (c'est le où).

4.2.2. Les principaux Diagrammesd'UML

Les diagrammes sont dépendants hiérarchiquement et se complètent, de façon à permettre la modélisation d'un projet tout au long de son cycle de vie.

*4.2.2.1***Diagrammes de structure ou diagrammes statiques :**

Les diagrammes de structure ou diagrammes statiquesrassemblent :

- **Diagramme de classes:** représentation des classes intervenant dans le système.
- **Diagramme d'objets:** représentation des instances de classes (objets) utilisées dans le système.
- **Diagramme de composants:** représentation des composants du système d'un point de vue physique, tels qu'ils sont mis en œuvre (fichiers, bibliothèques, bases de données…)
- **Diagramme de déploiement :** représentation des éléments matériels (ordinateurs, périphériques, réseaux, systèmes de stockage…) et la manière dont les composants du système sont répartis sur ces éléments matériels et interagissent entre eux.
- **Diagramme des paquets**: représentation des dépendances entre les paquets (un paquet étant un conteneur logique permettant de regrouper et d'organiser les éléments dans le modèle UML), c'est-à-dire entre les ensembles de définitions.

4.2.2.2Diagrammes de comportement

➢ Les diagrammes de comportementrassemblent :

- **Diagramme des cas d'utilisation :** représentation des possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire de toutes les fonctionnalités que doit fournir le système.
- **Diagramme états-transitions:** représentation sous forme de machine à états finis le comportement du système ou de ses composants.
- **Diagramme d'activité :** représentation sous forme de flux ou d'enchaînement d'activités le comportement du système ou de ses composants.
- **Diagramme d'interaction :**

➢ Les diagrammes d'interaction ou diagrammes dynamiquesrassemblent :

- **Diagramme de séquence:** représentation de façon séquentielle du déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs.
- **Diagramme de communication :** représentation de façon simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets (depuis UML 2.x).

- **Diagramme global d'interaction:** représentation des enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences (variante du diagramme d'activité) (depuis UML 2.x).

- **Diagramme de temps :** représentation des variations d'une donnée au cours du temps (depuis UML 2.3).[22]

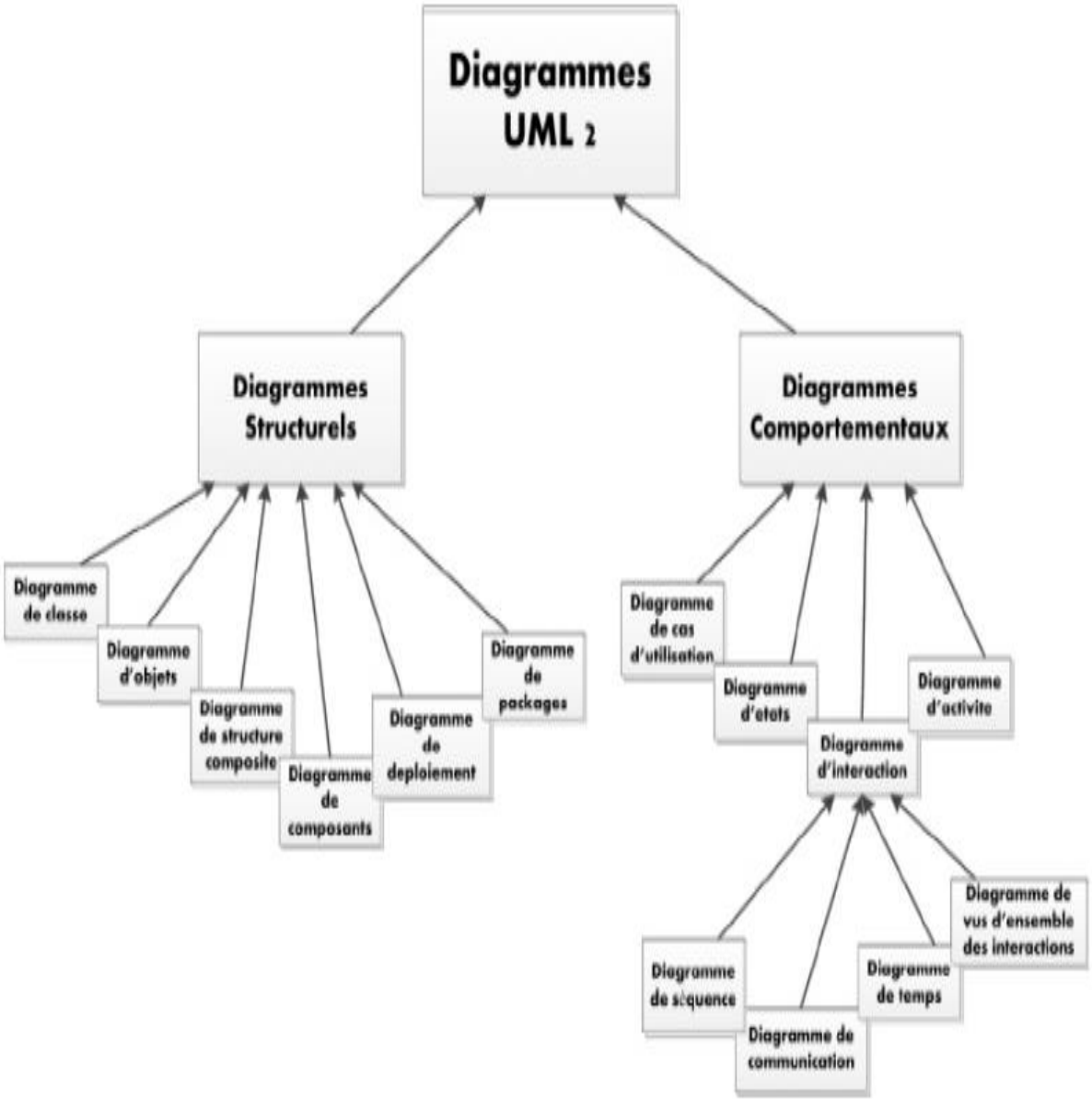


Figure 10: Les diagrammes d'UML[23]

4.3. Les cas d'utilisation (UC)

4.3.1. Identification des acteurs

Selon la nature des manipulations exercées sur le système expert on peut distinguer deux types d'utilisateurs :

- **Utilisateur simple :** En introduisantles faits au système expert pour diagnostiquer la panne et obtenir la solution. il peut être un informaticien, un magasinier, un vendeur de matériel informatique.

- **Expert (spécialiste de domaine de la maintenance informatique) :** il est chargé de la mise à jour de la base de règles de production.

4.3.2. Identification des cas d'utilisations :

Ces les séquences d'actions principales effectuées par l'utilisateur qui sont :

- S'Authentifier.
- Ajouter une / des règles ou faits.
- Modifier une / des règles ou faits.
- Supprimer une / des règles ou faits.
- Etablir le diagnostic.

4.3.3. Schéma de diagramme de cas d'utilisation :

Est une représentation graphique qui illustre les cas d'utilisations reliés par des lignes à leurs acteurs.

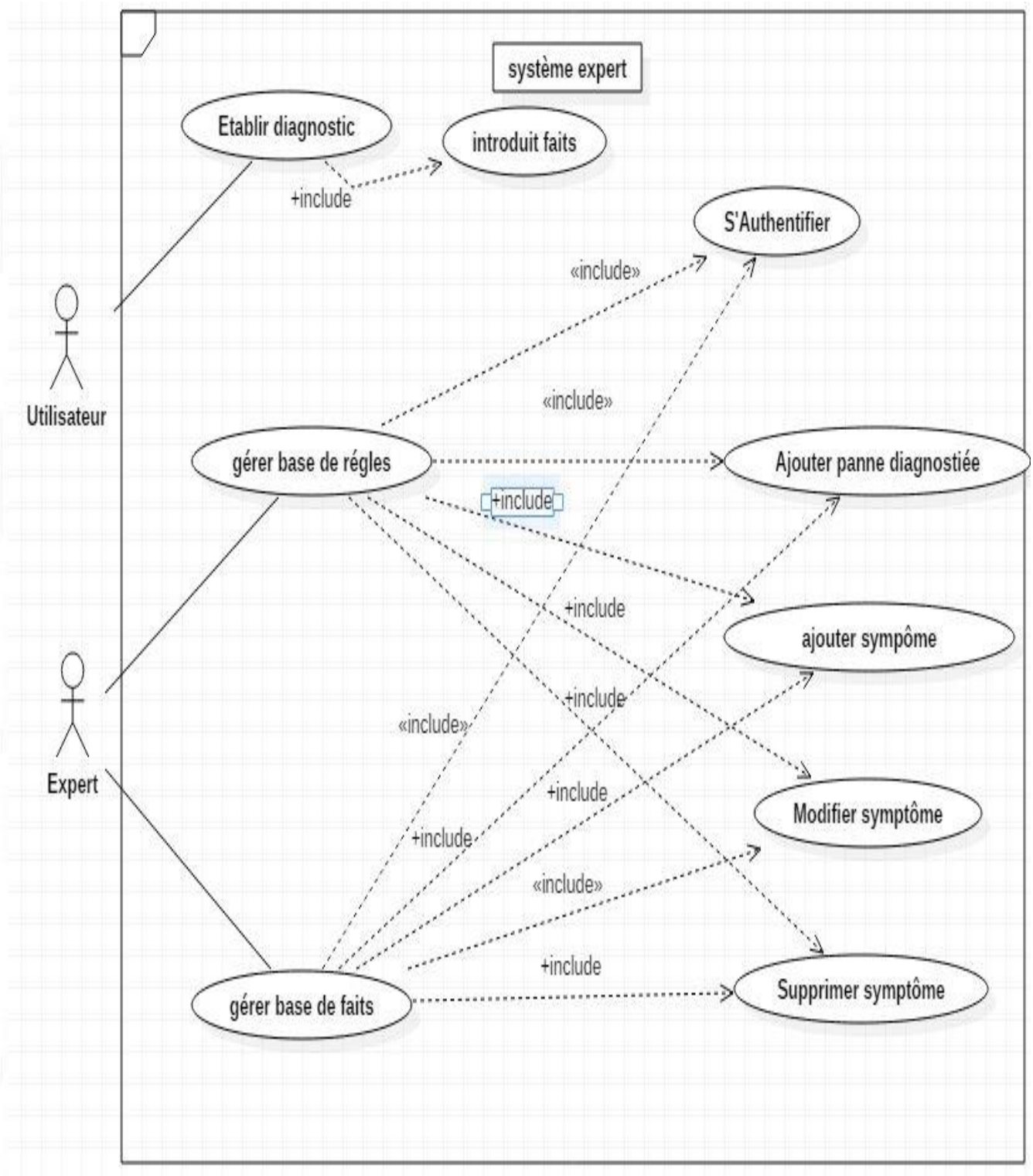


Figure 11 : diagramme de cas d'utilisation.

4.3.4. Description textuelle des cas d'utilisation

Nous allons décrire de l'interaction entre les acteurs et le système : il s'agit de décrire la chronologie des actions qui devront être réalisées par les acteurs et par le système lui-même. On parle d'ailleurs de scénarios.

La description d'un cas d'utilisation permet de :

- clarifier le déroulement de la fonctionnalité ;
- décrire la chronologie des actions qui devront être réalisées ;
- d'identifier les parties redondantes pour en déduire des cas d'utilisation plus précises qui seront utilisées par inclusion, extension ou généralisation/spécialisation. Et oui, dans ce cas nous réaliserons des itérations sur les diagrammes de cas d'utilisation ;
- d'indiquer d'éventuelles contraintes déjà connues et dont les développeurs vont devoir tenir compte lors de la réalisation du logiciel. Ces contraintes peuvent être de nature diverse.

Les descriptions peuvent aider à découvrir d'autres cas d'utilisation que l'on pourrait ajouter. Il s'agit, dans ce cas, d'une nouvelle itération sur les diagrammes de cas d'utilisation.**[24]**

Par exemple :
Si une personne n'est pas encore « cliente » sur la boutique en ligne et qu'elle souhaite réaliser un achat :

- Elledoit s'inscrire avant de commencer la procédure d'achat ?

On réalise alors une fiche descriptive pour chaque cas d'utilisation, de façon à raconter l'histoire du déroulement des différentes actions.

Cette fiche descriptive doit comporter 4 volets :

1. L'identification
2. La description des scénarios
3. La fin et les post-conditions
4. Les compléments

NB :

- Gérer la base de règles contient les cas utilisations : ajouter règle, modifier règle, supprimer règle.

- Gérer la base de faits contient les cas utilisations : ajouter fait, modifier fait, supprimer fait.

Description du cas « Ajouter règle »

Identification

Nom de cas : Ajouter règle.

But : citer les étapes qui permettront à l'expert d'ajouter une nouvelle règle dans la base de règles.

Acteur principal : L'expert.

Séquencement

Ce cas d'utilisation débutera une fois que l'utilisateur aura atteint l'interface permettant d'ajouter des règles à la base de règles.

Préconditions :

- Au préalable l'expert doit s'authentifier.

Enchainement nominal

1. Le système conduit l'expert à l'interface permettant de rajouter une règle.
2. L'expert choisit l'opération« Ajouter règle ».
3. Le système affiche un formulaire d'ajout de règle (Ajouter symptômes)
4. L'expert ajoute une nouvelle règle puis l'enregistre dans la base de règles.
5. Mise à jour de la base par le système.

Enchainements alternatifs

A1 : La règle existe déjà

L'enchainement démarre après le point (4) de la séquence nominale

5. le système indique que la règle ajoutée existe déjà dans la base de règles.

A1.1 : Ajouter une nouvelle règle

La séquence nominale reprend au point (3).

A1.2 : Quitter l'opération d'ajout de règles.

Post-conditions

- La base de règle sera mise à jour et nous pourrons constater sa présence dans la base de règles

Tableau 9. Description du cas « Ajouter règle».

Description du cas « Modifier règle»

Identification

Nom de cas : Modifier règle.

But : montrer les étapes permettant à l'expert de modifier la base de règles.

Acteur principal : L'expert.

Séquencement

Le cas d'utilisation débutera lorsque l'expert atteindra l'interface permettant de modifier des règles.

Préconditions :

- l'expert doit s'authentifier préalablement.
- au moins une règle doit être existée dans la base des règles pour que la modification soit valide.

Enchainement nominal

1. Le système affiche la session de l'expert.
2. L'expert choisit l'opération Modifier règle.
3. Le système affiche l'interface Modifier règle qui contient toutes les règles et leurs numéros, pour que l'expert puisse sélectionner la règle à modifier sans que sa ne crée d'ambigüités.
4. L'expert sélectionne la règle à modifier.
5. Le système permet à l'expert de modifier la règle sélectionnée.
6. L'expert modifie les symptômes de la règle puis enregistre la modification.
7. Le système met à jour la base de règles.

Enchainements alternatifs

- aucun

Post-conditions

La base de règle devra obligatoirement être mise à jour.

Tableau 10. Description du cas « Modifier règle»

Description du cas « Supprimer règle »
Identification **Nom de cas :** Supprimer règle. **But :** citer les étapes permettant à l'expert de supprimer une règle dans la base de règles. **Acteur principal :** L'expert.1
Séquencement Le cas commence lorsque l'expert atteint la fenêtre Supprimer règle. **Préconditions :** - l'expert doit s'authentifier préalablement. - La base des règles doit contenir au moins une règle pour que la suppression puisse avoir lieu. **Enchainement nominal** 1. Le système affiche la session de l'expert. 2. L'expert choisit l'opération Supprimer règle. 3. Le système affiche l'interface supprimer règle qui contient toutes les règles et leurs numéros, pour que l'expert puisse sélectionner la règle à supprimer sans que sa ne crée d'ambigüités. L'expert peut sélectionner une règle à supprimer. 5. L'expert sélectionne la règle puis confirme la suppression. 6. Le système supprime la règle et met à jour la base de règles. **Enchainements alternatifs** -Aucune **Post-conditions** - La base de règle devra obligatoirement être mise à jour et la règle doit être obligatoirement supprimée.

Tableau 11. Description du cas « Supprimer règle».

NB : *avec la même manière on fait la mise à jour de la base de faits.*

Description du cas « Etablir diagnostic »
Identification **Nom de cas :** Etablir diagnostic **But :** Décrire à l'utilisateur une série d'étapes àsuivre afin de remplir le formulaire que propose le système relativement aux symptômes qu'il a inséré, tout ceci dans le but de faire un diagnostic et de détecter le composant en panne. **Acteur principal :** L'utilisateur simple, Expert.
Séquencement Ce cas d'utilisation commence une fois que l'utilisateur aura atteint la session utilisateur. **Préconditions :** Au départ la base de faits doit être vide. **Enchainement nominal** 1. Le système affiche un Questionnaire à remplir contenant la question par catégories suivantes: Bipssonores,écran Bleu, autres pannes fréquentes. 2. remplissage de questionnaire et cliquement sur diagnostic par l'utilisateur. 3. Déduction de la panne correspondante aux symptômes par le système en donneront le matériel défectueux. **Enchainements alternatifs** A1 : Aucune panne ne correspond aux symptômes introduits L'enchainement démarre après le point (2) de la séquence nominale 3. D'autres alternatives de panne de matériel sont indiquées par le système. **Post-conditions** - Le système devra afficher un résultat mêmeapprocsimatif.

Tableau12: Description du cas « Etablir diagnostic »

4.4.Diagramme de séquence

Le diagramme de séquence permet de montrer les interactions d'objets dans le cadre d'un scénario d'un Diagramme des cas d'utilisation. Dans un souci de simplification, on représente l'acteur principal à gauche du diagramme, et les acteurs secondaires éventuels à droite du système. Le but étant de décrire comment se déroulent les actions entre les acteurs ou objets.

La dimension verticale du diagramme représente le temps, permettant de visualiser l'enchaînement des actions dans le temps, et de spécifier la naissance et la mort d'objets. Les périodes d'activité des objets sont symbolisées par des rectangles, et ces objets dialoguent à l'aide de messages.

✓ *Dialogue entre les objets*

Plusieurs types de messages (actions) peuvent transiter entre les acteurs et objets.

- message simple : le message n'a pas de spécificité particulière d'envoi et de réception.
- message avec durée de vie : l'expéditeur attend une réponse du récepteur pendant un certain temps et reprend ses activités si aucune réponse n'a lieu dans un délai prévu.
- message synchrone : l'expéditeur est bloqué jusqu'au signal de prise en compte par le destinataire. Les messages synchrones sont symbolisés par des flèches barrées.
- message asynchrone : le message est envoyé, l'expéditeur continue son activité que le message soit parvenu ou pris en compte ou non. Les messages asynchrones sont symbolisés par des demi-flèches.
- message dérobant : le message est mis en attente dans une liste d'attente de traitement chez le récepteur.

Le langage permet de décaler l'envoi et la réception des messages, pour montrer les délais de communication non négligeables. La plupart des ateliers UML ne prennent cependant pas en compte cette spécificité.

✓ *Cadres d'interaction*

Pour les cas plus complexes, on peut intégrer des algorithmes dans les diagrammes de séquences. Par le biais de cadres d'interaction, on peut préciser les opérantes d'un ensemble de messages :

- **alt** : fragments multiple alternatifs (si alors sinon)
- **opt** : fragment optionnel
- **par** : fragment parallèle (traitements concurrents)
- **loop** : le fragment s'exécute plusieurs fois
- **region** : région critique (un seul thread à la fois)
- **neg** : une interaction non valable
- **break** : représente des scenario d'exception
- **ref** : référence à une interaction dans un autre diagramme
- **sd** : fragment du diagramme de séquence en entier.**[25]**

4.4.1. Schéma de diagramme de séquence de cas d'utilisation (S'Authentifier) :

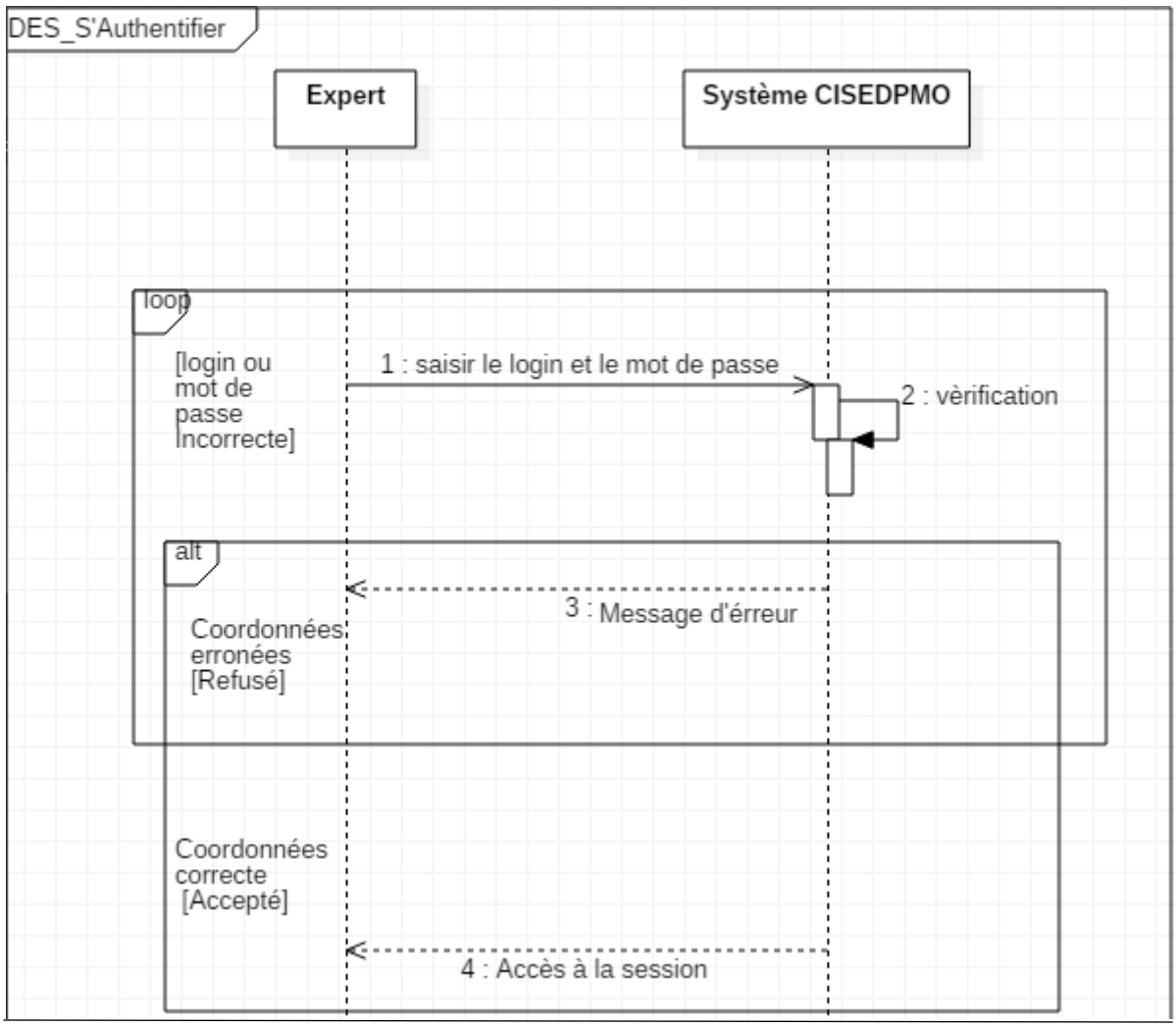


Figure 12 : Diagramme de séquences de cas d'utilisation « S'Authentifier »

4.4.2. Diagramme de séquencede cas d'utilisation(Etablir diagnostic)

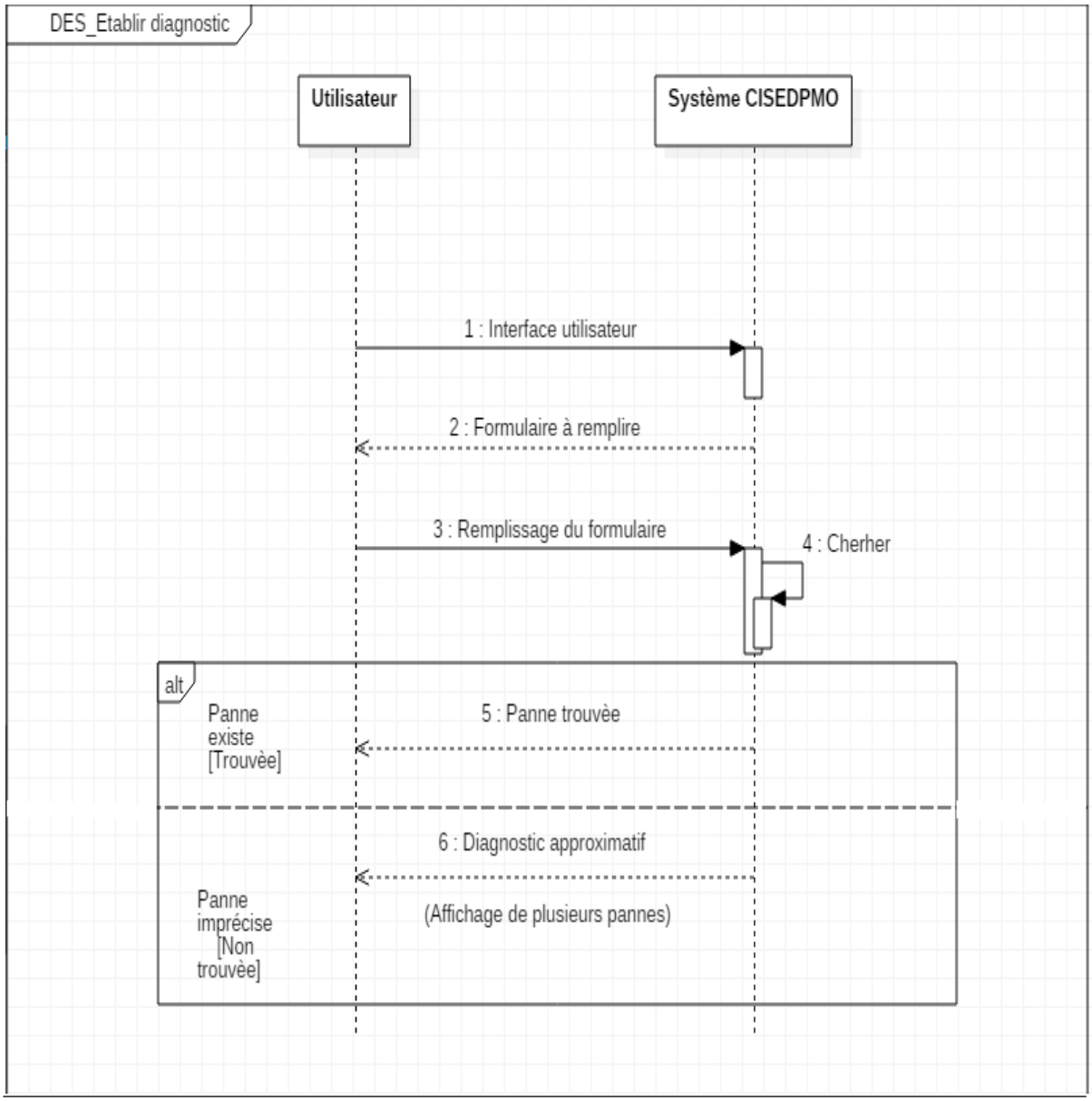


Figure 13 : Diagramme de séquence de cas d'utilisation « Etablir diagnostic »

4.4.3. Diagramme de séquence de cas d'utilisation (Ajouter règle)

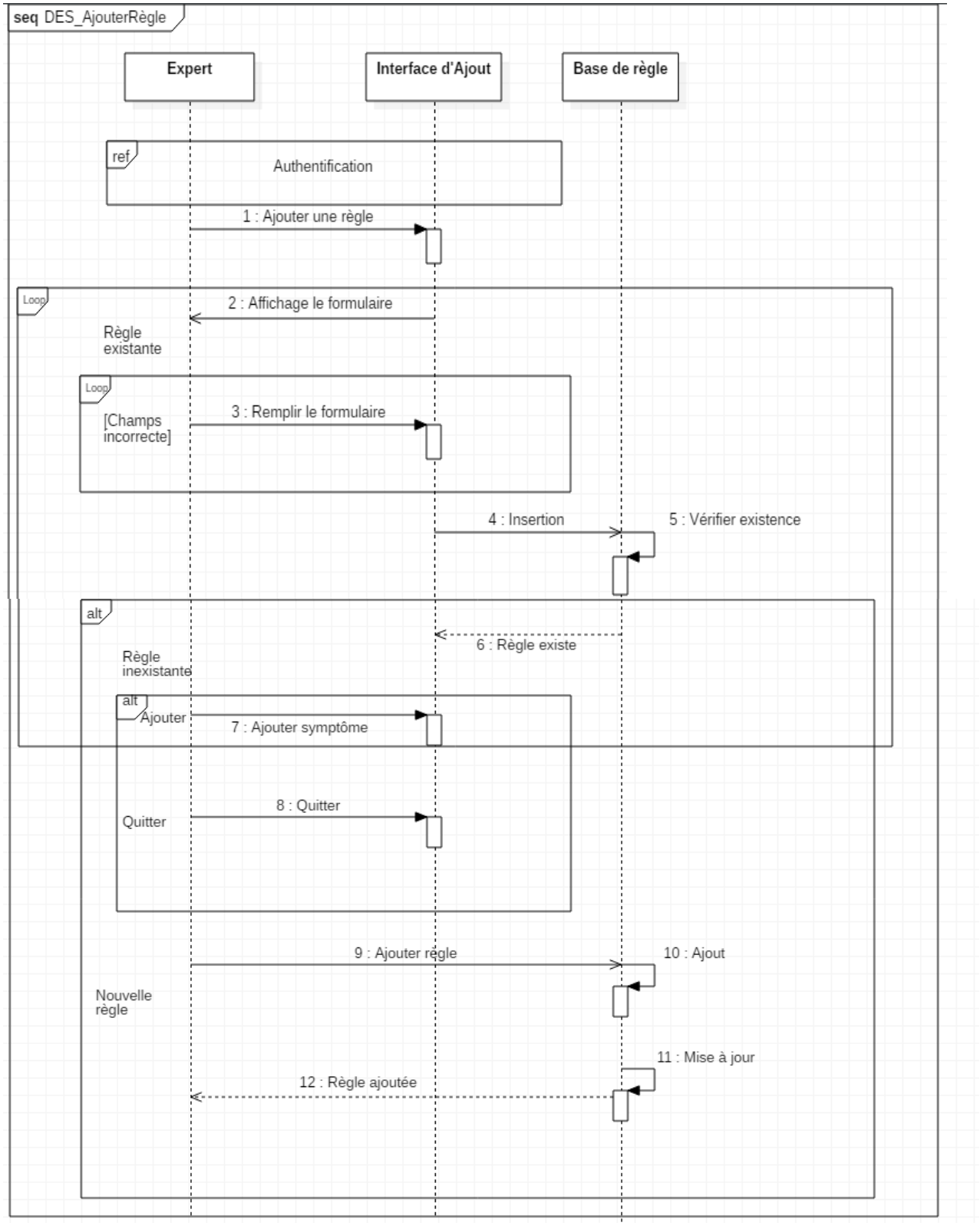


Figure 14: Diagramme de séquence de cas d'utilisation « Ajouter règle »

4.5. Diagramme de classes

Ce diagramme est réalisé par un outil nommé StarUML..

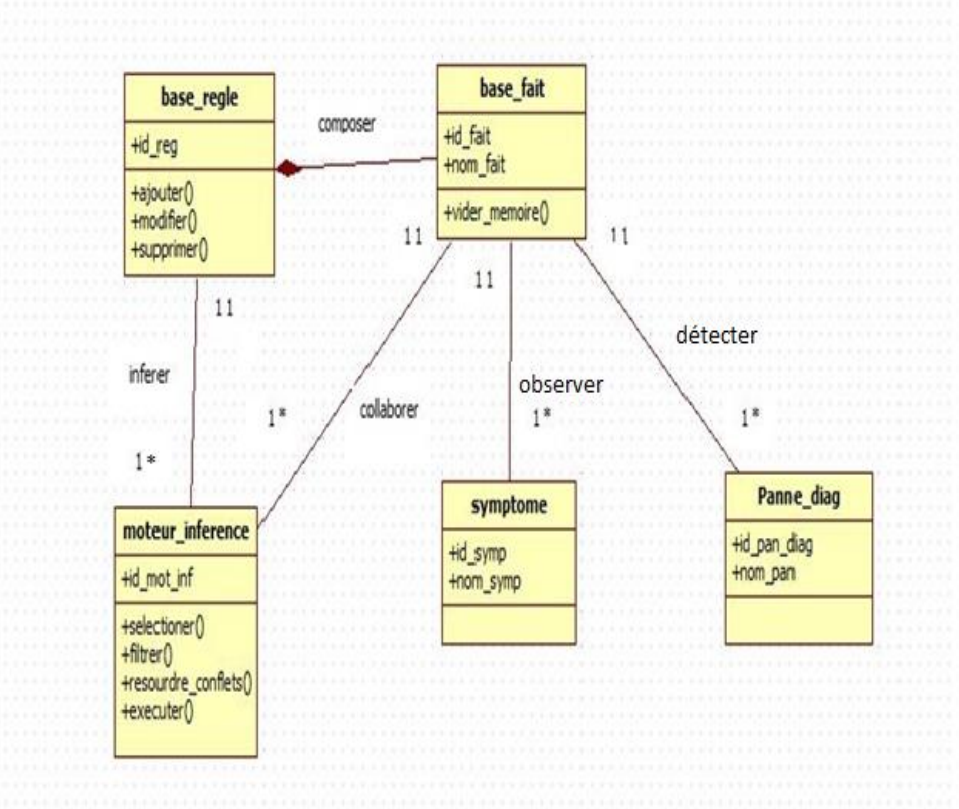


Figure 15 : Diagramme de classes

4.5.1. Dictionnaire de données épuré

Codification	Désignation	Type	Taille
Id_faitbf	Identifiant symptôme dans la BF	Int	05
Nom_fait	Nom du symptôme dans la BF	Varchar	500
Id_symp	Identifiant du symptôme	Int	05
Id_règle	Identifiant de la règle	Int	05
NomSymp	Nom du Symptôme	Varchar	500
Id_pan_diag	Identifiant de la panne diagnostiquée	Int	05
Nom_pan_diag	Nom de la panne diagnostiquée	Varchar	500
Id	Identifiant du moteur d'inférence	Int	01

Tableau 13 .Dictionnaire de données.

5.Modèle relationnel de données

5.1. Règles de passage du modèle de classes au modèle relationnel

Les règles de passage d'un diagramme de classes vers le schéma relationnel sont les suivantes :

- ✓ Toute entité donne naissance à une table dont la clé primaire est l'identifiant de l'entité.
- ✓ Toute association binaire (1..1) à (1..*) donne naissance à une table dérivée de l'entité du cardinalité (1..1), et sa clé primaire est déposé comme clé étrangère dans la table dérivée du cardinalité (1..*).
- ✓ Toute association binaire (1..*) à (1..*) avec une entité, donne naissance à trois tables, dont la table dérivée de l'entité association possèdera les clés primaires des deux autres tables comme clé primaire.
- ✓ Toute association binaire (1..1) à (1..1) pour le cas de participation obligatoire de part et de l'autre c'est-à-dire la table fils possèdera la clé primaire de la table père comme clé étrangère.
- ✓ Toute composante liée a une composite donne naissance à deux tables, dont la table dérivée de l'entité composante possédera la clé primaire de la table dérivée de l'entité composite comme clé.
- ✓ Dans le cas d'agrégation de composition donnera naissance à deux table dont la table agrégée possédera la clé primaire de la deuxième table de comme clé étrangère.
- ✓ Dans le cas d'héritage donnera naissance à deux tables, dont la table dérivée de l'entité héritasnte possédera la clé primaire de la table dérivée de l'entité héritée comme clé primaire.**[23]**

En appliquant ces règles de passages concernant la de gestion de pièce de rechange, nous avons obtenu le schéma relationnel suivant :

Base_Règle(**id_regle**, symptome,**Id_pannediag,id_mot_inf**).
Panne_diag(**id_pannediag**, nom_panneDiag).

Base_faits(Id_faitbf ,Nomfaitbf).

Symptome(**id_regle**,**Id_pannesu**).

Moteur_inférence(Id_faitbf ,**id_mot-inf**).

NB: Les clés primaires sont en gras soulignées alors que les clés étrangères sont en gras non soulignées.

Conclusion

Dans ce chapitre on a modélisé les processus de système à l'aide de quelques diagrammesd'UML(diagramme de cas d'utilisation, diagrammes de séquence, diagramme de classes) et on aaccomplis la conception de système qui nous permettra la réalisation de l'application dans le chapitre qui suit.

CHAPITRE III

Réalisation de système expert

Introduction

Parmi les technologies existantes nous allons choisir Java netbeans IDE afin de réaliser notre application.A cet effet nous commenceronspar l'implémentation (la programmation évènementielle)de code nécessaire pour obtenir les interfaces de l'application pour que l'utilisateur puisse interagir avec le système expert.
Puis nous allons prendre soins de design et de l'ergonomie de l'interfaces dont le but qu'elle soit agréable aux utilisateurs.

1. Plate-forme de développement Java et la raison de choix

Les raisons qui nous ont conduits à choisir finalement Java comme langage de programmation sont reprises ci-dessous :

- **Portabilité** : s'exécute presque n'importe où (Windows, Linux, Unix, Mac OS, téléphones mobiles, robots,...). Le slogan de JAVA est "Write Once runEverywhere

- **Extensibilité** : il existe de nombreuses bibliothèques et Frameworks tierces, chacun spécialisé dans un domaine spécifique. Il est donc rare de manquer la bibliothèque appropriée pour les besoins courants.

- **Orientée Objet** : l'approche objet permet de bien concevoir les applications orientées métiers.

- **Disponibilité des outils** : Java est un langage libre, ce qui fait que de nombreux éditeurs créent des outils de développement souvent libre. Ainsi, de nombreux IDE, serveurs web et serveurs d'application sont disponibles gratuitement sur le marché.

1.1. Présentation de java

Java est un langage de programmation à usage général, évolué et orienté objet dont la syntaxe est proche du C. Ses caractéristiques ainsi que la richesse de son écosystème et de sa communauté lui ont permis d'être très largement utilisé pour le développement d'applications de types très disparates. Java est notamment largement utilisépour le développement d'applications d'entreprises et mobiles.

1.1.1. Les caractéristiques

Java possède un certain nombre de caractéristiques qui ont largement contribué à son énorme succès :

Java est interprété	le source est compilé en pseudo code ou bytecode puis exécuté par un interpréteur Java : la Java Virtual Machine (JVM). Ce concept est à la base du slogan de Sun pour Java : WORA (Write Once, RunAnywhere : écrire une fois, exécuter partout). En effet, le bytecode, s'il ne contient pas de code spécifique à une plate-forme particulière peut être exécuté et obtenir quasiment les mêmes résultats sur toutes les machines disposant d'une JVM.
Java est portable : il est indépendant de toute plate-forme	il n'y a pas de compilation spécifique pour chaque plateforme. Le code reste indépendant de la machine sur laquelle il s'exécute. Il est possible d'exécuter des programmes Java sur tous les environnements qui possèdent une Java Virtual Machine. Cette indépendance est assurée au niveau du code source grâce à Unicode et au niveau du bytecode.
Java est orienté objet.	comme la plupart des langages récents, Java est orienté objet. Chaque fichier source contient la définition d'une ou plusieurs classes qui sont utilisées les unes avec les autres pour former une application. Java n'est pas complètement objet car il définit des types primitifs (entier, caractère, flottant, booléen,...).
Java est simple	le choix de ses auteurs a été d'abandonner des éléments mal compris ou mal exploités des autres langages tels que la notion de pointeurs (pour éviter les incidents en manipulant directement la mémoire), l'héritage multiple et la surcharge des opérateurs, ...
Java est fortement typé	toutes les variables sont typées et il n'existe pas de conversion automatique qui risquerait une perte de données. Si une telle conversion doit être réalisée, le développeur doit obligatoirement utiliser un cast ou une méthode statique fournie en standard pour la réaliser.
Java assure la gestion de la mémoire	l'allocation de la mémoire pour un objet est automatique à sa création et Java récupère automatiquement la mémoire inutilisée grâce au garbage collector qui restitue les zones de mémoire laissées libres suite à la destruction des objets.
Java est sûr	la sécurité fait partie intégrante du système d'exécution et du compilateur. Un programme Java planté ne menace pas le système d'exploitation. Il ne peut pas y avoir d'accès direct à la mémoire. L'accès au disque dur est réglementé dans une applet. Les applets fonctionnant sur le Web sont soumises aux restrictions suivantes dans la version 1.0 de Java : • aucun programme ne peut ouvrir, lire, écrire ou effacer un fichier sur le système de l'utilisateur • aucun programme ne peut lancer un autre programme sur le système de l'utilisateur • toute fenêtre créée par le programme est clairement identifiée

	comme étant une fenêtre Java, ce qui interdit par exemple la création d'une fausse fenêtre demandant un mot de passe • les programmes ne peuvent pas se connecter à d'autres sites Web que celui dont ils proviennent.
Java est économe	le pseudo code a une taille relativement petite car les bibliothèques de classes requises ne sont liées qu'à l'exécution.
Java est multitâche	il permet l'utilisation de threads qui sont des unités d'exécutions isolées. La JVM, elle même, utilise plusieurs threads.

*Tableau 14 : caractéristiques de java.***[27]**

1.1.2. Java netbeans IDE

Est un environnement de développement intégrés (IDE : Integrated Développent Environnent) est un ensemble d'outils qui en plus des tâches classiques d'un langage de Programmation offre des fonctionnalités étendues permettant de couvrir une plus large partie du cycle de création d'un logiciel. On peut distinguer des tâches telles que :

- Compilation et débogage
- Déploiement de l'application.
- Intégrer des outils de test et de vérification.
- Outil de création graphique (icône).
- Interface avec les SGBD.

2. Le serveur local XAMPP

XAMPP est un ensemble de logiciels permettant facilement de créer une interface interagissant avec une base de données SQL.
X pour cross-plateforme (LAMPP pour Linux, WAMPP pourWindows,...)
A pour Apachequi est un serveur http.
M pour MySQL qui est un système de gestion de bases de données SQL.
P pour PHPquiest un langage de programmation pour le web.
P pour Perlqui est un langage de programmation.

3. Programmation structurelle et programmation événementielle

On distingue deux types de programmation, chacune est spécialiséedans un domaine d'application donné et chacune possède un type spécifique :

- La Programmation structurée ou modulaire : le programme est vu comme un ensemble d'unités structurés hiérarchiquement. Il existe une interaction entre les modules et on fait généralement distinction entre les données et les traitements.

- La Programmation événementielle : Les composants d'une application événementielle c'est à dire les objets interagissent entre eux et avec l'environnement. Ils communiquent en réponse à des événements. Ces événements peuvent correspondre à une action de l'utilisateur : un click sur un bouton de commande, une écriture dans une zone de texte, un choix dans une case d'option ou une case à cocher, le déplacement d'un objet, ... Ils peuvent aussi être déclenchés par le système : chargement d'une feuille, un top déclenché par l'horloge.

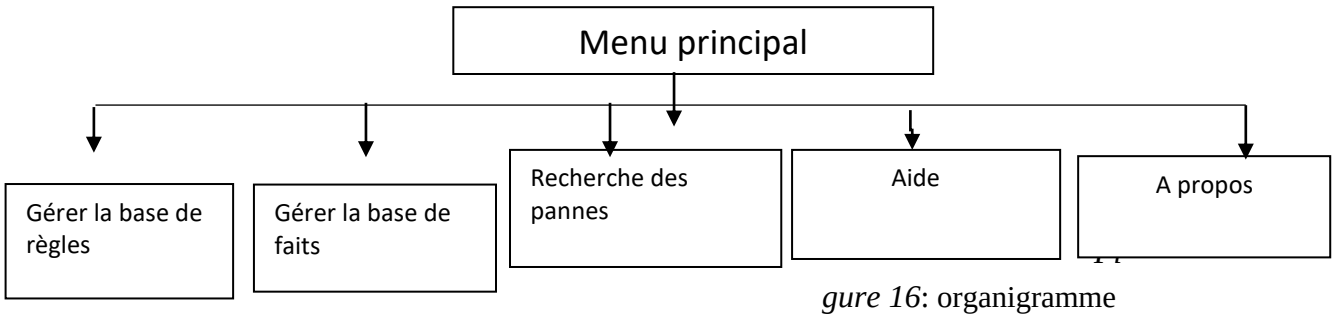
Les événements sont captés par le système d'exploitation, le traitement consiste en l'exécution des procédures événements associées à celui-ci s'il en existe. C'est le programmeur qui doit prévoir la procédure à exécuter en réponse à un événement donné. Par exemple, le déclenchement de l'événement click sur un bouton quitter doit terminer l'exécution, le choix d'un élément dans un menu doit déclencher certaines opérations, un top d'horloge doit modifier le contenu d'une zone d'image.[28]

3.1.Les concepts de base et les composants standards pour développer des applications événementielles

- Définir la notion de conteneur
- Présenter les principes de placement des contrôles
- Échanger des informations entre fenêtres
- Instanciation statique (design time) et dynamique (run time) des contrôles
- Utiliser les contrôles simples (texte, bouton, cases à cocher,…)
- Utiliser les contrôles de type liste (sélection, vue, images, …)
- Utiliser les contrôles hiérarchiques (vue arborescente)
- Utiliser les barres d'états et barre d'outils, les menus principaux et contextuels

3. Organigramme de l'application

IL constitue des menus et sous menus.



gure 16: organigramme de l'application.

5. Description de l'interface de notre application

On présente ici quelques fenêtres de notre application « MicroExpert »

5.1. Fenêtre Menu principal

Il représente le portail de l'application qui permet aux utilisateurs d'accéder aux autres fenêtres de l'application :

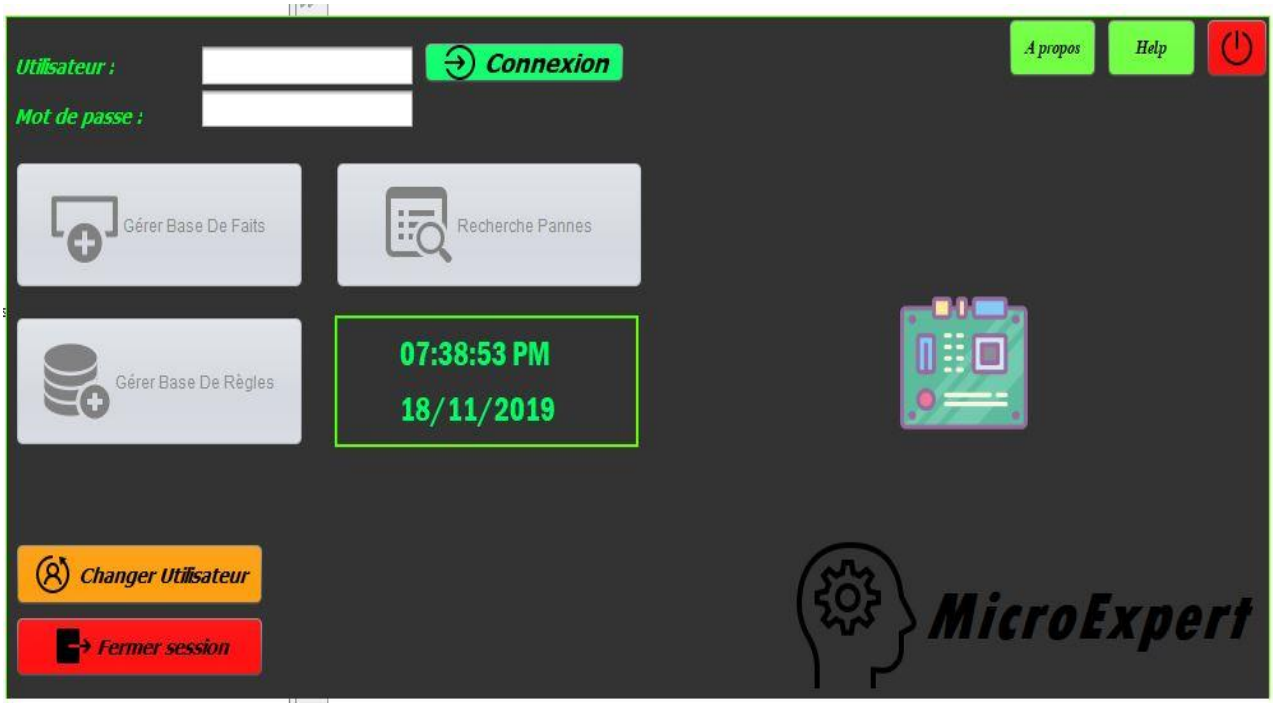


Figure 17 :Fenêtre Menu principal.

5.2. Fenêtre Gérer Base de règles

Cette interface permet d'ajouter, modifier et supprimer la /les règles de la base de règles.

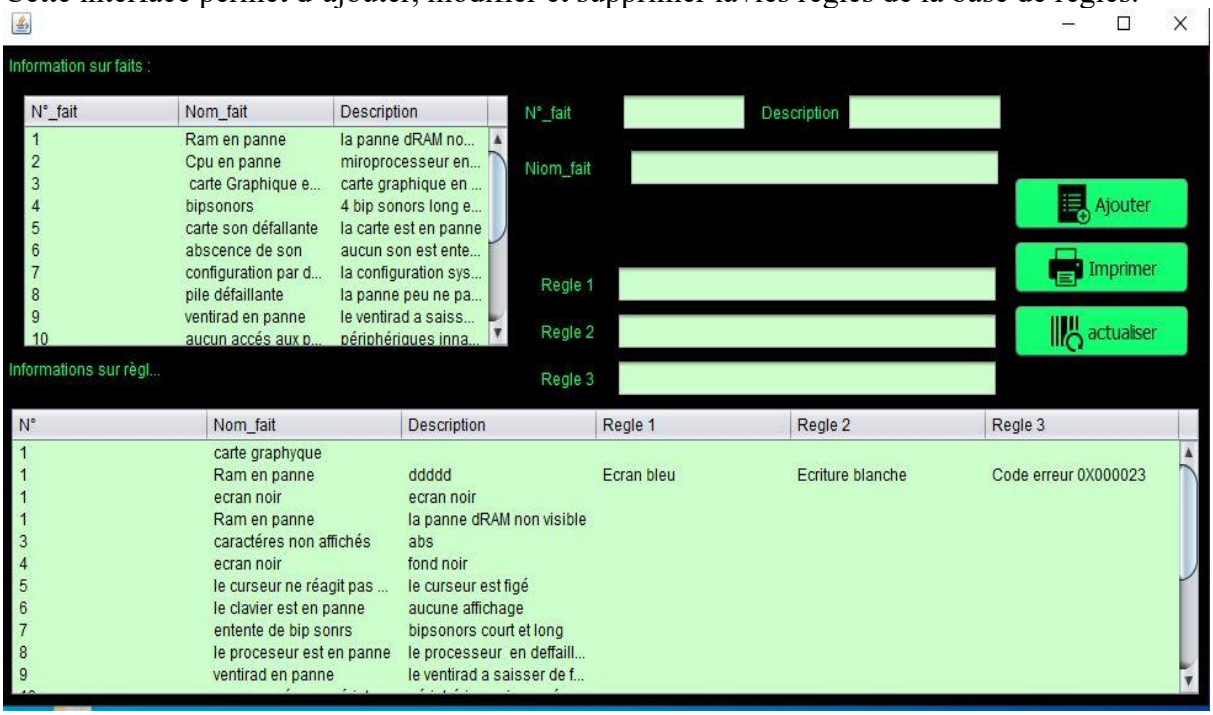


Figure 18 :Fenêtre Gérer Base de règles.

5.3.FenêtreGérer Base de faits

Cette interface permet d'ajouter, modifier et supprimer la /les faits de la base de faits.

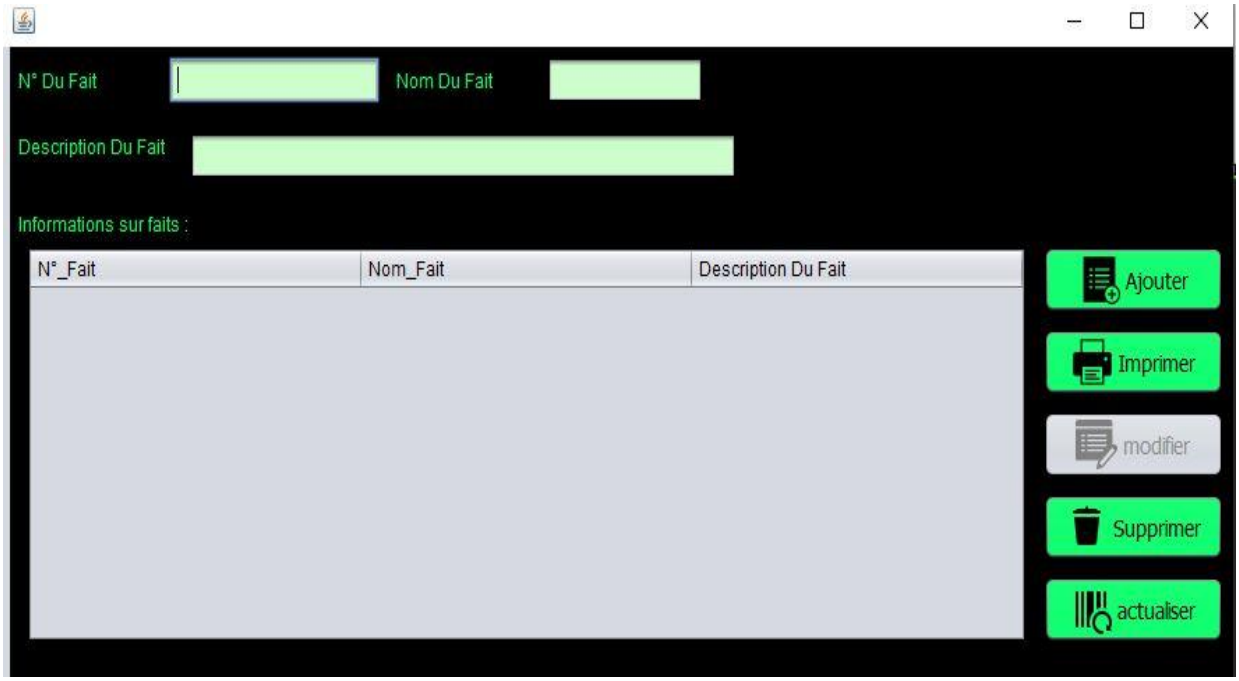


Figure 19 :Fenêtre Gérer Base de faits.

5.4.FenêtreRecherche et inférence

Dans cette fenêtre l'utilisateur introduit les symptômes ou le matériel visible abimé observés et le moteur d'inférence lui renvoie la panne (résultat).

Figure 20:Fenêtre Recherche et inférence.

6. Choix de la représentation des connaissances et le mode de raisonnement

La base de connaissances (base de règles et base de faits) est représentée par le modèle logique de données, afin de mettre en place le moteur d'inférence en mode de chainage avant.

Conclusion

Dans ce chapitre nous avons utilisés la plateforme netbeans IDE soutenu par un serveur local XAMPP pour créer l'interface de notre application, comme on aopté au langage de programmation java pour implémentés le code nécessaire pour la construction de moteur d'inférence afin d'identifier l'origine la panne qui a causé les symptômes entrer par l'utilisateur.

Conclusion générale

Maintenant, que notre travail est achevé, nous espérons bien que notre application aide les utilisateurs à diagnostiquer les pannes de leurs PC dans les meilleurs délais. Cette application représente un bon début pour nous dans le domaine des systèmes experts.

En premier lieu nous avons définis l'intelligence artificielle et les systèmes experts, puis nousavons collecté toutes les informations concernant notre domaine d'expertise (symptômes et pannes des composants d'un PC). En suite pour la conception de notre système nous avons utilisé L'UML qui est un langage adéquat a notre cas et un outil idéal pour la programmation orienté objet, car il nous a permet de modéliser toutes nos idées a travers les différents diagrammes : cas d'utilisation, séquence, classes … comme elle nous a étendu le fils jusqu'à ce que nous avons concrétisé ces idées par la réalisation de notre application.

En deuxième lieu afin de réaliser notre application nous avons choisis la technologie java netbeans IDE vu ces avantages qui nous a facilité la construction des interfaces ergonomiques grasse a la programmation événementielle, comme on a utilisé un serveur local (XAMPP) pour but de gérer la base de connaissances.

En dernier lieu pour répondre aux besoins fonctionnels (identifier l'origine de la panne à travers certain symptômes) des utilisateurs nous avons procéder à la programmation procédurale (chainage avant) en java.

Cependant des améliorations restent à envisager pour but de perfectionner notre application.

Bibliographie

Bibliographie :

[1] https://www.futura-sciences.com/tech/definitions/informatique-intelligence-artificielle-555/. consulté le 04/05/2019.

[2] https://www.nouvelobs.com/societe/20180228.OBS2855/dans-ces-8-domaines-l-intelligence-artificielle-va-changer-nos-vies.html. consulté le 04/05/2019.

[3] http://intelart.e-monsite.com/pages/ii-l-utilisation-de-l-ia/domaines-d-applications.html consulté le 04/05/2019.

[4] http://deptinfo.cnam.fr/Enseignement/CycleSpecialisation/IAB/Docs/Systemes_Experts-12.pdf consulté le 04/05/2019.

[5] http://deptinfo.cnam.fr/Enseignement/CycleSpecialisation/IAB/Docs/Systemes_Experts-12.pdf consulté le 06/05/2019.

[6] https://slideplayer.fr/slide/2579472/ consulté le 06/05/2019.

[7] https://perso.liris.cnrs.fr/alain.mille/enseignements/DEA-ECD/site_ia_emiage/session1/syst%E8mes_experts.htm.consulté le 07/05/2019.

[8] https://perso.liris.cnrs.fr/alain.mille/enseignements/DEA-ECD/site_ia_emiage/Session2/session2.htm#%C2%A72consulté le 07/05/2019.

[9] Pav91.e-monsite.com/pages/chainage-avant-arriere-mixte.html. consultéle 07/05/2019.

[10] François denis laboratoire d'informatique fondamentale, Marseille Laurent miclet ENSSAT-IRISA, Lannion « INTELLIGENCE ARTIFICIELLE : LES SYSTEMES EXPERTS ».-juillet 2009.

[11] Daston, Lorraine et Galison, Peter (en), "The Image of Objectivity [archive]", Représentations (en), autumn 1992, nº 40, p. 81-128.

-Bachimont, Bruno, Engagement sémantique et engagement ontologique: conception et réalisation d'ontologies en ingénierie des connaissances [archive] ; Ingénierie des connaissances : évolutions récentes [archive], 200.

[12] Lindsay et Norman (1980). Traitement de l'information et comportement humain. Ed. Etudes Vivantes
Alan Baddley (1993). La mémoire humaine. Théorie et pratique. Presses universitaires de Grenoble . p343-381.

[13] https://fr.wikipedia.org/wiki/Logique_d%27ordre_sup%C3% .consulté le

10/05/2019.

[14]https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificielscons

ulté le 10/05/2019.

[15] P. Jackson. Introduction to Expert Systems. Addition-wesley, 1986 consulté le 10/05/2019.

[16]https://hdd34.developpez.com/cours/artpoo/#L1.3.3. Consulté le 04/05/2019.

[17] http://dictionnaire.sensagent.leparisien.fr/heritage 20% informatique/fr-fr/. consulté le 15/05/2019.

[18]http://dictionnaire.sensagent.leparisien.fr/Polymorphisme%20(informatique)/fr-fr/consulté le 15/05/2019.

[19]http://programmationjava.1sur1.com/Java/Tutoriels/HeritagePolymorphisme/HeritagePolymorphisme.php.consulté le 17/05/2019.

[20] https://sites.google.com/site/iadescartes/2-les-systemes-experts/2-1-les-syt/2-1-3-le-moteur-d-inference-un-systeme-en-deux-phases.consulté le 20/05/2019.

[21] https://www.imedias.pro/cours-en-ligne/informatique/ordinateur/composants-ordinateur/consulté le 20/05/2019.

[22] https://fr.wikipedia.org/wiki/UML_(informatique) consulté le 21/05/2019.

[23] https://support.hp.com/fr-fr/document/bfh07107. consulté le 28/05/2019

[24] https://openclassrooms.com/fr/courses/2035826-debutez-lanalyse-logicielle-avec-uml/2094981-la-description-textuelle-d-un-cas-d-utilisation consulté le 21/05/2019.

[25] Laurent Audibert, « Diagramme de séquence » [archive], sur developpez.com. consulté le 25/05/2019.

[26] https://www.ult.bi/?q=student/chapitre-v-realisation-de-

l%E2%80%99application consulté le 02/06/2019.

[27] https://www.jmdoudoux.fr/java/dej/chap-presentation.htm.10/06/2019.
[28] https://java.developpez.com/tutoriels/programmation-orientee-objet/principes-avances/20/06/2019.
[29] http://lotfi-chaari.net/ens/NFP121/Chap_Bases_donnees.pdf. Consulté le consulté le 04/08/2019.

Annexes

Annexes

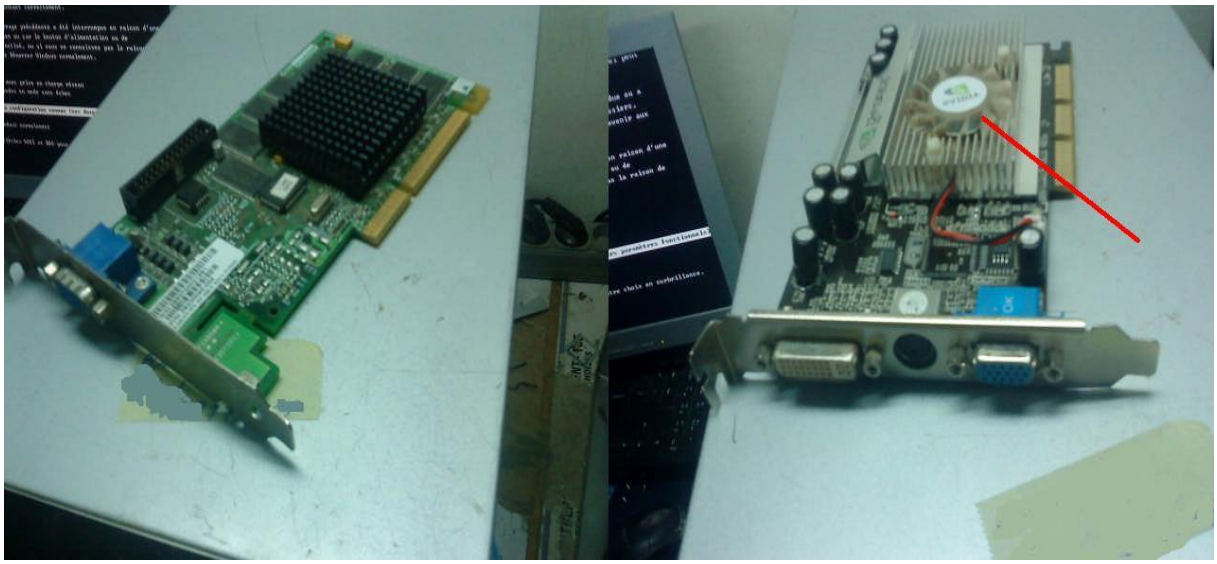


Figure 21 :Deux modèles de cartes graphiques en panne, l'une avec Radiateur (à gauche) et l'autre avec Ventirad (à droite), celle de droite affiche un symptôme : ventilateur poussiéreux qui ne marche plus

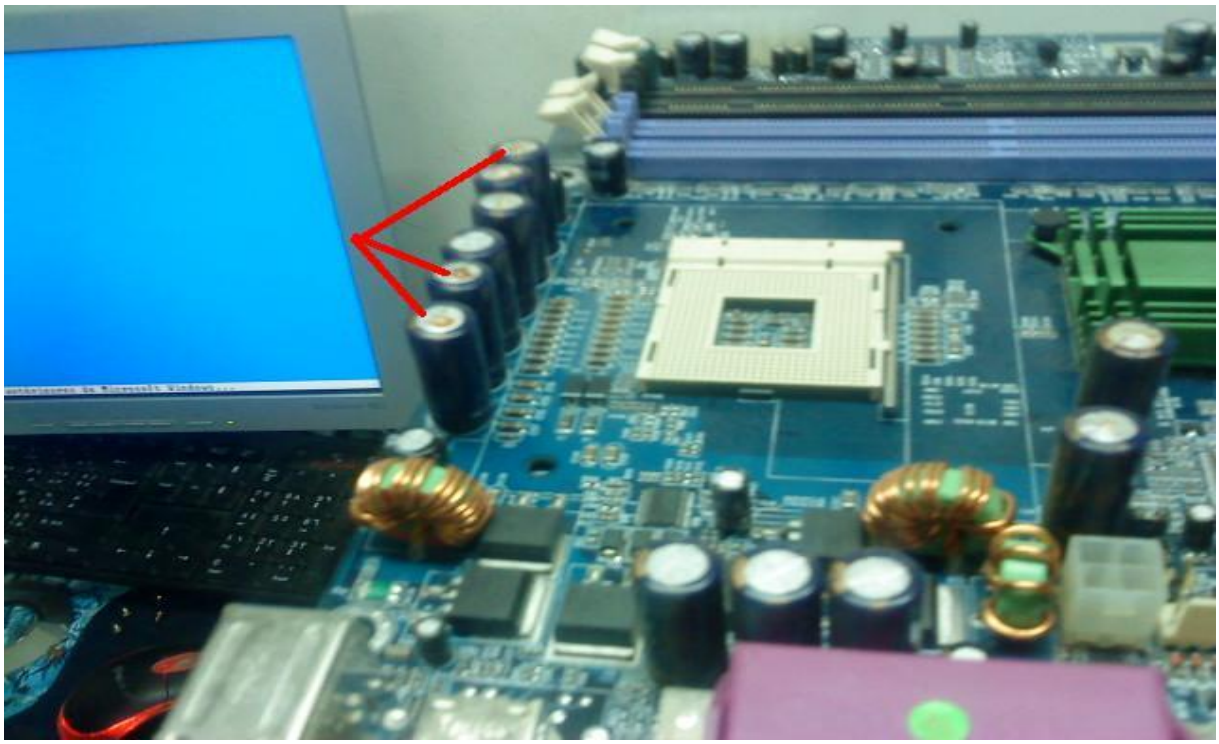


Figure 22 : Disfonctionnement de la carte mère à cause de condensateurs qui sont gonflés et ne fonctionnent pas.

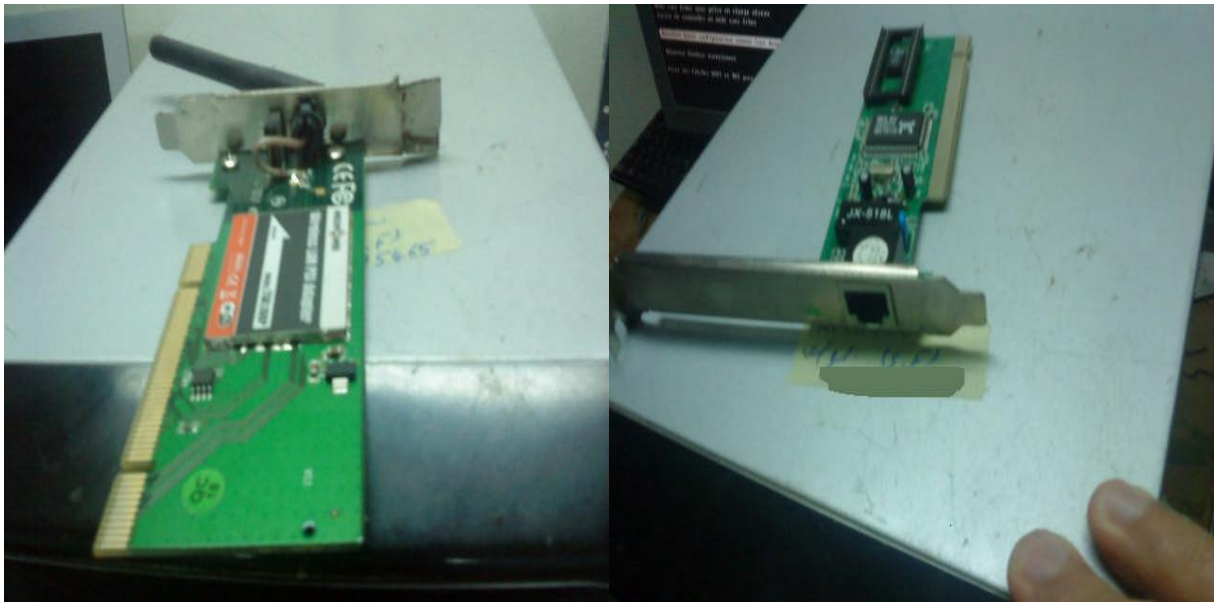


Figure 23:Cartes réseaux Wifi (à gauche) et Ethernet (à droite) qui ne fonctionnent pas, mais le symptôme n'est pas visible depuis les cartes .

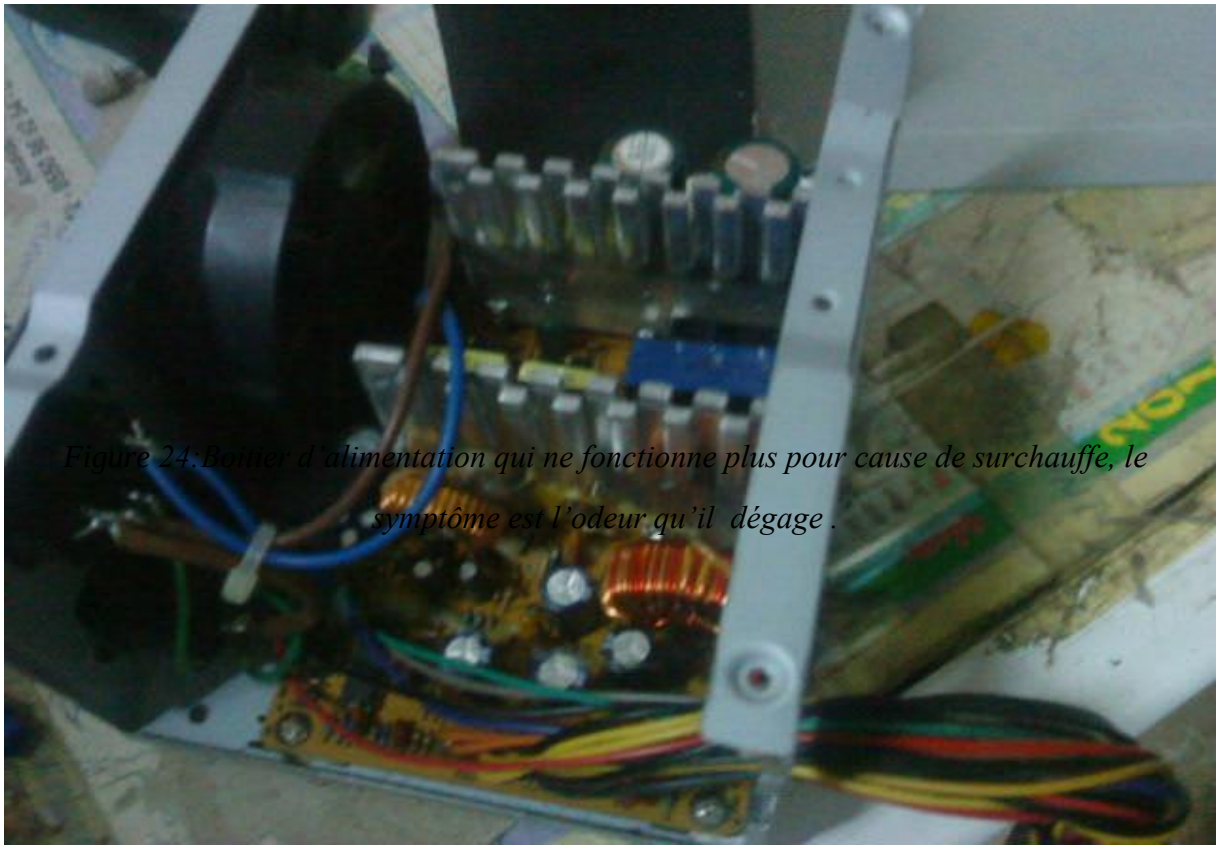


Figure 24:Boitier d'alimentation qui ne fonctionne plus pour cause de surchauffe, le symptôme est l'odeur qu'il dégage .

Figure 25: Disque dur qui ne fonctionne pas mais le symptôme n'est pas visible de l'extérieur

Figure 26 : Connecteur de clavier PS2 (à gauche) ; prise VGA à droite, les symptômes sont clairement visibles, celui de gauche à une pinnescassée, et celui de droite a des dents déformées

Figure 27 : Souris qui a soudainement cessée de fonctionner. Symptôme non visib

Annexes

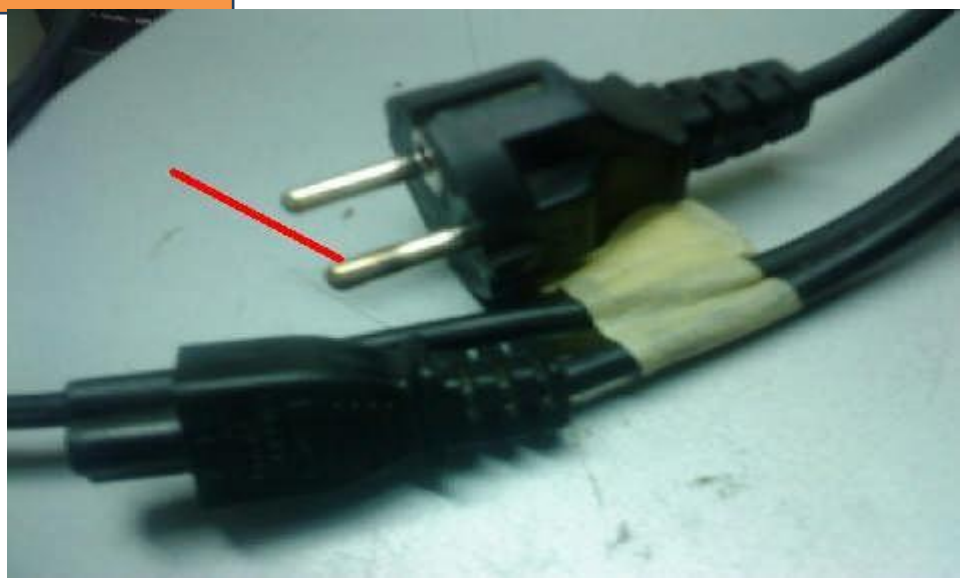


Figure 28 : Prise de courant qui ne fonctionne plus, Symptôme visible de l'extérieur

Figure 29:Généralement les RAM n'ont pas de symptôme visible de l'extérieur, à part celui du connecteur qui est déformé

Annexes

Figure30 : Ventilateur qui a cessé de fonctionner à cause d'une surchauf

Figure 31: Carte mère qui semble intacte, mais qui ne fonctionne pas, les symptômes sont ceux qu'affiche l'ordinateur

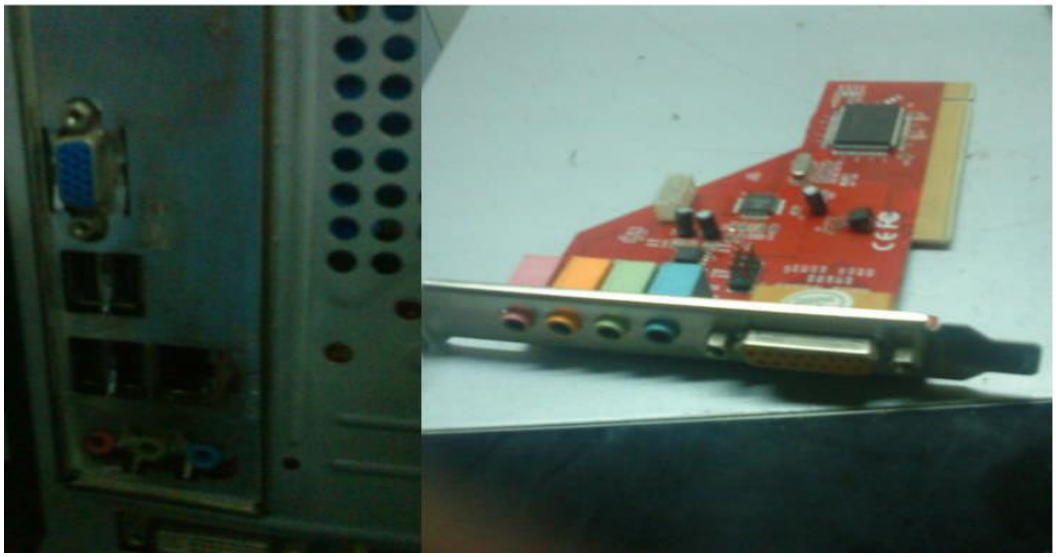


Figure 32 : Les deux composants ne sont pas de même nature (carte mère à gauche et carte son à droite) mais tous deux ont un problème de connecteur (interface) qui est une panne externe mais les symptômes ne sont pas visibles.

Figure 33 : Vu de loin, ce lecteur/ graveur DVD n'affiche aucun symptôme vu de l'extérieur, mais il est bien en panne.

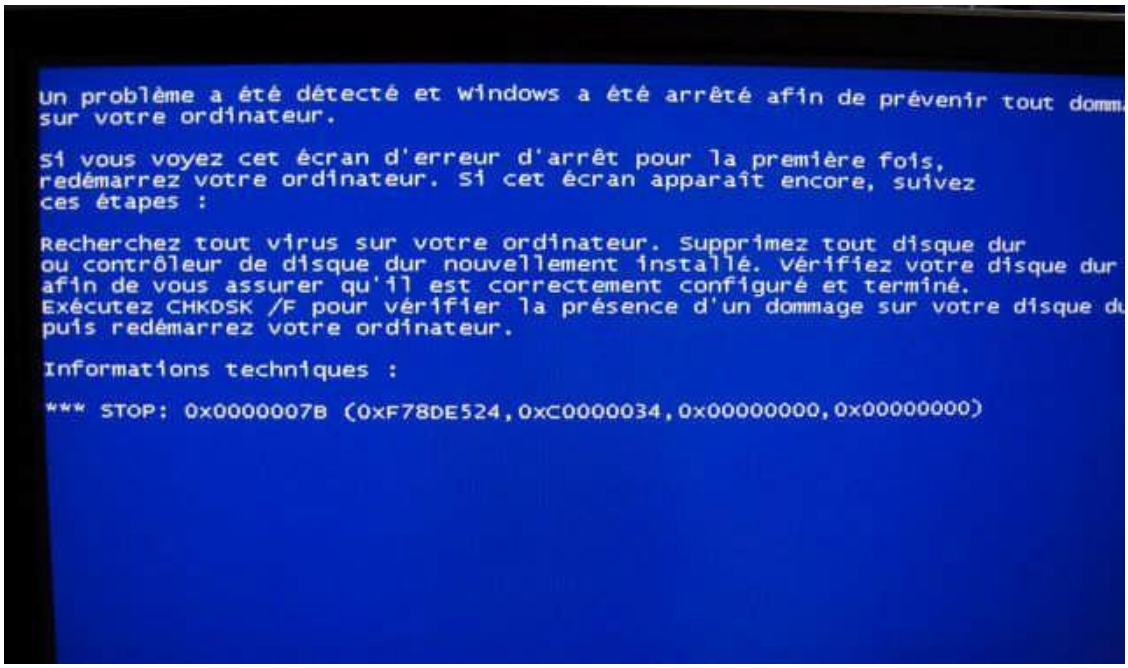


Figure 34 : Ecran Bleu (symptôme)

Figure 35 : Ecran noir(symptôme)

Résumé

L'intelligence artificielle et les systèmes expertsont met des pas très importants au point qu'ils envahissent tous les domaines que l'être humain exerce sur terre. Parmi ces domaines figure celui de l'informatique. Ce dernier nécessite des équipements spéciaux (micro-ordinateur (PC), imprimante, scanner…).L'ordinateur est devenu maintenant un moyen de communication et de recherche, vu cette importance sa maintenance est plus que primordiale.

Après avoir étudié les composants et les périphériques de cet outil, on a constaté que la durée de vie de ses composants est bien limitée. Ce dernier peut donc subir une défaillance, qui doit être rétabliele plus vite possible. Pour cette raison un système expert d'aide au diagnostic des pannes d'un ordinateur est conçu et implémenté.

Pour la conception le choix est tombé sur l'UML (Unified Modeling Langage) alors que pour la réalisation de l'application nommée MicroExpert, c'est le java (implémenter le code de chainage avant) et netbeans IDE (environnement de développement intégré) lié à XAMPP (Appache MariaDB Php Perl) afin de gérer la base de connaissances, qui ont étés choisis.

Mots clés : intelligence artificielle, systèmes experts, diagnostic, base de connaissances, moteur d'inférence, netbeans IDE.

Abstract

Artificial intelligence and expert systems have taken very important steps to the point where they are invadingevery area of life on earth. Among these areas is that of computing. The latter requires special equipments (microcomputer (PC), printer, scanner ...). The computer has nowbecome a means of communication and research, given this importance its maintenance is paramount.

After studying the components and peripherals of this tool, it was found that the life of its components is limited. The latter can there fore suffer a failure, which must be restored as soon as possible. For this reason an expert system for troubleshooting a computer is designed and implemented.

For the design the choice fell on the UML where as for the realization of the application namedMicroExpert, itis the java (to implement the code of inference engine) and netbeans IDE related to XAMPP to manage knowledg which were chosen.

Word key : Artificial intelligence, expert systèms, troubleshooting, inference engine, netbeans IDE.

Annexes