République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira-Bejaïa
Faculté des Sciences Exactes
Département d'informatique

Mémoire de Fin d'Etude

En vue de l'obtention d'un Master 2
Option Professionnel en Administration et sécurité des réseaux

Thème

Conception et réalisation d'un Système Expert d'aide au diagnostic de pannes d'un Micro-ordinateur

(Système COMBREDES)

Réalisé par **:**

DAOUDI Tarik

Encadré par **:**

Mr DEMOUCHE Mouloud

Devant le jury composé de **:**

Mr BAADACHE Abderahmane : Président

Mr Khenous Lachemi : Examinateur

Mr SAADI Mustapha : Examinateur

Promotion 2012-2013

Je tiens à remercier DIEU tout puissant de m'avoir donné la force, la santé, le courage et la patience pour accomplir ce travail et d'être arrivé à le faire.

Je tiens tout particulièrement et sans limites à remercier ma très chère mère sans qui je ne serais pas là, et rien de tout ceci ne serait fait, alors en quelques sortes ce travail lui revient, pour avoir veillée sur moi et m'avoir aidé dans tout ce que j'ai pu entreprendre.

De même je tiens à remercie beaucoup mon cher frère Abdenour qui a toujours pensé à moi malgré la distance qui nous sépare et mes deux chères sœurs et ma grand-mère de m'avoir toujours soutenu dans toutes les circonstances.

Je tiens à remercier aussi vivement mon promoteur Mr DEMOUCHE d'avoir accepté de me guider tout au long du travail et de m'avoir épaulé avec sa disponibilité et ses conseils.

Je remercie également tout le personnel de l'entreprise AMTHEC d'Akbou pour leur hospitalité et m'avoir accepté comme stagiaire.

Mes sincères remerciements s'adressent aussi à tous ceux qui ont contribué de prés ou de loin à la réalisation de ce travail et spécialement Mme Zerouati (ep. Bechir) pour tous les conseils que j'ai reçu de sa part durant tout mon cycle universitaire.

Enfin, je tiens à remercier tous les membres de jury pour avoir accepté d'évaluer mon travail.

Merci à tous

DAOUDI Tarik

Dédicaces

Je dédie ce modeste travail à :

Mes très chers parents (surtout ma mère)

Mon frère et mes deux sœurs

Ma grand-mère

Mes oncles d'Oran et de France

Mes cousins et cousines

Tous Mes adorables amis, surtout Madjid

Tous mes enseignants

Ceux qui cherchent leurs noms ici.

Table de matière

Chapitre I : Généralités sur les systèmes experts

Liste des figures

Page

Liste des tableaux

Page

Liste des abréviations

Abréviation	Signification
2TUP	**2 T**rack **U**nified **P**rocess
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
COMBRES	**COM**puter des **Br**eak Down **Ex**pert **S**ystem
DSE	**D**iagramme de **S**équence
FK	**F**oreign **Key**
FSB	**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
PK	**P**rimary **K**ey
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
SGBD	**S**ystème de **G**estion de **B**ase de **D**onnées
UML	**U**nified **M**odeling **L**anguage
UP	**U**nified **P**rocess
USB	**(Universal Serial Bus, en français Bus série universel)**
UC	**U**se **C**ase
Ventirad	**Venti**lateur + **Rad**iateur

Introduction générale

CHAPITRE I

Généralités sur les systèmes experts

«L'homme qui déplace une montagne commence par déplacer les petites pierres...»

Confucius

Introduction générale

Vivant dans un monde de technologies où les systèmes informatiques ont pris une place énorme au sein des populations, dans le but de servir l'être humain et son bienêtre, même si parfois au péril de son métier et de son travail, en le remplaçant par un système qui réfléchit et agit comme lui, tout en étant plus performant et productif que ce dernier.

Les systèmes experts élargissent le domaine de l'intelligence artificielle. Bien qu'ils y fassent parti, les systèmes experts, forment un sujet auquel se penchent plusieurs spécialistes de la recherche informatique et de l'intelligence artificielle, afin de rapprocher les décisions humaines expertes des systèmes informatiques, pour que ces derniers puissent déduire les mêmes résultats et donner les mêmes diagnostics qu'un expert. Sachant qu'un système expert peut donner de plus amples résultats en un temps très réduit par rapport à un humain en étant plus précis, ceci augmente l'intérêt des chercheurs à travailler dessus et les intégrer dans plusieurs domaines et métiers. Parmi ces métiers on distingue celui de la maintenance informatique.

L'ordinateur, un outil qui a révolutionné le monde tant en espace qu'en temps. Cet outil qui a été longtemps au service de l'être humain, en remplaçant la paperasse au-dessus de son bureau par un espace de stockage ou bien en s'adaptant parfaitement a son métier (Architecture, génie civil, commerce, etc.) avec l'évolution du génie logiciel, lui permettant de fournir moins d'efforts et de produire plus de résultats. Mais malheureusement ces ordinateurs sont sujets à des pannes et à des défaillances matérielles et ceci empêche le bon fonctionnement des programmes de l'ordinateur et freine l'utilisateur dans ses activités.

Dans le premier chapitre, nous allons nous introduire de façon brève à la discipline de l'intelligence artificielle, et puis nous mettrons en avant quelques définitions et quelques exemples illustratifs de ce domaine. Puis, nous donnerons quelques domaines d'applications se référant à celle-ci, et puis nous approfondirons sur les systèmes experts.

Dans le second chapitre nous présenterons et étudierons le métier sur lequel nous voulons travailler. Dans le prochain chapitre nous utiliserons les acquis du deuxième chapitre afin de faire une conception logiciel sûre et valide. Et puis, dans le dernier chapitre nous allons présenter l'application résultante de tout ce que nous avons fait précédemment. Enfin, dans la conclusion nous donnerons le bilan de tout ce que nous avons appris lors du développement du système depuis la conception jusqu'à la réalisation

1. L'intelligence Artificielle (IA)

1.1 Définitions de l'IA

L'intelligence artificielle peut être définie comme la science de la simulation des processus cognitifs. Ces processus comprennent:

- l'acquisition des connaissances,
- l'archivage des connaissances,
- l'application des connaissances. **[1]**

On distingue entre ***connaissances procédurales,*** qui sont indépendantes du domaine d'application, et ***connaissances spécifiques*** d'un domaine d'application. On cherche à séparer explicitement ces deux types de connaissances dans les systèmes intelligents, afin de permettre une transmission de la connaissance procédurale d'un domaine à un autre. **[1]**

Nous pouvons aussi définir l'IA comme la science dont le but est de faire exécuter par une machine des tâches que l'homme accomplit en utilisant son intelligence. La terminologie de d'Intelligence Artificielle est apparue en 1956. On peut lui donner aussi l'appellation d'Informatique Heuristique. **[2]**

Nous terminerons par la définition de (J.L.Laurière) qui la définit comme l'étude des activités intellectuelles de l'homme pour lesquelles aucune méthode n'est a priori connue. **[3]**

1.1.1. Exemples :

- Jeu d'échecs (bien qu'il n'y ait qu'un nombre fini de situations);
- Résumer un texte ou le traduire;
- Reconnaitre des lettres manuscrites, par exemple TAON ou THON;
- Faire un diagnostic (médical, **de pannes** (sujet de notre mémoire), ...) **[2]**

1.2 Domaine d'application de l'IA

1.2.1 Le traitement automatique du langage naturel

On regroupe sous le vocable de traitement automatique du langage naturel (TALN) l'ensemble des recherches et développements visant à modéliser et reproduire, à l'aide de machines, la capacité humaine à produire et à comprendre des énoncés linguistiques dans des buts de communication **[15]**

1.2.2 La traduction automatique

Est un système informatique qui a : pour entrée, un texte "t1", ou texte source écrit dans une langue "L1"ou langue d'origine, et n'ayant pas subi d'aménagements spéciaux préalables au traitement automatique qu'il va subir, pour sortie un texte "t2" ou texte traduit écrit dans une langue "L2"ou langue cible, tel qu'il n'ait pas à subir de transformations pour être reconnu par les utilisateurs comme une traduction du texte t1. **[16]**

1.2.3 La vision par ordinateur

La vision a pour objectif de rendre les machines indépendantes de l'homme dans l'environnement ou sa présence est impossible ou non souhaitée. Elle permet à partir de l'image d'un objet ou d'une scène réelle d'obtenir des données exploitables par une machine. **[4]**

1.2.4 Les systèmes experts

Un système expert est un logiciel capable de simuler le comportement d'un expert humain effectuant une tâche précise. Le succès de l'intelligence artificielle dans ce domaine est indéniable, dû au caractère ciblé de l'activité qu'on lui demande de simuler.
En conclusion l'intelligence artificielle est présente sous plusieurs formes parfois inattendues et dans des domaines variés, mais toutes ses applications ont pour but de simplifier la vie de l'être humain. **[6]**

2. SYSTEMES EXPERTS (SE)

2.1 Introduction

Les Systèmes experts sont conçus pour réfléchir d'une manière propre à un domaine, plus exactement, en utilisant la même méthodologie que l'expert du métier et ses connaissances afin d'arriver à des conclusions et des résultats et donner un diagnostic précis. Les Systèmes experts appartiennent à l'intelligence artificielle car ils ne font pas qu'exécuter les tâches seulement, comme le ferais un ordinateur utilisant un algorithme simple dont l'exécution se répète et est toujours la même ; les systèmes experts apprennent au fur et mesure qu'ils exécutent les instructions, c'est l'aspect le plus intéressant, qui les particularise et qui réduit la distance entre le système et l'expert permettant à celui-ci de travailler avec aisance et faire confiance aux diagnostics résultants.

2.1 Définition d'un SE

Un système-expert, selon la définition proposée **par J.C. Pomerol**, est un outil informatique d'intelligence artificielle, conçu pour simuler le savoir-faire d'un spécialiste, dans un domaine précis et bien délimité, grâce à l'exploitation d'un certain nombre de connaissances fournies explicitement par des experts du domaine. Il permet de **modéliser** le raisonnement d'un expert, de manipuler des connaissances sous une forme déclarative, d'en faciliter l'acquisition, la modification et la mise à jour et de produire des explications sur la façon dont sont obtenus les résultats d'une expertise**. [7]**

2.3 Caractéristiques des SE

Il y a séparation entre les connaissances (*bases de connaissances*) et le programme qui permet de les utiliser (*moteur d'inférence*). Le moteur d'inférence peut être écrit dans n'importe quel langage de programmation (Ex : C++, Pascal, etc…). La base de connaissances doit être écrite dans un langage déclaratif accessible à un expert non informaticien. **[2]**

- **Dans les SE il y a opposition entre le *déclaratif* et le *procédural :***

Exemple : "L'article s'accorde en genre et en nombre avec le nom." est déclaratif. Une expression procédurale serait "Quand on rencontre un nom, on cherche l'article qui lui est lié, on vérifie alors que le genre de l'article est le même que le genre du nom, puis que le nombre de l'article est le même que le nombre du nom." (Compréhension) ou "pour écrire l'article, on

cherche le nom qui lui est lié et on choisit un article qui a le même genre et le même nombre." (Génération). **[2]**

Le *déclaratif* est plus agréable. On peut donner les connaissances en vrac, en ajouter, en enlever, les modifier facilement. Mais il est beaucoup moins efficace. Maintenant, dans un souci d'efficacité, on réalise souvent une "compilation" des connaissances déclaratives (données par l'expert) en connaissances procédurales ou en programmes. **[2]**

Nous distinguons aussi les caractéristiques suivantes :

- **Haut rendement**: Le système doit avoir la capacité de répondre à un niveau de compétence égal ou supérieur à celui d'un spécialiste du domaine. Cela signifie que la qualité de conseil donné par un système doit être très haute. **[8]**
- **Temps de réponse adéquat:** Le système doit agir en un temps raisonnable, comparable ou meilleur au temps exigé par un spécialiste, pour prendre une décision. **[8]**
- **Fiabilité**: le système expert doit être fiable et ne doit pas connaître de "failles" sinon il ne sera pas utilisé. **[8]**
- **Compréhensible:** le système doit être capable d'expliquer les étapes de son raisonnement pendant qu'elles s'exécutent, au lieu d'être seulement une boîte noire qui produit une réponse miraculeuse. **[8]**
- **Flexibilité:** Vu la grande quantité de connaissance qu'un système expert peut avoir, il est important d'avoir un mécanisme efficient pour ajouter, modifier, et supprimer la connaissance. Une raison de la popularité des systèmes experts basés sur les règles est la capacité de stockage efficiente et modulaire des règles. **[8]**

On distingue aussi d’autres caractéristiques de nature différente :

2.3.1 Difficultés [2]

- Comment programmer une machine pour en faire un expert ?
- Comment gérer les contradictions éventuelles entre différents experts ?
- Comment faire évoluer un tel système sans que ses compétences se dégradent (le paradoxe du trop-plein de connaissance) ?

- D'une part utiliser une très grande quantité de connaissances, d'autre part réunir toutes les connaissances nécessaires, il est indispensable de les introduire sur les *méta-connaissances.*

Exemple : METADENDRAL crée de nouvelles règles. Il a permis l'analyse de familles chimiques mal connues et a donné lieu à des articles dans des revues internationales de chimie.

2.3.2 Apprentissage - Découverte (à partir de 1970) [2]

Il s'agit alors d'apprendre des règles ou des concepts, dans différents domaines :

- jeux : poker (Waterman, 1970), échecs (Pitrat, 1976)
- vision et reconnaissance des formes en robotique (Winston, 1975)
- découverte de concepts mathématiques, par exemple concept de nombre premier (AM, Lenat 1980)
- découverte des lois de Képler en astronomie (BACON, Langley, 1979)

(Il faut relativiser ces découvertes en examinant ce que reçoit le système (par exemple des définitions LISP (Locator Identifier Separation Protocol) de concepts mathématiques primitifs, des données recueillies par Képler)

2.3.3 Formalisation du sens commun (à partir de 1980) [2]

Les connaissances pragmatiques sont en général implicites.

Exemple : "En posant brusquement sa tasse sur la table, il la cassa."

Il faut tenir compte de l'incertitude, des ambiguïtés, de l'incomplétude, des imprécisions. On utilise des facteurs de certitude dans les systèmes experts :

- les probabilités
- des logiques non classiques - non monotones (Mc Carthy, 1980)
- défauts (Reiser, 1980)
- floue (Zadeh, 1978)
- possibilités (Dubois, Prade, 1987)

Des travaux ont été effectués en physique naïve et qualitative (Hayes, 1979 - De Kleer, Bron, 1984 - Forbus, 1984 - Knipers, 1984)

Exemple de règle : Dans les gaz parfaits, à volume constant, plus la pression augmente, plus la température augmente.

Le projet CYC (Lenat, à partir de 1984) est un projet ambitieux dont le but est de représenter les connaissances de sens commun. En effet, il ne suffit pas d'avoir des formalismes, il faut aussi recueillir et exprimer les connaissances. Plusieurs centaines de milliers de règles ont été écrites. Exemple : Quand on possède quelque chose, on en possède ses parties.

2.3.4 Utilisation du niveau méta [2]

Surtout à partir de 1980, mais déjà en 1966 (Pitrat, Démonstration de théorèmes en logique)et 1976 (METADENDRAL).

1979 - TEITESIAS est un module d'acquisition des connaissances pour MYCIN

Idée forte : plutôt que de donner des connaissances sophistiquées à un système, on lui donne la possibilité d'en acquérir et/ou la possibilité d'améliorer des connaissances simples qu'on lui donne.

2.4 Structure d'un système expert

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

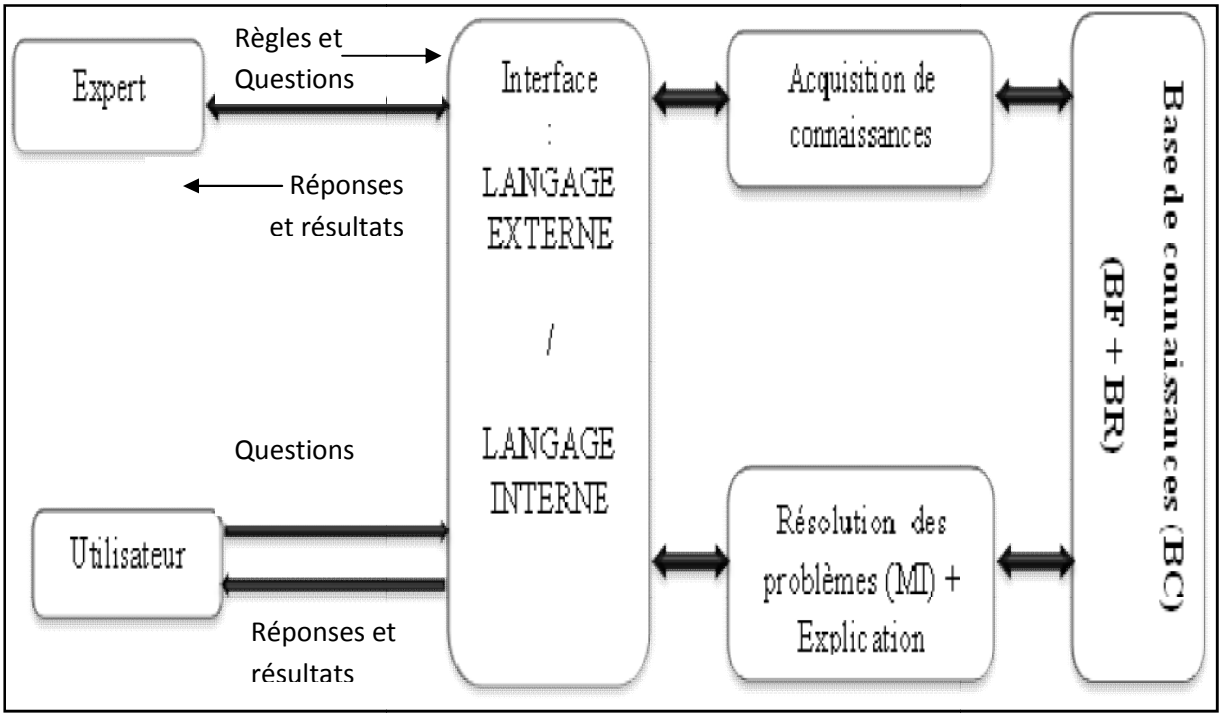


Figure 1. Architecture d'un Système Expert **[17]**.

- **L'interface utilisateur:** sert à simplifier la communication, elle peut utiliser la forme question-réponse, le menu, le langage naturel etc. **[17]**
- **La base de connaissances** : contient les connaissances concernant la résolution du problème et dispose des deux bases suivantes:
  - **La Base de Faits (BF)**: est l'une des entrées d'un moteur d'inférence. elle contient les connaissances représentant des états considérés comme prouvés. C'est la mémoire de travail du SE. Elle est variable au cours de l'exécution et vidée lorsque l'exécution est terminée. Les faits peuvent prendre des formes plus ou moins complexes. **[17]**.

  - **La Base de Règles (BR) :** La base de règles contient les connaissances expertes, c'est-à-dire qu'elles représentent les raisonnements effectués par un expert. Elles sont appelées les unes à la suite des autres afin de créer des enchaînements de raisonnements. Tous ces raisonnements peuvent être représentés sous la forme de règles de production du type « Si condition alors action ». Toutefois, cette représentation peut varier suivant le contexte de l'application.
  - **Le module d'explication** : permet au système expert d'expliquer son raisonnement.

2.4.1 Le moteur d'inférence [18]

Un moteur d'inférence permet de conduire des raisonnements logiques en utilisant conjointement la base de faits et la base de règles. Selon différentes stratégies, le moteur d'inférence utilise des règles, les interprète, les enchaîne jusqu'à arriver à un état représentant une condition d'arrêt. Ces dernières dépendent du moteur et de la base de connaissances implémentée. En général, l'exécution de règles par le moteur d'inférence influe sur l'état des faits et éventuellement sur les autres règles.

Un moteur d'inférence peut exécuter des règles, suivants différentes méthodes d'invocation:

2.4.1.1 Chaînage avant

Un moteur d'inférence fonctionne dans ce mode lorsque les faits de la base de faits représentent des informations dont la valeur de vérité a été prouvée. C'est-à-dire que ce mode de fonctionnement va des faits vers les buts. **[18]**

- **Algorithme du chainage avant**

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**

- **EXEMPLE**

Soit la base de faits initiale **BF = {B, C}**, on cherche à démontrer le fait **H**, 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 B *et* D *et* E *alors* F
2. Si G *et* D *alors* A
3. Si C *et* F *alors* A
4. Si B *alors* X
5.Si D *alors* E
6. Si X *et* A *alors* H
7. Si C *alors* D
8. Si X *et* B *alors* A

Traitement

N°cycle	Evaluation de la base des faits	N°règle
1	B, C +X	4
2	X, B, C+ D	7
3	D, X, B, C +A	8
4	A, D, X, B, C + H	6
5	**Base finale :{ A, D, X, B, C, H}** **il y'a une réussite**	

2.4.1.2 Chaînage arrière [18]

Un moteur d'inférence fonctionne dans ce mode lorsqu'il part d'un fait que l'on souhaite établir, qu'il recherche toutes les règles qui concluent sur ce fait, qu'il établit la liste des faits qu'il suffit de prouver pour qu'elles puissent se déclencher puis qu'il applique récursivement le même mécanisme aux autres faits contenus dans cette liste. **[18]**

- **Algorithme du chainage arrière**

Fonction **chainageArriere**
Paramètres : in BR, in BF, in listeButs.
SI est Vide(listeButs) **ALORS**
res ← SUCCES
SI
SI demBut(premier(listeButs)) **ALORS**
res←chaînageArriere(suite(listeButs))
SI
res ← ECHEC
FIN SI
FIN SI
retourner res

- **Exemple**

Soit la base de faits **BF= {I, G}**, 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 G *et* F *alors* E
3. Si H *et* I *alors* F
4. Si F *et* H *alors* G
5. Si I *e* G *alors* H

Traitement

N°cycle	Evaluation de la base des faits	N°règle
1	I, G, F, E	1
2	I, G, H, F	2
3	I, G, H	3
4	I, G	
	Base finale : {I, G} il n'y a plus de faits à démontrer en base de faits alors réussite	

2.4.1.3 Chaînage mixte [18]

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. **[18]**

2.4.1.4 Chaînage bidirectionnel [18]

Alternance de chaînage avant et arrière, mais jamais l'utilisation des deux à la fois

2.5 Domaine D'application des systèmes experts

2.5.1 Quelques systèmes experts classiques [10]

DENDRAL, 1969 : chimie, recherche la formule développée d'un corps organique à partir de la formule brute et du spectrogramme de masse du corps considéré.

MYCIN, 1974 : médecine, système d'aide au diagnostic et au traitement de maladies bactériennes du sang.

AM, 1977 : mathématiques, proposition de conjectures, de concepts intéressants.
(500 règles)

MOLGEN, 1977: biologie, engendre un plan de manipulations génétiques en vue de construire une entité biologique donnée.

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

CRYSALIS, 1979 : chimie, recherche la structure de protéines à partir de résultats d'analyse cristallographique.

MUSCADET, 1984 : mathématiques, démonstration de théorèmes.

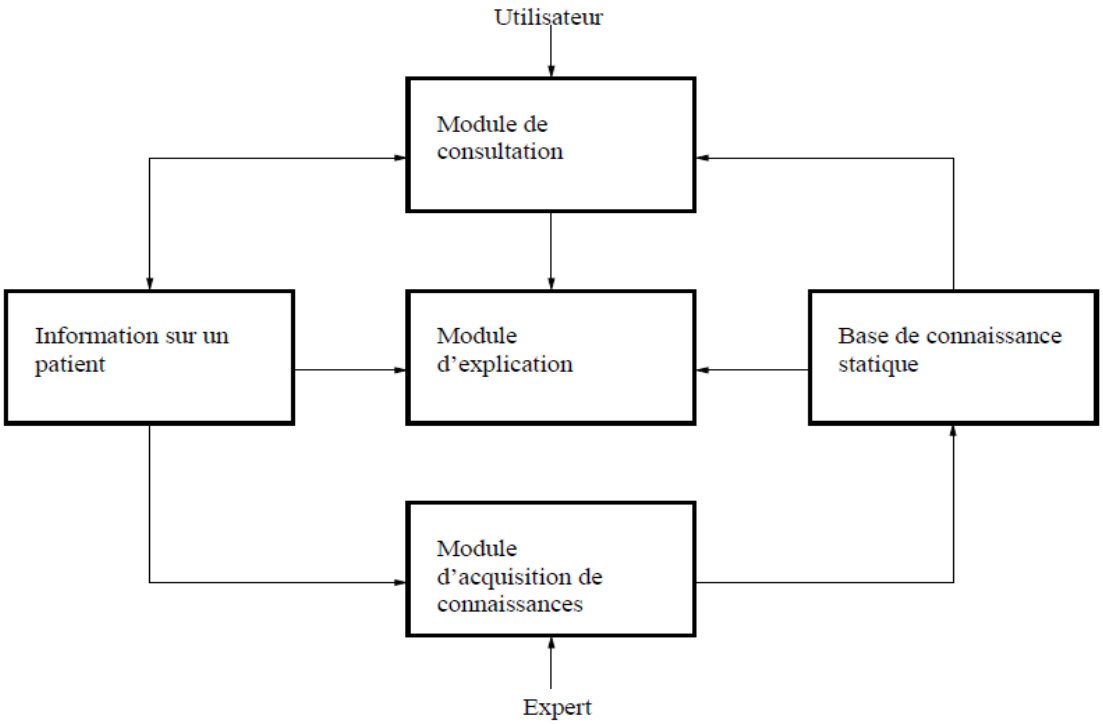


Figure 2. Organisation de MYCIN **[10]**

2.5.2 Exemple (RUFUS, UN Système- Expert de diagnostic de pannes électriques du MÉTRO) [9]

L'expérience du S.E. Rufus, comme d'autres expériences ultérieures de S.E. réalisées à et par la RATP (*Régie Autonome des Transports parisiens*), trouve son origine dans la réflexion menée en 1984, dans le cadre du programme de recherche « Réseau 2000 », par l'observatoire technologique sur les possibilités offertes par les nouvelles techniques informatiques et leur impact à plus ou moins long terme. Les techniques de programmation de l'I.A. en général, et les systèmes experts en particulier, étaient alors l'objet de débats dans les milieux scientifique set universitaires. Ils suscitaient l'intérêt des entreprises à travers la mise en œuvre de leurs premières applications expérimentales. Les systèmes-experts, par leurs qualités annoncées de capacité à traiter des connaissances pratiques non formalisées ou peu formalisées, de maintenabilité plus aisée, n'exigeant pas la remises à jour la refonte complète des programmes initiaux, se présentaient comme des outils informatiques permettant d'aborder des domaines non traitables jusqu'alors par l'informatique traditionnelle, et rendant possible une informatique plus proche des utilisateurs.

2.5.3 Représentation des connaissances [2]

Ce nouveau thème de recherche, déjà apparu avec le traitement des langues naturelles, se développe avec l'utilisation et l'étude des systèmes experts, que l'on appellera alors souvent systèmes à base de connaissances. Plusieurs formalismes sont utilisés :

- la logique avec le calcul des prédicats d'ordre 0 ou 1 ou intermédiaire (0+);
- les logiques non classiques (flou, possibilités, …);
- les règles (déduction, réécritures, actions conditionnelles, …);
- le formalisme objet (objets, classes, instances, propriétés, héritage)

2.5.3.1 Sortes de connaissances [11]

- **Déclarative : expression** symbolique (abstraite) d'une compétence.

Utilisé pour communiquer et pour raisonner sur des connaissances.

- **Procédurale :** expression "compilée" d'une compétence.

Utilisé pour optimiser le temps d'exécution.

- **Réactive** : association "stimuli" - "Réponse".

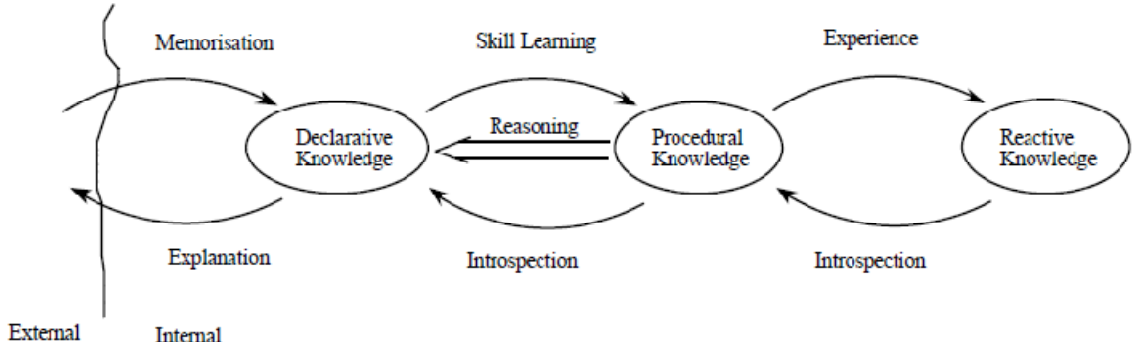


Figure 3.Schéma d'acquisition de la connaissance. **[11]**

Une autre distinction est la connaissance "superficielle " et connaissance "profonde"
Connaissance profonde : un modèle permettant le raisonnement par simulation
Superficielle : une expression symbolique des associations des faits permettant un raisonnement "abstrait".
La plupart des systèmes experts fonctionnent avec une connaissance "superficielle".

2.5.3.2 Comment représenter la connaissance ?

Pour représenter les connaissances nous avons les différents modèles suivants :

- Les réseaux sémantiques.
- Les représentations logiques.
- Les réseaux de neurones
- Les règles de production.
- Les objets structurés

2.5.4 Les réseaux sémantiques [11]

L'utilisation des réseaux sémantiques comme formalisme de représentation de connaissances remonte aux travaux du linguiste **Quillian** (en 1968) sur la mémoire sémantique. En effet, les réseaux sémantiques sont très utilisés dans les travaux sur le traitement et la compréhension des langages naturels.

Ce formalisme a comme caractéristique principale le fait que la signification d'un concept dépende du réseau sémantique dans lequel il s'inspire et de ses relations avec les autres concepts de ce réseau.

Dans les réseaux sémantiques, les concepts du domaine à modéliser sont représentés par des nœuds et les relations entre concepts par des arcs étiquetés interconnectant les nœuds associés à ces concepts. Un réseau sémantique peut être défini comme étant un formalisme de représentation structuré de connaissances sous forme de graphe sémantique. On rappelle qu'un graphe est une structure mathématique où nœuds et arcs n'ont pas de significations particulières alors que dans un réseau sémantique ils ont une signification spécifique d'où l'utilisation du mot sémantique.

Parmi les concepts qu'un réseau sémantique peut très bien modéliser on trouve les concepts fortement hiérarchise où les relations entre concepts sont de type : ***est-un sous-ensemble de*** (arc isa) ou ***est-un élément de***, comme ceci est illustré dans l'exemple suivant :

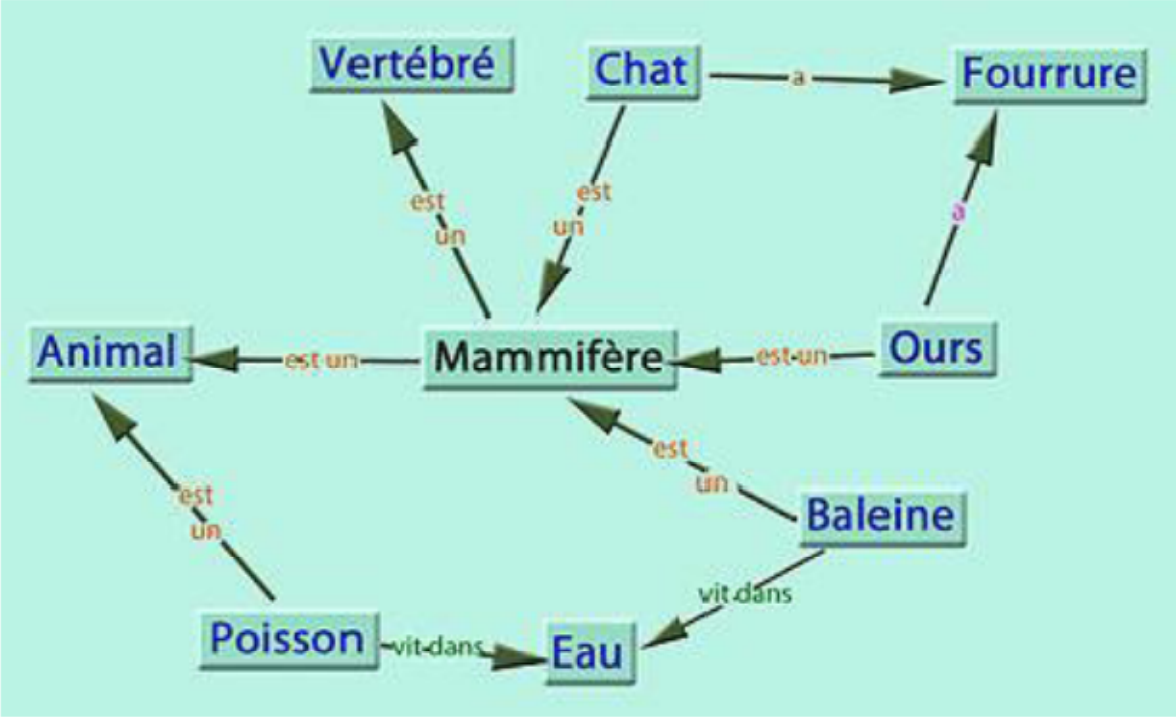


Figure 4. Représentation d'un réseau sémantique **[11]**

2.5.5 Les représentations logiques [13]

- **Logique d'ordre 0**

On parle indifféremment de logique booléenne, de logique des propositions: c'est la plus simple des logiques, appelée la logique d'ordre 0.

Dans cette logique, les formules sont appelées des propositions.

- On définit le calcul propositionnel, comme on définit le calcul arithmétique.
- Les formules sont construites à l'aide de variables issues d'un ensemble dénombrable et des symboles { }.

Exemple : Il n'y a pas d'homme qui marche = Aucun homme ne marche.

$\neg$ ($\exists$ X homme(X) => marche(X)).

$\forall$ X homme(X) =>¬ marche(X).

- **Logique d'ordre 0+**

D'un point de vue sémantique, cela revient à dire que l'on considère les fonctions et prédicats comme des objets à part entière, au même titre que, par exemple, un nombre entier. On s'autorisera ainsi, d'une part, à quantifier les prédicats et

fonctions et, d'autre part, à donner des fonctions ou des prédicats en arguments d'autres fonctions et prédicats. Néanmoins, on pourra se doter d'un système de typage qui restreindra le genre d'objet qui pourra être donné en tant que tel ou tel argument de tel ou tel prédicat ou telle ou telle fonction.

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. La même chose est valable pour les fonctions d'ordre supérieur.

Exemple : Si *heure = 20* ***Alors*** ***temps=*** *«Nuit»*

Prémisse **Conclusion**

- **Logique d'ordre 1**

C'est la logique du calcul des prédicats.
Définition du prédicat : Expression logique dont la valeur peut être vraie ou fausse selon la valeur des arguments, c'est-à-dire une fonction qui renvoie "vrai" ou "faux".
La logique d'ordre 1 contient :

* des variables substituables = variables d'individu,
* des quantificateurs : Pour tout, Quelque soit, ...

Exemples :
homme(x) => x est un homme.
mortel(x) => x est un mortel
aller_à_pied(x) => x peut aller a pied
- Tout homme est mortel : homme(x) -> mortel(x) pour tout x ;
- Quelque soit x, d(x) < 2 km implique aller_à_pied(x) ;
- On peut aussi utiliser des fonctions : inférieure (distance(x), 2 km) implique aller_à_pied(x).
En logique d'ordre 1, on doit choisir des instances pour les variables substituables : le x de distance(x).

Pour pouvoir programmer l'exemple du syllogisme de Socrate avec VBBrainBox, on doit descendre en logique d'ordre 0+ en définissant une session "Socrate", dans laquelle bHomme est Vrai, avec la règle Si bHomme Alors bMortel.

2.5.6 Les réseaux de neurones [13]

2.5.6.1 Définition

Un réseau de neurones est une fonction paramétrée qui est la composition d'opérateurs mathématiques simples appelés *neurones formels* (ou plus simplement *neurones*)pour les distinguer des neurones biologiques.

2.5.6.2 Les neurones

Un neurone est une fonction algébrique non linéaire, paramétrée, à valeur bornée de variables réelles appelées *entrées.*

Un neurone réalise une fonction non bornée $y=f\{x_1, x_2,\ldots, x_n ; w_1, w_2,\ldots, w_n\}$ où les :

$\{x\}_i$: sont les entrées

$\{w\}_j$: sont des poids

f : est la fonction d'activation

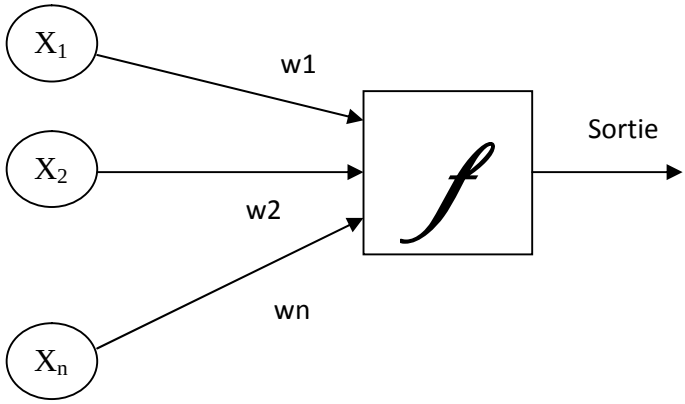


Figure 5. Représentation d'un réseau de neurones **[13]**

2.5.7 Règles de production [14]

Une **règle de production** se compose d'une partie gauche appelée « l'antécédent » ou la « prémisse» et d'une partie droite appelée le « conséquent » ou la « conclusion » qui se présente sous la forme : **Si condition(s) alors action(s)**

Une règle permet de représenter des éléments de connaissances très courant, utiles pour une analyse, un diagnostic, une reconnaissance, une classification, etc.

Le langage de description des règles peut être plus ou moins complexe, comporter des conjonctions, des disjonctions, des négations ...

- **Exemple**

(R1) Si l'animal a des poils, alors c'est un mammifère.

(R2) Si l'animal donne du lait, alors c'est un mammifère.

(R3) Si l'animal a des plumes, alors c'est un oiseau.

(R4) Si l'animal vole, et s'il pond des œufs, alors c'est un oiseau.

(R5) Si l'animal est un mammifère, et s'il mange de la viande, alors c'est un carnivore.

(R6) Si l'animal est un mammifère, et s'il a des dents pointues, et s'il a des griffes, et ses yeux pointent vers l'avant, alors c'est un carnivore.

2.5.8 Les objets structurés [19]

Un objet est une entité cohérente rassemblant des données et du code travaillant sur ses données là. Une classe peut être considérée comme un moule à partir duquel on peut créer des objets. La notion de classe peut alors être considérée comme une expression de la notion de classe d'équivalence chère aux mathématiciens. En fait, on considère plus souvent que les classes sont les descriptions des objets (on dit que les classes sont le méta-donné des objets), lesquels sont des instances de leur classe.
Pourquoi ce vocabulaire ? Une classe décrit la structure interne d'un objet : les données qu'il regroupe, les actions qu'il est capable d'assurer sur ses données. Un objet est un état de sa classe. Considérons par exemple la modélisation d'un véhicule telle que présentée par la figure suivante :

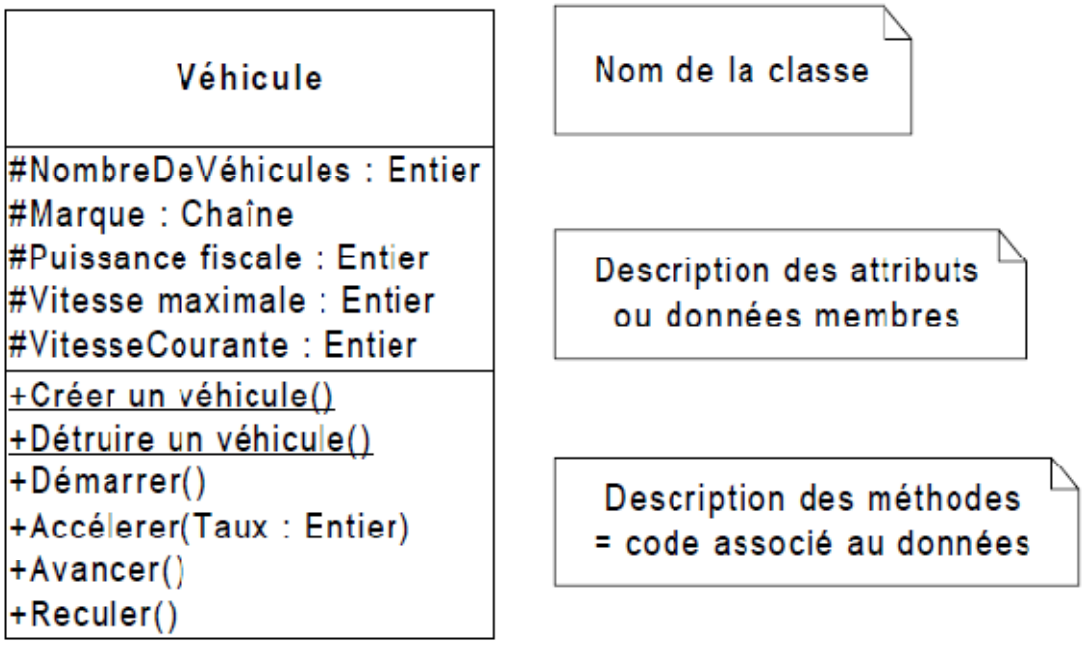


Figure 6.Représentation de la classe Véhicule **[19]**

Les principales caractéristiques dans la structure d'objet sont :

- **Encapsulation :**

  L'encapsulation est un mécanisme consistant à rassembler les données et les méthodes au sein d'une structure en cachant l'implémentation de l'objet, c'est-à-dire en empêchant l'accès aux données par un autre moyen que les services proposés. L'encapsulation permet donc de garantir l'intégrité des données contenues dans l'objet.

- **Héritage:**

  Est un principe propre à la programmation orientée objet, permettant de créer une nouvelle classe à partir d'une classe existante. Le nom d'"*héritage*" (pouvant parfois être appelé *dérivation de classe*) provient du fait que la classe dérivée (la classe nouvellement créée) contient les attributs et les méthodes de sa superclasse (la classe dont elle dérive). L'intérêt majeur de l'héritage est de pouvoir définir de nouveaux attributs et de nouvelles méthodes pour la classe dérivée, qui viennent s'ajouter à ceux et celles héritées.

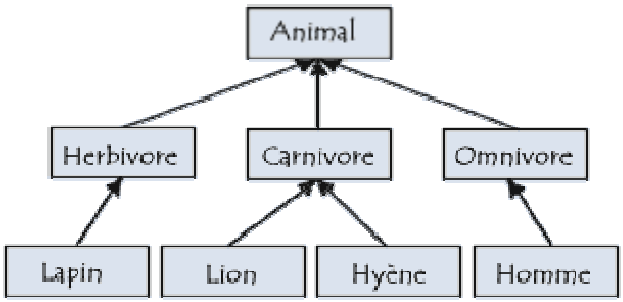


Figure 7 .Exemple d'héritage **[19]**

- **Polymorphisme:**

  Le terme polymorphisme indique qu'une entité peut apparaître suivant plusieurs formes. Dans le cas de notre hiérarchie de documents nous remarquons qu'il existe une méthode qui porte le même nom.

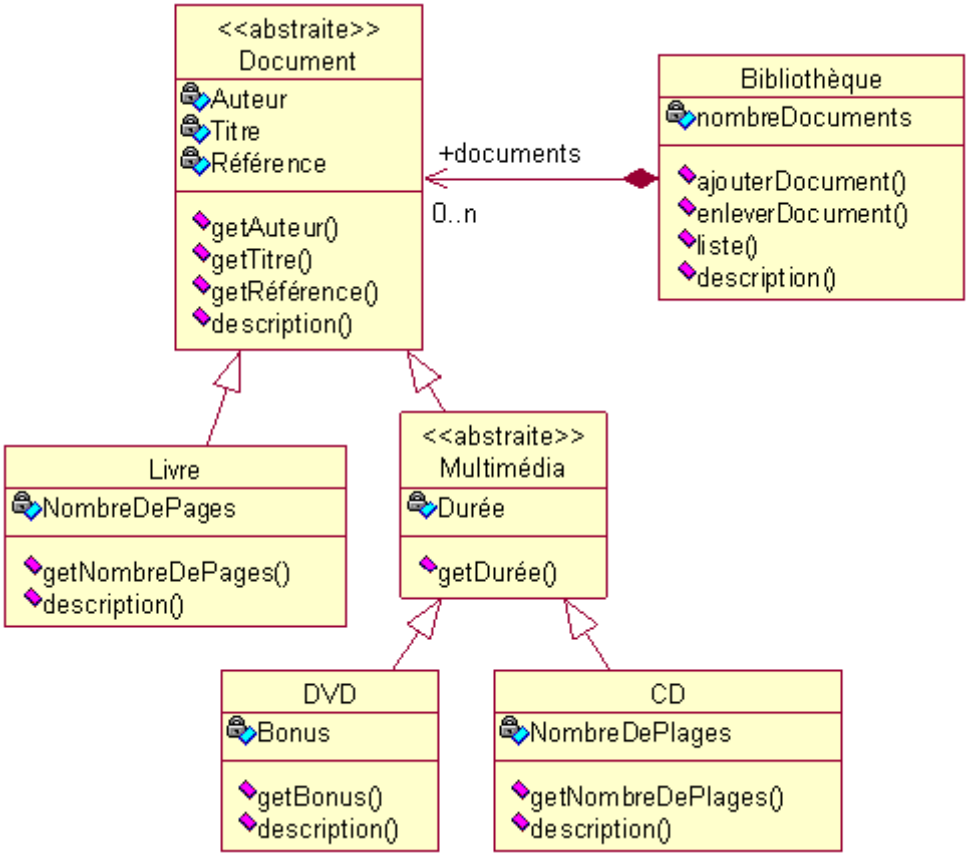


Figure 8.Exemple de polymorphisme **[19]**

3. Le cycle de base d'un moteur d'inférence [18]

Le moteur d'inférence (appelé parfois moteur de résolution) est le cœur d'un système expert. Celui-ci peut être indépendant du domaine d'application et il doit être capable d'exploiter la

base des connaissances afin de résoudre les problèmes posés par les utilisateurs. En effet, il doit être capable de voir quelles sont les règles applicables au vu de l'état courant de la base des faits et des faits à établir (exprimant les problèmes /questions posés par l'utilisateur). Ce processus de détection des règles applicables fait partie du cycle de travail d'un moteur d'inférence et particulièrement de sa **phase d'évaluation**.

Cette phase d'évaluation s'effectue généralement en trois étapes :

3.1 La sélection ou restriction

Cette étape concerne une première sélection d'un sous-ensemble de règles dans la base des règles qui à priori- mérite d'être considéré dans l'étape suivante. Ce choix dépend de l'état courant de la base de faits.

3.2 Le filtrage (*pattern matching*)

Parmi l'ensemble retenu à l' étape précédente, cette étape retient un sous-ensemble, dit de conflit, qui contient seulement les règles effectivement applicables qui sont, généralement, celles dont le côté gauche (côté « situation » dit aussi « prémisses ») est satisfait au vu de l'état actuel de la base de faits.

3.3 La résolution de conflits

Parmi les règles retenues suite au filtrage, le moteur décide de la (des) règle(s) qui sera (seront) exécutée(s). Ce choix est souvent indépendant du contexte d'application et de la sémantique des règles mais généralement guidé par des soucis d'efficacité (par exemple, choix des règles les moins complexes). Toutefois, des informations heuristiques ou des métarègles peuvent être exploitées pour fixer un ordre de priorité entre ces règles.

3.4 La Phase d'exécution qui, comme son nom l'indique, concerne l'exécution de la partie « action » des règles sélectionnées lors de la phase d'évaluation et à modifier en conséquence la base de faits.

Signalons ici que l'arrêt de ce cycle peut survenir aussi bien dans la phase d'évaluation (par exemple, dans le cas d'absence de règles applicables à la situation en cours) que dans la phase

d’exécution (par exemple, dans le cas de l’exécution d’une règle dont la partie « action » commande l’arrêt du cycle).

Dans la figure ci-dessus, il est illustré comment, en général, un moteur d'inférence enchaine des cycles de travail composés de ces deux phases :

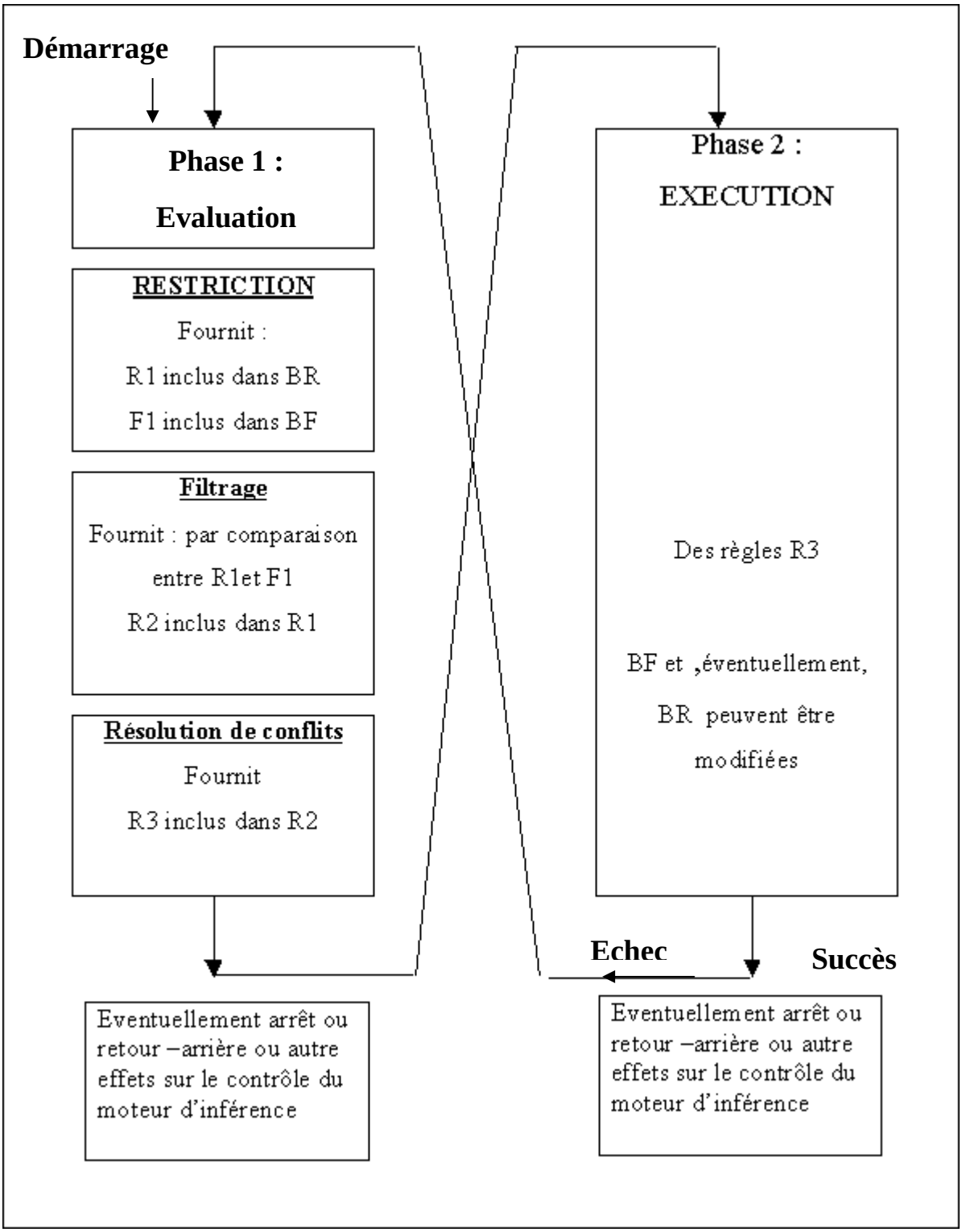


Figure 9. Cycle de base d'un moteur d'inférence

4. Stratégies de recherche [20]

4.1 La Recherche en profondeur d'abord

Dans ce cas on s'enfonce dans l'arbre de décision en passant d'état en état jusqu'à ce que le chemin se termine et cherche à explorer au maximum une possibilité en appliquant les règles de façon à obtenir le plus de détails

- Chaînage arrière : ajout du retour-arrière
- Chaînage avant : imposition d'une limite de recherche (éviter voie sans issue)

4.2 La Recherche en Largeur d'abord

Dans ce cas on parcourt à l'horizontal un niveau de l'arbre de décision avant d'aller au suivant et on commence par explorer toutes les possibilités présentes avant d'entrer dans les détails

4.3 La Recherche en profondeur limitée

La recherche en profondeur limitée combine les deux approches précédentes : profondeur d'abord et largeur d'abord.

- L'arbre des états est découpé en k niveaux.
- Une recherche en profondeur est effectuée dans chaque tranche avant de passer à la tranche inférieure.

4.4 La Recherche Heuristique

Des connaissances spécifiques sur le problème à traiter sont exploitées pour trouver une solution.

La connaissance du problème à traiter prend la forme d'une fonction d'évaluation qui mesure la « promesse » de se trouver sur un chemin intéressant.

Une heuristique garantit une bonne solution mais pas nécessairement la solution optimale.

5. Stratégies de contrôle

5.1 Système monotone

Dans un système monotone, à chaque cycle du moteur d'inférence, un groupe de règle peut être activé et leur exécution ne peut pas désactiver d'autres règles. **[18]**

5.2 Système non monotone

Dans un système non monotone, le moteur d'inférence choisit d'activer une seule règle à chaque cycle, éventuellement en fonction d'un poids précédemment attribué à cette règle. Son activation peut modifier l'état de la base de faits et donc l'activabilité de certaines règles. **[18]**

5.3 Fonctionnement irrévocable

Si au cours d'une résolution, lors de l'étape de déduction, le moteur ne peut déclencher aucune règle; il s'arrête s'il est en mode de fonctionnement irrévocable et ceci même si le but n'est pas atteint.

5.4 Fonctionnement par tentatives

Si l'étape de déduction n'a permis de déclencher aucune règle et que le but n'est pas atteint, le MI qui fonctionne par tentatives peut remettre en cause les inférences réalisées par les règles retenues lors d'une étape antérieure de résolution de conflits.

Conclusion

Depuis l'existence de l'intelligence artificielle, le domaine informatique c'est penché sur plusieurs problèmes qui se réfèrent le comportement humain et à son raisonnement, en créant des programmes de simulations, de diagnostic et de résolution de problèmes pour des domaines divers ; notons par exemple les logiciel pour le diagnostic de pannes de machines, diagnostics médicaux , ou les logiciels de simulation tel que les jeux vidéo (jeux d'échec, football, voitures, etc.) Et enfin dans la robotique dans tous ses états.

Les systèmes experts sont l'une des applications de l'intelligence artificielle qui ont quitté les laboratoires de recherche pour être utilisées dans le monde professionnel. De nombreux systèmes experts ont été implantés avec succès pour résoudre des problèmes concrets. Les utilisateurs de systèmes experts dans le domaine de la maintenance informatique, ont approuvé leur efficacité et ont été surpris de leurs précisions et de leurs performances dans la résolution de problèmes.

Les systèmes experts assistent, aident et orientent une personne qui travaille dans la maintenance informatique lui évitant parfois de refaire infiniment les tests en démontant les composants de l'ordinateur ou de chercher à chaque fois dans des manuels pour trouver l'origine des problèmes, car ces derniers sont munis d'un grand nombre de connaissances et

d’un système de déduction intelligent qui ciblera par lui-même l’origines de ou des pannes de l’ordinateur et feront l’objet de notre étude dans le chapitre suivant où nous allons faire le tour des pannes que peut subir un ordinateur.

CHAPITRE II

Etude du domaine d'expertise

«.On fait la science avec des faits, comme on fait une maison avec des pierres : mais une accumulation de faits n'est pas plus une science qu'un tas de pierres n'est une maison... ..»

Henri Poincaré

Introduction

Pour qu'un système expert puisse fonctionner correctement, il faut d'abord qu'il ait des connaissances, c'est à dire une base sur laquelle il se réfère quand il fait face à un problème. Les connaissances dont nous avons besoin sont les pannes d'un ordinateur, ces informations sont critiques et primordiales dans le développement de notre système expert vu que les pannes sont fréquentes et que la maintenance est toujours indispensable.

Il existe deux sortes de pannes sur un ordinateur, les pannes **Externes,** celles-ci font référence aux composants qui sont visible de l'extérieur, et les pannes **Internes**, désignent tous les composants non visible qui sont à l'intérieur de l'ordinateur.

Dans ce chapitre nous allons lister les différentes pannes internes et externes qu'un ordinateur peut subir afin d'enrichir le système expert et sa base de connaissance.

I. Généralités

1. Présentation des composants d'un ordinateur

Dans un ordinateur nous trouvons deux types de composants. Les composants internes qui sont des composants qui ne sont pas visibles de l'extérieur, car ils sont cachés par un boitier. Les composants externes sont soit des périphériques, ou bien des interfaces qui relient ces périphériques à l'ordinateur.

1.1 Composants Internes :

1.1.1 Processeur

Le processeur, aussi appelé unité centrale de traitement (UC ou CPU) est une grosse puce carrée connectée à la carte mère. On dit que le processeur est le cerveau du système informatique. Il est le principal responsable du traitement de l'information.

1.1.2 Mémoire Vive (RAM)

La **mémoire vive**, aussi appelée RAM (random access memory), est sous forme de barrettes qui se branchent à la carte mère. Elle permet d'emmagasiner des données pour ensuite y accéder très rapidement. Cette mémoire est cependant temporaire : aussitôt que nous éteignons l'ordinateur, tout son contenu est effacé.

La mémoire vive est une composante très importante dans un ordinateur. Plus il y a de mémoire vive, plus l'ordinateur est puissant.

1.1.2 Disque dur

Le **disque dur** est une composante matérielle essentielle de l'ordinateur, aussi importante que le processeur ou la mémoire vive. Le disque dur est comme un «entrepôt» qui stocke l'information dans l'ordinateur, même quand celui-ci est éteint. C'est lui qui conservera en permanence les travaux et informations.

1.1.3 Mémoire Flash (Facultatif)

La **mémoire flash** est une mémoire à semi-conducteurs, non volatile et réinscriptible, c'est-à-dire une mémoire possédant les caractéristiques d'une mémoire vive mais dont les données ne se volatilisent pas lors d'une mise hors tension. Ainsi la mémoire flash stocke les bits de données dans des cellules de mémoire, mais les données sont conservées en mémoire lorsque l'alimentation électrique est coupée.

1.1.4 Carte mère

La **carte mère** est l'élément constitutif principal de l'ordinateur (en anglais « *mainboard* » ou « *motherboard* »). La carte mère est le socle permettant la connexion de l'ensemble des éléments essentiels de l'ordinateur.

Comme son nom l'indique, la carte mère est une carte maîtresse, prenant la forme d'un grand circuit imprimé possédant notamment des connecteurs pour les cartes d'extension, appelées carte filles.

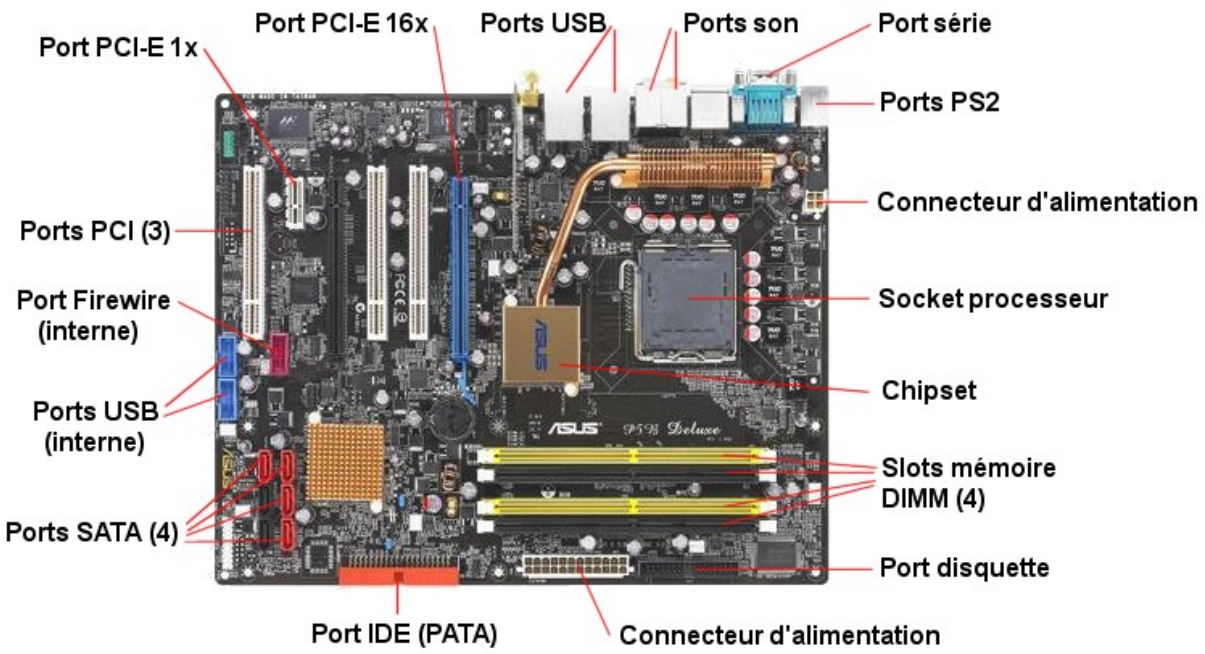


Figure10. Schéma d'une carte Mère

1.1.5 Cartes filles

Les cartes filles sont les cartes qui se connectent à la carte mère, chacune fonctionnent indépendamment de l'autre et fournit des résultats bien spécifiques (affichage, son, etc.)

1.1.5.1 Carte graphique

La **carte graphique** (en anglais *graphic adapter*), parfois appelée *carte vidéo* ou *accélérateur graphique*, est l'élément de l'ordinateur chargé de convertir les données numériques à afficher en données graphiques exploitables par un périphérique d'affichage. Cette carte peut être soit intégrée directement à la carte-mère ou bien une carte additionnelle. Le rôle de la carte graphique était initialement l'envoi de pixels graphique à un écran, ainsi qu'un ensemble de manipulation graphiques simples :

- déplacement des blocs (curseur de la souris par exemple) ;
- tracé de lignes ;
- tracé de polygones ;

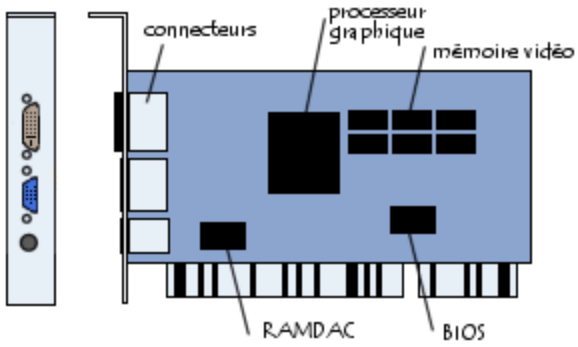


Figure 11. Schéma d'une carte graphique

1.1.5.2 Carte Son

La **carte son** (en anglais *audio card* ou *sound card*) est l'élément de l'ordinateur permettant de gérer les entrées-sorties sonores de l'ordinateur.

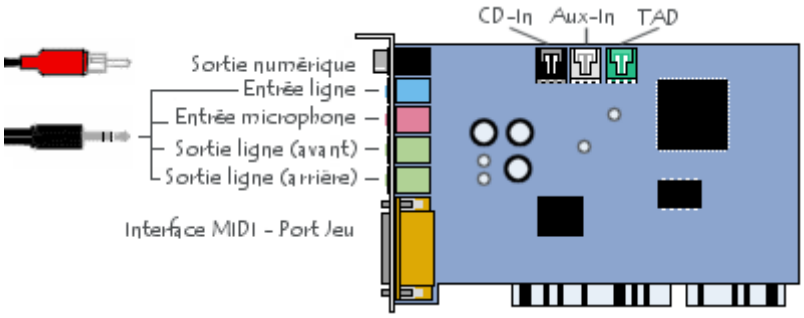


Figure 12. Schéma d'une carte son

Il s'agit généralement d'un contrôleur pouvant s'insérer dans un emplacement ISA ou PCI (pour les plus récentes) mais de plus en plus de cartes mères possèdent une carte son intégrée.

1.1.5.3 Carte Réseau

La **carte réseau** (appelée *Network Interface Card* en anglais et notée **NIC**) constitue l'interface entre l'ordinateur et le câble du réseau. La fonction d'une carte réseau est de préparer, d'envoyer et de contrôler les données sur le réseau.

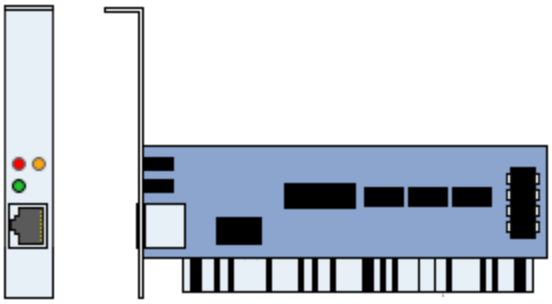


Figure 13. Représentation d'une carte réseau

La carte réseau possède généralement deux témoins lumineux (LEDs) :

- La LED verte correspond à l'alimentation de la carte ;
- La LED orange (10 Mb/s) ou rouge (100 Mb/s) indique une activité du réseau (envoi ou réception de données).

Pour préparer les données à envoyer, la carte réseau utilise un **transceiver** (transmetteur / récepteur) qui transforme les données parallèles en données séries. Chaque carte dispose d'une adresse unique, appelée *adresse MAC*, affectée par le constructeur de la carte, ce qui lui permet d'être identifiée de façon unique dans le monde parmi toutes les autres cartes réseau.

1.1.6 Ventilateur/Radiateur/ Ventirad

Dans la mesure où le processeur rayonne thermiquement, il est nécessaire d'en dissiper la chaleur pour éviter que ses circuits ne fondent. C'est la raison pour laquelle il est généralement surmonté d'un **dissipateur thermique** (appelé parfois *refroidisseur* ou *radiateur*), composé d'un métal ayant une bonne conduction thermique (cuivre ou aluminium), chargé d'augmenter la surface d'échange thermique du microprocesseur. Le dissipateur thermique comporte une base en contact avec le processeur et des ailettes afin d'augmenter la surface d'échange thermique. Un ventilateur accompagne généralement le dissipateur pour améliorer la circulation de l'air autour du dissipateur et améliorer l'échange de chaleur. Le terme « **ventirad** » est ainsi parfois utilisé pour désigner l'ensemble *Ventilateur + Radiateur*. C'est le ventilateur du boîtier qui est chargé d'extraire l'air chaud du boîtier et permettre à l'air frais provenant de l'extérieur d'y entrer. Pour éviter les bruits liés au ventilateur et améliorer la dissipation de chaleur, il est également possible d'utiliser un système de refroidissement à eau (dit **watercooling**).

1.1.7 Pile

Elle sert à alimenter le BIOS (Basic output input system)en énergie pour lui permettre de conserver sa mémoire volatile, il est parfois utile de l'enlever si le paramétrage du BIOS ne lui permet même plus de se lancer lui-même. Il faut quelques dizaines de secondes sans la pile pour que la mémoire s'efface

1.1.8 Bloc d'alimentation

Le bloc d'alimentation (*power supply unit* en anglais, souvent abrégé PSU), ou simplement l'alimentation, d'un PC est le matériel informatique l'alimentant. L'alimentation est chargée de convertir la tension électrique du secteur en différentes tensions continues TBT (très basse tension), compatibles avec les circuits électroniques de l'ordinateur.

1.1.9 Les Bus

On appelle **bus**, en informatique, un ensemble de liaisons physiques (câbles, pistes de circuits imprimés, etc.) pouvant être exploitées en commun par plusieurs éléments matériels afin de communiquer.

Les bus ont pour but de réduire le nombre de « voies » nécessaires à la communication des différents composants, en mutualisant les communications sur une seule voie de données. C'est la raison pour laquelle la métaphore d'« autoroute de données » est parfois utilisée.

1.1.9.1 Bus AGP

Le bus **AGP** (sigle de *Accelerated Graphics Port*, soit littéralement *port graphique accéléré*) est apparu en Mai 1997, sur des chipsets à base de «Slot One» un seul slot, puis est apparu par la suite sur des supports à base de Super 7 afin de permettre de gérer les flux de données graphiques devenant trop importants pour le bus PCI. Ainsi le bus AGP est directement relié au bus processeur (**FSB**, *Front Side Bus*) et bénéficie de la même fréquence, donc d'une bande passante élevée.

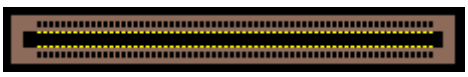


Figure 14. Connecteur AGP universel

1.1.9.2 Bus PCI

Le **bus PCI** (*Peripheral Component Interconnect*) a été mis au point par Intel le 22 juin 1992. Contrairement au bus VLB il ne s'agit pas à proprement parler d'un bus local mais d'un bus intermédiaire situé entre le bus processeur (*NorthBridge*) et le bus d'entrées-sorties (*SouthBridge*).

Figure 15. Connecteur PCI universel

1.1.9.3 Bus PCI Express

Le **bus PCI Express** (*Peripheral Component Interconnect Express*, noté *PCI-E* ou *3GIO* pour «*Third Generation I/O*»), est un bus d'interconnexion permettant l'ajout de cartes d'extension dans l'ordinateur. Le bus PCI Express a été mis au point en juillet 2002. Contrairement au bus PCI, qui fonctionne en interface parallèle, le bus **PCI Express** fonctionne en interface série, ce qui lui permet d'obtenir une bande passante beaucoup plus élevée que ce dernier.

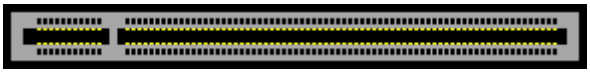


Figure 16. Connecteur PCI Express

1.1.9.4 Bus ISA

La version originale du **bus ISA** (*Industry Standard Architecture*), apparue en 1981, c'était un bus d'une largeur de 8 bits cadencé à une fréquence de 4,77 MHz.

Figure 17. Connecteur ISA

1.2 Composants Externes :

1.2.1 Interfaces d'entrée-sortie

1.2.1.1 Port Série et Port parallèle

- **Ports séries :**Les ports série (également appelés RS-232, nom de la norme à laquelle ils font référence) représentent les premières interfaces ayant permis aux ordinateurs d'échanger des informations avec le "*monde extérieur*". Le terme *série* désigne un envoi de données *via* un fil unique : les bits sont envoyés les uns à la suite des autres.

- **Ports parallèles :** La transmission de données en parallèle consiste à envoyer des données simultanément sur plusieurs canaux (fils). Les ports parallèles présents sur les ordinateurs personnels permettent d'envoyer simultanément 8 bits (un octet) par l'intermédiaire de 8 fils.

1.2.1.2 Port USB

Le bus **USB** (*Universal Serial Bus*, en français *Bus série universel*) est, comme son nom l'indique, basé sur une architecture de type série. Il s'agit toutefois d'une interface entrée-sortie beaucoup plus rapide que les ports série standards. L'architecture qui a été retenue pour ce type de port est en série pour deux raisons principales :

- L'architecture série permet d'utiliser une cadence d'horloge beaucoup plus élevée qu'une interface parallèle, car celle-ci ne supporte pas des fréquences trop élevées (dans une architecture à haut débit, les bits circulant sur chaque fil arrivent avec des d écalages, provoquant des erreurs) ;
- Les câbles série coûtent beaucoup moins cher que les câbles parallèles.

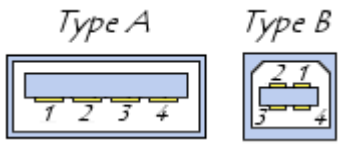


Figure 18. Connecteur USB

1.2.1.3 Le Connecteur FireWire

Le bus *IEEE 1394*(nom de la norme à laquelle il fait référence) a été mis au point à la fin de l'année 1995 afin de fournir un système d'interconnexion permettant de faire circuler des données à haute vitesse en temps réel. La société *Apple* lui a donné le nom commercial «**Firewire** »

Il s'agit ainsi d'un port, équipant certains ordinateurs, permettant de connecter des périphériques (notamment des caméras numériques) à très haut débit.

Figure 19. Connecteur FireWire

1.2.1.4 ATA ou IDE

Le standard **ATA** (*Advanced Technology Attachment*) est une interface standard permettant la connexion de périphériques de stockage sur les ordinateurs de type PC. Le standard ATA a été mis au point le 12 mai 1994 par l'ANSI (document X3.221-1994).

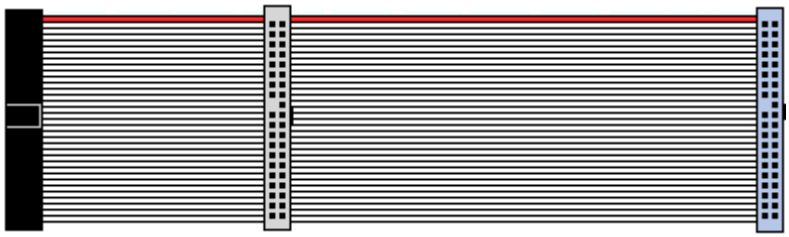


Figure 20. Connecteur IDE

1.2.1.5 Serial ATA (SATA ou S-ATA)

Le standard Serial ATA est apparu en février 2003 afin de pallier les limitations de la norme ATA (plus connue sous le nom "*IDE*" et rétro-activement appelée *Parallel ATA*), qui utilise un mode de transmission en parallèle. En effet, le mode de transmission en parallèle n'est pas

prévu pour supporter des fréquences élevées en raison des problèmes liés aux interférences électromagnétiques entre les différents fils.

Figure 21. Serial ATA

1.2.2 Périphériques

1.2.2.1 Clavier

Le **clavier** (en anglais **keyboard**) permet, à la manière des machines à écrire, de saisir des caractères (lettres, chiffres, symboles ...), il s'agit donc d'un périphérique d'entrée essentiel pour l'ordinateur, car c'est grâce à lui qu'il est possible d'envoyer des commandes.

Figure 22. Clavier

1.2.2.2 Souris

La **souris** (en anglais «*mouse*» ou «mice») est un périphérique de pointage (en anglais *pointing device*) servant à déplacer un curseur sur l'écran et permettant de sélectionner, déplacer, manipuler des objets grâce à des boutons. On appelle ainsi «**clic**» l'action consistant à appuyer (*cliquer*) sur un bouton afin d'effectuer une action.

1.2.2.3 Imprimante

L'**imprimante** (en anglais *printer*) est un périphérique permettant de faire une sortie imprimée (sur papier) des données de l'ordinateur.

1.2.2.4 Scanner

Un **scanner** (anglicisme pour le mot français «*numériseur*») est un périphérique d'acquisition permettant de numériser des documents, c'est-à-dire de transformer un document papier en image numérique.

1.2.2.5 Clé USB ou bien Flash Disk

La clef USB sert à remplacer les disquettes pour le transport des documents. Elle peut contenir beaucoup plus de documents qu'une disquette. Elle est aussi plus petite et plus résistante. Elle se branche à l'ordinateur par un port USB.

1.2.2.6 Lecteur et Graveur CD/DVD

Est un matériel qui sert à lire les données à partir d'un CD ou DVD Rom, et d'extraire ces donnais la, et s'ils possèdent la fonction de gravure alors ils peuventinsérer les données qui se trouvent dans l'ordinateur.

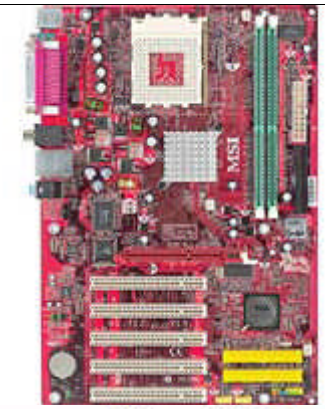
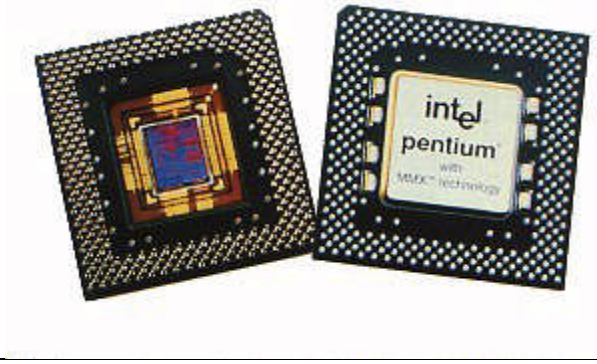
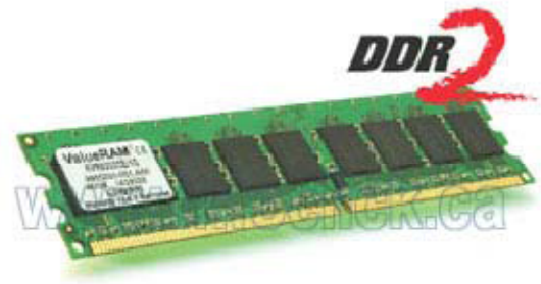
1.2.2.7 Lecteur de disquettes

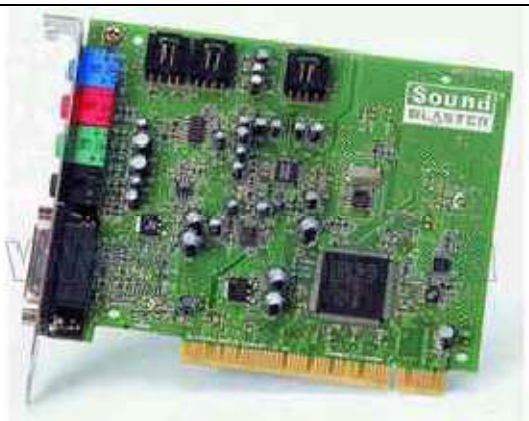
Les disquettes sont des supports qui permettent de transporter des documents informatiques d'un ordinateur à un autre. Ex: texte. Il peut contenir des documents peu volumineux ce qui en fait un support très limité pour le transport et le transfert de données.

1.2.2.8 Carte mémoire

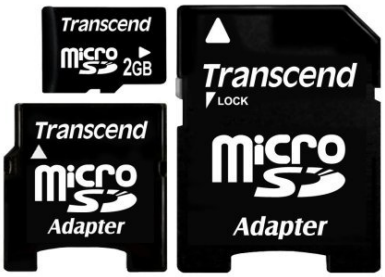
Une carte mémoire est une unité de stockage de données numériques utilisée le plus souvent pour le stockage des clichés numériques dans les appareils photo numériques (APN), pour la sauvegarde de données diverses, mais aussi dans des lecteurs des appareils électroniques professionnels.

2. Tableau récapitulatif des composants d'un ordinateur

Photographie	Description	Interne/externe	Entrée/sortie
	Carte mère Pièce maitresse de l'ordinateur : Cœur de l'ordinateur	Interne	**Entré/ sortie**
	Processeur CPU ou unité de calcul	Interne	**Entrée sortie**
	Mémoire vive (RAM) : stockage volatile mais très performant	Interne	**Entrée/sortie**
	Carte vidéo : Traduit les signaux électriques en image, vidéo, graphisme	Interne	**Entrée/sortie**

	Carte son : Traduit les signaux électriques en signaux sonores	Interne	**Entrée/Sortie**
	Modem : Pour Modulation/Dé modulation du signal de longue portée	Interne	**Entrée/sortie**
	Carte réseau : Carte servant pour se connecter vers l'extérieur (en réseau Local ou sur Internet)	Interne	**Entrée/sortie**
	Carte Wi-Fi : Une carte réseau sans fil	Interne	**Entrée/sortie**
	Disque dur : Sert a emmagasiner les données	Interne	**Entrée/sortie**

	Clavier : Sert à insérer de l'écriture ou bien des commandes à l'ordinateur	Externe	**Entrée**
	Souris : Contrôle les mouvements du curseur du Système d'exploitation	Externe	**Entrée**
	Baffles : Appareil de diffusion de son	Externes	**Sortie**
	Imprimante : sert à traduire images depuis l'ordinateur, pour le mettre sur papier	Externe	**Sortie**
	Scanner : traduit le contenu d'un papier, en image représentée sur l'écran	Externe	**Entrée**
	casque d'écoute et micro : Appareil servant pour la communication via le son	Externe	**Entrée Sortie**

	Web Cam ; appareil servant pour la communication via la vidéo	Externe	**Entrée**
	Lecteur Graveur CD/DVD : comme son nom l'indique il lit le contenu des CD/DVD et écrire dessus s'il en a la possibilité.	Externe	**Entrée sortie**
	Lecteur de disquette : comme son nom l'indique il lit le contenu des disquettes et peut écrire dessus aussi.	Externe	**Entrée**
	Ecran : Appareil servant pour l'affichage, il est lié à la carte vidéo	Externe	**Sortie**
	Carte mémoire : Support de stockage	Externe	**Entrée /sortie**

	Flash Disk Support de stockage externe	Externe	**Entrée/sortie**

Tableau 1. Composants internes et externes d'un ordinateur

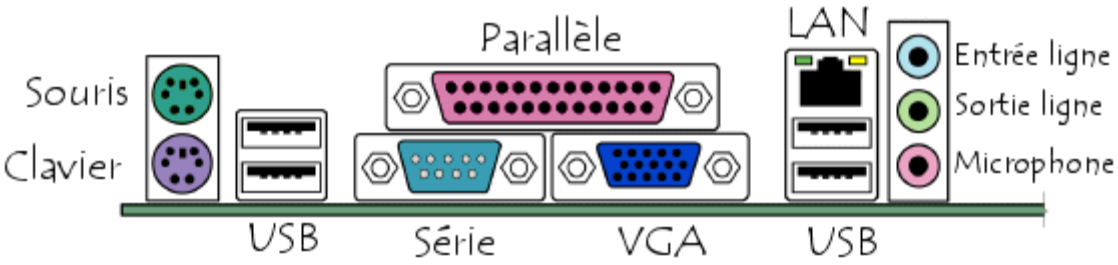


Figure 23. Composants externes d'un boitier

II. Domaine d'expertise

1 Etude des pannes d'un ordinateur

1.1 Pannes de composants Externes d'un ordinateur :

Dans le tableau ci-dessous sont présentées les pannes subis par les composants externes d'un 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

Tableau 2. Liste des pannes externes

1.2 Pannes de composants internes d'un ordinateur

Dans le tableau ci-dessous sont listées les pannes subis par les composants internes d'un ordinateur

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	

Câble IDE/ATA/SATA	**Aucune**	Aucun accès au disque dur
	Coupure ou égratignure sur le Câble	Ecran Bleu
		Blocage du PC

Tableau 3. Liste des pannes Internes

1.3 Code d'écran Bleu

Dans certains cas, il y a des erreurs tellement gênantes pour le système qu'il ne démarre plus, ou bien redémarre en plein milieu de la session et ceci donne lieu à des écrans bleus

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 4. Code d'écran bleu

1.4 Combinaisons de Bips sonores

Quand un ordinateur tombe en panne et quand c'est un organe critique qui est touché tel que le processeur alors, au démarrage de l'ordinateur on entend une combinaison de bips sonores chacune désigne un organe bien précis.

Nb : Nous avons un certain nombre de bips, une pause, de nouveau des bips, une pause etc.

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)**
1-3-1-3	**Erreur de la Carte mère**
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 5. Combinaison de Bips

Conclusion

Dans ce chapitre nous avons explorés les composants d'un ordinateur et nous avons trouvé se compose de plusieurs organes qui contribuent à sa mise en marche et à son bon fonctionnement. Mais, tous ces composants sont sujets à des pannes ou à des disfonctionnement, et vu qu'il existe plusieurs symptômes pour chaque pannes de composants, il devient complexe de trouver quel est le matériel qui ne fonctionne pas ou ne marche pas bien. Le fait de tout vérifier à chaque fois prend beaucoup de temps, c'est pour cela qu'il y a nécessité d'utiliser un système expert qui aide à faire des diagnostics plus rapidement où la marge d'erreur et moindre par rapport à une personne.

Maintenant que nous avons récoltés les informations qui nous intéressent, nous allons passer à la conception de notre système expert.

CHAPITRE III

Conception du système

«C'est une erreur capitale que de bâtir des théories tant qu'on n'a pas de données. Insensiblement, on se met à torturer les faits pour les faire cadrer avec les théories, au lieu d'adapter les théories aux faits....»

Sir Arthur Conan Doyle

Introduction

Maintenant que nous avons fait une étude sur les pannes que pourrait subir un ordinateur, et récolté des informations qui sont primordiales pour la construction du système, nous allons nous pencher dans ce chapitre sur la conception du système, elle aussi critique pour le développement d'un système expert valide, sure et survolant tous les cas possible et répondant aux besoins des utilisateurs, que nous appellerons COMBREDES (***Com***puter's ***Bre***ak***D***own ***E***xpert **S**ystem).

Pour concevoir ce système expert nous choisirons un formalisme de conception qui est l'UML (Unified Modelingl anguage) pour sa flexibilité, avoir un gain de précision et parce que c'est un support de communication performant qui offre plusieurs diagrammes détaillés afin de faciliter la compréhension. En ce qui concerne la démarche, nous choisirons la démarche d'un processus de conception qui est le 2 TUP (2 Tracks Unified Process) pour son efficacité et pour sa capacité à travailler dans deux axes différents qui sont la conception et la réalisation, ce qui permet un gain de temps énorme tout ceci en faveur des utilisateurs finaux.

I Généralités

1. UML

1.1 Définition [21]

UML se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue. UML unifie à la fois les notations et les concepts orientés objet. Il ne s'agit pas d'une simple notation graphique, car les concepts transmis par un diagramme ont une sémantique précise et sont porteurs de sens au même titre que les mots d'un langage.

UML unifie également les notations nécessaires aux différentes activités d'un processus de développement et offre, par ce biais, le moyen d'établir le suivi des décisions prises, depuis l'expression de besoin jusqu'au codage. Dans ce cadre, un concept appartenant aux exigences des utilisateurs projette sa réalité dans le modèle de conception et dans le codage.
Le fil tendu entre les différentes étapes de construction permet alors de remonter du code aux besoins et d'en comprendre les tenants et les aboutissants. En d'autres termes, on peut retrouver la nécessité d'un bloc de code en se référant à son origine dans le modèle des besoins.

1.2 Les Principaux diagrammes UML [21]

UML s'articule sur de 13 diagrammes différents, dont 4 nouveaux diagrammes introduits par UML 2.0. Chacun d'eux est dédié à la représentation d'un système logiciel suivant un point de vue particulier. Par ailleurs, UML modélise le système suivant deux modes de représentation : l'un concerne la structure du système pris « au repos », l'autre concerne sa dynamique de fonctionnement. Les deux représentations sont nécessaires et complémentaires pour schématiser la façon dont est composé le système et comment ses composantes fonctionnent entre elles.

1.2.1 Six diagrammes structurels :

– **Diagramme de classes** : Il montre les briques de base statiques : classes, associations, interfaces, attributs, opérations, généralisations, etc.
– **Diagramme d'objets**: Il montre les instances des éléments structurels et leurs liens à l'exécution.
– **Diagramme de packages** : Il montre l'organisation logique du modèle et les relations entre packages.
– **Diagramme de structure composite** – Il montre l'organisation interne d'un élément statique complexe.
– **Diagramme de composants** – Il montre des structures complexes, avec leurs interfaces fournies et requises.
– **Diagramme de déploiement** – Il montre le déploiement physique des « artefacts » sur les ressources matérielles.

1.2.2 Sept diagrammes comportementaux :

– **Diagramme de cas d'utilisation :** Il montre les interactions fonctionnelles entre les acteurs et le système à l'étude.

– Diagramme de vue d'ensemble des interactions : Il fusionne les diagrammes d'activité et de séquence pour combiner des fragments d'interaction avec des décisions et des flots.

– **Diagramme de séquence :** Il montre la séquence verticale des messages passés entre objets au sein d'une interaction.

– **Diagramme de communication** : - Il montre la communication entre objets dans le plan au sein d'une interaction.

– **Diagramme de temps :** Il fusionne les diagrammes d'états et de séquence pour montrer l'évolution de l'état d'un objet au cours du temps.

– **Diagramme d'activité** : Il montre l'enchaînement des actions et décisions au sein d'une activité.

– **Diagramme d'états** : Il montre les différents états et transitions possibles des objets d'une classe.

2. Le processus 2TUP

2TUP signifie « 2 Track Unified Process ». C'est un processus UP qui répond aux caractéristiques. Le processus 2TUP apporte une réponse aux contraintes de changement continuel imposées aux systèmes d'information de l'entreprise. En ce sens, il renforce le contrôle sur les capacités d'évolution et de correction de tels systèmes. « 2 Track » signifie littéralement que le processus suit deux chemins. Il s'agit des chemins « fonctionnels » et « d'architecture technique », qui correspondent aux deux axes de changement imposés au système informatique. **[21]**

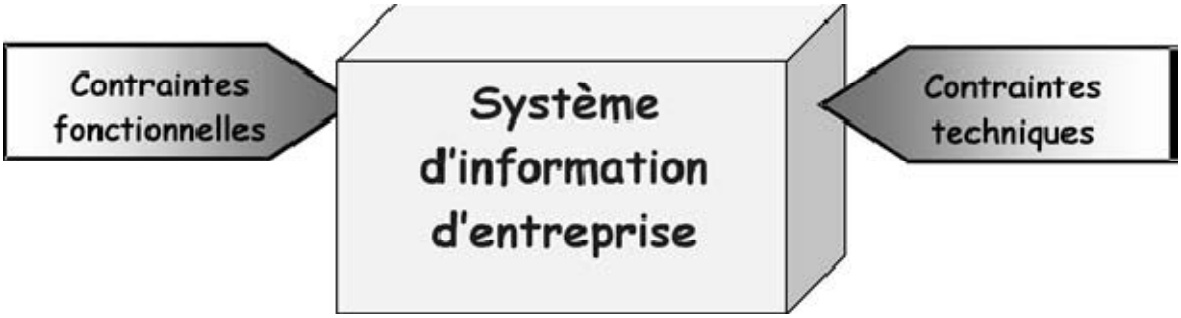


Figure 24. Le système d'information soumis à deux natures de contraintes **[21]**

- **La capture des besoins fonctionnels**, qui produit un modèle des besoins focalisé sur le métier des utilisateurs.
- **L'analyse**, qui consiste à étudier précisément la spécification fonctionnelle de manière à obtenir une idée sur ce que va réaliser le système en termes de métier.

L'architecture technique (**la branche droite**) comporte **:**

- **La capture des besoins techniques** : recense toutes les contraintes et les choix dimensionnant la conception du système. Les outils et les matériels sélectionnés ainsi que la prise en compte de contraintes d'intégration avec l'existant conditionnent généralement des pré-requis d'architecture technique.
- **La conception générique** : définit ensuite les composants nécessaires à la construction de l'architecture technique. Cette conception est la moins dépendante possible des aspects fonctionnels. Elle a pour objectif d'uniformiser et de réutiliser les mêmes mécanismes pour tout le système.

La branche du milieu comporte **:**

- **La conception préliminaire** : représente une étape délicate, car elle intègre le modèle d'analyse dans l'architecture technique de manière à tracer la cartographie des composants du système à développer.
- **La conception détaillée** : étudie ensuite comment réaliser chaque composant.
- **L'étape de codage**, qui produit ces composants et teste au fur et à mesure les unités de code réalisées.

L'étape de recette, qui consiste enfin à valider les fonctions du système développé

La branche du milieu : à l'issue des évolutions du *modèle fonctionnel* et de *l'architecture technique*, la réalisation du système consiste à *fusionner* les résultats des 2 branches. Cette fusion conduit à l'obtention d'un processus en forme de **Y.**

Cette branche comporte les étapes suivantes :

- La conception préliminaire.
- La conception détaillée.
- Le codage.
- L'intégration.

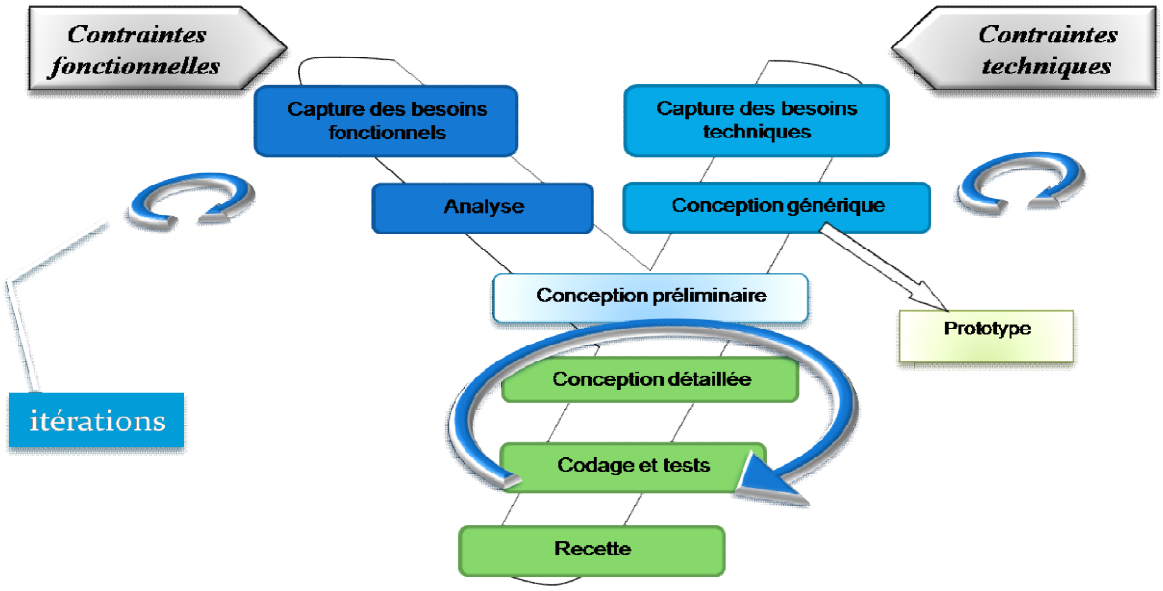


Figure 25. Le processus de développement en Y

3. La relation entre UML et 2TUP

UML est le formalisme de modélisation objet standard du processus **2TUP**. Chaque étape de ce dernier correspond aux différents diagrammes d'UML.

La première étape du côté fonctionnel, et une première ébauche des diagrammes de séquence pour connaitre les scénarios d'utilisation du système. Lors de la seconde étape, nous commencerons à faire un diagramme de classes (il représente les classes intervenantes dans le système).

Du côté technique, nous allons parler des contraintes liées au logiciel et des outils de développement.

Puis, à la jonction des deux branches, lors de la conception préliminaire, on complète le diagramme de déploiement et on fait le diagramme de composants (qui représente la structure physique des composants du système. Les composants du système sont les fichiers, les bibliothèques, les bases de données).

Enfin, lors de la conception détaillée, on affine la majorité des diagrammes effectués précédemment afin de permettre aux développeurs de coder les fonctionnalités de manière très précise et adaptée aux besoins des utilisateurs. **[23]**

II. Conception en utilisant le modèle en Y :

1. Capture des besoins techniques (Branche de droite)

- **sécurité de l'application**

L'application doit être absolument sécurisée, car la base de règle est le cœur de l'application, alors une violation de la sécurité de celle-ci, mettra en péril le bon fonctionnement du système.

- **Interface utilisateur**

L'interface doit être simple, intuitive et facile à manipuler pour l'utilisateur pour que celui-ci ne trouve pas de problèmes dans l'utilisation et ne se perd pas en cherchant la cohérence.

- **Développement de l'application**
  - Environnement de développement : Delphi 7
  - Langage de programmation : Pascal Objet
  - SGBD : Paradox 7
  - Environnement d'exécution : Windows XP

2. Capture des besoins fonctionnels

2.1 Présentation de COMBREDES

Le système expert COMBREDES, que nous allons réaliser est un système qui permet de diagnostiquer les pannes d'un ordinateur et de détecter quel organe est défectueux, en se basant sur des symptômes observés.

L'utilisateur de COMBREDES devra insérer les symptômes observés, autrement dit les faits ; et devra répondre tout de suite après répondre à un questionnaire généré par le système, pour plus de précision et essayer de détecter le matériel défectueux. Une fois le questionnaire terminé, le système affiche l'organe en panne, ainsi que d'autres probablement concernés par le symptôme.

2.2 Exigences liées au système

- **Coté utilisateur**

- Le système devra offrir une interface qui permettra à l'utilisateur d'accéder à l'application.
- Le système devra être capable

- D'insérer des faits (symptômes) via un formulaire facile d'utilisation et de les manipuler par la suite si nécessaire.
- Déduire les pannes relatives aux faits insérés.

- **Coté Expert**
  - Le système devra offrir une interface permettant à l'expert d'accéder à l'application
  - Le système devra reconnaitre l'expert grâce à son login et mot de passe afin de lui permettre de manipuler la base de règles.
  - Le système devra permettre à l'expert
    - D'ajouter une règle
    - De Modifier une règle
    - De Supprimer une règle

2.3 Les cas d'utilisation (UC)

2.3.1 Identification des acteurs

Un système expert est utilisé par deux types d'utilisateurs, ceux-ci sont distingués grâce à la nature des manipulations qu'ils exercent sur le système.

- **Utilisateur simple :** C'est une personne qui cherche un diagnostic sur la panne, en introduisant les faits (données) au système. Cette personne pourrait être soit un technicien, technicien supérieur en maintenance informatique ou bien un gérant de parc informatique ou enfin une personne désirant apprendre sur les pannes d'ordinateur.
- **Expert (Technicien supérieur en maintenance informatique) :** C'est la personne qui est chargée d'insérer les Nouvelles règle ou bien de les modifier ou enfin de les supprimer. En d'autres termes c'est lui qui est chargé de la manipulation de la base de règles.

2.3.2 Identification des UC

Chaque cas d'utilisation doit faire l'objet d'une définition a priori qui décrit l'intention de l'acteur lorsqu'il utilise le système et les séquences d'actions principales qu'il est susceptible d'effectuer. Ces définitions servent à fixer les idées lors de l'identification des cas d'utilisation et n'ont aucun caractère exhaustif.

Grâce aux besoins récoltés et aux acteurs détectés nous avons déduit les cas d'utilisations suivants :

- S'Authentifier
- Ajouter une/des règle
- Supprimer une /des règles
- Modifier une/des règles
- Faire le diagnostic

2.3.3 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation est un schéma qui montre les cas d'utilisation (ovales) reliés par des associations (lignes) à leurs acteurs (icône du « stick man », ou représentation graphique équivalente). Chaque association signifie simplement « participe à ». Un cas d'utilisation doit être relié à au moins un acteur. **[23]**

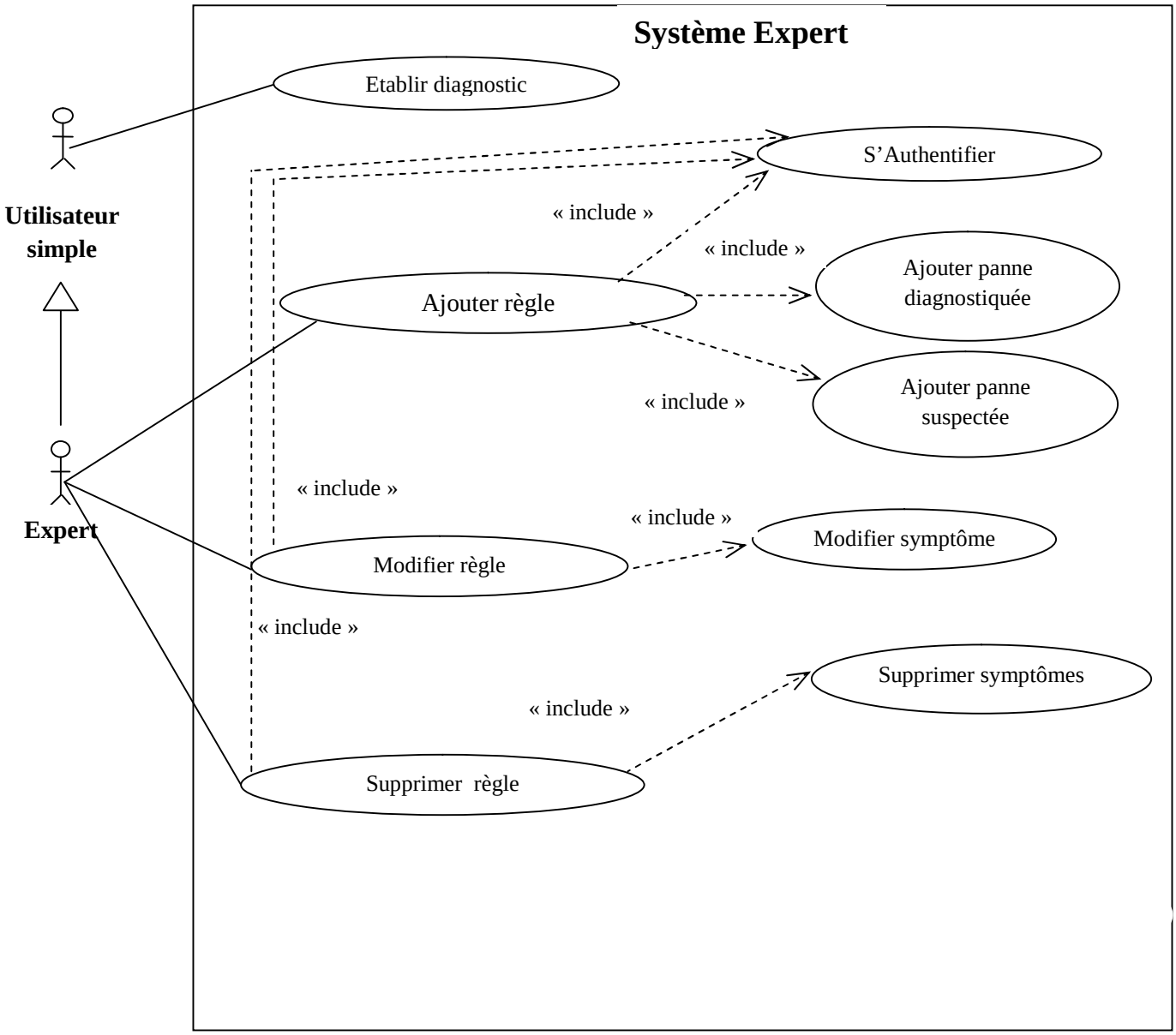


Figure 26. Diagramme de cas d'utilisation

2.3.4 Description textuelle des cas d'utilisation [23]

À chaque cas d'utilisation doit être associée une description textuelle des interactions entre l'acteur et le système et les actions que le système doit réaliser en vue de produire les résultats attendus par les acteurs.

• **Objectif :** Décrire succinctement le contexte et les résultats attendus du cas d'utilisation.

• **Acteurs concernés :** Le ou les acteurs concernés par le cas doivent être identifiés en précisant globalement leur rôle.

• **Scénario nominal :** Il s'agit là du scénario principal qui doit se dérouler sans incident et qui permet d'aboutir au résultat souhaité.

• **Scénarios alternatifs :** Les autres scénarios, secondaires ou correspondants à la résolution d'anomalies, sont à décrire à ce niveau. Le lien avec le scénario principal se fait à l'aide d'une numérotation hiérarchisée.

• **Objectif :** Décrire succinctement le contexte et les résultats attendus du cas d'utilisation.

• **Pré conditions** – Si certaines conditions particulières sont requises avant l'exécution du cas, elles sont à exprimer à ce niveau.

• **Post conditions** – Par symétrie, si certaines conditions particulières doivent être réunies après l'exécution du cas, elles sont à exprimer à ce niveau. Pour notre part, par souci de simplification nous n'avons pas traité ce point dans les exercices et études de cas présentés.

Description du cas « Etablir diagnostic »
Identification **Nom de cas :** Etablir diagnostic **But :** Décrire à l'utilisateur que ce soit l'utilisateur simple ou l'expert une série d'étapes à suivre 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 savoir quel est le matériel en panne. **Acteur principal :** L'utilisateur simple, Expert.
Séquencement Ce cas d'utilisation commence une fois que l'utilisateur aura atteint la session utilisateur. **Pré-conditions :** 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: écran Bleu, Bip sonores, autres pannes fréquentes. 2. L'utilisateur rempli le questionnaire et clic sur le diagnostic. 3. Le système déduit la panne correspondante aux symptômes et donnera 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. Le système indique d'autres alternatives de panne de matériel. **Post-conditions** - Le système devra afficher un résultat même s'il est approximatif.

Tableau 6. Description du cas « Etablir diagnostic».

Description du cas « Ajouter règle »

Identification

Nom de cas : Ajouter règle.

But : Décrire 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 commence une fois que l'utilisateur aura atteint l'interface permettant d'ajouter des règles à la base de règles.

Pré-conditions :

- 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 7. Description du cas « Ajouter règle».

Description du cas « Modifier règle»

Identification

Nom de cas : Modifier règle.

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

Acteur principal : L'expert.

Séquencement

Le cas d'utilisation commencera lorsque l'expert aura atteint l'interface permettant de modifier des règles.

Pré-conditions :

- Au préalable l'expert doit s'authentifier.
- La base des règles doit contenir au moins une règle 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 8. Description du cas « Modifier règle».

Description du cas « Supprimer règle »
Identification **Nom de cas :** Supprimer règle. **But :** Décrire 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. **Pré-conditions :** - Au préalable l'expert doit s'authentifier. - 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 9. Description du cas « Supprimer règle».

Maintenant que nous avons fait la capture des besoins fonctionnels et technique, maintenant nous allons nous approfondir dans la partie gauche du modèle en Y qui est l'analyse.

II.1. Analyse

L'Analyse consiste à étudier précisément la spécification fonctionnelle de manière à obtenir une idée de ce que va réaliser le système en termes de métier. Les résultats de l'analyse ne dépendent d'aucune technologie particulière. Ceci garantit la cohérence du modèle en Y par l'absence des aspects technique dans la branche de gauche. **[21]**

1. Diagramme de séquence (DSE)

L'objectif du diagramme de séquence est de représenter les interactions entre objets en indiquant la chronologie des échanges. Cette représentation peut se réaliser par cas d'utilisation en considérant les différents scénarios associés.

Un diagramme de séquence se représente globalement dans un grand rectangle avec indication du nom du diagramme en haut à gauche **[21]**

- **Messages synchrone et message asynchrone [21]**

Dans un diagramme de séquence, deux types de messages peuvent être distingués :

• **Message synchrone :** Dans ce cas l'émetteur reste en attente de la réponse à son message avant de poursuivre ses actions. La flèche avec extrémité pleine symbolise ce type de message. Le message retour peut ne pas être représenté car il est inclus dans la fin d'exécution de l'opération de l'objet destinataire du message.

•**Message asynchrone :** Dans ce cas, l'émetteur n'attend pas la réponse à son message, il poursuit l'exécution de ses opérations. C'est une flèche avec une extrémité non pleine qui symbolise ce type de message.

- **Ligne de vie [21]**

Elle représente la période de temps durant laquelle l'objet "existe". Les objets communiquent en échangeant des messages représentés sous forme de flèches. Lors de la création d'un objet, un message pointe sur le symbole de l'objet et quand il est détruit sa ligne de vie se termine par une croix en trait épais (**X**).

1.1 Diagramme de séquence de cas d'utilisation (S'Authentifier)

Lorsque l'expert choisira d'entrer dans sa session il entrera dans la page d'authentification pour rentrer dans sa session, il devra insérer un login et un mot de passe. Une fois que les deux champs sont remplis le système effectuera une vérification sur les informations insérées. Si les données sont correctes alors le système lui permettra d'accéder directement à son interface, dans le cas contraire, il pourra refaire l'insertion à chaque fois que login ou le mot de passe seront erronés.

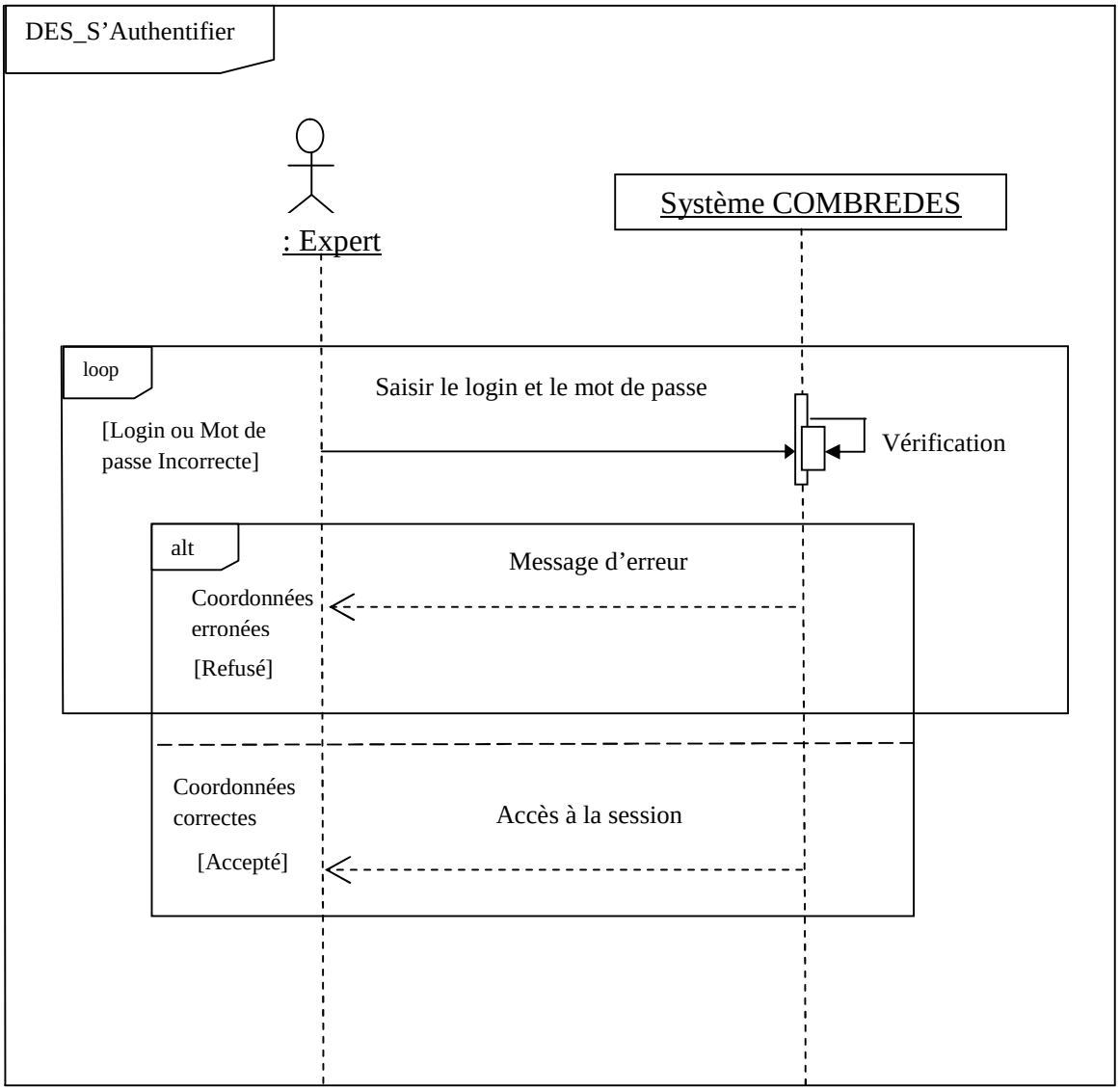


Figure 27. Diagramme de séquence de cas d'utilisation « S'Authentifier».

NB : La notation de diagramme de séquence qui sera utilisé est prise dans le livre de PASCAL ROQUES, UML2 par la pratique. Mise en œuvre guidée avec études de cas. 5eme édition 2006

1.2 Diagramme de séquence de cas d'utilisation (Etablir diagnostic)

Lorsque l'utilisateur se dirigera vers son interface pour établir des diagnostics celui-ci devra remplir un formulaire concernant la panne. Une fois le formulaire remplit le système effectuera une recherche en se basant les informations insérées afin d'établir un diagnostic. Lors qu'il aura terminé la recherche, il affichera la panne s'il l'a trouvé sinon il y aura nécessité un diagnostic avancé si nécessaire.

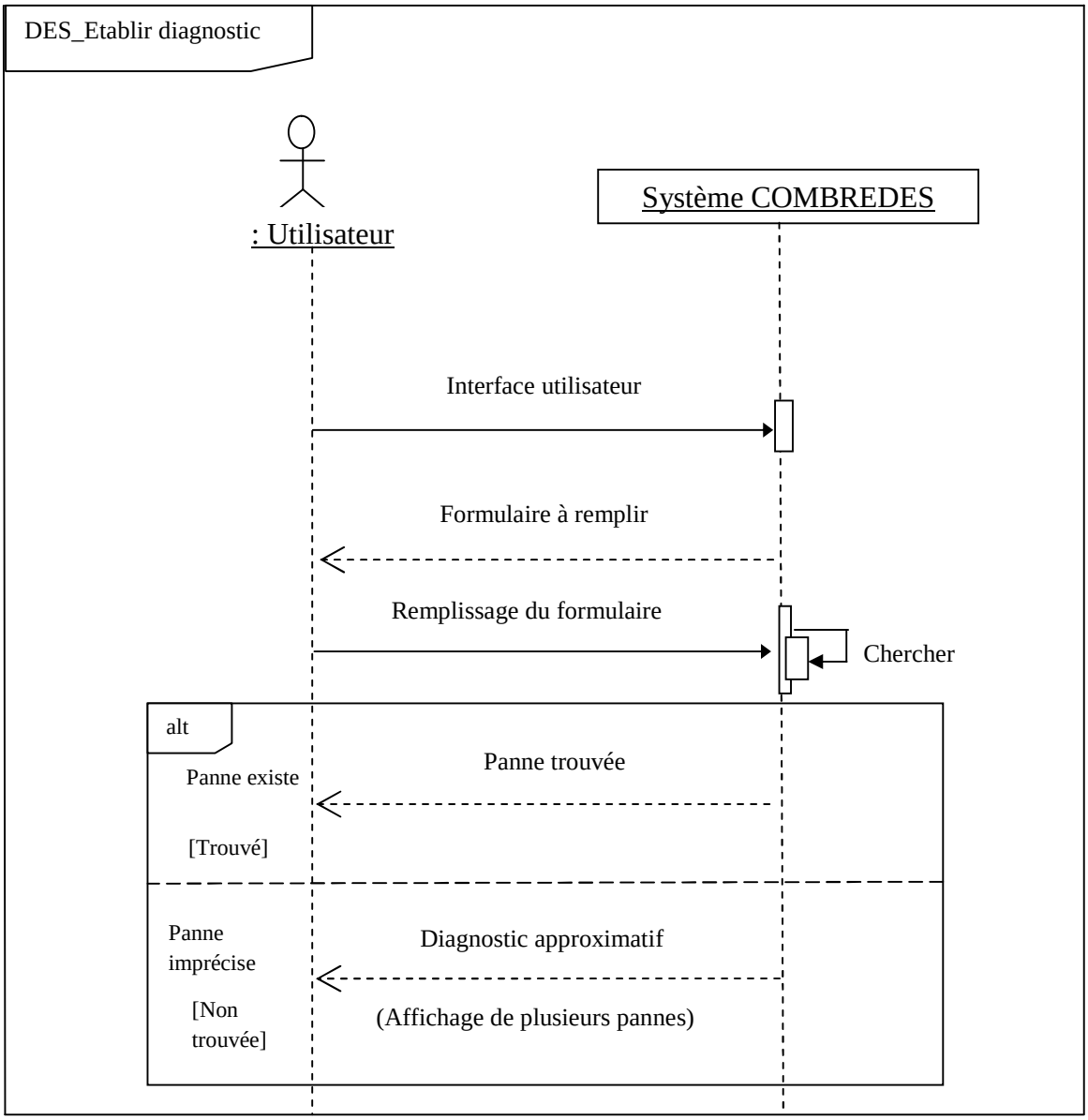


Figure 28. Diagramme de séquence de cas d'utilisation « Etablir diagnostic».

1.3 Diagramme de séquence de cas d'utilisation (Ajouter Règle)

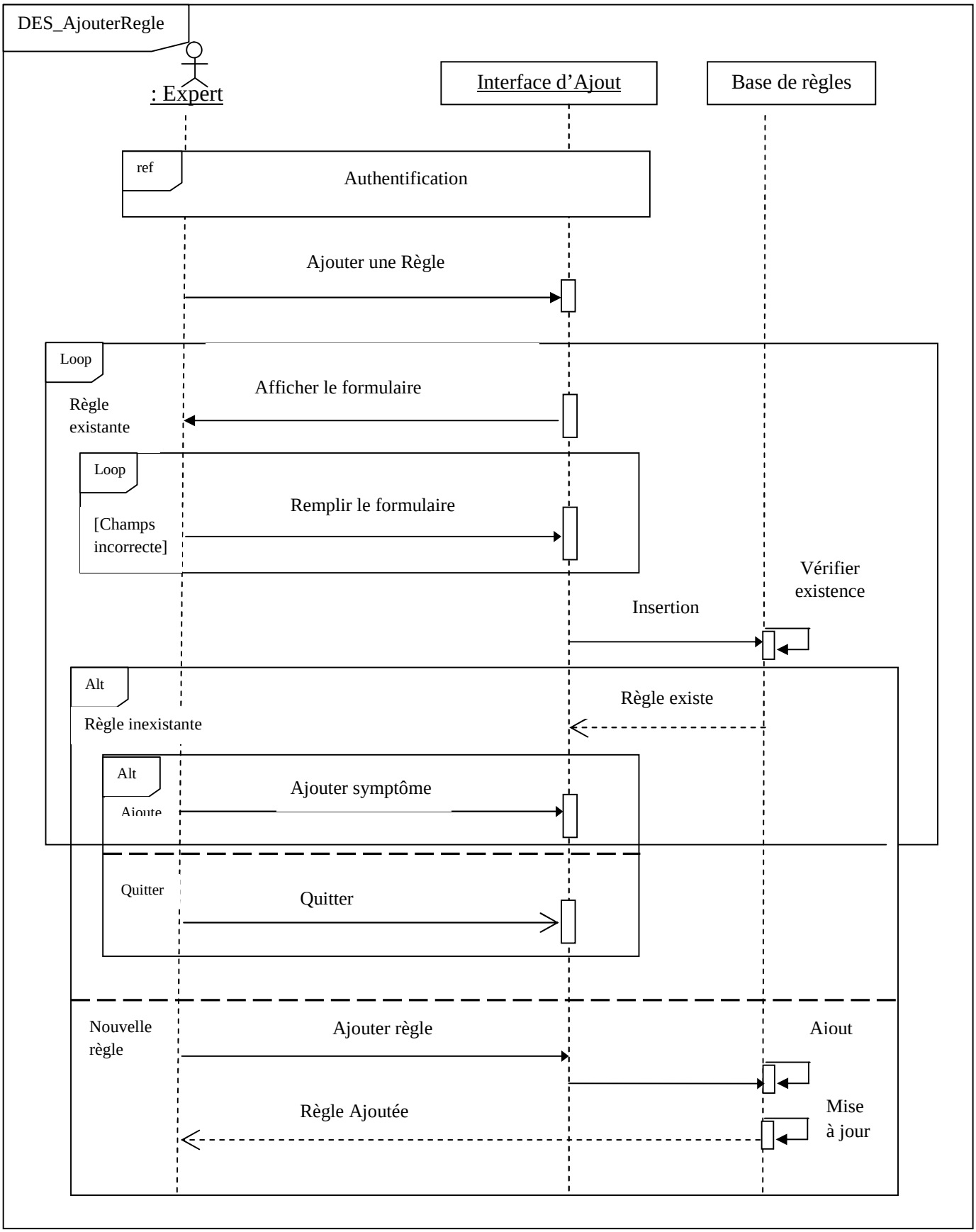


Figure 29.Diagramme de séquence de cas d'utilisation « Ajouter Règle».

1.4 Diagramme de séquence de cas d'utilisation (Modifier Règle)

Apres authentification et lorsque l'utilisateur se dirigera vers l'interface lui permettant de modifier une règle, il aura un choix entre les règles existantes dans la base de règles et choisira une en fonction de son numéro et pourra ainsi faire sa modification en remplissant un formulaire

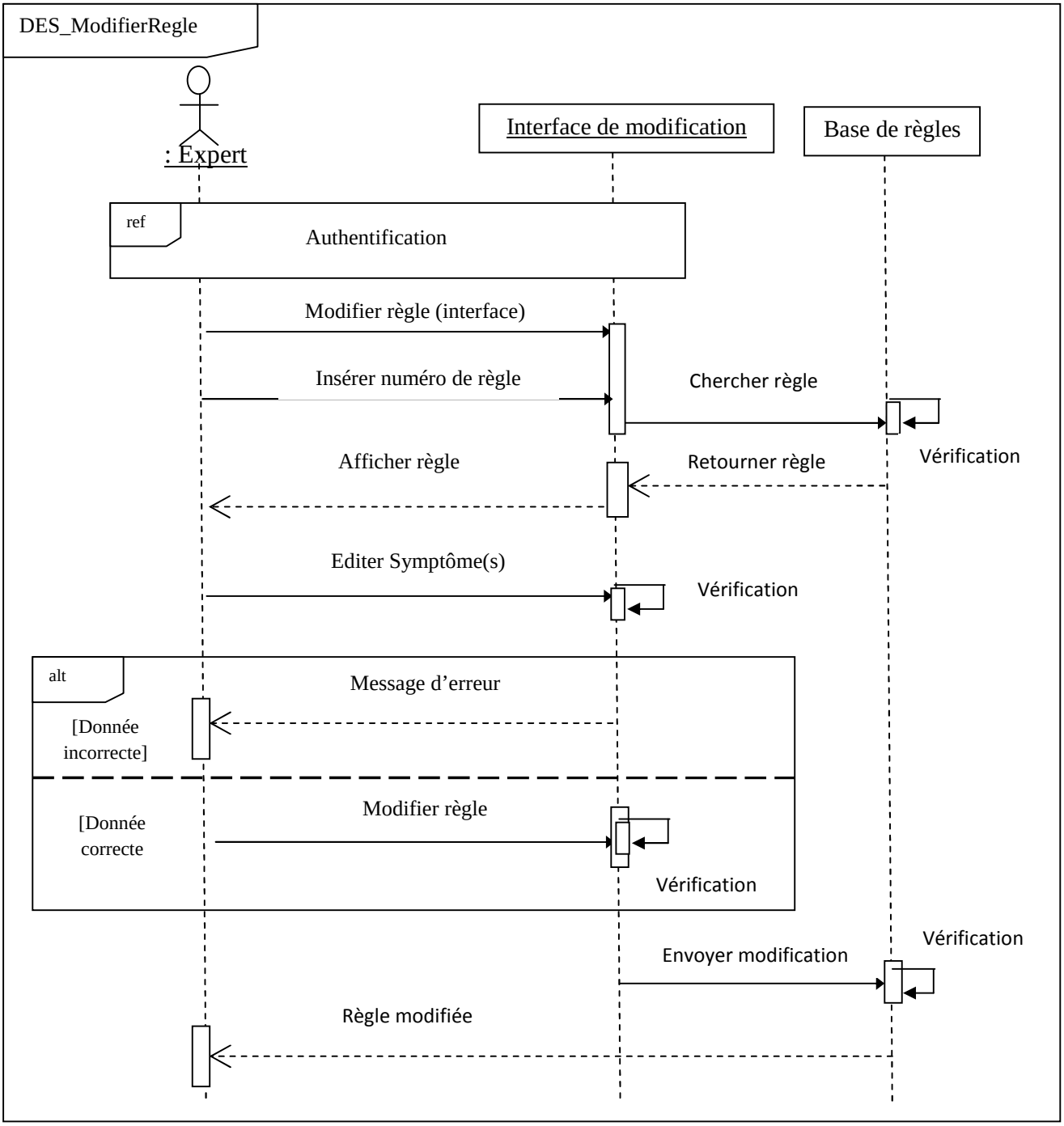


Figure 30. Diagramme de séquence de cas d'utilisation « Modifier Règle».

1.5 Diagramme de séquence de cas d'utilisation (Supprimer Règles)

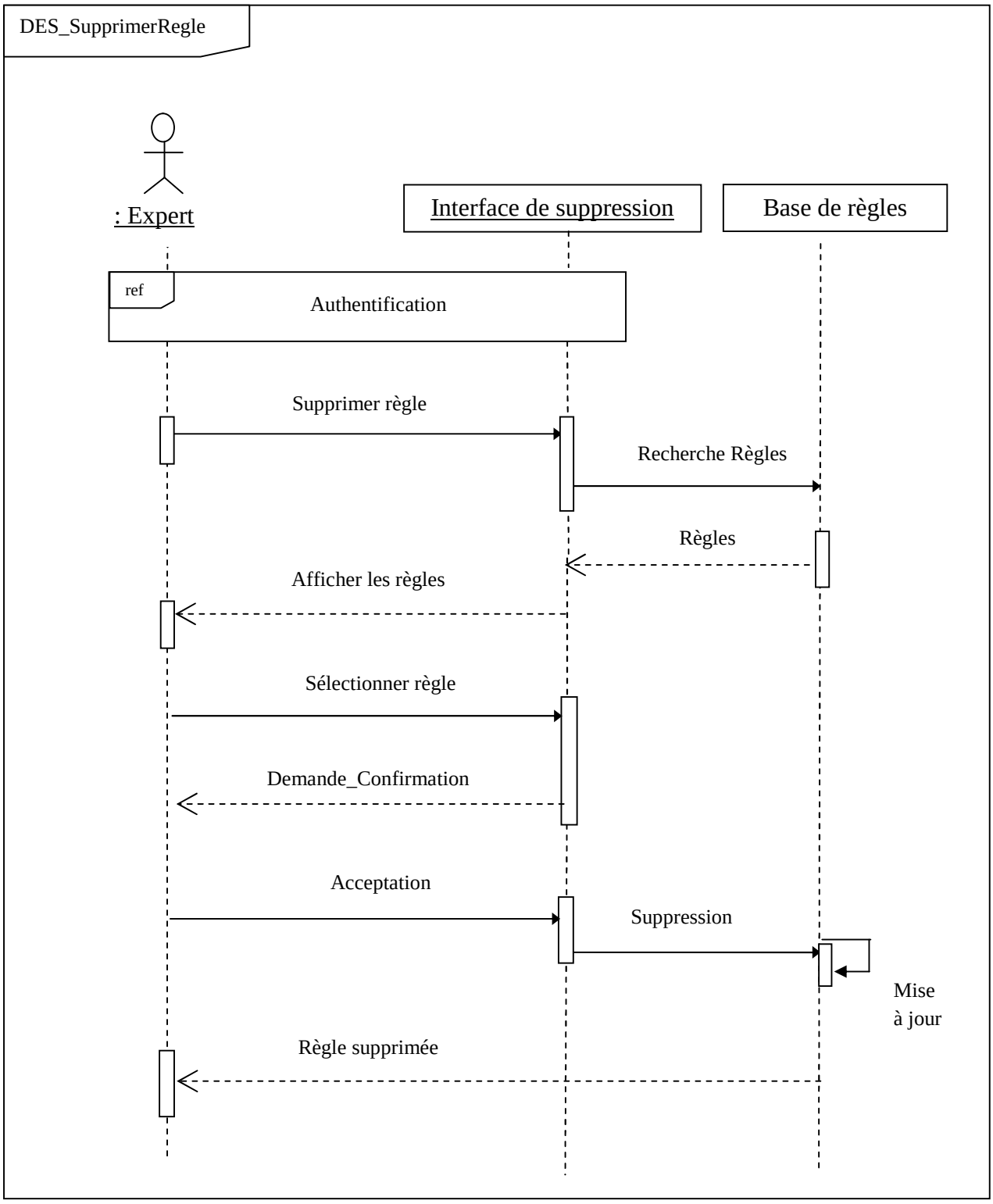


Figure 31.Diagramme de séquence de cas d'utilisation « Supprimer Règle».

II.2. Conception détaillée

1. Diagramme de Classe

Sachant que le diagramme de classe a été conçu avec un logiciel graphique des diagrammes UML qui est relationnal Rose

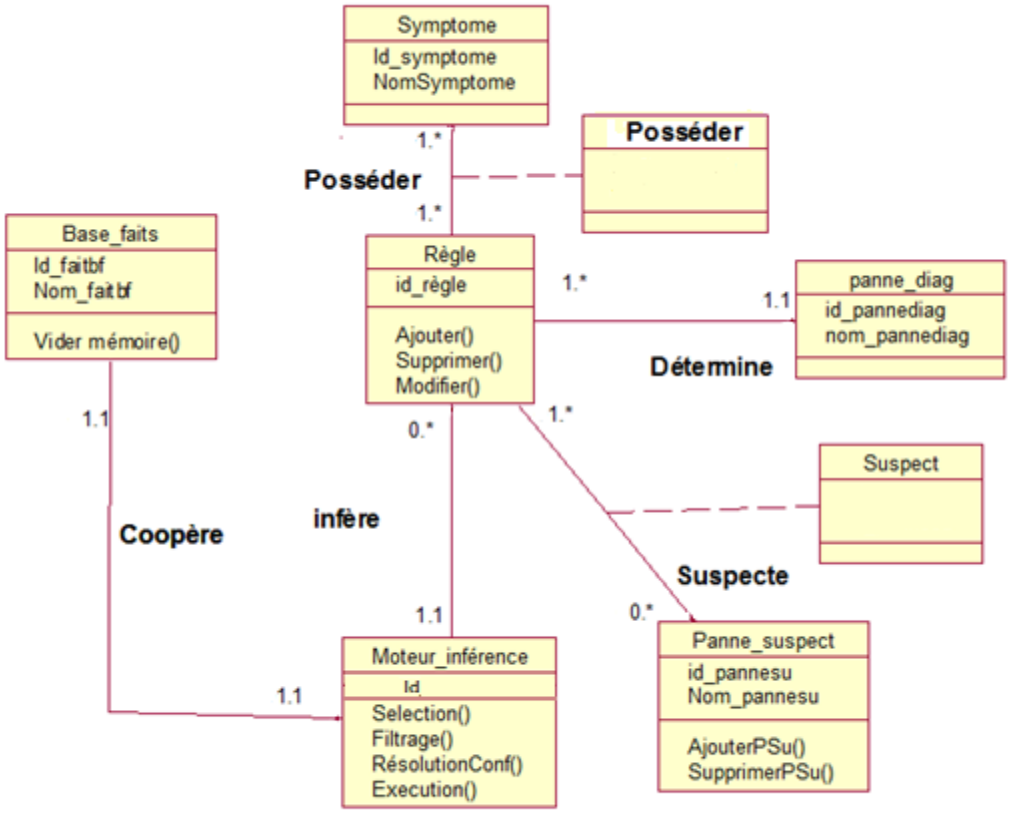


Figure 32. Diagramme de classe.

2. Dictionnaire de données :

Codification	Désignation	Type	Taille
Id_faitbf	Identifiant symptôme dans la BF	Int	
Nom_faitbf	Nom du symptôme dans la BF	Varchar	50
Id_symptome	Identifiant du symptôme	Int	
Id_règle	Identifiant de la règle	Int	
NomSymptome	Nom du Symptôme	Varchar	50

Id_panne diag	Identifiant de la panne diagnostiquée	Int	
Nom_panne diag	Nom de la panne diagnostiquée	Varchar	50
Id_pannesu	Nom de la panne suspectée	Int	
Nom_panneSU	Nom de la panne suspectée	Varchar	50
Id	Identifiant du moteur d'inférence	Int	

Tableau 10 .Dictionnaire de données

3. Modèle relationnel de données

3.1. 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 **[23] :**

- ✓ 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.

NB : Dans ce qui suit, nous avons mis pour les clés primaires {PK}qui signifie **primary key** et pour les clés étrangères {FK}qui signifie **foreign key** ces notations vue dans UML en action **[21]** .

En appliquant ces règles de passages concernant la de gestion de pièce de rechange, nous avons aboutit au schéma relationnel suivant **[21] :**

Règle (**id_regle** {PK}, symptome, **Id_pannediag** {FK}).

Panne_diag (**id_pannediag** {PK}, nom_panneDiag).

Panne_suspect({**id_pannesu**{PK}, nom_pannesu).

Base_faits (Id_faitbf {PK}, Nomfaitbf).

Suspecte (**id_regle**{FK}, **Id_pannesu**{FK}).

Symptome (**id_regle** {FK}, **Id_pannesu**{FK}).

Posséder (id_regle{FK}, **Id_symptome**{FK})

Moteur (Id_faitbf {FK}).

Conclusion

Dans ce chapitre nous avons pu remarquer l'importance de la conception pour préparer une réalisation efficace en s'appuyant sur UML, qui est un formalisme de conception très performant et adéquat à notre application et aux tâches que nous voulons réaliser, car celui-ci offre la possibilité de structurer les informations récoltées grâce aux différents diagrammes utilisés, tout ceci en travaillant avec un processus qui lie la conception à la réalisation qui est le processus 2TUP avec son modèle en Y pour avancer plus rapidement et travailler avec branches différentes afin d'aboutir à un seul objectif qui est la recette (en d'autres termes l'application finale). Dans le chapitre suivant nous allons passer à la réalisation et utiliser tout ce qui a résulté de la conception.

Conclusion générale

« On commence par
vouloir tout le bien,
et on finit par espérer
le moindre mal. »

Proverbe

Conclusion générale

Le travail maintenant terminé, il ne nous reste qu'à souhaiter que notre application réponde bien aux exigences fonctionnelles et ergonomiques de tout utilisateur, et aussi que l'application conçue c'est-à-dire le système expert pour l'aide au diagnostic de panne d'un ordinateur les aidera vraiment à bien cibler les pannes de manière stricte et efficace afin d'éviter tous les tests inutiles sur les composants.

Nous souhaitons aussi avoir bien automatisé le système en utilisant les bons concepts, les bonnes idées et les bons outils là où il faut. Automatiser un système tout en essayant de se rapprocher de la pensé humaine est très difficile parce que le niveau d'exigence est supérieur a celui d'une simple application. puisque l'utilisateur espérera toujours que l'application lui fera gagner du temps et pourra lui offrir de la précision afin bien réaliser ses tâches.

Débutant dans le domaine de la conception, nous avons eu recours à UML version 2.0 qui est un langage de modélisation très essentiel pour bien structurer, représenter et bien organiser nos idées (diagramme de classe, de séquence de case d'utilisation, etc.). UML met en œuvre un formalisme graphique qui est facile a lire et c'est l'outil idéal pour la programmation orienté objet. A travers UML nous avons découvert et utilisé le processus 2TUP qui nous a fait gagné du temps avec son modèle en Y, car malheureusement avec certaines contraintes de temps auxquelles nous avons fait face durant ces derniers temps nous n'avions pas d'autre choix que de l'utiliser car ce processus est assez intéressant vu que l'avancement de l'application et des besoins fonctionnels se sont fait en parallèles et que la fusion des deux axes ne nous a causé aucun problème; bien au contraire, il nous a permis d'avoir toujours un œil sur l'application en avançant dans nos besoins fonctionnels.

Pour notre application nous avons choisis d'utiliser Delphi 7, qui propose des interfaces agréables et le système de gestion de bases de données Paradox 7. Ce choix est du a la facilité d'utilisation et à la maniabilité de ce logiciel et surtout pour son efficacité, même si nous avons eu à faire à quelques erreurs et à quelques bugs de temps en temps , mais il reste un très bon logiciel et un outil parfait pour tout concepteur de logiciels quels qu'ils soient.

Avec toutes ces méthodes, techniques et outils nous avons pu finir notre conception et réaliser notre application, malgré l'insuffisance de temps imparti et les quelques petits obstacles organisationnels rencontrés. (Ex. aucune répartition des tâches alors que le travail en avait besoin).

Pour finir il ne nous reste qu'à espérer que notre travail soit à la hauteur des besoins et des exigences des utilisateur, et que tous les efforts fournis puissent représenter l'envie et les idées que nous avons voulus inclure et réaliser dans notre mémoire et notre projet.

Bibliographie

[1] Andreas NEUMANN Docteur de l'Ecole Nationale des Ponts et Chaussées. Thèse Doctorale : INTRODUCTION D'OUTILS DE L'INTELLIGENCE ARTIFICIELLE DANS LA PRÉVISION DE PLUIE PAR RADAR – Soutenue le 23 Novembre 2007

[2] Dominique Pastre Chercheur à l'université de Paris 5 - livre L'INTELLIGENCE ARTIFICIELLE DEFINITION - HISTORIQUE - DOMAINES – Edition Fayard 2005.

[3] J.L.Laurière livre Intelligence artificielle, Résolution de problèmes par l'homme et la machine - 1987

[4] Principes des systèmes intelligents livre de Paul Jorion Editions Fayard (23 novembre 2012).

[5] De l'intelligence humaine à l'intelligence artificielle de Hugues Bersini (12 avril 2006).

[6] Réseaux neuronaux : Une introduction accompagnée d'un modèle Java de Jean-Philippe Rennard (13 février 2006).

[7] Developing Expert Systems Livre auteur J-C. Pomerol , G.Benchimol, P. Levine (Auteurs) (décembre 1987).

[8] Schadrac KANDE KANUMUAMBIDI Université de Notre Dame du Kasayi titre : Réalisation d'un système expert pour la thérapeutique et le diagnostic des maladies de la tuberculose- 2009.

[9] Blanc M., Charron E., Freyssenet M., Le « développement » des systèmes-experts en entreprise, *Cahiers de recherche du GIP « Mutations Industrielles »*, n° 35, novembre 1989, 84 p. Édition numérique, freyssenet.com,

[10] François Denis Laboratoire d'Informatique fondamentale, Marseille Laurent Miclet ENSSAT-IRISA, Lannion «INTELLIGENCE ARTIFICIELLE : LES SYSTEMES EXPERTS ». - Juillet 2009

[11] James L. Crowley Conception de systèmes intelligents : Programmation des Systèmes Experts **2001-2002**

[12] REPRESENTATION DES CONNAISSANCES Université de Stendhal, Grenoble (France) Cédric Lopez Frédérique Segond - 2007.

[13] François Lévy Représentation des connaissances et logique modale MICR Institut Galilée - 5 mars 2006

[14] LES RESEAUX DE NEURONES ARTIFICIELS INTRODUCTION AU CONNEXIONNISME COURS, EXERCICES ET TRAVAUX PRATIQUES - Claude TOUZET Juillet 1992

[15] Laurent Audibert (LIPN - UMR CNRS 7030) Université Paris 13 – Laboratoire d'Informatique de Paris-Nord (LIPN) 4 novembre 2010

[16] Frédérique LAB Département de Recherches Linguistiques Université de Paris VII LE BULLETIN DE L'EPI - LA TRADUCTION AUTOMATIQUE - 2009-2010

[17] G.F. Luger and W.A. Stubblefield. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addition-wesley, 1999.

[18] *Albéric Martel - Romain Bouleis* Université de Savoie. MPI-2 les systèmes experts - 2007

[19] Introduction aux objets - ISIMA ECOLE DOCTORALE SCIENCES FONDAMENTALES –Université de Genève- 1999

[20] Application des réseaux de neurones et des systèmes experts Ecole de théchnologie supérieur – Université du Quebec – 2011

[21] PASCAL ROQUES et FRANCK VALLÉE, UML2 en action, De l'analyse des besoins à la conception, 4ème édition. 2007

[22] Pierre-Jean Bellavoine, Delphi 7 de Pierre-Jean Bellavoine Édition Dunod, 1 ère édition, 1er mai 2003 ISBN10 : 2100081098

[23] Joseph Gabay et David Gabay, UML Analyse et conception Dunod, Paris, 2008
ISBN 978-2-10-053567-5

V

Annexes

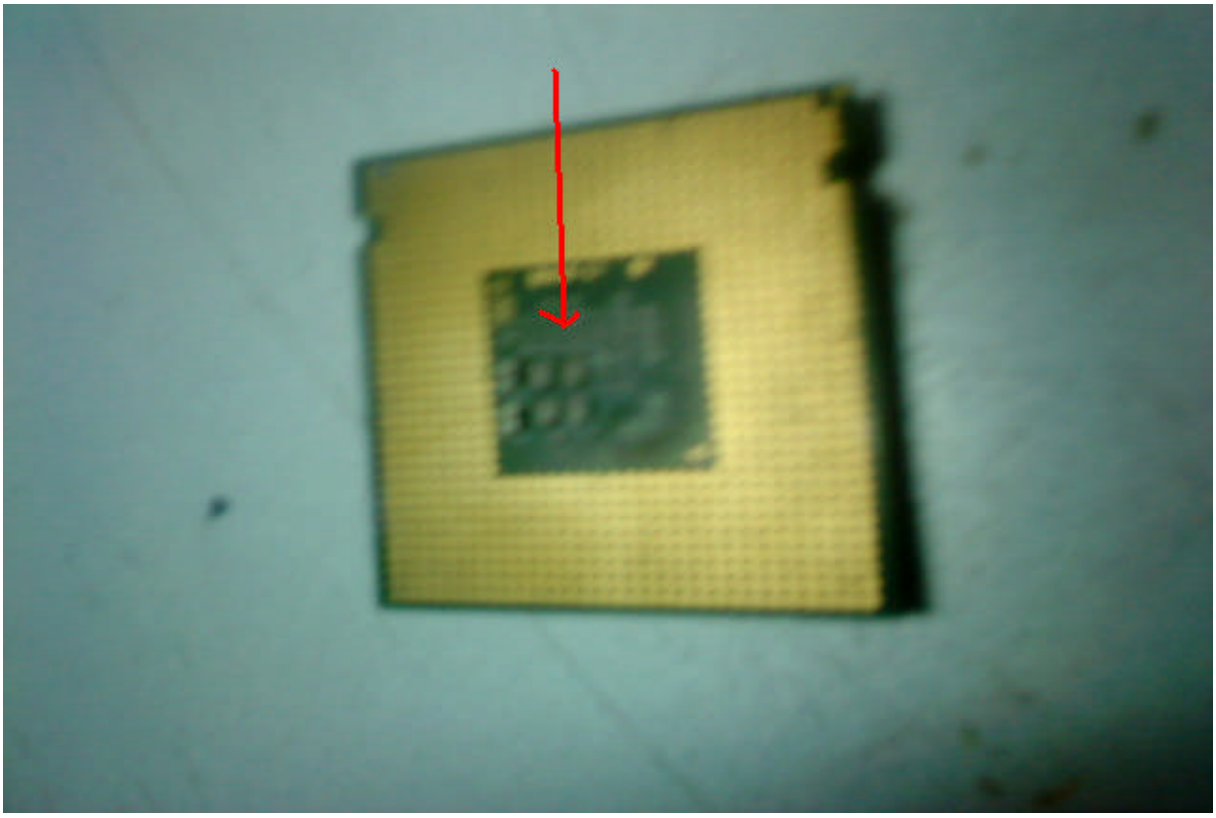


Figure 45. Micro processeur ne fonctionnant pas à d'une cause défaillance de registres CPU engendrant un problème dans l'ordinateur (Stage)

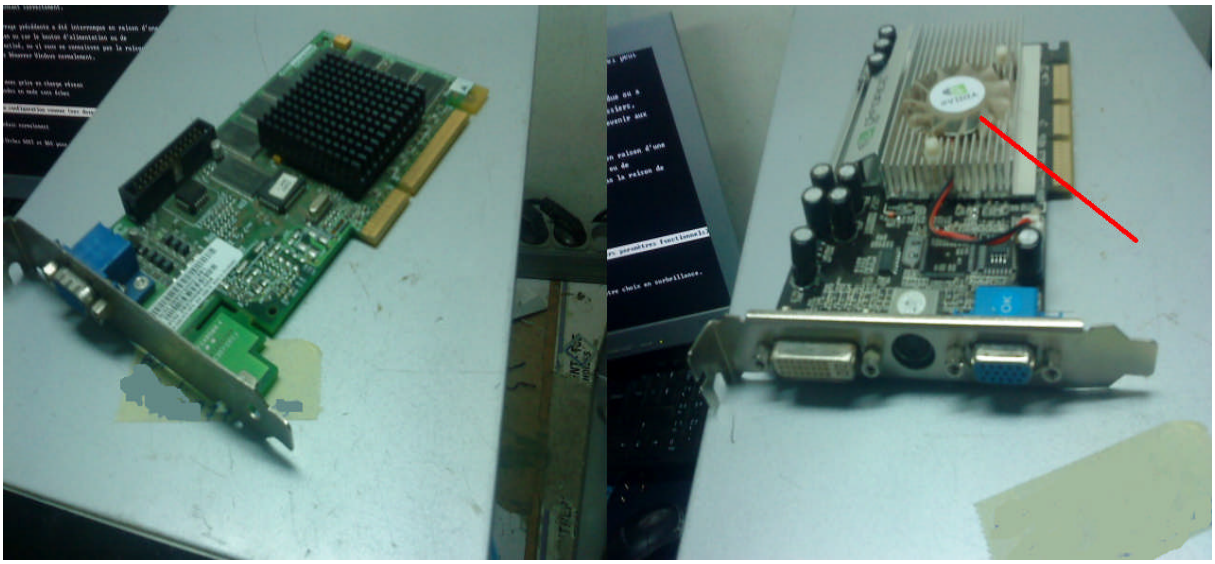


Figure 46. Deux modèles de cartes graphiques en panne, l'une avec Radiateur (à gauche) et l'autre avec Ventirad (à droite), celle de droite affiche un symptôme qui est le ventilateur qui est poussiéreux et qui ne marche plus (Stage).

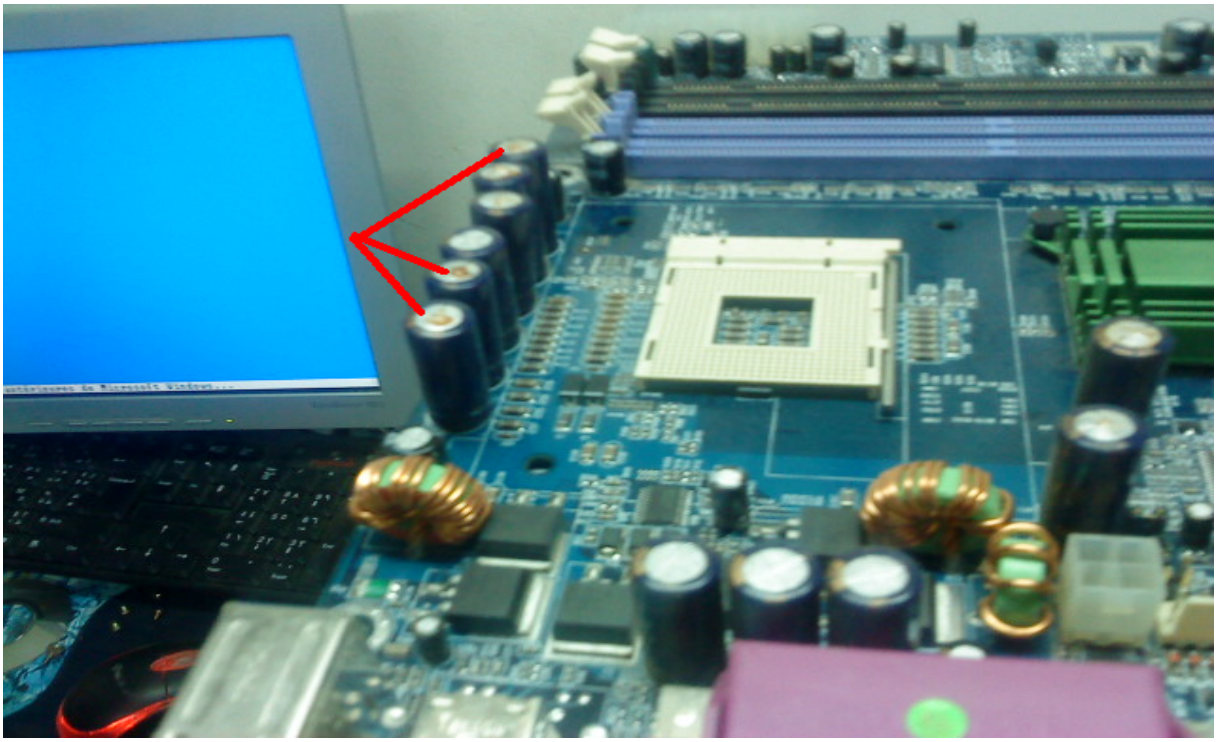


Figure 47. Disfonctionnement de la carte mère à cause de condensateurs qui ont gonflés et ne fonctionnent plus (Stage).

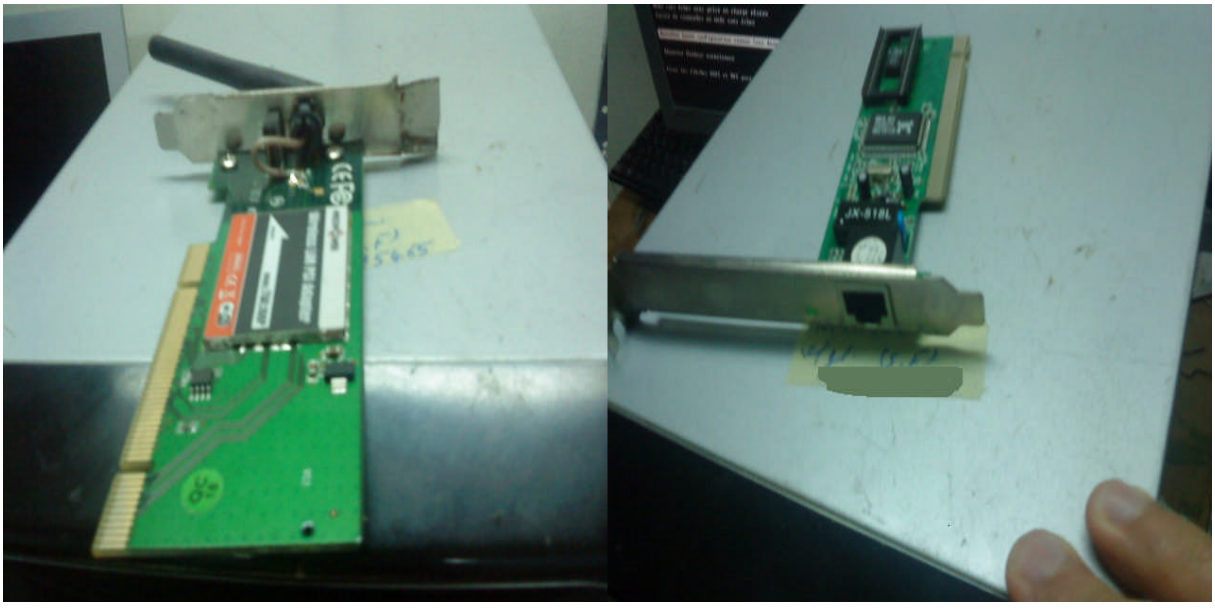


Figure 48. Cartes réseaux Wifi (à gauche) et Ethernet (à droite)qui ne fonctionnent pas, mais le symptôme n'est pas visible depuis les cartes (Stage).

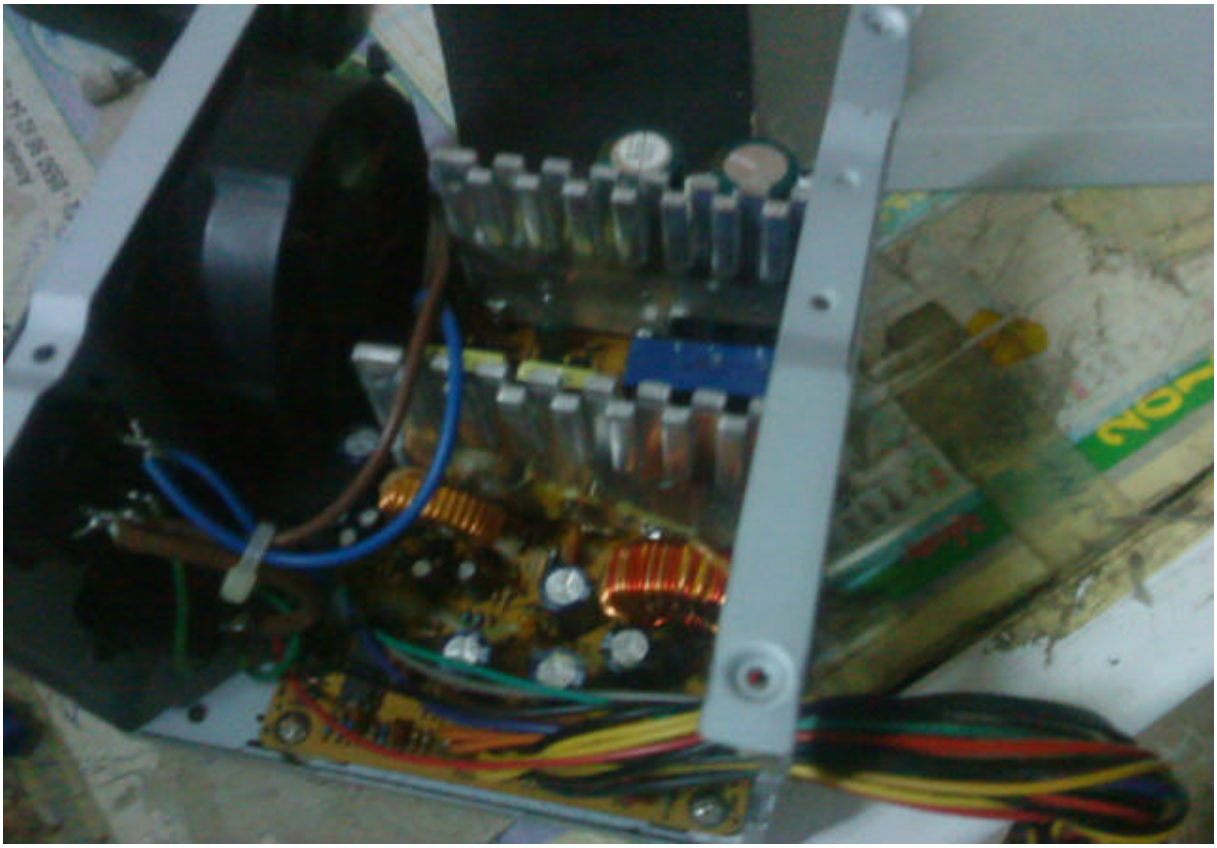


Figure 49. Boitier d'alimentation qui ne fonctionne plus pour cause de surchauffe, le symptôme est l'odeur qu'il dégage (Stage).

Figure 50. Disque dur qui ne fonctionne pas mais le symptôme n'est pas visible de l'extérieur (Stage).

Figure 51. Connecteur de clavier PS2 (à gauche) ; prise VGA à droite, les symptômes sont clairement visibles, celui de gauche à une dent cassée, et celui de droite a des dents déformées (Stage).

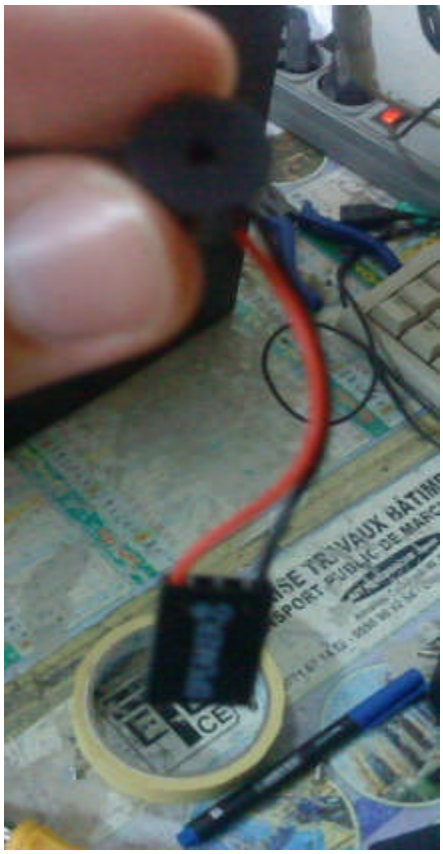


Figure 52. « Beeper », c'est grâce à celui-ci qu'on distingue quelques types de pannes matérielles (Stage).

Figure 53. Souris qui a soudainement cessée de fonctionner. Symptôme non visibles (Stage).

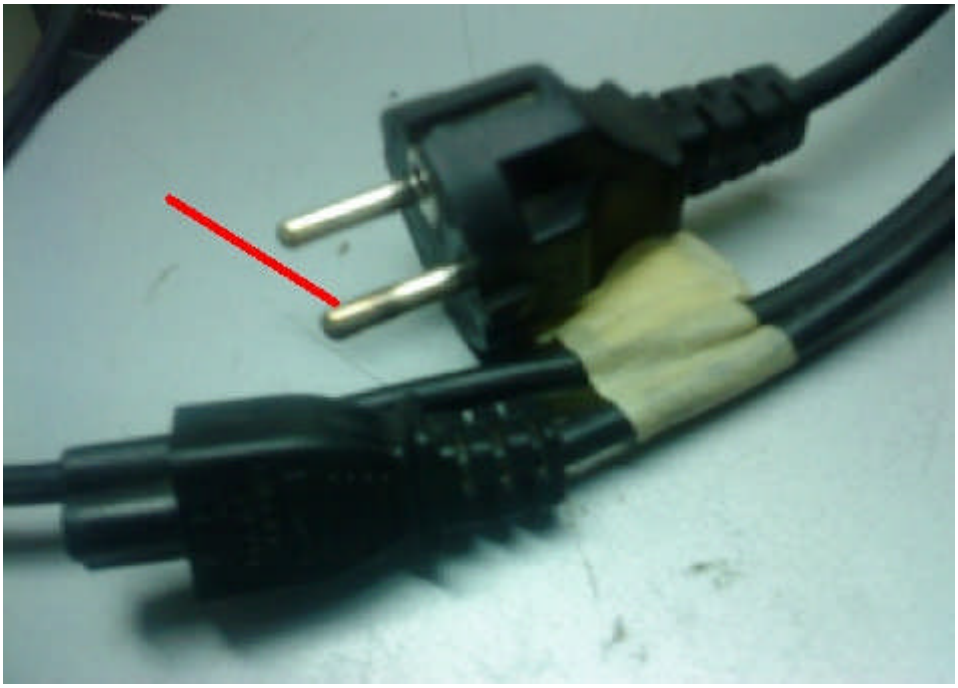


Figure 54. Prise de courant qui ne fonctionne plus, Symptôme visible de l'extérieur (Stage).

Figure 55.Généralement les RAM n'ont pas de symptôme visible de l'extérieur, à part celui du connecteur qui est déformé, mais sa reste rare (Stage).

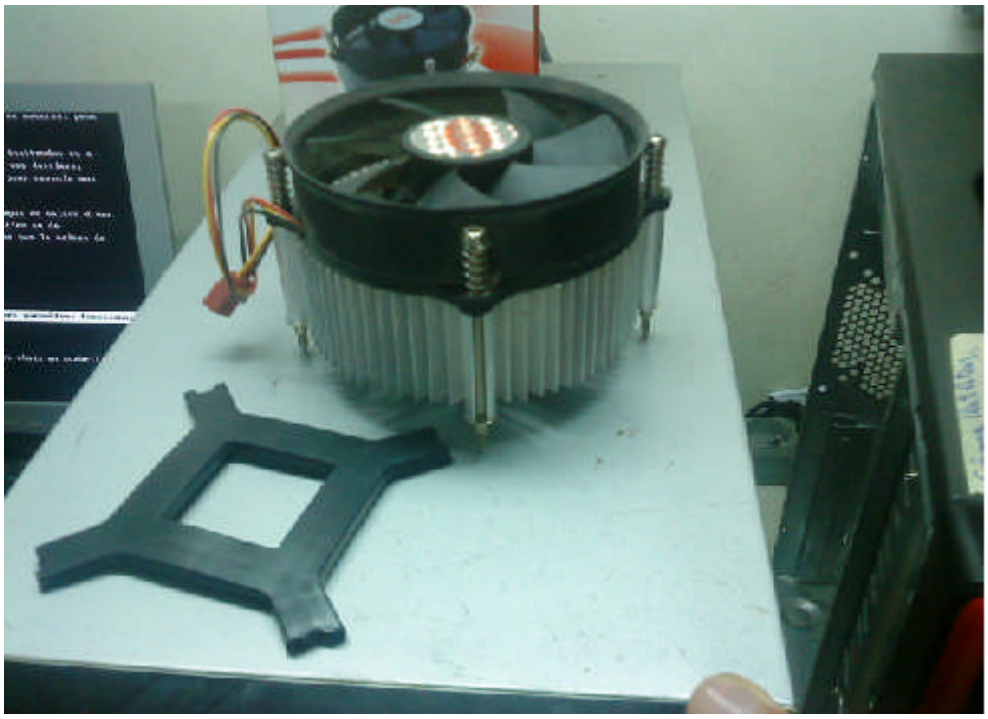


Figure 56. Ventilateur qui a cessé de fonctionner à cause d'une surchauffe (Stage).

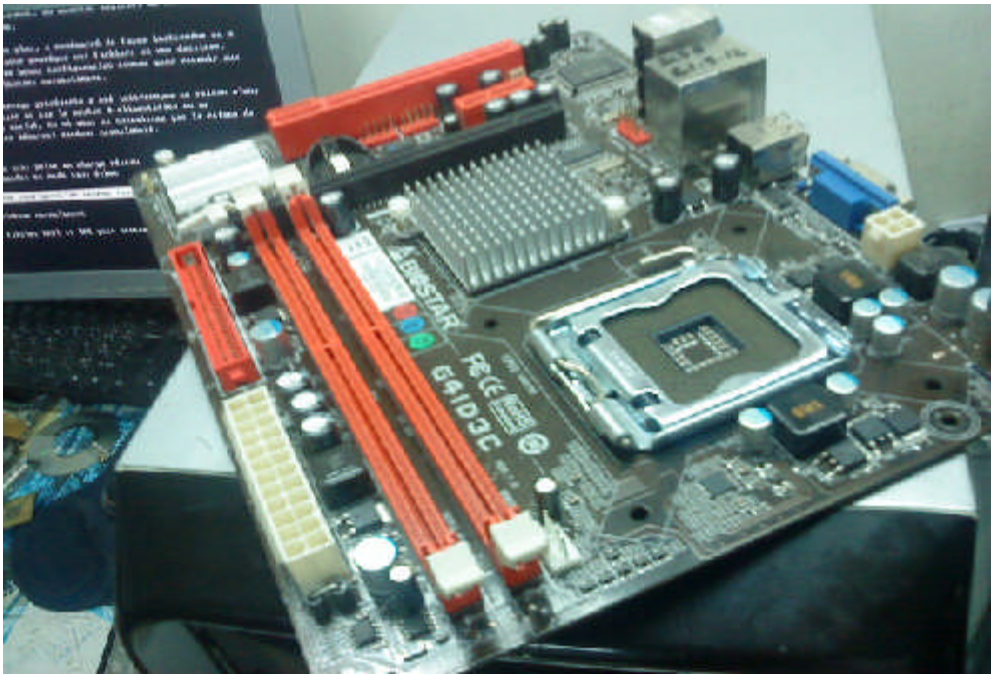


Figure 57. Carte mère qui semble intacte, mais qui ne fonctionne pas, les symptômes sont ceux qu'affiche l'ordinateur (Stage).

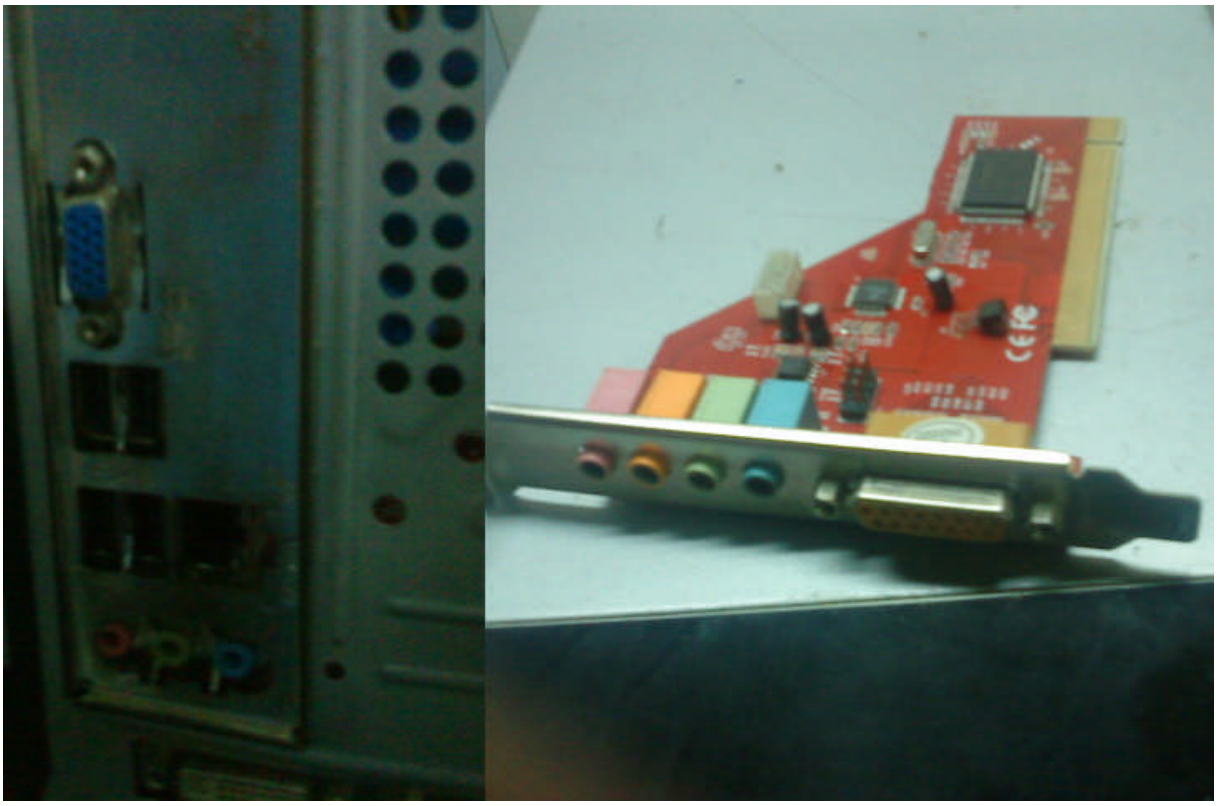


Figure 58 .Les deux composants ne sont pas de même nature (carte mère à gauche et carte son à droite) mais tout deux ont un problème de connecteur (interface) qui est une panne externe mais les symptômes ne sont pas visibles (Stage).

Figure 59. Vu de loin, ce lecteur/ graveur DVD n'a aucune problème, car il n'affiche aucun symptôme vu de l'extérieur, mais il est bien en panne (Stage).

Figure 60. Ecran Bleu

Résumé

De nos jours, l'informatique a pris une place énorme au sein des organismes, entreprises et particuliers, à tel point qu'on trouve son outil (Micro-ordinateur) partout où on va, à tel point qu'il devient inconcevable de ne pas en posséder un, car dans le monde dans lequel nous vivons, gagner du temps est crucial et synonyme de réussite afin d'aller plus loin et de se fixer des objectifs plus élevés pour percer dans le métier et le domaine dans lequel on exerce. Cependant, cet outil est sujet à des pannes matérielles et à des disfonctionnements, tout ceci perturbe le bon fonctionnement des tâches à faire. Alors il doit être maintenu dans les plus brefs délais pour renouer avec le travail, le plus rapidement possible. Toutefois, avant de réparer un ordinateur on doit savoir quelle est l'origine de sa panne. C'est ce qui est le plus pénible !

L'informatique en général et l'intelligence artificielle ont donnés naissance à un domaine qui est les systèmes experts qui donnent la possibilité à travers des programmes informatiques intelligents de connaitre l'origine des pannes, qu'elles soient de natures logicielles ou matérielles grâce un diagnostic qui est plus précis en faisant moins d'erreurs qu'un être humain, tout ceci dans un temps record. C'est pour cela que nous nous sommes penchés sur le diagnostic de pannes matérielles d'un Ordinateur en s'appuyant sur UML comme formalisme de conception soutenu par un processus très souple et rapide en deux axes qui est le 2 TUP. Pour le développement nous avons choisi Delphi 7 qui offre un bon espace de développement avec un langage facile à manipuler qui est le pascal objet, et pour les bases de données, Paradox 7 qui est un SGBD très facile à manipuler et qui offre beaucoup de fonctionnalités. Tout ceci afin d'aboutir à la réalisation d'un logiciel que nous avons nommé COMBREDES.

Mot clés :

INTELLIGENCE ARTIFICIELLE, SYSTEME EXPERT, UML ,2TUP, DELPHI 7, PASCAL OBJET, BASE DE DONNEES, PARADOX 7.

Abstract

Nowadays, computers have now become a huge part in organizations, businesses and private to the point of finding its tool (PC) wherever we go, to the point that it becomes inconceivable not to owning one, because in the world we live in, gain time is critical and means success to go further and to set higher goals to break into the business and the area in which it operates. However, the tool is subject to hardware failures and malfunctions. So it must be held to resume work as soon as possible. However, before repairing a computer we have to know what the cause of his failure is. That is what is most painful!

The general computing and artificial intelligence have given birth to a field that is the experts systems which give the opportunity through intelligent computing programs to know the origin of failures in diagnosis, whether plain or software systems material more accurately and with less error than a human being and in record time. This is why we focused on diagnosing hardware failures of a computer, based on UML formalism as design supported by a very flexible and fast in two axes which is 2 TUP process. For development we've chosen Delphi 7 which offers a good developing environment with an easy handling, and the object Pascal an easy programming language and Data bases manager system which is Paradox 7 very easy to use and offers many DBMS features language. All this lead to the creation of software, that we named COMBREDES.

Key words:

INTELLIGENCE ARTIFICIELLE, SYSTEME EXPERT, UML ,2TUP, DELPHI 7, PASCAL OBJET, BASE DE DONNEES, PARADOX 7.