Université de Nancy I — U.E.R. Sciences Mathématiques

Centre de Recherche en Informatique de Nancy

Méthodes, Langages et Outils de Spécification et de Construction des Systèmes de Diagnostic Technique

DOCTORAT DE L'UNIVERSITÉ DE NANCY I
Spécialité Informatique

Béchir EL AYEB

SOUTENUE LE 17 avril 1989

DEVANT LA COMMISSION D'EXAMEN :

Président	ROGER MOHR, Professeur à *l'Institut National Polytechnique de Grenoble*
Rapporteurs	JEAN-PAUL HATON, Professeur à *l'Université de Nancy* I JEAN-LOUIS LAURIERE, Professeur à *l'Université de Paris* VI
Examinateurs	JEAN-CLAUDE BOUSSARD, Professeur à *l'Université de Nice* JEAN-PIERRE FINANCE, Professeur à *Université de Nancy I* JEAN LIESCH, Chef de Service à *Arebed-RECHERCHES*

Université de Nancy I — U.E.R. Sciences Mathématiques

Centre de Recherche en Informatique de Nancy

Méthodes, Langages et Outils de Spécification et de Construction des Systèmes de Diagnostic Technique

DOCTORAT DE L'UNIVERSITÉ DE NANCY I
Spécialité Informatique

Béchir EL AYEB

SOUTENUE LE 17 avril 1989

DEVANT LA COMMISSION D'EXAMEN :

Président	ROGER MOHR, Professeur à *l'Institut National Polytechnique de Grenoble*
Rapporteurs	JEAN-PAUL HATON, Professeur à *l'Université de Nancy* I JEAN-LOUIS LAURIERE, Professeur à *l'Université de Paris* VI
Examinateurs	JEAN-CLAUDE BOUSSARD, Professeur à *l'Université de Nice* JEAN-PIERRE FINANCE, Professeur à *Université de Nancy I* JEAN LIESCH, Chef de Service à *Arebed-RECHERCHES*

Résumé

Si de nombreux travaux sont ménés dans le domaine des techniques de diagnostic, il n'en est pas de même dans le domaine de la méthodologie de diagnostic. L'objectif principal poursuivi par cette thèse est de spécifier et de construire **méthodiquement** les systèmes de diagnostic pour des grandes installations industrielles. La contribution et l'originalité de cette thèse se situent tant au niveau méthodologique que celui de la recherche d'un **langage** de **spécification** et des **outils** de construction des systèmes de diagnostic. Schématiquement, notre contribution consiste à proposer une méthode de spécification pour structurer une installation en hiérarchies et graphes de composants abstraits ainsi qu'un langage pour spécifier les différents composants. Nous définissons ensuite une méthode de diagnostic, qui fait **coopérer** un raisonnement de surface et un raisonnement profond. Nous proposons enfin les outils nécessaires pour construire le système de diagnostic.

mots-clés: Diagnostic de pannes, Génie logiciel, Installation industrielle, Intelligence artificielle, Langage de spécification, Méthodologie, Raisonnement profond, Raisonnement de surface.

Abstract

Our work is related to the building of fault Diagnosis Systems (DSs for short) for large industrial plants. Rather than presenting a particular Diagnostic System, we shall focus on the less studied step up to now: the methodology aspect. In fact, our main goal is to specify and construct **systematically** DSs for large industrial plants. For this end, we provide the user with a method for structuring the plant into hierarchies of abstract components. So, the set of hierarchies allows the user to bring to light more than a view of those components (e.g logical, physical, electrical, etc). We also define a *uniform* **specification language** for describing those components. Afterwards, we propose a *skillful* **cooperation** between deep and shallow reasonings to progressively locate the faulty components in hierarchies. Finally we define the set of **tools** to construct DSs for large industrial plants. The proposed methodology has been experimented on real-world applications.

key-words: Artificial Intelligence, Deep Reasoning, Large Industrial Plant, Methodology, Shallow Reasoning, Software Engineering, Specification Language, Troubleshooting.

Par le biais de ces quelques lignes, je souhaiterais exprimer ma gratitude à:

JEAN-PIERRE FINANCE, Professeur à l'Université de Nancy I et Directeur du CRIN, qui m'a donné l'opportunité de faire ce travail et m'a accueilli dans son équipe. Il m'a temoigné d'une grande confiance, y compris dans les moments difficiles, et a su orienté ce travail avec justesse. Aujourd'hui je lui dis: merci ...

JEAN-CLAUDE BOUSSARD, Professeur à l'Université de Nice, qui a examiné avec attention ce travail. Le grand intêrét qu'il y a porté, notamment sur l'aspect méthodologique, a été pour moi très stimulant. Je le remercie sincèrement pour l'enthousiasme avec lequel il m'a fait part de ses remarques.

JEAN-PAUL HATON, Professeur à l'Université de Nancy I, et JEAN-LOUIS LAURIERE, Professeur à l'Université de Paris VI, qui ont consacré beaucoup de leur temps pour être les rapporteurs de cette thèse. Je les remercie vivement parce que leurs points de vue éclairés ainsi que leurs jugements respectifs m'ont permis d'approfondir l'aspect Intelligence Artificielle de mon travail.

ROGER MOHR, Professeur à l'Institut National Polytechnique de Grenoble, qui a volontiers accepté de juger cette thèse. Je lui adresse mes sincères remerciements pour avoir présidé ce jury.

JEAN LIESCH, Chef de Service à AREBED-RECHERCHES, qui a permis la validation de ce travail dans le milieu industriel. Je remercie particulièrement JANNIK HAGEN, Ingénieur de Recherche, parce qu'il a été mon interlocuteur privilégié. C'est lui qui m'a permis de saisir le point de vue de l'industriel.

Je saisis cette occasion pour exprimer mes sincères remerciements à tous les membres de l'Institut d'Informatique de Namur et particulièrement au Professeur AXEL VAN LAMSWEERDE qui a été à l'origine de mes études de troisième cycle.

Si une thèse est essentiellement un travail personnel, son aboutissement n'est possible que grâce à l'environnement scientifique et humain. Je ne peux donc oublier de remercier tous les membres du CRIN, et spécialement MONIQUE DE SILVESTRI pour les remarques issues de sa lecture passionnée, DIDIER GALMICHE qui a accepté volontiers de lire cette thèse et m'a encouragé, ALAIN LESDALONS qui a consacré beaucoup de son temps pour lire cette thèse sans laisser échapper une seule erreur typographique ! Merci à CHRISTINE FERRARIS qui a corrigé également l'accentuation, que je contrôle toujours mal !, des toutes dernières parties.

J'adresse mes sincères remerciements à MARIE-CHRISTINE HATON et à DOMINIQUE MERY avec qui j'ai eu maintes fois des discussions très enrichissantes.

Merci aussi à ABDERRAFIAA KOUKAM, PIERRE MOUSEL et DJEMEL ZIOU qui n'ont jamais cessé de me soutenir et de m'encourager à tout moment: c'est très stimulant. Enfin à TOUS merci.

Quant à mes parents, ma famille et mes proches, je ne dirai rien, ils savent la suite.

Fait à Nancy le 1er mai 1989,

Béchir El. AYEB

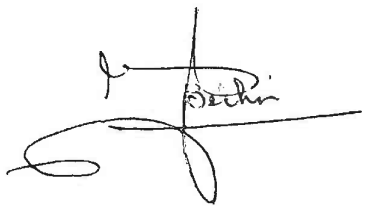


Table des matières

CHAPITRE 2

CHAPITRE 3

CHAPITRE 4

Prologue

Méthodes, langages et outils constituent les trois mots-clefs qui gouvernent cette thèse. Mais avant d'aborder le corps de ce travail, il convient de préciser à quelle classe de problèmes nous nous adressons, d'analyser et de déterminer ensuite nos objectifs, de donner l'approche globale de ce travail, d'illuster de manière informelle nos propositions sur un exemple avant de donner le plan de cette thèse.

1 La maintenance des grandes installations industrielles

Pourquoi le diagnostic ?

La recherche d'un accroissement de la productivité et l'effet de la concurrence conduisent les industriels à se soucier des problèmes posés par la maintenance des équipements de production [Gabriel 87]. Ces problèmes sont particulièrement délicats lorsqu'ils touchent les industries lourdes telles que les grandes installations sidérurgiques.

Les objectifs visés par la maintenance d'une grande installation sont:

(1) l'accroissement de la productivité par une disponibilité et une fiabilité accrues de l'outil de production,

(2) l'augmentation de la rentabilité par la réduction des frais d'entretien,

(3) l'amélioration de la qualité des produits.

De quoi s'agit-il ?

Pour atteindre ces objectifs, les responsables de l'installation ont recours à différentes techniques de diagnostic de pannes. Intuitivement, "diagnostiquer les pannes d'une grande installation", c'est

> *à partir d'une anomalie de fonctionnement qui induit un ensemble de symptômes, déterminer les composants en panne responsables de l'anomalie.*

Des systèmes de diagnostic sans méthode

Les relations entre les pannes et les symptômes ne sont ni simples à exprimer, ni biunivoques et compliquent ainsi le problème du diagnostic. Dès lors, une assistance au

dépannage devient vite nécessaire. Et, si, à ce jour, on peut dénombrer des travaux de réalisation de systèmes de diagnostic de pannes [Jakob 84], dans la plupart des cas ils ont malheureusement été conçus de façon quelque peu "anarchique": *sans méthode(s).* La rigidité, la difficulté d'évolutivité et de transportabilité constituent alors des propriétés inhérentes à ces systèmes construits ainsi sur-mesure (i.e ad-hoc) et sans méthodes.

2 Objectifs et préoccupations de ce travail

une méthode ...

Notre objectif principal n'est pas de mettre en place un système particulier de diagnostic pour une installation donnée, mais, au contraire, il est

de proposer une méthode de spécification et de construction de tels systèmes.

Les caractéristiques des grandes installations justifient pleinement la nécessité d'utiliser une méthode, en particulier:

- l'ampleur des installations,
- la diversité et la complexité des composants formant l'installation. On y trouve des composants du type électrique, mécanique, hydraulique, etc ...
- l'évolution continuelle de l'installation,
- les arrêts réduits au minimum. La plupart du temps, les installations fonctionnent en continu, et les interventions de dépannage se déroulent sur une installation en service.

des outils ...

S'il est indispensable de disposer d'une méthode de spécification et de construction des systèmes de diagnostic, il est aussi important de proposer un langage de spécification ainsi que les outils nécessaires à une bonne conduite et mise en oeuvre de la méthode proposée. Il est également nécessaire de l'expérimenter sur des cas réels pour fonder son applicabilité et sa réutilisabilité.

3 Premières analyses et détermination des sous-objectifs

Donnons une première spécification informelle du problème du diagnostic:

Soient $\mathcal{I}$ une installation et $\mathcal{S}$ la spécification du comportement normal de $\mathcal{I}$. Il s'agit alors de: (1) détecter les divergences entre le comportement de l'installation $\mathcal{I}$ et le comportement spécifié $\mathcal{S}$. (2) déterminer les composants ayant causé les divergences détectées.

Ce premier énoncé du problème fait ressortir les concepts suivants:

- "spécifier une installation"
- "détecter les divergences"
- "déterminer les composants en panne"

Si les deux derniers concepts relatifs aux techniques de diagnostic ont fait l'objet de nombreuses études [Alanche 86], [Ayache 80], [Bogler 87], [Davis 84], [DeKleer 87], [Descotes 84], [Genesereth 84], [Marin 76], [Reiter 87] il n'en est pas de même pour le premier concept à savoir spécifier une installation en vue de mettre en place un système de diagnostic de pannes. L'analyse de l'ensemble de ces concepts est à la base de notre démarche de travail et a déterminé la structuration de cette thèse.

3.1 Vers une méthode de spécification

A cette étape, il s'agit de répondre aux questions suivantes:

- quelle démarche adopter pour spécifier l'installation qui apparaît au départ comme un objet complexe à appréhender ?
- que spécifier ?
- comment exprimer la spécification ?

Répondre à ces questions revient à proposer une méthode de spécification qui consiste à:

- fournir un guide, sous forme de règles et d'étapes à suivre pour structurer l'installation en une *hiérarchie* d'objets abstraits. Nous appellerons ces objets des ***composants abstraits*** dans le sens où ils ne correspondent pas nécessairement à des composants réels de l'installation. Cette hiérarchie de composants abstraits donnera ainsi une "représentation logique" de l'installation.
- donner un ensemble de relations pour construire des *graphes d'adjacence* des composants de l'installation. Selon la relation utilisée, chaque graphe donnera une vue particulière (i.e électrique, physique, etc) de l'installation.
- proposer un langage de spécification[1] pour pouvoir exprimer la structure, le comportement, les lois et l'interface de chacun des composants. A ce niveau, il ne s'agit pas de proposer "le" langage "idéal" ou "universel", au contraire nous préconisons un langage simple mais puissant qui pourra être enrichi par des opérateurs spécifiques en fonction du cas traité.

[1] notre langage peut être vu comme un langage de "spécification" mais aussi de programmation dans le sens ou les spécifications peuvent être interprétées par Lisp. De ce fait, le terme spécification est alors utilisé dans un sens plutôt restreint car notre langage constitue, en fait, une couche au dessus du langage Lisp.

De cette manière, la méthode nous conduit tout d'abord à une hiérarchie dite *principale* formée essentiellement des composants abstraits, ensuite à des graphes d'adjacence, et enfin à des spécifications dites *partielles* exprimant la structure, le comportement, les lois et l'interface des composants.

Mais, ces spécifications se limitent à la description de l'installation et n'indiquent pas comment conduire un diagnostic pour déterminer les pannes de l'installation. D'où l'idée d'étudier, dans un premier temps, les techniques de diagnostic afin d'obtenir une synthèse globale. De proposer, par la suite, une méthode de diagnostic et de compléter les spécifications partielles en spécifications *complètes* incluant les connaissances nécessaires pour conduire un diagnostic.

3.2 Synthèse des techniques de diagnostic

Au risque de la caricaturer, l'activité de diagnostic d'une installation peut se résumer à:

- détecter les anomalies de fonctionnement;
- localiser les composants responsables des anomalies.

Si d'un point de vue macroscopique cette activité semble facile, il n'en est pas de même du point de vue microscopique. En effet, l'introduction des technologies dites de pointe et la diversification des équipements des installations rendent cette activité difficile. Face à cette difficulté, de nombreuses études ont été menées pour aboutir à différentes techniques de diagnostic [Alanche 86], [Ayache 80], [Bogler 87], [Davis 84], [DeKleer 87], [Descotes 84], [Genesereth 84], [Marin 76], [Reiter 87]. Notre première tâche va consister à faire une analyse de ces techniques de diagnostic.

De cette analyse, nous dégageons les mécanismes de base et négligeons un certain nombre de notions secondaires et conjoncturelles. Ceci nous conduit à partitionner les techniques de diagnostic en quatre grandes classes:

- les techniques intégrées basées sur l'analyse combinatoire. Parmi ces techniques, on trouve la technique dite du filtre [Alanche 86] qui consiste à placer un programme pour contrôler les ordres donnés par la partie de commande et les retours de la partie opérative.

- les techniques empiriques basées sur des relations "de cause à effet" et qui utilisent un raisonnement dit de *surface*.

- les techniques "par modèle" basées sur des algorithmes de diagnostic et qui utilisent un raisonnement dit *profond*.

- les techniques issues de la physique qualitative. Parmi ces techniques, on trouve celle de De Kleer [DeKleer 87] dont le but est de modéliser un phénomène physique et de

manipuler par la suite les ordres de grandeurs physiques qui les caractérisent.

Pour chaque classe, nous donnerons son principe de base, ses avantages et ses limites. Nous proposerons aussi des solutions aux restrictions exposées.

3.3 Elaboration du diagnostic

Partant de la synthèse effectuée mais aussi des caractéristiques des installations considérées, il s'agit maintenant de proposer notre méthode de diagnostic qui consiste à fournir:

- des *stratégies* et des *tactiques*. Intuitivement, les stratégies correspondent à des heuristiques d'exploration de la hiérarchie pour localiser progressivement la plus petite sous-hiérarchie de composants responsables des anomalies de fonctionnement observées. Quant aux tactiques, elles correspondent à des algorithmes qui servent à déterminer, parmi les composants de la sous-hiérarchie identifiée, ceux qui sont en panne.

- un *mécanisme de diagnostic* paramétré par les stratégies et les tactiques qui fait *coopérer* deux types de raisonnement souvent considérés comme "opposés":

  - Tout d'abord le raisonnement de surface qui utilise les stratégies pour localiser progressivement les sous-hiérarchies.
  - Ensuite le raisonnement profond pour déterminer les composants en panne.

  Nous montrerons par la suite les avantages d'une telle *coopération*. Pour diverses raisons telles que l'inexactitude des mesures fournies par un capteur ou encore la non vérification des préconditions de la tactique utilisée, le raisonnement profond peut engendrer des résultats incorrects. Nous sommes alors conduits à proposer:

- un *mécanisme de justification* qui, basé sur un raisonnement empirique, utilise des règles pour "valider" le résultat du raisonnement profond. Nous appellerons *paradigmes* ces règles fournies par les experts en dépannage.

Il s'agit ensuite de montrer l'utilisation des stratégies, des tactiques et des paradigmes et de spécifier le diagnostic, à conduire, au niveau de chaque composant de la hiérarchie principale. Ceci nous permet d'obtenir des spécifications complètes des composants qui incluent la structure, le comportement, les lois, l'interface ainsi que les connaissances nécessaires à conduire le diagnostic exprimées en termes de stratégies, de tactiques ou de paradigmes.

Nous n'avons pas l'intention d'affirmer que la méthode de diagnostic présentée ici est la plus idéale ou la plus universelle. Aussi la paramétrons-nous par les stratégies, les tactiques et les paradigmes pour augmenter sa flexibilité et sa puissance. C'est pourquoi

nous regroupons les stratégies, les tactiques et les paradigmes en *catalogues* de manière à pouvoir les enrichir au fur et à mesure des cas présentés ici.

4 Expérimentation

Au cours de l'étape initiale de ce travail, nous avons passé 6 mois au sein d'une société sidérurgique pour obtenir une première spécification du problème de diagnostic des grandes installations. Après des études de cas, nous avons élaboré une première version de notre méthode de spécification et de construction des systèmes de diagnostic. Dans le but de valider la méthode nous l'avons expérimentée sur une installation de mesure de longueurs de poutrelles chaudes. Ensuite, nous avons réalisé le prototype[2] *SIDI*[3]. Ce premier prototype, qui compte environ 10.000 lignes LISP, nous a permis de confronter ce travail à des cas réels comme par exemple l'installation de laminage. De plus, un projet portant sur des machines vibrantes est actuellement à l'étude. Ce projet devra aboutir sur le développement d'un second prototype industriel.

C'est ainsi que la plupart des exemples d'illustration que nous donnerons seront issus du domaine sidérurgique. Toutefois, pour des raisons de confidentialité mais aussi pour alléger et clarifier le texte, nous serons conduit à simplifier largement les exemples traités.

5 Choix d'une approche

D'après ce qui précède, notre approche peut être résumée comme suit: (1) identification d'une classe de problèmes, (2) analyse d'un nombre significatif de problèmes de cette classe et formalisation du processus de diagnostic, (3) construction d'un *modèle*[4] adapté à cette classe et proposition d'un guide pour utiliser et exploiter ce *modèle*: c'est-à-dire une méthode.

Il apparaît ainsi que cette approche privilégie une classe de problèmes pour lui construire un *modèle* approprié. C'est pourquoi nous la qualifions de constructive par opposition à l'approche expérimentale qui consiste à modéliser, analyser et structurer les connaissances d'un problème donné suivant un *modèle* choisi. Dans de nombreux cas, on est conduit à adapter et éventuellement enrichir le *modèle* initial avant de pouvoir modéliser le problème donné.

Actuellement, plusieurs *modèles* ont été proposés. Citons le *modèle* du blackboard qui a été utilisé dans de nombreuses applications [Laasri 89], ou encore celui des sociétés d'experts [Haton 89]. Ce dernier a été utilisé entre autres pour l'interprétation automatique des signaux [Gong 88]. Signalons enfin les travaux d'enrichissement du *modèle* du blackboard par l'adjonction d'un raisonnement hypothétique et temporel [Laasri 88].

adapter, valider et enrichir ou construire ?

---

[2]Réalisé pour la société ARBED-RECHERCHES qui a financé en partie ce travail.

[3]Acronyme de *S*ystème *I*nteractif de *D*iagnostic *I*ndustriel.

[4]Ce terme doit être considéré dans un sens large e.g une architecture, une organisation des connaissances, etc. C'est pourquoi nous l'écrirons en italique pour le distinguer du terme modèle au sens strictement mathématique.

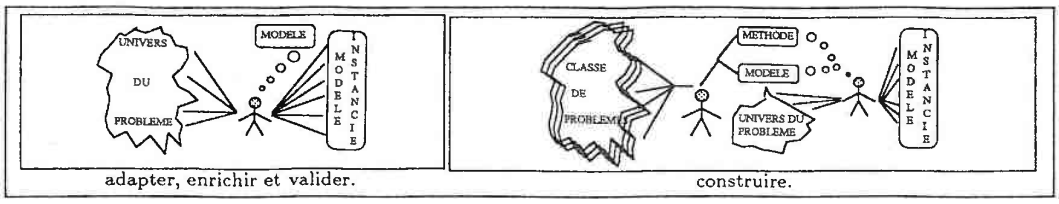


Figure 0.1: quelle approche choisir ?

A notre avis les deux approches sont complémentaires et de ce fait nécessaires. En effet l'approche expérimentale vise à valider et adapter des *modèles* préexistants à d'autres types de problèmes. A son tour l'approche constructive permet de proposer un *modèle* pour une classe donnée de problèmes. Ce dernier peut être adapté ulterieurement à d'autres classes de problèmes. Notons enfin que dans certains cas nous sommes conduits à utiliser conjointement les deux approches, c'est-à-dire expérimenter dans un premier temps un *modèle* déterminé pour analyser et comprendre le problème donné avant d'en construire un autre *modèle* plus approprié, ce dernier pouvant être hybride.

vers des recherches méthodologiques

Au délà du choix de l'approche, nous pensons que parallèlement aux travaux traitant les aspects fondamentaux tels ceux sur la classification [Chandrasekaran 83], et l'analyse des problèmes [Haton 86], [Lauriere 87], sur le mécanisme d'inférence [Hayes 83], [Blum 83a], la structure de contrôle [Laurent 84], la représentation des connaissances [Blum 83b], [Lauriere 82], [Reboth 83], [Weilinga 86], le type de raisonnement [Freiling 86], [Li 86], [Tong 83] ainsi qu'aux travaux de développement et de validation des *modèles* cités plus haut, il est important de conduire des recherches sur l'aspect méthodologique[5]. En effet, il nous semble que la mise en place d'une méthode aura comme conséquences non seulement de faciliter amplement l'activité de développement proprement dite, mais aussi de contribuer à résoudre des problèmes fondamentaux tels le choix du mécanisme d'inférence ou encore celui de la représentation de la connaissance. C'est ce que nous montrerons tout au long de cette thèse.

6 Sur un exemple

Avant d'aborder le corps de ce travail, illustrons de manière informelle nos propositions sur un exemple simple. A cet effet, considérons une partie d'une installation de laminage qui prend en charge la découpe des produits longs sortant des cages des laminoirs. Cette partie de l'installation que nous appellerons par la suite *sous-système* industriel est notre donnée de départ. Notre but maintenant est de présenter succinctement la démarche à suivre pour structurer et spécifier ce sous-système, de construire ensuite le système de diagnostic associé à ce sous-système industriel.

Schématiquement, ce sous-système se présente comme suit:

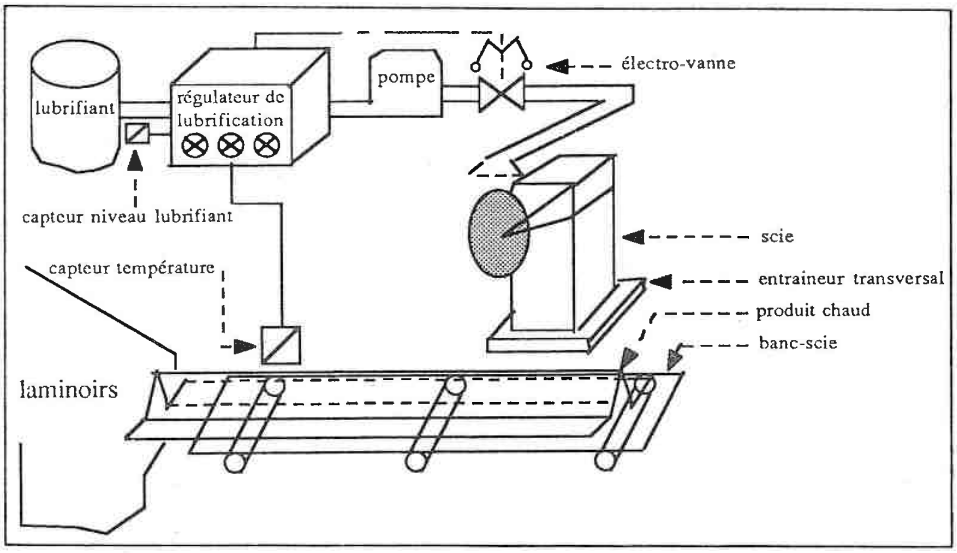


Figure 0.2: schéma du sous-système "cisaille".

Les produits chauds en provenance des cages de laminoirs sont entrainés jusqu'au dispositif de cisaille. La découpe des produits est assurée par une scie électrique qui peut se déplacer dans le sens longitudinal et transversal. De plus, on a un lubrificateur qui, en fonction de la température du produit détectée par un capteur, commande une électro-vanne pour réguler le débit de lubrification de la scie. En cas de haute température du produit ou de manque de lubrifiant, une alarme est déclenchée.

PREMIERE ETAPE : **Structuration hiérarchique**

Notre première tâche consiste à structurer ce sous-système en une hiérarchie de composants. A cet effet, on va procéder ici de manière descendante:

- considérer tout le sous-système comme un seul composant. Dans ce cas, on considère ce sous-système comme un composant qualifié d'*abstrait*: "cisaille".

- décomposer ce composant en d'autres composants abstraits de plus bas niveau. Cette décomposition est effectuée suivant la relation "utilise". Dans ce cas on a le composant abstrait "cisaille" qui utilise deux autres composants abstraits: "scie" et "lubrificateur". Intuitivement, cela signifie que les composants "scie" et "lubrificateur" interagissent pour produire le comportement global du composant "cisaille".

- ce processus de décomposition s'arrête dès que les composants de plus bas niveau (i.e les feuilles de la hiérarchie) ont une taille, jugée par les agents de maintenance,

"raisonnable". Par convention, nous appellerons ces composants, qui représentent les feuilles de la hiérarchie, des composants *corpusculaires*.

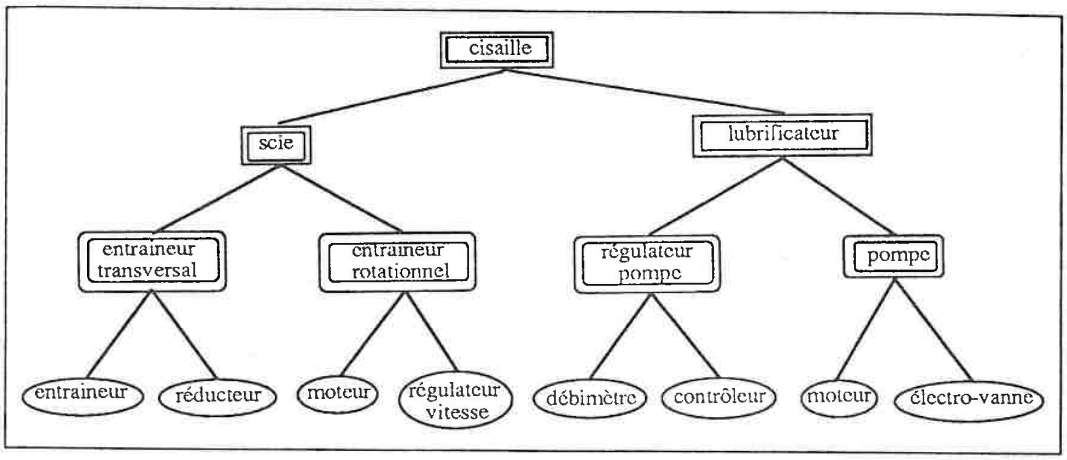


Figure 0.3: hiérarchie de composants du sous-système "cisaille".

DEUXIEME ETAPE : **Spécification des composants**

A cette étape, notre objectif est de spécifier les composants de la hiérarchie. La spécification d'un composant consiste à exprimer:

- sa structure en terme de variables d'état qui caractérisent le composant et des ports d'entrée et de sortie qui lui permettent de communiquer avec les autres composants.

- son comportement normal en décrivant les règles d'évolution des variables d'état. Pour cet exemple, nous décrivons le comportement des composants informellement, c'est-à-dire en langage naturel.

- les lois en terme de contraintes qui régissent son comportement. Nous exprimerons aussi ces lois en langage naturel.

- son interface, c'est-à-dire les règles d'acquisition des données de l'environnement extérieur (i.e les capteurs). Pour cet exemple, nous omettrons cette partie de la spécification.

Illustrons progressivement les différentes étapes de spécification sur la sous-hiérarchie du composant "lubrificateur" et commençons par spécifier la structure:

structure lubrificateur

les variables d'état

- état ∈ {marche,arrêt}
- pression ∈ {faible,moyenne,haute}
- débit ∈ {faible,moyen,haut}
- puissance ∈ {faible,moyenne,haute}

les ports en entrée

- p-mise-en-marche-arrêt % en provenance du composant abstrait "scie".
- p-température % en provenance du capteur détectant la température du produit.

Figure 0.4: spécification du composant abstrait "lubrificateur".

Intuitivement, pour spécifier la structure de ce composant (i.e "lubrificateur"), on est conduit à le considérer comme une boîte noire caractérisée d'une part par quatre variables d'état, son état (i.e en marche ou à l'arrêt), la pression, le débit de lubrification ainsi que la puissance, d'autre part par deux ports d'entrée, p-mise-en-marche-arrêt et p-température. Le premier port sert à recevoir des messages en provenance d'autres composants et le second sert à acquérir la température du produit chaud à découper.

comportement lubrificateur

- si le port "p-mise-en-M-A" indique la mise en marche
  alors mettre "état" à "marche".
- si le port "p-mise-en-M-A" indique la mise en arrêt
  alors mettre "état" à "arrêt".
- si le port "p-température" indique une hausse de température du produit
  alors si la puissance est "haute" alors diminuer la pression sinon augmenter la puissance
- si le port "p-température" indique une baisse de température du produit
  alors si la puissance est "haute" alors augmenter la pression sinon diminuer la puissance

Figure 0.4 (suite): spécification du composant abstrait "lubrificateur".

A cette étape de spécification, le but poursuivi est d'appréhender et de comprendre le comportement de la boîte noire qui représente le composant, c'est-à-dire l'évolution des variables d'état. Partant de cette spécification comportementale, nous sommes conduits à décrire les lois qui gouvernent le comportement du composant considéré (i.e le lubrificateur). Cette étape intermédiaire de spécification du comportement constitue donc un outil conceptuel[5] d'aide à la compréhension et à l'identification des lois du composant.

lois lubrificateur

% on contraint l'évolution du débit:

(1) la croissance du débit est proportionnelle à la croissance de la puissance.

(2) la décroissance du débit est inversement proportionnelle à la croissance de la pression.

% on contraint l'évolution des mesures du capteur température

(3) la température reçue sur le port "p-température" doit être entre 300°C et 1000°C.

Figure 0.4 (suite): spécification du composant abstrait "lubrificateur".

[5]Comme nous le verrons, actuellement la spécification comportementale n'est pas exploitée par le mécanisme de diagnostic. Toutefois, la réalisation d'un outil d'interprétation des spécifications comportementales peut constituer une aide pour mettre au point, corriger et valider les spécifications des lois.

Notre but, à présent, est de raffiner ces spécifications en exprimant les spécifications des composants de plus bas niveau. Poursuivons la même démarche de spécification que nous avons entreprise pour le composant "lubrificateur" et commençons par la structure du composant "régulateur".

structure **régulateur**
les variables d'état
- alarme ∈ {déclenchée,non-déclenchée}
- pression ∈ {faible,moyenne,haute}

les ports en entrée
- p-niveau-lubrifiant % en provenance du capteur contrôlant le niveau du liquide lubrifiant.
- p-température % en provenance du capteur détectant la température du produit.

les ports de sortie
- p-arrêt-pompe % vers le composant pompe.
- p-puissance % vers le composant pompe.

Figure 0.5: spécification du composant terminal "régulateur".

Ensuite le comportement.

comportement **régulateur**
- si le port "p-niveau-lubrifiant" indique un niveau bas
  alors - mettre "alarme" en état "déclenchée",
  - signaler, par le port "p-arrêt-pompe", l'arrêt de la pompe.
- si le port "p-température" indique une hausse de température du produit
  alors si la pression est "faible"
  alors signaler, par le port p-puissance, une augmentation de la puissance.
- si le port "p-température" indique une baisse de température du produit alors si la pression est "haute"
  alors signaler, par le port "p-puissance", une diminution de la puissance.

Figure 0.5 (suite): spécification du composant terminal "régulateur".

Enfin les lois.

lois **régulateur**
% on contraint l'évolution des mesures du capteur niveau lubrifiant
(1) la mesure transmise par ce capteur doit être comprise entre 5 et 10 Volts.
% on contraint l'évolution des mesures du capteur température
(2) la température reçue sur le port "p-température" doit être entre 300°C et 1000°C

Figure 0.5 (suite): spécification du composant terminal "régulateur".

De manière similaire, nous spécifions les autres composants:

structure pompe
les variables d'état
- état ∈ {marche,arrêt}
- état-électro-vanne ∈ {faible,moyenne,grande}
- puissance ∈ {faible,moyenne,haute}
- débit ∈ {faible,moyen,haut}

les ports en entrée
- p-arrêt-pompe % en provenance du composant abstrait régulateur.
- p-puissance % en provenance du composant abstrait régulateur.

comportement
- si le port "p-arrêt-pompe" indique un arrêt de la pompe
  alors mettre "état" en position "arrêt".
- si le port "p-puissance" indique une augmentation
  alors si l'électro-vanne est dans la puissance "moyenne"
  alors mettre l'électro-vanne dans la position "grande".
  sinon faire croître la puissance.
- si le port "p-puissance" indique une diminution
  alors si l'électro-vanne est grande
  alors mettre l'électro-vanne dans la position "moyenne".
  sinon faire décroître la puissance.

lois
% on contraint l'état de l'électro-vanne.
(1) le débit est proportionnel à l'ouverture de l'électro-vanne.
% on contraint le débit.
(2) le débit est proportionnel à la puissance.

Figure 0.6: spécification du composant terminal "pompe".

structure débimètre
les variables d'état
- voyants ∈ {allumés, éteints}
- température ∈ {faible,moyenne,haute}
- débit ∈ {faible,moyen,haut}

les ports en entrée
- p-température % en provenance du capteur détectant la température du produit.

les ports de sortie
p-débit % vers le composant contrôleur.

comportement
- si le port "p-température" indique 1500°C
  alors - mettre voyants à "allumés".
- si le port "p-température" indique une baisse de température du produit
  alors signaler, par le port "p-débit", une diminution du débit.
- si le port "p-température" indique une hausse de température du produit
  alors - signaler, par le port p-débit, une augmentation du débit.

lois
% on contraint l'évolution des mesures du capteur température
(1) la température reçue sur le port "p-température" doit être entre 300°C et 1000°C
% on contraint l'évolution du débit en fonction de la température
(2) le débit est proportionnel à la température

Figure 0.7: spécification du composant corpusculaire "débimètre".

structure pompe
les variables d'état
– état ∈ {marche,arrêt}
– état-électro-vanne ∈ {faible,moyenne,grande}
– puissance ∈ {faible,moyenne,haute}
– débit ∈ {faible,moyen,haut}
les ports en entrée
– p-arrêt-pompe % en provenance du composant abstrait régulateur.
– p-puissance % en provenance du composant abstrait régulateur.
comportement
– si le port "p-arrêt-pompe" indique un arrêt de la pompe
alors mettre "état" en position "arrêt".
– si le port "p-puissance" indique une augmentation
alors si l'électro-vanne est dans la puissance "moyenne"
alors mettre l'électro-vanne dans la position "grande".
sinon faire croître la puissance.
– si le port "p-puissance" indique une diminution
alors si l'électro-vanne est grande
alors mettre l'électro-vanne dans la position "moyenne".
sinon faire décroître la puissance.
lois
% on contraint l'état de l'électro-vanne.
(1) le débit est proportionnel à l'ouverture de l'électro-vanne.
% on contraint le débit.
(2) le débit est proportionnel à la puissance.

Figure 0.6: spécification du composant terminal "pompe".

structure débimètre
les variables d'état
– voyants ∈ {allumés, éteints}
– température ∈ {faible,moyenne,haute}
– débit ∈ {faible,moyen,haut}
les ports en entrée
– p-température % en provenance du capteur détectant la température du produit.
les ports de sortie
p-débit % vers le composant contrôleur.
comportement
– si le port "p-température" indique 1500°C
alors – mettre voyants à "allumés".
– si le port "p-température" indique une baisse de température du produit
alors signaler, par le port "p-débit", une diminution du débit.
– si le port "p-température" indique une hausse de température du produit
alors – signaler, par le port p-débit, une augmentation du débit.
lois
% on contraint l'évolution des mesures du capteur température
(1) la température reçue sur le port "p-température" doit être entre 300°C et 1000°C
% on contraint l'évolution du débit en fonction de la température
(2) le débit est proportionnel à la température

Figure 0.7: spécification du composant corpusculaire "débimètre".

structure contrôleur

les variables d'état

- état ∈ {marche,arrêt}
- diodes ∈ {éteintes,clignotantes}

les ports en entrée

- p-niveau-lubrifiant % en provenance du capteur contrôlant le niveau du liquide lubrifiant.
- p-débit % en provenance du composant débimètre.

les ports de sortie

- p-vitesse % vers le composant moteur.
- p-position-électro-vanne % vers le composant électro-vanne.

comportement

- si le port "p-niveau-lubrifiant" indique une valeur inférieure à 5 Volts
  alors – mettre diodes en "clignotement",
  –signaler, par le port "p-arrêt-moteur", l'arrêt moteur.
- si le port "p-débit" indique une hausse du débit
  alors si la position de l'électro-vanne est "moyenne"
  alors signaler, par le port p-électro-vanne, une ouverture de l'électro-vanne.
  sinon signaler, par le port vitesse, d'augmenter la vitesse moteur.
- si le port "p-débit" indique une baisse du débit
  alors si la position de l'électro-vanne est "grande"
  alors signaler, par le port "p-électro-vanne", de diminuer l'ouverture de l'électro-vanne.
  sinon signaler, par le port "vitesse", de diminuer la vitesse moteur.

lois

% on contraint l'évolution des mesures du capteur niveau lubrifiant

(1) la mesure transmise par ce capteur doit être comprise entre 5 et 10 Volts.

% on contraint l'état des diodes

(2) si les diodes clignotent alors le capteur du niveau de lubrifiant doit indiquer une valeur inférieure à 5 Volts.

Figure 0.8: spécification du composant corpusculaire "contrôleur".

structure électro-vanne

les variables d'état

- état ∈ {marche,arrêt}
- position-ouverture ∈ {faible,moyenne,grande}
- indicateur-vanne ∈ $[5, 10]mA$

les ports en entrée

- p-position-électro-vanne

comportement

- si le port "p-position-électro-vanne" indique une augmentation d'ouverture
  alors si position-ouverture est "faible" alors mettre position-ouverture à "moyenne".
- si le port "p-position-électro-vanne" indique une augmentation d'ouverture
  alors si position-ouverture est "moyenne" alors mettre position-ouverture à "grande".
- si le port "p-position-électro-vanne" indique une diminution d'ouverture
  alors si position-ouverture est moyenne alors mettre position-ouverture à "faible".
- si le port "p-position-électro-vanne" indique une diminution d'ouverture
  alors si position-ouverture est "grande" alors mettre position-ouverture à "moyenne".

lois

% on contraint la position de la vanne en fonction de l'indicateur.

(1) si la vanne est en position "faible" alors l'indicateur doit indiquer une valeur ∈ $[5, 6[mA$.

(2) si la vanne est en position "moyenne" alors l'indicateur doit indiquer une valeur ∈ $[6, 8[mA$.

(3) si la vanne est en position "grande" alors l'indicateur doit indiquer une valeur ∈ $[8, 10]mA$.

Figure 0.9: spécification du composant corpusculaire "électro-vanne".

structure moteur
les variables d'état
– état-moteur ∈ {marche,arrêt}
– vitesse ∈ {0,500,1000,1500,2500,3000}
– intensité ∈ $[0, 150A]$
les ports en entrée
– p-vitesse % en provenance du composant contrôleur
comportement
– si le port "p-vitesse" indique "arrêt"
alors mettre "vitesse" à zéro.
– si le port "p-vitesse" indique une augmentation
alors faire croître la vitesse.
– si le port "p-vitesse" indique une diminution
alors faire décroître la vitesse.
lois
% on contraint globalement la vitesse du moteur.
(1) si le moteur est en marche alors la vitesse doit être comprise entre 1000 et 3000.
% on contraint la vitesse du moteur en fonction de l'intensité.
(2) si la vitesse ∈ $[500, 1000[$ alors l'intensité doit être comprise entre $[0, 50[$.
(3) si la vitesse ∈ $[1000, 2500[$ alors l'intensité doit être comprise entre $[50, 125[$.
(4) si la vitesse ∈ $[2500, 3000]$ alors l'intensité doit être comprise entre $[125, 150]$.

Figure 0.10: spécification du composant corpusculaire "moteur".

Remarques:

- Contrairement à la présentation faite ici, la construction de la hiérarchie ainsi que l'écriture des spécifications ne se font pas de manière linéaire mais au contraire de nombreux retours arrière sont souvent nécessaires.

- Alors que, délibérement nous avons adopté une démarche descendante, on pourrait également adopter la démarche ascendante en commençant par les composants corpusculaires ou combiner les deux. Nous reviendrons ultérieurement sur ce problème.

- L'examen des spécifications à partir des composants corpusculaires vers ceux abstraits fait apparaître une généralisation croissante des spécifications, c'est-à-dire une abstraction de plus en plus élevée. Au contraire, l'examen des spécifications des composants abstraits vers ceux corpusculaires montre une spécialisation croissante qui se traduit par un renforcement des contraintes.

troisième étape : **Elaboration du diagnostic**

Après avoir décrit l'installation, il nous faut résoudre le problème de diagnostic. Nous allons donc montrer notre démarche pour diagnostiquer les pannes, ce qui nous conduit à présenter succinctement:

- notre mécanisme de diagnostic,

- notre méthode de spécification des connaissances nécessaires au mécanisme de diagnostic.

Commençons par donner une idée schématique sur le mécanisme de diagnostic. A cet effet, reprenons le schéma de la hiérarchie de composants déjà construite:

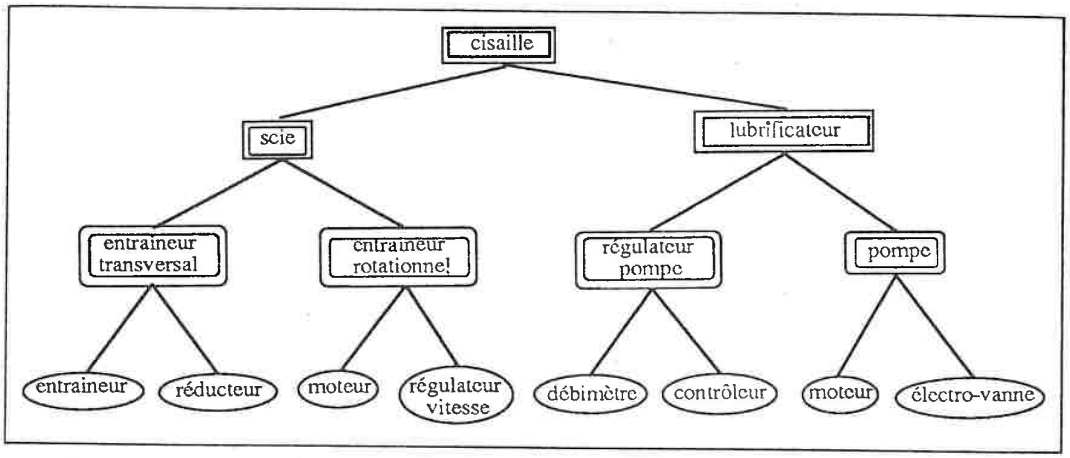


Figure 0.3 (reproduite): hiérarchie de composants du sous-système "cisaille".

Maintenant, supposons un dysfonctionnement au niveau du composant "lubrificateur": par exemple un problème au niveau du débit du lubrifiant. C'est ce que les dépanneurs appellent un symptôme. Ce disfonctionnement doit se traduire au niveau de ce composant par la violation d'une ou de plusieurs lois (Cf. lois sur le débit fig. 0.4). Le but du mécanisme du diagnostic est de rechercher la cause de ce problème du débit, c'est-à-dire rechercher, parmi les composants utilisés par le "lubrificateur", les composants-cause. Pour effectuer cette recherche, le mécanisme utilise des heuristiques d'exploration de la hiérarchie, appelées stratégies, qui sont spécifiées par l'utilisateur. Un exemple de stratégie dite explicite est: *focaliser la recherche sur tels composants.*

Supposons ici que la stratégie spécifiée par l'utilisateur au niveau du composant "lubrificateur" indique qu'il faut focaliser la recherche au niveau du composant "pompe". Le mécanisme de diagnostic considère alors ce composant comme la cause[6] du problème de débit.

Comme le composant "pompe" n'utilise que des composants corpusculaires, le mécanisme de diagnostic applique alors une procédure locale de diagnostic pour déterminer, parmi les composants moteur et électro-vanne, celui qui est la cause de départ, c'est-à-dire les composants corpusculaires en panne. De nouveau cette procédure est spécifiée par l'utilisateur. Dans la plupart des cas, ces procédures, que nous appellerons des tactiques locales, sont fournies par les constructeurs des composants. Dans ce cas simple, l'application de la tactique devra identifier le composant corpusculaire "électro-vanne" comme étant le

[6]Comme nous le verrons, l'application de toute stratégie est en réalité conditionnée par des préconditions. Le mécanisme ne considère alors ce composant comme étant la cause que lorsque les préconditions sont vérifiées.

composant en panne, c'est-à-dire responsable de la violation des contraintes relatives au débit.

Remarques:

- Alors que les tactiques ne sont appliquées qu'une seule fois au niveau des composants terminaux (i.e ceux qui utilisent uniquement des composants corpusculaires), les stratégies sont appliquées plusieurs fois. Dans ce cas, si le mécanisme de diagnostic avait démarré à partir du composant "cisaille", ce sont les stratégies spécifiées au niveau de ce composant "cisaille" qui auraient conduit au composant de plus bas niveau "lubrificateur".

- En réalité, l'application de chaque tactique est soumise à certaines hypothèses comme par exemple: *un composant en panne n'influence en aucun cas les autres composants.* Il est clair que ce type d'hypothèses est difficile à vérifier. Face à ce problème, le mécanisme de diagnostic utilise des règles empiriques, appelées par la suite des paradigmes, pour confirmer ou infirmer la panne des composants corpusculaires. Un exemple d'une règle fournie par les experts est: *si l'électro-vanne est en panne alors la tension à ses bornes doit osciller rapidement.*

La tâche de l'utilisateur consiste donc à:

(1) analyser les lois de chaque composant appartenant à la hiérarchie.

(2) spécifier:

  - les stratégies à conduire au niveau de chaque composant abstrait.
  - les tactiques à mettre en oeuvre au niveau de chaque composant terminal.
  - les paradigmes à mettre en oeuvre au niveau de chaque composant corpusculaire.

7 Plan de cette thèse

Après ce bref exposé informel de nos propositions, voyons à présent le plan de cette thèse. Suivant l'analyse effectuée au paragraphe 3, nous avons organisé la thèse comme suit:

- le premier chapitre sera consacré à la présentation de notre méthode de spécification que nous illustrerons par des exemples dans le domaine de l'industrie sidérurgique, mais qui seront toutefois présentés ici de manière simplifiée.

  Dans ce chapitre nous montrerons tout d'abord comment structurer une grande installation en différentes hiérarchies et différents graphes de composants "abstraits"

et "corpusculaires". Ensuite, nous verrons comment spécifier partiellement ces composants en exprimant successivement la structure, le comportement, les contraintes et l'interface de chacun d'entre eux.

Nous montrerons enfin la spécification de l'environnement (i.e le contexte) de l'installation en le considérant comme un composant, virtuel.

- dans le second chapitre, nous présenterons notre synthèse des techniques de diagnostic. Il ne s'agit nullement d'un exposé linéaire des différentes et nombreuses techniques utilisées, mais au contraire, c'est le résultat de notre étude et analyse bibliographique que nous présentons ici. De cette étude, il ressort 4 classes de techniques. Pour chacune, nous présentons principe, avantages, et restrictions, ainsi que nos propositions vis à vis de ces restrictions.

  Ce chapitre est en fait la "charnière" entre les chapitres 1 et 3. Il constitue aussi un prélude au chapitre 3.

- dans le troisième chapitre, c'est la méthode de diagnostic qui sera détaillée. Nous commencerons tout d'abord par définir très précisément les concepts de panne et de symptôme qui se trouvent mal définis dans la littérature. Nous définirons par la suite les stratégies d'exploration de la hiérarchie des composants ainsi que les tactiques qui déterminent les composants en panne. La méthode de justification et les paradigmes, qui servent à confirmer ou à infirmer la culpabilité des composants en panne, seront alors introduits. Nous montrerons ensuite comment compléter les spécifications partielles pour inclure les données du mécanisme du diagnostic. Nous présenterons enfin notre mécanisme de diagnostic. Nous donnerons alors notre principe de coopération entre deux types de raisonnement (i.e le raisonnement profond et celui de surface) qui sont souvent considérés comme opposés dans la littérature et nous présenterons les avantages de cette coopération.

  En quelque sorte ce chapitre peut être considéré comme un chapitre "pivot" de cette thèse.

- dans le quatrième chapitre, nous nous consacrerons d'une part aux outils de spécification et de construction des systèmes interactifs de diagnostic industriel, d'autre part à l'expérimentation sur des cas réels. Nous commencerons par définir le concept d'environnement de spécification et de construction de ces systèmes. Les différentes constituantes (i.e méthodologique, fonctionnelle et ergonomique) seront alors introduites. Nous présenterons ensuite l'organisation des connaissances en bibliothèques de catalogues ainsi que la structuration des outils en modules fonctionnels. Par la suite, nous traiterons l'aspect lié à l'expérimentation. Nous décrirons alors brièvement IOTA, sous-système d'une installation de laminage, et nous présenterons un

exemple de session de travail du système *SIDI* sur IOTA. Nous clôturerons ce chapitre en décrivant quelques exemples de systèmes significatifs de diagnostic connus dans la littérature.

Ce chapitre traite donc les aspects techniques de ce travail. Il inclut aussi une étude bibliographique complémentaire de celle faite au chapitre 2.

- en épilogue, nous ferons le bilan de ce travail et analyserons chaque chapitre, pour souligner très précisément notre contribution personnelle. Nous procéderons ensuite à une critique de ce travail et proposerons alors des prolongements possibles tant au niveau méthodologique que théorique.

Chapitre 1

Structuration et spécification de l'installation

La spécification d'une installation donnée doit être simple *à construire mais aussi* riche *en connaissances. Elle doit être également* précise *et bien* adaptée *au problème traité, à savoir ici le diagnostic de pannes. La facilité et la richesse recherchées peuvent être obtenues en disposant d'un guide méthodologique et en adoptant une structure arborescente pour décomposer l'installation en plusieurs hiérarchies pour mettre en évidence plusieurs vues (i.e logique, physique, électrique) de l'installation. La précision et l'adéquation nécessitent, quant à elles, de disposer d'un langage formel de spécification doté d'opérateurs appropriés au problème du diagnostic.*

Avec ces objectifs nous définissons tout d'abord les concepts de composant **abstrait**, *de composant* **terminal**, *de composant* **corpusculaire**, *de la relation de* **base** *"utilise" et enfin des relations dites* **secondaires** *qui apparaissent comme des outils d'abstraction et d'agrégation pour structurer l'installation. Nous introduisons ensuite notre langage de spécification pour pouvoir exprimer progressivement la structure, le comportement, les lois et enfin l'interface des composants abstraits et corpusculaires identifiés auparavant. La préoccupation majeure de ce chapitre se situe donc tant au niveau méthodologique qu'au niveau spécification formelle et rejoint ainsi les motivations initiales de ce travail.*

Ainsi, au premier paragraphe, les différents concepts, évoqués plus haut, seront définis. Dans un but illustratif, des exemples d'installations seront également présentés. La spécification de la **structure** *des composants exprimée en terme de variables d'état, de ports d'entrée, de sortie ainsi que des informations générales (i.e l'identifiant et ses synonymes, la date de mise en service, la localité, ...) occupera le §2. La spécification du* **comportement** normal *des composants fera l'objet du §3. Elle sera exprimée par des expressions formées d'opérateurs de manipulation des variables d'état et des ports. Quant au §4, il sera consacré à la spécification des* **lois** *des composants exprimées par des expressions conditionnelles. Les lois apparaîtront alors comme des contraintes sur le fonctionnement normal des composants. L'***environnement** *(i.e le contexte) de l'installation, défini comme un composant virtuel, sera spécifié au §5. Le §6 sera lui consacré à la spécification de l'***interface** *des composants. On y précisera les règles d'acquisition des données externes (i.e directives de l'agent opérateur, mesures fournies par les capteurs, ...). Dans le dernier paragraphe, nous récapitulerons les différentes étapes conduisant aux spécifications partielles des composants.*

1 Structuration de l'installation

Une installation industrielle est en général un objet complexe et difficile à appréhender comme un tout. On se propose alors de la structurer de manière hiérarchique en utilisant les concepts définis comme suit:

Définition 1: Composant Abstrait ($\mathcal{CA}$)
un $\mathcal{CA}$ est une boîte noire caractérisée par:

(1) un ensemble de variables d'état,

(2) un ensemble de ports d'entrée,

(3) un ensemble de ports de sortie.

□

Exemple, considérons une pompe comme un $\mathcal{CA}$:

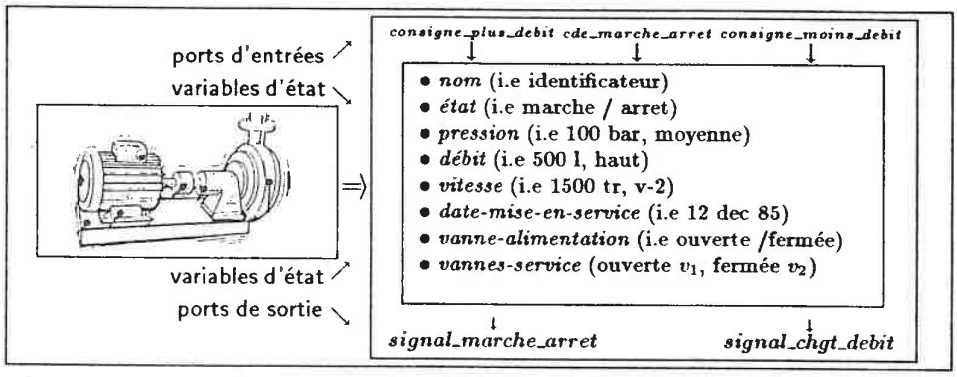


Figure 1.1: exemple d'abstraction.

Définition 2: Relation de Base $\mathcal{U}$tilise ($\mathcal{U}$)
la relation de base $\mathcal{U}$ joue le rôle d'agrégation entre composants. Intuitivement cette relation permet de construire d'autres composants abstraits à partir de ceux déjà construits. Elle permet également de jouer le rôle de désagrégation en vue de décomposer un composant abstrait donné en d'autres composants.
Nous écrirons $\mathcal{U}(C, c_1, \cdots, c_n)$ pour exprimer que les $\mathcal{CA}$s notés $c_1, \cdots, c_n$ interagissent pour produire le comportement global du $\mathcal{CA}$ noté C. De manière duale nous exprimons que le comportement du $\mathcal{CA}$ noté C est équivalent à la mise en interaction des $\mathcal{CA}$s notés $c_1, \cdots, c_n$.

□

Exemple: supposons que l'on a défini trois composants: un moteur, une vanne d'alimentation et une vanne de service. Par cette relation "utilise" on peut alors former le $\mathcal{CA}$ "pompe".

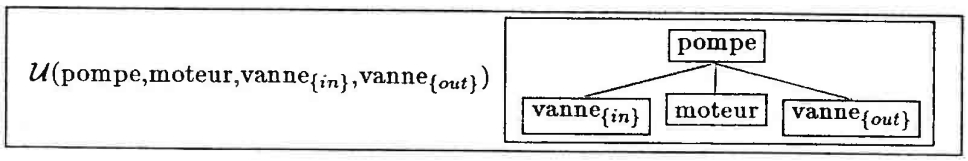


Figure 1.2: exemple d'agrégation par la relation "utilise".

Définition 3: Composant Corpusculaire ($\mathcal{CC}$)
un $\mathcal{CC}$ est analogue à un $\mathcal{CA}$. Sa particularité est qu'il ne peut être décomposé en d'autres composants. Intuitivement un $\mathcal{CC}$ correspond à la plus petite unité structurelle ou encore le composant de "base". Il constitue ainsi notre critère d'arrêt lors de la décomposition.

□

Le choix des $\mathcal{CC}$s rêvet une grande importance de la part des agents de maintenance de l'installation. En effet, traduisant le degré de granularité de l'installation, ce choix a des conséquences sur le mécanisme de diagnostic. Pour fixer les idées, considérons le $\mathcal{CA}$ "pompe" comme un $\mathcal{CC}$. Cela signifie alors que le mécanisme de diagnostic ne peut discerner le dysfonctionnement du moteur de la pompe de celui des vannes. Le mécanisme sera limité à diagnostiquer une panne au niveau de la pompe.

Définition 4: Composant Terminal ($\mathcal{CT}$)
un $\mathcal{CT}$ est analogue à un $\mathcal{CA}$. Sa particularité est qu'il n'utilise que des $\mathcal{CC}$s.

□

Définition 5: Relations Secondaires
On distingue deux familles de relations secondaires. Tout d'abord les relations inter-$\mathcal{CA}$s comme par exemple contrôle, communique etc. Ensuite les relations inter-$\mathcal{CC}$s comme par exemple adjacence-électrique, adjacence-électro-magnétique, etc.

□

La structuration d'une installation donnée consiste à:

(1) construire une première hiérarchie, dite principale, avec la relation "utilise". Cette hiérarchie, dont les noeuds sont des $\mathcal{CA}$s et les feuilles sont des $\mathcal{CC}$s, donne une vue "logique" de l'installation.

(2) exploiter les relations inter-$\mathcal{CA}$s pour construire des hiérarchies et des graphes secondaires. Selon la relation utilisée, chacune de ces hiérarchies et ces graphes donne, à travers les $\mathcal{CA}$s, une vue "flot-d'information", "contrôle", ... de l'installation.

(3) utiliser les relations inter-$\mathcal{CC}$s et construire des graphes étiquetés pour donner, à travers les $\mathcal{CC}$s, une vue électrique, physique, etc de l'installation.

Remarque: dans la suite de ce chapitre uniquement, sauf mention contraire, on désignera un composant abstrait ou terminal par $\mathcal{CA}$. De même, on utilisera le terme "composant" pour désigner indifférement un $\mathcal{CA}$, un $\mathcal{CT}$ ou un $\mathcal{CC}$.

1.1 Exemples

Donnons à présent deux exemples d'installations sidérurgiques: le haut-fourneau et les laminoirs. Il ne s'agit pas ici de traiter en détail ces installations mais seulement de montrer des exemples de hiérarchies et de graphes d'adjacences.

Exemple 1: l'installation du haut-fourneau.

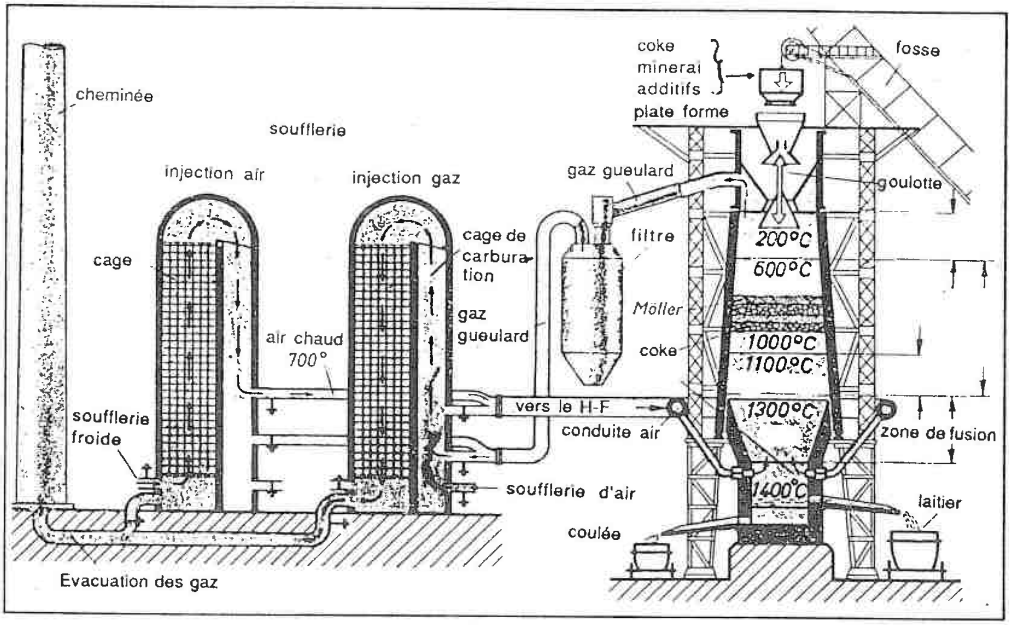


Figure 1.3: schéma simplifié du haut-fourneau.

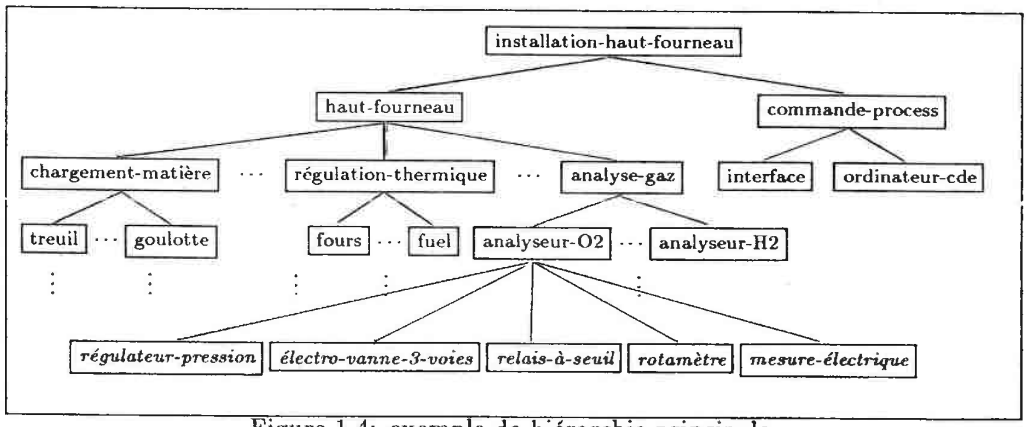


Figure 1.4: exemple de hiérarchie principale.

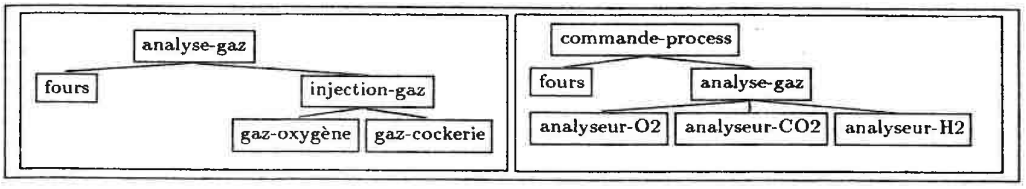


Figure 1.5: exemple de hiérarchies secondaires "contrôle".

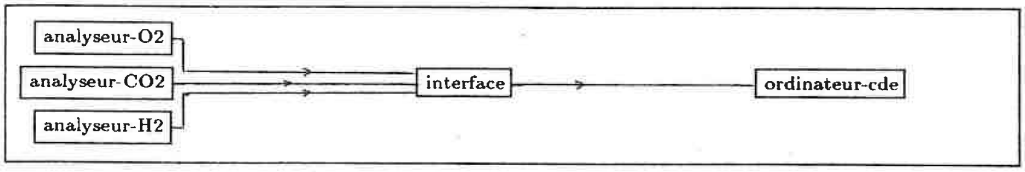


Figure 1.6: exemple de graphe "flot-informations".

La figure 1.4 donne une partie de la hiérarchie principale de cette installation où les *CC*s sont en italique. La figure 1.5 présente des exemples de hiérarchies secondaires construites par la relation inter-*CA*s "**contrôle**". Enfin, la figure 1.6 à son tour donne un exemple de graphe construit par la relation inter-*CA*s "**communique**".

Exemple 2: l'installation de laminage.

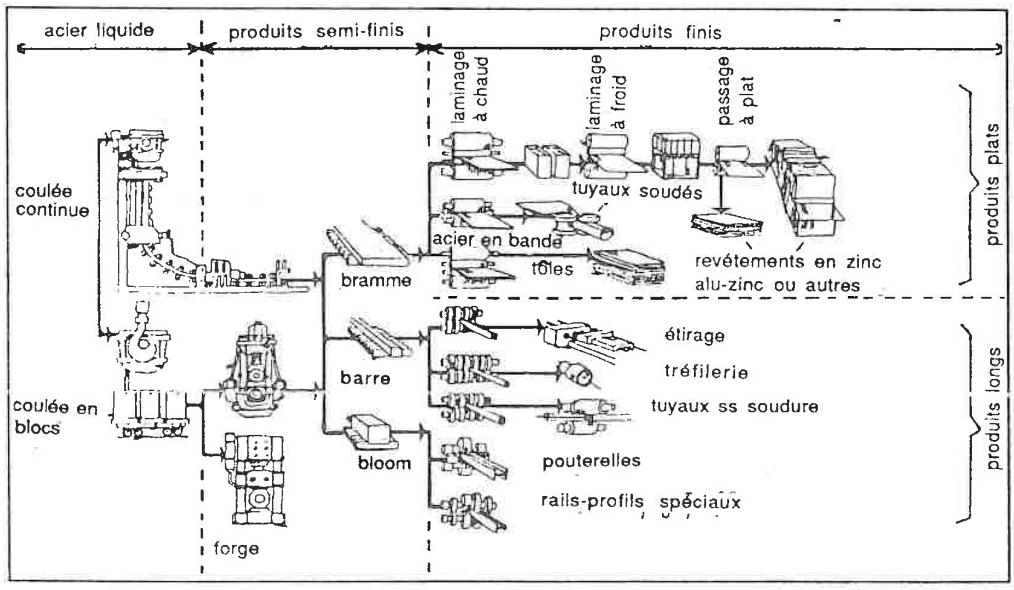


Figure 1.7: schéma des laminoirs.

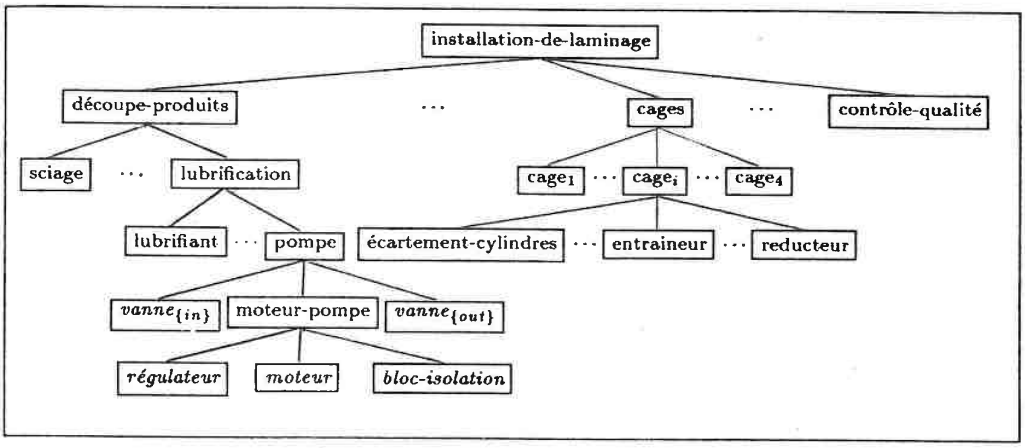


Figure 1.8: un autre exemple de hiérarchie principale.

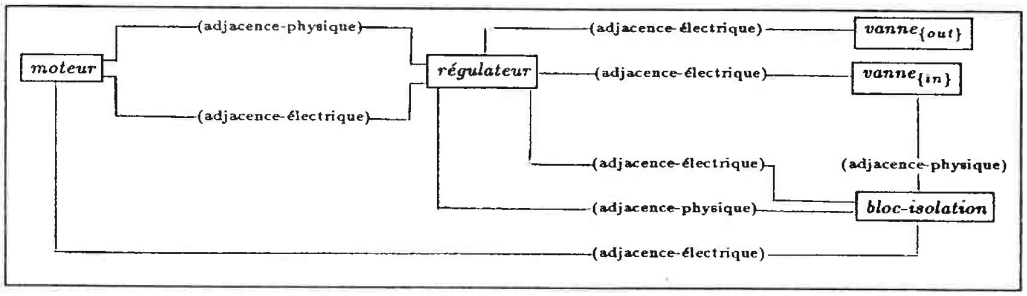


Figure 1.9: exemple de graphe d'adjacences des $\mathcal{CC}$s.

De nouveau la figure 1.8 donne un autre exemple de hiérarchie principale où le $\mathcal{CA}$ "pompe" a été décomposé en 2 niveaux. La figure 1.9 présente un exemple de graphe des $\mathcal{CC}$s où l'on a utilisé les relations "**adjacence-électrique**" et "**adjacence-physique**".

1.2 Démarche descendante vs ascendante

En fonction de l'ampleur de l'installation, l'étape de construction de la hiérarchie principale peut s'avérer parfois complexe. Un "artifice", pour atténuer cette difficulté, consiste à partitionner l'installation en *sous-systèmes* avant de procéder à la construction des hiérarchies. Pour cela, deux démarches peuvent être utilisées:

- la première consiste à construire progressivement la hiérarchie en commençant par la racine pour atteindre finalement les feuilles (i.e les composants corpusculaires), ce qui revient à faire une première abstraction considérant le sous-système comme un seul composant abstrait et à décomposer progressivement ce composant en d'autres composants abstraits de plus bas niveau. Cette démarche, souvent utilisée dans la conception des systèmes d'information, est qualifiée de *descendante.*

- au contraire, la seconde démarche, qualifiée d'ascendante, consiste à identifier en premier lieu les composants corpusculaires (i.e les feuilles) de la hiérarchie et construire progressivement par la suite les composants abstraits jusqu'à atteindre la racine. Cette démarche est également utilisée dans la conception des systèmes d'information.

Dans les systèmes d'information, la démarche ascendante est conseillée si l'on maîtrise parfaitement l'univers du problème (i.e le sous-système industriel) pour pouvoir identifier dès le départ les unités de base (i.e les composants corpusculaires). Au contraire, la démarche descendante peut être conseillée si l'on maîtrise peu ou mal l'univers du problème. L'expérience nous a montré qu'il n'y a pas une utilisation exclusive de l'une ou de l'autre des démarches mais que l'on est souvent conduit à combiner les deux. Dans la plupart des cas, on adopte tout d'abord une démarche descendante pour faire une structuration grossière. Ensuite, on corrige cette structuration en adoptant une démarche ascendante. Ce mécanisme de "va-et-vient" entre les deux démarches peut se répéter plusieurs fois avant d'aboutir à une structure hiérarchique définitive.

1.3 Remarques

- A ce stade, l'exploitation des hiérarchies et des graphes d'adjacence est à un niveau purement conceptuel. Le choix arbitraire des relations secondaires peut alors sembler peu naturel. Nous verrons au chapitre 3 que ce choix est en réalité guidé par le mécanisme de diagnostic. En effet, les graphes, les hiérarchies et les tactiques constituent en fait des données centrales pour le mécanisme proposé.

- Nous avons déjà souligné auparavant qu'un composant abstrait ne correspond pas nécessairement à un composant réel dans l'installation; c'est la première justification du terme abstrait. A travers les exemples donnés ci-dessus, nous constatons de plus que les composants abstraits ne correspondent pas forcément à des stades de fabrication, ni à des identités géographiques de lieu. C'est la deuxième justification du terme composant abstrait: entité logique résultant d'une abstraction conduite par l'utilisateur.

2 Spécification de la structure

Un composant étant une *boîte noire* constituée de trois ensembles:
(1) les variables d'état qui le caractérisent,
(2) les ports d'entrée qui lui permettent d'envoyer des messages aux autres composants,
(3) les ports de sortie qui lui servent à recevoir des messages d'autres composants.

C'est ce triplet que nous appellerons la structure d'un composant. Ce concept de boîte noire a été utilisé, de manière sensiblement différente, par Davis [Davis 84] dans le domaine

de l'électronique. Lors de la présentation de notre synthèse des techniques de diagnostic au chapitre 2, nous reviendrons sur ce concept tel qu'il est perçu par d'autres.

Notre objectif, à présent, est donc de décrire la structure de chaque composant appartenant à la hiérarchie principale. Ceci nous conduit à classer les variables d'état et les ports et à préciser la connexion des ports de sortie au préalable avant d'en donner une description formelle à l'aide de notre langage de spécification.

2.1 Les classes des variables d'état

Actuellement, on distingue trois classes de variables d'état que nous avons décidé d'appeler:

- **available**: cette classe regroupe les variables d'état pour lesquelles on dispose d'un moyen direct (i.e un ou plusieurs capteurs) pour connaître la valeur réelle de cette variable dans l'installation.

- **deduced**: cette classe concerne les variables d'état pour lesquelles on n'a pas de moyen direct mais des équations permettant la déduction de sa valeur réelle. Par exemple, supposons que la variable d'état "*debit*" du composant "pompe" est de cette classe. On peut, déduire cette valeur de manière quantitative par la formule:

  $$debit = \frac{puissance}{pression}$$

  ou encore, par des règles "qualitatives" telles que

  $$\gg(debit) \text{ :- } \gg(puissance), \ll(pression).$$
  $$\gg(debit) \text{ :- } \gg(vitesse).$$
  $$\ll(debit) \text{ :- } \ll(vitesse).$$

  qui, traduisent l'évolution de cette variable. La première règle exprime que *si la puissance croît et la pression décroît, on peut alors déduire que le débit croît.* Le but des règles qualitatives est de rendre compte de l'évolution (i.e croissante, décroissante, stable) d'une variable d'état donnée. Nous reviendrons sur ce sujet au §6 de ce chapitre qui est consacré à la spécification de l'interface des composants.

- **asked**: cette classe indique que pour connaître la valeur réelle de la variable d'état appartenant à cette classe, on est conduit à interroger l'opérateur. Notons que pour des raisons techniques de réalisation, seules les variables d'état d'un *CC* peuvent être de cette classe. Ces raisons sont détaillées au chapitre 4.

2.2 Les classes des ports

Au niveau des ports, ou plus précisement des messages échangés entre les ports de sortie et ceux d'entrée des composants, on distingue aussi trois classes appelées:

- detected: cette classe concerne les messages, échangés entre composants, pour lesquels on a un moyen direct (par exemple un capteur) pour détecter leurs occurrences dans l'installation.

- undetected: cette classe indique qu'il n'est pas possible de détecter, dans l'installation, les occurrences des messages reçues (ou envoyées) par le port considéré. Cette impossibilité peut être dûe à des raisons techniques (e.g coût élévé) ou encore à des problèmes de sécurité (e.g présence de gaz, haute température).

- deduced: cette classe concerne les messages d'un port pour lesquels, on ne dispose pas d'un moyen direct de détection mais pour lesquels on dispose d'équations permettant la détection de leurs occurrences dans l'installation. Exemple l'équation:

  $\#(signal\text{-}chgt\text{-}debit)$:- $\#(consigne\text{-}plus\text{-}debit) \vee \#(consigne\text{-}moins\text{-}debit)$

  indique que la détection d'une occurrence sur le port "signal-chgt-debit" revient à détecter un message, soit sur le port "consigne-plus-debit", soit sur le port "consigne-moins-debit". L'opérateur noté #(p), qui signifie intuitivement "détecter un message sur le port p", sera défini au §3.

L'ensemble des formules, des règles qualitatives et des équations de déduction seront détaillées à la spécification de l'interface présentée au §6.

2.3 Exemples

Comme le montrent les exemples suivants, la syntaxe de notre langage pour spécifier la structure des composants est facilement compréhensible[1]. Contentons nous ici des quelques exemples commentés:

Exemple 1: spécification de la structure du *CC* "vanne" de la figure 1.8.

```
(1)  vanne{in}(pompe);
(2)  synonym:  alim, valve;
(3)  location:  local-scierie;
(4)  belongs:  graphe1;
(5)  type:  active
     STATE % les variables d'état
(6)  état:  asked init fermée restricted {ouverte,fermée};
     INPUT % les ports d'entrée
(7)  cde-ouverture-fermeture:  undetected;
```

Figure 1.10: spécification de la structure du *CC* "vanne$_{\{in\}}$".

Cette spécification comprend:

- les informations "générales" sur ce composant:

[1]La syntaxe complète du langage est donnée à l'addenda B.

- ses *synonymes*, permettant ainsi l'introduction du vocabulaire technique et les conventions propres aux agents de maintenance pour une installation particulière.
- sa *localité* ou lieu physique. On verra, par la suite que ce détail technique est très important pour le mécanisme de diagnostic.
- les *graphes* auxquels il appartient. Ce qui permet de procéder à des contrôles élémentaires (Cf. §6).
- son "type". Nous distinguons trois types de composants: **active**, **passive** et **virtual**.

  Le type *"active"* est réservée aux composants actifs, c'est-à-dire ceux qui interagissent avec d'autres composants et jouent un rôle comportemental dans l'installation. De ce fait ces composants, peuvant tomber en panne, admettent des spécifications du comportement, des lois, etc (cf. paragraphes suivants).

  Au contraire, le type "passive" est réservé aux composants passifs tel qu'un réservoir par exemple. Ces composants ont un rôle purement conceptuel. La spécification de ce type de composants se limite à exprimer uniquement leurs structures.

  Enfin, le type "virtual" est reservé au composant, virtuel, qui est l'environnement de l'installation. On reviendra sur ce sujet au paragraphe 5.

  Dans la suite de ce travail et sauf mention contraire, seuls les composants actifs seront considérés, nous omettrons alors de donner le type des composants.

- les variables d'état. Dans ce cas on a une seule variable "état", de la classe "asked", initialisée à la valeur "fermée". Les valeurs que peut prendre cette variable sont restreintes à l'ensemble {ouverte, fermée}.
- les ports d'entrée qui permettent à ce composant de recevoir des messages d'autres composants. Seul le port "cde-ouverture-fermeture" de la classe "undetected" est spécifié ici.

Exemple 2: spécification de la structure du $\mathcal{CA}$ "pompe" de la figure 1.8.

```
(1)  pompe(lubrification);
(2)  synonym:  lubrificateur;
(3)  location:  local-scierie;
(4)  belongs:  graphe_1;

     STATE % les variables d'état
(5)  puissance :  available range [0,2000] ;
(6)  pression :  available ;
(7)  vitesse :  available ;
(8)  débit :  deduced ;
(9)  vanne_{in}:  available ;
(10) vanne_{out}:  available ;
(11) statut :  available ;

     INPUT % les ports d'entrée
(12) cde-marche-arret :  detected ;
(13) consigne-plus-débit :  detected ;
(14) consigne-moins-débit :  detected ;

     OUTPUT % les ports de sortie et leurs connexions
(15) signal-M-A : undetected to tableau-de-bord ;
(16) signal-chgt-débit :  deduced to régulateur;
```

Figure 1.11: spécification de la structure du $\mathcal{CA}$ "pompe".

Par rapport à l'exemple précédent, cette spécification fait apparaître l'étendue ou le domaine (i.e "range") dans lequel varie continuellement la variable d'état "puissance". On retrouve la même notion (i.e restricted), introduite dans l'exemple précédent, pour la variable à valeurs discontinues "état". Nous reviendrons sur ce sujet au paragraphe suivant.

Exemple 3: spécification de la structure du $\mathcal{CA}$ "analyseur-O2" de la figure 1.4.

```
(1)   analyseur-O2(analyseurs-gaz);
(2)   synonym:  analyse-O2, gaz-O2;
(3)   location:  salle-opértateur;
(4)   belongs:  graphe4;
      STATE % les variables d'état
(5)   cumul-entrée-air :  deduced ;
(6)   moyenne :  available ;
(7)   teneur-mesurée-O2 :  deduced ;
(8)   teneur-gaz-étalon :  available ;
(9)   position-rélais-taquet :  available ;
      LOCAL % les variables auxiliaires
(10)  analyseurs :  local init {analyseur-CO2, analyseur-H2} ;
      INPUT % les ports d'entrée
(11)  teneur-O2-point-haut :  detected ;
(12)  teneur-O2-point-bas :  detected ;
(13)  signal-analyse :  detected ;
(14)  entrée-air :  detected ;
(15)  temperature-ligne-trt :  undetected ;
(16)  teneur-O2-HF : detected ;
(17)  teneur-gaz-bouteille :  detected ;
      OUTPUT % les ports de sortie
(18)  signal-electrovanne-O2 :  undetected to electro-vanne-O2 ;
(19)  signal-frigatron :  undetected to dispositif-sechage ;
(20)  signal-relais-taquet :  deduced to relais-à-seuil ;
(21)  signal-teneur-resultante :  detected to ! analyseurs ;
```

Figure 1.12: spécification de la structure du $\mathcal{CA}$ "analyseur-O2".

Dans cette spécification on voit apparaître des variables auxiliaires qui, contrairement aux variables d'état, ne caractérisent pas le composant mais jouent de manière locale le rôle des variables intermédiaires. Ici c'est la variable "analyseurs", initialisée par les identifiants des composants auxquelles le port "signal-teneur-resultante" est connecté, qui constitue la variable intermédiaire de cette spécification. Comme on le verra par la suite les variables intermédiaires ne seront pas directement exploitées. Leur rôle est alors purement conceptuel, c'est-à-dire faciliter les étapes suivantes de spécification.

3 Spécification du comportement

Après avoir spécifié la structure des composants, nous nous intéressons à présent de décrire leur fonctionnement en temps *normal*. Le but poursuivi par la spécification du comportement est de comprendre et d'appréhender le fonctionnement des boîtes noires, qui représentent les composants, pour identifier par la suite les lois qui gouvernent le fonctionnement de ces composants. Cette étape *intermédiaire* de spécification apparait

donc comme un *outil conceptuel* d'aide à l'identification des lois régissant le comportement normal des composants de l'installation.

Pour spécifier le comportement des composants, deux approches ont été proposées:

- l'approche fonctionnelle de Davis [Davis 84] utilisée dans le domaine de l'électronique. Dans ce cas, spécifier le comportement d'un composant revient à le considérer comme une fonction mathématique. Malheureusement, il n'est pas toujours possible de décrire précisément la fonction d'un composant quelconque. En effet, si la fonction d'un additionneur à deux entrées e_1 et e_2 est facile à exprimer (i.e fonction: $plus(e_1, e_2)$), il n'en est pas de même pour des composants tels qu'une pompe hydraulique, un talkie-walkie, etc [David 86].

- l'approche par la logique proposée par Reiter [Reiter 87]. A priori, cette approche n'est pas limitée à un type particulier de composants car on dispose de la puissance du langage du CP1. Malheureusement, la lourdeur des spécifications obtenues conduit à une inefficience du mécanisme de diagnostic [ElAyeb 88c].

Pour prendre en compte différents types de composants (e.g mécanique, électrique, etc), nous nous inspirons d'une approche par règles de production. Nous exprimons, par des règles, les relations entre ports d'entrée, variables d'état et ports de sortie. C'est la description de l'ensemble de ces règles que nous appellerons *spécification du comportement* ou encore *spécification comportementale* d'un composant. C'est aussi l'objectif de cette étape: spécifier le comportement de chaque composant appartenant à la hiérarchie principale afin de comprendre son fonctionnement normal.

Le concept de règle, souvent appelée règle de production, est très utilisé dans la littérature. La forme générale la plus répandue est:

$$\underbrace{op_{p_1} \cdots op_{p_i} \cdots op_{p_n}}_{\text{les opérateurs de la partie prémisse}} \quad \mapsto \quad \underbrace{op_{a_1} \cdots op_{a_i} \cdots op_{a_m}}_{\text{les opérateurs de la partie action}}$$

Figure 1.13: règle de production.

Si, à première vue, cette forme semble à la fois simple et facilement compréhensible, elle peut cependant devenir très vite illisible dès que le nombre d'opérateurs augmente sensiblement. C'est pourquoi nous proposons une autre forme (cf. figure 1.14) qui permet de structurer les différents opérateurs utilisés dans une règle.

Les opérateurs à résultat booléen de la partie prémisse se trouvent ainsi regroupés en trois catégories selon qu'ils manipulent les messages des ports d'entrée, les variables d'état ou le contenu des messages déjà reçus.

En poursuivant cette démarche, nous étiquetons les règles pour former des paquets de règles. Un paquet est une suite ordonnée de règles portant toutes la même étiquette et organisées comme suit: les premières sont celles qui ont les plus fortes conditions et les dernières sont celles qui ont les conditions les plus faibles. Cette convention doit alors permettre une plus grande lisibilité.

3.1 Paquets et unités de comportement

Plutôt que d'utiliser le terme "règle" nous convenons d'employer le terme **unité de comportement**. Nous parlerons aussi de paquets d'unités de comportement étiquétés. La spécification du comportement normal d'un composant consiste ainsi à décrire les différentes unités de comportement et à les structurer en paquets. A titre d'exemple, voici une unité de comportement:

on #(signal) in ≫(puissance) ↦ /<(débit,haut)

la sémantique intuitive de cette unité est: *si un message est reçu sur le port d'entrée "signal" et que la variable d'état "puissance" croît alors affecter "haut" à la variable d'état "débit"*. Maintenant, en utilisant la notation BNF, donnons la syntaxe pour décrire le comportement d'un composant:

<*comportement*> ::= begin <*paquets*> end
<*paquets*> ::= <*paquet*> { <*paquet*> }*
<*paquet*> ::= <*unite*> { <*unite*> }*
<*unite*> ::= LABEL ':' [on <$exp_{\{p\epsilon\}}$>] [in <$exp_{\{v\epsilon\}}$>] [with <$exp_{\{m\}}$>] ↦
<*action*> {, <*action*> }';'

Figure 1.14: syntaxe concrète pour la spécification du comportement.

Où

<$exp_{\{p\epsilon\}}$> est une expression conditionnelle portant sur la file des messages reçus par les ports d'entrée du composant.

<$exp_{\{v\epsilon\}}$> est une expression conditionnelle formée à l'aide des opérateurs de manipulation des variables d'état du composant.

<$exp_{\{m\}}$> est une expression qui manipule le contenu des messages reçus aux ports d'entrée du composant.

<*action*> est une opération qui doit être exécutée.

Donnons à présent la sémantique intuitive d'un paquet P formé de $u_1, \cdots, u_n$ unités de comportement:

l'évaluation des unités $u_1, \cdots, u_n$ se fait de manière séquentielle. A la première unité u_i où chaque opérateur des expressions $exp_{\{pc\}}$, $exp_{\{ve\}}$ et $exp_{\{m\}}$ renvoie la valeur vrai, exécuter les actions de cette unité et ignorer toutes les autres unités u_j telles que $u_j \in P$ et $j > i$.

3.2 Les opérateurs

Etant donnée la multiplicité des composants des installations industrielles, il sera vain de chercher à donner ici tous les opérateurs susceptibles d'être utilisés dans la construction des unités de comportements. De plus, sachant que ces spécifications ne seront pas directement exploitées mais constituent un étape intermédiaire pour les spécifications des lois, nous nous contenterons ici de donner les opérateurs de base.

Tout d'abord, nous présentons les opérateurs de la partie <*action*>:

- $/\#(exp, p)$: cet opérateur a pour effet d'associer la valeur de l'expression exp au port de sortie p. On appellera la valeur de cette expression le message associé au port p.
- $/!(p)$ envoie le message, associé au port de sortie p, aux composants qui ont un port d'entrée identifié par p.
- $\not<(v, exp)$ sauvegarde l'ancienne valeur de la variable v avant de lui affecter la valeur de l'expression exp.
- $@(v)$ cet opérateur unaire permet d'accéder à l'ancienne valeur de v donné en argument.

Ensuite pour la partie <$exp_{\{pc\}}$>, on a:

- $\#(port)$: renvoie *vrai* s'il existe au moins une occurrence d'un message sur le port donné en argument, faux sinon. Dans le cas où l'opérateur renvoie la valeur *vrai*, le port est dit *actif*.
- $\prec (p_1, p_2)$ renvoie *vrai* si les deux ports p_1 et p_2 sont actifs et que le premier message du port p_1 a été reçu avant celui sur le port p_2, faux sinon.
- $\equiv (p_1, p_2)$ renvoie *vrai* si le deux ports p_1 et p_2 sont actifs et que le premier message du port p_1 et celui du port p_2 ont été reçus simultanément.

Le second groupe d'opérateurs peut être qualifié de générique dans le sens où le résultat booléen de l'opérateur est conditionné par des règles explicitement données par l'utilisateur. Ces règles doivent être spécifiées à l'étape de la spécification de l'interface des composants (Cf. §6). Exemple: soit l'opérateur $\ll$, associons à cet opérateur et à la variable v les règles suivantes:

$\ll(v)$:- $\geq$(v,400) $\leq$(@(v),500)
$\ll(v)$:- $\geq$(v,300) $\leq$(@(v),400)

cet opérateur renvoie alors *vrai* si l'expression booléenne de l'une des règles est vérifiée. Intuitivement ces règles définissent les conditions dans lesquelles on peut considérer que la variable v est décroissante. En pratique, pour manipuler les variables d'état, nous avons besoin de savoir si la variable est croissante, décroissante ou stable. C'est pourquoi nous avons défini:

- $\gg(ve)$: cet opérateur renvoie vrai si au moins une des expressions associées à cet opérateur et à cette variable est évaluée à vrai, faux sinon.

  L'interprétation intuitive ici est: "vérifier si la variable donnée en argument est *croissante* ?".

- $\ll(ve)$: c'est le second opérateur qui fonctionne de la même façon que l'opérateur précédent. Dans ce cas l'interprétation est: "vérifier si la variable donnée en argument est *décroissante*".

- $\approx(ve)$: cet opérateur vient compléter les deux précédents. Il fonctionne aussi de la même façon que les deux premiers. Son interprétation est: "vérifier si la variable donnée en argument est *stable*".

Il est clair que l'interprétation intuitive n'a qu'un rôle illustratif et qu'en réalité ce sont les règles qui définissent la sémantique exacte de ces opérateurs.

De manière similaire, nous avons deux autres opérateurs génériques $/\gg(v)$ et $/\ll(v)$ pour manipuler les variables d'état dans la partie <*action*>. Dans ce cas, les règles, données aussi à l'étape de la spécification de l'interface, précisent les actions à entreprendre sur la variable d'état donnée en argument. Exemple: soit l'opérateur $/\gg$ défini sur la variable v et auquel on associe les règles suivantes:

$/<(v,$"moyen"):- $\#(m) = (@(v),$"bas")
$/<(v,$"haut"):- $\#(m) = (@(v),$"moyen")

l'effet de cet opérateur est alors d'affecter "moyen" à la variable v à condition que le port m soit actif et que l'ancienne valeur de cette variable soit "bas". De même en utilisant la seconde règle, l'effet de cet opérateur est d'affecter "haut" à cette variable à condition que le port m soit actif et que l'ancienne valeur de cette variable soit "moyen". Si les conditions de chacune des règles ne sont pas vérifiées, l'opérateur n'a aucun effet et la variable reste inchangée.

3.3 Exemples

A titre illustratif, donnons la spécification comportementale des composants "pompe" dont la structure a été spécifiée précédement.

Exemple 1

```
    begin
(18)init:  on #(cde-marche-arret) in =(statut,arret) ↦
               /<(statut,marche), /<(vanne{in},ouverte), />>(vitesse);
(19)init:  on #(cde-marche-arret) in =(statut,marche) ↦
               /<(statut,arret), /<(vanne{in},fermé), /<(vitesse,0);
(20)augmenter:  on #(consigne-plus-débit) in =(statut,arret) ↦ true;
(21)augmenter:  on #(consigne-plus-débit) in =(vitesse,3000) ↦ true;
(22)augmenter:  on #(consigne-plus-débit) ↦
               />>(vitesse),/>>(débit),/<<(pression);
(23)diminuer:  on #(consigne-moins-débit) in =(statut,arret) ↦ true;
(24)diminuer:  on #(consigne-moins-débit) in =(vitesse,1500) ↦ true ;
(25)diminuer:  on #(consigne-moins-débit) ↦
               /<<(vitesse),/<<(débit),/>>(pression);
    end
```

Figure 1.11 (suite): spécification du comportement du $\mathcal{CA}$ "pompe".

Voici une traduction algorithmique de cette spécification comportementale:

```
tant que "aucune divergence comportementale n'est détectée".
    si le port "cde-marche-arret" est actif et le statut de la pompe est en arrêt
        alors
            - mettre le statut de la pompe en marche.
            - mettre la vanne d'alimentation à l'état ouverte.
            - mettre la vitesse en état de croissance.
        sinon
                si le port "cde-marche-arret" est actif et le statut de la pompe est en marche
                    alors
                        - mettre le statut de la pompe en arrêt.
                        - mettre la vanne d'alimentation à l'état fermée.
                        - mettre la vitesse à zéro.
                fin.
    fin.

    si le port consigne-plus-débit est actif et le statut de la pompe est en arrêt
        alors
            - ne rien faire.
        sinon
                si le port consigne-plus-débit est actif et la vitesse de la pompe est maximale
                    alors
                        - ne rien faire.
                    sinon
                        si le port consigne-plus-débit est actif
                            alors
                                - mettre la vitesse en état de croissance.
                                - mettre le débit en état de décroissance.
                                - mettre la pression en état de croissance.
                        fin.
                fin.
    fin.

    si le port consigne-moins-débit est actif et le statut de la pompe est en arrêt
        alors
            - ne rien faire.
        sinon
                si le port consigne-moins-débit est actif et la vitesse de la pompe est minimale
                    alors
                        - ne rien faire.
                    sinon
                        si le port consigne-moins-débit est actif
                            alors
                                - mettre la vitesse en état de décroissance.
                                - mettre le débit en état de décroissance.
                                - mettre la pression en état croissance.
                        fin.
                fin.
    fin.
fin.
```

Exemple 2

Donnons maintenant la spécification du composant "analyseur-O2".

```
    begin
(23)uc1:  on #(signal-analyse) #(entrée-air) ≺(signal-analyse,entrée-air)
              ↦ /≻("Surveiller le capteur --entrée-d'air--")
(24)uc1:  on #(entrée-air) ↦ /≺(cumul-entrée-air,+(&cumul-entrée-air,1))
(25)uc1:  on #(signal-analyse) ↦ /!(signal-frigatron,signal-electrovanne-02)
              /≺(position-rélais-taquet,haut)
(26)uc2:  on #(temperature-ligne-trt) in
              =(position-rélais-taquet,haut) ↦ /!(signal-relais-taquet);
(27)uc2:  on #(temperature-ligne-trt) #(teneur-02-HF) ↦ true;
(28)uc2:  on #(temperature-ligne-trt) #(teneur-gaz-bouteille) ↦ true;
(29)uc2:  on #(teneur-02-HF) #(teneur-gaz-bouteille)
              in >(-(teneur-mesurée-02,teneur-gaz-étalon),25)
              ↦ /≺(moyenne,&lissage(teneur-mesurée-02,teneur-gaz-étalon));
              ⋮
    end
```

Figure 1.12 (suite): spécification du comportement du $\mathcal{CA}$ "analyseur-O2".

Voici une traduction algorithmique de cette spécification:

```
tant que "aucune divergence comportementale n'est détectée".
   si les ports "signal-analyse" et "entrée-air" sont actifs et
     le premier message sur le port "signal-analyse" a été reçu avant celui du port "entrée-air"
      alors – signaler à l'opérateur de "Surveiller le capteur entrée-d'air"
      sinon
            si le port "entrée-air" est actif
               alors – incrémenter "entrée-air" de 1.
                  sinon si le port "signal-analyse" est actif
                        alors – envoyer le message à partir du port "signal-frigatron"
                              – envoyer le message à partir du port "signal-électrovanne-O2"
                              – mettre "position-relais-taquet" en position "haut"
                  fin.
            fin.
   fin.
   si le port "température-ligne-trt" est actif et
      la "position-rélais-taquet" est position "haut"
      alors – envoyer le message du port "signal-relais-taquet"
      sinon
            si les ports "température-ligne-trt" et "teneur-O2-HF" sont actifs
               alors – ne rien faire.
               sinon
                  si les ports "température-ligne-trt" et "teneur-gaz-bouteille" sont actifs
                     alors – ne rien faire.
                     sinon si les ports "teneur-O2-HF" et "teneur-gaz-bouteille" sont actifs
                           et qu'il existe un grand écart (i.e 25 mA) entre les deux teneurs
                        alors – calculer la moyenne de la teneur gaz par la fonction
                                "lissage".
                              ⋮
                  fin.
            fin.
   fin.
fin.
```

Bien sûr, cette spécification est largement simplifiée ici. En réalité, elle est plus longue mais pas nécessairement plus complexe. En effet, l'objectif poursuivi par la spécification comportementale n'est pas de décrire en détail le fonctionnement des composants mais au contraire, par un effort d'abstraction, seul le comportement **externe** doit être pris en compte, *c'est-à-dire l'observation de l'évolution des variables d'état.*

4 Spécification des contraintes et des lois

Maintenant que nous avons appréhendé le fonctionnement des composants, notre but est de spécifier les lois en terme de contraintes sur leur fonctionnement normal. Voici des exemples simples de contraintes:
(c_1) *si l'intensité croît et que la tension est stable alors la puissance doit croître.*
(c_2) *si l'intensité décroît et que la tension est stable alors la puissance doit décroître.*

Supposons que chacune de ces grandeurs physiques soit représentée par une variable d'état, c_1 et c_2 apparaissent ainsi comme des contraintes régissant l'évolution de la variable "puissance". Dans la suite, nous appellerons une *loi* comme étant une disjonction de contraintes sur une ou plusieurs variables d'état données. Cette étape de spécification consiste donc à:

- identifier les lois régissant l'évolution des variables d'état de chaque composant.

- exprimer ces lois en terme de disjonction de contraintes.

Ce concept de contrainte a été utilisé, de manière sensiblement différente, dans [Gallanti 85] et [Gallanti 86]. Toutefois dans la plupart des cas, l'hypothèse suivante: *"les données primaires ne sont pas erronées"* est supposée vérifiée. C'est ainsi que dans l'exemple de l'additionneur-complet[2] ([Davis 84], [DeKleer 87], [Geffener 87]), les entiers fournis à l'additionneur sont supposés non erronés. Malheureusement, cette hypothèse, souvent implicite [ElAyeb 88a], peut s'avérer, dans de nombreux cas, non vérifiée (e.g les mesures fournies par les capteurs peuvent être erronées [Cohen 86], [Milne 87], [Rajagoplan 84], [Taunton 86]). Nous sommes alors conduits à exprimer aussi des contraintes sur les données primaires en provenance de l'environnement externe permettant ainsi de diagnostiquer les pannes induites par ces données erronnées.

Voici la forme générale d'une contrainte:

[2]Cet exemple, qui sera présenté dans le chapitre suivant, peut être considéré comme l'exemple de référence en diagnostic.

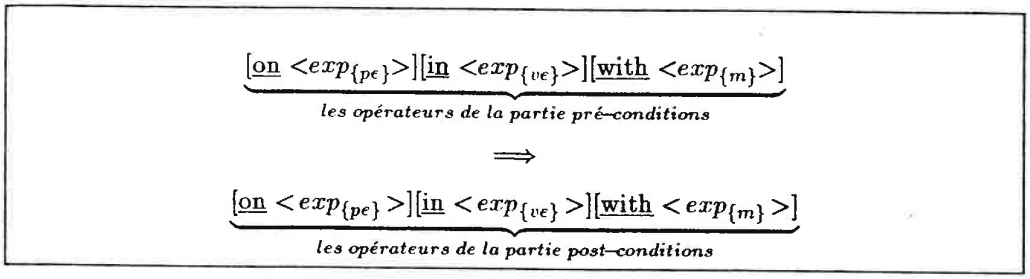


Figure 1.15: forme générale d'une contrainte.

où, comme auparavant, nous avons procédé à la structuration des différents opérateurs selon qu'ils font référence aux ports d'entrée (i.e on $<exp_{\{pe\}}>$), aux variables d'état (i.e in $<exp_{\{ve\}}>$) ou aux messages (i.e with $<exp_{\{m\}}>$).

Définition 6

Une contrainte est dite **violée** si les deux conditions suivantes sont satisfaites:

(1) chaque opérateur de la partie pré-condition est évalué à vrai.

(2) il existe au moins un opérateur de la partie post-condition qui est évalué à faux.

Autrement la contrainte est dite **respectée**.

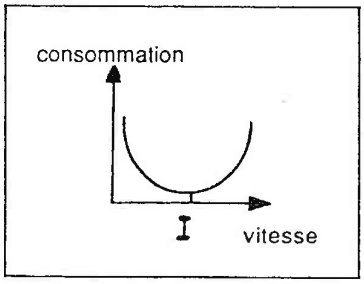
□

4.1 Exemples

Pour fixer les idées, écrivons quelques exemples typiques de contraintes:

- par la mise en relation des valeurs des variables d'état de façon monotone telle que:
  in $\gg(pression) \Longrightarrow$ in $\ll(debit)$;
  cette contrainte exprime la règle suivante: *si la pression croît alors le débit doit décroître.* La contrainte suivante exprime à son tour la règle: *si la vitesse est non nulle alors l'écart entre la tension précédente et actuelle doit être inférieure à 20.*
  in $>(vitesse, 0) \Longrightarrow$ in $<(-(@(tension), tension), 20)$;

- par la mise en relation des variables de façon non monotone mais selon une fonction concave ou convexe comme par exemple:



(1) $\underline{\text{in}} < (v, I) \gg (v) \Longrightarrow \underline{\text{in}} \ll (c)$
(2) $\underline{\text{in}} \leq (v, I) \gg (v) \Longrightarrow \underline{\text{in}} \gg (c)$
où la première contrainte exprime la règle: *si la vitesse v croît et qu'elle est inférieure à la constante I alors la consommation c doit décroître.* la seconde contrainte exprime à son tour *si la vitesse v décroît et qu'elle est superieure à la constante I alors la consommation c doit croître.*

- on peut aussi exprimer des contraintes qui mettent en relation les messages et les variables d'état comme par exemple:

  $\underline{\text{on}}\ \#(\textit{consigne-plus-debit})\ \#(\textit{consigne-moins-debit}) \Longrightarrow \underline{\text{in}} \approx (\textit{debit})$

  cette contrainte exprime la règle: *une consigne d'augmentation du débit et une autre de diminution impliquent sa stabilisation.*

- en utilisant des opérateurs $true$ et $false$, on peut exprimer des contraintes qui doivent être respectées de manière permanente comme par exemple: $true \Longrightarrow \underline{\text{in}} < (ve, 5)$ où inversement comme: $\underline{\text{on}}\ \#(\text{m}) \Longrightarrow false$;

4.2 Le concept de loi

Pour hiérarchiser les contraintes, nous procédons de la même manière que pour les unités de comportement. Nous étiquetons les contraintes en paquets que nous appellerons *lois.*

Définition 7
Une loi est un ensemble de contraintes ayant toutes la même étiquette. Par convention, cette étiquette sera appelée l'identificateur de la loi.

□

Définition 8
Soit L une loi formée de l'ensemble $C = \{c_1, \cdots, c_n\}$ de contraintes. La loi L est dite vérifiée s'il existe au moins une contrainte $c_i \in C$ telle que c_i soit respectée.

□

Intuitivement, cela signifie qu'une loi est une disjonction de contraintes.

Définition 9
Un composant est considéré en état de fonctionnement correct si chacune de ses lois est vérifiée. Autrement il est dit en état anormal.

□

A ce stade, il ne s'agit pas de déterminer s'il est réellement en état anormal ou non. C'est au mécanisme de diagnostic de décider s'il en est ainsi et déterminer les causes. Au

chapitre 3, consacré à l'élaboration du diagnostic, nous définissons plus précisément les notions de "fonctionnement correct" et "anormal" qui sont liées aux concepts de "panne" et de "symptôme". A présent, en utilisant la notation BNF, écrivons la syntaxe concrète pour spécifier les lois d'un composant:

<*lois*> ::= begin <*contraintes*> end
<*contraintes*> ::= <*contrainte*> { <*contrainte*> }*
<*contrainte*> ::= LABEL ':'
[on <$exp_{\{pe\}}$>] [in <$exp_{\{ve\}}$>] [with <$exp_{\{m\}}$>] $\Longrightarrow$
[on <$exp_{\{pe\}}$>] [in <$exp_{\{ve\}}$>] [with <$exp_{\{m\}}$>]

Figure 1.16: syntaxe concrète pour la spécification des lois.

4.3 Exemples

A titre d'illustration, nous donnerons les spécifications des lois du $\mathcal{CA}$ "pompe".

```
(1) pompe(lubrification) ;
    % déclaration des différentes lois suivies d'un entier
    % dénotant leurs priorités.
(2) loi1 :  3 ;
(3) loi2 :  2 ;
(4) loi3 :  4 ;
(5) loi4 :  1 ;
    begin
    % Commentaire sur la loi:  contraindre globalement la puissance
(6) loi1:  true ⟹ in ≤(puissance,1800) ≥(puissance,1000);
    % Commentaire sur la loi:  contraindre globalement la pression
(7) loi2:  true ⟹ in ≤(pression,100) ≥(pression,15) ;
    % Commentaire sur la loi:  contraindre l'évolution du débit
(4) loi3:  in ≫(pression) ≈(puissance) ⟹ in ≪(débit);
(5) loi3:  in ≪(pression) ≈(puissance) ⟹ in ≫(débit);
    % Commentaire sur la loi:  contraindre l'état des vannes
(6) loi4:  in >(débit,0) ⟹ in =(vanne{in},ouverte)
           =(vanne{out},ouverte);
      ⋮
    end
```

Figure 1.17: spécification des lois du $\mathcal{CA}$ "pompe".

Intuitivement, les priorités associées aux lois fixent au mécanisme de diagnostic le degré d'urgence de chaque loi. Avec les stratégies, qui seront décrites au chapitre 3, les priorités constituent les paramètres de contrôle du mécanisme de diagnostic. Nous définirons ces concepts au chapitre 3 et nous en montrerons également l'exploitation.

```
(1) analyseur-O2(analyseurs-gaz) ;
(2) loi1 :  6 ;
(3) loi2 :  1 ;
(4) loi3 :  4 ;
      ⋮
    begin
    % Commentaire sur la loi:  contraindre la teneur O2 aux points haut et bas
(5) loi1 :  on #(teneur-O2-point-haut,teneur-O2-point-bas)
            with ≫(teneur-O2-point-haut) ⟹ with ≫(teneur-O2-point-bas);
(6) loi1 :  on #(teneur-O2-point-haut,teneur-O2-point-bas)
            with ≈(teneur-O2-point-haut) ⟹ with ≈(teneur-O2-point-bas);
    % Commentaire sur la loi:  contraindre le relais O2
(7) loi2 :  on #(entrée-air,signal-analyse)
            in =(position-relais-taquet,haut) ⟹ false;
    % Commentaire sur la loi:  contraindre la teneur O2
(8) loi3 :  on #(teneur-O2-point-haut,teneur-O2-point-bas) ⟹
            in ≈(moyenne,@(moyenne))
            ⋮
    end
```

Figure 1.18: spécification des lois du $\mathcal{CA}$ "analyseur-O2".

5 Spécification de l'environnement

Jusqu'à présent, nous avons fait abstraction de l'environnement de l'installation dans lequel les composants interagissent. Cet environnement peut inclure, par exemple, le mode d'exploitation de l'installation (i.e production au ralenti, normale, etc), le type de produit fabriqué, la pression atmosphérique, etc. Au delà de son influence sur le fonctionnement de toute l'installation, l'environnement peut conditionner le mécanisme de diagnostic. C'est ainsi, par exemple, que le fait que la pression atmosphérique soit faible peut expliquer le dysfonctionnement de certains composants. De même, la marche au ralenti d'une installation peut permettre d'acquérir des informations qu'il serait dangereux (haute température, ...) d'obtenir en cas de marche normale de l'installation.

Dans la plupart des systèmes de diagnostic, le concept d'environnement, appelé aussi contexte [Clema 86], [Khanna 86], [Ricard 86], ne possède pas une spécification précise. Les informations qu'englobe ce concept se trouvent souvent dispersées et "mal structurées", ce qui entraine à la fois une confusion quant à l'interaction entre l'environnement et l'installation et un manque de structuration des données englobées dans ce concept. Pour remédier à ces problèmes, nous sommes conduits à considérer le concept d'environnement comme un composant, virtuel, qui est défini de la même manière que les autres composants. Au cas où l'installation est partitionnée en plusieurs sous-systèmes, on construit un

environnement pour chacun d'eux.

Dans un but d'illustration, donnons la spécification de l'environnement associé à l'installation du haut-fourneau dont la hiérarchie principale a été présentée à la figure 1.4.

```
(1) environnement-h-f(installation-haut-fourneau);
(2) synonym:  contexte-du-haut-fourneau;
(3) belongs:  hiérarchie-mère;
(4) type:  virtual;
    STATE
(5) mode-de-marche:  available restricted {ralenti,normal} ;
(6) pression-atmospherique:  available;
(7) equipe:  asked restricted {jour,nuit};
    LOCAL
(8) message-hp:  local init {"pression-trop-haute"};
(9) message-bp:  local init {"pression-trop-basse"};
    INPUT
(10)chgt-pression:  detected;
    OUTPUT
(11)opérateur:  undetected;
    begin
(12)loi1:  on #(chgt-pression) in ≫(pression-atmospherique)
    =(mode-de-marche,ralenti) ⟹ true;
(13)loi1:  on #(chgt-pression) in ≫(pression-atmospherique)
    =(equipe,nuit) ⟹ true;
(14)loi1:  on #(chgt-pression) in ≫(pression-atmospherique)
    =(equipe,jour) ⟹ false;
(15)     ⋮
    end
```

Figure 1.19: exemple d'une spécification de l'environnement.

De manière analogue à toute autre spécification, on retrouve ici les informations générales sur ce composant (i.e ses synonymes, son type qui est virtuel, etc), ses variables d'état, ses variables intermédiaires (Cf. spécification de la figure 1.12), ses ports d'entrée et de sortie, etc. Intuitivement la loi donnée en exemple pour ce composant indique: ***en cas de changement et de croissance de la pression atmosphérique alors si le mode de marche du haut-fourneau est au ralenti ou que c'est le régime "équipe de nuit" alors ignorer ce changement sinon (i.e regime "équipe de jour") alors considérer que la loi identifiée loi$_1$*** *est non vérifiée.* Au chapitre 3, nous verrons l'exploitation de cette information (i.e la non-vérification de cette loi) par le mécanisme de diagnostic.

6 Spécification de l'interface

C'est l'aspect technique qui va être traité ici. Plus précisément, il s'agit de spécifier pour chaque composant:

- les règles d'acquisition des données externes comme, par exemple, les grandeurs physiques en provenance des capteurs de température, de pression, de la teneur-O2, etc. Ces règles concernent donc les ports de la classe "detected" ainsi que les variables d'état de la classe "available" et "asked".
- les règles associées aux opérateurs génériques (e.g $\ll$, $\gg$, etc). Dans ce cas, ce sont les variables d'état de la classe "deduced" et qui sont utilisées dans la spécification des lois qui sont concernées.

Pour illustrer la spécification de l'interface, considérons tout d'abord la spécification de l'interface du composant "pompe". Nous commençons tout d'abord par spécifier les règles associées aux opérateurs génériques des variables d'état de la classe "deduced" (lignes 3 et 4) où l'on a exprimé la croissance (resp. la décroissance du débit en fonction du la puissance et de la pression. Ensuite (ligne 5–10) nous spécifions, pour les variables d'état de la classe "available" l'unité logique d'acquisition des données externes. De manière similaire, (lignes 11–14) on traite les différents ports: pour ceux qui sont classés "detected". Enfin, on spécifie les règles associées aux opérateurs génériques des variables d'état de la classe "detected" (lignes 15–21). Rappelons que l'opérateur @(v) renvoie l'ancienne valeur de la variable v.

```
(1) pompe(lubrification);
    begin
    STATE
(2) deduced
(3) ≫(débit):- ≈(puissance) ≪(pression);
(4) ≪((débit):- ≈(puissance) ≫(pression);
(5) available
(6) puissance:  assign historique-mesure;
(7) pression:  assign historique-mesure;
(8) vitesse:  function vit(puissance,pression);
(9) vanne{in}:  assign historique-état;
(10)statut:  assign historique-état;
    PORTS
(11)detected
(12)cde-marche-arret:  assign historique-commande;
(13)consigne-plus-débit:  assign historique-commande;
(14)consigne-moins-débit:  assign historique-commande;
    RULES
(15)rules of ≫ ;
(16)≫(pression) :- >(-(pression,@(pression)),5);
(17)rules of ≪ ;
(18)≪(pression) :- ≥(-(@(pression),pression),5);
(19)≪(pression) :- =(pression,0);
(20)rules of ≈ ;
(21)≈(puissance) :- <(-(puissance,@(puissance)),150);
    end
```

Figure 1.20: spécification de l'interface du $\mathcal{CA}$ "pompé".

Considérons maintenant le composant, virtuel, "environnement-h-f" introduit au paragraphe précédent. De la même manière que pour les autres composants, nous spécifions aussi l'interface:

```
(1) environnement-h-f(installation-haut-fourneau);
    STATE
(2) available
(3) mode-de-marche:  assign historique-état;
(4) pression-atmospherique:  assign historique-état;
(5) asked
(6) equipe:  assign operator-console;
    PORTS
(7) detected
(8) chgt-pression:  assign historique-commande;
    RULES
(9) :
(10):
    GRAPHS
    % On décrit le graphe construit selon la relation "adjacence-physique"
(11)identification:  graphe1;
(12)relation:  adjacence-physique;
(13)properties:  symetry,transitive;
(14)components:  {moteur,régulateur,bloc-isolation,vanne{in}};
(15)definition:
(16)adjacent(moteur,régulateur);
(17)adjacent(régulateur,bloc-isolation);
(18)adjacent(bloc-isolation,vanne{in});

    % On décrit le graphe construit selon la relation "adjacence-électrique"
(19)identification:  graphe2;
(20)relation:  adjacence-électrique;
(21)properties:  symetry;
(22)components:  {moteur,régulateur,bloc-isolation,vanne{in},vanne{out}};
(23)definition:
(24)adjacent(moteur,régulateur);
(25)adjacent(régulateur,vanne{out});
(26)adjacent(régulateur,vanne{in});
(27)adjacent(régulateur,bloc-isolation);
(28)adjacent(moteur,bloc-isolation);
(29):
    % On décrit les liens logiques-physiques d'acquisition des données.
    LINKS
(30)historique-mesure linked unité-db1;
(31)operator-console linked tty-device;
(32)historique-état linked unité-db2;
    end
```

Figure 1.21: spécification de l'interface du composant "environnement-h-f".

Outre les règles associées aux variables d'état et aux ports (lignes 2–10), ce sont les différents graphes de l'installation (lignes 11–29) ainsi que les liens entre les unités logiques et physiques qui sont spécifiés au niveau de l'interface de ce composant.

7 En résumé

Abstraction et *structuration* telles sont les principales particularités de la méthode de spécification présentée dans ce chapitre. Elle comprend trois étapes:

(1) décomposer l'installation en une hiérarchie principale de composants à l'aide de la relation de base "utilise". Cette hiérarchie donne ainsi une vue logique de l'installation. Elle constitue aussi un guide méthodologique pour spécifier l'ensemble des composants.

(2) construire des hiérachies secondaires et des graphes de composants pour donner, selon la relation utilisée, une vue physique, électrique, géographique, ... de l'installation.

(3) spécifier chaque composant en exprimant:

- sa *structure* en terme de variables d'état, de ports d'entrée et de sortie. A ce niveau, le but est de donner une "bonne" représentation du composant.
- son *comportement* en décrivant les relations entre les variables d'état et les différents ports. A ce niveau, le but est de comprendre le fonctionnement du composant et de faciliter l'identification de ses lois.
- ses *lois* en termes de contraintes qui portent sur l'évolution des variables d'état et des ports du composant. A ce niveau, le but est de régir le comportement du composant en capturant les contraintes sur son fonctionnement normal.
- son *interface* pour définir d'une part le mode d'acquisition des données, d'autre part les règles associées aux opérateurs génériques. A ce niveau, le but est de regrouper et d'isoler les aspects techniques susceptibles d'évoluer car ils sont liés à une implantation particulière.

L'environnement de l'installation est à son tour un composant et donc spécifié en tant que tel. Nous qualifierons ces spécifications de *partielles* dans le sens où elles sont limitées à la description de l'installation. Notre objectif, à présent, est de résoudre le problème du diagnostic en spécifiant les stratégies, les tactiques et les paradigmes nécessaires au mécanisme de diagnostic.

Avant d'aborder ces problèmes nous présentons une synthèse des techniques connues de diagnostic.

Chapitre 2

Synthèse des techniques du diagnostic industriel

Jusqu'à présent, la méthode exposée au chapitre précédent nous a permis de:

(1) *construire la hiérarchie principale donnant une vue logique de l'installation, les hiérarchies secondaires et les graphes qui donnent quant à eux d'autres vues de l'installation.*

(2) *spécifier l'ensemble des composants qui appartiennent à la hiérarchie principale.*

Mais comme nous l'avons souligné auparavant ces spécifications sont purement descriptives et ne permettent pas de résoudre le problème de diagnostic. Aussi notre préoccupation est-elle de proposer une méthode de diagnostic et de spécifier les stratégies, les tactiques et les paradigmes nécessaires pour mettre en oeuvre le mécanisme du diagnostic de la méthode proposée. C'est pourquoi nous allons étudier, dans un premier temps, les techniques de diagnostic exposées dans une littérature abondante. De cette étude, nous dégageons les mécanismes de base et négligeons un certain nombre de notions secondaires et conjoncturelles. Ceci nous conduit à partitionner ces techniques en 4 grandes classes.

L'objet de ce chapitre est alors de présenter chaque classe en donnant son principe de base, ses avantages et ses limites. Nous proposons également des solutions à ces restrictions.

Ainsi au paragraphe 1, nous présenterons la classe des techniques **intégrées** *qui, sans doute, est la plus répandue actuellement. La classe des techniques* **empiriques** *qui a trait aux systèmes experts sera exposée au §2. Les techniques* **"par modèle"** *feront l'objet du §3. Au §4, ce sont les techniques issues de la* **physique qualitative** *que nous présenterons. Enfin nos conclusions occuperont le paragraphe 5.*

1 La technique intégrée

Cette technique, utilisée notamment par les automaticiens, a suscité plusieurs travaux de recherche [Ayache 80], [Descotes 84], [Gondran 79], [Lievens 76], [Marin 76], [Thelliez 80]. De nombreuses variantes de cette technique ont été présentées. Citons la procédure d'analyse de signature [Pau 75], la procédure d'analyse des données [Pau 76], la procédure du filtre [Alanche 86]. Il ne s'agit pas de décrire ici en détail ces procédures, mais de donner succinctement le principe de base de cette technique.

1.1 Schéma de synthèse

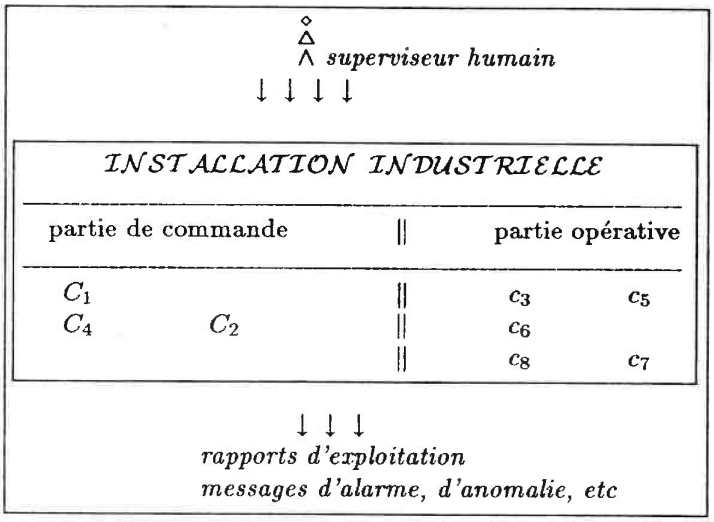


Figure 2.1: système conventionnel de diagnostic.

où

- les C_i dénotent les composants, principalement des automates, appartenant à la partie de commande du procédé. C'est cette partie qui assure la conduite du procédé industriel.

- les c_j dénotent les composants tels que les capteurs, les vannes, les bascules, etc faisant partie de la partie opérative. Le rôle de cette partie se résume à la réception des commandes, en provenance de la partie de commande, et à leur exécution.

1.2 Principe

Schématiquement, à l'étape de conception[1] de la partie de contrôle, il s'agit:

[1]Cette étape est aussi appelée étape d'analyse fonctionnelle.

- d'établir une table de décision dont les entrées sont les défauts élémentaires susceptibles de survenir dans la partie opérative et les actions correspondent à un traitement élémentaire qui doit être éxécuté par la partie de commande.

et à l'étape de réalisation, on est conduit:

- à programmer les deux tâches à savoir la détection des différents défauts et l'exécution des actions correspondantes. Ces deux tâches sont prises en charge et assurées par la partie de commande.

Exemples

Défauts élémentaires: perte de tension, rotor bloqué, bascule hors-service, manque d'air comprimé.

Actions associées: arrêt d'urgence, temporiser l'envoi du chargement, imprimer le message "Manque air comprimé".

1.3 Limites

Remarquons tout d'abord, que suivant cette technique, la partie de commande assure deux fonctions: la conduite du process mais aussi la détection et l'exécution des actions associées ce qui justifie le qualificatif *intégrée.* Ce cumul des fonctions implique un traitement nécessairement élémentaire des défauts conduisant ainsi à:

- une exclusion de toute analyse élaborée[2] des défauts visant à faire face au phénomène "de symptômes et de pannes en cascade",
- une impossibilité de filtrer ainsi l'ensemble des messages à délivrer.

Remarquons enfin que la mise en oeuvre de cette technique ne peut être réalisée pour une installation existante, mais au contraire, elle doit être conduite en même temps que la conception et la réalisation de toute l'installation.

1.4 Conclusions

Bien que cette technique joue un rôle important au niveau sécurité et sûreté de fonctionnement de l'installation, son intérêt au niveau du diagnostic est plutôt limité. En effet en pratique on est conduit à se contenter d'une aide au diagnostic. Cette aide consiste à fournir des indications servant à déceler les anomalies de fonctionnement et éventuellement à localiser les composants en défaut.

[2] Au contraire, un traitement rigoureux s'accompagne inévitablement d'une lourdeur de mise en oeuvre parfois insurmontable.

2 La technique empirique

Contrairement à la technique intégrée, dans ce cas, on suppose déjà l'existence d'une installation; le système de diagnostic vient alors s'adjoindre au pilotage de l'installation. Les systèmes de diagnostic qui font appel à cette technique, appelés aussi des systèmes experts, sont basés sur des connaissances empiriques et utilisent un mécanisme de raisonnement qui est dit de surface: *"shallow reasoning"* [Richardson 85], [Xiang 86].

2.1 Schéma de synthèse

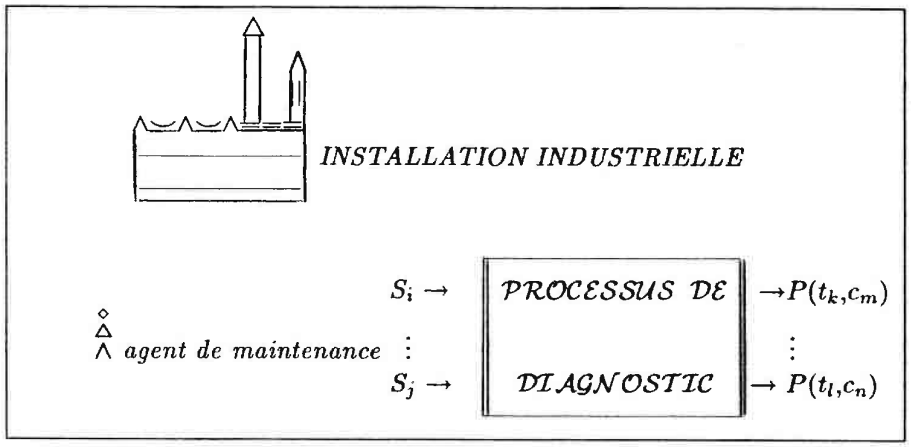


Figure 2.2: $1^{ère}$ génération "le raisonnement de surface".

où

- S_i désigne un symptôme donné, tel une observation de l'opérateur comme "l'afficheur clignoté' ou encore un message délivré: "le défaut 200 est signalé par l'automate de mesuré'.

- $P(t_j,c_k)$ désigne le "type" de panne t_j du composant c_k.
  Par exemple: panne(*sensibilité*,*capteur-de-base*), panne(*blocage*,*bascule*), etc.

2.2 Principe

Supposons que les ensembles $\mathcal{S}$ et $\mathcal{P}$ sont définis en extension et dénotent respectivement l'ensemble des symptômes et celui des pannes. On peut alors schématiser le mécanisme de diagnostic comme suit [ElAyeb 85], [Xiang 86]:

Partant d'un ensemble de symptômes donnés par l'utilisateur (i.e., $\{S_i, \cdots, S_j\} \subset \mathcal{S}$) et en utilisant des relations de cause à effet entre pannes et symptômes, le processus de diagnostic tente de construire une chaîne d'inférences filtrant progressivement les éléments

de $\mathcal{P}$. En cas de succès, ce processus résultera sur le sous-ensemble des pannes (i.e., $\{P_{k,m}, \cdots, P_{l,n}\} \subset \mathcal{P}$) qui ont la plus forte association avec les symptômes donnés au départ.

Des nombreux systèmes expérimentaux ou opérationnels font appel à cette technique. Citons DANTES [Mathonet 87], DIAMON [Skatteboe 86], EXTASE [Jakob 84], [Ancelin 87] EXTRA etc. Outre l'aspect méthodologique, abordé ultérieurement, qui peut être utilisé comme le critère principal pour une classification de ces systèmes, nous donnons tout d'abord les critères secondaires suivants:

- la capacité d'expression du langage du système pour décrire les pannes, les symptômes et les relations de cause à effet. Dans la plupart des cas les pannes et les symptômes sont souvent exprimés par des "frames" ou par des formules atomiques alors que les relations sont exprimées par des règles de production (i.e si ... alors ...) ou par des clauses de Horn [Fink 86], [Xiang 86].

- la capacité du processus de diagnostic à traiter le caractère incertain des données mais aussi à inclure plusieurs stratégies de raisonnement [Khanna 86].

2.3 Limites

- la difficulté que l'on peut avoir pour effectuer l'énumération des symptômes et des pannes ainsi que la dérivation de relations empiriques de cause à effet entre pannes et symptômes [Davis 84],

- les concepts de symptôme, de panne, d'indice de panne, etc ... utilisés par cette technique ont généralement une signification différente selon les installations, voire selon les dépanneurs. On assiste alors à une prolifération des définitions informelles de ces concepts [ElAyeb 89].

Signalons enfin les deux propriétés suivantes qui sont inhérentes à ce type de système:

- la propriété "ad-hoc", il est difficile d'adapter un système construit pour une installation donnée à une autre similaire mais qui physiquement est légèrement différente [Xiang 86],

- la propriété "d'incomplétude", seules les pannes prévues et codées à l'avance dans le système peuvent être diagnostiquées [Davis 84]

2.4 Conclusions

Malgré ces limites des nombreux systèmes opérationnels ou expérimentaux font appel à cette technique. Ce succès pourrait être dû aux raisons suivantes:

- l'étape de réalisation peut être relativement rapide. En effet la difficulté principale réside dans l'étape d'énumération des symptômes, des pannes et des relations de cause à effet.

- moyennant l'effort fourni à l'étape de l'énumération, la capacité des tels systèmes pour diagnostiquer les pannes *classiques* est en général satisfaisante.

Nous pensons qu'il est possible d'apporter plusieurs améliorations substantielles à cette technique, en particulier:

- suivant l'aspect de modélisation. Il s'agit d'exprimer de manière concise et explicite les différents types de connaissances (i.e descriptives, stratégiques, etc) de l'installation. C'est ce que nous avons commencé à faire au cours du premier chapitre et que nous continuerons à faire au chapitre suivant.

- suivant l'aspect méthodologique. Il s'agit de proposer une méthode permettant l'énumération systématique des pannes. Cet aspect qui constitue notre préoccupation majeure dans ce travail sera particulièrement développé aux paragraphes réservés aux paradigmes (i.e règles empiriques) et au mécanisme de justification du chapitre suivant. Nous donnerons des guides d'extraction de ces paradigmes.

- suivant l'aspect d'apprentissage. Il s'agit de proposer et de greffer un mécanisme d'apprentissage au système de diagnostic permettant l'adjonction des "nouvelles" pannes imprévues initialement. C'est ce qui sera évoqué dans le dernier chapitre.

3 La technique par modèle

Cette technique est souvent appelée: "diagnosis based on structure and function" ou encore "diagnosis from first principles". Le mécanisme de raisonnement est qualifié de profond *"deep reasoning"*. Il utilise des connaissances "modélisées" par opposition aux connaissances empiriques [Chandrasekaran 83], [Davis 84], [Fink 86], [Lepetit 87], [Xiang 86]. Introduite initialement par Nau [Nau 83] et reprise par Chandrasekaran et Davis [Chandrasekaran 83,Davis 84], cette technique a fait récemment l'objet d'une étude théorique par Reiter [Reiter 87] et par De Kleer [DeKleer 87].

3.1 Principe

Contrairement à la technique empirique qui se base sur des relations de cause à effet, cette technique exige la spécification du comportement normal des composants. Dès lors les concepts de panne, de symptôme et celui des relations de cause à effet disparaissent. En effet, suivant cette technique, diagnostiquer un composant pour savoir s'il est en panne

ou non revient à "comparer" le comportement observé ou réel du composant considéré avec son comportement spécifié.

3.2 Technique de relaxation de Davis

La technique de Davis [Davis 84] se décompose en deux étapes:

(1) décrire la structure des composants,

(2) spécifier le comportement des composants.

DESCRIPTION DE LA STRUCTURE

A cet effet, Davis introduit trois concepts: celui de module, de port et de terminal. Par exemple, considérons un composant tel qu'un additionneur; on a alors:

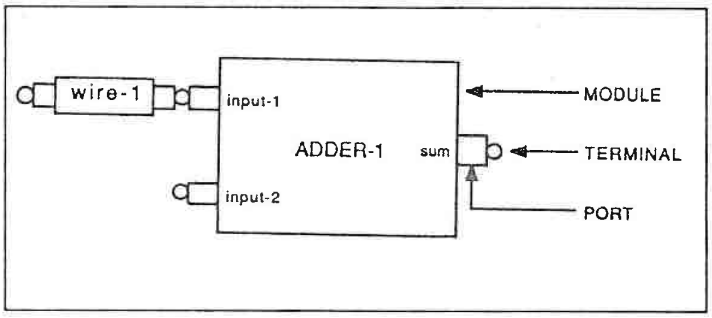


Figure 2.3: description d'un additionneur

SPÉCIFICATION DU COMPORTEMENT

Deux types de règles: celles qui simulent le comportement normal du composant, appelées règles de simulation, et celles appelées règles d'inférence. Reprenons l'exemple de l'additionneur. On a:

une règle de simulation:

to get *sum* from ($input_1$ $input_2$) do (+ $input_1$ $input_2$)

et deux règles d'inférence:

to get $input_1$ from (*sum* $input_2$) do (- *sum* $input_2$)

to get $input_2$ from (*sum* $input_1$) do (- *sum* $input_1$)

MISE EN OEUVRE SUR UN EXEMPLE:

Voyons à présent le déroulement du processus du diagnostic. A cet effet, considérons l'exemple suivant:

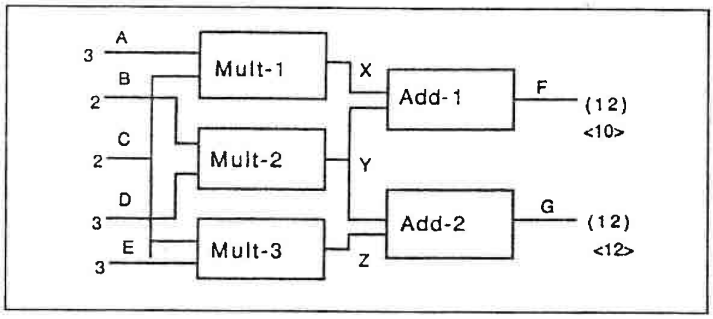


Figure 2.4: description d'un additionneur-multiplicateur

où les valeurs mises entre parenthèses sont celles attendues (i.e. celles qui doivent être données par un comportement normal) et celles entre chevrons sont les valeurs réelles ou observées. Il s'agit donc de mettre en oeuvre un algorithme qui détermine les composants en panne. Intuitivement, l'algorithme proposé par Davis se base sur les principes suivants:

-1- considérer l'ensemble des composants comme un réseau de contraintes. En utilisant les données initiales et la spécification comportementale de chaque composant, générer les valeurs attendues.

-2- mettre en contradiction les valeurs observées et celles attendues. Toute anomalie de fonctionnement doit alors se traduire par une inconsistance du réseau.

-3- la détermination du composant ayant conduit à l'inconsistance se passe comme suit:

(a) sélectionner un composant et suspendre ses contraintes, i.e. les règles de simulation.

(b) si le réseau atteint un état consistant alors retenir ce composant comme candidat potentiel sinon ignorer ce composant et en choisir un autre.

Illustrons de manière détaillée l'algorithme proposé par Davis sur l'exemple de l'additionneur-multiplicateur de la figure précédente.

(1) RASSEMBLER LES DIVERGENCES:

1.1 Insérer les entrées du composant dans le réseau des contraintes.

% e.g. insérer 3, 2, 2, 3 et 3 dans les entrées primaires A ... E.

% la simulation donnera alors les valeurs, mises entre

% parenthèses, aux points F et G.

(2) DETERMINATION DES CANDIDATS POTENTIELS VIA LES DEPENDANCES:

2.1 Pour chaque divergence détectée à l'étape 1:

- via les dépendances trouver tous les composants ayant contribué à fournir la valeur attendue.

% e.g. à partir de F, on trouve *adder-1*, *mult-1* et *mult-2* comme

% composants en aval ayant contribué à donner cette valeur.

2.2 Prendre l'intersection de tous les ensembles obtenus à l'étape 2.1.

% dans l'exemple ci-dessus il y a une seule divergence et donc un seul

% ensemble.

(3) DETERMINATION DES CANDIDATS VIA LA SUSPENSION DES CONTRAINTES:

3.1 Pour chaque composant trouvé à l'étape 2.2.

3.1.1 suspendre les contraintes modélisant son comportement,

3.1.2 insérer les valeurs observées comme valeurs-résultat dans le réseau de contraintes,

% les valeurs en entrée sont déjà inserées à l'étape 1.1.

3.1.3 si le réseau atteint un état consistant

- le composant est alors un candidat potentiel,
- prélever les valeurs en entrée du composant,
- ajouter ce candidat et ces valeurs en entrée à la liste des candidats.

% e.g. ajouter adder-1 et ses valeurs 6, 6 et 10.

sinon le composant n'est pas un candidat potentiel, l'ignorer

% e.g. mult-2,

3.1.4 retirer les valeurs au réseau des contraintes,

3.1.5 remettre les contraintes qui ont été suspendues à l'étape 3.1.1.

Figure 2.5: algorithme de Davis.

3.3 Limites

- il est difficile de décrire de manière précise la fonction de chaque composant [David 86]. En effet, si, dans certains cas, la fonction d'un composant électronique tel qu'un additionneur par exemple est évidente, il n'en sera pas de même pour un composant tel qu'une imprimante ou un "talkie-walkie".
- cette technique est dans l'incapacité de diagnostiquer les pannes globales [David 86], [ElAyeb 85]. Autrement dit si, on prend isolément chaque composant, il fonctionne correctement (i.e on ne peut remarquer aucune contradiction entre le comportement observé et celui attendu) alors que la mise en interaction de tous les composants conduit à une anomalie de fonctionnement. Ce cas est très fréquent dès qu'il s'agit des composants ayant pour tâche la propagation des signaux par exemple.
- enfin la possibilité d'aboutir à un diagnostic erroné [ElAyeb 88c]. En effet, l'exactitude des résultats fournis par l'algorithme de Davis est sous-jacente à l'hypothèse suivante: *toute communication entre composants est explicitement spécifiée.* Malheureusement cette hypothèse, difficile à vérifier, peut parfois s'avérer fausse. Par exemple, dans l'additionneur-multiplicateur de la figure 2.4, seule la communication électrique est spécifiée alors qu'il est possible d'avoir une communication ou une interférence du type électro-magnétique entre composants.

3.4 Technique de Reiter

S'inspirant des travaux de Davis, Reiter [Reiter 87] propose une autre technique de diagnostic qui s'appuie sur la logique. Illustrons cette technique sur l'exemple suivant:

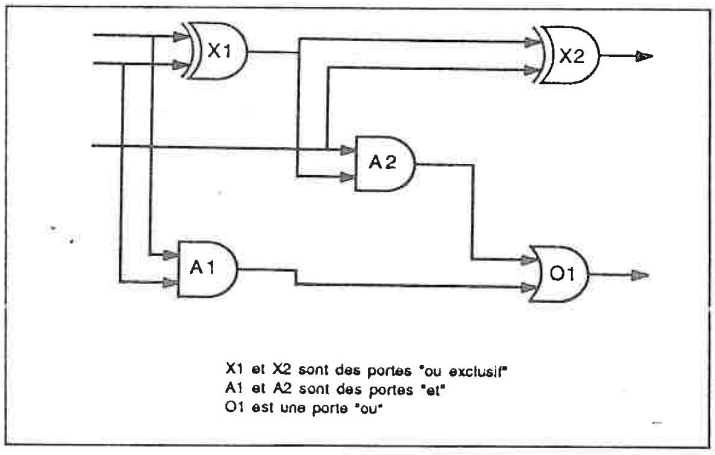


Figure 2.6: schéma d'un additionneur complet.

Reiter définit tout d'abord la notion de système comme une paire $(DS, COMP)$ où

- DS est une description du système, i.e un ensemble de formules du premier ordre,

- $COMP$ est l'ensemble des composants du système, i.e un ensemble fini de constantes.

Exemple: la figure donnée ci-dessus décrit un additionneur complet avec les différents types de portes. Considérons que ce circuit forme un système de Reiter. On a alors:

$COMP = \{A_1, A_2, X_1, X_2, O_1\}$

En se donnant un prédicat unaire AB qui signifie "anormal" ainsi que les prédicats unaire $ANDG(x), XORG(x)$ et $ORG(x)$ qui signifient respectivement que x est une porte "*et*", "*ouexclusif*" et "*ou*", la description du système DS est alors:

$$(A): \begin{cases} ANDG(x) \wedge \neg AB(x) \Rightarrow out(x) = and(in1(x), in2(x)), \\ XORG(x) \wedge \neg AB(x) \Rightarrow out(x) = xor(in1(x), in2(x)), \\ ORG(x) \wedge \neg AB(x) \Rightarrow out(x) = or(in1(x), in2(x)). \end{cases}$$

$$(B): \begin{cases} ANDG(A_1), \\ ANDG(A_2), \\ XORG(X_1), \\ XORG(X_2), \\ ORG(O_1) \end{cases}$$

$$(C): \begin{cases} out(X_1) = in2(A_2), \\ out(X_1) = in1(X_2), \\ out(A_2) = in1(O_1), \\ in1(A_2) = in2(X_2), \\ in1(X_1) = in1(A_1), \\ in2(X_1) = in2(A_1), \\ out(A_1) = in2(O_1). \end{cases}$$

Remarquons que les formules dénotées par (A) décrivent le comportement des portes "et', "ou exclusif" et celui des portes "ou". Ainsi la première formule peut être lue comme suit: si x est du type $ANDG$ et est "normal" alors sa "fonction" est de faire le "*and*" logique entre se deux entrées. Les formules atomiques dénotées (B) sont des assertions et les formules dénotées par (C) décrivent les connexions entre les composants. Ajoutons enfin les contraintes sur les entrées du système comme suit:

$$(D): \begin{cases} in1(X_1) = 0 \vee in1(X_1) = 1, \\ in2(X_1) = 0 \vee in2(X_1) = 1, \\ in1(A_1) = 0 \vee in2(A_1) = 1. \end{cases}$$

Définissons à présent la notion d'observation comme étant un triplet $(DS, COMP, OBS)$ pour un système $(DS, COMP)$, où OBS désigne un ensemble de formules du premier ordre. Considérons l'exemple de l'additionneur complet; une observation pourra être:

$$(E): \begin{cases} in1(X_1) = 1 \\ in2(X_2) = 0, \\ in1(A_1) = 1, \\ out(X_2) = 1, \\ out(O_1) = 0. \end{cases}$$

On peut alors formaliser aisément le concept de fonctionnement correct d'un ensemble de composants. En effet, supposons que le système $(DS, \{c_1, \cdots, c_n\})$ présente une anomalie de fonctionnement, c'est à dire qu'il existe une observation OBS qui est en contradiction avec la description du système DS. On peut alors écrire:

$DS \cup \{\neg AB(c_1), \cdots, \neg AB(c_n)\} \cup OBS$ est inconsistant

Il s'agit maintenant de définir la notion de diagnostic comme suit: un diagnostic pour un système $(DS, COMP, OBS)$ est un ensemble minimal $\Delta \subseteq COMP$ tel que:

$DS \cup OBS \cup \{AB(c)/c \in \Delta\} \cup \{\neg AB(c)/c \in COMP - \Delta\}$ est consistant.

En d'autres termes, un diagnostic est déterminé par le plus petit ensemble de composants ayant la propriété suivante:

> l'hypothèse que chaque composant est en défaut (anormal) avec la propriété que tous les autres composants fonctionnent correctement (non anormal) est consistante avec la description du système SD et l'observation OBS.

Le problème du diagnostic est ainsi ramené à rechercher un ensemble minimal de composants préservant la consistance du système. A l'exception de certains cas (i.e le calcul de propositions) où la consistance est décidable la plupart des cas de taille réelle se trouvent confrontés à un problème d'indécidabilité.

Poursuivant son étude, en supposant que les cas à traiter sont décidables, Reiter propose un algorithme qui calcule de manière optimale l'ensemble Δ.

3.5 Limites

Elles se situent:

- sur le plan de la complexité et de l'efficacité [ElAyeb 88a]. En effet le nombre de formules qui ont été nécessaires pour spécifier l'exemple de l'additionneur est assez instructif. On pourra alors difficilement envisager de traiter, de manière efficiente, des applications en vraie grandeur.

- sur le plan de la faisabilité [ElAyeb 88a]. En effet, la satisfiabilité d'un ensemble de formules du calcul des prédicats est un problème indécidable.

- on retrouve enfin la limite relative aux pannes globales.

3.6 Conclusions

L'intérêt de cette technique est sans doute de proposer un cadre et une base théorique du diagnostic industriel où les différents concepts sont bien définis [Reiter 87] mais aussi de remédier à la quasi-totalité des limites de la technique empirique. Notons toutefois que les deux techniques ne traitent pas l'aspect méthode de conception et de réalisation des systèmes de diagnostic pour des applications en vraie grandeur (i.e les grandes installations).

Afin de remédier à l'ensemble de ces limites, voici nos propositions. Elles concernent les aspects suivants:

- l'aspect méthodologique.
  Il s'agit de disposer d'un guide, pour structurer l'installation en hiérarchies de composants, et d'une méthode de spécification. C'est ce que nous avons proposé au chapitre précédent.

- l'aspect de la richesse d'expression du langage de spécification.
  Ce langage doit permettre la description de différents types de composants. C'est ce que nous avons fait où nous avions proposé un langage simple et indépendant d'un type particulier des composants.

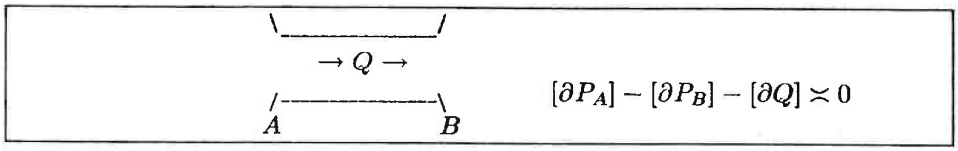
- l'aspect de l'efficacité du système de diagnostic.
  Le point majeur ici concerne le temps de réponse. C'est ce que nous traiterons au chapitre suivant où il s'agit d'exploiter la structure hiérarchique de l'installation pour faire une première localisation des composants potentiellement responsables de la divergence comportementale détectée.

4 La technique par la physique qualitative

Le but initial de cette technique est de comprendre, voire de calquer, le raisonnement d'un physicien. Plus précisement il s'agit de simuler de manière qualitative un phénomène physique qui se prête difficilement à une simulation classique par un calcul numérique. L'emploi du terme qualitatif, par opposition à quantitatif où l'on manipule davantage des quantités numériques, peut se justifier dans la mesure où la simulation qualitative s'appuie sur des ordres de grandeurs (i.e., le signe des grandeurs) ainsi que sur les influences entre ces ordres de grandeurs. Actuellement dans la littérature consacrée à ce sujet, on peut distinguer trois variantes: la technique de De Kleer basé sur le modèle des confluences [DeKleer 84], celle de Forbus introduisant la théorie des processus qualitatifs [Forbus 84], et la technique de Kuipers [Kuipers 86] qui se fonde sur la simulation qualitative. En guise d'illustration, nous synthétisons la technique de De Kleer.

4.1 La technique de De Kleer: le modèle de confluence

Cette technique est constituée de deux étapes: la modélisation et le raisonnement. L'étape de modélisation est basée sur le concept de *confluence* qui est une représentation d'une loi physique. Exemple: soit un tuyau (A,B) dans lequel s'écoule un fluide. Il est soumis à la confluence:



où $[v]$ est le signe de la grandeur physique v (i.e +, −, 0) , ∂v une variation de la quantité physique v, et P_A, P_B les pressions en A et B, Q le flux de A vers B.

La modélisation d'un appareil physique, appelé aussi système, consiste à:

- trouver les lois physiques auxquelles le système est soumis.
- exprimer qualitativement ces lois par des confluences.

L'étape de raisonnement exploite ces confluences, appelées aussi équations qualitatives, en vue de:

- simuler le comportement des composants du système pour générer des diagrammes de transition résumant l'évolution globale du système.
- connaître l'évolution immédiate ou instantanée du système.

Comme le fait remarquer Dormoy dans [Dormoy 86], dans le premier cas l'objectif est de comprendre "*comment ça marche*"; on veut connaître l'évolution du système à long terme (i.e jusqu'à ce que l'on atteigne un état d'équilibre). Au contraire, dans le second, on souhaite connaître l'évolution immédiate du système.

Dans [DeKleer 84], De Kleer propose un programme, nommé ENVISION, pour assurer cette étape de raisonnement. Schématiquement, ce programme consiste à manipuler les signes des variables et à les propager par la suite dans les équations qualitatives.

4.2 Sur un exemple: le régulateur de pression

On a un fluide qui s'écoule de `A` vers `E`, `PA` et `PE` sont les pressions des points `A` et `E`, `Q` le flux du fluide et `S` l'aire laissée par la soupape à l'écoulement du fluide.

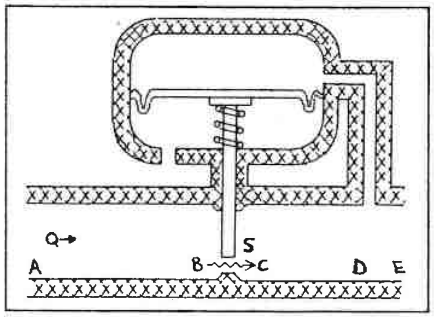


Figure 2.7: le régulateur de pression.

MODELISATION

Elle consiste à écrire le modèle qualitatif du régulateur de pression:

$$(MQ): \begin{cases} (1)\ [\partial P_A] - [\partial P_B] - [\partial Q] \asymp 0 \\ (2)\ [\partial P_B] - [\partial P_C] - [\partial Q] + [\partial S] \asymp 0 \\ (3)\ [\partial P_C] - [\partial P_D] - [\partial Q] \asymp 0 \\ (4)\ [\partial P_D] - [\partial P_E] - [\partial Q] \asymp 0 \\ (5)\ [\partial P_D] + [\partial S] \asymp 0 \end{cases}$$

Ces équations qualitatives (MQ) peuvent être déduites des lois d'équilibre formulées de manière classique ou encore à partir d'abaques établies de manière empirique.

RAISONNEMENT

Supposons une variation de la pression aux points A et B, on est alors conduit à ajouter à MQ les équations:

$$(IN): \begin{cases} (6)\ [\partial P_A] = x \\ (7)\ [\partial P_B] = y \end{cases}$$

Notre but, à présent, est de savoir les conséquences de telles variations sur l'état du système. Le "raisonneur" doit alors conclure:

$$(OUT): \begin{cases} (8)\ [\partial P_E] = - \\ (7)\ [\partial S] = - \end{cases}$$

c'est-à-dire que l'on constate que le piston s'abaisse et que la pression diminue.

4.3 Conclusions

Bien que la physique qualitative, introduite par De Kleer [DeKleer 84], suscite un intérêt croissant, les problèmes traités à ce jour sont très simples. On peut noter cependant certaines tentatives [Gentil 87], [Lepetit 87] visant l'exploitation de cette technique pour le diagnostic industriel. Il nous semble prématuré de tirer des conclusions immédiates; néanmoins, l'idée principale que l'on peut retenir ici est la manipulation des ordres de grandeur plutôt que des valeurs algébriques. Il reste à savoir si cela peut suffir réellement ou, à défaut, à chercher la classe des problèmes à traiter selon cette approche.

5 Conclusions

L'objectif de ce chapitre est de présenter brièvement les différentes techniques. Par notre synthèse, nous avons ainsi mis en évidence les principes de base, les limites et des solutions à ces restrictions. Notre méthode de diagnostic, détaillée au cours du chapitre suivant, s'inspire d'une part, de la technique empirique, d'autre part de celle par modèle. Plus précisément nous allons proposer une extension de la technique par modèle tout en faisant coopérer la technique empirique. Nous "héritons" ainsi de la puissance de la première technique ainsi que de l'efficacité (dans le sens rapidité d'exécution) de la technique empirique.

Quant à la technique intégrée, présentée succinctement ici, elle ne peut être incluse dans notre méthode. En effet, rappelons que cette technique doit être mise en oeuvre à l'étape de conception et de réalisation de toute l'installation.

Avant de clore ce chapitre, signalons qu'une présentation des systèmes opérationnels ou expérimentaux, construits suivant ces techniques, sera donnée à la fin du chapitre 4.

Chapitre 3

Spécification et élaboration du diagnostic

Si l'établissement d'un diagnostic est important, la méthode qui y a conduit l'est encore davantage. Notre travail, maintenant, consiste donc à s'interroger sur les qualités d'une méthode de spécification et d'élaboration de diagnostic avant d'étudier et de construire la nôtre. Celle-ci doit être facile *à utiliser,* simple *mais aussi* puissante. *La facilité de la méthode nécessite qu'elle soit basée sur des concepts clairs et précis. Elle nécessite aussi que l'on dispose d'un guide méthodologique pour écrire les spécifications du diagnostic. Il est important que ce guide ne provoque aucune rupture mais au contraire qu'il soit le prolongement "naturel" permettant de compléter les spécifications de description de l'installation. Quant à la puissance de la méthode, elle est liée au mécanisme du diagnostic qui doit être* efficace, flexible *et* fiable *mais surtout* indépendant *de toute installation particulière. La rapidité d'exécution et l'efficacité peuvent être obtenues en fondant un mécanisme, qui, exploitant la structure hiérarchique de l'installation, fait coopérer le raisonnement de surface et le raisonnement profond. La définition des stratégies et des tactiques de diagnostic comme des paramètres du mécanisme assure les qualités d'indépendance et de flexibilité. Enfin la fiabilité, au sens de l'exactitude des résultats du diagnostic, nécessite quant à elle de disposer d'un mécanisme de justification des résultats obtenus.*

Ce chapitre peut être considéré comme le pivot du travail présenté. Etudiant l'aspect méthodologique du diagnostic, il permet de compléter les spécifications de description de l'installation, écrites au chapitre 1, de définir très précisement les concepts de panne et de symptôme qui se trouvent souvent mal définis et de décrire ensuite le mécanisme de diagnostic.

Ainsi le premier paragraphe sera consacré à l'introduction de notre méthode de diagnostic. On y définira alors les **concepts** *de* panne *et celui de* symptôme. *Ensuite, aux paragraphes 2 et 3 ce sont les* **stratégies** *et les* **tactiques** *de diagnostic qui seront définies. Au paragraphe 4 nous présentons notre méthode de* **justification**, *les concepts de panne confirmée et infirmée seront alors introduits. Nous définirons ensuite les* **paradigmes** *(§5) qui constituent les données principales de la justification. Des exemples de spécifications de diagnostic seront alors présentés au §6. Quant au mécanisme de diagnostic, il occupera le §7. Nous exposerons alors notre principe pour faire* **coopérer** *les deux types de raisonnement: le raisonnement profond et celui de surface. Les avantages d'une telle coopération et un scénario typique du déroulement du mécanisme seront également donnés. Quant au dernier paragraphe, il sera réservé à un résumé du chapitre.*

1 Méthode de diagnostic

Dans la littérature du diagnostic de pannes et particulièrement dans celle consacrée aux systèmes experts en diagnostic, les concepts de panne et de symptôme sont souvent mal définis sinon ambigus. Notre but est ici de définir très précisément ces concepts.

A cet effet, considérons L une loi donnée et définissons alors le prédicat $V(L)$ évalué à *vrai* si la loi L est vérifiée, *faux* sinon. Selon les définitions 5, 6 et 7 du chapitre 1, on a:

$$V(L) = R(c_1) \vee \cdots \vee R(c_n)$$

où

- $c_1, \cdots, c_n$ sont les contraintes de la loi L,
- $R(c_i)$ est un prédicat évalué à *vrai* si la contrainte c_i est respectée, *faux* sinon.

A présent, soit $\mathcal{C}$ l'ensemble des composants appartenant à la hiérarchie principale d'une installation donnée et considérons C, un composant donné, appartenant à $\mathcal{C}$ et SP=<SS, SC, SL, SI> sa spécification:

- SS la spécification de sa structure
- SC la spécification de son comportement
- SL la spécification de ses lois $L_1, \cdots, L_m$
- SI la spécification de son interface

Supposons que les spécifications SC et SL sont correctes, ce qui signifie que SC définit parfaitement le comportement normal de C et que l'ensemble des lois SL régit entièrement son comportement. Définissons alors le prédicat unaire FC (i.e. Fonctionne Correctement) comme:

$$FC(C) = V(L_1) \wedge \cdots \wedge V(L_m)$$

nous avons:

$$\neg FC(C) = \neg V(L_1) \vee, \cdots, \vee \neg V(L_m)$$

exprimant ainsi qu'un composant n'est plus en fonctionnement correct dès qu'une de ses lois n'est plus vérifiée. Maintenant notons par $\mathcal{H}$ la hiérarchie principale et définissons le prédicat S:

$$S(\mathcal{H}) = \bigvee_{C \in \mathcal{C}} \neg FC(C)$$

Intuitivement, ce prédicat S (i.e symptôme) dénote le non-respect d'un nombre quelconque de lois d'un ou de plusieurs composants appartenant à la hiérarchie principale, c'est-à-dire une divergence comportementale. C'est aussi ce que les agents de maintenance et les dépanneurs appellent "un signe", "une indication", ou encore "quelque chose qui ne va pas" e.g *"la pression est anormale"*.

Généralisons à présent la définition de ce prédicat S sur l'ensemble des composants abstraits et terminaux appartenant à $\mathcal{C}$. Nous avons:

$$S(C) = \neg FC(C_1) \vee, \cdots, \vee \neg FC(C_q)$$

où $C, C_1, \cdots, C_q \in \mathcal{C}$ et $\mathcal{U}(C, C_1, \cdots, C_q)$, c'est-à-dire C utilise $C_1, \cdots, C_q$. Notons A l'ensemble des composants $\{C_i, \cdots, C_j\} \subset \{C_1, \cdots, C_q\}$ tel que:

$$x \in A \Leftrightarrow \neg FC(x)$$

Par convention, nous appellerons les composants appartenant à A des **composants-cause** alors que le composant C sera appelé **composant-effet**.

D'un point de vue macroscopique, notre démarche est similaire à celle employée par les dépanneurs. Intuitivement, elle consiste à partir des effets ou des symptômes, matérialisés ici par la non-vérification des lois, pour aboutir progressivement aux causes, c'est-à-dire aux pannes. Mais la démarche inverse, qui consiste à partir des causes pour vérifier les effets est aussi utilisée. C'est ce que nous ferons plus loin, où nous définirons alors le concept de *panne* pour les composants corpusculaires (§3). Nous aborderons ensuite le concept de panne *confirmée* ainsi que celui de panne *infirmée* au paragraphe suivant.

Mais ce qui précède reste à un niveau définitionnel et n'indique pas ni comment localiser les symptômes dans la hiérarchie, ni a fortiori comment déterminer les composants corpusculaires en panne. Or, pour les dépanneurs et les agents de maintenance, ce sont ces derniers qui constituent le résultat d'un diagnostic. Ceci nous amène alors à:

- adopter tout d'abord des stratégies, qui consistent en des heuristiques, pour guider l'exploration de la hiérarchie, c'est-à-dire localiser progressivement les composants effet/cause. Cette localisation se poursuit jusqu'à ce que *l'on atteigne le niveau des composants terminaux de la hiérarchie.*

- introduire, ensuite, au niveau des composants terminaux, des tactiques algorithmiques. Intuitivement, le but poursuivi par une tactique, définie au niveau d'un composant terminal *ct*, est *d'identifier, parmi les composants corpusculaires utilisés par ct, ceux qui sont en panne.*

Notre tâche immédiate consiste donc à:

(1) analyser les lois de chaque composant appartenant à la hiérarchie,

(2) décrire:

  - les stratégies à conduire au niveau de chaque composant abstrait,
  - les tactiques à mettre en oeuvre au niveau de chaque composant terminal,

C'est l'analyse des lois ainsi que la description des stratégies, des tactiques mais aussi des paradigmes (Cf. §5) que nous appellerons spécification du diagnostic. Mais avant d'aborder l'aspect spécification, il convient de définir plus précisement les stratégies et les tactiques.

2 Les stratégies: des heuristiques d'exploration

Par les stratégies que nous allons décrire, nous n'avons pas l'intention de couvrir toutes les possibilités pour explorer la hiérarchie. Au contraire, nous éviterons les stratégies spécifiques qui conduisent à utiliser des "astuces" de dépannage. Nous ne retiendrons ici que les stratégies de "base" qui peuvent s'appliquer dans de très nombreux cas. Actuellement, nous avons identifié quatre stratégies[1]:

- **focus**: la stratégie de focalisation,
- **ignore**: la stratégie d'ignorance,
- **up-stream**: la stratégie de suspicion en aval,
- **down-stream**: la stratégie de suspicion en amont.

2.1 Les stratégies explicites: "focus" et "ignore"

Soit C un composant abstrait et $E = \{C_1, \cdots, C_n\}$ un ensemble de composants abstraits ou terminaux. Considérons la relation $\mathcal{U}(C, C_1, \cdots, C_n)$ et B, un sous-ensemble de composants de E, $B \subset E$.

- **focus**(B): dans cette stratégie, on énumère le sous-ensemble de composants où la recherche doit se focaliser. L'application de cette stratégie est conditionnée par:

  $$x \in B \Rightarrow \neg FC(x)$$

  Intuitivement, cette stratégie indique: *partant du composant-effet C, l'exploration de la hiérarchie doit se poursuivre dans les composants-cause donnés en argument à condition que ces derniers ne fonctionnent pas correctement.* Mais à leur tour, ces composants-cause deviennent des composants-effet, on applique alors des stratégies pour trouver les composants-cause, etc.

- **ignore**(B): contrairement à la stratégie précedente, dans ce cas on donne en argument le sous-ensemble de composants qui ne doivent pas être explorés. L'application de cette stratégie est conditionnée par:

  $$x \in B \Rightarrow FC(x)$$

  Selon cette stratégie, l'exploration dans la hiérarchie doit se poursuivre dans $\overline{B}$ l'ensemble complémentaire à B dans E. Notons que cette stratégie n'est pas une simple variante de la précédente. Ainsi, la stratégie *focus(B)* n'est pas équivalente à *ignore*($\overline{B}$) avec $\overline{B} = \{x/x \in E \wedge x \notin B\}$. En effet, les préconditions de chacune sont différentes.

[1] Toutefois, comme nous le verrons au chapitre suivant, les stratégies seront définies dans un catalogue, l'utilisateur (e.g les dépanneurs et les agents de maintenance) peuvent alors enrichir ce catalogue en définissant leurs propres stratégies.

2.2 Les stratégies calculées: "up-stream" et "down-stream"

En utilisant les notations du paragraphe précédent, considérons:

- $G(E, \Gamma)$, un graphe orienté défini sur l'ensemble E des composants.
- P, l'ensemble des composants prédécesseurs aux composants du sous-ensemble B dans le graphe G:
$$P = \bigcup_{k \in I\!N^*} (\Gamma^{-1})^k y, y \in B$$
- S, l'ensemble des composants successeurs aux composants du sous-ensemble B dans le graphe G:
$$S = \bigcup_{k \in I\!N^*} \Gamma^k y, y \in B$$

définissons alors:

- **up-stream**(G, B): cette stratégie est analogue à la stratégie de focalisation avec un renforcement des préconditions. Le sous-ensemble B énumère les composants sur lesquels la recherche doit se focaliser. L'application de cette stratégie est conditionnée par:
$$x \in B \Rightarrow \neg FC(x)$$
$$x \in P \Rightarrow FC(x)$$
où P est l'ensemble des composants prédécesseurs aux composants appartenant au sous-ensemble B. Intuitivement, cette stratégie indique: *l'exploration doit se poursuivre dans les composants appartenant au sous-ensemble B à condition que ces derniers ne fonctionnent pas correctement et que l'ensemble des composants prédecesseurs fonctionnent correctement.* Exemple, soient les composants analyseur-O2 (A-O) relais-à-seuil (RAS), rotamètre (ROT), régulateur-de-pression (R-P), électro-vanne-3-voix (E-V), mesure-électrique (M-E), la relation secondaire inter-$\mathcal{CA}$s "contrôle" et considérons:

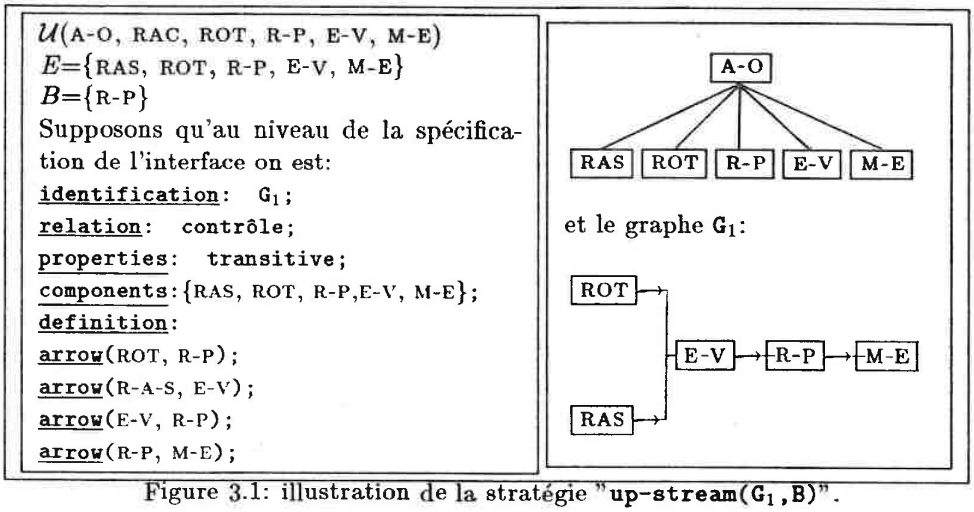


Figure 3.1: illustration de la stratégie "**up-stream(G_1,B)**".

La spécification de la stratégie "**up-stream(G_1,B)**" au niveau du composant A-O signifie que: *l'exploration de la hiérarchie doit se poursuivre au niveau du composant* R-P *à condition que ce dernier ne fonctionne pas correctement et que, de plus, on soit assuré que tous les composants en aval (i.e ceux qui le contrôlent) sont en bon état.*

- down-stream(G, B): cette stratégie est complémentaire à la précédente. Dans ce cas aussi le sous-ensemble B énumère les composants sur lesquels la recherche doit se poursuivre. L'application de cette stratégie est conditionnée par:

$$x \in B \Rightarrow \neg FC(x)$$
$$x \in S \Rightarrow \neg FC(x)$$

où S est l'ensemble des composants successeurs aux composants appartenant au sous-ensemble B. Intuitivement, cette stratégie indique: *l'exploration doit se poursuivre dans les composants appartenant au sous-ensemble B à condition que ni ces derniers ni leurs successeurs ne fonctionnent correctement.* Exemple, soient les composants cage_1 (CG) reducteur (RD), entraineur (EN), régulateur (RG), ecartement-cylindres (EC) la relation secondaire inter-$\mathcal{CA}$s "communique-vers" et considérons:

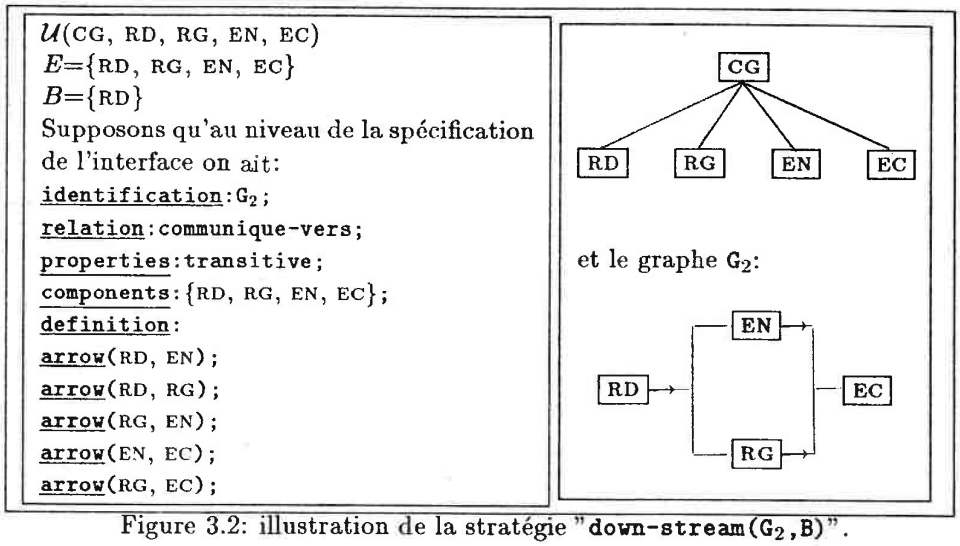


Figure 3.2: illustration de la stratégie "**down-stream(G_2,B)**".

La spécification de la stratégie "**down-stream(G_2,B)**" au niveau du composant CG signifie que *l'exploration de la hiérarchie doit se poursuivre au niveau du composant* RD *à condition que ce dernier et tous ceux qui sont en amont de ce composant ne fonctionnent pas correctement.*

3 Les tactiques: des algorithmes de diagnostic

Une tactique consiste en la mise en oeuvre d'un algorithme de diagnostic, lequel admet un ensemble E de composants corpusculaires et construit un sous-ensemble R, éventuellement vide, de E. Nous appellerons les composants obtenus par l'application d'une tactique des composants déclarés en panne. Plus précisement, nous définissons le prédicat unaire DP (i.e Déclaré en Panne):

$$x \in R \Rightarrow DP(x)$$

où R est le sous-ensemble de composants corpusculaires engendré par la tactique.

Intuitivement, notre démarche peut se résumer comme suit:

- *progressivement, à l'aide des stratégies, nous procédons à l'exploration de la hiérarchie à travers les composants abstraits jusqu'à atteindre le niveau des composants terminaux.*
- *cette exploration aboutit à l'identification d'un ensemble de composants-cause qui sont tous des composants terminaux.*
- *maintenant, chacun de ces composants-cause devient, à son tour, un composant-effet auquel nous appliquons une tactique algorithmique pour identifier, parmi les compo-*

sants corpusculaires qu'il utilise, les composants-cause: les composants déclarés en panne par la tactique.

Ainsi, par l'introduction des tactiques dans notre démarche, nous retrouvons ici un point de vue assez voisin de l'utilisation locale des procédures de diagnostic par les dépanneurs et les agents de maintenance. Malheureusement, ces procédures, souvent fournies par les constructeurs, sont très spécifiques à une gamme particulière de composants et ne peuvent être présentées telles ici. Nous sommes alors conduits à en étudier certaines pour s'en inspirer et dégager un ensemble de principes de base qui nous permettront de proposer des tactiques générales et indépendantes. Ici, nous allons nous limiter à présenter trois tactiques[2]:

- **trudge**: la tactique par couches,
- **climb**: la tactique par ascendance,
- **relax**: la tactique par relaxation.

Les deux premières tactiques, que nous avons appelées *"trudge"* et *"climb"*, sont issues de notre étude des procédures de diagnostic. Quant à la dernière tactique *"relax"* est proposée par Davis dans [Davis 84] et ne sera que brièvement présentée ici.

3.1 La tactique par couches: "trudge"

Principe

- en choisissant une relation d'ordre (e.g communique vers), organiser l'ensemble des composants corpusculaires en différentes couches $\mathcal{C}_0, \mathcal{C}_1, \cdots, \mathcal{C}_n$ superposées.
- en partant de la couche la plus interne $\mathcal{C}_0$, vérifier incrémentalement l'état des composants de chaque couche.
- en supposant que, dans le pire des cas, les composants d'une et une seule couche peuvent tous tomber en panne en même temps, appliquer le principe de culpabilisation suivant: *les composants $c_1^i, \cdots c_m^i$ de la couche $\mathcal{C}_i$ sont déclarés en panne si:*

  (1) il existe au moins un composant de $\mathcal{C}_i$ qui ne fonctionne pas correctement,
  (2) tous les composants des couches englobant $\mathcal{C}_i$ fonctionnent correctement.
  nous avons ainsi:

$$DP(c_1^i), \cdots, DP(c_m^i) \Leftrightarrow \begin{cases} \exists c \in \{c_1^i, \cdots, c_m^i\} \text{tel que } \neg FC(c) \\ \forall j > i, \forall c^j \text{ dans la couche } \mathcal{C}_j, FC(c^j) \end{cases}$$

[2]Tout comme pour les stratégies, les tactiques seront aussi définies dans un catalogue permettant ainsi l'adjonction d'autres tactiques.

Algorithme

Notations	
n	: le nombre de couches,
A_i	: l'ensemble des composants de la couche $\mathcal{C}_i$.
dp	: l'ensemble des composants déclarés en "panne". i.e $\forall x \in dp, DP(x)$
suspects	: l'ensemble des composants supposés suspects.
innocents	: l'ensemble des composants innocents.
$\cup$	: l'union ensembliste.
$\setminus$	: la différence ensembliste.
$\emptyset$	: l'ensemble vide.

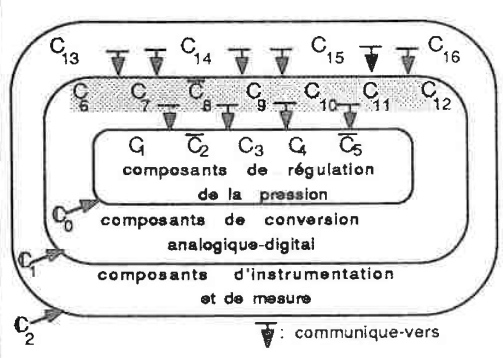
dp = trudge($A_0, \cdots, A_n$):

suspects $\leftarrow \emptyset$
innocents $\leftarrow \emptyset$
POUR i DE 0 À n FAIRE
 SI $x \in A_i \Leftrightarrow FC(x)$
  ALORS
   innocents $\leftarrow$ innocents $\cup \{x/x \in A_i\}$
  SINON
   suspects $\leftarrow$ suspects $\cup \{x/x \in A_i\}$
   suspects $\leftarrow$ suspects $\setminus \bigcup_{0<j<i} \{x/x \in A_j\}$
 FSI
FPOUR
dp $\leftarrow$ suspects

Figure 3.3: la tactique "trudge($A_0, \cdots, A_n$)"

Exemple

Considérons la relation "communique-vers" et illustrons cette tactique sur l'exemple suivant:



Initialement on a:
- $A_0 = \{C_1, C_2, C_3, C_4, C_5\}$,
- $A_1 = \{C_6, C_7, C_8, C_9, C_{10}, C_{11}, C_{12}\}$,
- $A_2 = \{C_{13}, C_{14}, C_{15}, C_{16}\}$,
- suspects=$\emptyset$ et innocents=$\emptyset$

Etape 1: traitement de A_0
supposons: $\neg FC(C_2)$ et $\neg FC(C_5)$ on a:
- suspects=$\{C_1, C_2, C_3, C_4, C_5\}$,

Etape 2: traitement de A_1
supposons: $\neg FC(C_8)$ on a:
- suspects=$\{C_6, C_7, C_8, C_9, C_{10}, C_{11}, C_{12}\}$,

Etape 3: traitement de A_2
supposons: $x \in A_2 \Leftrightarrow FC(x)$ on a:
- innocents=$\{C_{13}, C_{14}, C_{15}, C_{16}\}$

Finalement dp = suspects = A_1:
$x \in dp \Leftrightarrow DP(x)$

Figure 3.4: illustration de la tactique "trudge".

Hypothèses

L'exactitude du résultat (i.e l'ensemble dp) de cette tactique est conditionnée par:

(H_1) au pire des cas, les composants d'une seule et une seule couche peuvent tous être en état de dysfonctionnement. Notons que contrairement aux autres travaux qui considèrent cette hypothèse[3] au niveau de chacun des composants de toute l'installation, ici seuls les composants diagnostiqués par la tactique sont concernés par cette hypothèse.

(H_2) la non-interférence entre couches de composants. Plus précisément: *le fonctionnement des composants d'une couche d'un niveau donné ne peut en aucun cas être altéré par un dysfonctionnement quelconque des composants d'une couche d'un niveau inférieur.*

Si la première hypothèse est classique et se trouve souvent validée la seconde l'est moins et se prête difficilement à une vérification. En effet le caractère aléatoire d'un composant en panne, ou déclaré comme tel, peut induire des effets de bord imprévisibles sur les autres composants. Il suffit d'imaginer, par exemple, un court-circuit d'une résistance qui, faisant croître l'intensité globale, altère le fonctionnement des autres composants. Au paragraphe suivant (§4) nous verrons comment remédier à ce problème.

3.2 La tactique d'ascendance: "climb"

Principe

- en utilisant, la relation d'ordre "contrôle", trier topologiquement l'ensemble des composants corpusculaires. Nous noterons par $\mathbb{R}^i$ l'ensemble des composants de rang i, par $\mathbb{R}^0$ l'ensemble des composants qui n'ont pas de composants-prédécesseurs et par $\mathbb{R}^n$ l'ensemble des composants qui n'ont pas de composants-successeurs.
- en partant des composants appartenant à $\mathbb{R}^n$, examiner de manière ascendante les composants de $\mathbb{R}^{n-1}, \cdots, \mathbb{R}^0$.
- en considérant l'hypothèse qu'il y a au plus un composant en panne, appliquer le principe de culpabilisation suivant: *un composant $c \in \mathbb{R}^i$ est déclaré en panne si*:
  (1) il contrôle au moins deux composants qui ont été déjà déclarés en panne,
  (2) il n'existe aucun autre composant qui contrôle ce composant et qui soit déclaré en panne.

  nous avons:

$$DP(c) \Leftrightarrow \begin{cases} \exists\, c_j^{i+1}, c_k^{i+1} \text{ tel que } \{c_j^{i+1}, c_k^{i+1}\} \subset dp_{i+1} \\ \forall m > i,\ dp_m = \emptyset \end{cases}$$

[3] Connue sous le nom d'hypothèse de la panne unique.

où $dp_{i+1} \subset I\!R^{i+1}$ dénote l'ensemble des composants déclarés en panne au niveau $i+1$.

L'idée intuitive sous-jacente à cette tactique est: *d'après l'hypothèse de la panne unique, le constat du dysfonctionnement de plus d'un composant nous conduit à rechercher un autre composant, unique, qui, parce qu'il les contrôle, provoque les dysfonctionnements constatés. Il suffit alors de rechercher l'ancêtre commun de tous ces composants en dysfonctionnement.*

Algorithme

Notations	
G_c	: le graphe des composants suivant la relation "contrôle".
$I\!R^i$	: l'ensemble des composants de rang i.
dp	: l'ensemble des composants déclarés en "panne".
suspects_i	: l'ensemble des composants suspects appartenant à $I\!R^i$
n	: le nombre de niveaux du graphe G_c,
$\cap$	: l'intersection ensembliste.
$Pred(x, G)$	: primitive qui donne l'ensemble des predécesseurs du composant x dans G. $Pred(x) = \{y/(\Gamma)^{-1}y, y \in G\}$
card(E)	: fonction qui renvoie le cardinal d'un ensemble.
$\emptyset$	: l'ensemble vide.

dp = climb(G_c,$I\!R^1, \cdots, I\!R^n$):

dp $\leftarrow \emptyset$
i $\leftarrow$ n
$\text{suspects}_i \leftarrow \emptyset$
TANT-QUE $\text{suspects}_i = \emptyset \wedge i \geq 0$ FAIRE
  $\text{suspects}_i \leftarrow \{x/\neg FC(x) \wedge x \in I\!R^i$
  i $\leftarrow$ i -1
FTTQUE
TANT-QUE $\text{suspects}_i \neq \emptyset \wedge i \geq 0$ FAIRE
  $\text{suspects}_i \leftarrow \bigcap_{x \in \text{suspects}_{i+1}} Pred(x)$
  i $\leftarrow$ i -1
FTTQUE
CAS:
  $\text{card}(\text{suspects}_{i+1}) = 0$ % echec
  $\text{card}(\text{suspects}_{i+1}) = 1$ dp $\leftarrow \text{suspects}_i$
  AUTRES % echec
FCAS

Figure 3.5: la tactique "climb($G_c, I\!R^1, \cdots, I\!R^n$)".

Exemple

considérons la relation "contrôle" et illustrons cette tactique sur l'exemple suivant:

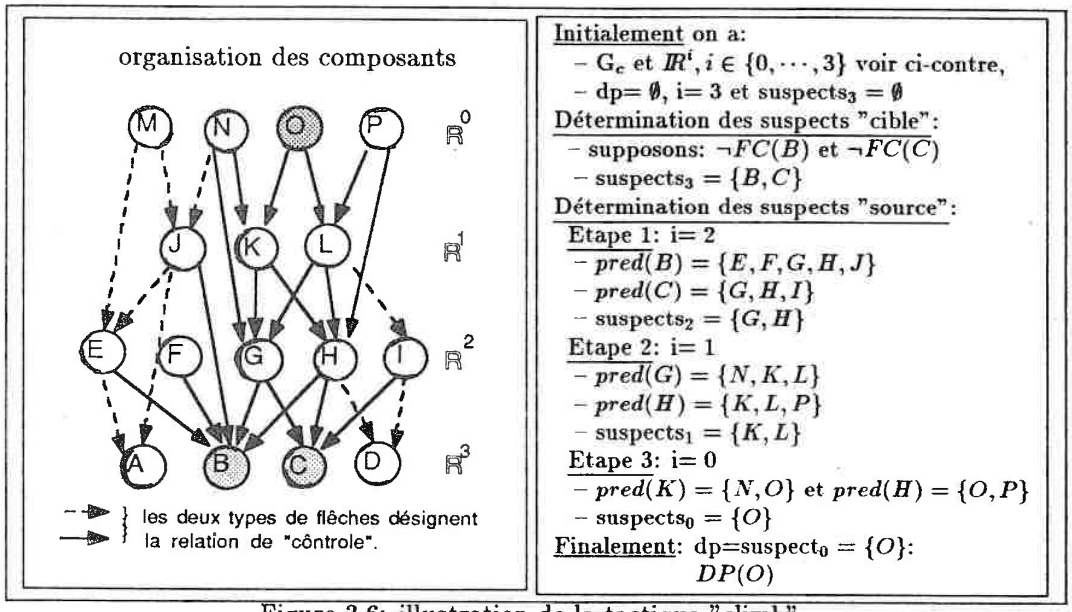


Figure 3.6: illustration de la tactique "climb".

Hypothèses

(H_1) l'hypothèse de la panne unique au niveau des composants diagnostiqués par la tactique.

(H_2) il n'existe pas de composant jouant le rôle de "catalyseur de pannes". Autrement dit: *il ya une propagation du dysfonctionnement à travers les composants.* Pour des raisons analogues à celles données précédemment (Cf. H_1 au §3.2), cette hypothèse est aussi difficile à vérifier.

3.3 La tactique de relaxation: "relax"

Cette tactique est basée sur l'algorithme de Davis [Davis 84] qui a été présenté au chapitre précédent. Pour utiliser cet algorithme, nous sommes conduits, ici, à:

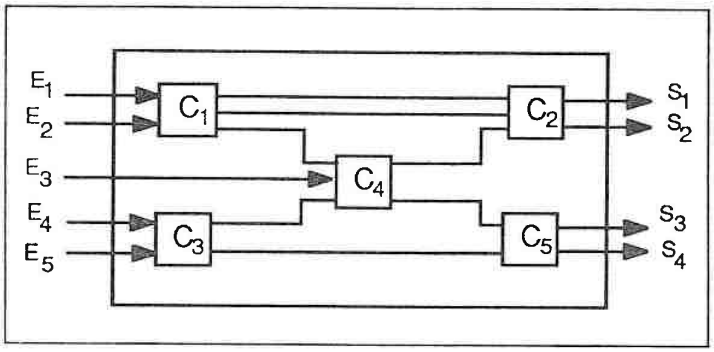


Figure 3.7: modélisation des composants pour la tactique "relax".

- considérer l'ensemble des composants comme un réseau caractérisé par un ensemble de valeurs en entrée (i.e $E_1, \cdots, E_5$) et un ensemble de valeurs en sortie (i.e $S_1, \cdots, S_4$),
- donner pour chaque sortie les liens de dépendance, c'est-à-dire les composants qui contribuent à fournir cette sortie,
- définir les valeurs intermédiaires des composants pour lesquelles on dispose des valeurs observées dans l'installation,

et à appliquer, par la suite, l'algorithme de Davis qui a été détaillé au §3.2 du chapitre précédent.

Hypothèses

(H_1) on retrouve de nouveau l'hypothèse de la panne unique.

(H_2) toute communication, quelle qu'elle soit, entre les composants est explicitement spécifiée [Davis 84]. Autrement dit, la structure des composants ne joue aucun rôle[4]. Ainsi, l'adjacence physique de deux composants électriques ne doit, en aucun cas, entraîner d'autres types de communications (i.e électro-magnétiques) que celles spécifiées explicitement. Ce qui est très difficile à vérifier.

[4]Cette hypothèse, connue sous le nom "*no function in structure*", a fait couler beaucoup d'encre [DeKleer 86], [Iwasaki 86a], [Iwasaki 86b].

4 Méthode de justification

Comme nous l'avons indiqué, les hypothèses des tactiques sont souvent très difficiles à vérifier. De ce fait, on ne peut affirmer avec certitude que l'ensemble *dp* des composants déclarés en panne par une tactique donnée, le sont réellement. Mais jusqu'à présent, notre démarche a été de partir des effets pour arriver aux causes, il nous faut maintenant procéder de manière ascendante, c'est-à-dire des causes vers les effets et ceci pour "valider" le résultat d'une tactique: les composants déclarés en panne. Cette validation[5] consiste à conduire un raisonnement de surface, qui utilisant des règles empiriques, calcule pour chaque composant déclaré en panne deux degrés: $dc \in [0,1]$ et $di \in [0,1]$. Intuitivement *dc* mesure le degré de culpabilité du composant alors que *di* mesure, quant à lui, son degré d'innocence.

Les règles empiriques, appelées par la suite *justifications*, sont fournies par les dépanneurs et les agents de maintenance et se scindent en deux types:

- les *confirmatives* qui visent à justifier la culpabilité du composant. Exemple d'une justification confirmative:

Notation: $exp \overset{cj}{\rightsquigarrow} \overset{c}{C}$ où	Cette confirmative se lit comme suit:
– *exp*: est une expression quelconque formée par les opérateurs: $\wedge, \vee, \neg, \leq, \geq \cdots$.	l'expression *"exp"* contribue à justifier
– $\overset{c}{C}$: dénote la culpabilité du composant C.	la culpabilité du composant C avec
– *cj*: un coefficient de justification donnée $\in [0,1]$.	un coefficient égal à *cj*.
Exemple: $delta \geq 0.5 \wedge mode = normal \overset{0.8}{\rightsquigarrow} \overset{c}{carte\text{-}interface}$	
Commentaire: Si la différence entre deux mesures données par le capteur "IR" est ≥ 0.5 mètres *et* que le mode de marche est normal alors confirmer avec un coefficient de justification de 0.8 la culpabilité du composant "carte-interface".	

Figure 3.8: exemple d'une confirmative.

- les *infirmatives*, qui au contraire, visent à justifier son innocence. Exemple d'une justification infirmative:

Notation: $exp \overset{cj}{\rightsquigarrow} \overset{i}{C}$ où	Cette infirmative se lit comme suit:
– *exp*: est une expression quelconque formée par les opérateurs: $\wedge, \vee, \neg, \leq, \geq \cdots$.	l'expression *"exp"* contribue à justifier
– $\overset{i}{C}$: dénote l'innocence du composant C.	l'innocence du composant C avec
– *cj*: un coefficient de justification donnée $\in [0,1]$.	un coefficient égale à *cj*.
Exemple: $afficheur = allumé \vee led = clignote \overset{1.0}{\rightsquigarrow} \overset{i}{carte\text{-}interface}$	
Commentaire: Si l'afficheur est allumé *ou* la led clignote alors infirmer avec un coefficient de justification de 1.0 la culpabilité du composant "carte-interface". Ce qui revient aussi à: confirmer son innocence avec un coefficient de 1.0.	

Figure 3.9: exemple d'une infirmative.

[5] C'est le terme anglais "endorsement", sans équivalent en français, qui conviendrait le mieux.

C'est la description de l'ensemble des ces justifications (i.e confirmatives et infirmatives) ainsi que la détermination du degré de culpabilité et celui d'innocence pour chacun des composants déclarés en panne, que nous appelons *méthode de justification.* Mais l'intérêt de cette méthode est plutôt limité si on ne dispose pas d'un guide pour énumérer les différentes justifications. C'est pourquoi nous définissons des paradigmes, dont le but est justement de proposer un guide méthodologique d'énumération.

Maintenant et avant d'introduire les paradigmes, il nous faut examiner le mécanisme de raisonnement que l'on peut qualifier d'incertain. La littérature à ce sujet est abondante et couvre plusieurs "types" de raisonnement: possibiliste, flou, hypothétique, causal, par défaut, etc, une ébauche de comparaisons peut être trouvée dans [Sombe 88] (voir également [Bibel 85]). Pour fonder notre mécanisme, deux critères nous ont conduit à nous inspirer essentiellement de la théorie de l'évidence proposée par [Shafer 76] ainsi que sur les travaux relatifs aux fonctions de croyance[6] [Eddy 86], [Fagin 88], [Lee 88], [Martins 88]. Le premier critère est la facilité de mise en oeuvre du mécanisme de raisonnement. Quant au second critère, il est liée à l'adéquation du mécanisme au problème de diagnostic. A cet effet, nous nous sommes basés sur les résultats satisfaisants qui ont été obtenus pour résoudre des problèmes voisins tels celui de la surveillance des radars [Bogler 87] ou encore de la prédiction des pannes [Garvey 80]. Signalons enfin que la confrontation de notre mécanisme sur un cas réel (i.e mesure de longueurs des poutrelles chaudes) a satisfait pleinement les ingénieurs et les agents de maintenance concernés. Bien sûr, ce qui précède ne peut être assimilé à une preuve et de ce fait reste ouvert à des critiques possibles. Malheureusement, l'absence d'un cadre unique[7] ou l'on peut formuler les différents types de raisonnement, évoqués plus haut, et ainsi les classifier formellement, exclut une telle preuve.

A présent, considérons C un composant et exp une expression. Adoptons alors les conventions de Driankov [Driankov 86] pour noter $/\overset{c}{C}/$ le degré de culpabilité de C, $/\overset{i}{C}/$ son degré d'innocence et $/exp/$ le degré de justification de l'expression exp.

Convenons enfin de noter indifféremment $\overset{i}{C}$ et $\overset{c}{C}$ par $\overset{\star}{C}$ et donnons les règles d'inférence [Driankov 86] pour calculer le degré de culpabilité et celui d'innocence des composants.

La première règle d'inférence, appelée règle de détachement, indique comment calculer le degré de culpabilité (resp. d'innocence) pour un composant en fonction d'une confirmative (resp. d'une infirmative).

$$\frac{exp \overset{cj}{\rightsquigarrow} \overset{\star}{C}}{/\overset{\star}{C}/ = /exp/ \times cj}$$

Figure 3.10: la règle de détachement.

[6] Traduction de l'anglais de "belief functions".

[7] A ce niveau, on peut se poser la question de savoir si ce cadre est possible, voire souhaitable.

où la fonction de multiplication "$\times$" choisie ici, remplit les 3 propriétés réquises [Driankov 86], c'est-à-dire qu'elle est *croissante* avec les deux arguments et que $\times(x,1) = x$ et qu'enfin on a: $\times(0,y) = 0$. Intuitivement cette règle exprime que : ***le degré de culpabilité (resp. d'innocence) croît proportionnellement avec le coefficient de la confirmative (resp. de l'infirmative) ainsi qu'avec le degré de justification de son antécédent.***

La seconde règle, appelé règle d'agrégation, montre comment combiner deux confirmatives (resp. infirmatives) pour calculer le degré de culpabilité (resp. d'innocence) d'un composant.

$$\frac{exp_1 \overset{cj_1}{\rightsquigarrow} \overset{\star}{C}\ ,\ exp_2 \overset{cj_2}{\rightsquigarrow} \overset{\star}{C},}{/\overset{\star}{C}/ = max(/exp_1/ \times cj_1, /exp_2/ \times cj_2)}$$

Figure 3.11: la règle d'agrégation.

de nouveau nous retrouvons les propriétés requises [Driankov 86] de la fonction "*max*", c'est-à-dire: *commutative, associative, monotone* (avec les deux arguments) et *continue.* Intuitivement cette règle indique que: ***pour combiner deux justifications, on est conduit à prendre le degré maximal.***

La troisième règle, appelé règle de la négation duale, indique comment définir l'opérateur de négation, noté par une barre, sur les deux dégrés de confirmation et d'innocence.

$$\frac{\overline{/\overset{c}{C}/}}{\overline{/\overset{c}{C}/}} \quad \text{de même que} \quad \frac{\overline{/\overset{i}{C}/}}{\overline{/\overset{i}{C}/}}$$

Figure 3.12: la règle de la négation duale.

où $\overline{\overset{c}{C}} = \overset{i}{C}$ et $\overline{\overset{i}{C}} = \overset{c}{C}$. Intuitivement cette règle indique que: ***la négation du degré de culpabilité d'un composant revient à considérer son degré d'innocence et inversement.***

En fait ces règles, adaptées pour le cas traité ici, ne sont pas du tout particulières à une théorie mais au contraire sont classiques[8]. Par contre, l'idée d'exploiter conjointement deux types de justifications (i.e les confirmatives et les infirmatives) offre, à notre avis, deux avantages:

- tout d'abord, elle permet de capturer le maximum de justifications, c'est-à-dire les règles empiriques, auprès des dépanneurs et des agents de maintenance, autrement dit le maximum d'expertise.

- ensuite, comme le souligne Driankov [Driankov 86], elle permet de distinguer clairement les cas d'ignorances et de contradictions. Ces deux cas sont souvent confondus

[8]Pour une justification de ces règles, on peut se référer à [Driankov 86].

dans la littérature sur ce sujet. L'ignorance, qui signifie une "sous-information", se produit lorsque:

$/\overset{c}{C}/ \leq S_c$ et $/\overset{i}{C}/ \leq S_i$

alors que la contradiction, qui signifie la "sur-information", survient à son tour lorsque:

$/\overset{c}{C}/ > S_c$ et $/\overset{i}{C}/ > S_i$

où $S_i \in [0,1]$ et $S_c \in [0,1]$ sont deux seuils donnés.

Pour compléter les définitions des concepts de panne et de symptôme (Cf. §1). Considérons $S_c \in [0,1]$ un seuil de confirmation et $S_i \in [0,1]$ un seuil d'infirmation, et définissons les prédicats unaires PC (i.e Panne Confirmée) et PI (i.e Panne Infirmée):

$$PC(C) \Leftrightarrow DP(C) \wedge /\overset{c}{C}/ > S_c$$
$$PI(C) \Leftrightarrow DP(C) \wedge /\overset{i}{C}/ > S_i$$

où C est un composant corpusculaire donné.

5 Les Paradigmes: des schémas de règles

Comme nous l'avons indiqué au paragraphe précédent, le problème majeur que pose un raisonnement de surface est l'énumération des règles empiriques, c'est-à-dire les confirmatives et les infirmatives. A cet effet, nous allons proposer quatre paradigmes. A chacun nous associons:

- un ensemble de confirmatives ou d'infirmatives
- un guide méthodologique d'énumération

5.1 Le paradigme de liaison: "link"

Guide:	Notation:
Considérer que le composant est soit coupable soit innocent, exprimer, par la suite, les conditions *nécessaires* à l'hypothèse considérée.	link$(exp_1, \cdots, exp_n, cj, \overset{\star}{C})$ Justifications associées: $\overset{\star}{C} \overset{cj}{\rightsquigarrow} exp_i, \ i \in \{1, \cdots, n\}$ $exp_1 \wedge, \cdots, \wedge, exp_n \overset{1}{\rightsquigarrow} \overset{\star}{C}$

Figure 3.13: le paradigme "link".

Illustrons sur un exemple simple, ce paradigme, pour confirmer la culpabalisation:

Soit le composant "carte-interface" (CI) qui est contrôlé par un bit de défaut (BD) positionné en logique négative et qui, par une ligne (AEG), transmet une mesure quelconque. Considérons à présent que la carte est coupable; alors on doit d'une part avoir le bit à zéro et d'autre part la ligne bloquée.

Traduction de l'exemple:

link(BD = 0, AEG = bloquée, 1, $\overset{c}{\text{CI}}$)

$\overset{c}{\text{CI}} \overset{1}{\rightsquigarrow}$ BD = 0, $\overset{c}{\text{CI}} \overset{1}{\rightsquigarrow}$ AEG = bloquée,

BD = 0 $\wedge$ AEG = bloquée $\overset{1}{\rightsquigarrow} \overset{c}{\text{CI}}$

Figure 3.13 (suite): illustration du paradigme "link".

5.2 Le paradigme d'inclusion: "include"

Guide:

Considérer que le composant est soit coupable soit innocent, et en déduire alors les *conséquences.*

Notation:

include($exp_1, \cdots, exp_n, cj, \overset{\star}{C}$)

Justifications associées:

$\overset{\star}{C} \overset{cj}{\rightsquigarrow} \neg exp_i, \quad i \in \{1, \cdots, n\}$

$exp_1 \vee, \cdots, \vee exp_n \overset{cj}{\rightsquigarrow} \overset{\star}{C}$

Figure 3.14: le paradigme "include".

Illustrons sur un exemple simple, ce paradigme, pour confirmer l'innocence:

Soit le composant "optocoupleurs" (OC), qui joue le rôle d'interface d'acquistion des données. A ce composant, est associée une led (LD) et un afficheur local (AFL) qui clignotent en cas d'un dysfonctionnement quelconque. Supposons que OC est innocent, on doit alors avoir la led ainsi que l'afficheur local éteints. Toutefois, il est nécessaire que ni la led ni l'afficheur local ne soient hors-service.

Traduction de l'exemple:

include(AFL = éteint,LD = éteint,0.2, $\overset{i}{\text{OC}}$)

$\overset{c}{\text{OC}} \overset{0.2}{\rightsquigarrow} \neg$ (AFL = éteint),

$\overset{c}{\text{OC}} \overset{0.2}{\rightsquigarrow} \neg$ (LD = éteint),

AFL = éteint $\vee$ LD = éteint $\overset{0.2}{\rightsquigarrow} \overset{i}{\text{OC}}$

Figure 3.14 (suite): illustration du paradigme "include".

5.3 Le paradigme de propagation: "propagate"

Guide:

Commencer par se donner un ensemble d'hypothèses, considérer par la suite que le composant est soit coupable soit innocent. Propager ensuite l'effet de toutes les hypothèses prises et exprimer l'effet de cette propagation

Notation:

propagate($exp_1, \cdots, exp_n, cj, h_1, \cdots, h_m, \overset{\star}{C}$)

Justifications associées:

$\overset{\star}{C} \overset{1}{\rightsquigarrow} h_i, \ i \in \{1, \cdots, m\}$

$\neg exp_1 \wedge, \cdots, \wedge, \neg exp_n \overset{1}{\rightsquigarrow} \overline{\overset{\star}{C}}$

$exp_1 \vee, \cdots, \vee exp_n \overset{cj}{\rightsquigarrow} \overset{\star}{C}$

Figure 3.15: le paradigme "propagate".

Illustrons sur un exemple simple, ce paradigme, pour confirmer la culpabilité:

Soit le composant disjoncteur de courant (DC) qui, via un autre composant, un sectionneur manuel (SM), alimente deux autres composants: un régulateur (RG) et une scie (SIE). Supposons que le sectionneur manuel est fermé et considérons que le disjoncteur est coupable. Dans ce cas, on doit alors avoir la scie ainsi que le régulateur hors-service (HS). Notons "x = hors-service" resp. "x = en-service" par "hs(x)" resp. "es(x)"

Traduction de l'exemple:

propagate(hs(RG),hs(SIE),1,SM=fermée,$\overset{c}{\text{DC}}$)

$\overset{c}{C} \overset{1}{\rightsquigarrow}$ SM = fermée,

es(RG) $\wedge$ es(SIE) $\overset{1}{\rightsquigarrow} \overset{i}{\text{DC}}$

hs(RG) $\vee$ hs(SIE) $\overset{1}{\rightsquigarrow} \overset{c}{\text{DC}}$

Figure 3.15 (suite): illustration du paradigme "propagate".

5.4 Le paradigme d'exclusion "exclude"

Guide:

Soit considérer le composant comme coupable et chercher tous les autres composants qui ne peuvent l'être, soit inversement, le considérer comme innocent et chercher les autres composants qui doivent être nécessairement coupables.

Notation:

exclude($exp_1, \cdots, exp_n, cj, \overset{\star}{C}$)

Justifications associées:

$\overset{\star}{C} \overset{1}{\rightsquigarrow} \neg exp_i \ \ i \in \{1, \cdots, n\}$

$exp_1 \vee, \cdots, \vee exp_n \overset{cj}{\rightsquigarrow} \overline{\overset{\star}{C}}$

Figure 3.16: le paradigme "exclude".

Illustrons ce paradigme sur un exemple:

Soient les composants: un convertisseur analogique-digital (CAD), automate (AUT) et un entraineur (ENT). le convertisseur est en communication à la fois avec l'automate et avec l'entraineur. Maintenant supposons que ce convertisseur est coupable, on peut alors innocenter les deux autres composants.

Traduction de l'exemple:

exclude($\overset{c}{\text{AUT}}$, $\overset{c}{\text{ENT}}$, 0.5,$\overset{c}{\text{CAD}}$)

$\overset{c}{\text{CAD}} \overset{1}{\rightsquigarrow} \overset{i}{\text{AUT}}$

$\overset{c}{\text{CAD}} \overset{1}{\rightsquigarrow} \overset{i}{\text{ENT}}$

$\overset{c}{\text{ENT}} \vee \overset{c}{\text{AUT}} \overset{0.5}{\rightsquigarrow} \overset{i}{\text{CAD}}$

Figure 3.16 (suite): illustration du paradigme "exclude".

6 Spécification du diagnostic

A présent, nous abordons l'étape de la spécification du diagnostic qui consiste ainsi à:

- analyser les lois des composants abstraits et terminaux et décrire: "*que faut-il faire, en cas de violation d'une ou de plusieurs lois ?*" Cette description est exprimée en terme de stratégies et de tactiques.

- décrire les paradigmes de chaque composant corpusculaire pour justifier sa culpabilité ou son innocence.

La syntaxe de spécification du diagnostic est:

<*diagnostic*> ::= begin <*directives*> end
<*directives*> ::= <*directive*> { <*directive*> }*
<*directive*> ::= <*unite*> { <*unite*> }*
<*unite*> ::= LABEL ':' when <*id-loi*> { <*id-loi*> }*
[on <$exp_{\{pe\}}$>] [in <$exp_{\{ve\}}$>] [with <$exp_{\{m\}}$>]
$\mapsto$ [<*id-diagnostic*>] [true] ';'

Figure 3.17: syntaxe concrète pour la spécification du diagnostic.

où <*id-loi*> dénote l'identificateur d'une loi et <*id-diagnostic*> dénote une stratégie ou une tactique.

Intuitivement, la sémantique d'une unité de diagnostic est: *lorsque chacune des lois identifiées par les <id-loi>s n'est pas vérifiée et que chacune des expressions <$exp_{\{pe\}}$>, <$exp_{\{ve\}}$> et <$exp_{\{m\}}$> est évaluée à vrai alors mettre en oeuvre la stratégie ou la tactique identifiée par <id-diagnostic>*. De nouveau ici, nous adoptons la même démarche qu'auparavant pour regrouper les unités de diagnostic en paquets étiquetés que nous appellerons des directives de diagnostic. Une directive est ainsi une suite ordonnée d'unités de diagnostic portant la même étiquette. Voici un exemple:

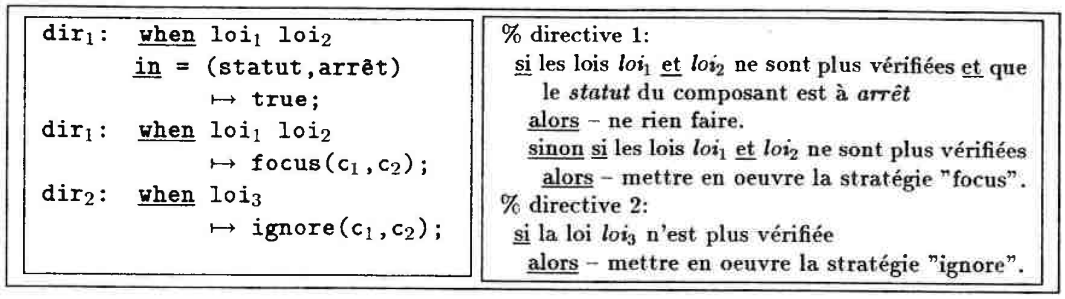


Figure 3.18: illustration des directives de diagnostic sur un exemple.

6.1 Exemples

A titre illustratif, écrivons les spécifications de diagnostic des composants "pompe" et "analyseur-O2".

Exemple 1: les stratégies du $\mathcal{CA}$ "pompe".

```
(1) pompe(lubrification);
    % déclaration des différentes directives de
    % diagnostic et de leurs priorités.
(2) diag1 : 3;
(3) diag2 : 2;
(4) diag3 : 1;
    begin
(5) diag1: when loi3 in ≤(puissance,1100) ↦ true;
(6) diag1: when loi3 in >(puissance,1700) ↦ true;
(7) diag1: when loi3 ↦ focus(moteur-pompe);
(8) diag2: when loi1 ↦ ignore(moteur-pompe);
(9) diag3: when loi2 in =(état,arret) ↦ true;
(10)diag3: when loi2 ↦ ignore(moteur-pompe);
    end
```

Figure 3.19: spécification du diagnostic du $\mathcal{CA}$ "pompe".

Outre les stratégies "focus" et "ignore" qui ont été utilisées dans cet exemple, remarquons, comme à la ligne 5, la présence de la stratégie *fictive* "true". Cette stratégie renvoie toujours la valeur *vrai* et consiste à ne rien faire. Commentons la première unité de diagnostic "diag$_1$": *si la loi 3 n'est plus vérifiée (i.e problème de débit) et que la puissance est pratiquement au minimum (i.e 1100) ou au maximum (i.e 1700) alors ne rien faire. Autrement, focaliser le mécanisme de diagnostic sur le "moteur-pompe".*

Exemple 2: les tactiques du composant "analyseur-O2".

```
(1) analyseur-02(analyseur);
(2) diag1 :  5;
(3) diag2 :  2;
(4) diag3 :  1;
             ⋮
    begin
(5) diag1:   when loi1 in =(mode-de-marche,ralenti) ↦ true;
(6) diag1:   when loi1 in =(mode-de-marche,ralenti)
                        =(pression-atmospherique,bas) ↦
                         signal(Dérivation-A-02);
(7) diag1:   when loi1 ↦ user(procédure1-diag-A02);
(8) diag2:   when loi3 in =(pression-atmospherique,haut) ↦ true
(9) diag3:   when loi3 ↦ user(procédure2-diag-A02);
             ⋮
    end
```

Figure 3.20: spécification du diagnostic du CT "analyseur-O2".

Dans cette spécification, on peut remarquer la tactique signal(x) qui a pour effet d'envoyer à l'opérateur le message donné en argument. Cette tactique renvoie toujours la valeur *vrai*. Nous avons aussi la tactique, *virtuelle*, user qui a pour effet de mettre en oeuvre une procédure de diagnostic définie par l'utilisateur. Remarquons enfin que les lignes 5 et 6 font référence au composant virtuel "environnement" de l'installation "haut-fourneau" à laquelle appartient ce composant.

7 Mécanisme de diagnostic: vers une coopération

D'après ce qui précède, il apparaît clairement que nous allons mettre en oeuvre à la fois un raisonnement de surface et un raisonnement profond. Le premier utilisera les stratégies et les paradigmes alors que le second utilisera, quant à lui, les tactiques. Comme nous l'avons indiqué au cours du second chapitre, le raisonnement profond et celui de surface sont malheureusement souvent considérés comme distincts voire, parfois, opposés [ElAyeb 88c]. Or, il nous semble plutôt judicieux de les considérer comme complémentaires et de ce fait les exploiter conjointement. Ceci permet de tirer avantage de chacun. Bien sûr, il ne s'agit pas de faire une juxtaposition passive mais au contraire une exploitation "*habile*" que nous avons appelée jusque là: *coopération*.

7.1 Principe

Soit C le composant-effet d'une hiérarchie $\mathcal{H}$ d'une installation donnée, la coopération peut se résumer, schématiquement, en trois étapes:

Etape 1: LOCALISATION PROGRESSIVE DES COMPOSANTS-CAUSE

En utilisant les stratégies de la spécification du diagnostic du composant-effet C et la hiérarchie $\mathcal{H}$ de l'installation, le mécanisme du type **raisonnement de surface** *localise les composants-cause de C dans $\mathcal{H}$. Les composants-cause deviennent, à leurs tour, des composants-effet, on réitère alors ce processus jusqu'à atteindre le niveau des composants terminaux. En cas d'échec d'une itération, c'est-à-dire si les préconditions de la stratégie ne sont pas vérifiées, un retour arrière est effectué pour sélectionner, s'il en existe, une autre stratégie ou un autre composant-effet en attente. Ce sont les priorités qui jouent le rôle de critère de sélection.*

Etape 2: DÉCLARATION DES COMPOSANTS CORPUSCULAIRES EN PANNE: *les suspects*

A cette étape, il s'agit de mettre en oeuvre un mécanisme du type **raisonnement profond** *qui utilisant la tactique spécifiée au niveau du composant-effet identifié plus haut, détermine l'ensemble "dp" des composants corpusculaires déclarés en panne. De nouveau en cas d'échec d'une tactique, c'est-à-dire que l'ensemble "dp" est vide, un retour arrière est effectué pour sélectionner une autre tactique.*

Etape 3: JUSTIFICATION DE LA CULPABILITÉ VS DE L'INNOCENCE

Cette dernière étape de justification est réalisée par un mécanisme du type **raisonnement de surface** *qui utilise des paradigmes pour confirmer et / ou infirmer la culpabilité de chaque composant corpusculaire déclaré en panne à l'étape précédente.*

Figure 21: Principe de la coopération.

Bien que ce principe soit illustré de manière séquentielle, il n'exclut nullement la possibilité d'explorer en parallèle plusieurs composants de la hiérarchie. Ce problème sera brièvement évoqué en épilogue.

7.2 Avantages de la coopération

Voyons à présent, de quelle manière cette coopération nous permet de cumuler les avantages de chaque type de raisonnement. Nous verrons aussi comment nous avons pu remédier aux inconvénients qui leurs sont inhérents.

A cet effet, rappelons tout d'abord que l'avantage principal du raisonnement de surface se situe au niveau de son efficacité ou encore sa rapidité d'exécution. Or c'est ce type de raisonnement qui est utilisé à la première étape pour localiser les composants-cause.

Rappelons ensuite que la limite majeure liée à ce type de raisonnement est son incapacité à diagnostiquer les pannes non prévues à l'avance. A cet effet, il suffit de remarquer que l'avantage principal du raisonnement profond est justement de remédier à cette limite,

ce qui nous conduit à utiliser ce type de raisonnement à la deuxième étape pour déterminer les suspects.

Remarquons enfin que le but de la troisième étape de justification est de remédier à la limite du raisonnement profond, à savoir la possibilité d'aboutir à un diagnostic érroné. Cette étape vient alors "valider" le diagnostic fourni par le raisonnement profond à la seconde étape. Mais, ici on est face à la difficulté d'énumération des règles. C'est pourquoi nous avons proposé les schémas de règles: les paradigmes.

7.3 Un scénario typique

Pour fixer les idées, donnons un scénario typique du déroulement d'un diagnostic. Notre but ici étant limité à montrer schématiquement ce déroulement, nous prendrons un exemple fictif évitant ainsi la lourdeur du vocabulaire technique d'un exemple réel. Considérons la figure 3.22 qui représente une partie de la hiérarchie principale des composants d'une installation donnée.

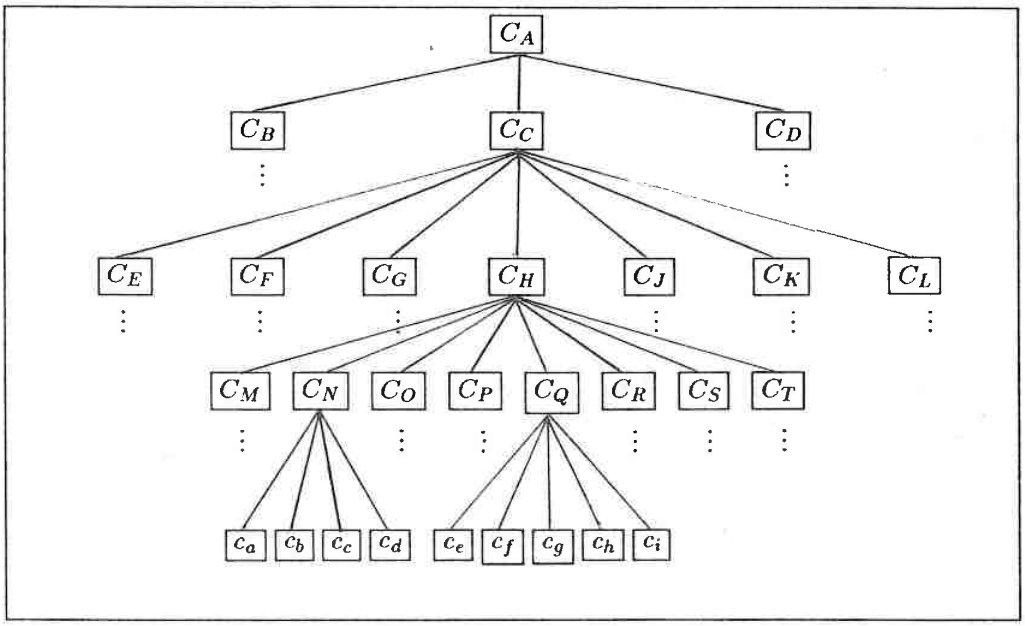


Figure 3.22: une partie d'une hiérarchie principale de composants d'une installation.

Supposons que les lois $L_1^{C_A}$, à priorité égale à 3, et $L_2^{C_A}$, à priorité de 1, du composant C_A ne sont plus vérifiées. Ce composant devient donc un composant-effet. Voyons alors le déroulement du mécanisme:

La loi $L_2^{C_A}$ ayant la plus haute priorité, c'est elle qui sera considérée en premier lieu. Supposons qu'au niveau de la spécification du diagnostic de C_A, c'est la stratégie "focus($\{C_B, C_D\}$)" qui est appliquée mais que celle-ci échoue. Aucune autre stratégie n'est en attente, c'est la loi $L_1^{C_A}$ qui est considérée et les stratégies, qui lui sont associées, sont examinées. De nouveau, le mécanisme sélectionne la stratégie ayant la plus haute priorité. Supposons que l'application de cette stratégie désigne , avec succès, le composant C_C comme un composant-cause.

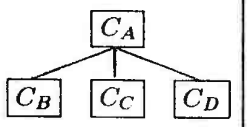


Figure 3.23: déroulement du diagnostic: étape 1.1 de localisation.

En considérant C_C comme un composant-effet, le mécanisme se poursuit:

Après un traitement analogue à celui réalisé plus haut, admettons que c'est C_H qui a été sélectionné comme composant-cause.

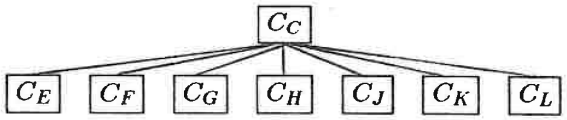
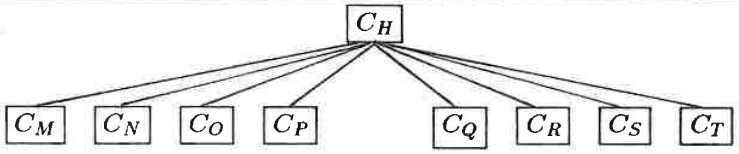


Figure 3.23 (suite): étape 1.2 de localisation.

De nouveau, on considere C_H comme un composant-effet, le mécanisme de diagnostic se poursuit:



Ici ce sont les composants C_N et C_Q qui ont été sélectionnés par la stratégie mise en oeuvre. A ce niveau, admettons que la priorité associée aux lois non vérifiées de C_N est la même que celle associée aux lois non vérifiés de C_Q. C'est la priorité associée au diagnostic qui va être considérée comme critère de sélection. Supposons que c'est C_Q qui est sélectionné.

Figure 3.23 (suite): étape 1.3 de localisation.

C'est l'étape de détermination des suspects qui va entrer en jeu.

A ce niveau c'est la tactique spécifiée au niveau de C_Q qui est mise en oeuvre. Admettons que celle-ci génère l'ensemble: $\{c_e, c_f, c_h\}$ comme l'ensemble des composants déclarés en panne.

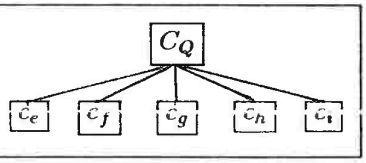


Figure 3.23 (suite): étape 2.1 de détermination des suspects.

A présent, il s'agit de valider le résultat de la tactique par l'étape 3.1, c'est-à-dire, mettre en oeuvre le mécanisme de justification pour culpabiliser ou innocenter les composants c_e, c_f et c_h. Ceci est réalisé en utilisant les paradigmes spécifiés au niveau de chacun

de ces composants. Admettons qu'à cause d'une sous-information, le mécanisme n'arrive ni culpabiliser les composants, ni à les innocenter. Puisqu'il existe encore un choix en attente, le mécanisme de diagnostic fait un retour arrière et sélectionne le composant C_N.

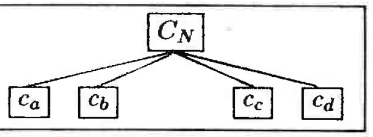
De manière analogue à l'étape 2.1, on applique la tactique spécifiée au niveau de ce composant. Admettons que celle-ci déclare le composant c_b en panne.	

Figure 3.23 (suite): étape 2.2 de détermination des suspects.

Nous atteignons alors l'étape 3.2 qui, de nouveau, mettra en oeuvre le mécanisme de justification en oeuvre pour culpabiliser ou innocenter le composant c_b. En admettant que le mécanisme de justification confirme la culpabilité de c_b, le résultat est communiqué à l'utilisateur.

8 En résumé

Dans ce chapitre nous nous sommes efforcés de définir clairement des concepts (e.g symptôme, panne) qui sont d'une importance capitale en diagnostic. Nous avons alors défini des stratégies pour explorer progressivement la hiérarchie des composants abstraits. Certaines de ces stratégies utilisent des graphes construits selon une relation donnée. La construction de ces graphes est réalisée au préalable, c'est-à-dire à l'étape de spécification de l'installation (Cf. chapitre 1). Ensuite, ce sont les tactiques qui sont définies. De nouveau les tactiques, visant à déclarer les composants corpusculaires en panne, utilisent des hiérarchies et des graphes préalablement définies. Enfin, nous avons proposé des paradigmes pour confirmer ou infirmer la culpabilité des composants.

Les stratégies, les tactiques et les paradigmes étant définis, nous avons alors procédé à l'étape de la spécification du diagnostic des composants. Cette étape nous a permis ainsi de compléter les spécifications descriptives des composants obtenues au chapitre 1. Par la suite et en se basant sur une coopération entre un raisonnement de surface et un raisonnement profond, nous avons alors proposé un mécanisme de diagnostic.

Nous disposons maintenant des spécifications de description et celles de diagnostic des composants de l'installation ainsi qu'un mécanisme de diagnostic. Il s'agit alors de voir comment construire le système de diagnostic de l'installation spécifiée. C'est l'objectif du chapitre suivant.

Chapitre 4

Construction des systèmes de diagnostic

Nous disposons actuellement des méthodes de spécification et d'élaboration du diagnostic des installations. Nos objectifs sont, maintenant d'une part, de proposer les outils nécessaires pour la construction et la réalisation des systèmes de diagnostic: c'est-à-dire la mise en oeuvre des méthodes proposées et la construction de ces systèmes, d'autre part, de confronter l'ensemble des outils et des méthodes proposées à des cas réels. Mais, pour qu'ils soient utiles, ces outils doivent veiller, sans pour autant qu'ils soient contraignants, à la bonne conduite *des méthodes proposées. Ils doivent également encourager la* réutilisation *des différentes parties (i.e structure, comportement, lois, etc) des spécifications, ce qui nous conduit à nous intéresser à l'interface entre ces outils et l'utilisateur. Cet interface doit être* convivial, interactif *et distinguer différents* modes *d'exploitation. De manière duale, nous nous interrogerons sur l'organisation des connaissances utilisées par ces outils. Cette organisation doit être* modulaire *pour faciliter la réutilisation des différentes parties des spécifications.*

*Avec ces objectifs, nous introduisons tout d'abord le concept d'***environnement** *de* **spécification** *et de* **construction** *des* **S**ystèmes **I**nteractifs de **D**iagnostic **I**ndustriel. *Ensuite, nous traiterons l'aspect expérimentation sur des cas réels et présenterons un exemple d'application. Nous donnerons enfin un panorama sur les systèmes de diagnostic réalisés. La préoccupation majeure de ce chapitre se situe donc tant au niveau de la recherche des outils méthodologiques qu'au niveau de l'expérimentation et répond ainsi aux préoccupations initiales de ce travail.*

Ainsi le premier paragraphe sera consacré à définir les différentes constituantes d'un environnement. Au paragraphe 2, nous donnerons l'organisation des connaissances en bibliothèques et en catalogues. Quant à la structuration des outils en différents modules fonctionnels, elle sera introduite au paragraphe 3. Au paragraphe 4, nous traiterons IOTA, *un sous-système industriel de mesure et de découpe de produits longs d'une installation de laminage. Un exemple de session de diagnostic effectuée avec le prototype SIDI sera également donnée. Au paragraphe 5, nous donnerons un panorama des systèmes de diagnostic opérationnels ou expérimentaux existants dans la littérature. Quant au paragraphe 6, il sera consacré à nos conclusions.*

1 Vers l'élaboration d'un environnement

De nombreuses recherches ont été et sont conduites sur le concept d'environnement de programmation [Borras 87,Habermann 81,VanLamsweerde 82]. Suivant les travaux, le but poursuivi par un environnement peut se limiter à fournir une aide à l'édition, à la mise au point et à la maintenance des programmes en général. Dans ce cas, l'environnement peut être vu comme une boîte à outils mise à la disposition de l'utilisateur. Ces outils peuvent être passifs, c'est-à-dire qu'ils ne déclenchent aucun traitement de manière automatique, comme par exemple ceux qu'on trouve généralement dans UNIX. Ils peuvent également être actifs comme par exemple les éditeurs dirigés par la syntaxe. Plus ambitieux encore, d'autres travaux visent à couvrir les différents cycles de vie des logiciels: spécification, conception, développement, programmation, et maintenance. Dans ce cas, on parle alors d'atelier de génie logiciel [Godart 86]. Deux critères peuvent être utilisés pour caractériser ces ateliers. Le premier critère est relatif à la classe de problèmes à résoudre. On trouve ainsi des ateliers spécialisés pour la conception de certains types d'applications de gestion, de contrôle de procédés industriels, etc ... Au contraire, d'autres ateliers se veulent non spécialisés et ne font aucune restriction sur le type d'application à considérer. Le second critère est au niveau méthodologique, les ateliers peuvent alors être sans méthode ou au contraire dotés d'une, voire de plusieurs méthodes.

Avec un objectif plus modeste, nous ne visons pas un tel atelier mais seulement un *environnement* bâti autour d'outils et de méthodes pour la spécification et la construction des systèmes de diagnostic industriel. Mais avant d'exposer nos propositions sur un tel environnement, il convient de définir quelles sont les constituantes d'un environnement [Jacquot 84] et surtout quel est le rôle de chacune. On peut voir trois constituantes:

La première constituante est **méthodologique**. Elle doit comprendre d'une part une ou plusieurs méthodes à conduire pour analyser les problèmes, d'autre part le(s) langage(s) d'expression pour décrire l'univers du problème posé. Cette constituante est sans doute capitale dans un environnement quelconque, c'est pourquoi elle a été largement développée tout au long des chapitres précédents.

La seconde constituante est **fonctionnelle**. Il s'agit de proposer un ensemble d'outils relatifs à l'édition, à la vérification[1], à la traduction ainsi qu'à l'interprétation des spécifications. Toutefois pour veiller, sans contraindre, à la bonne conduite des méthodes proposées, il importe que ces outils soient à la fois *cohérents* et *intégrés*.

La dernière constituante est **ergonomique**. Ici, le point central est relatif à l'aspect communication[2] homme/machine. On retrouve alors essentiellement deux composantes

[1] Outre l'analyse syntaxique classique des spécifications, ici nous entendons par vérification, des contrôles analogues à ceux réalisés par la commande *lint* de UNIX sur des programmes écrits en C.

[2] Signalons que cet aspect ne sera pas très approfondi ici.

intimement liées. Tout d'abord la composante interface entre l'utilisateur et la machine ensuite la composante guide ou pilotage de l'ensemble des outils.

Après ce bref aperçu, voyons à présent nos propositions quant à l'organisation générale des deux dernières constituantes: fonctionnelle et ergonomique.

1.1 La constituante fonctionnelle

L'architecture proposée dans la figure 4.1 fait apparaître une structuration des connaissances en différentes bibliothèques. Ces dernières sont, à leur tour, structurées en catalogues permettant ainsi une organisation arborescente de l'ensemble des connaissances.

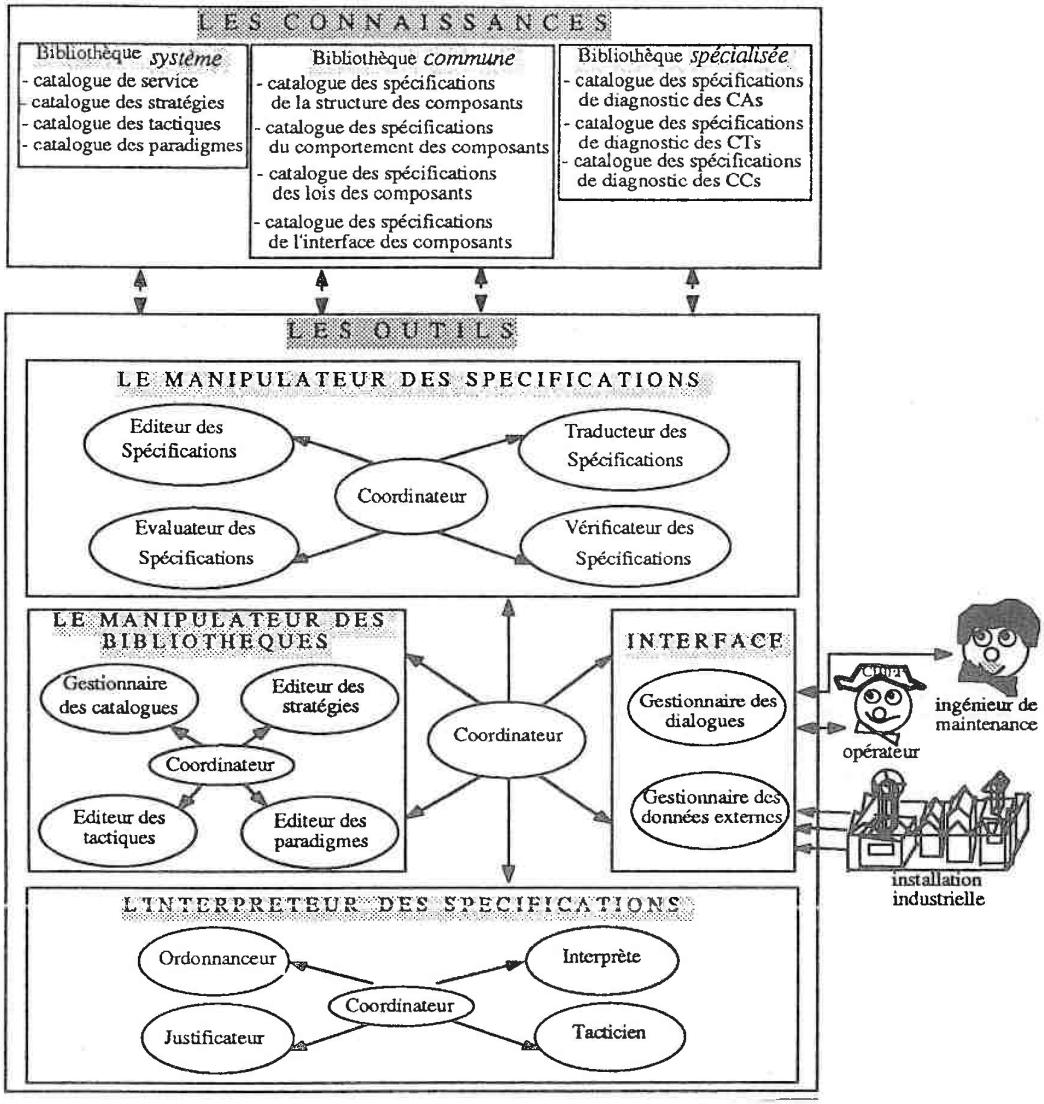


Figure 4.1: architecture générale proposée.

Nous avons ainsi:

- une bibliothèque *système* composée de quatre catalogues. Tout d'abord un catalogue de service pour stocker les données et les mesures en provenance des capteurs de l'installation ainsi que des informations concernant l'état proprement dit du système. Ensuite trois autres catalogues pour stocker les définitions des stratégies, des tactiques et des paradigmes.

- une bibliothèque *commune* composée de quatre catalogues pour stocker les spécifications de la structure, du comportement, des lois et de l'interface de chacun des composants de l'installation.

- une bibliothèque *spécialisée* composée de trois catalogues pour stocker les spécifications des stratégies pour les composants abstraits, des tactiques pour les composants terminaux et des paradigmes pour les composants corpusculaires.

Quant à la gestion de ces connaissances, elle est assurée par les outils suivants: le manipulateur des spécifications, le manipulateur des bibliothèques, l'interpréteur des spécifications et l'interface. Chacun de ces outils est, à son tour, structuré en plusieurs modules contrôlés par un coordinateur. On a:

- *le manipulateur des spécifications* constitué de quatre modules: les deux premiers modules permettent l'édition et la vérification des spécifications. Le troisième assure la traduction des spécifications vers des fonctions Lisp. Le quatrième module permet l'évaluation de manière locale et pas à pas des S-expressions LISP générées par le module de traduction.

- *le manipulateur des bibliothèques* constitué également de quatre modules: on trouve trois modules réservés respectivement à l'édition des stratégies, des tactiques et des paradigmes. Le quatrième module est consacré à la gestion des différents catalogues.

- *l'interpréteur des spécifications* est constitué à son tour de quatre modules: un premier module prend en charge l'interprétation des spécifications, le module ordonnanceur vient ensuite: il gère une structure de données "agenda" qui planifie l'activité de l'interprète. Quant au module tacticien, il prend en charge la mise en oeuvre des tactiques. Enfin, le module justificateur prend en charge les paradigmes.

- *l'interface* est constitué de deux modules: tout d'abord le gestionnaire des données externes qui assure l'acquisition et le prétraitement des données en provenance des capteurs, thermo-couples, etc et ensuie le module gestionnaire des dialogues qui assure, quant à lui, la communication avec les différents utilisateurs: les opérateurs, les agents de maintenance, etc.

Après avoir donné le rôle de chacun des modules, notre tâche, à présent, consiste à:
- présenter, dans un premier temps, la constituante ergonomique et donner les choix effectués sur l'interface.
- donner, par la suite, la représentation des spécifications, des stratégies, des tactiques, etc dans les différents catalogues.
- spécifier, enfin, les entrées, les sorties de chacun des modules et montrer ainsi l'interaction entre les modules et les connaissances.

1.2 La constituante ergonomique

Comme nous l'avons dit, deux composantes peuvent être distinguées dans cette constituante:

L'objectif poursuivi par la première est une aide à l'organisation et l'enchaînement des différentes tâches ainsi que la prise en charge des activités secondaires et fastidieuses pour l'utilisateur. Il est clair que satisfaire pleinement cet objectif nécessite une étude approfondie qui risquerait de dépasser le cadre de ce travail. Aussi nous nous limitons ici à mettre en place des coordinateurs pour assurer l'enchaînement des différents modules de chaque outil.

Quant à la seconde composante de cette constituante, son objectif est de mettre en place un interface adapté entre les outils et les utilisateurs. A défaut d'un interface graphique[3] qui conviendrait parfaitement ici, nous mettons l'accent sur la nécessité de distinguer plusieurs modes d'exploitation du système de diagnostic et, de ce fait, différents types de dialogues. Or, dans les chapitres précédents nous avons mis en évidence:

- d'une part, le suivi des différentes étapes conduisant à des spécifications complètes. Mais, si notre présentation est faite de manière séquentielle, il est clair qu'en réalité il n'en pas ainsi et que de nombreux retours arrières sont souvent nécessaires, d'ou la nécessité d'un mode *interactif* à cette activité qu'est la spécification.

- d'autre part, l'enrichissement des différents catalogues de stratégies, tactiques et paradigmes par l'utilisateur au fur et à mesure des cas rencontrés. Cet enrichissement, qui apparaît ainsi comme une garantie de la qualité d'extensibilité, nécessite ainsi un mode spécifique.

Nous distinguons enfin un troisième mode: l'exploitation du système pour conduire un diagnostic. Nous avons donc:

Un premier mode, appelé *spécification*, qui est réservé aux agents de maintenance de l'installation. Avec ce mode, qui met en jeu à la fois le manipulateur des spécifications et

[3]Ce type d'interface peut alors être d'une grande aide pour la description des hiérarchies et des graphes.

celui des bibliothèques, on doit pouvoir créer, modifier, supprimer et d'une façon générale manipuler et éditer les spécifications des composants.

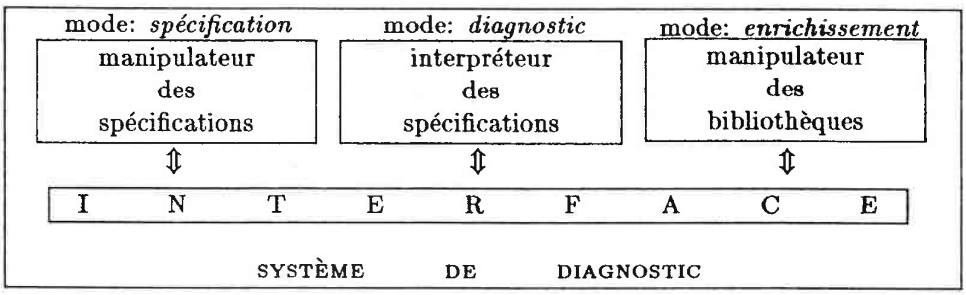


Figure 4.2: les différents modes d'exploitation.

Un second mode appelé *enrichissement*, qui utilise le manipulateur des bibliothèques, est réservé à une catégorie particulière d'agents de maintenance pour l'édition et l'adjonction des stratégies, des tactiques et des paradigmes.

Quant au dernier mode appelé *diagnostic*, il utilise l'interpréteur des spécifications et le manipulateur des bibliothèques. Il constitue donc le mode d'exploitation du système. Il est utilisé essentiellement par les opérateurs et les dépanneurs.

Maintenant avant de donner un exemple de session de travail avec le prototype *SIDI* et de préciser ce qui a été réalisé (§4), voyons nos propositions quant à l'organisation des connaissances en bibliothèques (§2) ainsi que la structuration des outils en modules (§3).

2 Les connaissances

Comme nous l'avons signalé (§1.1), les connaissances sont structurées de manière arborescente en différentes bibliothèques de catalogues. On a ainsi:

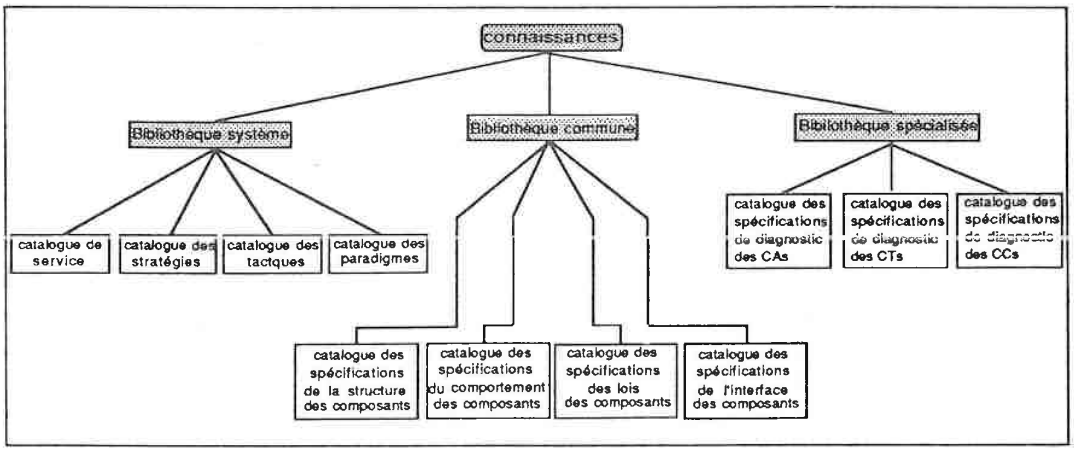


Figure 4.3: structuration arborescente des connaissances.

Contrairement aux spécifications du diagnostic qui peuvent être exploitées uniquement pour le diagnostic des pannes, les spécifications descriptives[4] peuvent être *réutilisées* pour conduire un autre type de diagnostic: le contrôle de qualité par exemple. Autrement dit, admettons que l'on veuille mettre en place une méthode de diagnostic pour contrôler la qualité des produits fabriqués dans l'installation, on est alors conduit tout d'abord à construire une nouvelle bibliothèque spécialisée qui contient les connaissances de contrôle de qualité, à réutiliser par la suite la hiérarchie de l'installation déjà construite ainsi que les spécifications descriptives écrites auparavant: c'est-à-dire le contenu de la bibliothèque commune.

Dans les paragraphes suivants, nous verrons que la poursuite de cette démarche nous conduit à construire une spécification d'un composant donné par simple réutilisation des différentes parties des spécifications d'autres composants déjà écrites.

2.1 La bibliothèque système

LE CATALOGUE DE SERVICE

Dans ce catalogue nous avons:

D'une part les structures des données qui sont utilisées uniquement par le système. De ce fait, elles sont complétement ignorées par les utilisateurs. Parmi ces données, on trouve notamment la structure de données *agenda* dont la complexité dépend de l'ampleur de l'installation. Cette structure, représentée sous forme d'arbre et jouant le rôle de "carnet-de-bord" pour le système, comprend en particulier:

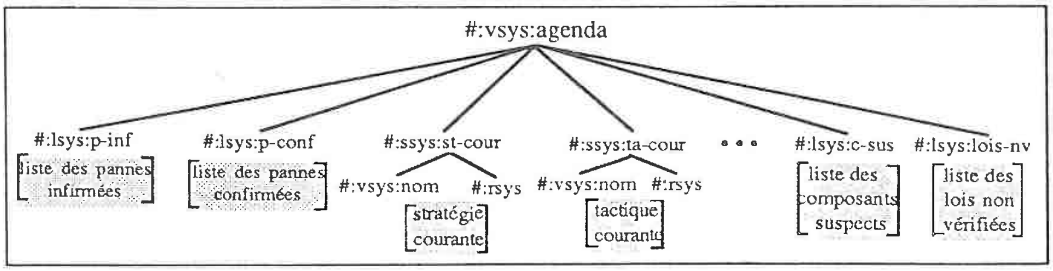


Figure 4.4: exemple de structure de données du système.

D'autre part, nous avons les structures des données qui représentent les mesures en provenance des capteurs et sont ainsi très dépendantes de l'installation considérée. Aussi nous ne nous y attardons pas ici. Notons toutefois les conventions utlisées par la suite:

[4]Rappelons que ces spécifications incluent la structure, le comportement, les lois et l'interface des composants.

- #:vsys est le préfixe d'une variable système.
- #:lsys est le préfixe d'une liste système.
- #:ssys est le préfixe d'une structure système.
- #:psys est le préfixe d'un paramètre système.
- #:fsys est le préfixe d'une fonction système.

LE CATALOGUE DES STRATEGIES

Comme le montre la figure 4.5, les stratégies sont organisées en une structure d'arbre. L'avantage de cette organisation réside dans la facilité de mise à jour de cette structure des données qu'est l'arbre. Par exemple, enrichir ce catalogue par une nouvelle stratégie revient à ajouter simplement un noeud à cet arbre.

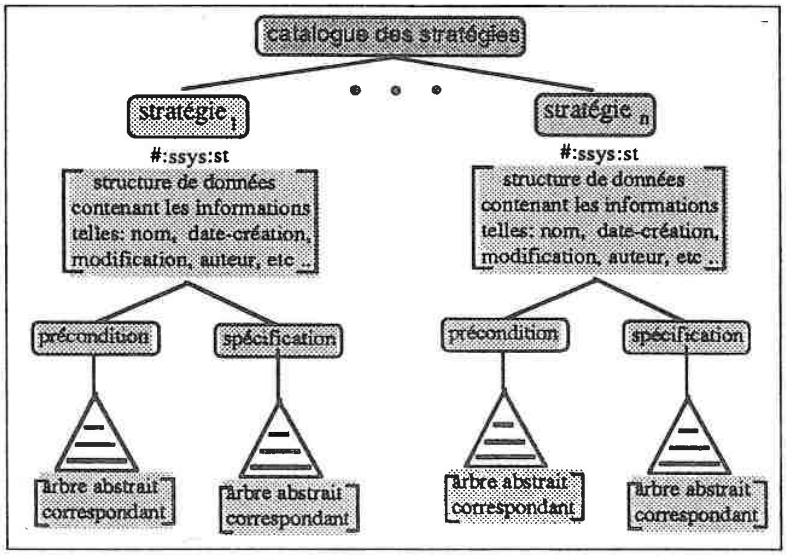
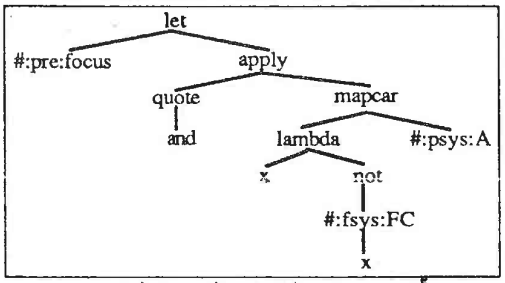


Figure 4.5: organisation des stratégies dans le catalogue.

A titre d'exemple, prenons la stratégie "focus(A)" (Cf. §2.1 du c hapitre précédent). La précondition de cette stratégie étant: $x \in A \Rightarrow \neg FC(x)$. Nous avons alors:



(a) représentation interne[5]

```
(let ((#:pre:focus
  (apply (quote and)
    (mapcar
      (lambda (x) (not (#:fsys:FC x)))
       #:psys:A)
    )
  )
```

(b) S-expression correspondant

Figure 4.6: représentation d'une stratégie.

[5]L'arbre abstrait étant peu lisible, nous nous contenterons ici d'une représentation simplifiée mais lisible.

Remarque: l'organisation du catalogue des tactiques et celui des paradigmes étant similaire à ce catalogue, nous omettrons de les présenter ici. Toutefois nous les détaillerons à l'addenda A.

2.2 La bibliothèque commune

Quatres arbres sont construits pour construire les les spécifications partielles des composants. Les noeuds du premier arbre représentent les spécifications de la structure des composants. Les noeuds du second arbre représentent quant à eux les spécifications du comportement des composants. Les noeuds du troisième représentent les spécifications des lois des composants. Finalement, les noeuds du quatrième arbre représentent les spécifications de l'interface des composants. Remarquons que ce choix de construire quatre arbres, plutôt q'un seul représentant toutes les spécifications partielles, contribue à faciliter la réutilisation des parties des spécifications. A titre d'exemple, supposons que l'on ait déjà la spécification partielle du composant pompe electrique et que, actuellement, on est conduit à spécifier le composant pompe hydraulique. On peut alors, par simple accès au premier arbre des spécifications structurelles, réutiliser la spécification de la structure du composant pompe electrique.

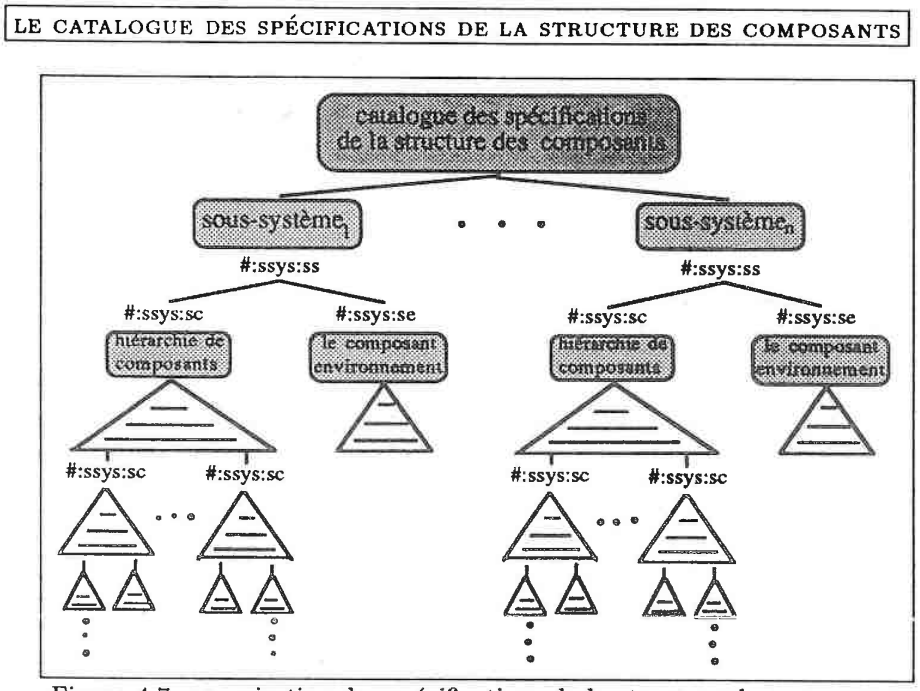


Figure 4.7: organisation des spécifications de la structure des composants.

Comme nous l'avons indiqué au paragraphe 1.2 du premier chapitre, en fonction de la complexité de l'installation, on est parfois conduit à partitionner cette installation en plusieurs sous-systèmes industriels, à associer ensuite à chacun d'entre eux un environnement

local.

Cet environnement n'est rien d'autre qu'un composant comme les autres. C'est ce que nous avons ici à la figure 4.7. Pour illustrer la spécification de la structure des composants, nous prenons le composant "pompe" dont la structure a été donnée à la figure 1.11 du premier chapitre.

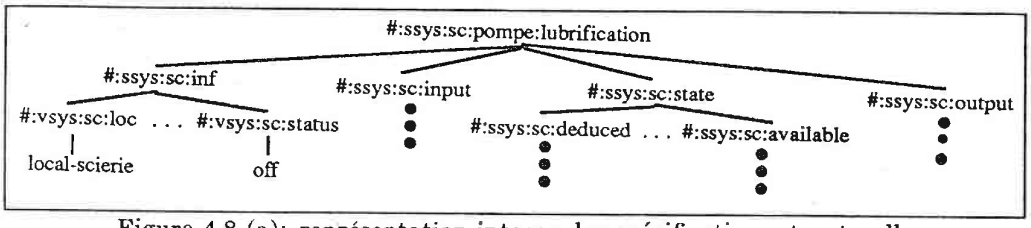


Figure 4.8 (a): représentation interne des spécifications structurelles.

Remarquons que cette réprésentation arborescente est formée de quatre noeuds: le premier est reservé aux informations générales, le second aux ports d'entrée, le troisième aux variables d'état et enfin le quatrième aux ports de sortie. Chacun de ces noeuds est à son tour subdivisé en d'autres sous noeuds ... etc

```
(de #:fsys:sc:pompe:lubrification (#:psys)
   (let ( ...)
    (setq #:vsys:loc:pompe:lubrification 'local-scierie)
    (setq #:vsys:status:pompe:lubrification 'off)
     ...
    (setq #:vsys:pression:pompe:lubrification ())
    (putprop '#:vsys:pression:pompe:lubrification 'available 'classe)
     ...
   )
)
```

Figure 4.8 (b): *S*-expressions correspondant à l'exemple.

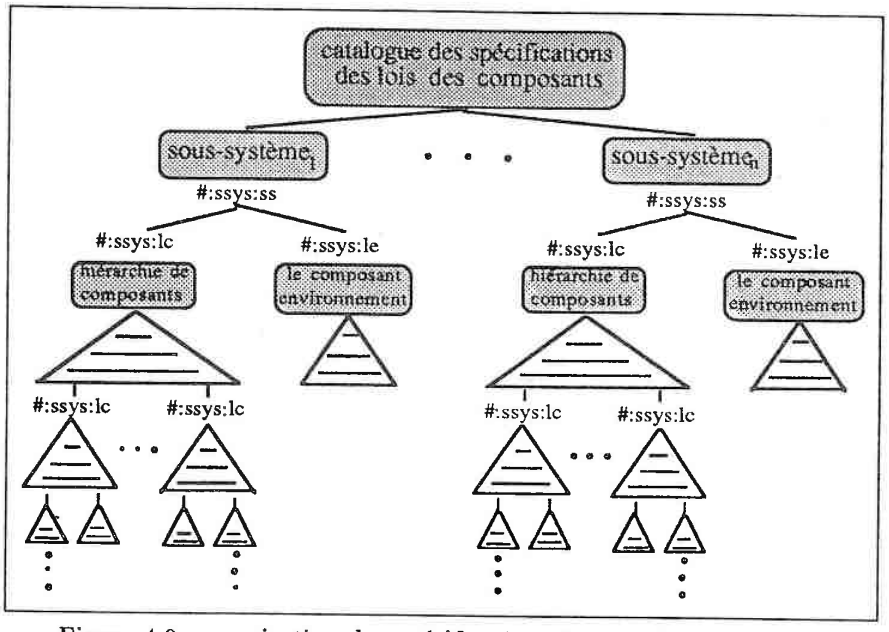


Figure 4.9: organisation des spécifications des lois des composants.

De nouveau, nous avons une structure qui reflète la demarche poursuivie au paragraphe 4 du chapitre 1 pour spécifier les lois.

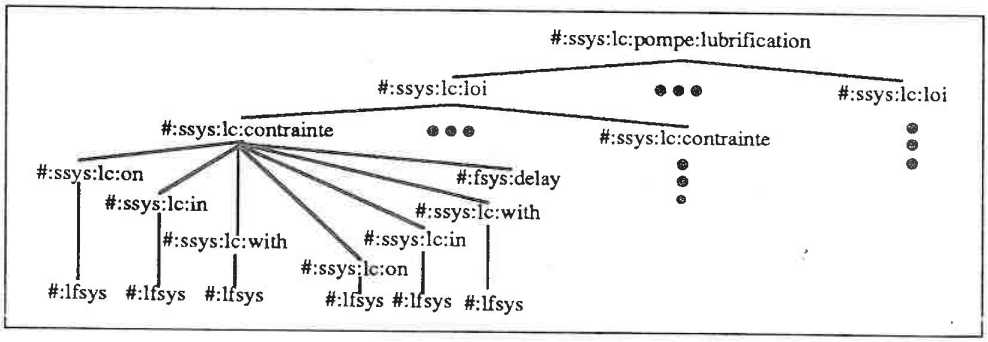


Figure 4.10 (a): représentation interne des spécifications des lois.

L'exemple choisi ici est le composant pompe dont les lois ont été spécifiées à la figure 1.17 du chapitre 1.

```
(dc #:fsys:lc:pompe:lubrification (#:psys)
  (let ( ...)
    ; loi3
    (cond  ((and(#:fsys:increase #:pression:pompe:lubrification)
                (#:fsys:steady #:vsys:puissance:pompe:lubrification))
            (if (#:fsys:decrease(#:fsys:get '#:fsys:debit:pompe:lubrification))
                () (#:fsys:signal '#:vsys:loi3:pompe:lubrification)
            )
           ...
    )
    ...
)))
```

Figure 4.10 (b): S-expressions correspondant à l'exemple.

Remarque: de nouveau, les catalogues des spécifications du comportement et de l'interface des composants seront présentés à l'addenda A.

2.3 La bibliothèque spécialisée

Pour des raisons analogues à celles données au paragraphe 2.2 nous construisons trois arbres: un premier arbre dont les noeuds représentent les spécifications de diagnostic, en terme de stratégies, pour les composants abstraits; un second dont les noeuds représentent les spécifications de diagnostic, en terme de tactiques, pour les composants terminaux; finalement un troisième arbre dont les noeuds représentent les spécifications de diagnostic, en terme de paradigmes, pour les composants corpusculaires. Cette distinction est d'autant plus justifiée que chacune de ces connaissances est utilisée par un module particulier. C'est ce que nous verrons au paragraphe 3 qui expose les différents modules.

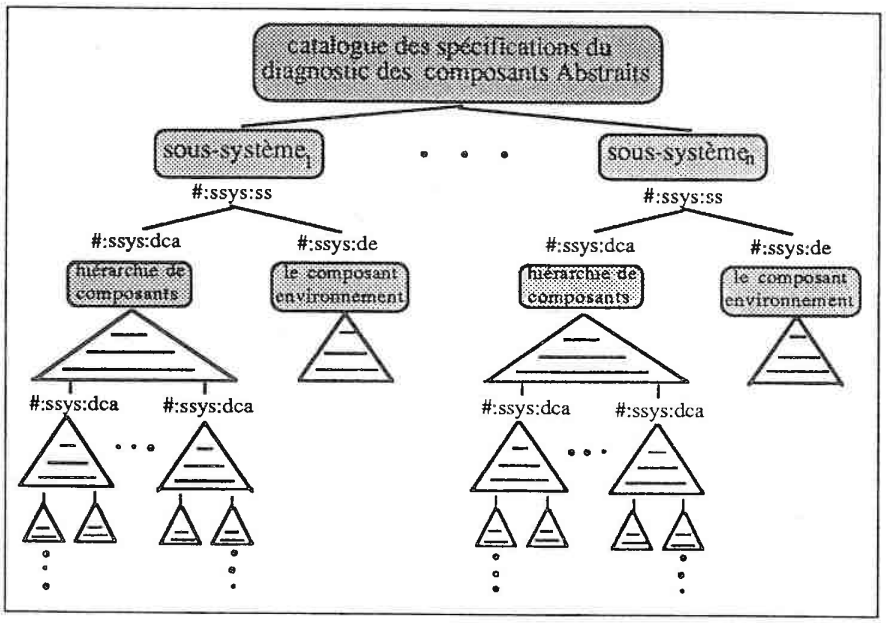


Figure 4.11: organisation des spécifications du diagnostic des composants abstraits.

En considérant l'exemple du composant pompe, dont la spécification du diagnostic a été donnée à la figure 3.19 du chapitre 3, nous avons:

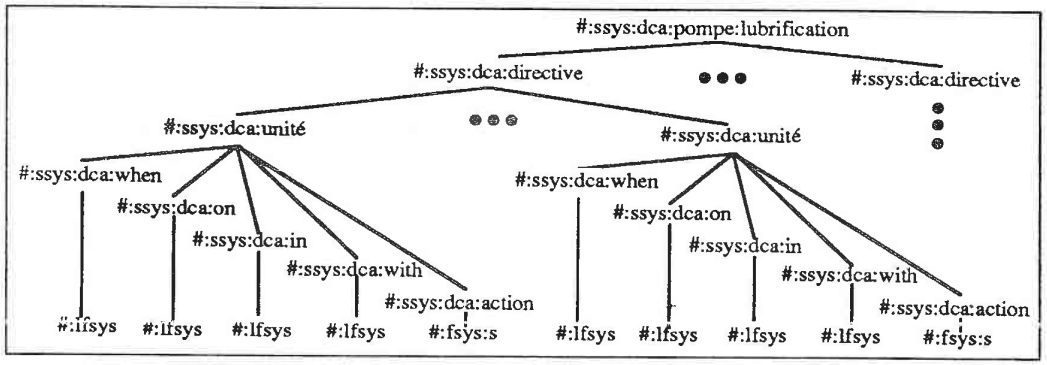


Figure 4.12 (a): représentation interne des spécifications du diagnostic des $\mathcal{CA}$s.

De nouveau, on peut remarquer que cette structure reflète la démarche de spécification de diagnostic poursuivie au paragraphe 6 du chapitre 3.

```
(de #:fsys:dca:pompe:lubrification (#:psys)
   (let ( ...)
        ...
        ; directive diag 2
        (cond  ((#:fsys:uncheck '#:vsys:loi1:pompe:lubrification )
                 (#:fsys:focus '#:vsys:pompe:moteur-pompe))
               (t          nil)
        )
        ...
   )
)
```

Figure 4.12 (b): *S*-expressions correspondant à l'exemple.

Remarque: les catalogues des spécifications du diagnostic des composants terminaux et celui des composants corpusculaires sont détaillés à l'addenda A.

3 Les outils

D'après ce qui précède, il apparaît clairement que la principale structure de données que nous avons ici est l'arbre. De ce fait, il est nécessaire de disposer des différentes fonctions de base pour gérer l'ensemble des arbres. Parmi ces fonctions, on retrouve celles dont le but est de créer les arbres, d'autres pour les parcourir, les modifier, etc .... Une étude de ces fonctions que nous avons effectuée antérieurement [ElAyeb 84] a fait ressortir plusieurs familles de fonctions dont:

- la famille *construction*: le but de cette famille est de construire les différents arbres de spécifications de la structure, du comportement, ... des composants.
- la famille *parcours*: le but de cette famille est de parcourir les arbres construits par la famille précédente.
- la famille *manipulation*: le but poursuivi ici est de permettre la modification de la structure d'un arbre par l'adjonction ou la suppression d'un noeud par exemple. Elle doit permettre également l'accès aux differentes informations dans les noeuds et les feuilles des arbres.
- la famille *analyse*: ces fonctions, obtenues par un générateur d'analyseurs syntaxiques tel que Leyacc [Berry 84], prennent en charge l'analyse syntaxique des spécifications. En fait, cette famille de fonctions peut être vue comme une sous-famille de la famille construction.

Voyons à présent d'une part l'utilisation de ces familles par les différents modules qui composent les outils, d'autre part l'interaction entre les connaissances, c'est-à-dire les catalogues, et les modules:

3.1 Le manipulateur des spécifications

A cet outil correspond le mode *"spécification"* que nous avons identifié au paragraphe 1.2 de ce chapitre. Détailler chacun des modules n'apporterait rien de plus ici. Aussi contenterons nous de donner pour chacun: ses entrées, ses sorties, les catalogues qu'il consulte, ceux qu'il modifie ainsi que les familles de fonctions qu'il utilise.

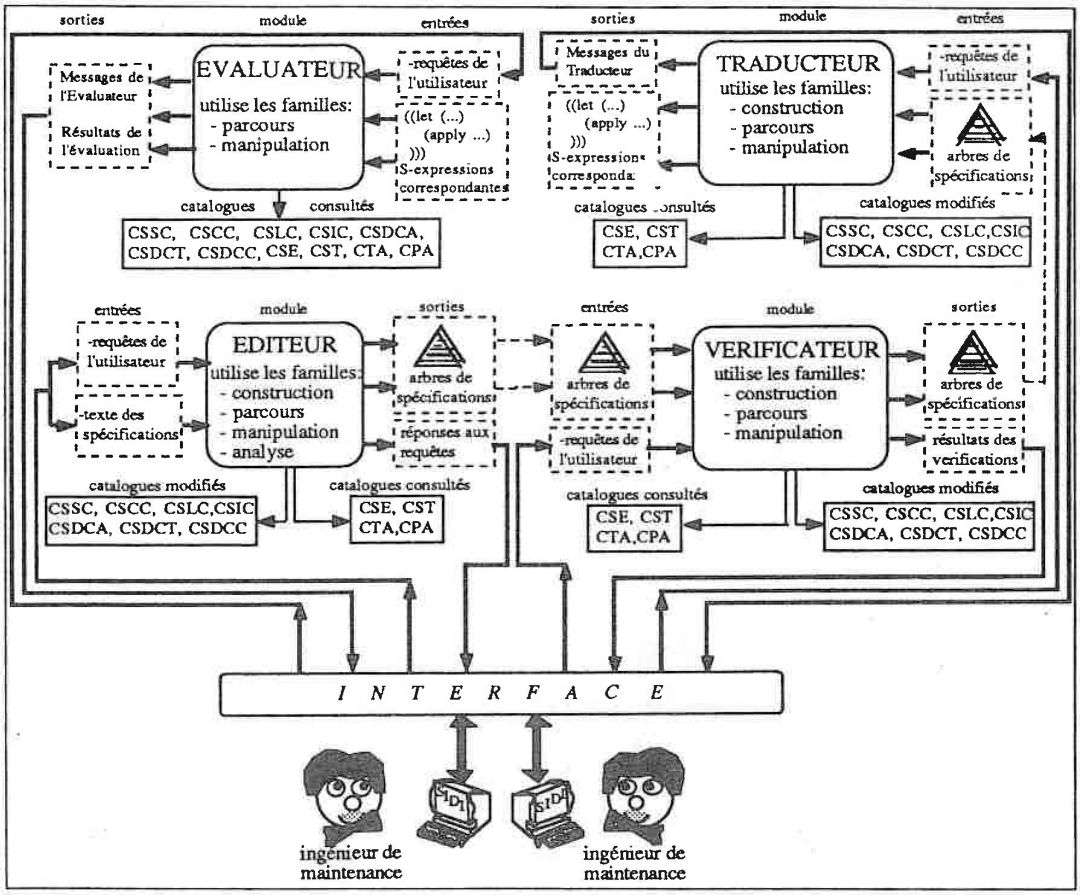


Figure 4.13: les modules du manipulateur des spécifications.

Avec les conventions suivantes utilisées pour les catalogues:

- CSE: catalogue de service.
- CST: catalogue des stratégies.
- CTA: catalogue des tactiques.
- CPA: catalogue des paradigmes.
- CSSC: catalogue des spécifications de la structure des composants.
- CSCC: catalogue des spécifications du comportement des composants.
- CSLC: catalogue des spécifications des lois des composants.
- CSIC: catalogue des spécifications de l'interface des composants.
- CSDCA: catalogue des spécifications du diagnostic des composants abstraits.
- CSDCT: catalogue des spécifications du diagnostic des composants terminaux.
- CSDCC: catalogue des spécifications du diagnostic des composants corpusculaires.

3.2 Le manipulateur des bibliothèques

A cet outil correspond le mode *"enrichissement"* que nous avons identifié au paragraphe 1.2 de ce chapitre. De nouveau, nous adopterons la même démarche de présentation:

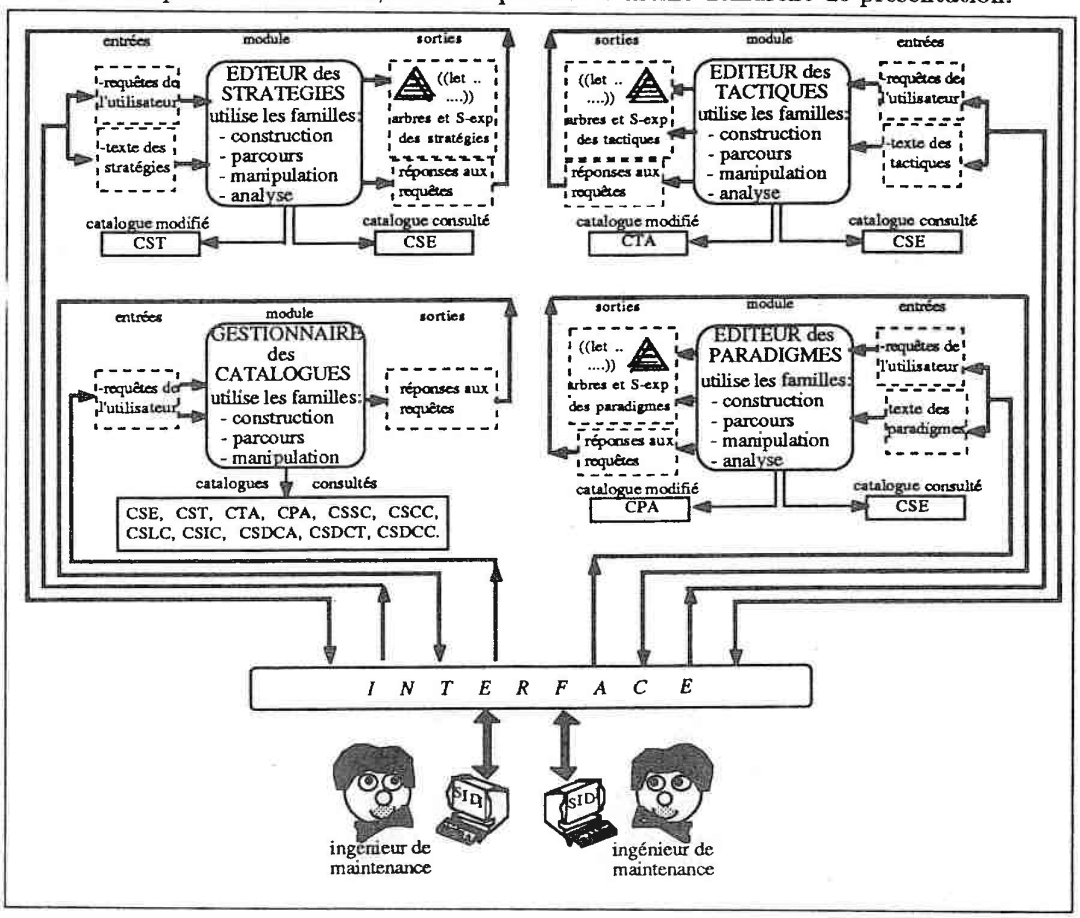


Figure 4.14: les modules du manipulateur des bibliothèques.

- CSE: catalogue de service.
- CST: catalogue des stratégies.
- CTA: catalogue des tactiques.
- CPA: catalogue des paradigmes.
- CSSC: catalogue des spécifications de la structure des composants.
- CSCC: catalogue des spécifications du comportement des composants.
- CSLC: catalogue des spécifications des lois des composants.
- CSIC: catalogue des spécifications de l'interface des composants.
- CSDCA: catalogue des spécifications du diagnostic des composants abstraits.
- CSDCT: catalogue des spécifications du diagnostic des composants terminaux.
- CSDCC: catalogue des spécifications du diagnostic des composants corpusculaires.

3.3 L'interpréteur des spécifications

A cet outil correspond le mode *"diagnostic"* que nous avons identifié au paragraphe 1.2 de ce chapitre. Nous avons:

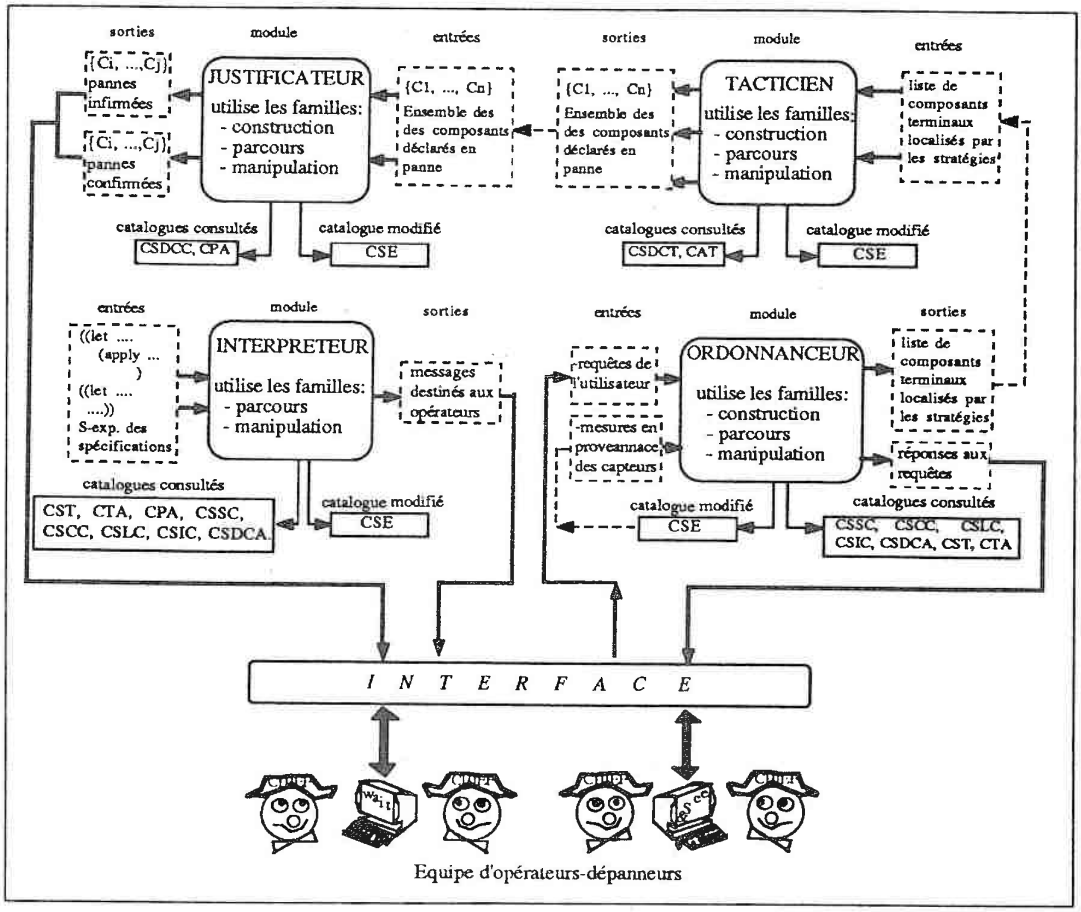


Figure 4.15: les modules de l'interpréteur des spécifications.

- CSE: catalogue de service.
- CST: catalogue des stratégies.
- CTA: catalogue des tactiques.
- CPA: catalogue des paradigmes.
- CSSC: catalogue des spécifications de la structure des composants.
- CSCC: catalogue des spécifications du comportement des composants.
- CSLC: catalogue des spécifications des lois des composants.
- CSIC: catalogue des spécifications de l'interface des composants.
- CSDCA: catalogue des spécifications du diagnostic des composants abstraits.
- CSDCT: catalogue des spécifications du diagnostic des composants terminaux.
- CSDCC: catalogue des spécifications du diagnostic des composants corpusculaires.

4 Expérimentation avec le prototype *SIDI*

L'expérimentation est sans doute une étape importante mais aussi critique. Elle est importante car elle permet d'évaluer les idées soumises. Elle est critique car elle nécessite une analyse rigoureuse permettant de remettre en cause certains choix effectués et d'en confirmer d'autres. Habituellement, l'expérimentation passe par trois stades qui peuvent être à leur tour décomposés en sous-stades.

- tout d'abord, une expérimentation "papier" sur des petits exemples, suivie éventuellent par la réalisation d'une maquette. Au niveau de cette maquette, on peut ignorer certaines qualités telles la convivialité de l'interface par exemple.

- après avoir tiré certaines conclusions sur la maquette, on passe au stade de la réalisation d'un prototype prennant en compte un cas en vraie grandeur. A ce niveau, l'efficacité du prototype, par exemple, peut être ignorée.

- la dernier stade consiste à passer au stade d'un produit final remédiant aux limites du prototype réalisé. Les problèmes tels celui de l'efficacité ou encore de l'interface "on line" doivent alors être résolus à ce niveau.

L'objectif de ce travail n'est pas de fournir un produit final destiné à la diffusion commerciale. De plus, notre préoccupation majeure est avant tout au niveau méthodologique, nous nous sommes arrêtés au second stade. Nous avons alors réalisé le prototype[6] qui compte environ 10.000 lignes LISP. Ce prototype *ne* comprend *pas* tous les outils identifiés à la section précédente, il inclut *seulement* les modules de l'outil d'interprétation des spécifications qui sont nécessaires pour conduire un diagnostic. Toutefois l'architecture que nous avons donnée a été *réellement* suivie, les autres modules peuvent alors être réalisés et adjoints ultérieurement sans aucune difficulté.

Afin d'illustrer *SIDI* au travail, nous allons donner un exemple de session typique réalisé avec ce prototype sur le sous-système industriel de découpe à chaud des produits longs, appelé IOTA, d'une installation sidérurgique de laminage à chaud.

4.1 Description du sous-système industriel IOTA

Il n'est pas question de détailler ici ce sous-système, mais seulement d'en faire une présentation succincte:

[6] Prototype financé en partie par une société industrielle.

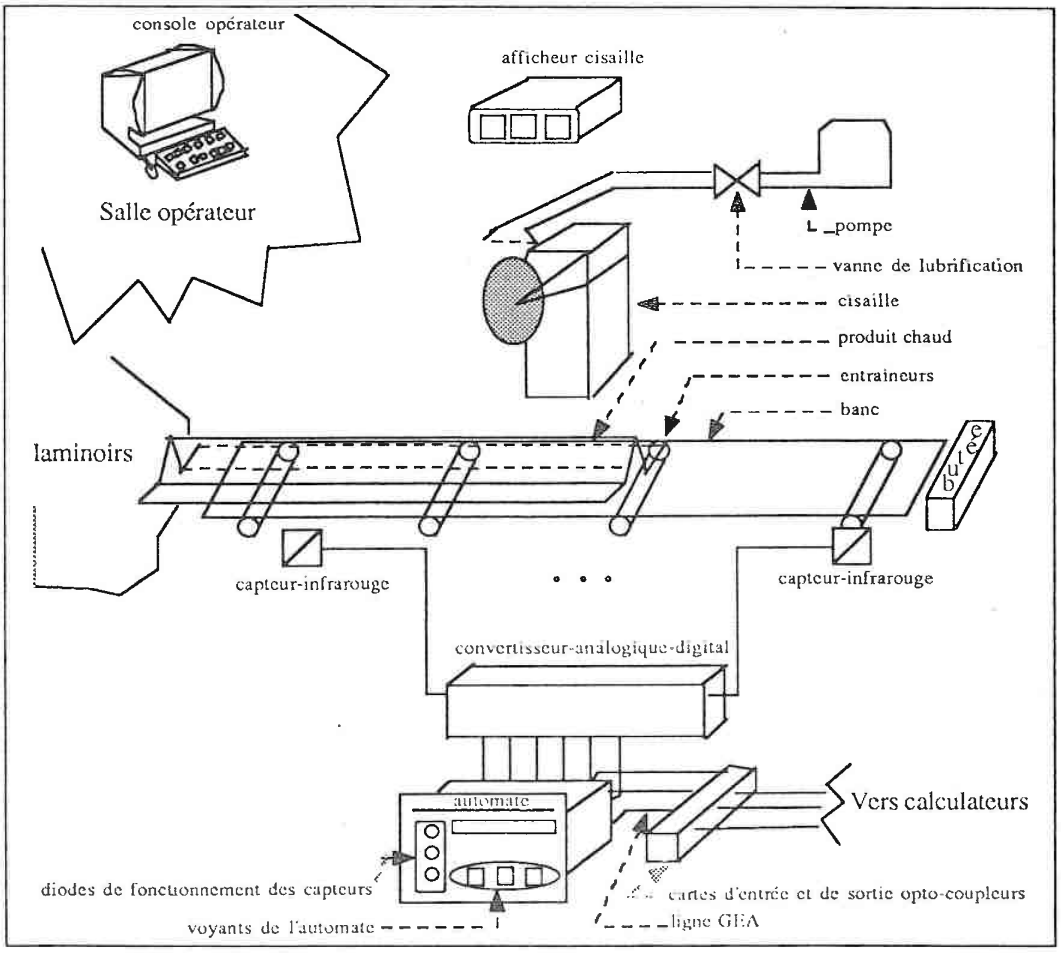


Figure 4.16: le sous-système de l'installation de laminage.

A la sortie de la dernière cage de laminage sont disposés une série de capteurs infrarouges qui détectent la chaleur émise par les produits chauds en mouvement. Ces capteurs sont reliés à un automate qui, à partir d'une séquence et du nombre de signaux obtenus, calcule la longueur du produit. Les produits restant en mouvement continu, arrivent à la butée; c'est au tour du dispositif de cisaille d'entrer en jeu. Ce dispostif recevant les ordres pour une découpe optimale d'un autre calculateur, commande alors le déplacement latéral de la scie pour découper les produits. L'ensemble de ce sous-système fonctionne de manière automatique et en continu. L'agent opérateur supervise l'ensemble des opérations à partir de sa console. Outre le problème des pannes dites franches qui consistent à arrêter tout le sous-système, il existe des pannes dites cachées qui n'arrêtent pas entièrement le sous-système mais entraînent une marche dégradée difficilement détectable par l'agent-opérateur. Cette marche dégradée peut occasionner un manque de précision de la mesure des longueurs et ainsi une perte de profits due à la mauvaise optimisation de la découpe.

4.2 Exemple d'une session de travail du prototype *SIDI*

Voici un exemple de session du prototype *SIDI*. Supposons que l'agent maintenance ait choisi l'item "diagnostic" au niveau de ce sous-système IOTA. Le dialogue suivant s'instaure:

```
  ***** SIDI: le diagnostic commence sur IOTA.

  ** J'essaye de voir s'il s'agit d'une panne capteur.
     -- Observer si les diodes restent allumees ?
     -- Repondre Oui (allume'es), Non (eteintes), autre (ignore).

# ?

  ** Pour le moment je ne peux rien deduire.

  ** J'essaye de voir un probleme carte-entree-opto-coupleurs.
  ** Pour le moment la carte entree-opto-coupleurs est reconnue innocente.

  ** J'essaye de voir s'il s'agit d'un probleme fenetre-capteurs.
     -- Quand a eu lieu le dernier entretien des capteurs ?
     -- Repondre L (Plus d'un mois) l (Moins d'un mois), autre (ignore).

# l

  ** Pour le moment je me trouve en face de contradictions.
  ** Je ne peux ni innocenter les capteurs ni les culpabiliser.

  ** J'essaye de voir un probleme ligne-GEA.
  ** Pour le moment je ne peux rien deduire.

  ** J'essaye de voir un probleme carte-sortie-opto-coupleurs.
     -- Observer si l'afficheur cisaille clignote ?
     -- Repondre Oui (clignote), Non (eteint ou allume), autre (ignore).

# O

  ** Je crois avoir trouve' la panne, il s'agit:
      --  le disjoncteur a' la salle agent-operateur est sur la position maintenance
```

```
** Je conseille:
    -- basculer le disjoncteur.
    -- ne rien faire d'autre.
***** Veuiller me confirmer si c'est bien ca.
***** Que SIDI continue la surveillance pour vous.
```

Notons que les interventions de l'utilisateur sont toujours précédées d'un dièze (#) alors que les messages du système sont précédés de plusieurs étoiles (*).

D'après cet exemple typique, réalisé en "off-line", on peut remarquer tout d'abord l'effort entrepris pour faire participer l'utilisateur à l'élaboration du diagnostic. En effet, dans ce type de problème, s'il est important que le système donne un diagnostic correct, il est aussi important de convaincre l'utilisateur par ces résultats. Ceci n'est pas toujours immédiat puisque celui-ci peut ne pas tenir compte du résultat du système du diagnostic et mener son propre diagnostic!

Bien que les ingénieurs de maintenance soient satisfaits du système, on se doit de signaler les deux problèmes suivants:

- puisque nous n'avons pas d'interface très "sophistiqué" (i.e graphique), permettant aux ingénieurs de structurer et de spécifier eux même leurs sous-systèmes, nous avons joué le rôle d'interface entre les ingénieurs et le système! De ce fait, nous avons passé des longues discussions avec les ingénieurs pour structurer et spécifier ce sous-système. Il est alors intéressant de mettre en place un interface graphique, de laisser les ingénieurs spécifier seuls et de voir alors les résultats. Nous reviendrons sur ce problème au chapitre suivant.

- en réalité, les messages imprimés par le système proviennent pratiquement tous du mécanisme de justification. Un examen de ces messages, fait vite apparaître que l'information contenue dans ces messages est plutôt pauvre. Autrement dit en suivant le dialogue entre le système et l'utilisateur, il est très difficile de suivre le raisonnement du diagnostic conduit. Sans doute un effort doit être fait à ce niveau pour analyser l'arbre de raisonnement construit par le système et ainsi fournir de plus amples explications.

5 Autres systèmes de diagnostic

Le but de ce paragraphe n'est pas de faire une classification des systèmes de diagnostic existants, car cette classification peut être à la fois incomplète voire erronée: en effet, à cause de l'enjeu économique, les documents techniques sur les systèmes de diagnostic sont souvent considérés comme confidentiels par les industiels et, de ce fait, excluent toute étude

exhaustive. De même l'existence de certains systèmes de diagnostic "hybrides" conduit nécessairement à une classification hasardeuse.

Le but poursuivi ici est d'une part d'illustrer les techniques de diagnostic que nous avons présentées au second chapitre, d'autre part, identifier les caractéristiques des différents systèmes et dégager les points qu'ont ces systèmes en commun.

5.1 Le système "IN-ATE"

Le but poursuivi par ce système [Cantone 83] est de guider et d'assister un technicien lors du dépannage des composants électroniques. Le mécanisme de diagnostic de ce système, qui est basé sur des règles empiriques, a été bati de manière à prendre en compte:

- la topologie (i.e les connexions inter-composants) du circuit à diagnostiquer. Par exemple, soit le circuit suivant à trois composants:

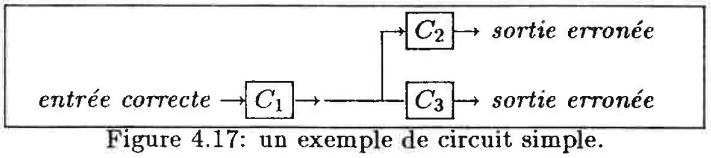


Figure 4.17: un exemple de circuit simple.

  le mécanisme de diagnostic devra soupçonner les composants C_2 et C_3 comme étant responsables du dysfonctionnement du circuit.

- le coût de chaque test à effectuer pour connaître l'état d'un composant donné du circuit. Ce coût est donné a priori au système.

- le coefficient estimant la probabilité de panne de chaque composant. En cas d'omission de ce coefficient par l'utilisateur, le système calcule alors un coefficient par défaut. Par exemple reprenons le circuit de la figure 4.17 et soit x la probabilité que l'entrée du composant C_1 soit correcte. Le système doit alors inférer une probabilité de $\frac{x}{2}$ pour chacun des composants C_2 et C_3.

- les probabilités conditionnelles résultantes des tests effectués. Elles sont exprimées sous forme de règles. Exemple:

> *si* x_i *est la probabilité que la sortie* o_i *d'un composant* C *ayant* n *sorties est correcte alors pour chaque composant, en amont de* C *et contribuant à la sortie* o_i*, lui assigner l'évidence (i.e la probabilité* $\sum_{i=1}^{n} \frac{x_i}{n}$*) qu'il n'est pas en panne.*

Le mécanisme de diagnostic est inspiré de la logique de vraisemblance et en particulier des travaux de Dempster-Shafer sur la théorie de l'évidence. Illustrons ce mécanisme sur un exemple. Soit C un composant auquel sont associés les coefficients x_1 et y_1. Le premier

coefficient est le degré de certitude qu'il fonctionne correctement et le second qu'il est en panne. Définissons le degré d'ignorance $z_1 = 1 - (x_1 + y_1)$ qui peut être supérieur à zéro.

Supposons qu'un nouveau test, ayant un coefficient x_2, vient confirmer que C fonctionne correctement, on doit alors procéder à la mise à jour des coefficients x_1, y_1 et z_1. Dans ce cas on a $y_2 = 0$ et donc $z_2 = 1 - x_2$. En supposant que les différents tests sont indépendants et en appliquant la règle de combinaison de Dempster on a:

$$x'_3 = x_2 \times x_1 + x_2 \times z_1 + z_2 \times x_1, \quad y'_3 = z_2 \times y_1 \quad \text{et} \quad z'_3 = z_2 \times z_1$$

Pour normaliser, on pose: $N = \frac{1}{x'_3 + y'_3 + z'_3}$ et pour obtenir les nouvelles valeurs de x_3, x_2 et x_1.

$$x_3 = x'_3 \times N, \quad x_2 = x'_2 \times N \quad \text{et} \quad x_1 = x'_1 \times N.$$

La règle de combinaison de Dempster, justifiée dans [Shafer 76], [Hummel 88], ainsi que les règles de distribution de probabilité constituent en fait la base de tout le mécanisme proposé dans IN-ATE.

L'intérêt de ce système est de proposer un raisonnement incertain, basé sur la théorie de l'évidence de Dempster-Shafer, qui est assez facile à implanter. Notons toutefois que l'exploitation de la fonction (Cf. technique de Davis au chapitre 2) de chaque composant n'est malheureusement pas utilisée au cours du diagnostic. Et ceci bien que le domaine de l'électronique soit approprié pour une telle exploitation.

5.2 Le système "PICON"

Ce système est implanté sur une machine Lisp LAMBDA-PLUS et peut gérer jusqu'à 20.000 données [Moore 86]. Sa principale caractérestique est en fait qu'il est opérationnel sur plusieurs installations industrielles[7]. Malgré les nombreuses publications sur ce système, on dispose de peu d'informations techniques. La plupart des articles écrits sur PICON se contentent de donner des informations générales telles que:

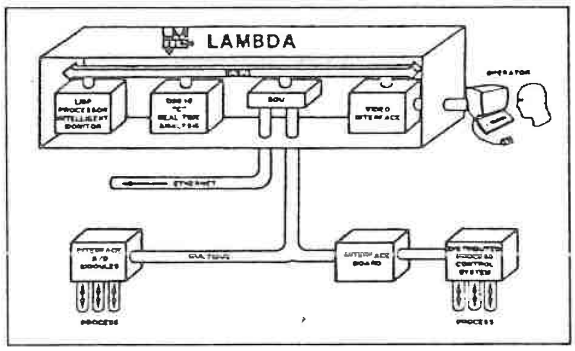


Figure 4.18: architecture de PICON.

[7]Parmi les utilisateurs cités dans [Moore 86]. En chimie: Texaco (Texas USA), Exxon (New-Jersay USA); Contrôle de procédés industriels: Johnson Controls Plant.

if then rule frame		
categories	safety	If T1006-T1007 > 4.5 and T1006 > 1000
Associated Unit	furnace-7, P7	Then - Conclude défaut-haut-fourneau
Priority	1	- Focus on H-F-7
Trace This	No	- Send "Stop H-F-7 " à opérateur.
Scan Interval	10 sec.	
Alert Interval	2 sec.	
Authors	Knick, RLM	
Latest Change	10 sep 85 10.00	
Usage	In Use	

Figure 4.19: un "frame" représentant une règle.

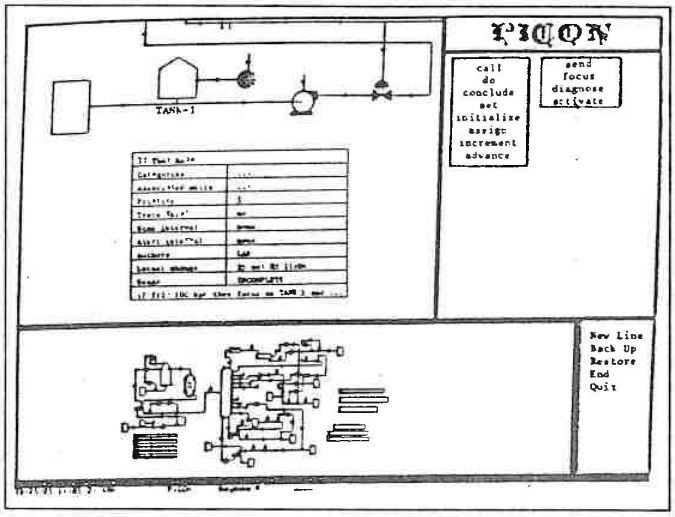


Figure 4.20: l'interface graphique de PICON.

Les particularités de ce système sont d'une part son interface graphique assez puissant [Khanna 86] et, d'autre part, de tourner sur une machine LISP.

5.3 Le système "SE.DIAG"

L'objectif de ce système [Marrakchi 86] est de traiter une large partie du matériel industriel utilisé dans les unités de production relevant du domaine de l'hydraulique, de la mécanique et de l'électrotechnique.

Trois niveaux de connaissances sont distingués dans ce système:

- le premier niveau est réservé à la représentation fonctionnelle des entités structurelles élémentaires. Le modèle de description d'un composant est:

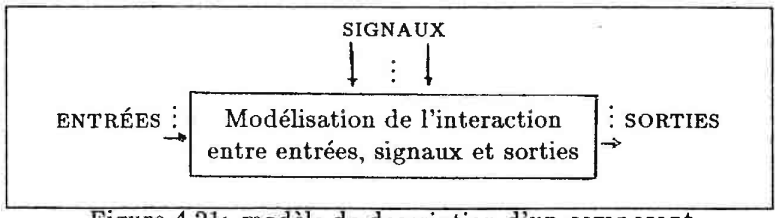


Figure 4.21: modèle de description d'un composant.

- le second niveau est réservé à la représentation structurelle du matériel à diagnostiquer. Cette description est réalisée en trois niveaux. Elle est exprimée en termes de système, de sous-système et de composant. La structure de données choisie est "le frame".

- le troisième niveau est consacrée à la modélisation du savoir-faire humain dans le domaine du diagnostic en termes de règles empiriques en utilisant des coefficients de certitude. Exemple:

> *si les entrées du composant sont normales et le dysfonctionnement ne concerne pas la sortie du composant alors écarter à 0.8 les prédécesseurs du composant.*

Partant d'un ensemble de symptômes donnés par l'utilisateur, le cycle de fonctionnement de la procédure de diagnostic est le suivant:

(1) définition de l'ensemble des suspects,

(2) choix d'un suspect à étudier,

(3) étude de l'état du suspect.

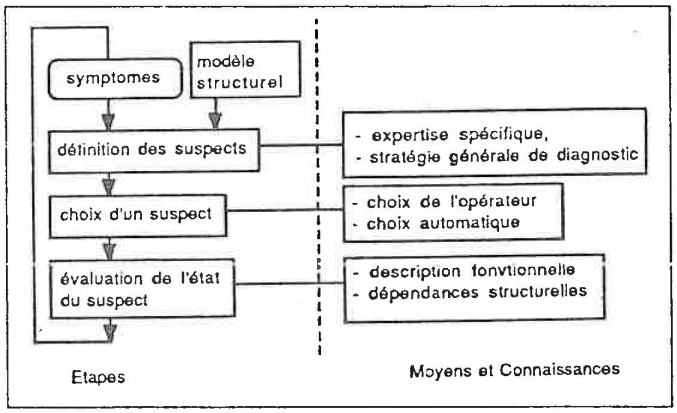


Figure 4.22: mécanisme de diagnostic.

Le système ne s'arrête que lorsqu'il se trouve dans l'une de deux situations suivantes:

- le composant responsable du disfonctionnement a été localisé, ce qui traduit le succès du diagnostic.
- l'univers d'investigation ne contient plus des composants soupçonnables, c'est l'échec de la procédure.

5.4 Le système "DEDALE"

L'objectif de ce système [Dague 86] est le diagnostic des pannes des circuits électroniques. Le mécanisme de diagnostic de ce système est inspiré de la physique qualitative. "DEDALE" peut être vu comme un système formel manipulant des expressions symboliques. En guise d'illustration de ce système, définissons trois opérateurs de comparaison qualitatifs:

$A \ll B$, A est négligeable devant B.
$A \approx B$, A et B sont comparables.
$A \simeq B$, A et B sont voisins.

Introduisons ensuite les règles[8] de propagation qualitatives [Dague 86] comme suit:

(1) $A \ll B \wedge B \ll C \Rightarrow A \ll C$
(2) $A \approx B \wedge B \ll C \Rightarrow A \ll C$
(3) $A \simeq B \wedge B \simeq C \Rightarrow A \simeq C$
(4) $A \ll B \Rightarrow A + B \simeq B$
(5) $A \simeq B \Rightarrow [A] = [B]$
(6) $A \approx B \wedge \neg([B] = 0) \Rightarrow \neg(A \ll B) \wedge \neg(B \ll A)$
(7) $[A] = [B] \wedge A + B \approx C \wedge A \approx B \Rightarrow A \approx C$

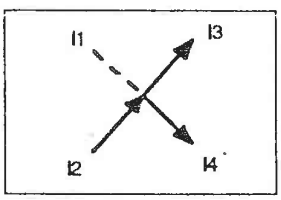
où $[X]$ représente le signe de l'expression X. Illustrons sur quelques exemples simples l'utilisation des ces règles. A cet effet considérons un noeud où arrivent les courants I_1 et I_2 et repartent les courants I_3 et I_4, on peut alors écrire la loi reliant les intensités:

$$I_1 + I_2 = I_3 + I_4$$

cette loi, écrite sous sa forme algébrique et connue sous le nom de la loi de kirchhoff, peut être remplacée par la loi écrite sous forme qualitative:

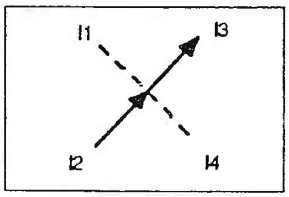
$$I_1 + I_2 \simeq I_3 + I_4$$

Voyons à présent différentes utilisations de ce système pour analyser un circuit:



supposons que $I_1 \ll I_2$ et $I_3 \approx I_4$,
on en déduit successivement par:
par (4) $I_1 + I_2 \simeq I_2$
par (3) $I_2 \simeq I_3 + I_4$
par (7) $I_2 \approx I_3 \approx I_4$
i.e le courant I_2 et ressort par I_3 et I_4.

[8] Seule une partie des règles est donnée. L'interprétation de ces règles est intuitive.



supposons que $I_1 \ll I_4$ et $I_2 \simeq I_3$,
on en deduit successivement par:
$I_2 \simeq I_3 + I_4$
$I_4 \ll I_3$
$I_4 \ll I_2$
i.e le courant entre par I_2 et ressort par I_3.

Ces exemples, quoique simples, montrent la puissance du système. Malheureusement, en pratique l'analyse d'un noeud est plus complexe. En effet les relations qualitatives entre les intensités ne sont pas accessibles directement et ne peuvent être connues. Pour faire face à ce problème les auteurs de Dedale [Dague 86] proposent un mécanisme analogue à celui qui en donné ci-dessus mais en utilisant seulement les relations entre intensités nominales, les variations de ces intensités par rapport à leurs valeurs nominales, les signes des intensités nominales et enfin les sens de variation des intensités.

5.5 Conclusions

Parmi ces systèmes, on peut remarquer deux tendances: ceux qui privilégient les connaissances par rapport au mécanisme de diagnostic comme le système IN-ATE par exemple et ceux qui, au contraire, privilégient davantage le mécanisme de diagnostic comme le système PICON. Alors que les mécanismes de diagnostic de ces systèmes sont basés soit sur un raisonnement profond soit sur un raisonnement de surface, notre mécanisme fait coopérer les deux types de raisonnement. Par contre le point faible de notre système, notamment vis-à-vis de PICON est au niveau de l'interface. En effet, selon la plupart des articles qui ont été consacrés sur ce système mettent l'accent sur son interface.

Notons enfin qu'au delà des particularités de chacun de ces systèmes, une caractéristique commune à tous est *l'absence d'une méthode quelconque pour leurs mise en place.* C'est ce que nous avons indiqué dès le départ et c'est ce qui distingue notre travail de ceux déjà réalisés.

6 En résumé

Nous venons de décrire un environnement de spécification et de construction des systèmes de diagnostic, où nous avons défini l'organisation des connaissances en bibliothèques de catalogues, la représentation des spécifications, la structuration des outils en modules ainsi que les différentes structures de données. Mais, comme nous l'avons signalé au début de ce travail, s'il est important de disposer des outils, il est également nécessaire de procéder à l'étape de l'expérimentation. C'est pourquoi nous avons réalisé le prototype *SIDI* et l'avons confronté à des sous-systèmes industriels. Toutefois, sachant que notre objectif principal est avant tout d'ordre méthodologique, nous avons pas réalisé tous les modules pour couvrir toutes les fonctionnalités de l'environnement. Nous nous sommes contenté à réaliser les modules nécessaires afin de valider et de confronter les idées présentées ici.

Afin de mieux situer notre travail par rapport à ce qui existe, nous avons présenté ensuite un panorama des systèmes de diagnostic réalisés.

A présent, nous sommes conduits à procéder à une critique afin de montrer les prolongements de ce travail tant au niveau théorique que méthodologique.

Epilogue

Notre but ici est de faire une évaluation et une critique de l'ensemble du travail effectué, permettant ainsi d'ouvrir des perspectives à venir.

1 Bilan de ce travail

Tout au long de ce travail, nous nous sommes efforcés de mettre en place un cadre comprenant méthodes, langage et outils pour la spécification et la construction des systèmes de diagnostic des grandes installations industrielles. Délibérément, nous nous sommes donnés des contraintes sur la définition de ce cadre: uniformité, cohérence et intégrité. Par exemple, la syntaxe du langage a été choisie de manière quasi identique pour exprimer les différentes parties de la spécification des composants. De même le principe de coopération proposé intègre parfaitement les deux mécanismes de raisonnement. Sachant que ce cadre est destiné à des utilisateurs non spécialistes, nous avons évité autant que possible le formalisme au profit des définitions simples mais précises des concepts de base tels ceux de symptôme ou de panne. A présent, analysons rapidement ce qui a été fait:

Partant d'une installation industrielle, qui est en général complexe et difficile à appréhender comme un tout, le premier chapitre propose une méthode de spécification de l'installation. Celle-ci consiste à structurer l'installation en une hiérarchie de composants abstraits, terminaux et corpusculaires, à construire ensuite des hiérarchies et des graphes secondaires, à exprimer enfin les spécifications descriptives de ces composants. C'est sans doute le chapitre le plus original. Par son étude méthodologique, ce chapitre contraste notre travail des autres travaux.

Quant au chapitre 2, qui présente un étude de synthèse des techniques de diagnostic, son mérite est de dégager les principes de base et de négliger les notions secondaires et conjoncturelles. Il propose également des remèdes aux restrictions de certaines techniques de diagnostic. C'est ainsi que l'idée de proposer des paradigmes[1] vient remédier au problème lié à la difficulté d'énumération des règles empiriques utilisées par un raisonnement de surface.

[1]Notons que nous avons exploité, avec succès, cette idée dans un domaine voisin, qui est celui du diagnostic nutritionnel [ElAyeb 88d].

A son tour le chapitre 3 vient compléter le travail accompli au premier chapitre. Après avoir défini les stratégies et les tactiques, il propose une méthode de spécification et d'élaboration du diagnostic. Il pose également un principe de coopération entre deux types de raisonnement: le raisonnement de surface et le raisonnement profond qui sont considérés dans la littérature comme dissociés sinon opposés. Il définit finalement une méthode de justification, basée sur des paradigmes, facile à implanter et à exploiter. L'originalité de ce chapitre apparaît au niveau de la clarification de l'activité de diagnostic (Cf. paragraphes 1 et 7) ainsi qu'au niveau de la coopération[2]. Dans une moindre mesure, elle apparaît également au niveau de la méthode de justification.

Enfin, le quatrième chapitre propose les outils nécessaires pour conduire à bien les méthodes proposées et construire le système de diagnostic. L'organisation des connaissances en bibliothèques de catalogues permet une facilité d'enrichissement de ces connaissances. De même le partitionnement des spécifications ainsi que la représentation arborescente choisies contribuent à faciliter la réutilisation des parties des spécifications. Enfin la structuration des outils en modules facilite peut être l'adjonction d'un interface plus élaboré et plus adapté.

Mais ce travail n'est pas clos; des approfondissements tant au niveau théorique que méthodologique restent à faire. Voyons ce qu'il en est.

2 Prolongements et approfondissements

Nous distinguons ici deux types d'activités, l'une théorique, et l'autre méthodologique. Bien sûr ces deux activités ne sont pas indépendantes, mais, au contraire, elles sont complémentaires et peuvent être menées de front. Comme nous allons le voir, à chaque activité théorique correspond en général une ou plusieurs activités méthodologiques et inversement. Commençons par les approfondissements théoriques.

2.1 Approfondissements théoriques

[a] *Description d'une sémantique formelle*

Il s'agit de définir un langage de haut niveau pour spécifier très facilement une large classe de composants, de donner ensuite à ce langage une sémantique opérationnelle très précise qui permet l'analyse des spécifications écrites et la détection des inconsistances s'il y en a. Parallèlement, il est très intéressant d'étudier les travaux entrepris sur les réseaux de Petri, le grafcet et d'autres modèles de description utilisés principalement par les automaticiens. Comme nous l'avons indiqué, les spécifications du comportement ne

[2]Dans un tout autre domaine qui est celui des bases des données, nous avons déjà exploité cette idée [ElAyeb 88b] en faisant coopérer un mécanisme de recherche classique et un mécanisme d'inférence basé sur la méthode de résolution.

sont pas exploitées. Une idée consiste alors à écrire un interprète pour ces spécifications pour "valider" les lois spécifiées des composants.

[b] *Définition d'un univers de composants*

Il s'agit de faire tout d'abord un inventaire des différents types de composants couramment utilisés dans les installations industrielles, de les étudier par la suite pour dégager un noyau de composants de base, de définir enfin des opérateurs pour pouvoir former d'autres composants. Voici un exemple simple:

Soient les composants de base:
- pompe que nous représentons graphiquement par ⟹
- vanne que nous représentons graphiquement par ⋈

A présent, considérons l'opérateur *jonction*, qui intuitivement joint deux composants donnés. On peut alors combiner par cet opérateur, ces deux composants de base pour former le composant "lubrificateur"

jonction(⟹, ⋈) ⊢ *lubrificateur*

Informellement, cela veut dire que la spécification du composant ***lubrificateur*** est "l'union" des spécifications de deux composants de base: *pompe* et *vanne*.

Mais il n'est pas question de combiner les composants de base pour former n'importe quoi. D'où la nécessité de règles de composition sur l'application des opérateurs.

[c] *Elaboration d'un mécanisme d'apprentissage*

Supposons qu'une tactique ait complètement échoué, c'est-à-dire qu'elle a généré plusieurs fois un ensemble vide de composants déclarés en panne. Le but du mécanisme d'apprentissage est alors d'analyser l'échec de la tactique avant de construire progressivement des préconditions d'application de la tactique évitant ainsi un nouveau échec. Shématiquement, l'activité du mécanisme peut se résumer comme suit:

- analyser le contexte dans lequel cette tactique a été appliquée. Ce contexte comprendra notamment:
  - les valeurs prises par les variables d'état ainsi que les lois non vérifiées des composants sur lesquels la tactique a été appliquée.
  - les valeurs prises par les variables d'état ainsi que les lois non vérifiées, s'il y en a, du composant virtuel "environnement" de l'installation.
- déduire par la suite les conditions d'échec de cette tactique.
- considérer la négation de ces conditions d'échec comme des préconditions d'application de la tactique considérée.

2.2 Approfondissements méthodologiques

[a] *Extension vers d'autres activités voisines*

On peut voir au moins trois activités:

La première activité concerne le diagnostic de contrôle de qualité des produits fabriqués dans l'installation. Il s'agit alors de voir tout d'abord comment adapter la méthode de diagnostic de pannes vers cette activité, de rechercher ensuite les stratégies, les tactiques et les paradigmes nécessaires pour conduire ce type de diagnostic.

La détection des pannes logicielles (ou bogues) constitue la seconde activité. En effet dans ce travail, seules les pannes matérielles sont prises en considération, il est alors intéressant de voir dans quelle mesure on peut prendre en compte ce type de pannes. Pour ce faire, on est conduit à faire dans un premier temps une synthèse des techniques employées dans ce domaine, qui est appelé la sûreté des programmes [Kanoun 87], [Kreutzer 88], [Laprie 85], et de voir ensuite comment les inclure dans la méthode de diagnostic donnée ici.

Le diagnostic de pannes des grandes installations se caractérise par la phrase suivante: *"les dépanneurs sont débrouillards, l'essentiel est de les orienter vers la panne sans avoir besoin de les noyer dans les details"*. Aussi nous avons tenu compte dans ce travail de cette caractéristique. Une extension de ce travail consiste alors à voir dans quelle mesure il est possible d'adapter ce travail à des "petites installations" telle que les chaudières d'eau par exemple. Ce qui nécessite:

- une simplification de la méthode de spécification. En effet si les différentes étapes de spécification peuvent être justifiées dans le cas des laminoirs ou d'un haut-fourneau, il n'en sera pas de même lorsqu'il s'agit d'une chaudière par exemple.
- une plus grande précision du mécanisme de diagnostic pour localiser de manière détaillée les pannes.

[b] *Assistance à la réutilisation des spécifications*

Bien que la représentation arborescente des connaissances favorise la réutilisation des "morceaux" ou des parties de spécifications, ceci reste toutefois insuffisant. Dans le but d'accroître cette réutilisation, et donc faciliter cette tâche qu'est la spécification, il est nécessaire de concevoir un interface graphique qui comprend notamment:

- un langage graphique doté des primitives de haut niveau pour manipuler ces morceaux de spécifications. Reste à définir précisément cette notion de "morceau" ou partie de spécification ! Jusqu'à présent nous avons assimilé cette notion comme le résultat d'une étape de spécification (ainsi la spécification de la structure d'un composant est un morceau, la spécification du comportement en est une autre ...).

- un ensemble de règles associées à chaque primitive. Ces règles doivent alors proposer à l'utilisateur des actions à entreprendre en fonction des primitives utilisées.

[c] *Recherche des heuristiques*

Faire une "bonne" abstraction conduisant à sélectionner l'ensemble minimum de variables d'état caractérisant parfaitement un composant donné est sans doute une tâche assez délicate; d'où la nécessité de rechercher des heuristiques pour conduire ce choix. Le même problème se pose pour sélectionner les lois d'un composant.

Au cours du troisième chapitre, nous avons insisité sur l'enrichissement des catalogues des stratégies, des tactiques et des paradigmes. Prenons le cas des stratégies par exemple, le problème est alors de savoir comment en découvrir de nouvelles. La résolution de ce (méta) problème est loin d'être simple et nécessite une étude approfondie. Le même problème a été posé dans le domaine de la construction des programmes où la démarche suivante a été proposée [Finance 79] :

(I) commencer par dégager et structurer les stratégies déjà identifiées,

(II) en déduire ensuite des règles de construction.

Si le premier point a été partiellement abordé[3] au cours du troisième chapitre, le second reste à réaliser entièrement.

[3] C'est ainsi que nous avons commencer à distinguer deux types de stratégies: explicites et calculées. Toutefois ceci reste très insuffisant.

Addenda

Addenda A: Description des catalogues

A.1 Les catalogues de la bibliothèques système

LE CATALOGUE DES TACTIQUES

Comme les stratégies, les tactiques sont aussi organisées en une structure d'arbre:

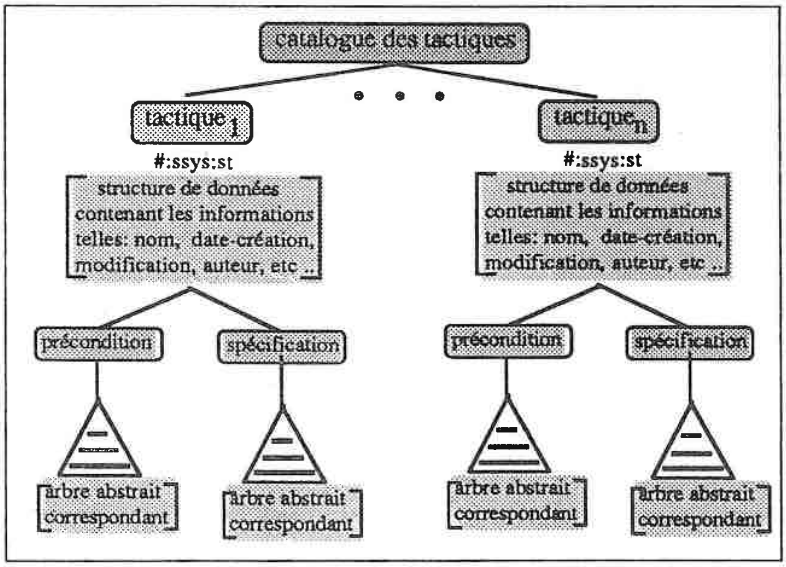
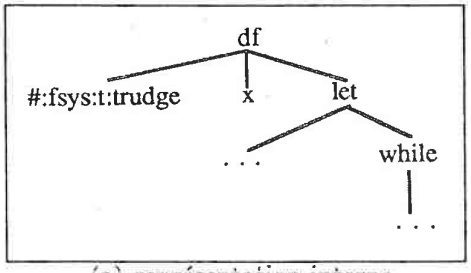


Figure A.A.1: organisation des tactiques dans le catalogue.

A titre d'exemple, voici la représentation de la tactique "trudge" dont l'algorithme a été donné au paragraphe 3.1 du chapitre 3.



(a) représentation interne

```
(df #:fsys:t:trudge x
   (let ((#:vsys:sus) (#:vsys:inn) (n (length x)))
     (while (> n 0)
       (if (#:fsys:eq n (quote #:fsys:FC))
           ...
       )
     ...
     )
   )
```

(b) F-expression correspondante

Figure A.A.2: représentation d'une tactique.

LE CATALOGUE DES PARADIGMES

Pour les paradigmes, nous adoptons la même structure qu'auparavant où la précondition, dans ce cas, est reduite à la constante "*vrai*". Nous avons:

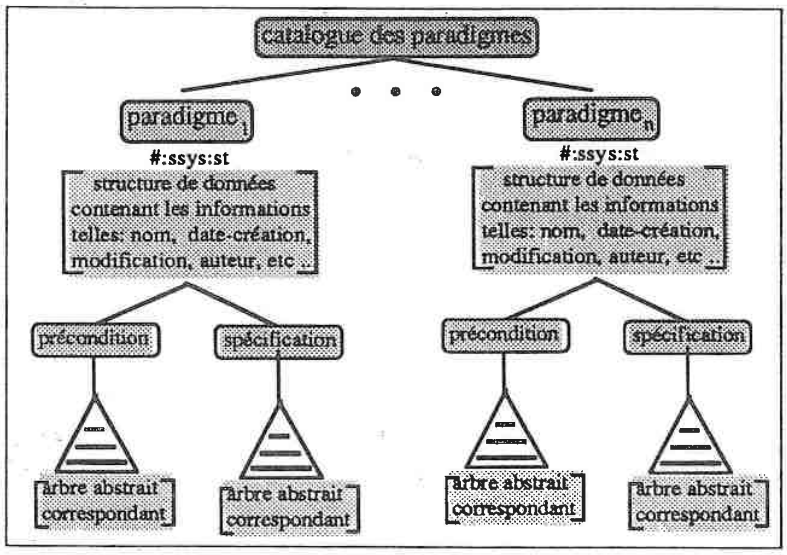
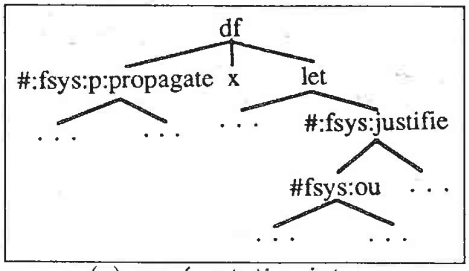


Figure A.A.3: organisation des paradigmes dans le catalogue.

A titre d'exemple, considérons le paradigme "propagate":



(a) représentation interne

```
(df #:fsys:p:propagate x
    (let ( (#:vsys:p:dj) (#:vsys:p:exp) (#:vsys:p:C) )
      (#:fsys:justifie
        (#:fsys:ou  #:vsys:p:exp #:vsys:p:dj)
          #:vsys:p:C)
        ...
      )
      ...
)
```

(b) F-expression correspondante

Figure A.A.4: représentation d'un paradigme.

A.2 Les catalogues de la bibliothèque commune

LE CATALOGUE DES SPÉCIFICATIONS DU COMPORTEMENT DES COMPOSANTS

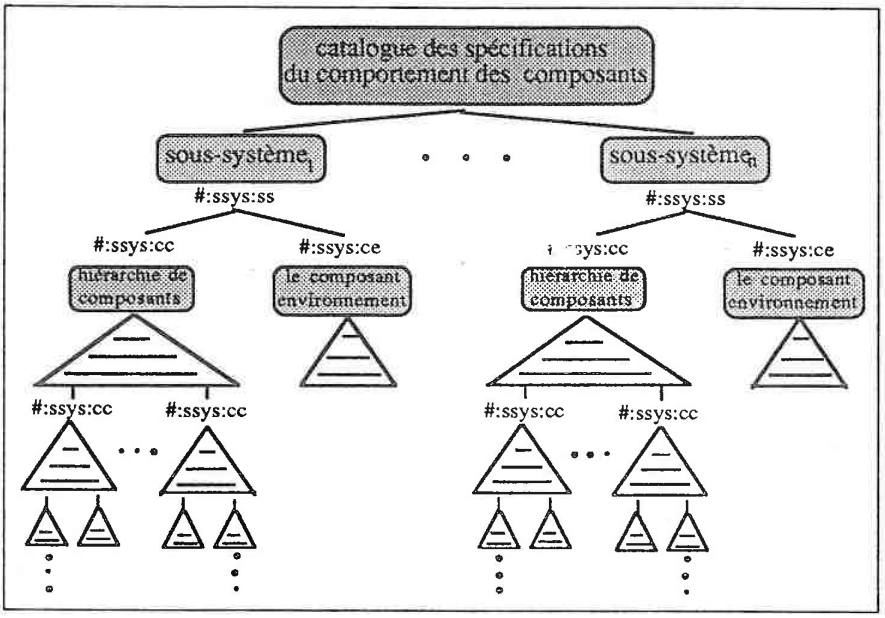


Figure A.A.5 : organisation des spécifications du comportement des composants.

Remarquons que cette structure d'arbre reflète la démarche qui a été poursuivie au paragraphe 3 du chapitre 1 structurant la spécification du comportement d'un composant en paquets formés d'unités de groupe d'opérateurs.

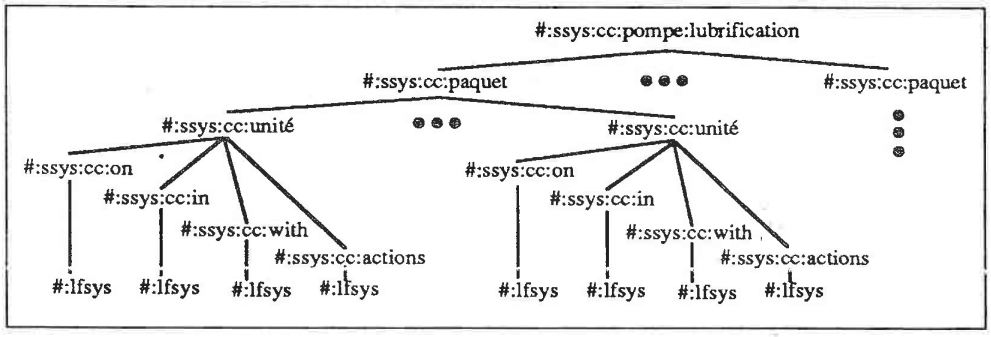


Figure A.A.6 (a): représentation interne des spécifications des lois.

A titre d'exemple, voici la représentation de la spécification du composant "pompe" qui a été donnée à la figure 1.11 (suite) du premier chapitre.

```
(de #:fsys:cc:pompe:lubrification (#:psys)
   (let ( ...)
    (without-interrupts
       ...
      ; paquet augmenter
      (cond  ((and (#:fsys:received #:vsys:consigne-plus-debit:pompe:lubrification)
                   (equal #:vsys:statut:pompe:lubrification))
              (#:fsys:nothing)
             )
             (and (#:fsys:received #:vsys:consigne-plus-debit:pompe:lubrification)
                  (equal #:vsys:vitesse:pompe:lubrification 3000))
              (#:fsys:nothing)
               ...
       )
      ...
   ))
)
```

Figure A.A.6 (b): *S*-expressions correspondantes à l'exemple.

LE CATALOGUE DES SPÉCIFICATIONS DE L'INTERFACE DES COMPOSANTS

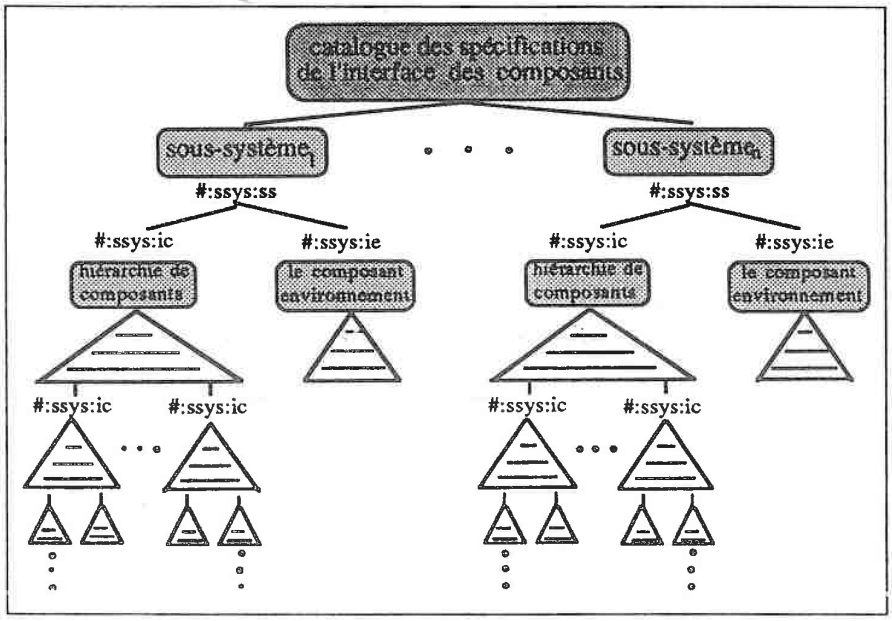


Figure A.A.7: organisation des spécifications de l'interface des composants.

Ce dernier exemple choisi est le composant, virtuel, environnment dont la spécification a été donnée à la figure 1.21 du chapitre 1.

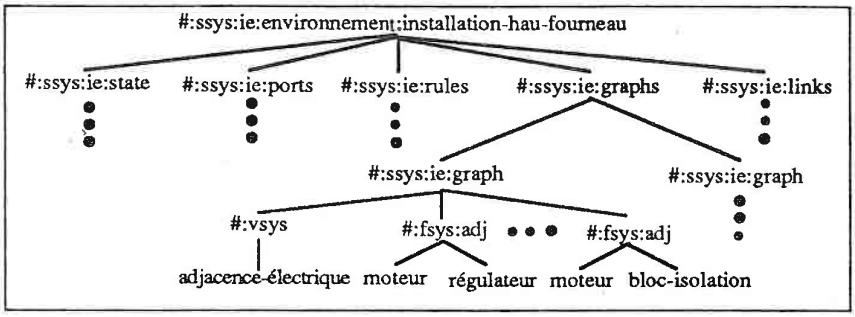


Figure A.A.8 (a): représentation interne des spécifications de l'interface.

```
(de #:fsys:ic:installation-haut-fourneau (#:psys)
   (let ( ...)
      ; Graphe 2
   (#:fsys:adjacent '#:vsys:adjacence-électrique
                    '#:vsys:moteur:pompe
                    '#:vsys:régulateur:pompe)
   (#:fsys:adjacent '#:vsys:adjacence-électrique
                    '#:vsys:moteur:pompe
                    '#:vsys:bloc-isolation:pompe)
      ...
   )
)
```

Figure A.A.8 (b): S-expressions correspondantes à l'exemple.

A.3 Les catalogues de la bibliothèque spécialisée

LE CATALOGUE DES SPÉCIFICATIONS DU DIAGNOSTIC DES COMPOSANTS TERMINAUX

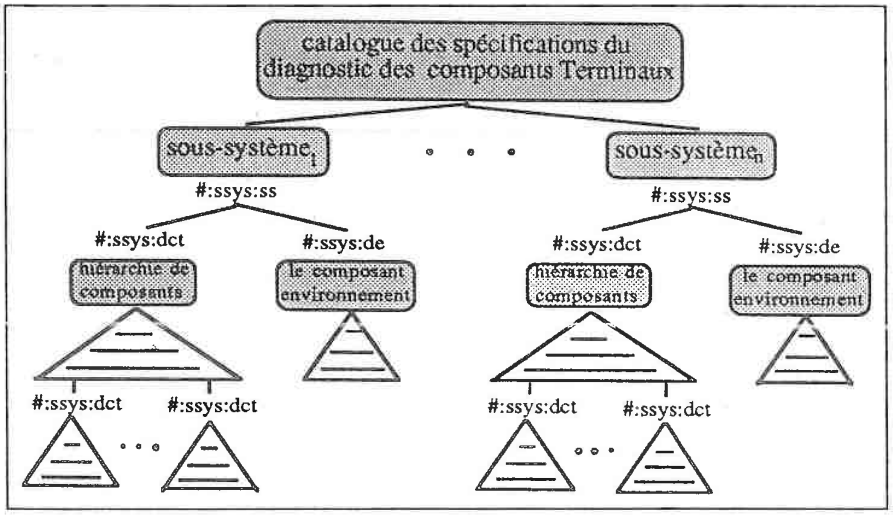


Figure A.A.9: organisation des spécifications du diagnostic des composants terminaux.

Dans ce cas, nous remarquons que l'arbre est d'une profondeur de 3. A titre d'exemple, voici la représentation de la spécification du diagnostic du composant terminal "analyseur-O2" qui a été donnée à la figure 3.20 du chapitre 3.

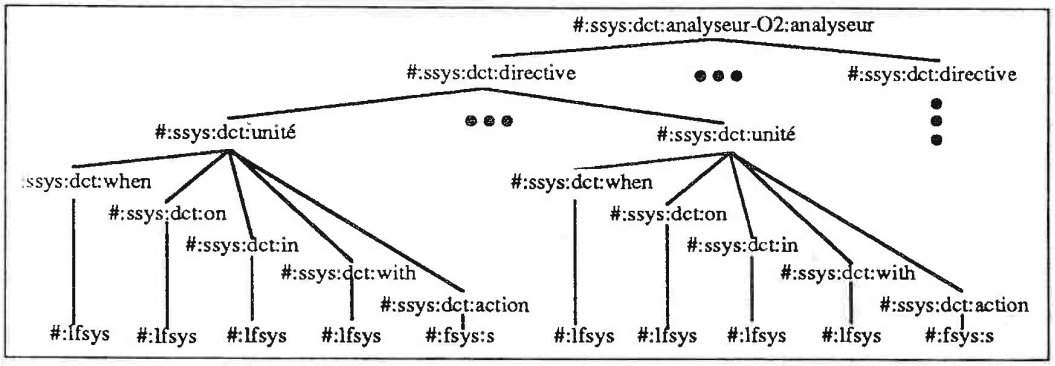


Figure A.A.10 (a): représentation interne des spécifications du diagnostic des $\mathcal{CT}$s.

```
(de #:fsys:dct:analyseur-O2:analyseur (#:psys)
   (let ( ...)
      ...
      ; directive diag 3
      (cond  ((#:fsys:uncheck '#:vsys:loi3:analyseur-O2-analyseur )
               (#:fsys:user '#:vsys:pompe:moteur-pompe #:fsys:t:procédure-diag-AO2))
             (t          nil)
      )
      ...
   )
)
```

Figure A.A.10 (b): S-expressions correspondant à l'exemple.

LE CATALOGUE DES SPÉCIFICATIONS DU DIAGNOSTIC DES COMPOSANTS CORPUSCULAIRES

Finalement, les spécifications du diagnostic des composants corpusculaires, en terme de paradigmes, sont représentées par un arbre de profondeur 2.

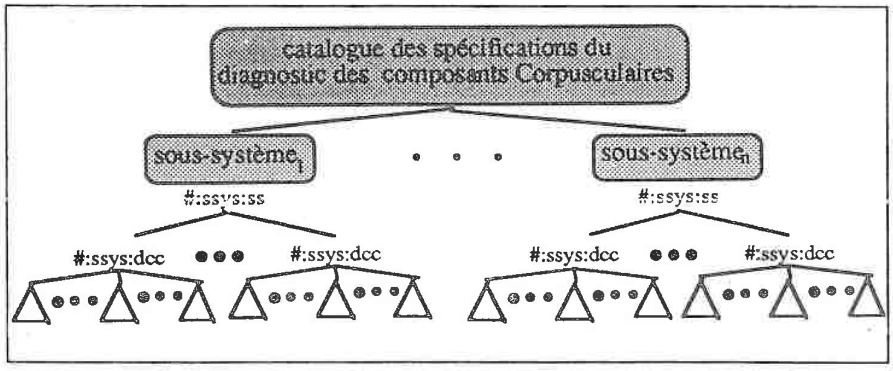


Figure A.A.11: organisation des spécifications du diagnostic des $\mathcal{CC}$s.

Addenda B: Syntaxe concrète de spécification

B.1 Conventions

Le vocabulaire terminal est formé de toutes les représentations usuelles des caractères ascii. Les mots réservés sont soulignés dans les règles de grammaire. Les non-terminaux sont présentés entre chevrons. Les unités lexicales sont notées en MAJUSCULES. Les parties facultatives sont notées entre crochets. Les terminaux du langage sont mis entre quotes. Enfin une séquence entre accolades et post-fixée par une etoile signifie que l'on peut la répeter 0 à n fois.

B.2 Syntaxe

<*spécification*> ::= <*entête*> <*informations*> <*variables-état*> <*variables-locales*>
<*ports-entrées*> <*ports-sortie*> <*comportement*> <*lois*> <*diagnostic*>

<*entête*> ::= IDENTIFICATEUR '(' IDENTIFICATEUR ')' ';'

<*informations*> ::= <*information*> { <*information*> }*
<*information*> ::= synonym ':' IDENTIFICATEUR {',' IDENTIFICATEUR}* ';'
<*information*> ::= location ':' IDENTIFICATEUR ';'
<*information*> ::= belongs ':' IDENTIFICATEUR {',' IDENTIFICATEUR}* ';'
<*information*> ::= type ':' active ';'
<*information*> ::= type ':' passive ';'
<*information*> ::= status ':' on ';'
<*information*> ::= status ':' off ';'

<*variables-état*> ::= <*variable-état*>
<*variable-état*> ::= IDENTIFICATEUR ':' <*type-ve*>
<*type-ve*> ::= deduced ';'
<*type-ve*> ::= available ';'
<*type-ve*> ::= asked ';'

<*variables-locales*> ::= <*variable-locale*>
<*variable-locale*> ::= IDENTIFICATEUR ':' local range '[' NOMBRE ',' NOMBRE ']' ';'
<*variable-état*> ::= IDENTIFICATEUR ':' local restricted '{' IDENTIFICATEUR ',' IDENTIFICATEUR}*
'}' ';'

<*ports-entrées*> ::= <*port-entrée*>
<*port-entrée*> ::= IDENTIFICATEUR ':' <*type-pes*>

<*ports-sortie*> ::= <*port-sortie*>
<*port-sortie*> ::= IDENTIFICATEUR ':' <*type-pes*>

<*type-pes*> ::= undetected ';'
<*type-pes*> ::= deduced ';'
<*type-pes*> ::= detected ';'

<*comportement*> ::= begin <*paquets*> end
<*paquets*> ::= <*paquet*> { <*paquet*> }*
<*paquet*> ::= <*unite*> { <*unite*> }*
<*unite*> ::= LABEL ':' [on $<exp_{\{pe\}}>$] [in $<exp_{\{ve\}}>$] [with $<exp_{\{m\}}>$] $\mapsto$
<*action*> [delay INTEGER] {',' <*action*> [delay INTEGER] }';'

<*lois*> ::= <*déclaration-lois*> begin <*contraintes*> end
<*déclaration-lois*> ::= <*déclaration-loi*> { <*déclaration-loi*> }*
<*déclaration-loi*> ::= LABEL ':' NOMBRE ';'
<*contraintes*> ::= <*contrainte*> { <*contrainte*> }*
<*contrainte*> ::= LABEL ':'
[on $<exp_{\{pe\}}>$] [in $<exp_{\{ve\}}>$] [with $<exp_{\{m\}}>$] $\Longrightarrow$
[on $<exp_{\{pe\}}>$] [in $<exp_{\{ve\}}>$] [with $<exp_{\{m\}}>$] [<delai>]

<delai> ::= [delayed INTEGER]
<delai> ::= [-delayed- INTEGER]

<*diagnostic*> ::= <*déclaration-directives*> begin <*directives*> end
<*déclaration-directives*> ::= <*déclaration-directive*> { <*déclaration-directive*> }*
<*déclaration-directive*> ::= LABEL ':' NOMBRE ';'
<*directives*> ::= <*directive*> { <*directive*> }*
<*directive*> ::= <*unite*> { <*unite*> }*
<*unite*> ::= LABEL ':' when <*id-loi*> { <*id-loi*> }*
[on $<exp_{\{pe\}}>$] [in $<exp_{\{ve\}}>$] [with $<exp_{\{m\}}>$]
$\mapsto$ [<*id-diagnostic*>] [true] ';'

Remarque: pour faciliter la lecture, nous avons gardé les mêmes notations que celles employées au cours des différents chapitres. Toutefois, en pratique on est conduit à adopter une syntaxe différente.

Addenda C: Spécification de l'exemple

Au cours du chapitre 0, nous avons traité informellement l'exemple "sous-cisaille". Nous allons ici donner les spécifications formelles correspondantes à cet exemple.

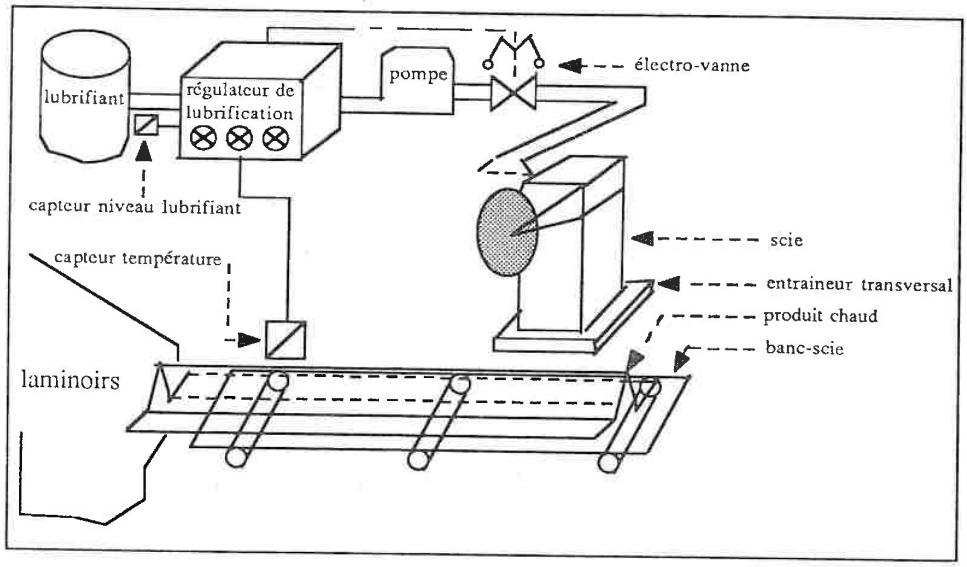


Figure 0.2 (réproduction): schéma du sous-système "cisaille".

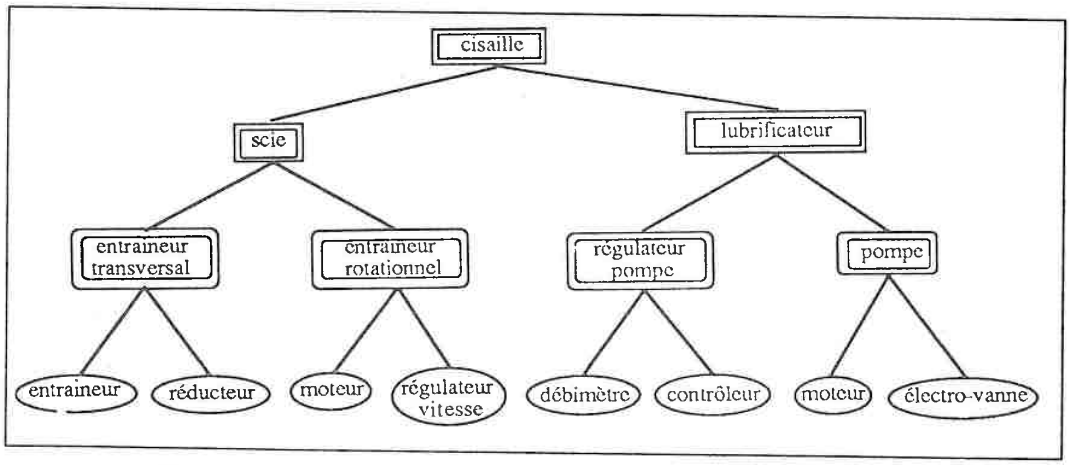


Figure 0.3 (réproduction): hiérarchie du sous-système "cisaille".

C.1 Spécification de la structure

(1) lubrificateur(cisaille);
(2) synonym: lubrifiant;
(3) location: banc-cisaille;
(4) belongs: main-hierarchy;
(5) type: active
STATE % les variables d'état
(6) état : deduced restricted {marche,arret} init arret;
(7) pression : deduced restricted {faible,moyenne,haute};
(8) débit : available restricted {faible,moyen,haut};
(9) puissance : available restricted {faible,moyenne,haute};
INPUT % les ports d'entrée
(10) p-mise-M/A : detected;
(11) p-température : detected;

Figure A.C.1: Structure du $\mathcal{CA}$ "lubrificateur".

(1) régulateur(lubrificateur);
(2) synonym: reégu;
(3) location: banc-cisaille;
(4) belongs: main-hierarchy;
(5) type: active
STATE % les variables d'état
(6) alarme : available restricted {déclenchée,non-déclenchée};
(7) pression : available restricted {faible,moyenne,haute};
INPUT % les ports d'entrée
(8) p-niveau-lubrifiant : detected;
(9) p-température : detected;
OUTPUT % les ports de sortie
(10) p-arret-pompe: undetected to pompe;
(11) p-puissance : detected to pompe;

Figure A.C.2: Structure du $\mathcal{CT}$ "régulateur-pompe".

(1) pompe(lubrificateur);
(2) synonym: lub;
(3) location: banc-cisaille;
(4) belongs: main-hierarchy;
(5) type: active
STATE % les variables d'état
(6) état : deduced restricted {marche,arret} init arret;
(7) état-électro-vanne : available restricted {faible,moyenne,grande};
(8) puissance : available restricted {faible,moyenne,haute};
INPUT % les ports d'entrée
(9) p-arret-pompe: undetected;
(10) p-puissance : detected;

Figure A.C.3: Structure du $\mathcal{CT}$ "pompe".

(1) débimètre(régulateur);
(2) synonym: débiteur;
(3) location: banc-cisaille;
(4) belongs: main-hierarchy;
(5) type: active
STATE % les variables d'état
(6) voyants : deduced restricted {allumés,éteints};
(7) température : available ;
(8) débit : available restricted {faible,moyen,haut};
INPUT % les ports d'entrée
(9) p-température : detected;
OUTPUT % les ports de sortie
(10) p-débit : undetected to controleur;

Figure A.C.4: Structure du $\mathcal{CC}$ "débimètre".

(1) controleur(régulateur);
(2) synonym: fox;
(3) location: banc-cisaille;
(4) belongs: main-hierarchy;
(5) type: active
STATE % les variables d'état
(6) état : deduced restricted {marche,arret} init arret;
(7) diodes : asked restricted {éteintes,clignotantes};
INPUT % les ports d'entrée
(8) p-débit : undetected ;
(9) p-niveau-lubrifiant : detected ;
OUTPUT % les ports de sortie
(10) p-vitesse : undetected to moteur;
(11) p-position-électro-vanne : undetected to électro-vanne;

Figure A.C.5: Structure du $\mathcal{CC}$ "contrôleur".

(1) électro-vanne(pompe);
(2) synonym: vanne;
(3) location: banc-cisaille;
(4) belongs: main-hierarchy;
(5) type: active
STATE % les variables d'état
(6) état : deduced restricted {marche,arret} init arret;
(7) position-ouverture : available restricted {faible,moyenne,grande};
(8) indicateur-vanne : available range [5, 10];
INPUT % les ports d'entrée
(9) p-position-électro-vanne : undetected ;

Figure A.C.6: Structure du $\mathcal{CC}$ "électro-vanne".

(1) moteur(pompe);
(2) synonym: alim;
(3) location: banc-cisaille;
(4) belongs: main-hierarchy;
(5) type: active
STATE % les variables d'état
(6) état : deduced restricted {marche,arret} init arret;
(7) vitesse : available restricted {0,500,1000,1500,2500,2500,3000};
(8) intensité : available range [0, 150];
INPUT % les ports d'entrée
(9) p-vitesse : undetected ;

Figure A.C.7: Structure du $\mathcal{CC}$ "moteur".

C.2 Spécification des lois

(1) lubrificateur(cisaille) ;
(2) loi_1 : 3 ;
(3) loi_2 : 2 ;
(4) loi_3 : 2 ;
begin
% contraindre la température
(5) loi_1: *true* $\Longrightarrow$ with $\leq$(p-température,1000) $\geq$(p-température,300);
% contraindre le débit
(6) loi_2: in $\gg$(débit) $\approx$(pression) $\Longrightarrow$ $\gg$(puissance)
(7) loi_2: in $\ll$(débit) $\approx$(pression) $\Longrightarrow$ $\ll$(puissance)
(8) loi_2: in $\approx$(débit) $\approx$(pression) $\Longrightarrow$ $\approx$(puissance)
(9) loi_3: in $\ll$(débit) $\approx$(puissance) $\Longrightarrow$ $\gg$(pression)
(10) loi_3: in $\gg$(débit) $\approx$(puissance) $\Longrightarrow$ $\ll$(pression)
end

Figure A.C.1 (suite): Lois du $\mathcal{CA}$ "lubrificateur".

(1) régulateur(lubrificateur) ;
(2) loi_1 : 3 ;
(3) loi_2 : 2 ;
begin
% contraindre le niveau du lubrifiant
(4) loi_1: *true* $\Longrightarrow$ with $\leq$(p-niveau-lubrifiant,10) $\geq$(p-niveau-lubrifiant,5);
% contraindre la température
(5) loi_2: *true* $\Longrightarrow$ with $\leq$(p-température,1000) $\geq$(p-température,300);
end

Figure A.C.2 (suite): Lois du $\mathcal{CT}$ "régulateur".

(1) pompe(lubrificateur);
(2) loi_1 : 3 ;
(3) loi_2 : 2 ;
begin
% contraindre l'état de l'électro-vanne
(4) loi_1: in =(état-électro-vanne,faible) ≈(puissance) ⟹ =(débit,faible)
(5) loi_1: in =(état-électro-vanne,moyenne) ≈(puissance) ⟹ =(débit,moyen)
(6) loi_1: in =(état-électro-vanne,grande) ≈(puissance) ⟹ =(débit,haut)
% contraindre le débit
(7) loi_2: in ≫(pression) ≈(puissance) ⟹ ≪(débit)
(8) loi_2: in ≪(pression) ≈(puissance) ⟹ ≫(débit)
end

Figure A.C.3 (suite): Lois du $\mathcal{CT}$ "pompe".

(1) débimètre(régulateur) ;
(2) loi_1 : 2 ;
(3) loi_2 : 1 ;
begin
% contraindre la température
(4) loi_1: *true* ⟹ with ≤(p-température,1000) ≥(p-température,300);
(5) loi_2: in =(température,faible) ⟹ in =(débit,faible)
(6) loi_2: in =(température,moyenne) ⟹ in =(débit,moyen)
(7) loi_2: in =(température,haute) ⟹ in =(débit,haut)
end

Figure A.C.4 (suite): Lois du $\mathcal{CC}$ "débimètre".

(1) controleur(régulateur) ;
(2) loi_1 : 2 ;
(3) loi_2 : 1 ;
begin
% contraindre le niveau du lubrifiant
(4) loi_1: *true* ⟹ with ≤(p-niveau-lubrifiant,10) ≥(p-niveau-lubrifiant,5);
% contraindre la température
(5) loi_2: in =(diodes,clignotantes) ⟹ with ≤(p-niveau-lubrifiant,5)
end

Figure A.C.5 (suite): Lois du $\mathcal{CC}$ "contrôleur".

(1) électro-vanne(pompe) ;
begin
% contraindre l'indicateur-vanne
(2) loi_1: in =(position-ouverture,faible) ⟹ in <(indicateur-vanne,6) ≥(indicateur-vanne,5);
(3) loi_1: in =(position-ouverture,moyenne) ⟹ in <(indicateur-vanne,7) ≥(indicateur-vanne,6);
(4) loi_1: in =(position-ouverture,grande) ⟹ in ≤(indicateur-vanne,8) ≥(indicateur-vanne,7);
end

Figure A.C.6 (suite): Lois du $\mathcal{CC}$ "électro-vanne".

```
(1) moteur(pompe) ;
(2) loi1 : 2 ;
(3) loi2 : 1 ;
    begin
    % contraindre globalement la vitesse
(4) loi1: in =(état,marche) ⟹ in ≤(vitesse,3000) ≥(vitesse,1000);
(5) loi2: in <(vitesse,1000) ≥(vitesse,500) ⟹ in ≥(intensité,0) <(intensité,50) ;
(6) loi2: in <(vitesse,2500) ≥(vitesse,a000) ⟹ in ≥(intensité,50) <(intensité,125) ;
(7) loi2: in ≤(vitesse,2500) ≥(vitesse,3500) ⟹ in ≥(intensité,150) ≤(intensité,125) ;
    end
```

Figure A.C.7 (suite): Lois du $\mathcal{CC}$ "moteur".

C.3 Spécification de l'interface

```
(1) lubrificateur(cisaille) ;
    begin
    STATE
(2) available;
(3) débit : assign historique-capteurs-dbi;
(4) puissance : assign historique-capteurs-pss;
(5) pression : assign historique-capteurs-pre;
    PORTS
(6) detected
(7) p-mise-M/A : assign historique-opérateur-cisaille;
(8) p-température : assign historique-capteurs-tmp;
    RULES
(9) rules of ≫
(10) ≫(pression) :- =(@(pression),faible) =(pression,moyenne);
(11) ≫(pression) :- =(@(pression),moyenne) =(pression,haute);
(12) ≫(puissance) :- =(@(puissance),faible) =(puissance,moyenne);
(13) ≫(puissance) :- =(@(puissance),moyenne) =(puissance,haute) ;
(14) ≫(débit) :- =(@(débit),faible) =(débit,moyen);
(15) ≫(débit) :- =(@(débit),moyen) =(puissance,haut) ;
(16) rules of ≪
(17) ≪(pression) :- =(@(pression),moyenne) =(pression,faible) ;
(18) ≪(pression) :- =(@(pression),haute) =(pression,moyenne) ;
(19) ≪(puissance) :- =(@(puissance),moyenne) =(puissance,faible) ;
(20) ≪(puissance) :- =(@(puissance),haute) =(puissance,moyenne) ;
(21) ≪(débit) :- =(@(débit),moyen) =(débit,faible);
(22) ≪(débit) :- =(@(débit),haut) =(puissance,moyen) ;
(23) rules of ≈
(24) ≈(puissance) :- =(@(puissance),puissance);
(25) ≈(pression) :- =(@(pression),pression);
    end
```

Figure A.C.1 (suite): Interface du $\mathcal{CA}$ "lubrificateur".

```
(1) régulateur(lubrificateur) ;
    begin
    PORTS
(2) detected
(3) p-niveau-lubrifiant : assign historique-capteurs-niv;
(4) p-température : assign historique-capteurs-tmp;
    end
```

Figure A.C.3 (suite): Interface du $\mathcal{CT}$ "régulateur".

```
(1) pompe(lubrificateur);
    begin
    STATE
(2) available;
(3) état-électro-vanne : assign historique-états;
(4) puissance : assign historique-capteurs-pss;
(5) débit : assign historique-capteurs-dbi;
    PORTS
(6) detected
(7) p-puissance : assign historique-capteurs-pss;
    RULES
(8) rules of ≫
(9) ≫(pression) :- =(@(pression),faible) =(pression,moyenne);
(10) ≫(pression) :- =(@(pression),moyenne) =(pression,haute);
(11) ≫(débit) :- =(@(débit),faible) =(débit,moyen);
(12) ≫(débit) :- =(@(débit),moyen) =(débit,haut);
    rules of ≪
(13) ≪(pression) :- =(@(pression),moyenne) =(pression,faible) ;
(14) ≪(pression) :- =(@(pression),haute) =(pression,moyenne) ;
(15) ≪(débit) :- =(@(débit),moyenne) =(débit,faible) ;
(16) ≪(débit) :- =(@(débit),haute) =(débit,moyenne) ;
    rules of ≈
(17) ≈(puissance) :- =(@(puissance),puissance);
    end
```

Figure A.C.3 (suite): Interface du $\mathcal{CT}$ "pompe".

```
(1) débimètre(régulateur) ;
    begin
    STATE
(2) available;
(3) température : assign historique-capteurs-tmp;
    PORTS
(4) detected
(5) p-température : assign historique-capteurs-tmp;
    end
```

Figure A.C.4 (suite): Interface du $\mathcal{CC}$ "débimètre".

```
(1) controleur(régulateur) ;
    begin
    STATE
(2) asked;
(3) diodes : assign console-opérator;
    PORTS
(4) detected
(5) p-niveau-lubrifiant : assign historique-capteurs-niv;
    end
```

Figure A.C.5 (suite): Interface du $\mathcal{CC}$ "contrôleur".

```
(1) électro-vanne(pompe) ;
    begin
    STATE
(2) available;
(3) position-ouverture : assign historique-états;
(4) indicateur-vanne : assign historique-états;
    end
```

Figure A.C.6 (suite): Interface du $\mathcal{CC}$ "électro-vanne".

```
(1) moteur(pompe) ;
    begin
    STATE
(2) available;
(3) vitesse : assign historique-états;
(4) intensité : assign historique-états;
    end
```

Figure A.C.7 (suite): Interface du $\mathcal{CT}$ "moteur".

C.4 Spécification du diagnostic

```
(1) lubrificateur(cisaille);
    % déclaration des différentes directives de
    % diagnostic et de leurs priorités.
(2) diag1 : 1;
(3) diag2 : 2;
    begin
(4) diag1: when loi1 ↦ />("Surveiller capteur température");
(5) diag2: when loi2 ↦ focus(pompe);
(6) diag2: when loi3 ↦ focus(régulateur);
    end
```

Figure A.C.1 (suite): Diagnostic du $\mathcal{CA}$ "lubrificateur".

```
(1) régulateur(lubrificateur);
    % déclaration des différentes directives de
    % diagnostic et de leurs priorités.
(2) diag1 : 1;
(3) diag2 : 2;
    begin
(4) diag1: when loi1 ↦ />("Surveiller capteur niveau lubrifiant");
(5) diag2: when loi2 ↦ user(procédure1-diag-pompe);
    end
```

Figure A.C.2 (suite): Diagnostic du $\mathcal{CT}$ "régulateur".

```
(1) pompe(lubrificateur);
    % déclaration des différentes directives de
    % diagnostic et de leurs priorités.
(2) diag1 : 1;
(3) diag2 : 1;
    begin
(4) diag1: when loi1 ↦ user(procédure2-diag-pompe);
(5) diag2: when loi2 ↦ user(procédure3-diag-pompe);
    end
```

Figure A.C.3 (suite): Diagnostic du $\mathcal{CT}$ "pompe".

Bibliographie

[Alanche 86] A. Alanche, P. Lhoste, G. Morel, M. Salmi, et P. Salvi. Application de la modélisation de la partie opérative à la structuration de la commande. *Proceedings of Congrés Afcet Automatique. Méthodes et Outils Modernes de Conception et d'Exploitation de la Commande Numerique des Procédés Discontinus Complexes, 6-7 Mars Montpelleir (France)*, pages 1–14, 1986.

[Ancelin 87] J. Ancelin, J.P Gaussot, et P. Legaud. Un système expert temps réel pour le traitement des alarmes. *Proceedings of Cognitiva Image Electronique Paris (France), 18–22 Mai*, pages 215–217, 1987.

[Ayache 80] J.M. Ayache, M. Diaz, et J. Noubel. Conception d'un logiciel d'autotest pour systèmes temps réel distribuées. Internal Report LAAS Toulouse (France),, 1980.

[Berry 84] G. Berry et B. Serlet. Cyacc et lex-kit: générateurs d'analyseurs syntaxiques et lexicaux pour ceyx/lelisp. Internal Report INRIA Paris (France), 1984.

[Bibel 85] W. Bibel. An advanced course. *Fundamentals of Artificial Intelligence*, Lecture Notes in Computer Science (232), Springer Verlag, Munchen (R.F.A) and Grenoble (France), April 1985.

[Blum 83a] R.L Blum. Detecting ambiguity: an example in knowledge evaluation. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 182–184, 1983.

[Blum 83b] R.L Blum. Representation of empirically derived causal relationships. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 268–271, 1983.

[Bogler 87] P.L Bogler. Shafer-dempster reasoning with applications to multisensor target identification systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 968–977, Nov./Dec. 1987.

[Borras 87] P. et al. Borras. Centaur: the system. Internal Report num. 777, INRIA Paris (France), 1987.

[Cantone 83] R. Cantone, J. Pipitone, B. Lander, et M. Marrone. Model-based probabilistic reasoning for electronics troubleshooting. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 207–211, 1983.

[Chandrasekaran 83] B. Chandrasekaran et S. Mittal. On deep versus compiled approaches to diagnostic problem solving. *International Journal of Man Machine Studies*, 19:425–436, 1983.

[Clema 86] J.K. Clema et A. Werling, R. Chande. Knowledge bases and machine learning. *Proceedings of the First International Conference on Applications of Artificial Intelligence in Engineering Londres (U-K)*, pages 67–75, 1986.

[Cohen 86] P. Cohen. Numeric and symbolic reasoning in expert systems. *Proceedings of the European Conference on Artificial Intelligence*, pages 413–427, 1986.

[Dague 86] P. Dague, P. Devès, et O. Raiman. Raisonnement qualitatif dans le diagnostic de pannes. *Journées Les Systèmes Experts Avignon (France)*, pages 613–631, 1986.

[David 86] J-M. David et J-P. Kirvine. Reasoning from structure and behavior: four relevance criteria. *Proceedings of the European Conference on Artificial Intelligence*, pages 114–119, 1986.

[Davis 84] R. Davis. Diagnostic reasoning based on structure and behaviour. *Artificial Intelligence*, 24:347–410, 1984.

[DeKleer 84] J. De-Kleer et J. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83, 1984.

[DeKleer 86] J. De-Kleer et J.S. Brown. Theories of causal ordering. *Artificial Intelligence*, 29:33–61, 1986.

[DeKleer 87] J. De-Kleer. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

[DeMarco 79] T. De-Marco. *Structured Analysis and System specification.* A Yourdon Book, Prentice Hall Englewood Cliffs, 1979.

[Descotes 84] B. Descotes, B. Luca, E. Picard, et G. Poncet. Surveillance et controle en ligne du fonctionnement des rames de metro. *Colloque International S.E.E Automatique Appliquée Nice (France)*, 1984.

[Dormoy 86] J.L Dormoy et O. Raiman. Physique qualitative: la représentation des connaissances en physique qualitative pour l'intelligence artificielle. *Proceeding of Journées Nationales sur l'Intelligence Artificielle Aix-les-Bains 20–21 Novembre*, pages 42–51, 1986.

[Driankov 86] D. Driankov. Uncertainty in knowledge-based systems. *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Springer Verlag, Paris (France), July 1986.

[Eddy 86] W.F. Eddy et G.P. Pei. Structures of rule-based belief functions. *IBM Research and Development Journal*, 30:93–101, 1986.

[ElAyeb 84] B. El Ayeb. Conception et réalisation d'une boîte à outils pour un environnement de spécification. Mémoire de stage de 5ième Année, Institut d'Informatique à Namur (Belgique), 1984.

[ElAyeb 85] B. El Ayeb. Le diagnostic de pannes: l'exemple delta. 1985. Technical Report, organization: crin (France).

[ElAyeb 88a] B. El Ayeb. Méthodologie du diagnostic industriel. *Ecole de Printemps de Mathématiques et d'Intelligence Artificielle à Nabeul (Tunisia)*, Mars/Avril 1988.

[ElAyeb 88b] B. El Ayeb et J-P. Finance. Conventional databases / inference process. *4ième Journées Internationales des Sciences Informatiques à Tunis (Tunisia)*, April 1988.

[ElAyeb 88c] B. El Ayeb et J-P. Finance. On cooperation between deep and shallow reasoning. *Proceedings of the Third International Conference on Applications of Artificial Intelligence in Engineering, Palo Alto–Stanford USA*, August 1988.

[ElAyeb 88d] B. El Ayeb et J. Parravano. Une méthode de diagnostic et un domaine d'application. *4th. International Conference in Health Care Systems, Lyon (France)*, July 1988.

[ElAyeb 89] B. El Ayeb. Le diagnostic de pannes dans les grandes installations industrielles. *Procceeding of Conférence et Exposition Européenne*

sur les Techniques et les Applications Avancées d'IA en Milieu industriel, Porte de Versailles Paris (France), A paraître, Junnary 1989.

[Fagin 88] R. Fagin et J.Y. Halpern. Belief, awareness, and limited reasoninig. *Artificial Intelligence*, 34:39–76, 1988.

[Finance 79] J-P. Finance. Etude de la construction des programmes: méthodes et langages de spécification et de résolution de problèmes. Thèse d'état Université de Nancy I, 1979.

[Fink 86] P.K. Fink, J.C. Lusth, et J.W. Duran. A general expert system design for diagnostic problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 553–560, Sep. 1986.

[Forbus 84] K. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:169–203, 1984.

[Foucaut 82] O. Foucaut. Modèle et outil pour la conception des systèmes d'information dans les organisations, projet rrmora. Thèse d'état Université de Nancy I, 1982.

[Freiling 86] M.J. Freiling, S. Rehfuss, S.L. Alexander, J.H Messick, et Shulman S.J. The ontological structure of a troubleshooting system for electronic instruments. *Proceedings of the First International Conference on Applications of Artificial Intelligence in Engineering Londres (U-K)*, pages 609–619, 1986.

[Gabriel 87] M. Gabriel et J-C. Rault. *Systèmes Experts et Maintenance.* Editions Masson, 1987.

[Gallanti 85] M. Gallanti, G. Giovanni, L. Spampinato, et A. Stefanini. Representing procedural knowledge in expert systems: an application to process control. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 345–352, 1985.

[Gallanti 86] M. Gallanti, L. Gilardino, G. Guida, et A. Stefanini. Exploiting physical and design knowledge in the diagnosis of complex industrial systems. *Proceedings of the European Conference on Artificial Intelligence*, pages 335–349, 1986.

[Garvey 80] T. Garvey et Fishler. The integration of multisensor data for threat assessment. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 343–347, 1980.

[Geffener 87] H. Geffener et J. Pearl. An improved constraint-propagation algorithm for diagnosis. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1105–1111, 1987.

[Genesereth 84] M.R. Genesereth. The use of design descriptions in automated diagnosis. *Artificial Intelligence*, 24:411–436, 1984.

[Gentil 87] S. Gentil, Féray B. S., et Caloud P. Qualitative modelling for process supervision systems. *Proceeding of First European Meeting on Cognitive Sciences Approaches to Process Control*, Octobre 1987.

[Godart 86] C. Godart, K. Benali, et J.C. Derniame. Propositions sur un système de gestion d'objets. *Proceedings of 3ième Colloque du Génie Logiciel, 27-30 May, Versailles (France)*, 1986.

[Gondran 79] M. Gondran et A. Pages. *Fiabilité des Systèmes.* Editions Eyrolles, 1979.

[Gong 88] Y. Gong. *Contribution à l'interprétation automatique des signaux en présence d'incertitude.* Thèse de Doctorat, Université de Nancy I, 1988.

[Habermann 81] A.N. Habermann. System development environments. ***Ecole d'Hiver sur les Outils de Construction des Programmes, Nice (France)*, Décembre 1981.**

[Haton 86] J-P. Haton et al. *Systèmes Experts: Vers la maîtrise technique.* Inter-Editions, 1986.

[Haton 88] J-P. Haton, M-C. Haton, B. Lupin, E. Vion, et T. Riggy. Formation aux consignes d'exploitation en metallurgie assisté par ordinateur: le système CONSOL. *Journées Les Systèmes Experts Avignon (France)*, pages 455–465, 1988.

[Haton 89] J-P. Haton. Panorama des systèmes multi-agents. *Proceedings of onzièmes journées francophones sur l'informatique, Architectures Avancées pour l'Intelligence Artificielle Nancy (France)*, pages 247–261, 1989.

[Hayes 83] R. Hayes. *Building Expert Systems.* Addison-Wesley, 1983.

[Hummel 88] R.A. Hummel et M.S. Landy. A statistical viewpoint on the theory of evidence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 235–247, Mar. 1988.

[Iwasaki 86a] Y. Iwasaki et A. H. Simon. Causality in device behavior. *Artificial Intelligence*, 29:3–32, 1986.

[Iwasaki 86b] Y. Iwasaki et A. H. Simon. Theories of causal ordering: reply to de kleer and brown. *Artificial Intelligence*, 29:63–72, 1986.

[Jacquot 84] J.P Jacquot. Etude d'un outil d'assistance à la conception méthodique de programme: réalisation et évaluation d'une maquette. Thèse de Docteur-Ingénieur Institut Polytechnique de Lorraine Nancy (France),, 1984.

[Jakob 84] F. Jakob et D. Vernet. Extase: un système expert de diagnostic d'alarmes. 1984. Technical Report Marcoussis 84 CGE (France).

[Kanoun 87] K. Kanoun et T. Sabourin. Analyse des défaillences et evaluation de la sûreté de fonctionnement du logiciel d'un autocommutateur téléphonique. *Techniques et Sciences de l'Informatique*, 6:287–303, 1987.

[Khanna 86] R. Khanna et R.L. Moore. Expert systems involving dynamic data for decisions. *Proceedings of the Second International Expert Systems Conference Londres (U-K)*, pages 73–83, 1986.

[Kreutzer 88] S.E. Kreutzer et S.L. Hakimi. Distributed diagnosis and the system user. *IEEE Transactions on Computers*, 71–78, Jun. 1988.

[Kuipers 86] B. Kuipers. Qualtative simulation. *Artificial Intelligence*, 29:289–338, 1986.

[Laasri 88] H. Lâasri, B. Maître, T. Mondot, F. Charpillet, et J-P. Haton. ATOME a blackboard architecture with temporal and hypothetical reasoning. *Proceedings of the* 8^{th} *European Conference on Artificial Intelligence*, pages 5–10, 1988.

[Laasri 89] H. Lâasri et B. Maître. *Coopération dans un univers multi-agents basée sur le modèle du blackboard: Etudes et réalisations.* Thèse de Doctorat, Université de Nancy I, 1989.

[Laprie 85] J.C. Laprie. Sûreté de fonctionnement des systèmes informatiques et tolérance aux fautes: concepts de base. *Techniques et Sciences de l'Informatique*, 4:419–429, 1985.

[Laurent 84] J-P. Laurent. La structure de contrôle dans les systèmes experts. *Techniques et Sciences de l'Informatique*, 3, 1984.

[Lauriere 82] J-L. Laurière. Représentation et utilisation des connaissances. *Techniques et Sciences de l'Informatique*, 1:109–133, 1982.

[Lauriere 87] J-L. Laurière. *Intelligence Artificielle Résolution de problèmes par l'Homme et la machine.* Editions Eyrolles, 1987.

[Lee 88] C.H. Lee. A comparison of two evidential reasoning schemes. ***Artificial Intelligence***, 35:127–134, 1988.

[Lepetit 87] M. Lepetit et D. Vernet. Physique qualitative et contrôle de processus. *Journées Les Systèmes Experts Avignon (France)*, pages 1230–1248, 1987.

[Li 86] T. Li. Heuristic search in digital system diagnostic. *Proceedings of the First International Conference on Applications of Artificial Intelligence in Engineering Londres (U-K)*, pages 937–945, 1986.

[Lievens 76] C. Lievens. *Sécurité des Systèmes.* Editions SUP'AERO (France), 1976.

[Marin 76] J. Marin, C. André, et F. Boeri. Conception de systèmes séquentiels totalement auto-testables à partir des réseaux de petri. *R.A.R.O Informatique*, 10:5–22, 1976.

[Marrakchi 86] M. Marrakchi. Représentation des connaissances pour l'aide au diagnostic industriel: application au système expert: s.e.diag. Thèse de Docteur Ingénieur -Université de Valenciennes et Hainnaut Cambresis, 1986.

[Martins 88] J.P Martins et S.C. Shapiro. A model for belief revision. ***Artificial Intelligence***, 35:25–79, 1988.

[Mathonet 87] R. Mathonet, H. Van-Cotthem, et L. Vanryckeghem. Dantes an expert system for real-time network troubleshooting. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 527–530, 1987.

[Milne 87] R. Milne. On-line artificial intelligence. *Journées Les Systèmes Experts Avignon (France)*, pages 437–451, 1987.

[Moore 86] R.L. Moore. Expert systems in process control: applications experiences. *Proceedings of the First International Conference on Applications of Artificial Intelligence in Enginnering Londres (U-K)*, pages 21–30, 1986.

[Nau 83] D. Nau. Expert computer system. *IEEE Computer*, 63–85, Feb. 1983.

[Pau 75] L.F. Pau. *Failure Diagnosis and Performance Monitoring.* Editions Marcel Dekker –INC– New York and Basel, 1975.

[Pau 76] L.F. Pau. *Diagnostic des pannes dans les systèmes.* Editions Cepadues (France), 1976.

[Rajagoplan 84] R. Rajagoplan. Qualitative modeling in the turbojet engine domain. *Proceedings of the AAAI 84*, pages 283–287, 1984.

[Reboth 83] R. Reboth. Extracting useful advice from conflicting expertise. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 145–150, 1983.

[Reiter 87] R. Reiter. A theory of diagnostic from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[Ricard 86] B. Ricard, B. Monnier, et J. Morel. Système expert d'aide au diagnostic de défauts d'un groupe turbo-alternateur. *Proceedings of the Second International Conference on Artificial Intelligence*, pages 233–259, IIRIAM, Marseille (France), 1986.

[Richardson 85] Richardson et al. *Artificial Intelligence in Maintenance: Synthesis of Technical Issues.* Rapport no. AFHRL-TR-85-7, Air Force Systems Command, Air Force Human Resources Laborotary, Brooks Air Force Base, Texas 78235, 1985.

[Ross 77] D.T Ross et K.G. Schoman. Structured analysis for requirements definition. *IEEE Transactions on Software Engineering*, 1–65, Sep. 1977.

[Shafer 76] G. Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, 1976.

[Skatteboe 86] R. Skatteboe, E. Lihovd, et R.A. Hystad. Diamon: a knowledge based system for fault diagnosis and maintenance planning for rotating machnery. *Proceedings of the Second International Conference on Artificial Intelligence*, pages 633–647, IIRIAM, Marseille (France), 1986.

[Sombe 88] L. Sombé. Inférences non-classiques en intelligence artificielle. *Proceeding of Journées Nationales sur l'Intelligence Artificielle Toulouse 14–15 Mars*, pages 137–230, 1988.

[Taunton 86] J.C. Taunton et D.W. Haspel. Rule based expert systems– an application in real time process control and optimisation. *Proceedings of the Second International Expert Systems Conference Londres (U-K)*, pages 55–66, 1986.

[Thelliez 80] S. Thelliez et J-M. Toulotte. *Grafcet et Logique Industrielle Programée*. Eyrolles (France), 1980.

[Tong 83] R.M. Tong et D.G. Shapiro. A comparison of uncertainty calculi in an expert system for information retreival. *Proceedings of the International Joint Conference c Artificial Intelligence*, pages 194–197, 1983.

[VanLamsweerde 82] A. Van Lamsweerde. Automatisation de la production du logiciels d'application: quelques approches. *Techniques et Sciences de l'Informatique*, 1, 1982.

[Weilinga 86] B.J. Weilinga et J.A. Breuker. Models of expertise. *Proceedings of the European Conference on Artificial Intelligence*, pages 306–318, 1986.

[Xiang 86] Z. Xiang et S. N. Srihari. A strategy for diagnosis based on empirical and model knowledge. *Journées Les Systèmes Experts Avignon (France)*, pages 835–848, 1986.

NOM DE L'ETUDIANT : EL AYEB Béchir

NATURE DE LA THESE : Doctorat de l'Université de NANCY I en Informatique

VU, APPROUVE ET PERMIS D'IMPRIMER

NANCY, le 23.02.1989 n°366

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

M. BOULANGÉ

COMPENDIUM

Si de nombreux travaux sont menés dans le domaine des techniques de diagnostic, il n'en est pas de même dans le domaine de la méthodologie de diagnostic. L'objectif principal poursuivi par cette thèse est de spécifier et de construire **méthodiquement** les systèmes de diagnostic pour les grandes installations industrielles. La contribution et l'originalité de cette thèse se situent tant au niveau méthodologique qu'à celui de la recherche d'un **langage** de **spécification** et des **outils** de **construction** des systèmes de diagnostic. Schématiquement, notre contribution peut se résumer comme suit :

Partant d'une installation, qui est en général un objet complexe et difficile à appréhender, nous proposons *une méthode de spécification* qui permet de :

(1) structurer l'installation en hiérarchies et graphes de différents niveaux d'abstraction.

(2) spécifier partiellement les composants de l'installation en exprimant successivement la structure, le comportement, les contraintes et l'interface de chacun d'entre eux.

Mais, ces spécifications se limitent à la description de l'installation et n'indiquent pas comment conduire un diagnostic pour déterminer les pannes de l'installation. Nous sommes conduits à faire *une étude de synthèse des techniques de diagnostic* dont le but est de dégager les mécanismes de base tout en négligeant les notions secondaires et conjoncturelles. De cette étude, il ressort quatre grandes classes de techniques. Pour chaque classe nous identifions son principe de base, ses avantages et ses restrictions. Nous proposons ensuite des solutions aux restrictions exposées.

Partant de cette synthèse mais aussi des caractéristiques des grandes installations, nous proposons *une méthode d'élaboration du diagnostic* où:

(1) nous définissons, dans un premier temps, les concepts de base (i.e symptôme, panne) avant d'introduire les notions de stratégies d'exploration, de tactiques de "suspiscion", et de paradigmes de justification.

(2) nous montrons, dans un deuxième temps, comment compléter les spécifications partielles pour inclure les connaissances de diagnostic.

(3) nous proposons enfin, un mécanisme de diagnostic qui fait coopérer deux types de raisonnements parfois considérés comme opposés.

Mais s'il est indispensable de disposer des méthodes de spécification et d'élaboration de diagnostic, il est aussi important :

(1) de proposer les outils nécessaires à la bonne conduite et la mise en oeuvre des méthodes proposées; c'est pourquoi nous proposons *un ensemble d'outils* d'aide à la spécification et à la construction des systèmes de diagnostic. Ces outils adjoints aux méthodes proposées constituent un *environnement* permettant la génération des systèmes de diagnostic pour les grandes installations industrielles.

(2) d'expérimenter les méthodes et les outils sur des cas réels pour les valider; c'est pourquoi nous avons réalisé en LISP le prototype *SIDI* que nous avons expérimenté sur des installations sidérurgiques.