USTL FLANDRES ARTOIS — LIFL U.A. 369 du C.N.R.S.
LABORATOIRE D'INFORMATIQUE FONDAMENTALE DE LILLE

N° d'ordre : 630

THESE

Nouveau régime

présentée à

l'Université des Sciences et TECHNIQUES de Lille Flandres Artois

pour obtenir le titre de

DOCTEUR en INFORMATIQUE

par

Claude LECLERCQ

UN PROBLEME DE SYSTEME EXPERT TEMPS REEL LA GESTION DE CENTRES INFORMATIQUES

Thèse soutenue le 11 Décembre 1990 devant la commision d'examen

Membres du jury

Président	E. DELATTRE
Rapporteurs	E. DELATTRE
	P. CORNU
Directeur de thèse	J.P. DELAHAYE
Examinateurs	B. DRIEUX
	D. WILLAEYS
Invité	L.M. DESMARCHELIER

UNIVERSITE DES SCIENCES
ET TECHNIQUES DE LILLE
FLANDRES ARTOIS

DOYENS HONORAIRES DE L'ANCIENNE FACULTE DES SCIENCES

M.H. LEFEBVRE, M. PARREAU.

PROFESSEURS HONORAIRES DES ANCIENNES FACULTES DE DROIT ET SCIENCES ECONOMIQUES, DES SCIENCES ET DES LETTRES

MM. ARNOULT, BONTE, BROCHARD, CHAPPELON, CHAUDRON, CORDONNIER, DECUYPER, DEHEUVELS, DEHORS, DION, FAUVEL, FLEURY, GERMAIN, GLACET, GONTIER, KOURGANOFF, LAMOTTE, LASSERRE, LELONG, LHOMME, LIEBAERT, MARTINOT-LAGARDE, MAZET, MICHEL, PEREZ, ROIG, ROSEAU, ROUELLE, SCHILTZ, SAVARD, ZAMANSKI, Mes BEAUJEU, LELONG.

PROFESSEUR EMERITE

M. A. LEBRUN

ANCIENS PRESIDENTS DE L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

MM. M. PAREAU, J. LOMBARD, M. MIGEON, J. CORTOIS.

PRESIDENT DE L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE FLANDRES ARTOIS

M. A. DUBRULLE.

PROFESSEURS - CLASSE EXCEPTIONNELLE

M. CONSTANT Eugène	Electronique
M. FOURET René	Physique du solide
M. GABILLARD Robert	Electronique
M. MONTREUIL Jean	Biochimie
M. PARREAU Michel	Analyse
M. TRIDOT Gabriel	Chimie Appliquée

PROFESSEURS - 1ère CLASSE

M. BACCHUS Pierre	Astronomie
M. BIAYS Pierre	Géographie
M. BILLARD Jean	Physique du Solide
M. BOILLY Bénoni	Biologie
M. BONNELLE Jean-Pierre	Chimie-Physique
M. BOSCQ Denis	Probabilitéss
M. BOUGHON Pierre	Algèbre
M. BOURIQUET Robert	Biologie Végétale
M. BREZINSKI Claude	Analyse Numérique

M. BRIDOUX Michel	Chimie-Physique
M. CELET Paul	Géologie Générale
M. CHAMLEY Hervé	Géotechnique
M. COEURE Gérard	Analyse
M. CORDONNIER Vincent	Informatique
M. DAUCHET Max	Informatique
M. DEBOURSE Jean-Pierre	Gestion des Entreprises
M. DHAINAUT André	Biologie Animale
M. DOUKHAN Jean-Claude	Physique du Solide
M. DYMENT Arthur	Mécanique
M. ESCAIG Bertrand	Physique du Solide
M. FAURE Robert	Mécanique
M. FOCT Jacques	Métallurgie
M. FRONTIER Serge	Ecologie Numérique
M. GRANELLE Jean-Jacques	Sciences Economiques
M. GRUSON Laurent	Algèbre
M. GUILLAUME Jean	Microbiologie
M. HECTOR Joseph	Géométrie
M. LABLACHE-COMBIER Alain	Chimie Organique
M. LACOSTE Louis	Biologie Végétale
M. LAVEINE Jean-Pierre	Paléontologie
M. LEHMANN Daniel	Géométrie
Mme LENOBLE Jacqueline	Physique Atomique et Moléculaire
M. LEROY Jean-Marie	Spectrochimie
M. LHOMME Jean	Chimie Organique Biologique
M. LOMBARD Jacques	Sociologie
M. LOUCHEUX Claude	Chimie Physique
M. LUCQUIN Michel	Chimie Physique
M. MACKE Bruno	Physique Moléculaire et Rayonnements Atmosph.
M. MIGEON Michel	E.U.D.I.L.
M. PAQUET Jacques	Géologie Générale
M. PETIT Francis	Chimie Organique
M. POUZET Pierre	Modélisation - calcul Scientifique
M. PROUVOST Jean	Minéralogie
M. RACZY Ladislas	Electronique
M. SALMER Georges	Electronique
M. SCHAMPS Joel	Spectroscopie Moléculaire
M. SEGUIER Guy	Electrotechnique
M. SIMON Michel	Sociologie
Melle SPIK Geneviève	Biochimie
M. STANKIEWICZ François	Sciences Economiques
M. TILLIEU Jacques	Physique Théorique
M. TOULOTTE Jean-Marc	Automatique
M. VIDAL Pierre	Automatique
M. ZEYTOUNIAN Radyadour	Mécanique

PROFESSEURS - 2ème CLASSE

M. ALLAMANDO Etienne	Composants Electroniques
M. ANDRIES Jean-Claude	Biologie des organismes
M. ANTOINE Philippe	Analyse
M. BART André	Biologie animale
M. BASSERY Louis	Génie des Procédés et Réactions Chimiques

Mme BATTIAU Yvonne	Géographie
M. BEGUIN Paul	Mécanique
M. BELLET Jean	Physique Atomique et Moléculaire
M. BERTRAND Hugues	Sciences Economiques et Sociales
M. BERZIN Robert	Analyse
M.BKOUCHE Rudolphe	Algèbre
M. BODARD Marcel	Biologie Végétale
M. BOIS Pierre	Mécanique
M. BOISSIER Daniel	Génie Civil
M. BOIVIN Jean-Claude	Spectroscopie
M. BOUQUELET Stéphane	Biologie Appliquée aux enzymes
M. BOUQUIN Henri	Gestion
M. BRASSELET Jean-Paul	Géométrie et Topologie
M. BRUYELLE Pierre	Géographie
M. CAPURON Alfred	Biologie Animale
M. CATTEAU Jean-Pierre	Chimie Organique
M. CAYATTE Jean-Louis	Sciences Economiques
M. CHAPOTON Alain	Electronique
M. CHARET Pierre	Biochimie Structurale
M. CHIVE Maurice	Composants Electroniques Optiques
M. COMYN Gérard	Informatique Théorique
M. COQUERY Jean-Marie	Psychophysiologie
M. CORIAT Benjamin	Sciences Economiques et Sociales
Mme CORSIN Paule	Paléontologie
M. CORTOIS Jean	Physique Nucléaire et Corpusculaire
M. COUTURIER Daniel	Chimie Organique
M. CRAMPON Norbert	Tectolique Géodynamique
M. CROSNIER Yves	Electronique
M. CURGY Jean-Jacques	Biologie
Melle DACHARRY Monique	Géographie
M. DEBRABANT Pierre	Géologie Appliquée
M. DEGAUQUE Pierre	Electronique
M. DEJAEGER Roger	Electrochimie et Cinétique
M. DELAHAYE Jean-Paul	Informatique
M. DELORME Pierre	Physioliogie Animale
M. DELORME Robert	Sciences Economiques
M. DEMUNTER Paul	Sociologie
M. DENEL Jacques	Informatique
M. DE PARIS Jean Claude	Analyse
M. DEPREZ Gilbert	Physique du Solide - Cristallographie
M. DERIEUX Jean-Claude	Microbiologie
Melle DESSAUX Odile	Spectroscopie de la réactivité Chimique
M. DEVRAINNE Pierre	Chimie Minérale
Mme DHAINAUT Nicole	Biologie Animale
M. DHAMELINCOURT Paul	Chimie Physique
M. DORMARD Serge	Sciences Economiques
M. DUBOIS Henri	Spectroscopie Hertzienne
M. DUBRULLE Alain	Spectroscopie Hertzienne
M. DUBUS Jean-Paul	Spectrométrie des Solides
M. DUPONT Christophe	Vie de la firme (I.A.E.)
Mme EVRARD Micheline	Génie des procédés et réactions chimiques
M. FAKIR Sabah	Algèbre
M. FAUQUAMBERGUE Renaud	Composants électroniques

M. FONTAINE Hubert	Dynamique des cristaux
M. FOUQUART Yves	Optique atmosphérique
M. FOURNET Bernard	Biochimie Sturcturale
M. GAMBLIN André	Géographie urbaine, industrielle et démog.
M. GLORIEUX Pierre	Physique moléculaire et rayonnements Atmos.
M. GOBLOT Rémi	Algèbre
M. GOSSELIN Gabriel	Sociologie
M. GOUDMAND Pierre	Chimie Physique
M. GOURIEROUX Christian	Probabilités et Statistiques
M. GREGORY Pierre	I.A.E.
M. GREMY Jean-Paul	Sociologie
M. GREVET Patrice	Sciences Economiques
M. GRIMBLOT Jean	Chimie Organique
M. GUILBAULT Pierre	Physiologie animale
M. HENRY Jean-Pierre	Génie Mécanique
M. HERMAN Maurice	Physique spatiale
M. HOUDART René	Physique atomique
M. JACOB Gérard	Informatique
M. JACOB Pierre	Probabilités et Statistiques
M. Jean Raymond	Biologie des populations végétales
M. JOFFRE Patrick	Vie de la firme (I.A.E.)
M. JOURNEL Gérard	Spectroscopie hertzienne
M. KREMBEL Jean	Biochimie
M. LANGRAND Claude	Probabilités et statistiques
M. LATTEUX Michel	Informatique
Mme LECLERCQ Ginette	Catalyse
M. LEFEBVRE Jacques	Physique
M. LEFEBVRE Christian	Pétrologie
Melle LEGRAND Denise	Algèbre
Melle LEGRAND Solange	Algèbre
M. LEGRAND Pierre	Chimie
Mme LEHMANN Josiane	Analyse
M. LEMAIRE Jean	Spectroscopie hertzienne
M. LE MAROIS Henri	Vie de la firme (I.A.E.)
M. LEROY Yves	Composants électroniques
M. LESENNE Jacques	Systèmes électroniques
M. LHENAFF René	Géographie
M. LOCQUENEUX Robert	Physique théorique
M. LOSFELD Joseph	Informatique
M. LOUAGE Francis	Electronique
M. MAHIEU Jean-Marie	Optique-Physique atomique
M. MAIZIERES Christian	Automatique
M. MAURISSON Patrick	Sciences Economiques et Sociales
M. MESMACQUE Gérard	Génie Mécanique
M. MESSELYN Jean	Physique atomique et moléculaire
M. MONTEL Marc	Physique du solide
M. MORCELLET Michel	Chimie Organique
M. MORTREUX André	Chimie Organique
Mme MOUNIER Yvonne	Physiologie des structures contractiles
Mme MOUYART-TASSIN Annie Françoise	Informatique
M. NICOLE Jacques	Spectrochimie
M. NOTELET Françis	Systèmes électroniques
M. PARSY Fernand	Mécanique

M. PECQUE Marcel	Chimie organique
M. PERROT Pierre	Chimie appliquée
M. STEEN Jean-Pierre	Informatique

Je tiens à remercier

Monsieur **Jean-Paul DELAHAYE**, *professeur à l'Université de Lille I, qui a dirigé mes recherches, pour son amabilité et sa disponibilité tout au long du projet, et pour le vif intérêt qu'il a porté pour mes développements,*

Messieurs **E. DELATTRE** *et* **P. CORNU** *d'avoir montré un intérêt évident pour ce projet en acceptant d'être rapporteurs de mon mémoire, et de l'aide considérable qu'ils ont apportée pour son élaboration,*

Messieurs **B. DRIEUX, D. WILLAEYS** *d'avoir accepté d'être examinateurs de mon travail,*

Monsieur **L.M. DESMARCHELIER** *d'avoir suivi et contribué à l'élaboration du système expert, et d'avoir accepté l'invitation de participation au jury,*

Monsieur **A. SPRIET**, *Président Directeur Général de CdFi-NATEL, qui m'a accueilli dans son entreprise*

Je remercie tout particulièrement **SYLVIE**, *mon épouse, pour l'aide et le soutien qu'elle m'a apportés tout au long du projet.*

Je tiens à remercier Monsieur **Jean-Pierre BACQUART** *qui m'a intégré dans son service au début de la convention CIFRE, et qui m'a enseigné toutes les bases concernant le fonctionnement du DPS 8, afin d'aborder la partie cognitive avec des connaissances suffisantes.*

Je remercie également tous mes collègues de CdFi-NATEL qui m'ont chaleureusement accepté à leurs cotés.

Claude LECLERCQ

SOMMAIRE

INTRODUCTION

La thèse présentée dans ce mémoire est le résultat de 3 ans de collaboration entre l'Université de Lille I et la Société CdFi-Natel de Villeneuve d'Ascq. Elle faisait l'objet d'une convention C.I.F.R.E., gérée par l'A.N.R.T.

Tout les travaux de recherche et de développement ont été effectués dans les locaux de l'entreprise.

Depuis longtemps déjà, il est convenu de considérer deux approches de l'informatique : l'une dite algorithmique ou impérative et l'autre que l'on appelle l'Intelligence Artificielle (I.A.).

Il est facile de définir l'intelligence artificielle comme la solution informatique à utiliser quand l'algorithmique classique ne suffit pas pour résoudre un problème. En fait, l'I.A. est une discipline dont le but principal est de construire des systèmes dits *"intelligents"*, à savoir qui sont capables de comprendre, ou tout au moins de modéliser une situation donnée. Un système de type I.A. doit être capable de s'adapter à une situation, de faire des analogies et des projections d'idées. Tout cela requiert une souplesse au niveau de la description des objets ou des événements et des différentes stratégies à appliquer. Les domaines concernés par l'intelligence artificielle sont la reconnaissance vocale, la lecture de textes manuscrits, la compréhension de textes, la reconnaissance des formes, la traduction automatique de texte, etc. Les livres de BARR et FEIGENBAUM [BAR 81], WINSTON [WIN 84], RICH [RIC 85] et LAURIERE [LAU 86] sont des ouvrages de référence de présentation générale de l'intelligence artificielle.

Les différents domaines cités ci-dessus font partie de l'Intelligence Artificielle. Il existe un autre domaine de l'I.A. dont on parle depuis les années 70, et de plus en plus maintenant : les Systèmes Experts (S.E.).
Pour en savoir plus sur la technique des S.E., on pourra par exemple consulter [FAR 85].

Le but des S.E. est de simuler le raisonnement humain : il s'agit de modéliser la démarche et le cheminement de la pensée d'un expert.

Depuis 20 ans, de nombreux systèmes experts ont été réalisés et ont fait leurs preuves : MYCIN en 76 (diagnostics de maladies bactériennes du sang), PROSPECTOR en 78 (prospection minière), R1 en 80 (configuration d'ordinateurs VAX), DIABETO en 84 (diagnostic et traitement du diabète) font partie des plus connus.

Un système expert est un logiciel composé d'une base de faits, d'une base de règles et d'un moteur d'inférences. La base de faits constitue la description du système sur lequel le S.E. opère, et la base de règles correspond à la modélisation du cheminement de la pensée d'un expert. Le moteur d'inférences est aussi appelé machine de déductions et permet la simulation de la stratégie de l'expert.

De nombreux S.E. s'exécutent sous forme de dialogues homme-machine par enchaînements de questions réponses et déductions.
On trouve maintenant des systèmes experts temps réel automatisés, où l'homme n'intervient plus pendant le fonctionnement du S.E. Ce cas de figure (c'est-à-dire sans intervention humaine) est fréquemment rencontré pour les S.E. de contrôle de "process" où la description du process est réalisée à partir de capteurs et par des machines spécialisées [SAC 86], [MOO 85]. Dans certains systèmes experts de ce type, il y a interaction entre le système de captage d'information et le moteur d'inférences [NEM 87] et [MOO 85].

Le système expert présenté dans ce mémoire fonctionne avec la même logique que les S.E. de contrôle de processus. La prise d'information est entièrement automatique (dans toute la suite du mémoire, nous appellerons "*métrologie*", la prise ou la capture d'informations).

Le fruit de ces travaux se matérialise aujourd'hui par un système expert d'assistance au pilotage des moniteurs transactionnel sur DPS 8/88/90, prêt à être commercialisé : SEAT (Système Expert d'Assistance au Tuning).

Les motivations qui ont poussé à réaliser ce produit sont diverses, mais toutes caractérisent l'environnement de production de ressource informatique : un grand ordinateur est un environnement complexe ; il est constitué de multiples entités de types différents (unité centrale, disques, frontal, réseau, etc) ; les clients qui utilisent cette ressource sont exigeants et demandent un taux de service et une qualité de ce service très élevés ; la sollicitation machine est de plus en plus élevée ; la charge machine est fluctuante et peu planifiable. Sur les sites informatiques où il y a des grands ordinateurs, se trouvent plusieurs ingénieurs système. Ils constituent un capital considérable grâce à leur expérience et leurs connaissances.

Une question importante vient à l'esprit : "Comment utiliser au mieux cette compétence ?"

Il ne faut pas oublier que le travail de l'expert n'est pas seulement de contrôler un grand ordinateur et d'assurer son bon fonctionnement, c'est aussi d'assurer l'évolution technique des matériels, de mettre en place des nouvelles applications et des nouveaux outils. L'inconvénient de cela est que la partie "noble" de son travail passe souvent en second plan derrière les travaux de routine. Le but des développements présentés dans ce mémoire est justement de modéliser le savoir-faire des ingénieurs systèmes pour simuler leur comportement par un système expert.

Voici toutes les étapes qui ont permis d'aboutir à ce résultat final. Elles présentent toutes les études et tous les développements que j'ai effectués pour parvenir à la version actuelle du logiciel SEAT.

- Etude de faisabilité :
  Cette étude a commencé par une description du travail de l'ingénieur système et de ses principales préoccupations pour veiller au bon fonctionnement d'un moniteur transactionnel sur DPS 8/88/90.
  Ensuite, une maquette a été réalisée en utilisant un programme générateur standard.de systèmes experts. Les résultats satisfaisants et encourageants de cette première étape ont autorisé la suite des événements et donc la deuxième étape.

- Elaboration d'un cahier des charges :
A l'origine du projet, on avait évoqué le concept de la *salle blanche*, c'est-à-dire une salle machine sans présence humaine. Plus réaliste, le cahier des charges décrivait un système expert temps réel de contrôle des moniteurs transactionnels sur DPS 8/88/90.

- Construction d'un prototype :
Basé sur ce cahier des charges, un prototype a été développé : le logiciel APSYST (Assistance au Pilotage des SYStèmes Transactionnels).

- Elaboration d'un progiciel :
Le prototype (utilisé sur le site de CdFi-Natel) apportait des résultats satisfaisants mais n'était pas commercialisable. Il a donc fallu revoir sa convivialité. C'est comme cela que l'on est passé du prototype APSYST au progiciel SEAT.

- Evolution du progiciel :
Des enrichissements fonctionnels ont été apportés pour faire évoluer le progiciel de la version 1 à la version 2.

Ces étapes ont constitué l'essentiel du travail jusqu'en Octobre 1990. Nous les avons toutes menées à terme

Détaillons maintenant le contenu du mémoire qui est composé de six chapitres et de deux annexes (l'annexe A présente des résultats d'utilisation du logiciel, l'annexe B présente la grammaire des règles).

Dans le **premier chapitre**, nous allons voir le rôle d'une société de services en informatique, et une description technique de l'environnement sur lequel s'exécute le système expert temps réel.

Puis, nous verrons une description succincte du DPS 8/88/90 - les entités qui le composent et les différentes sortes de travaux qui tournent sur un grand ordinateur de ce type. Nous nous intéresserons ensuite de plus près à un type particulier d'application qui tourne sur un DPS 8/88/90 : le transaction processor (T.P. ou moniteur transactionnel). Nous verrons plus en détails les différentes parties d'un T.P., leurs rôles respectifs, et les méthodes utilisées pour leur gestion.

Nous verrons, sur un tableau récapitulatif, des données correspondant au fonctionnement de plusieurs moniteurs transactionnels différents, qui supportent des applications diverses et variées. Ces chiffres permettent de se rendre compte des difficultés qu'il y a à classifier un moniteur transactionnel suivant son type d'application. Ce qui rend impossible la définition de seuils exacts et adaptables à tous les sites. La conclusion à tirer de cette étude est que seul l'administrateur du TP est capable de donner des valeurs de seuils parfaitement adaptées au TP.

La fin du premier chapitre sera consacrée à l'étude du travail de l'ingénieur système sur DPS 8/88/90, à savoir : quand il doit intervenir et quels sont les outils dont il dispose pour contrôler un TP. Nous verrons qu'il existe peu d'outils, et particulièrement dans la version DMIV-TP, qui date de la fin des années 1970, du moniteur transactionnel. Il y a deux types d'outils : ceux qui fournissent des informations après l'arrêt du TP et ceux qui fournissent des informations en temps réel. C'est ce deuxième type auquel nous allons nous intéresser dans la cadre du développement d'un système automatisé de surveillance des TP.

De tout cela découle une question dont la réponse est à l'origine du cahier des charges : quels sont les travaux habituellement réalisés par un ingénieur système et qui sont réalisables par un outil expert automatisé.

Le **chapitre deux** sera consacré à la description des solutions conceptuelles adoptées pour la réalisation du système expert.

Tout d'abord nous allons voir la "genèse" du système expert, c'est-à-dire le passage de l'étude de faisabilité au prototype APSYST, en passant par la maquette de test de faisabilité.

Ensuite, nous verrons quels sont les choix qui ont été faits pour le logiciel, et quelles sont les limites que nous nous sommes imposées. A noter que ces choix découlent tous de l'utilisation de la maquette et de critiques qui en ont résultées.

La suite du chapitre contient une description d'APSYST du point de vue utilisateur.
Tout d'abord l'environnement matériel de développement et d'exploitation du système expert. Ensuite l'organisation générale du fonctionnement du système expert à l'aide d'organigrammes simples.

Chacun des modules présentés dans l'organigramme sera décrit d'un point de vue fonctionnel, côté utilisateur final.

Nous terminerons le chapitre deux par un exemple de session en mode expert : l'écriture d'une base de règles (la grammaire de la syntaxe des règles est donnée en annexe) puis par deux exemples de diagnostics émis par le système expert.

Le **chapitre trois** contient la description de la solution théorique retenue pour que le système expert soit compatible avec une utilisation en temps réel, à savoir la gestion et la prévention des contradictions et l'optimisation du fonctionnement du système expert, en insistant sur la partie inférence.

Pour cette solution théorique, nous commencerons par une description et une caractérisation de la base de connaissances du système expert.

Nous enchaînerons par une description de deux types de chaînage avant : le chaînage avant standard avec au départ une base de connaissances ne contenant que des faits issus de métrologie et le chaînage avant optimisé qui tient compte de l'évolution d'un contexte afin de diminuer le nombre de règles exécutées à chaque cycle. La partie théorique se termine par la démonstration que l'application des deux types de chaînages avant donne les mêmes résultats, quelles que soient les bases de faits initiales.

Le **chapitre quatre** est consacré entièrement à la description technique du logiciel.

Tout d'abord, nous verrons quels langages ont été utilisés pour quels modules.

Chacun des modules importants du système expert sera expliqué et présenté sous forme d'algorithme soit en pseudo-langage structuré, soit en pseudo-langage PROLOG. La présentation des algorithmes a pour but d'expliquer simplement les solutions techniques retenues.

Pour l'analyse syntaxique de la base de règles, la méthode utilisée est complètement détaillée. Après transformation de chaînes de caractères en listes d'éléments (analyse lexicale), l'analyse syntaxique est réalisée par unifications successives de sous-listes d'éléments de façon à ce qu'il n'y ait jamais de backtracking (on a l'équivalent d'une grammaire LL(3))

Le code interne prolog de représentation de la base de règles généré par l'analyseur syntaxique est tel qu'une condition ou une conclusion est remplacée par un seul prédicat. Les règles ainsi transformées sont représentées par un numéro d'ordre, un indice, une liste de prédicats conditions et une liste de prédicats conclusions.

Le graphe de dépendances est calculé et construit exactement comme il a été présenté dans la partie théorique : calcul des relations directes de faits à faits et de faits à règles, puis clôture transitive des relations.
Un exemple détaillé illustrera cette explication.

Nous verrons que seul le mode consultant contient des modules écrits en d'autres langages que PROLOG. Il s'agit des modules de métrologie qui sont écrits en C et des modules de journalisation et de restitution graphique qui sont écrits en PASCAL.

Les modules du mode consultant sont expliqués dans l'ordre de leur exécution.

Nous verrons en détails comment on adapte APSYST au site sur lequel on l'implante. Cette adaptation est faite en utilisant des paramètres de deux types : BOOLEENS et REELS. Les BOOLEENS correspondent à la configuration du système expert, (imprimante, pilote de connexion, restriction du champ d'action, connexion transparente, durée de temporisation, etc). Les REELS représentent des seuils critiques qui seront comparés à des faits de base ou calculés.

Pour permettre l'adaptation facile du module de connexion à tous les sites, nous avons utilisé un fichier externe au programme qui contient l'ensemble des messages échangés entre le micro et le mainframe. L'ordonnancement des messages est géré par un automate d'état finis à "n" états d'entrée et 2 états de sortie (un pour la fin de dialogue normale et un pour les problèmes de connexion). Cette méthode sera décrite complètement dans le chapitre quatre.

Nous verrons que chaque écran capté lors de la connexion contient des informations qui sont rangées différemment. Il faut soit faire des moyennes sur plusieurs écrans successifs, soit prendre des maxima, soit prendre des informations étalées sur trois ou quatre écrans consécutifs.

Comme il s'agit d'un système expert temps réel, le moteur d'inférences doit comporter des contrôles particuliers de cohérence et de consistance. Nous détaillerons ensuite le fonctionnement du moteur d'inférence et en particulier l'utilisation du graphe de dépendances. L'algorithme de saturation en chaînage avant y sera détaillé également.
L'explication du fonctionnement du moteur d'inférence sera complétée par un exemple de gestion de la base de connaissances, à partir d'une base de 4 règles et 8 faits.

Un autre module important sera détaillé dans ce chapitre, il s'agit du module de changement de paramètres qui permet d'ajuster la valeur des seuils en temps réel .

Dans la version prototype APSYST, il était possible de justifier les commentaires en temps réel, ce qui n'est plus possible dans le progiciel SEAT. En effet, justifier en temps réel interrompt la surveillance sur un ordinateur mono-tâche.

Nous avions le choix entre deux méthodes pour la justification : d'une part un chaînage arrière et d'autre part un marquage des faits calculés. La première demandait l'écriture d'un moteur d'inférences tandis que l'autre ne demandait qu'une légère modification du moteur chaînage avant, à savoir, le marquage de chaque fait par le numéro de la règle dont il est issu. C'est la deuxième méthode qui a été retenue, car elle n'agrandit presque pas la taille du code. Elle sera présentée de manière plus complète dans ce chapitre.

Nous finirons le chapitre par l'explication du module de restitution de séries sous forme graphique.

Le **chapitre cinq** a pour objet la description de la base de connaissances.

Tout d'abord nous verrons quelles sont les informations dont dispose le système expert en commentant les différents écrans tels qu'ils sont captés par le module de connexion.

La deuxième partie de ce chapitre présente les différents types d'informations contenues dans ces écrans et la méthode utilisée pour les transformer en faits. Ensuite, nous verrons comment ces faits sont utilisés par le moteur d'inférences.

La troisième partie du chapitre cinq présente une structuration de la base de connaissances. Nous verrons qu'il est possible de la structurer en sous-ensembles de faits et de règles qui satisfont des relations précises.

Le **chapitre six** explique ce qu'il a fallu faire pour que le prototype de recherche APSYST devienne le progiciel commercialisable SEAT.

Nous verrons que toutes les solutions techniques retenues restent valables, mais qu'il a fallu améliorer la convivialité du système expert. Certaines remarques relatives à l'utilisation d'APSYST seront présentées dans ce chapitre. Elles ont permis d'ajouter de nouvelles fonctionnalités qui sont exploitées dans le progiciel SEAT.

La version 1 de SEAT existe depuis Janvier 1990. La version 2 est sur le point d'être commercialisée dans le dernier trimestre 1990. Le chapitre six présente les modifications caractérisant cette évolution.

Nous terminerons par une réflexion sur l'avenir de SEAT.

I Un problème de système expert temps réel
La gestion des centres informatiques

1) Les Sociétés de Services

Le rôle des sociétés de services est de vendre de la disponibilité machine, du temps processeur et des ressources à des clients, ou alors de vendre des applications élaborées dans les bureaux d'études. Il est loin le temps des programmes sur cartes perforées et des services de saisie où les seules machines étaient des perforatrices. A l'origine de l'utilisation des gros ordinateurs, les programmes étaient exécutés en traitements par lots. Depuis le début des années 1970, un nouveau type d'activité est proposé aux clients: les services TEMPS REELS.

Depuis lors, ce type d'activité prend une part de plus en plus importante parmi l'ensemble des services informatiques, et notamment par rapport aux travaux BATCH. L'environnement des services d'exploitation des gros systèmes a considérablement changé avec cette évolution. La majorité des applications offertes par les sociétés de services sont des systèmes de gestion de bases de données (facturation, paye, comptabilité, etc). Les taux de mise à jour augmentent de plus en plus.
D'autres caractéristiques des systèmes informatiques utilisant le TEMPS REEL ont contribué à compliquer leur exploitation :

- Diversification et multiplication des systèmes de communications :
  Au départ, un client d'une société de services qui désirait utiliser des applications TEMPS REEL devait disposer chez lui d'un ou plusieurs terminaux compatibles (voire de la même marque) avec l'ordinateur central. Actuellement, cette configuration est toujours utilisée et reste la plus efficace. On trouve en plus des micro-ordinateurs (munis de cartes d'émulation) utilisés comme simples terminaux. L'évolution la plus importante est l'apparition d'une configuration plus simple, plus conviviale et moins couteuse en investissement matériel : le minitel. Mais cette simplicité et convivialité n'est ressentie que par l'utilisateur. Les applications supportant ce type de configuration sont bien plus complexes. Elles demandent une quantité de ressources plus importante pour traiter des quantités d'informations plus grandes et venant de tous les horizons.
  L'arrivée des ces terminaux "grand public" n'a fait qu'augmenter cette croissance des taux d'utilisation et de mise à jour dans les bases de données.

- Qualité des services :
  L'utilisation des services TEMPS REEL oblige à renforcer la qualité des applications. La demande des utilisateurs au "tout, tout de suite" en croissance constante implique des applications plus rapides et plus variées. Les résultats sont édités et exploités en TEMPS REEL. Les temps de réponse doivent être surveillés étroitement pour éviter tout mécontentement des clients.

- Simultanéité des accès :
  L'utilisation de la même application par plusieurs utilisateurs simultanément, avec les mêmes fichiers de bases de données pose des problèmes d'accès concurrents. Des contrôles soignés doivent être effectués à tout moment.

- Confidentialité :
  A cause de la diversification des possibilités d'accès au central, il est nécessaire de protéger les informations : ne pas autoriser n'importe qui à disposer des informations d'autrui. Le contrôle des échanges de messages est aussi important.

- Allocation de ressources :
  Pour un nombre élevé d'utilisateurs, les ressources nécessaires sont nettement supérieures aux ressources disponibles. Elles doivent donc être allouées puis libérées sans arrêt afin d'être partagées. Ces opérations doivent être effectuées de manière la plus transparente possible pour les utilisateurs.

Cette liste n'est pas exhaustive mais présente quelques caractéristiques parmi les plus importantes.

2) Le DPS 8

Regardons de près la structure d'un DPS 8, ainsi que le rôle des éléments qui le composent, suivant les types d'application.

Le DPS 8 est un grand ordinateur sur lequel tournent essentiellement 2 systèmes d'exploitation :
- MULTICS : pour les applications à caractère scientifique
- GCOS : pour les applications de gestion

Le cas qui nous intéresse est le DPS 8 sous le système GCOS.

Dans ce système d'exploitation, on distingue 3 types d'environnement principaux [BUL 86] (voir figure I.1).

- Le T.S.S. (Time Sharing System) : C'est un système interactif utilisé pour la préparation et la programmation d'applications. C'est le système utilisé pour les développements.

- Le BATCH : A l'opposé du T.S.S., cet environnement n'est pas interactif, il ne sert qu'à l'exécution des programmes en traitement par lots.

-Le T.P. (Transaction Processor) : C'est un système interactif spécialisé dans la gestion d'applications. Celui-ci permet d'utiliser des applications avec un maximum d'efficacité et de sécurité. C'est le seul environnement auquel ont accès les clients.

Le T.S.S. permet de programmer des modules et de créer des exécutables destinés au T.P. et de préparer les fichiers base de données utilisés par ces modules.

Sur le DPS 8 peuvent fonctionner plusieurs activités du même type. Le nombre n'est pas limité, si ce n'est par la place mémoire et les ressources disponibles (voir figure I.2).

Dans l'exécutif du T.P., toutes les ressources sont confiées à des "MANAGERS". Ce sont des programmes systèmes qui gèrent leurs allocations :

- DATA BASE MANAGER
- MESSAGE MANAGER
- TRANSACTION MANAGER

Tous les périphériques sont gérés par le PALC (peripheral allocator).

Même si plusieurs travaux résident simultanément en mémoire et sont considérés comme actifs en même temps, le parallélisme physique est très limité. En effet les DPS 8 sont au maximum quadri-processeur.

DPS 8 SOUS GCOS

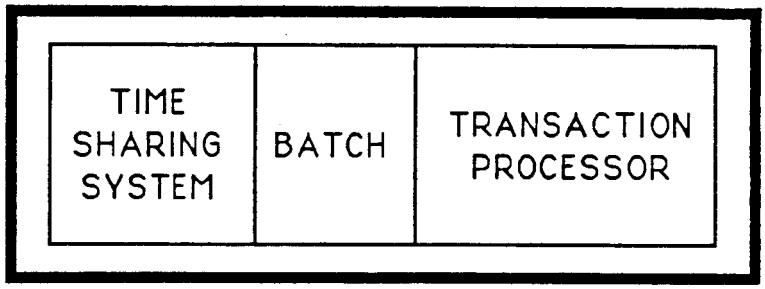


FIGURE I.1a
Environnements sous GCOS

LE SYSTEME TRANSACTIONNEL

EXECUTIF		
TPR1	TPR2	TPR3
MB1	MB2	MB3
DBB1	DBB2	DBB3
MC1	MC2	MC3
. . .	. . .	. . .

FIGURE I.1b
Le moniteur transactionnel

TPRi = zone mémoire réservée pour les TPR dans le couloir i
MBi = message buffers alloués au couloir i
DBBi = data base buffers alloués ou couloir i
MCi = zone mémoire réservée pour le couloir i

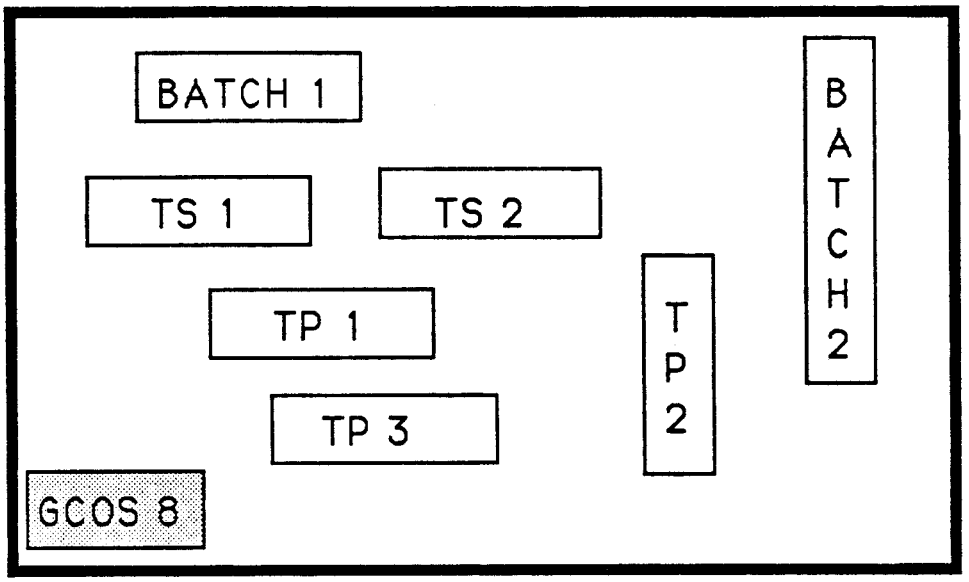


FIGURE I.2
Applications sous GCOS

BATCH i représente un environnement batch
TP i représente un environnement TP
TS i représente un environnement TSS
GCOS 8 représente le système d'exploitation

Les jobs appartiennent à des classes qui correspondent à des priorités :

- Classe A : c'est la classe par défaut, la classe la moins prioritaire. Chaque activité de classe A dispose d'un temps d'occupation de la ressource processeur et le système GCOS reprend la main à chaque opération d'Entrée/Sortie.

- Classe B : le temps n'est pas limité, mais le système d'exploitation reprend la main à chaque opération d'Entrée/Sortie. A noter que cette classe est utilisée pour la plupart des applications, et particulièrement pour le transaction processor.

- Classe C : la classe la plus prioritaire. GCOS ne reprend pas la main, c'est l'activité de classe C qui la lui rend. La priorité de cette classe sur le système la rend dangereuse à utiliser. Son utilisation est très rare.

L'environnement qui nous intéresse pour le système APSYST est le TRANSACTION PROCESSOR.

2.1) Le MESSAGE MANAGER

Les accès au TP sont gérés de deux manières différentes :

1 - Le Message Manager se trouve sur le DPS 8, et c'est lui qui gère les communications (voir figure I.3). Ce mode de gestion des messages s'appelle le "DACQ".

2 - Les "message buffers" se trouvent dans le fontal (ou Datanet), et c'est là que sont gérés tous les messages. (voir figure I.4). Ce mode de gestion des messages s'appelle le "DACQ QUEUED".

Comment s'effectue une connexion au TP ?

L'ordre $*$CN NOM-TP correspond à la demande de connexion au TP de nom NOM-TP.

Ensuite, le DPS 8 demande le "LOGICAL-IDENTIFIER" appelé LID, qui représente le nom logique de l'utilisateur.qui lui permettra d'utiliser le TP (en quelque sorte le mot de passe).

Il faut noter que chaque LID ne peut être utilisée que par un utilisateur à la fois. Une fois la LID reconnue par le central, l'utilisateur doit demander une transaction. Chacune d'elle est appelée par un message spécifique : le MESSAGE IDENTIFIER.(MSG-ID ou MID).

Le MID est un message initiateur de transaction. Une transaction est une suite de modules enchaînés, appelés des TRANSACTION PROCESSOR ROUTINE (ou TPR) entrecoupés par des dialogues homme/machine.

Toutes les TPR utilisables dans un TP sont contenues dans un fichier librairie. Elles sont chargées en mémoire dans une zone réservée quand des transactions les demandent. Certaines peuvent être déclarées résidantes et donc rester constamment en mémoire, limitant ainsi la place restante pour les autres TPR, mais diminuant les accès physiques à la librairie.

FIGURE I.3
DACQ

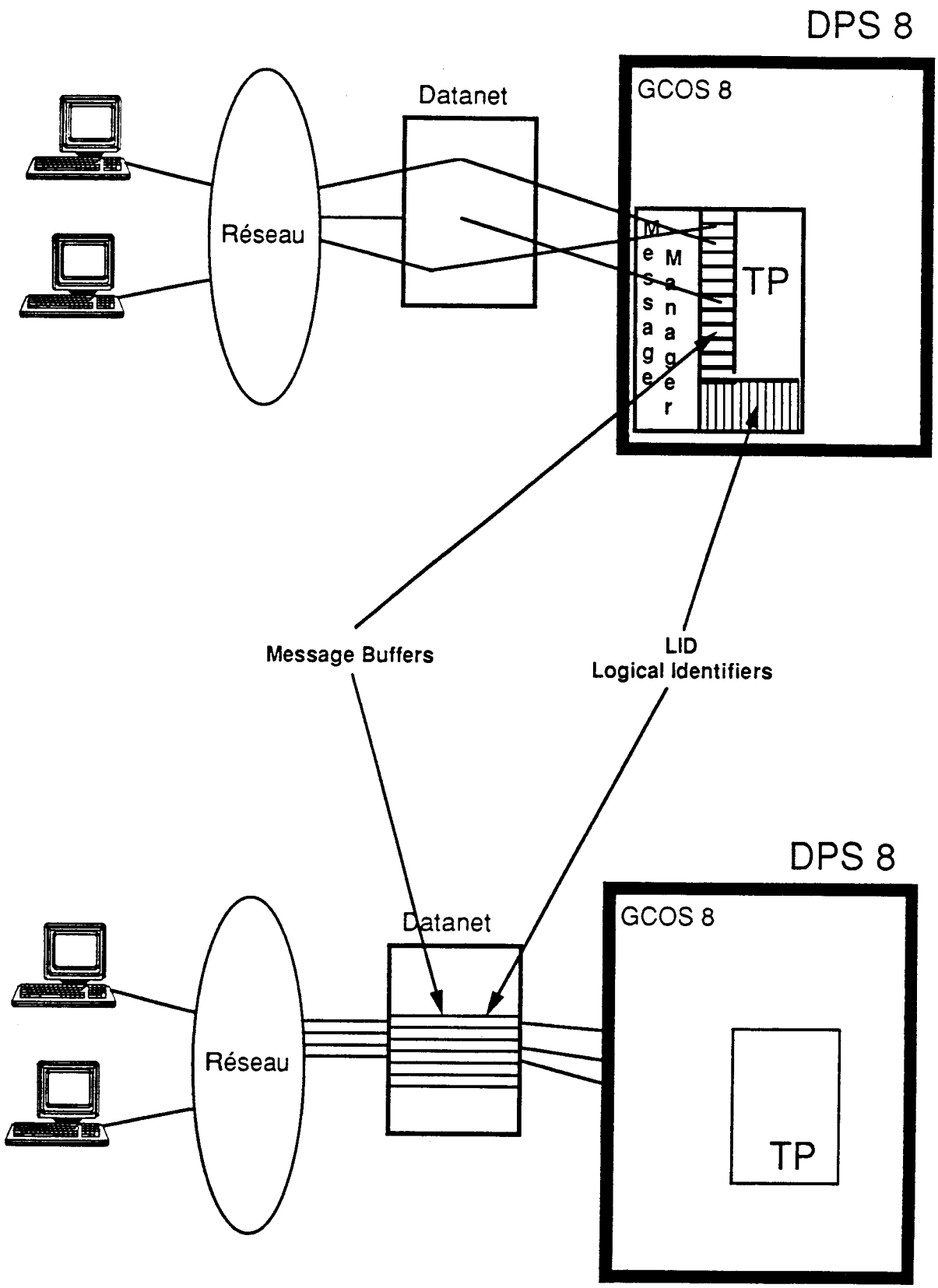


FIGURE I.4
DACQ QUEUED

La figure 5 explique les messages échangés entre l'utilisateur, le Datanet et le DPS 8, ainsi que la zone mémoire réservée pour les TPR.

1 - L'utilisateur demande la connexion au TP de nom TP1.
Le message commence par "$*$". Cela signifie que la requête est adressée au Datanet.

2- Le message Manager a reçu la demande de connexion et demande à l'utilisateur son Logical-Identifier. La connexion est bloquée au niveau Datanet.

3- L'utilisateur répond par un nom qui lui a été donné par l'administrateur TP ou par le responsable d'exploitation du TP (par exemple LOGIC).

4- Le Logical-Id est valide, la connexion est effectivement réalisée entre le TP du DPS 8 et l'utilisateur.

5- L'utilisateur demande la transaction TX1 en envoyant son Message Identifier (ou Msg-Id ou MID). Le Message Manager cherche la première Transaction Processor Routine (ou TPR) correspondante à ce Msg-Id. Ensuite, il la charge en mémoire, puis commande son exécution.

6- La TPR "TPR1" se termine par l'affichage d'une question, et attend la réponse de l'utilisateur. Pendant l'attente de la réponse, la zone mémoire est libérée pour être éventuellement occupée par d'autres TPR.
Par "question", il faut comprendre soit une vraie question, soit l'affichage d'un masque de saisie vierge, auquel cas la réponse correspond au remplissage des zones.
Avant de libérer la zone mémoire, la TPR indique au TP quelle sera la prochaine TPR à utiliser (et éventuellement à charger) après la réponse. A noter qu'il peut s'agir de la même TPR ou d'une autre.

7- Réponse de l'utilisateur.

8- Demande de déconnexion. C'est le Message Manager qui se charge de lancer une TPR particulière de fin de session. (TP-DIS).

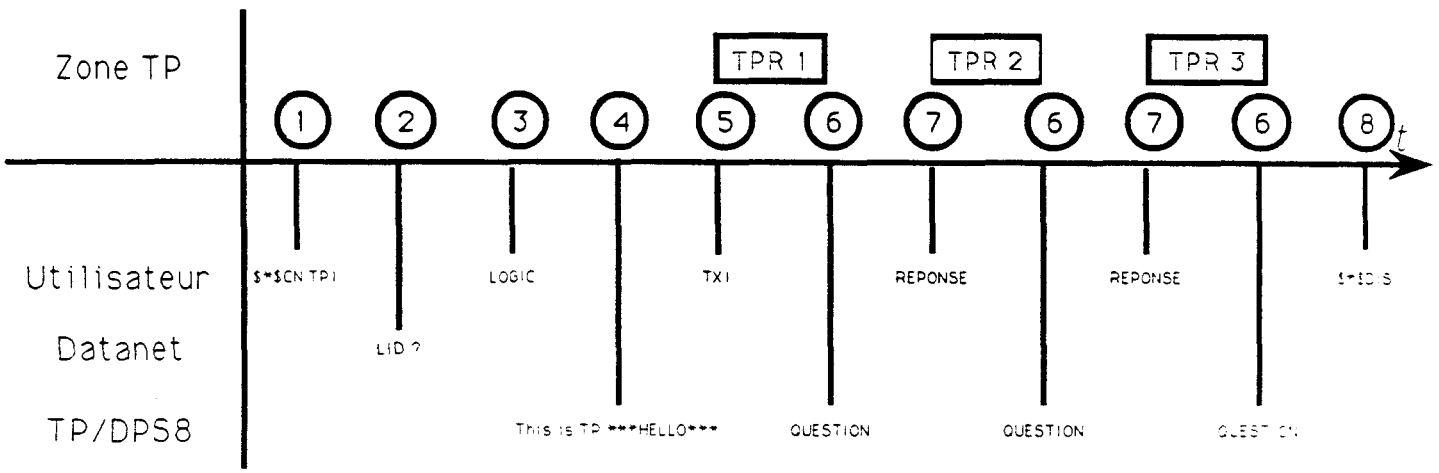


FIGURE I.5
Une session utilisateur

Le Message Manager gère tous les messages échangés.

Ils sont de 3 types :
- Logical Identifier (LID)
- Message Identifier (Msg-Id ou MID)
- Questions / Réponses

En cas de connexion au TP à partir d'un minitel, les LID ne sont pas donnés par l'utilisateur. Ils sont attribués par le Datanet dynamiquement en incrémentant un compteur. Dès qu'un minitel est déconnecté, son numéro se libère et sera donné au prochain qui se connectera.

2.2) Le DATA BASE MANAGER

Tout TP sur le DPS 8 travaille avec une base de données (IDS II le plus souvent). La qualité première de la gestion d'une base de données par un TP est d'assurer à tout moment l'intégrité des informations qui s'y trouvent. Quel que soit l'incident qui pourrait survenir durant l'exécution du TP (par exemple : coupure de réseau, parasites, erreurs, disques défectueux, ...) le TP assure la reprise du travail de tous les gens connectés avant le problème de manière transparente, ainsi que l'intégrité de la base de données. Pour cela, la gestion de cette base de données est assurée par le Data Base Manager. Toute requête effectuée par le TP à la base de données est adressée au Data Base Manager.

Dans la base de données, les informations sont stockées dans des pages dont la taille est définie lors de sa création: les DATA BASE BUFFERS. Dans le TP une zone destinée à contenir les D.B.B. est réservée A chaque consultation ou modification d'information, le D.B.Manager va chercher la page correspondante sur fichier pour la transférer dans un D.B.Buffer (si elle n'y est pas déjà).

Pour assurer l'intégrité de la base, 2 opérations sont effectuées:
- Avant l'écriture sur la page, le D.B.M. écrit sur le journal du TP l'image de la page : l'IMAGE BEFORE.
- Après chaque écriture sur la page, le D.B.M. écrit sur le fichier journal du TP : l'IMAGE AFTER.

D'autre part, le TP écrit sur son journal des points de reprise (ou CHECKPOINT) à chaque conversation ou sur demande explicite dans les programmes. Ceux-ci permettent, avec les IMAGES BEFORE et les IMAGES AFTER, d'assurer l'intégrité de la base

Les allocations de pages sont gérées par des tables dans lesquelles sont associés les "Logical Identifier" et les numéros de pages accédées. Le système détecte les conflits d'accès par consultation de ces tables.

De la même manière, il détecte les étreintes fatales.

Ex : La LID A1 accède les pages M et P.
La LID A2 accède les pages Q, R et S.

A1	M
A1	P
A2	Q
A2	R
A2	S
...	...

Si A1 demande l'accès à la page R, il y a attente pour A1
Si A2 demande l'accès à M, il y a étreinte fatale
Dans ce cas, le système "aborte" la transaction demandée par A2. Ensuite, il reprend les "Before Images" en remontant jusqu'au dernier point de reprise, puis il termine la transaction de A1 et seulement après, il relance la transaction de A2.

2.3) Le TX MANAGER

C'est lui qui assure la multi-programmation à l'intérieur du TP.
Le taux de multi-programmation d'un TP n'est pas dynamique, donc pas extensible. Il est fixé lors de la génération du TP (appelé SYSGEN sur DPS 8). Ce taux est aussi appelé nombre de couloirs ou "Normal Load" dans le SYSGEN. Sa valeur varie de 1 à 36.

Dans le SYSGEN, toutes les caractéristiques du système TP sont définies : les tailles des différents modules, les fichiers alloués, les programmes disponibles, etc. Pour chaque transaction on donne un nombre minimal de Data Base Buffers, de Message Buffers, de Page Reservation, ainsi que le nom de la première TPR qui doit être exécutée pour la transaction. De plus, on donne les modes d'accès à certains fichiers (lecture concurrente, écriture unique, etc...) et le niveau de priorité de la transaction par rapport aux autres.

Le rôle du Transaction Manager commence quand l'utilisateur envoie un Message-Id (voir fig I.5 n°5).

Son premier travail pour un utilisateur qui vient de donner son Message-Id est de lui réserver ce dont il a besoin pour exécuter son travail :

- Data Base Buffers
- Message Buffers
- Table de gestion du Concurrent Access Control
- Espace mémoire pour les TPR
- Un couloir d'exécution
- Une TPR

Pendant une session, le TX manager gère un contexte par utilisateur. Celui-ci est constitué de la TP-Storage et de la TX-Storage.
La TP-Storage est la même pour toutes les transaction, et permet essentiellement l'enchaînement des TPR à l'intérieur d'une transaction. Sa taille est d'environ 300 caractères. La TX-Storage est la zone de communication de TPR à TPR dans la même transaction. Si deux TPR s'enchaînent, elles auront la même TX-Storage.

3) Le comportement d'un TP suivant le type d'application.

Voici un tableau qui contient des informations sur quelques TP qui tournent sur le DPS 8 de la Société CdFi-NATEL.
Toutes les données de ce tableau sont extraites des rapports d'exécution de ces TP (deux sessions correspondent au même T.P., deux jours différents et non consécutifs)

Nom des TP	TP1	TP1	TP2	TP3	TP4
Informations					
Temps écoulé	21 h	105 h	16 h	4,5 h	10,5 h
Nb de terminaux	21	28	127	14	277
Temps Processeur	0,43 h	2,64 h	2,69 h	0,07 h	2,65 h
Temps IO	1,24 h	3,29 h	8,43 h	0,27 h	6,38 h
Nb bases	28	28	37	18	39
Nb TPR exécutées	26779	355512	337082	1715	85540
Nb TPR chargées	3989	74701	12496	380	12870
Nb TX existantes	23	23	91	88	39
Nb TPR existantes	160	148	540	281	540
Nb d'accès bases	132206	661242	627534	54434	1127981
Page Res. Space	1024	1025	505	501	2403
Data Base Buffers	6	10	35	20	25
Message Buffers	16	49	65	29	163
Sleep Table entry	24	34	34	30	30

TP1 et TP2 sont des applications où les utilisateurs se connectent à partir du minitel. Par contre, pour les autres, les utilisateurs disposent de terminaux BULL Questar (ou de micro-ordinateurs munis de cartes d'émulation) reliés par lignes spécialisées ou par le réseau Transpac.

Caractéristiques des TP présentés dans le tableau.

TP1 : Banque de données
Messagerie
TP2 : Serveur minitel
Gestion de notes d'examens
Fédérations Sportives
TP3 : Comptabilité
TP4 : Facturation de l'eau
Paye
Gestion de stocks

Ces chiffres et les caractéristiques des différents TP montrent les difficultés de gestion des ressources des TP. On n'y retrouve aucune relation de dépendance. Il est donc très difficile de trouver, pour un TP, des normes d'utilisation en fonction des types d'application, et donc de définir des règles d'allocation des ressources.
Pour réaliser un bon réglage de TP, il est en plus nécessaire de connaître le travail de chaque TPR, leurs enchaînements, et leur consommation. Nous reviendrons plus en détail sur ces considérations dans la partie II.4 et au chapitre IV consacré à la base de connaissances d'APSYST.

4) Le travail de l'ingénieur système

Sur tous les sites, il est faux remarquer que ce sont toujours les utilisateurs les premiers au courant quand il y a un problème, et les ingénieurs système les dernières personnes averties.

Presque à chaque fois qu'un incident survient, le même scénario se produit :

1 - L'utilisateur est mécontent des temps de réponse
2 - L'utilisateur téléphone au centre de calcul pour exprimer son mécontentement.
3 - L'ingénieur système ou l'administrateur TP entre en scène pour découvrir l'origine de cette dégradation et intervient dans le but d'y remédier.

Il faut cependant apporter une précision : l'appréciation de l'utilisateur est subjective et dépend de beaucoup de facteurs. Pour certains, un temps de réponse de 5 secondes n'est pas très bon, et pour d'autres, il est catastrophique à partir de 2 secondes. Il faut aussi tenir compte de la durée du phénomène. Un temps de réponse de 10 secondes à un moment précis de la journée, alors qu'il est de 1 seconde en moyenne sur la journée, n'est pas à prendre en compte.

Il y a quelques années, seuls deux outils existaient pour l'ingénieur système :

Le rapport TP : rapport d'exécution des TP donnant une vue globale de la journée de l'activité TP. (Consommation et allocation de ressources, nombre d'accès base de données, conflits d'accès, étreintes fatales, taux de journalisation, ...) Ce rapport représente un listing de plus de 100 pages lourd à manipuler. Le rapport TP ne permet pas de suivre le déroulement des TP.

Les seules information qu'il fournit sont de la forme :

- La transaction " T " a provoqué "n" aborts
- Il y a eu 50 conflits sur le ficher FF
- Le TP a été 3 fois à cours de ressources page base de données
- Au maximum 50% des messages buffers ont été simultanément utilisés durant la journée.

Il est facile de voir qu'il n'y a que 2 types d'information :

- Valeur minimale ou maximale d'un compteur.
- Nombre total cumulé sur la journée.

Les journaux TP (cf I.2.2) : Ils contiennent la copie de toutes les opérations effectuées par le TP au cours de la journée (plusieurs millions d'informations). L'étude des journaux permet de savoir quelle TPR et quelle transaction a créé des problèmes. La difficulté de leur utilisation et surtout la lourdeur de leur manipulation (parfois plus de 20 bandes par jour) dissuadent souvent les ingénieurs système d'avoir recours à eux. Ils représentent toujours la dernière solution.

Ces outils vont permettre de régler le fonctionnement du TP après une vacation complète. Leur utilisation est complexe et peu pratique. De toute façon ils ne permettent pas de voir les problèmes en temps réel.

Pour faciliter la surveillance de la machine et des T.P., le constructeur a développé et diffusé des outils temps réel : **VIDEO** pour le suivi du système GCOS 8 et **TPMON** pour le suivi des T.P..

Ces outils se présentent sous forme d'écrans rafraîchis régulièrement de manière continue. La fréquence de rafraîchissement est modifiable par l'utilisateur à tout moment, et peut aller jusqu'à un écran par seconde. Chaque écran contient un cinquantaine de valeurs. L'avantage de disposer d'informations en temps réel n'est pas négligeable. Mais la quantité de chiffres qui défilent sur l'écran et leur caractère fugace rendent leur utilisation pénible à la longue. Un expert habitué à manipuler ce genre d'outils va en un clin d'œil relever les quelques informations les plus importantes. A la seconde suivante, sur l'écran enchaîné au premier, il va pouvoir faire la même chose tout en retenant les valeurs du premier. Mais au bout de quelques écrans, mémorisation deviendra de plus en plus difficile puis impossible.

Il est inacceptable que l'ingénieur système d'un site reste plusieurs minutes devant ces écrans qui changent toutes les secondes.

En réalité, il y a deux moments particuliers où l'ingénieur système consulte les moniteurs temps réels:

- **En cours de journée**, pour un "contrôle de routine". Il regarde les quelques valeurs qui lui paraissent les plus significatives. Ce rapide coup d'œil lui permet d'évaluer l'importance de l'activité du TP, les besoins en ressources fichiers ou mémoire, les attentes et les allocations de périphériques. Il se connecte alors au TP sur une application particulière pour mesurer le temps de réponse tel qu'il est perçu par l'utilisateur.

- **Suite à des appels de clients** insatisfaits du temps de réponse des applications. Dans ce cas, en plus des quelques valeurs de routines, il observe les consommations de ressources des applications, les conflits d'accès aux fichiers, les programmes susceptibles de boucler, des applications gourmandes en ressources diverses.
Il lui est possible dans ce cas de diagnostiquer presque en temps réel un dysfonctionnement du TP. Le décalage est dû au temps qu'a mis l'utilisateur avant d'appeler le centre de calcul. Aussi petit soit-il (une ou quelques minutes), le temps écoulé entre le dysfonctionnement et l'intervention de l'expert peut suffire pour que tout soit revenu dans l'ordre. Dans ce cas, l'expert ne peut rien voir.

L'inconvénient, est que dans le premier cas, l'expert ne voit rien car tout fonctionne bien, et dans la seconde situation, il arrive trop tard, car le phénomène de dégradation des temps de réponse peut avoir une durée de vie relativement courte.

5) Synthèse

La présentation de l'environnement DPS 8 montre sa complexité et la difficulté pour un ingénieur système de suivre en temps réel le bon fonctionnement des TP, tout en assurant les travaux quotidiens de suivi d'évolution des produits, des tests de nouveaux environnements, d'optimisation des progiciels, etc.

Il est difficile d'avoir une action de surveillance préventive, de voir la dégradation du fonctionnement d'un TP avant que les utilisateurs ne s'en rendent compte. De plus rester devant un écran sur lequel défilent plusieurs dizaines de valeurs à longueur de journée (ou même seulement pendant quelques minutes) n'est pas très passionnant.

Il est facile de voir l'intérêt d'un outil qui serait en permanence en train de scruter les informations données par un mainframe DPS 8. Couplé à un système expert qui traiterait toutes ces données, l'outil serait une aide considérable pour l'ingénieur système. Il lui enlèverait la partie la plus laborieuse et la moins intéressante de son travail.

Le sujet de ce travail est justement la conception et la réalisation d'un tel outil.

II La solution retenue et description du système

1) Historique

Le projet APSYST a débuté en Juillet 1986 par une étude de faisabilité. L'étude consistait en la réalisation, sur un micro-ordinateur de type PC, avec un générateur standard d'un prototype de système expert : "Intelligence Service" de GSI-TECSI qui utilise les chaînages avant, arrière et mixte. Ce prototype, en cas de réponse positive devait servir de base d'un projet de réalisation d'un produit d'assistance aux Ingénieurs Système de DPS 8.

Le prototype était constitué de 3 parties :

- Une base (d'environ 150 règles sans variable) qui opérait sur environ 250 faits.

- Un module d'entrée des données qui allaient constituer la base de faits initiale. Ce module devait être exécuté avant l'utilisation du système expert. Ce programme était réalisé en PASCAL et permettait de remplir des tableaux de valeurs. Celles-ci étaient sélectionnées à partir du rapport d'exécution des TP par l'ingénieur système. Ensuite, le programme construisait plusieurs fichiers d'entrée pour le système expert.

- Un module d'affichage des commentaires qui était activé par l'exécution de certaines règles.

Suite à cette étude, il fût conclu qu'un tel système expert était tout à fait réalisable et les perspectives de résultats étaient tout à fait encourageantes quant à l'aide qu'il peut apporter aux ingénieurs système DPS8.

Cependant, certains choix qui ont été faits pour le prototype étaient à revoir dans la perspective de la réalisation d'un produit performant et utile :

- Entrer les données manuellement à partir d'un rapport TP représente un travail laborieux. De plus, le dépouillement des informations qu'il contient dans le but d'en extraire des faits pour le S.E. suffit à l'ingénieur système pour savoir comment a fonctionné le TP. Il est donc évident que le système expert, si

l'on veut qu'il rende des services, et surtout qu'il soit utilisé, doit obligatoirement travailler en connexion directe avec le DPS 8.

- L'utilisation des informations contenues dans la rapport TP ne suffit pas à le surveiller. En effet, celui-ci ne donne qu'une vision consolidée de son fonctionnement en fin de session. Il ne permet pas de connaître l'évolution de la charge TP durant la journée. Celle-ci est très variable et dépend de beaucoup de paramètres incontrôlables mais aussi de l'heure (voir figure II.1). Notamment, il ne permet pas de détecter des pointes de charges qui suffiraient à détériorer son fonctionnement. Il est donc nécessaire de travailler avec les outils de surveillance Temps Réel du TP.

2) Les contraintes d'APSYST

Les spécifications du système APSYST sont articulées autour de quelques éléments précis :

TEMPS REEL
Le système APSYST agit sur des données captées en temps réel. Ces données sont exactement celles dont dispose un ingénieur système pour faire du contrôle manuel de TP. Elles sont fournies par les outils standard qu'offre le constructeur (voir chap. I).

AUTOMATISME
Les phases de métrologie et de captage des informations qui vont constituer la base de faits initiale sont automatisées. Le micro-ordinateur qui contient le système expert est munie d'un carte d'émulation de terminal standard de DPS 8. La connexion du micro-ordinateur aux outils de surveillance du TP et du système GCOS 8 sans aucune intervention humaine, fonctionne. Les faits sont captés, stockés, ordonnés et triés de manière transparente.

PARAMETRISATION
Le système expert est paramétré autant que possible, afin de permettre une adaptation facile au spécificités du site sur lequel il est greffé. La base de connaissances est paramétrée ainsi que la connexion. Le passage d'un site à l'autre ne

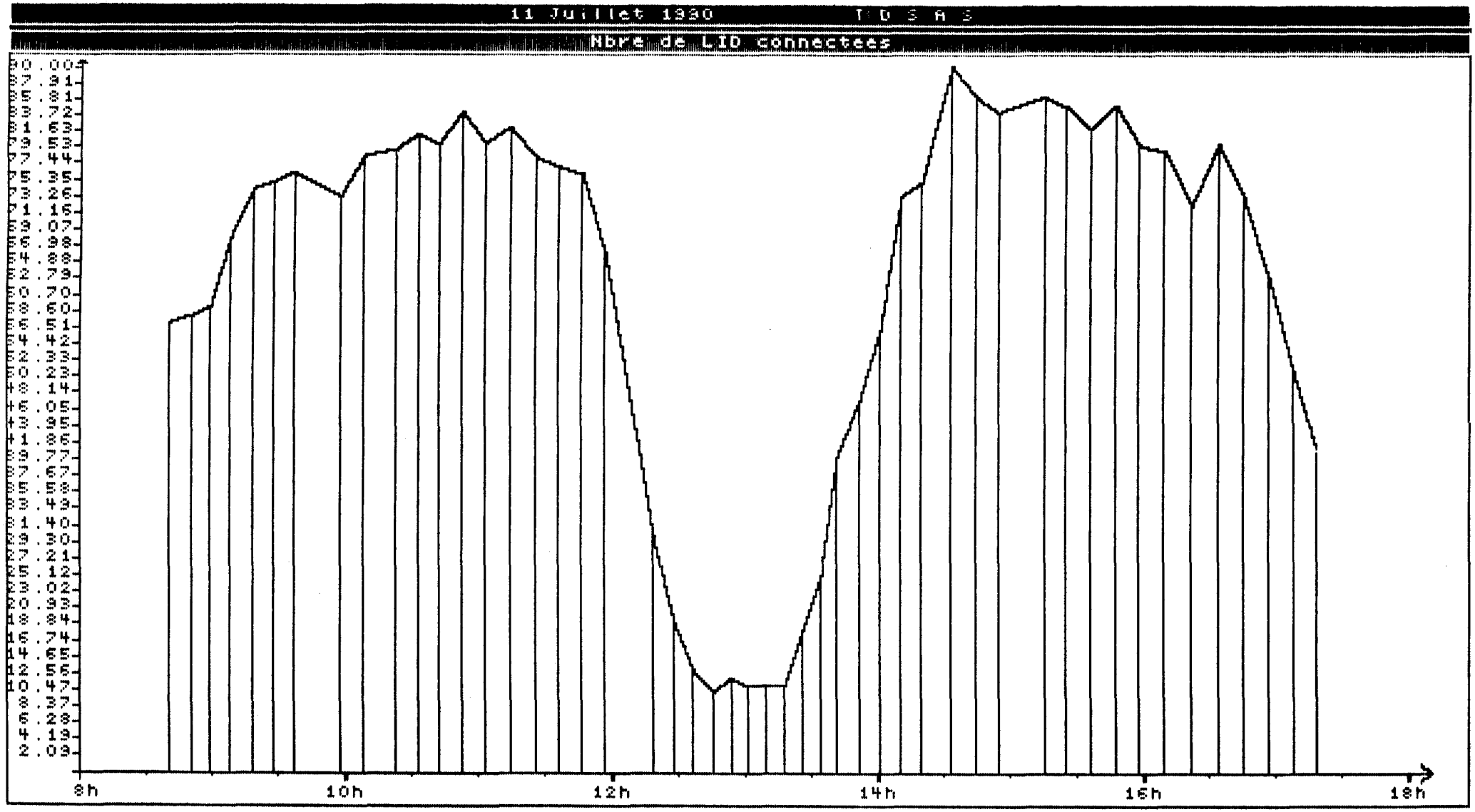


FIGURE II.1
Variations horaires de charge d'un TP
(nombre de personnes connectées)

demandera qu'un minimum de modifications, à savoir les différentes normalités des TP et les messages de connexion.

SYSTEME EXPERT EXTERNE AU CENTRAL
Ce choix s'explique facilement. Il n'est pas concevable qu'un système de contrôle et de surveillance de TP soit sur le DPS 8. Il apporterait une charge supplémentaire au TP, au moment où il serait le plus utile (surcharge ou dysfonctionnement). De plus si le TP se bloque ou s'arrête anormalement, le fait d'être externe permet à APSYST de s'en apercevoir.

DEVELOPPEMENT D'UN NOUVEAU P.G.S.E.
Les fonctionnalités des différents programmes générateurs de système expert disponibles sur le marché ne répondaient pas aux besoins du temps réel.

SYSTEME EXPERT SUR MICRO-ORDINATEUR
Le système expert est installé sur un micro-ordinateur de type PC. Les avantages sont évidents:

- Facilité de développement
- Facilité d'installation
- Connexion aisée
- Possibilité de plus large diffusion
- Potentiel de généralisation des développements à d'autres problèmes du même type
- ...

3) Etude de quelques S.E. temps réel

Un certain nombre de systèmes experts temps réel sont des logiciels d'aide au contrôle de processus. Ils ont des caractéristiques et des contraintes de fonctionnement bien précises dont les plus importantes sont:

- rapidité d'exécution
- prise de décision immédiate
- métrologie par capteurs
- bases de connaissances de très grande taille
- utilisation de machines spécialisées pour l'inférence
- utilisation de machines spécialisées pour la métrologie
- inférence en chaînage avant, arrière et mixte
- utilisation de Langages Orientés Objets pour la représentation des connaissances

Ces caractéristiques ne sont pas toutes dans tous les systèmes experts temps réel, mais on retrouve la plupart dans les applications de ce type.

Le système expert ESCORT (Expert System for Complex Operations in Real Time) [SAC 86] a été réalisé pour aider à surveiller tous les processus actifs d'une usine. Les personnes chargées de cette surveillance sont dans une salle de contrôle centralisé. L'intérêt du système expert ESCORT est triple :

- reduire le nombre d'interventions humaines en utilisant un système de surveillance automatisé.

- disposer d'informations constamment mises à jour en temps réel pour permettre une efficacité optimale des opérateurs.

- réduire les risques de dégats importants dans l'usine en prenant l'initiative de désactiver certains processus dont les caractéristiques sont en dehors des normes d'utilisation.

Le système expert ESCORT est implémenté sur une station de travail LISP Xerox 1108.

D'un point de vue quantitatif, ESCORT opère sur une base de faits qui évolue à raison de plusieurs centaines de faits par minutes.

ESCORT a été développé pour effectuer le contrôle centralisé d'une usine parce qu'un humain ne peut le faire efficacement à cause de la complexité du système à surveiller.

Le système expert PICON (Process Intelligent CONtrol) [MOO 84] [MOO 85], est un système général de contrôle des processus industriels.

Son principe de fonctionnement est :

- captage d'informations
- reconnaissance de l'état du processus
- sélection d'une base de règles
- inférence et génération de diagnostics, d'alarmes et de conseils de contrôles
- modification du module de captage d'informations

Le système PICON est implémenté sur deux machines : un processeur LISP pour l'inférence et un processeur 68010 pour le captage d'informations. Les deux processeurs sont interconnectés et dans certains cas, les résultats fournis par l'inférence ont pour effet de focaliser les prises de mesures.

NEMO est un outil de développement de systèmes experts temps réel [NEM 87]. Les domaines d'application concernés par NEMO sont très variés : contrôle d'opérations en mer sur plate-formes offshore, centrales nucléaires, sidérurgie, chimie, raffinage, etc. Des applications ont été développées sur HP 9000/320, VAX, Cartes industrielles VME/Multibus, Bull SPS7.

Le système NEMO est aussi basé sur l'interaction entre le module d'inférence et le module de captage d'informations. Et comme pour le système PICON, les parties conclusions peuvent demander des informations supplémentaires et ainsi modifier le module de captage d'informations.

Le système expert HORSES (Human - Orbital Refueling System - Expert System) [BOY 86] a été développé pour aider les astronautes pendant les opérations spatiales. HORSE est basé, lui aussi, sur le principe de reconnaissance de situation, mais son rôle n'est pas uniquement de contrôler et de donner des diagnostics concernant des dysfonctionnements, il est aussi de guider les astronautes durant leurs interventions.

Jens RASMUSSEN [RAS 87] présente une architecture de base de connaissances propre au contrôle de processus industriels mais il se base sur une préoccupation qui est inverse de la nôtre : il s'intéresse aux problèmes rares à cause de leur complexité - nous nous intéressons aux problèmes les plus fréquents et qui demandent une surveillance continue, et qui sont peu intéressants pour les experts.

4) Environnement matériel

L'architecture du système expert APSYST est basée sur la connexion d'un micro-ordinateur de type PC AT et d'un DPS 8. (voir figure II.2)

Dans cette configuration, le micro-ordinateur se comporte comme un terminal standard du DPS 8. Grâce à une carte d'émulation de terminal D.K.U. (Display Keyboard Unit) de type 7007 ou 7107 , il est branché sur un contrôleur de grappe T.C.U. (Terminal Control Unit). Il n'a pas de priorité d'accès particulière. Il est muni d'une imprimante.

Grâce à cette configuration, le micro-ordinateur a la possibilité de surveiller des TP fonctionnant sur des DPS 8 distants par l'intermédiaire du réseau sur lequel sont reliés les ordinateurs frontaux (Front Network Processor). (voir figure II.3)

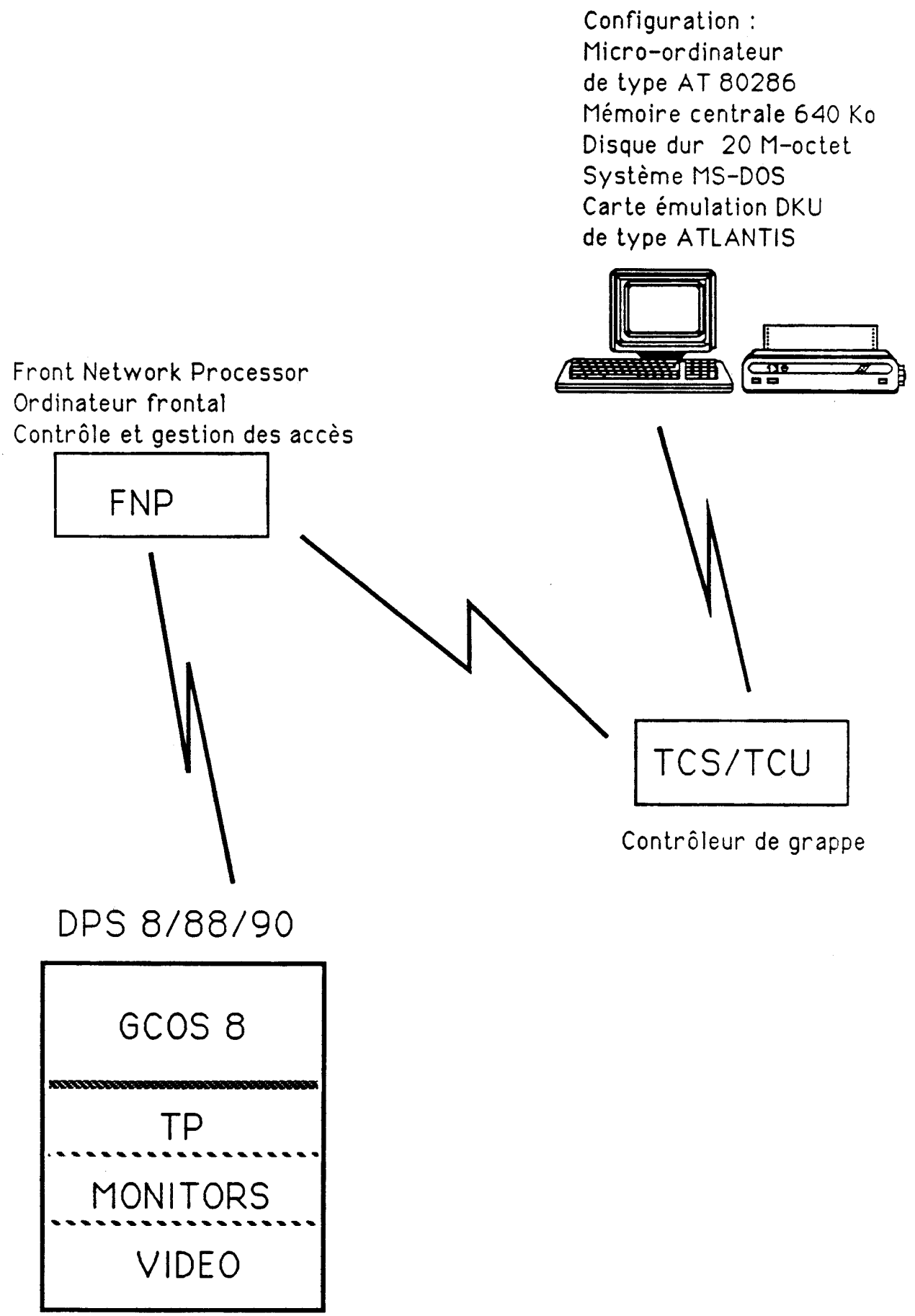


FIGURE II.2
Environnement matériel

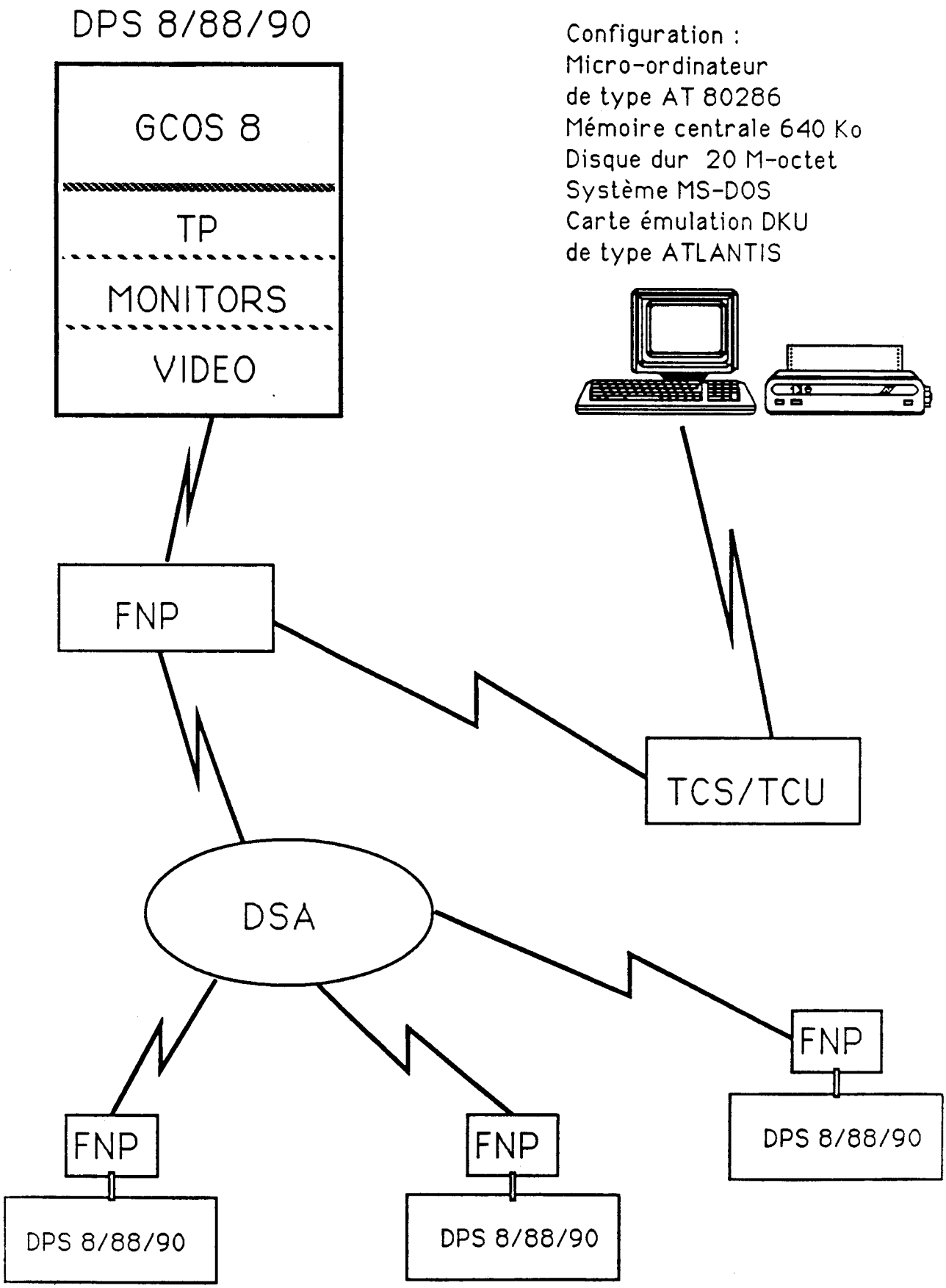


FIGURE II.3
Environnement Matériel
Plusieurs DPS en Réseau DSA

5) Environnement logiciel

Tous les modules qui constituent APSYST, qu'ils soient écrits en PROLOG, PASCAL ou C, sont des fichiers exécutables de MS-DOS.

La figure II.4 montre les différents modules du mode expert : ceux qui permettent l'élaboration de la base de règles.

La figure II.5 montre l'enchaînement des modules du mode consultant : l'utilisation du système expert.

La figure II.6 représente une hardcopy du menu général d'APSYST

6) Les différents modules

6.1) Modules d'élaboration de la base de connaissances

Analyse syntaxique: (fig. II.7)

Ce module permet l'écriture de la base de connaissances :

La base de règles :
Deux possibilités sont données :

1 - créer une nouvelle base de règles
2 - modifier une base existante.

Les règles sont affichées dans un éditeur pleine page qui dispose des fonctionnalités de l'éditeur Turbo-Prolog (copie, déplacement, suppression de blocs et travail sur des blocs externes).
L'analyse syntaxique est réalisée en dehors de l'éditeur sur la totalité de la base de règles. Lors de l'analyse, les faits utilisés sont déclarés au fur et à mesure qu'ils sont découverts. Les règles sont transformées en notation interne (cf III.3.1). Il est possible de visualiser les règles sans les modifier et de les imprimer. Il en va de même pour les faits.

Les commentaires :
Ils seront affichés lors de la phase d'inférence. Chaque commentaire est composé d'un texte générique et d'une suite de noms de faits ou d'indices de variables qui y seront inclus lorsque le commentaire sera généré.

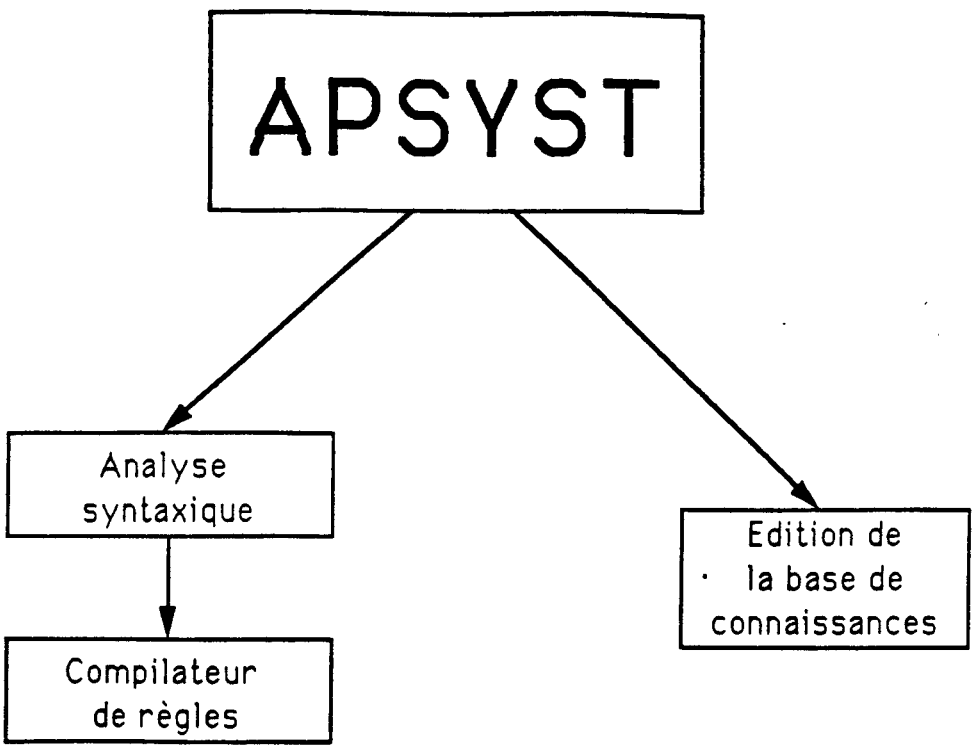


FIGURE II.4
Le mode EXPERT

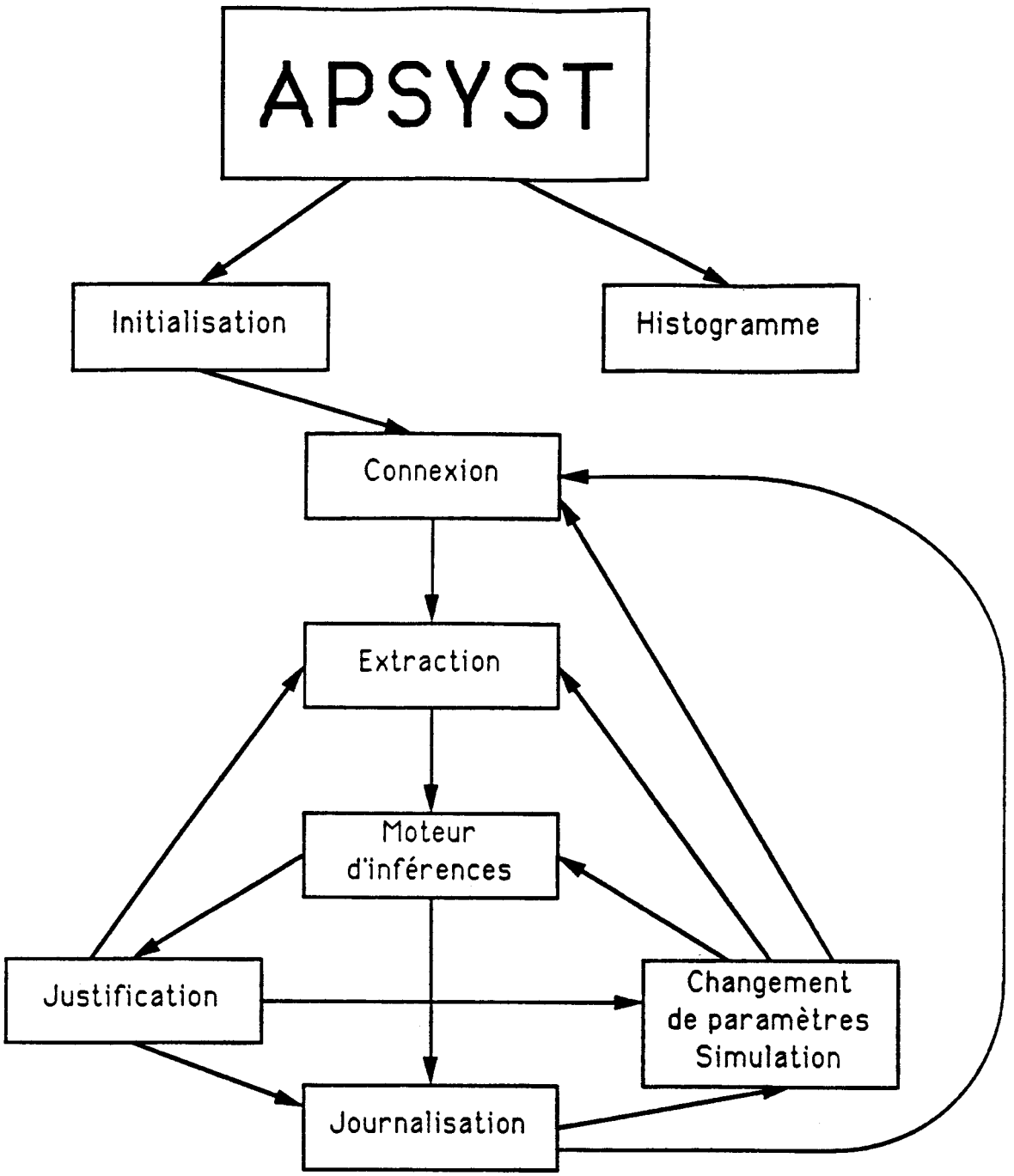


FIGURE II.5
Le mode CONSULTANT

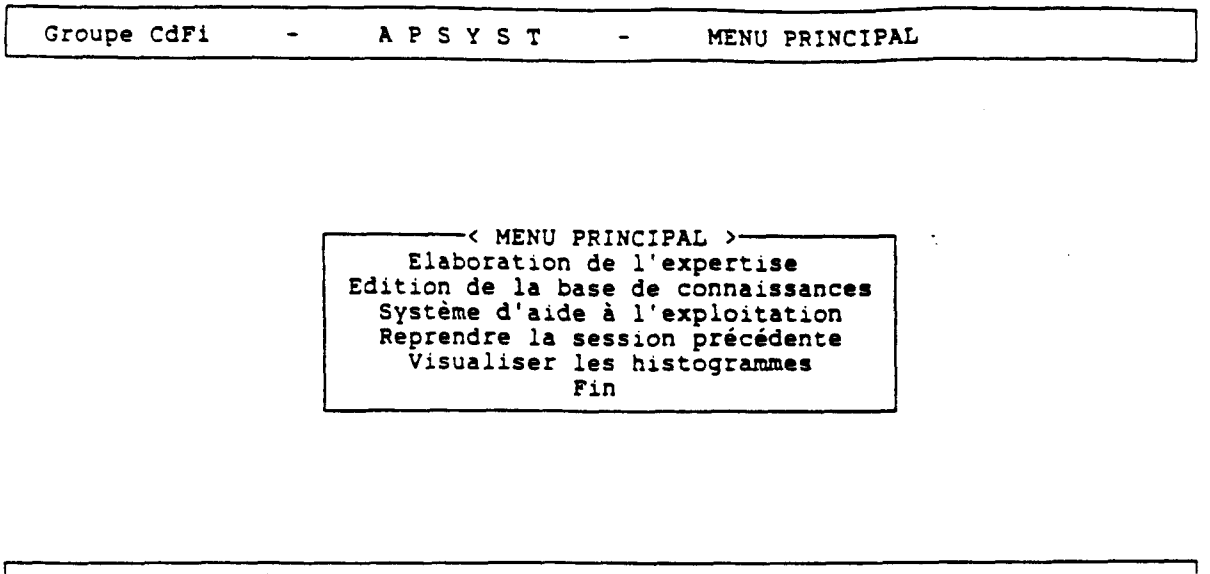


FIGURE II.6
Le menu général

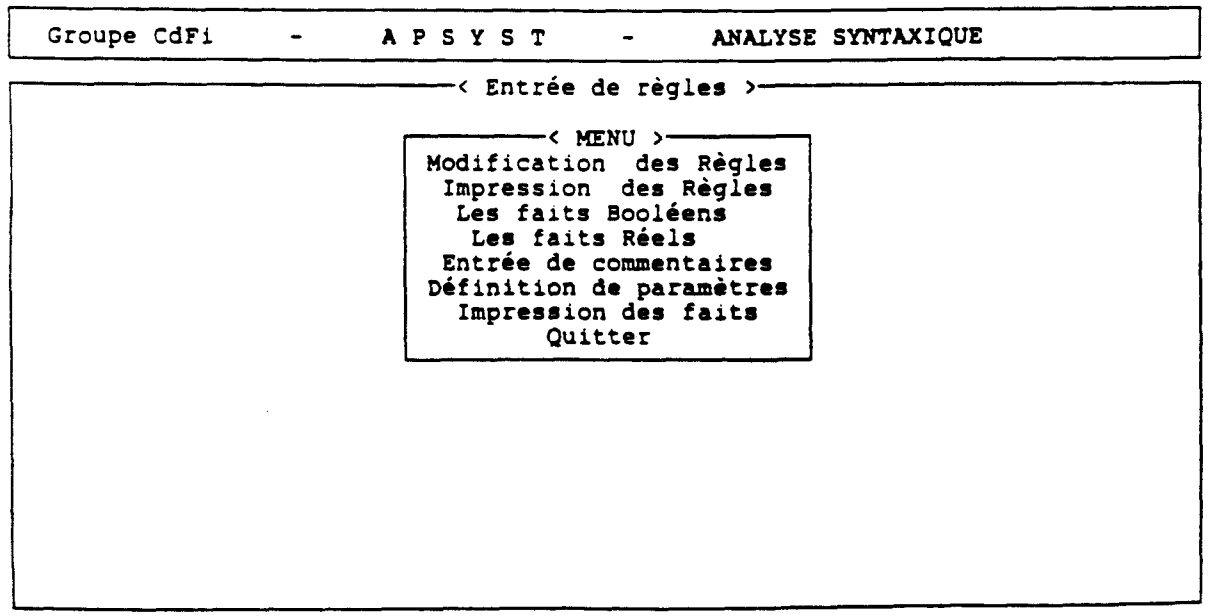


FIGURE II.7
Le menu du mode EXPERT

Les paramètres :
Un paramètre doit tout d'abord être déclaré par l'analyse syntaxique comme un fait de type réel sans variable. Il est alors possible de le déclarer comme paramètre en lui attribuant une valeur par défaut et une explication sur ce que représente cette valeur. Celle-ci sera modifiable à l'initialisation du système d'aide à l'exploitation pour adapter la base de connaissances au site d'installation. Par exemple, un paramètre peut représenter un seuil critique.

Compilateur de règles : (fig. II.8)

A partir de la base de règles, sont calculées les relations entre les faits et règles dans la but de construire un graphe de dépendances. La seule action que fait l'utilisateur dans ce module est la sélection de la base de connaissances sur laquelle doit être calculé le graphe.

Edition de la base de connaissances : (fig. II.9)

C'est un outil d'aide à la mise au point des bases de connaissances. Il permet d'avoir des vues différentes de cette base.

Toutes les fonctionnalités sont doubles : sur écran ou sur imprimante.

- Edition des règles numérotées.

- Edition des faits séparés en 4 groupes :
  1 Faits booléens sans variable
  2 Faits booléens avec variable
  3 Faits réels sans variable
  4 Faits réels avec variable

- Edition des paramètres - 3 informations par paramètre :
  1 Nom du paramètre
  2 Valeur par défaut
  3 Explication

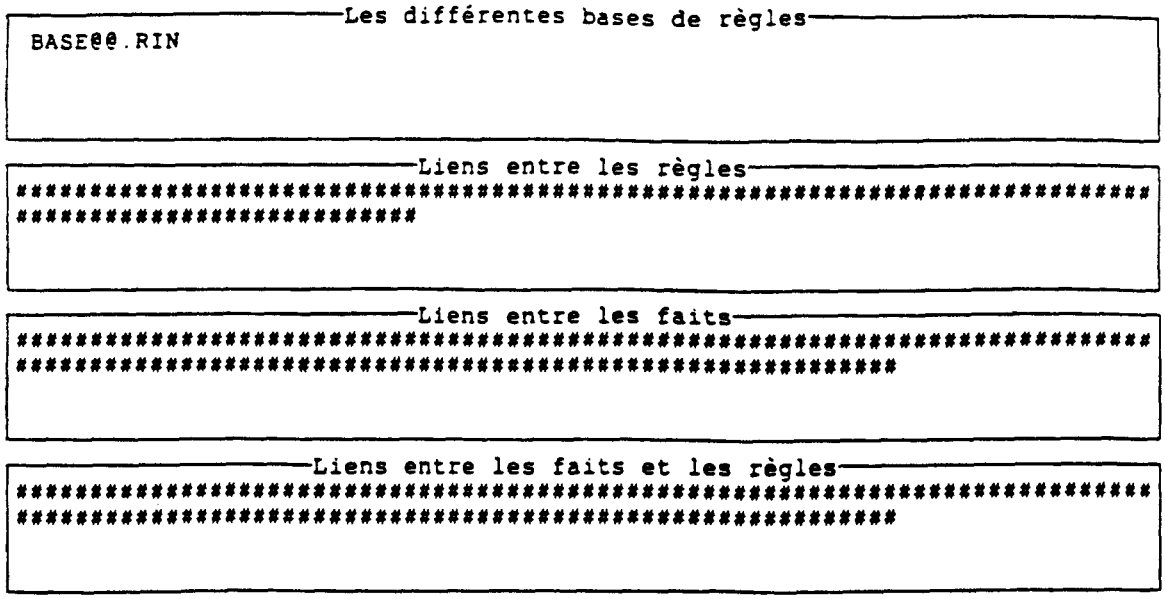


FIGURE II.8
Le compilateur de règles

Fenêtre 1 : Choix de la base
Fenêtre 2 : Etablissement du graphe
Fenêtre 3 : Définition des Faits_avant
Fenêtre 4 : Définition des Règles_arrière_avant

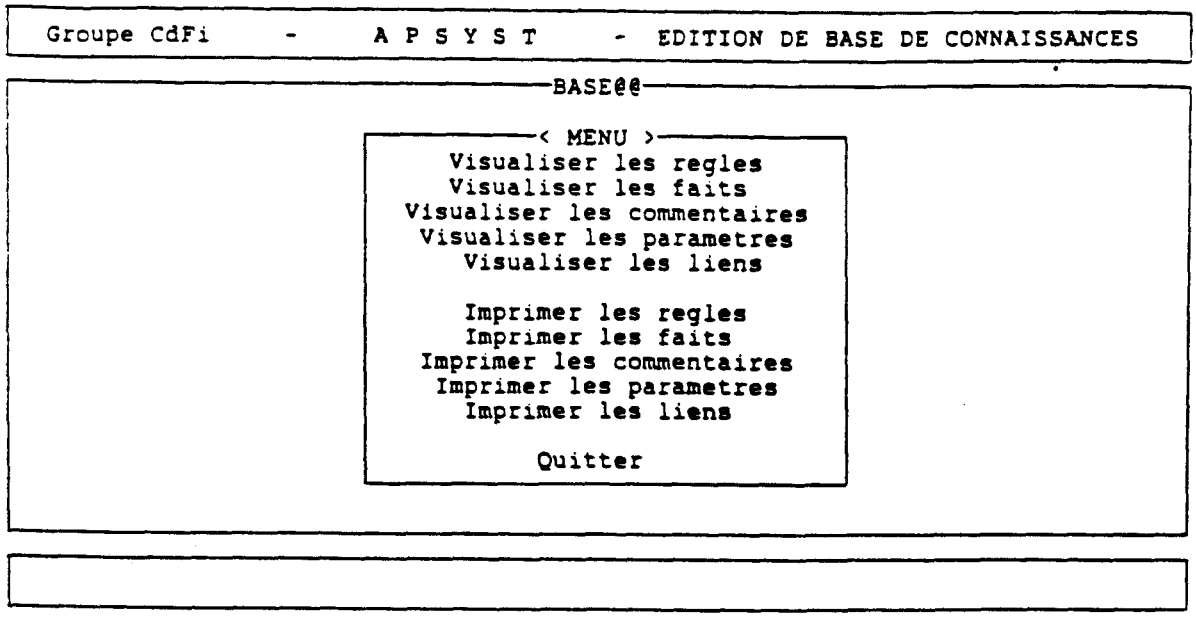


FIGURE II.9
Le menu d'édition
de la base de connaissances

- Edition des commentaires - 3 informations par commentaire :
  1 Numéro de commentaire
  2 Texte générique
  3 Liste des faits et indices à inclure

- Edition du graphe de dépendances - 3 informations par fait :
  1 Nom du fait
  2 Liste des faits à annuler
  3 Liste des règles à utiliser

6.2) Modules d'exploitation du système expert

Initialisation : (fig. II.10)

Ce module permet l'adaptation du système expert au site d'implantation. Après avoir choisi la base de connaissances sur laquelle on veut faire opérer APSYST il faut adapter la valeur de certains paramètres.

Les paramètres propres à l'expertise :
Ce sont les paramètres déclarés dans la phase d'élaboration de la base de connaissances, leur valeur doit être changée en fonction des "normalités" du TP que l'on veut surveiller.

Les paramètres de contexte :
Ils permettent de personnaliser la surveillance. Tous les outils de contrôle du DPS 8 et du TP ne sont pas toujours tous utiles. Il est possible de sélectionner les sources d'informations pour les faits, en ne demandant pas, lors de la connexion, tous les écrans de TPMON ou VIDEO (voir Chap I) plus l'éventuelle demande du contrôle du temps de réponse du TP. Il est possible aussi de demander une connexion transparente ou de voir afficher tous les messages envoyés par le micro-ordinateur.
La présence d'une imprimante reliée au micro-ordinateur est aussi un paramètre, de même que l'exécution du module de justification du raisonnement.

Une fois tous ces paramètres mis à jour, le module d'initialisation rassemble tous les éléments nécessaires au fonctionnement du SE : il crée une base de connaissances complète initiale à partir des règles, des déclarations de faits, du graphe de dépendances, des commentaires et des paramètres.

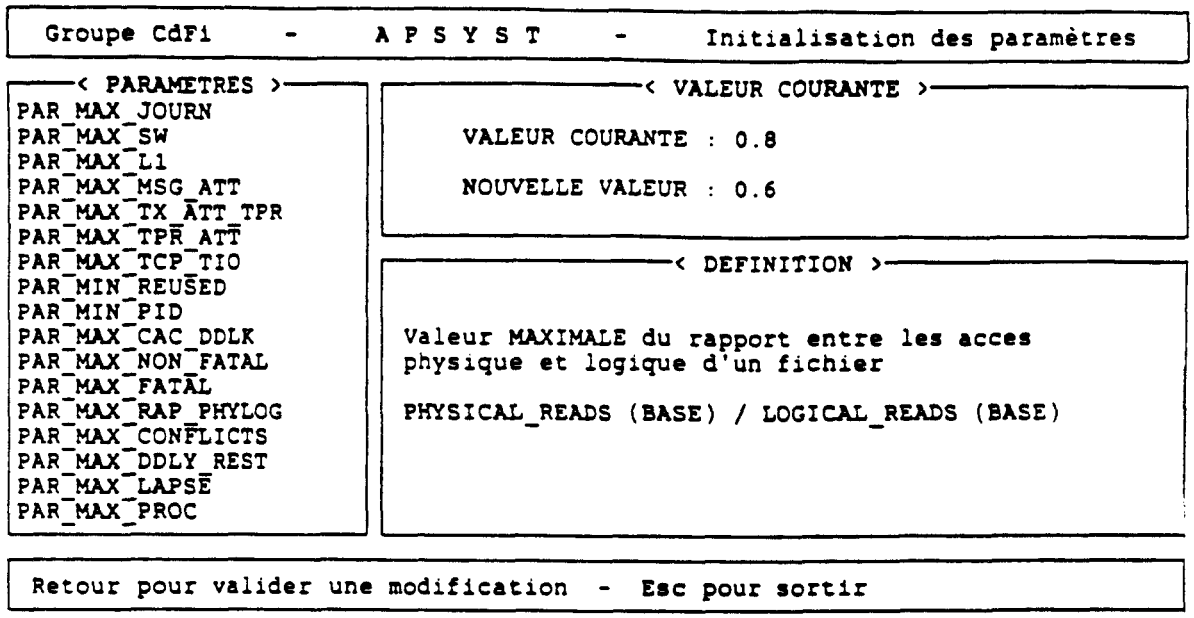


FIGURE II.10
Le module d'initialisation

Les modules:

Connexion

Extraction

Moteur d'inférences

Journalisation

constituent le cycle normal d'utilisation du SE et sont enchaînés sans intervention humaine.

Connexion : (fig. II.11)

Le micro-ordinateur est muni d'une carte d'émulation de terminal standard (voir Environnement matériel). La première étape de la connexion consiste à avertir le contrôleur de grappe de la présence d'un terminal supplémentaire. Ensuite, le micro-ordinateur se connecte au DPS 8 puis demande successivement le lancement des outils de contrôle. Si l'utilisateur a demandé la surveillance des temps de réponse, le micro-ordinateur se connecte tout d'abord au TP en mesurant les différents temps de réponse (Contrôleur de grappe, Frontal, DPS 8, TP).
Si la connexion est transparente l'écran du PC reste figé, sinon les messages envoyés par le micro-ordinateur sont affichés, ainsi que le temps d'attente de la réponse du DPS 8.
Les informations obtenues lors de la phase de connexion sont stockées dans des fichiers.
Seul le fichier contenant l'ordre et le contenu des messages est à modifier pour passer d'un site DPS 8 à l'autre.

Extraction :

Ce module est enchaîné au précédent.
Les fichiers créés par le module de connexion sont traités pour présentation des faits au moteur d'inférences :

- remise en forme
- tri
- calcul de minima et de maxima
- calcul de moyenne.

Tous les faits à intégrer dans la base de connaissances sont stockés dans des fichiers. Le module d'extraction construit aussi des fichiers contenant des séries de chiffres correspondant à des faits particuliers significatifs du fonctionnement du T.P. en vue de construire des histogrammes.

```
Groupe CdFi     -     A P S Y S T     -          CONNEXION

Calcul du temps de réponse du TP
Information à prendre dans EXEC
Information à prendre dans TASK
Information à prendre dans SSA
Information à prendre dans FILE
Information à prendre dans VIDEO
Ouverture de la ligne
 Adresse station = 4
 La connexion a été correctement réalisé
-->'$*$DIS'
```

FIGURE II.11
Le module de connexion

Remarque à propos du calcul de moyenne :
La valeur de certaines données doit rester inférieure à un seuil.
Quand celui-ci est dépassé, 2 cas sont à distinguer :

- le phénomène est durable, et donc il met en évidence une anomalie
- il est ponctuel et représente une situation tout à fait normale.

Le calcul est effectué sur une vingtaine de secondes. Ainsi, les phénomènes ponctuels sont ignorés.

Moteur d'inférences :

L'exécution de ce module se découpe en trois parties

1 - Reprise de la base de connaissances du cycle précédent
2 - Intégration des nouveaux faits
3 - Gestion des éventuelles contradictions
et lancement de l'inférence

La seule chose que voit l'utilisateur pendant l'exécution de ce module, est l'affichage des commentaires.
Ces commentaires sont aussi écrits sur disques dans le but d'être imprimés dans la phase de journalisation.
En fin d'exécution, la base de connaissances est sauvegardée.

Journalisation : (fig. II.12)

Ce module a 3 intérêts :

- Il effectue la journalisation des commentaires sur imprimante ou sur disque (avec l'environnement transactionnel du TP).

- Il permet de revisualiser les commentaires générés par le moteur d'inférence.

- Il évite de faire des cycles trop rapprochés, ce qui surchargerait la grappe de terminaux et modifierait la perception d'évolution du TP

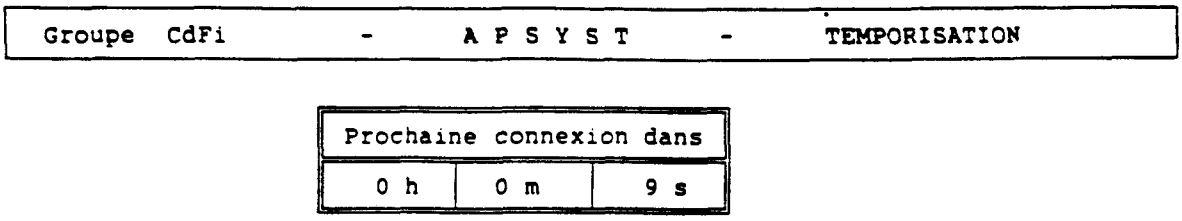


FIGURE II.12
Le module de journalisation

Les modules

Justification

Simulation

Histogramme

des possibilités supplémentaires offrant à l'utilisateur du SE.

Justification : (fig. II.13)

Chaque commentaire généré par le système expert est affiché et imprimé avec à l'heure et la date courante. A chaque cycle de fonctionnement du système expert où il y a génération d'un diagnostic, l'ensemble des écrans constituant la métrologie est stocké sur disque avec l'heure et la date correspondante. Cette opération permet de justifier un commentaire à n'importe quel moment.

Les écrans sont stockés sans aucun travail de mise en forme préalable. De cette manière l'expert aura la possibilité de revoir les informations qui l'intéressent en visualisant les écrans archivés. Ainsi, les écrans lui seront présentés tels qu'il a l'habitude de les voir.

Dans le cas d'une justification à chaud, (i.e. juste après sa génération) le programme retrouve pour chaque fait à justifier son origine. Il affiche la règle dont il est issu et permet de "remonter" de manière récursive jusqu'aux faits provenant de la métrologie.

Dans le cas d'une justification à froid le programme sauvegarde le contexte courant (les écrans venant d'être captés), puis remet les écrans stockés à l'heure de génération du commentaire que l'on veut justifier à l'état courant. A partir de là, sont réactivés le module d'extraction puis le moteur d'inférences qui va permettre de regénérer les commentaires à justifier. Il est possible de justifier ces anciens commentaires comme s'il s'agissait de justifications à chaud. Pour terminer la justification à froid, le S.E. récupère les vrais écrans courants.

Remarque : pendant la justification, la surveillance est interrompue.

Simulation et changement de paramètres : (fig. II.14)

La partie simulation permet de modifier la valeur de certains faits réels normalement issus de la métrologie avant de relancer l'inférence sans repasser par les modules de Connexion et Extraction.

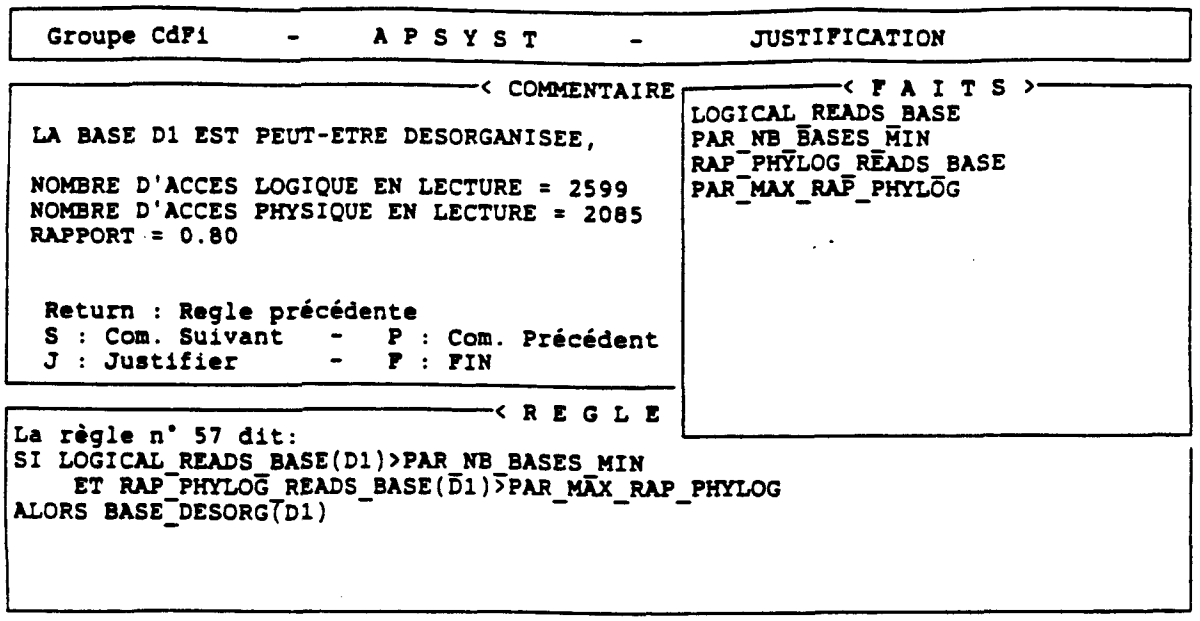


FIGURE II.13
Le module de justification

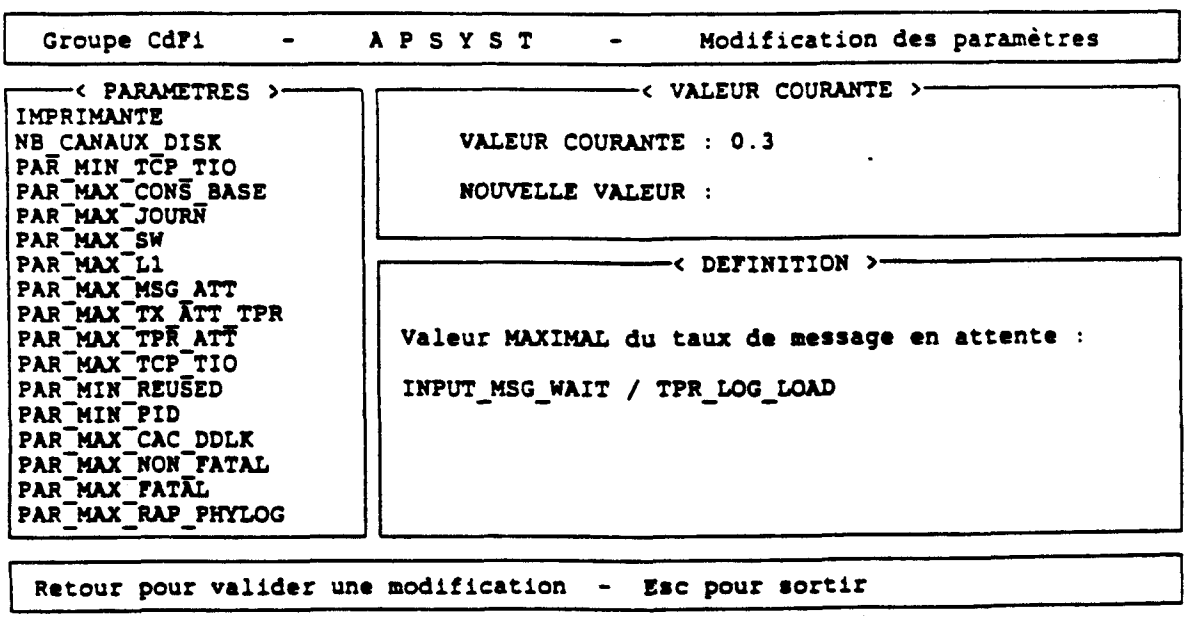


FIGURE II.14
Le module de changement de paramètres

Cette partie permet de répondre à un type de questions que se posent souvent les experts DPS 8 :

"Et si la machine était plus puissante ?",
ou "Et si on limitait les accès ?"
ou "Et si on dédoublait le fichier SWAP ?"

La partie changement de paramètres permet d'adapter les valeurs de certains seuils critiques suivant l'évolution du TP, ou alors suite à une dégradation de la configuration machine (arrêt d'un processeur, crash disque, problèmes réseau, etc). Suite à une modification de la valeur de certains paramètres, il est possible, soit de relancer l'inférence sur la métrologie précédente et donc de réévaluer le diagnostic, soit de reprendre le cycle normal et repartir en Connexion - dans ce cas les modifications de paramètres ne seront prises en compte qu'au cycle suivant.

Histogrammes : (fig. II.15)

Lors de la phase d'Extraction, 4 faits sont régulièrement stockés sous forme de séries. Ces séries de chiffres sont utilisées ici pour construire des histogrammes. Cette sélection de faits présentée graphiquement, permet de se faire une idée globale du fonctionnement du TP pendant toute la durée d'utilisation du système expert.

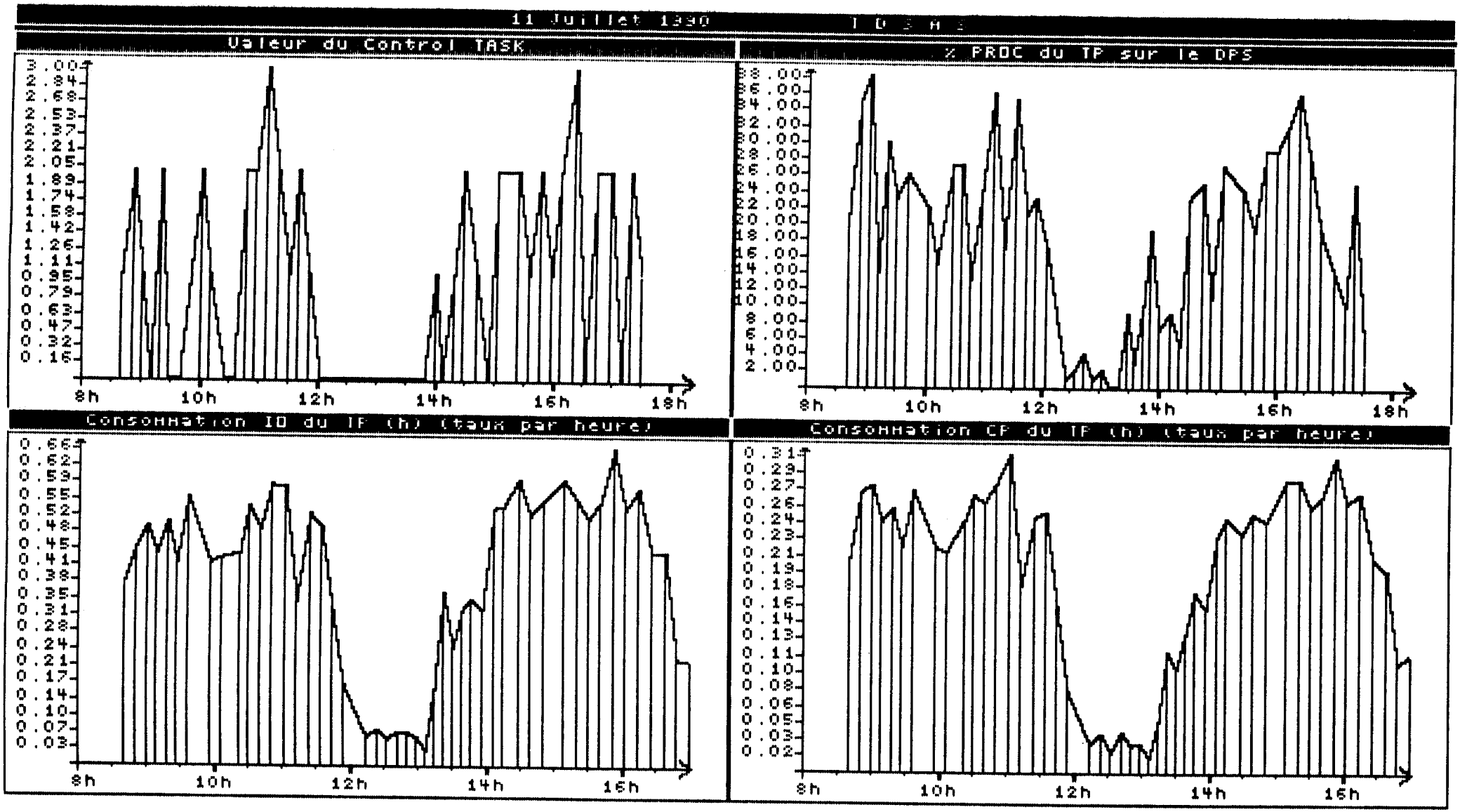


FIGURE II.15
Les histogrammes

7) Exemple de session en mode EXPERT

Nous allons voir une session en mode expert sous forme de quatre images d'écran.

L'expert écrit ses règles dans l'éditeur pleine page (la grammaire B.N.F. des règles est donnée en annexe).

```
Groupe CdFi          -      A P S Y S T      -          ANALYSE SYNTAXIQUE
                              < Entrée de règles >
 F10 pour valider    Line 9     Col 16    Indent  Insert    Esc pour terminer

SI PROCESSOR_TIME=CONNU ET CHANNEL_TIME=CONNU
ALORS RAP_TCP_TIO:=PROCESSOR_TIME/CHANNEL_TIME;

SI RAP_TCP_TIO>PARAM_MAX
ALORS COMMENTAIRE 1;

SI CHANNEL_TIME_BASE(I)=CONNU ET CHANNEL_TIME=CONNU ET CHANNEL_TIME<>0
ALORS TAUX_BASE:=CHANNEL_TIME_BASE(I)/CHANNEL_TIME;

SI TAUX_BASE(I)>PARAM_TAUX
ALORS COMMENTAIRE 2;
```

Il demande la validation de la base de règles qu'il vient d'écrire. L'analyseur syntaxique signale une faute en affichant en bas de l'écran un message d'erreur précis.

```
Groupe CdFi          -      A P S Y S T      -          ANALYSE SYNTAXIQUE
                              < Entrée de règles >
 F10 pour valider    Line 1     Col 1     Indent  Insert    Esc pour terminer
SI PROCESSOR_TIME=CONNU
 ET CHANNEL_TIME=CONNU
ALORS RAP_TCP_TIO:=PROCESSOR_TIME/CHANNEL_TIME;

SI RAP_TCP_TIO>PARAM_MAX
ALORS COMMENTAIRE 1;

SI CHANNEL_TIME_BASE(I)=CONNU
 ET CHANNEL_TIME=CONNU
 ET CHANNEL_TIME<>0
ALORS TAUX_BASE:=CHANNEL_TIME_BASE(I)/CHANNEL_TIME;

  SI TAUX_BASE(I)>PARAM_TAUX ALORS COMMENTAIRE 2;



***ERREUR : TAUX_BASE
 est un fait deja déclaré réel sans variable
```

En effet, en règle 3 l'expert a écrit TAUX_BASE et en règle 4, il a écrit TAUX_BASE(I). Une des deux écritures est erronée, il s'agit de la première. Noter que l'erreur est découverte lors de l'analyse syntaxique de la quatrième règle : cette dernière n'est donc pas remise en forme.
L'expert corrige la faute en écrivant TAUX_BASE(I) dans la règle 3 Ensuite, il demande à nouveau l'analyse syntaxique.

```
Groupe CdFi          -      A P S Y S T      -          ANALYSE SYNTAXIQUE
                          < Entrée de règles >
 F10 pour valider    Line 12    Col 19    Indent  Insert    Esc pour terminer

SI PROCESSOR_TIME=CONNU
 ET CHANNEL_TIME=CONNU
ALORS RAP_TCP_TIO:=PROCESSOR_TIME/CHANNEL_TIME;

SI RAP_TCP_TIO>PARAM_MAX
ALORS COMMENTAIRE 1;

SI CHANNEL_TIME_BASE(I)=CONNU
 ET CHANNEL_TIME=CONNU
 ET CHANNEL_TIME<>0
ALORS TAUX_BASE(I):=CHANNEL_TIME_BASE(I)/CHANNEL_TIME;

  SI TAUX_BASE(I)>PARAM_TAUX ALORS COMMENTAIRE 2;
```

L'analyse syntaxique affiche à nouveau toutes les règles remises en forme : il n'y a plus d'erreur.

```
Groupe CdFi          -      A P S Y S T      -          ANALYSE SYNTAXIQUE
                          < Entrée de règles >
 F10 pour valider    Line 1     Col 1     Indent  Insert    Esc pour terminer

SI PROCESSOR_TIME=CONNU
 ET CHANNEL_TIME=CONNU
ALORS RAP_TCP_TIO:=PROCESSOR_TIME/CHANNEL_TIME;

SI RAP_TCP_TIO>PARAM_MAX
ALORS COMMENTAIRE 1;

SI CHANNEL_TIME_BASE(I)=CONNU
 ET CHANNEL_TIME=CONNU
 ET CHANNEL_TIME<>0
ALORS TAUX_BASE(I):=CHANNEL_TIME_BASE(I)/CHANNEL_TIME;

SI TAUX_BASE(I)>PARAM_TAUX
ALORS COMMENTAIRE 2;
```

Cet exemple montre bien que l'analyseur syntaxique annule toute déclaration de fait avant le début de chaque analyse (puisque l'expert a modifié la première occurrence de TAUX_BASE et non la deuxième).

8) Résultats :

Pour voir un exemple des résultats fournis par une session complète d'utilisation du le S.E., voir en annexe

La structure et le format d'un commentaire sont choisis par l'expert qui entre les règles. Il faut noter que les commentaires sont plus appréciés des utilisateurs quand ils sont découpés en 3 parties : (voir figure II.16)

1 Description du dysfonctionnement
2 Explication
3 Conseil

Les commentaires générés par le S.E. sont affichés à l'écran mais n'y restent que quelques secondes. Il est possible de les revoir lors de la phase de Journalisation, où le programme attend une frappe au clavier après chaque affichage.
Tous les commentaires sont imprimés (lorsque la configuration du système comporte une imprimante) avec à l'heure de la métrologie, et le contexte transactionnel du TP (voir figure II.17)

La figure II.17 représente un diagnostic associé à un extrait de l'écran TASK de TPMON (voir V pour plus de détails).
Il faut retenir ici que sans ce contexte transactionnel, le diagnostic serait difficilement utilisable. Chaque transaction est identifiée par un numéro d'ordre d'arrivée dans le transaction processor: le SERIAL NUMBER. Ce contexte permet de faire le lien entre le ce nombre et les noms de transaction et de TPR.

Plusieurs transactions se sont retrouvées en attente de pages "Data Base Buffers".

On suppose que :

- soit on n'a pas réservé suffisamment de pages au SYSGEN

- soit trop de pages Base de Données ont été accédées entre deux points de phase.

Il faudrait :

- soit ajouter des pages au SYSGEN

- soit analyser le contenu et la structure de la Base de Données

- soit examiner les transactions en cours.

FIGURE II.16
Commentaire en 3 parties

480 pages Base de Données ont été
utilisées par la transaction
de serial number 286, phase n° 15.

Le fonctionnement de la Base de Données
et de la transaction sont à examiner.

Informations sur les transactions en exécution :

```
14:53:51

SER#/PHZ# LID  MSG-ID TPR     CORE M.B. D.B. PID    STATUS
 286/15   MN03 UCXSA  UCTPR1   5K  0/1  4/2 20/480 SYS  IO
 304/12   MN00 UCXSA  UCTPR4   9K  0/0  2/0  5/0   CMPLT
 295/3         HFMRX  HFDEB    3K  1/0  0/0  3/2   PG(A4)
 289/1    MN46 CDFCX  CDINP    4K  1/1  2/0  0/0   CMPLT
```

FIGURE II.17
Commentaire avec
contexte transactionnel

III Gestion de la base de connaissances et temps réel : aspect théorique

La décision de créer un nouveau P.G.S.E. temps réel sur micro nous a amené à étudier précisément les problèmes de gestion de base de connaissances.

Nous avons vu au chapitre précédent les caractéristiques et les contraintes de nos développements. Il faut noter qu'il n'est pas nécessaire pour nous d'avoir des temps de réponse de quelques secondes seulement, et ceci pour deux raisons : les systèmes informatiques que nous surveillons ne permettent pas de réglage en temps réel. Il est cependant important de diagnostiquer en temps réel le fonctionnement du grand ordinateur afin de capitaliser les événements que l'on pourrait qualifier de "critiques".

Une fois le type de matériel choisi (micro-ordinateur compatible IBM PC) il fallait encore choisir un langage de programmation. Le décision d'utiliser PROLOG s'explique facilement quand on connaît les possibilités offertes par ce langage et son adaptabilité aux systèmes experts [BRA 86] [ROS 86], en particulier pour les problèmes de représentations et d'utilisations des bases de connaissances dans un système expert (en particulier pour l'accès associatifs aux données) [GHA 88].

Le fait que le système expert fonctionne en temps réel nous impose un certain nombre de contraintes dont les 2 plus importantes sont :

- gérer les inconsistances de la base de faits engendrées par des modifications partielles de la valeur de certains faits en cours de fonctionnement.
- veiller à ce que les temps de réponse ne soient pas trop élevés afin de rester compatible avec une utilisation en temps réel.

Notre problème d'inconsistance ne résulte pas de l'inférence comme le "Assumption-based truth maintenance system" (A.T.M.S.) de DE KLEER [KLE 86], mais de l'évolution des faits due au temps réel.

La compilation des bases de connaissances est un problème important pour les systèmes experts temps réel, particulièrement en terme d'optimisation et de contrôle de consistance [HAY 85].

Les deux contraintes citées précédemment ont motivé l'étude théorique de la méthode de gestion de la base de connaissances et de la technique d'implémentation du moteur d'inférences.

Après une première partie qui présente les notations utilisées, nous comparerons deux moteurs d'inférences que nous appelons chaînage total et chaînage total.

Chaînage total :

Quelle que soit la valeur des faits du cycle précédent, il enlève de la base de faits tous les faits calculés et applique l'algorithme de saturation en chaînage avant sur la totalité de la base de règles.

Chaînage partiel :

Il n'enlève de la base de faits que les faits calculés qui risquent d'avoir une valeur différente de celle du cycle précédent. Ensuite, il applique l'algorithme de saturation en chaînage avant sur un sous-ensemble minimal mais suffisant de la base de règles.

Le résultat qui termine la partie théorique est que les deux types de chaînage sont équivalents : quelle que soit la base de connaissances de la fin du cycle précédent et quels que soient les nouveaux faits apportés par la métrologie, l'utilisation des deux chaînages donne le même résultat final, à savoir les mêmes commentaires et la même base de connaissances terminale.

Le chaînage partiel opère sur moins de règles que le chaînage total, il est donc bien évidemment plus rapide.

1) Notations et hypothèses

On définit les deux ensembles **Règles** et **Faits** par :

Règles = { $R_1, R_2, R_3, ..., R_n$ }
= un ensemble de règles d'ordre 0 sans cycle

Faits = { $F_1, F_2, F_3, ..., F_p$ }
= un ensemble de faits de type booléens ou réels

R_i= SI conditions ALORS conclusions

conditions = ensemble de prédicats à résultat booléen
conclusions = ensemble d'ordres à exécuter

Une règle est dite **DECLENCHABLE** si tous les prédicats de la partie
condition sont VRAIS

On appelle **DECLENCHEMENT** d'une règle, l'exécution de tous les ordres de la partie conclusion

On appelle **EXECUTION, INTERPRETATION** ou **PRISE EN COMPTE** d'une règle, l'évaluation de la liste de conditions puis l'exécution de la liste de conclusions si elle est déclenchable.

Soit $R \in$ **Règles**

On appelle

cond(R) = ensemble des faits contenus dans la liste de conditions
= { $F \in$ Faits | $F \in$ conditions de R }

concl(R) = ensemble de faits contenus dans la liste de conclusions
= { $F \in$ Faits | $F \in$ conclusions de R }

Faits_de_base = { $F \in$ Faits | $\forall R \in$ Règles, $F \notin$ concl(R)
et $\exists R' \in$ Règles, $F \in$ cond(R') }

On définit la relation '<' entre deux faits :

Soient F ∈ Faits et F' ∈ Faits

F < F' ⇔ ∃ R ∈ Règles telle que F ∈ cond(R)
et F' ∈ concl(R)

Soit la relation '$<^*$' : clôture réflexive et transitive de '<'

∀ F ∈ Faits F $<^*$ F

∀ F, F' ∈ Faits
F $<^*$ F' ⇔ ∃ F" ∈ Faits tel que F < F" et F" $<^*$ F'

Soit la relation '$<^+$' : clôture transitive de '<'

On a bien sûr :

∀ F, F' ∈ Faits
F $<^+$ F' ⇔ F $<^*$ F' et F ≠ F'

F $<^+$ F' ⇒ F $<^*$ F'

On définit les ensembles :

∀ F ∈ Faits
Faits_avant (F) = { F' ∈ Faits, F $<^*$ F' }

∀ F ∈ Faits
Regles_arrière (F) = { R ∈ Règles telles que F ∈ concl(R) }

On étend cette définition aux ensembles.

Soit EF un ensemble de faits
$EF \subseteq Faits$

$\forall EF \subseteq Faits$

$Faits\_avant\ (EF) = \{ F \in Faits \text{ tels que } \exists F' \in EF,\ F' <^* F \}$

$= \bigcup_{F \in EF} Faits\_avant\ (F)$

$\forall EF \subseteq Faits$

$Règles\_arrière\ (EF)\ \{ R \in Règles \text{ telles que } \exists F \in EF, F \in concl(R) \}$

$= \bigcup_{F \in EF} Règles\_arrière\ (F)$

On définit la relation composée :

$\forall F \in Faits$

$Règles\_arrière\_avant\ (F) = Regles\_arrière\ (Faits\_avant\ (F))$

$= \bigcup_{F' \in Faits\_avant(F)} Règles\_arrière\ (F')$

De même, on étend cette relation aux ensembles :

$\forall EF \subseteq Faits$

$Règles\_arrière\_avant(EF) = Règles\_arrière(Faits\_avant\ (EF))$

$= \bigcup_{F' \in Faits\_avant(EF)} Règles\_arrière\ (F')$

$= \bigcup_{F \in EF} Règles\_arrière\_avant\ (F)$

2) Chaînage TOTAL et chaînage PARTIEL

2.1) Mémoire de travail

On appelle **ETAT DE LA MEMOIRE DE TRAVAIL** l'ensemble des faits avec leurs valeurs s'ils sont instanciés.

Soit **ET** un **ETAT DE LA MEMOIRE DE TRAVAIL.**

On définit **ET** par :

ET = { (F,V) F ∈ Faits et V ∈ $\Re$ ∪ {vrai , faux , ind} }
où $\Re$ est l'ensemble des réels
et ind représente l'indétermination d'un fait
(ou non instantiation de ce fait).

On appelle **MODIFICATION** un ensembles de faits de base instanciés.

Soit **M** une **MODIFICATION**

On définit **M** par

M = { (F,V) tels que F ∈ Faits_de_base
et V ∈ $\Re$ ∪ {vrai , faux} }

On définit l'exécution d'une règle R sur un état ET par :

EXEC (ET, R, ET')

où

ET = état de la mémoire de travail avant exécution de R
R = une règle de Reg
ET' = état de la mémoire de travail après exécution de R

Soit ET et ET' deux états de la mémoire de travail
et Reg un sous-ensemble de Règles

On dit que **SATURATION (ET, Reg, ET')** si l'application de l'algorithme de saturation en chaînage avant à partir de **ET** avec les règles de **Reg** donne **ET'**.

Soit ET un état de la mémoire de travail
On note B(ET) le sous-ensemble de ET formé uniquement des couples de faits de base

$ET = \{ (X,V) \mid X \in Faits \}$
$B(ET) = \{ (X,V) \in ET \mid X \in Faits\_de\_base \}$

On dit que ET est SATURE pour Reg
si et seulement si (par définition)
SATURATION (B(ET) , Reg , ET)

Propriété évidente :

ET est saturé $\Leftrightarrow$ SATURATION (ET , Reg , ET)

2.2) Chaînage TOTAL

Soit R un ensemble de règles
F un ensemble de faits
ET un état de la mémoire de travail.
M une modification

On définit le chaînage TOTAL :

CT(ET, M, R, F, ETT)
où ETT représente un état de la mémoire de travail

par l'application de l'algorithme :

$\forall X \in F$ tel que $\exists (X,V) \in M$
si $\exists (X,Z) \in ET$ **CT1(ET,M,F,ETT1)**
remplacer (X,Z) par (X,V) dans ET
sinon ajouter (X,V) dans ET

enlever tous les (X,V) de ETT1 **CT2(ETT1,M,F,ETT2)**
tels que $X \notin$ Faits_de_base

Appliquer SATURATION (ET, R, ETT) **CT3(ETT2,M,F,ETT)**

2.3) Chaînage PARTIEL

Soit R un ensemble de règles
F un ensemble de faits
ET un état de la mémoire de travail.
M une modification

On définit le chaînage PARTIEL :

CP(ET, M, R, F, ETP)
où ETP représente un état de la mémoire de travail

par l'application de l'algorithme :

$\forall$ X $\in$ F tel que $\exists$ (X,V) $\in$ M
si $\exists$ (X,Z) $\in$ ET **CP1(ET,M,F,ETP1)**
remplacer (X,Z) par (X,V) dans ET
sinon ajouter (X,V) dans ET

$\forall$ Y$\in$ F tel que $\exists$ (X,V) $\in$ M
et Y $\in$ Faits_avant(X) **CP2(ETP1,M,F,ETP2)**
enlever de ET le couple (Y,T) s'il existe

REGP=$\varnothing$
$\forall$ X $\in$ F tel que $\exists$ (X,V) $\in$ M **CP3(ETP2,M,F,ETP)**
REGP=REGP $\cup$ Règles_arrière_avant(X)
Appliquer SATURATION (ET, REGP, ETP)

2.4) Equivalence

Proposition : ETT = ETP

Tout d'abord, voyons les algorithmes de plus près.

Chacun est découpé en 3 étapes.
A chaque étape de l'algorithme, l'état de la mémoire de travail évolue.

Au départ, l'état de la mémoire de travail est ET

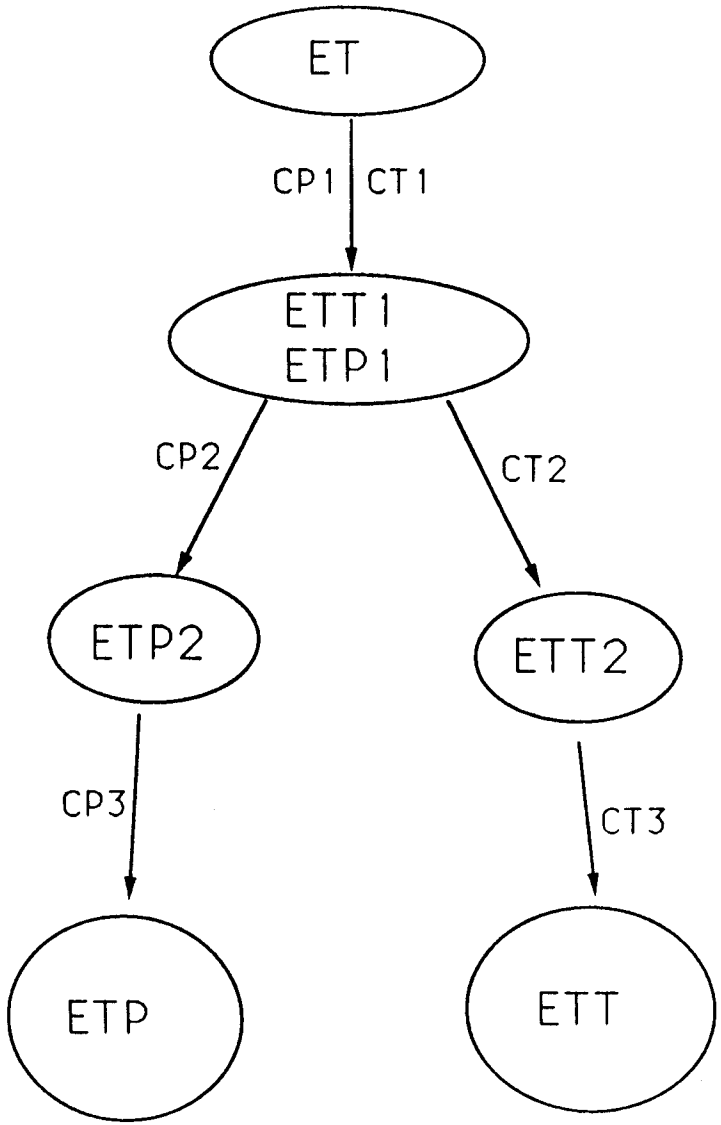
Après CT1 : l'état de la mémoire de travail est : ETT1
Après CP1 : l'état de la mémoire de travail est : ETP1

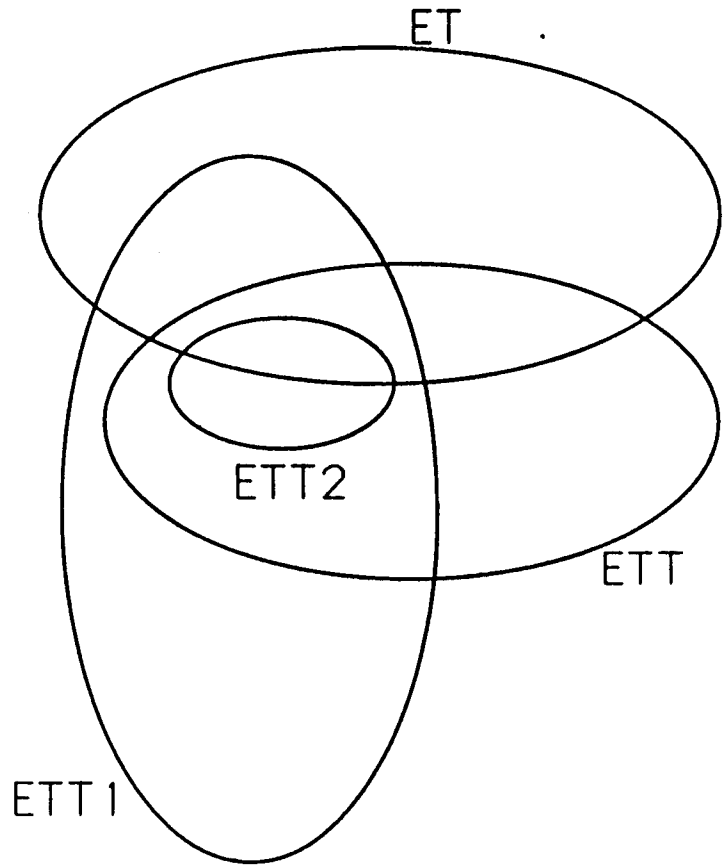
Après CT2 : l'état de la mémoire de travail est : ETT2
Après CP2 : l'état de la mémoire de travail est : ETP2

Après CT2 : l'état de la mémoire de travail est : ETT
Après CP2 : l'état de la mémoire de travail est : ETP

On a les relations suivantes entre les ensembles :

$ETT1 = ETP1$
$ETT2 \subseteq ETP2$
$ETP2 \subseteq ETP1$
$ETT2 \subseteq ETT1$
$ETT2 \subseteq ETT$
$ETP2 \subseteq ETP$


ET
CP1
CT1
ETT1
ETP1
CP2
CT2
ETP2
ETT2
CP3
CT3
ETP
ETT


ET
ETT2
ETT
ETT1

La démonstration de l'égalité des ensembles ETP et ETT à partir d'un état initial ET et d'une modification M est constitué de deux étapes :

1 - Quel que soit le couple (F,V) de ETT1 ou ETP1 tel que F $\notin$ Faits_de_base
S'il n'est pas enlevé par CP2, c'est parce qu'il sera recalculé exactement par CT3.

2 - Quelle que soit la règle n'appartenant pas à l'ensemble des Règles_arrière_avant de M, son EXECUTION sur l'état ETP2 est sans conséquence.

1 - (F,V) n'est pas enlevé de ETT1 se traduit par :
F $\notin$ Faits_avant(M)

On sait que F Faits_de_base

Par définition de l'ensemble des Faits_avant on a:

$\forall$ F1 et F2 $\in$ Faits
F1 $<^+$ F2 $\Leftrightarrow$ F2 $\in$ Faits_avant (F1)
et donc
F2 $\notin$ Faits_avant(F1) $\Leftrightarrow$ non(F1 $<^+$ F2)

Il s'agit de montrer que :

$\forall$ X $<^+$ F , $\forall$ Y $\in$ M, on a non(Y $<^+$ X)

ce qui doit se comprendre comme

[On convient de dire : A **"dépend"** de B si B$<^+$A]

Si X est tel que F "dépend" de X
alors X ne "dépend" d'aucun fait de M

Supposons qu'il existe Y $\in$ M tel que Y $<^+$ X

Autrement dit on suppose que X "dépend" d'un fait de M

Dans ce cas X $\in$ Faits_avant(M)

Comme F∈ Faits_avant(X)

Alors F∈ Faits_avant(M)

La contradiction prouve que

$\forall$ F tel que (F,V) n'est pas enlevé par CP2

$\forall$ X tel que $X <^+ F$

$X \notin M$

et donc

$\forall X <^+ F, X \in$ Faits_base

$(X,T) \in ET$

$(X,T) \in ETT1$

$(X,T) \in ETP2$

$(X,T) \in ETT2$

ainsi

(F,V) sera recalculé par CT3

2 - Soit R ∉ Règles_arrière_avant(M)

On peut aussi écrire :

R ∉ Règles_arrière (Faits_avant (M))

Donc

$\forall$ F ∈ Faits_avant (M)

R ∉ Règles_arrière (F)

$\forall$ Y ∈ concl(R)

$\forall$ X ∈ M

On n'a pas : $X <^{+} Y$ sinon Y appartiendrait à Faits_avant(M)
et donc R appartiendrait à Règles_arrière_avant(M)
$\forall Z \in$ cond(R)
$Z \notin M$
$Z \notin$ Faits_avant(M)

$\forall R' \in$ Règles_arrière_avant(M)
$Z \notin$ cond(R')
$Z \notin$ concl(R')

Et donc
R n'a aucun effet sur ETP2.

Les deux conditions étant vérifiées, on a bien ETT=ETP

et donc CP=CT

Nous venons de démontrer que les deux chaînages sont équivalents au sens des résultats. L' implémentation du chaînage partiel dans le logiciel est intéressante : les éventuelles inconsistances de la base de connaissances (dûes au temps réel) sont gérées et le moteur d'inférences est optimisé.

IV Description technique d'APSYST

1) Généralités

- 3 langages ont été utilisés pour l'écriture des différents modules qui constituent le S.E.:

TURBO PROLOG

Menu Principal
Analyse syntaxique
Compilation des règles
Edition des règles
Initialisation
Moteur d'inférences
Justification
Changement de paramètre / Simulation.

TURBO C

Connexion
Extraction

TURBO PASCAL

Journalisation
Histogramme

Chacun de ces langages permet de créer des "EXECUTABLES" et donc de ne pas utiliser les compilateurs pendant le fonctionnement d'APSYST.

Un autre avantage de ces langages, du point de vue commercial, est la facilité de distribution et de vente de programmes issus des compilateurs de chez BORLAND.

Les bibliothèques de sous-programmes et les outils mis à disposition avec ces compilateurs ont amélioré la qualité des résultats obtenus.

En TURBO-PROLOG tous les prédicats concernant les fenêtres ont été utilisés : créations et effacement de fenêtres temporaires - possibilité d'avoir plusieurs fenêtres simultanément et de les activer une à la fois - utilisation des menus déroulants. En TURBO-PROLOG, il est nécessaire de déclarer les structures de données des fonctions et prédicats qui seront utilisés. Etant donné que ces structures sont les mêmes pour tous les modules du logiciel écrits.en PROLOG, elles sont incluses dans le programme source avant la compilation à partir du fichier APDECL.PRO et sont intégrées et transparentes dans les exécutables. Il en est de même pour les prédicats utilitaires (gestion du clavier, gestion des menus déroulants, opération sur les chaînes de caractères, sur les listes concaténation, membre, etc...) qui sont contenus dans le fichier APUTIL.PRO.
En TURBO-PASCAL, les histogrammes ont été créés en utilisant la librairie GRAPHIX-TOOLBOX.

2) Le menu principal : APSYST.EXE

C'est le menu de lancement du SE.
Son exécution se limite à l'affichage d'un menu et à la sélection du module d'utilisation du SE : mode expert ou consultant.
Dans tous les modules du logiciel, la gestion d'un menu déroulant est assurée par un groupe de prédicats utilitaires dont voici l'algorithme simplifié de fonctionnement.

Le menu d'APSYST.EXE propose 6 possibilités :

1 Elaboration de l'expertise
2 Edition de la base de connaissances
3 Système d'aide à l'exploitation
4 Reprise de la session précédente
5 Visualisation des histogrammes
6 Fin

(voir Fig II.6)

3) Le mode expert

Dans ce mode de fonctionnement, comme nous l'avons vu précédemment, les 3 modules ne peuvent être dissociés.
Voir figure IV.1

3.1) Analyse syntaxique : APA1

Il est possible de disposer de plusieurs bases de connaissances dont une sera choisie lors de l'initialisation du SE. Cela ouvre les possibilités d'utilisation d'APSYST en particulier dans l'avenir de contrôler diverses applications du DPS 8. Actuellement, l'effort a été porté sur la construction d'une base de connaissances relative au T.P. Les fonctionnalités du module APA1 sont proposées par l'intermédiaire d'un menu.
La base de règles est éditée entièrement dans un éditeur pleine page, sous forme d'une chaîne de caractères unique. L'analyse syntaxique est réalisée sur la totalité de la base (page 4 de la Documentation Technique).

Remarque : on appelle DATABASE en Turbo-Prolog, la zone mémoire où sont stockés les éléments déclarés par des "asserts".

Voici l'algorithme simplifié du module APA1.

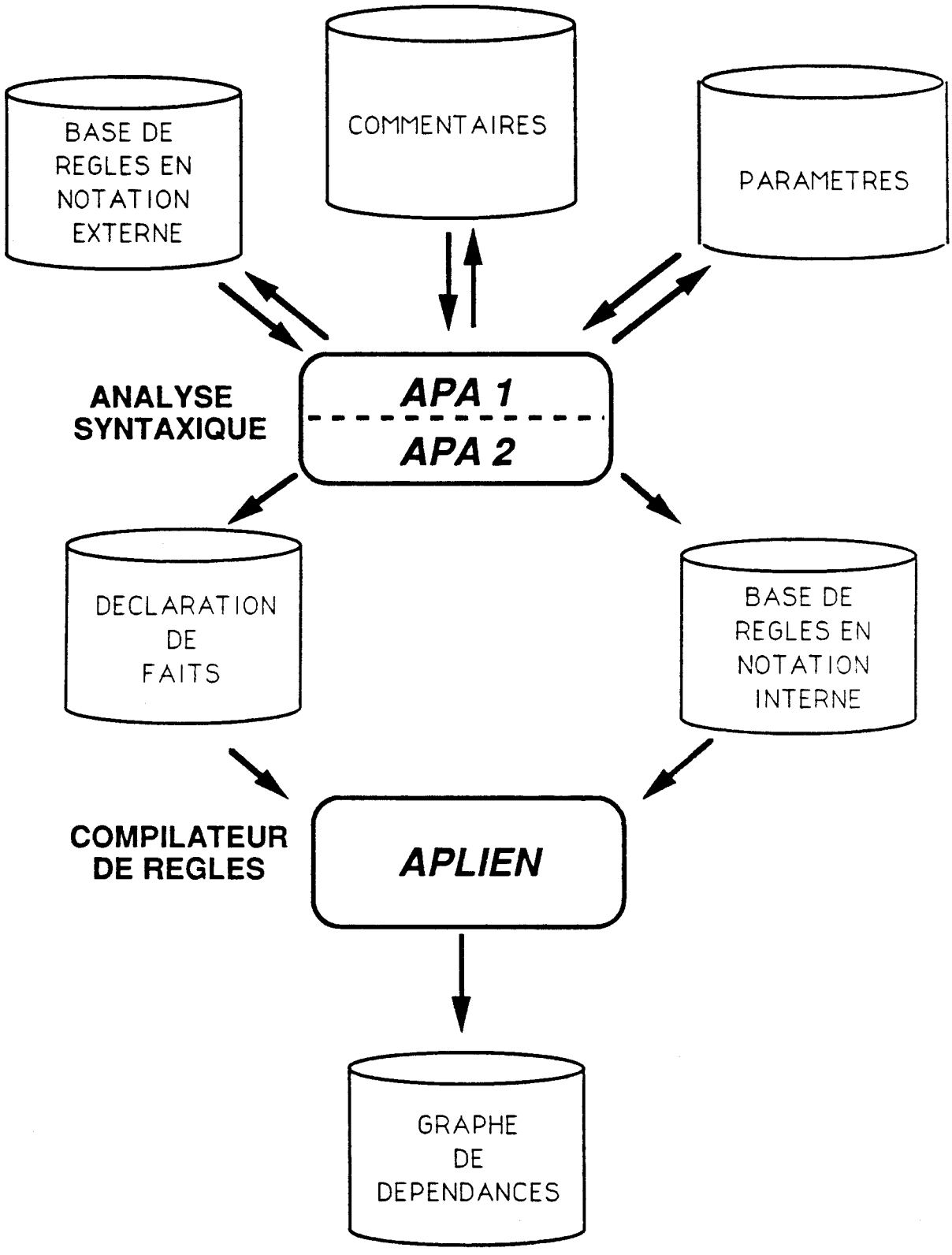


FIGURE IV.1
Les modules du mode EXPERT

```
Début
    . choix de la base de connaissances
    . lecture de la base de connaissances
    . CHOIX ← " " ;
    . tq CHOIX <> "QUITTER" faire
        . afficher le menu
        . CHOIX ← sélection dans le menu
        . CAS CHOIX
            "Modification de règles"
              Répéter
                . Editer la base de règles sous forme d'une chaîne de
                caractères
                . si l'utilisateur demande la validation alors
                    .récupérer la chaîne représentant la base de
                    règles à valider
                    . séparer les règles
                    .mettre dans la database PROLOG toutes les
                    règles à valider
                    . pour chaque règle à valider faire
                        découper la règle en liste de mots (1)
                        analyse syntaxique de la règle (2)
                        si la règle est syntaxiquement correcte alors
                        | remettre la règle en forme (3)
                        sinon
                        | laisser la règle sur une ligne
                        fsi
                    . fpour
                    . reconstruire la chaîne de caractères à éditer
                fsi
              jusqu'à ce que l'utilisateur sorte de l'éditeur par "ESC"
            "Entrée de commentaire"
              Répéter
                . Demander le numéro du commentaire à entrer
                . si le commentaire n'existe pas alors
                    .créer le commentaire en database avec un texte
                    vierge et une liste de faits vide
                fsi
                . Editer le texte du commentaire (4)
                . si le nouveau texte est validé alors
                    Editer la liste de variables associées
                    si la liste est validée alors
                    | . Sauvegarder le nouveau commentaire en
                    | database
                    fsi
                fsi
              jusqu'à ce que l'utilisateur quitte par "ESC"
```

```
                "Définition des paramètres"
                  Répéter
                    . Demander le nom du paramètre à définir
                    . si ce paramètre est déclaré fait réel sans variable
                        alors
                      si le paramètre existe déjà  alors
                        éditer sa valeur par défaut précédemment
                        définie
                        demander la modification de cette valeur
                        éditer l'application associée
                        demander la modification de cette application
                      sinon
                        demander l'entrée d'une valeur par défaut
                        demander l'entrée d'une explication
                      fsi
                      mettre le paramètre dans la database (5)
                    fsi
                  Jusqu'à ce que le nom du paramètre soit "f"
                "Les faits booléens"
                  Editer dans un éditeur pleine page qui ne permet pas les
                  modifications la liste des faits booléens sans variables
                  puis avec variables
                "Les faits réels"
                  Idem "Les faits booléens"
                "Quitter"
                  . Demander la confirmation de la commande "Quitter"
                  si commande confirmée alors
                    demande de sauvegarde de la nouvelle base de règles
                    si l'utilisateur veut sauvegarder  alors
                      écrire le contexte complet sur disque dans un fichier
                      temporaire (6)
                    fsi
                  fsi
              fcas
        ftq
fin
```

Les nombres entre parenthèses renvoient aux explications ci-après.

(1) Analyse lexicale :

Cette opération s'exécute en 2 parties.
Tout d'abord le découpage s'effectue sous forme de mots PROLOG.
Ensuite, les mots qui sont associés sont regroupés :

- Les faits variables avec leur variable ex : A(I)
- Les comparateurs multiples ex : <= , <> , etc
- Les signes multiples ex : :=

Ex : La règle

"Si A(I) <= 5 ALORS B(I) := 4

devient

["Si" , "A" , "(" , "I" , ")" , "<" , "=" , "5" ,
"alors" , "B" , "(" , "I" , ")" , ":" , "=" , "4"]

puis

["Si" , "A(I)" , "<=" , "5" , "alors" , "B(I)" , ":=" , "4"]

(2) Analyse syntaxique de le règle :

Les règles étant ainsi découpées en listes de termes, l'analyse syntaxique est réalisée par unification avec les têtes de listes.
En unifiant avec plusieurs termes, on arrive à une analyse syntaxique sans back-tracking sur une grammaire LL.
Les règles qui sont correctes sont stockées dans la database avec la mention "valide" et les autres avec la mention "invalide".

Au début de l'analyse syntaxique, toutes les précédentes déclarations de faits sont annulées. Pendant l'analyse, chaque fois qu'un fait est rencontré, il est déclaré avec son type suivant le contexte dans lequel il se trouve.
Si le fait existe déjà, il y a vérification du type et génération d'un message d'erreur s'il y a incompatibilité avec la déclaration précédente.

Un certain nombre de messages d'erreur facilite la mise au point de la base de règles. La méthode utilisée pour gérer ces messages est basée sur le fait que l'analyse est réalisée sans back-tracking :
Le succès de l'unification permet de connaitre la suite de la règle.

Un fait database PROLOG contient le numéro d'erreur courant. Il est mis à jour avant chaque vérification. Si une règle n'est pas correcte, l'analyse syntaxique s'arrête pour cette règle, et le message d'erreur courant sera donné, ce qui correspond à la première faute (ou dernière vérification effectuée) dans la règle non valide.

Exemple d'analyse syntaxique d'une règle :

Si A(I) <= 5 alors B(I) := A(I) + 4 et C

1°) élimination du "si" et étude de la partie condition.
["A(I) , <= , 5 , alors , ...]
. Déclaration du fait A de type Réel avec variable
la vérification du comparateur <= est réalisé par l'unification
. Vérification que 5 est soit un nombre soit un fait réel.
La suite est une partie conclusion.

2°) élimination du "alors" et étude de la conclusion
["B(I) , := , A(I) + , 4 , et , ...]
. Déclaration de B de type Réel avec variable
La vérification de signe d'affectation est réalisé par l'unification
. Vérification de la syntaxe de l'expression arithmétique et du type de A déjà déclaré
La suite est encore une conclusion.

3°) élimination du "et" et étude de la conclusion
["C"]
. Déclaration de C de type Booléen :
Fin de l'analyse de la règle avec succès

(3) Remise en forme de la règle :

Dans le but de faciliter la lecture d'une base de règles, celles-ci sont remises en forme. Mais cette opération n'est effectuée que quand la règle est syntaxiquement correcte. Dans le cas contraire, elles sont laissées sous forme d'une seule ligne et sont facilement repérables pour être modifiées.

La forme générale des règles est :

"Si" condition
ET condition
ALORS conclusion
ET conclusion ;

(4) Edition du texte de commentaire :

A chaque texte de commentaire est associée une liste de variables.
Lors de l'affichage du commentaire pendant le fonctionnement du système expert, chaque caractère # du texte du commentaire sera remplacé par une variable de la liste associée (dans l'ordre).

Ces variables peuvent être de 3 types :

- Réel sans variable :
  Le # est remplacé par la valeur du fait
- Réel avec variable :
  Comme une règle ne peut avoir qu'une seule variable, ce sera la valeur de la variable de la règle qui sera pris en compte lors de l'affichage du commentaire
- Variable :
  Les valeurs des indices sont allouées dynamiquement lors de la phase de métrologie et associées à des entités précise du TP (voir explication du moteur d'inférence).

Dans le commentaire, le # est remplacé par la chaîne de caractère associé à l'indice et à sa valeur dans la règle.

Dans la database, le commentaire est déclaré avec 3 paramètres :
. son numéro
. le texte générique (avec des #)
. la liste de variables.

Exemple :

Texte générique : "Le rapport de # est trop fort : #"

Liste associée : [I, RAPPORT]

Base de faits RAPPORT[FIC1] = 0,4

Diagnostic généré : "Le rapport de FIC1 est trop fort : 0,4

(5) Déclaration de paramètre en database :

Dans la database : un paramètre est déclaré avec 3 variables :
. son nom (le même qu'un fait réel sans variable)
. sa valeur par défaut
. son explication.

(6) Sauvegarde temporaire :

Pour éviter que la taille du fichier exécutable ne soit trop importante, l'analyse syntaxique et la génération du code ont été séparées en deux modules.
La méthode utilisée pour passer d'un module à l'autre est la sauvegarde complète du contexte.

3.2) Génération du code interne : APA2

Les 2 modules APA1 et APA2 sont enchaînés sans qu'il y ait intervention de l'utilisateur entre les deux. La base de connaissances complète a été définie dans le module APA1. L'utilisateur n'agit pas pendant l'exécution du module de génération de code interne.

La base de connaissances complète est constituée de 5 fichiers après l'exécution de APA2 :

- notation externe des règles
- notation interne des règles
- définition des faits
- commentaires
- paramètres

voir Figure IV.1

Voici l'algorithme simplifié de APA2 :

```
Début
    . Reprise du contexte sauvegardé en fin d'APA1
    . Effacement ou création des 5 fichiers concernant la base de
    connaissances
    . Pour toutes les règles validées par l'analyse syntaxique faire
        | écrire les règles en notation externe sur le fichier
        | correspondant (1)
    fpour
    . Pour tous les faits déclarés par l'analyse syntaxique faire
        | écrire le fait ainsi que son type sur le fichier
        | correspondant (2)
    fpour
```

```
    . Pour tous les paramètres déclarés faire
        | écrire les caractéristiques des paramètres sur le fichier
        | correspondant (3)
    fpour
    . Pour tous les commentaires faire
        | écrire les caractéristiques des commentaires sur le fichier
        | correspondant (4)
    fpour
    . Pour toutes les règles validées faire
        | générer la notation interne sur le fichier correspondant (5)
    fpour
fin
```

(1) Ecriture en notation externe :

Seules les règles syntaxiquement correctes sont sauvegardées telles qu'elles sont vues dans l'éditeur
Ces règles sont disponibles dans la database sous la forme :
VALIREG (valid , "si...alors...")

L'écriture sur fichier est simplifiée par l'utilisation du PROLOG.

Voici l'algorithme PROLOG de la partie (1).

```
algo 1 :- ouvrir fichier des règles en notation externe, échec
algo 1 :- VALIREG (valid, X),
          écrire_sur_fichier ("VALIREG(valid,",X,")"), échec
algo 1 :- fermer fichier des règles en notation externe.
```

Le même type d'algorithme est utilisé pour (2), (3) et (4).

(2) Déclaration des faits :

Les faits déclarés lors de l'analyse syntaxique sont dans la database sous les formes :

BOOLFACT (BF)
BOOLFACTV (BFV)
REELFACT (RF)
REELFACTV (RFV).

(3) Déclaration des paramètres :

On trouve les paramètres dans la database sous la forme :
PARAM (Nom de paramètre, Explication, Valeur par défaut).
Le nom du paramètre correspond à un fait déclaré réel sans variable
L'explication est un texte.associé au paramètre. Ce texte sera affiché en phase d'initialisation, pour permettre à un *"NON EXPERT"* d'adapter convenablement sa valeur.
La valeur par défaut est indicative.

(4) Déclaration des commentaires :

Un commentaire est en database sous la forme :
COMAF (Numéro, Texte, Liste de fait).

(5) Génération de code interne :

L'algorithme PROLOG de cette partie peut s'écrire ainsi :

```
algo 5 :- ouvrir le fichier des règles en notation interne, échec
algo 5 :- VALIREG (valid, X),
          transformation (X,Y),
          écrire (Y),
          échec
algo 5 :- fermer le fichier
```

La transformation en notation interne se décompose en plusieurs étapes. Comme pour l'analyse syntaxique, la chaîne de caractères représentant une règle est transformée en liste de mots, mais les éléments multiples ne sont pas rassemblés.

Ex : "Si A(I) alors B(I) := 5"
["si" , "A" , "(" , "I" , ")" , "alors" , "B" , "(" , "I" , ")" , ":" , "=" , "5"]

Le programme cherche ensuite s'il s'agit d'un règle avec ou sans variable. Il cherche l'indice de la règle par le prédicat SYVARE : la présence d'un indice dans une règle est repérée par l'existence de la suite de mots : "(" , "?" , ")".
Dans ce cas "?" représente l'indice.
Ceci est valable dans tous les cas en partie condition de règle, par contre, dans la partie conclusion cette suite ne doit pas apparaître entre :
un ":" , "=" et un "ET"
ou un ":" , "=" et un ";"

Ex : SYVARE (["si","A","(","I",")","alors","B","(","I",")",":","=","5"] , X)
1 solution
X = "I"

SYVARE (["si","A",">","2","alors","B"] , X)
1 solution
X = "sans"

Attention, chaque règle ne peut avoir qu'un seul indice.en notation interne.
Si plusieurs indices différents sont présents dans la règle seul le premier compte. Dans la formulation interne de la règle en notation externe, l'indice n'apparaît qu'une fois en notation interne :

SYVARE (["si","A","(","I",")","alors","B","(","J",")",":","=","5"] , X)
1 solution
X = "I"

La règle est ensuite transformée en notation interne par le prédicat GENER.

L'appel se fait sous la forme :

GENER (N, I, R)
où **N** représente le numéro de la règle vers la base
I représente l'indice de la règle ("sans","I","J","etc...)
R représente la règle sous forme de liste de mots.

Exemple :

GENER (4,"I",["si","A","(","I",")","alors","B","(","I",")",":","=","5"])
La règle en notation interne générée est :
REG (4,"I",[est_il_vrai_var (A)] , [affect_var ("B",5)])

Voici 2 tableaux récapitulatifs des équivalences entre notations externe et interne :

CONDITION

EXTERNE	INTERNE
F	est_il_vrai (F)
F (I)	est_il_vrai_var (F)
NON F	est_il_faux (F)
NON F (I)	est_il_faux_var (F)
I == n	associe (I, n)
F = connu	connu (F)
F (I) = connu	connu_var (F)
F = G	est_ce_que (eg, F, G)
F (I) = G	est_ce_que_var_1 (eg, F, G)
F = G (I)	est_ce_que_var_2 (eg, F, G)
F (I) = G (I)	est_ce_que_var_3 (eg, F, G)
=	eg
<	inf
>	sup
<=	infeg
>=	supeg
<>	dif

CONCLUSION

EXTERNE	INTERNE
F	doncvrai (F)
F (I)	doncvrai_var (F)
NON F	doncfaux (F)
NON F (I)	doncfaux_var (F)
Commentaire N	callcom (N)
F := S	affect (F, post_fix (S)) *
F (I) := S	affect_var (F, post_fix (S)) *

Pour accélérer l'exécution des inférences, toutes les expressions sont écrites en code interne en notation post-fixée. De ce fait, la liste contenant le calcul à effectuer est gérée comme une pile, d'où rapidité du calcul (des compléments seront donnés lors de l'explication du moteur d'inférences).

Exemple :

B(I) := ((3 * A(I)) + (6 * C(I)))

devient

REG (..., I , [...] , [... , affect_var (B , [3 , "A(I)" , "*" , 6 , "C(I)" , "*" , "+"] , ...])

Les prédicats database choisis pour représenter les règles en notation interne paraissent lourds à utiliser. Mais cette notation n'est pas connue de l'utilisateur final du logiciel APSYST, qu'il soit expert ou non.

La gestion des prédicats database par TURBO-PROLOG est très efficace :

- **Rapidité** : des tests ont été effectués par Jean-Paul.DELAHAYE en comparaison avec d'autre PROLOG, et TURBO-PROLOG se situe toujours en tête ou alors parmi les meilleurs. Les tests présentés dans un article [CHE 86] de comparaison entre Arity/Prolog, Prolog II/MALI et Prolog II, constituait un tableau que Jean Paul DELAHAYE a complété avec Turbo-Prolog. Parmi les résultats les plus importants, il faut remarquer que Turbo-Prolog permet de construire des listes de plus de 32000 entiers contre seulement 14000 pour Prolg II/MALI, Turbo-Prolog inverse un liste de 100 éléments en 1 seconde contre 10 secondes pour Arity/Prolog. Turbo-Prolog construit une liste de 4050 éléments en 2 secondes, contre 19 secondes pour Prolog II/MALI.
- **Place mémoire** : d'autres tests effectués prouvent que la place mémoire n'est pas fonction de la longueur du nom de prédicat database (bien évidemment).

Pour terminer l'explication de la transformation en notation interne, voici l'exemple de quelques règles.

Notation externe :

```
si A(I) >= 5 alors B(I);

si A(I) = connu
  et X(I) = connu
  et Y(I) = connu
alors M(I) := ( (X(I) + Y(I) / A(I)) ) ;

si M(I) < 1
alors commentaire 1 ;

si A(I) < SEUIL(I)
alors commentaire 2 ;
```

Notation interne :

```
REG (1 , I , [est_ce_que_var_1 (supeg,A,5)] ,
             [doncvrai_var (B)] )

REG (2 , I , [connu_var (A) , connu_var (X) , connu_var (Y)]
             [affect_var (M,[X(I),Y(I),"+",A(I),"/"]] )

REG (3 , I , [est_ce_que_var_1 (inf,M,1)] ,
             [callcom (1)] )

REG (4 , I , [est_ce_que_var_3 (inf (A,SEUIL)] ,
             [callcom (2)] )
```

3.3) Calcul du graphe de dépendances : APLIEN

Le calcul du graphe de dépendances est réalisé en suivant les définitions théoriques définies en II.3.
Pour ce module, 2 fichiers sont nécessaires en entrée :

1) La base de règles en notation interne

2) La déclaration de faits

Le module APLIEN est obligatoirement enchaîné aux 2 modules d'analyse syntaxique et génération du code interne. Cependant, la construction du graphe peut être réalisée sur n'importe quelle base de

connaissances. La première chose à faire est de choisir la base de connaissances pour laquelle il faut calculer le graphe.

Une fois ce choix fait, le module charge en mémoire les 2 fichiers nécessaires, puis lance le calcul du graphe.

Voici l'algorithme de construction

```
Début
  Pour chaque règle faire
    . construction de la liste des faits en partie condition FCD (1)
    . construction de la liste des faits en partie conclusion FCL (2)
    Pour chaque fait de FCD faire
      déclarer la règle comme règle_arrière_avant de ce fait (3)
    fpour
    Pour chaque fait de FCL faire
      déclarer la règle comme règle_arrière_avant de ce fait (4)
    fpour
    . déclarer chaque fait de FCL comme fait_à_modifier de chaque
    fait de FCD (5)
  fpour
  Pour chaque fait déclaré faire
    . prendre tous les faits_devant par clôture transitive et les
    déclarer comme les faits_à_modifier de ce fait (6)
  fpour
  Pour chaque fait qui possède une liste de fait_à_modifier faire
    construire la liste de règles composé des règles_arrière_avant
    des faits_à_modifier (7)
  fpour
  Pour chaque fait qui possède des faits_avant faire
    . le définir comme fait_de_base s'il n'est fait_avant d'aucun
    autre fait (8)
  fpour
Fin
```

(1) & (2) Listes des faits des parties condition et conclusion :

Ces listes sont créées sans aucune préoccupation de la présence ou non d'un indice dans la règle.
La méthode utilisée est basée sur l'unification des prédicats de la partie condition et conclusion avec les prédicats existants.
Il n'est pas utile de faire des vérifications d'ordre syntaxique ou sémantique : la base de règles étant en notation interne, il ne peut y avoir d'erreur.

(3) & (4) Déclaration de règle_arrière_avant :

Si le fait possède déjà des règles_arrière_avant, le numéro de la nouvelle règle est ajouté à la liste sinon la liste est créée avec ce numéro de règle.

(5) Déclaration de fait_à_modifier :

A chaque fait de FCD est associée la liste de fait_à_modifier FCL en entier

(6) Clôture transitive des faits_à_modifier:

Faire l'itération dans la boucle sur les faits définis, permet, dans le cas d'une partie conclusion réduite à l'appel d'un commentaire, de mettre par défaut une liste vide de fait_à_modifier.

(7) Clôture transitive des règles_arrière_avant :

Tout fait déclaré possède une liste de fait_à_modifier. C'est sur le prédicat database "fait_à_modifier" qu'est réalisée l'itération.

(8) Déclaration des faits_de_bases :

Il suffit de vérifier pour chaque fait qu'il n'appartient à aucune liste de "fait_à_modifier". Autrement dit il s'agit d'un fait issu de la métrologie ou d'un paramètre.

Les 8 points précédents demandent l'utilisation des listes. Pour toutes ces manipulations, un certain nombre de prédicats de base de travail sur les listes ont été développés :

- addition d'élément en tête de liste
- addition d'élément en queue de liste
- concaténation de listes
- inversion de liste
- suppression des éléments doublés.

Exemple :

Base de règles :

R1 si F1 et F2 alors F6
R2 si F3 et F4 alors F7
R3 si F4 et F5 alors F8
R4 si F6 et F7 alors F9
R5 si F9 et F8 alors F

La figure IV.2 représente le graphe de dépendances correspondant.

Grâce à ce graphe, on peut constater que quand la valeur de F1 est modifiée, l'inférence n'est utile que sur les règles R1, R4 et R5 (c'est à dire où la valeur de F1 intervient directement ou indirectement).

Voici en exemple les différentes étapes de la construction du graphes :

R1 : faits de la partie condition : [F1 , F2] (1)
faits de la partie conclusion : [F6] (2)
Assertions :
règle_arrière_avant (F1 , [1])
règle_arrière_avant (F2 , [1]) (3)
règle_arrière_avant (F6 , [1]) (4)
fait_à_modifier (F1 , [F6])
fait_à_modifier (F2 , [F6]) (5)

R2 : faits de la partie condition : [F3 , F4]
faits de la partie conclusion : [F7]
Assertions :
règle_arrière_avant (F3 , [2])
règle_arrière_avant (F4 , [2])
règle_arrière_avant (F7 , [2])
fait_à_modifier (F3 , [F7])
fait_à_modifier (F4 , [F7])

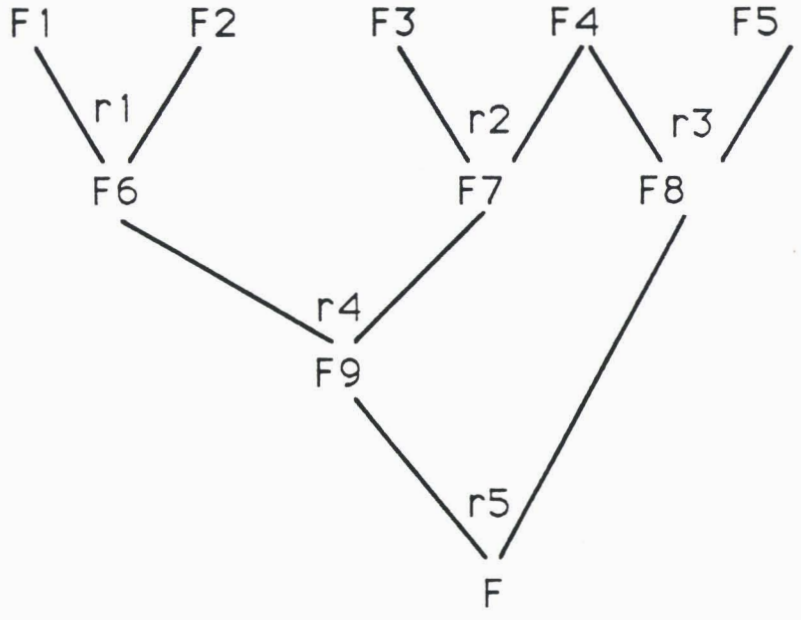


FIGURE IV.2
Graphe de dépendances

R3 : faits de la partie condition : [F4 , F5]
faits de la partie conclusion : [F8]
Assertions :
règle_arrière_avant (F4 , [3])
règle_arrière_avant (F5 , [3])
règle_arrière_avant (F8 , [3])
fait_à_modifier (F4 , [F7 , F8])
fait_à_modifier (F5 , [F8])

R4 : faits de la partie condition : [F6 , F7]
faits de la partie conclusion : [F9]
Assertions :
règle_arrière_avant (F6 , [4])
règle_arrière_avant (F7 , [4])
règle_arrière_avant (F9 , [4])
fait_à_modifier (F6 , [F9])
fait_à_modifier (F7 , [F9])

R5 : faits de la partie condition : [F8 , F9]
faits de la partie conclusion : [F]
Assertions :
règle_arrière_avant (F9 , [5])
règle_arrière_avant (F8 , [5])
règle_arrière_avant (F , [5])
fait_à_modifier (F9 , [F])
fait_à_modifier (F8 , [F])

Un premier passage a permis de définir les relations de dépendance immédiates de fait à fait et de règle à fait.
Il faut maintenant appliquer un algorithme de clôture transitive sur ces relations :
(6) pour les faits_à_modifier
et (7) pour les regles_arrière_avant

F1 : fait_à_modifier (F1 , [F6]) **(6)**
& fait_à_modifier (F6 , [F9])
==> fait_à_modifier (F1 , [F6 , F9])

fait_à_modifier (F1 , [F6 , F9])
& fait_à_modifier (F9 , [F])
==> fait_à_modifier (F1 , [F6 , F9 , F])

F2 : fait_à_modifier (F2 , [F6])
& fait_à_modifier (F6 , [F9])
==> fait_à_modifier (F2 , [F6 , F9])

fait_à_modifier (F2 , [F6 , F9])
& fait_à_modifier (F9 , [F])
==> fait_à_modifier (F2 , [F6 , F9 , F])

F3 : fait_à_modifier (F3 , [F7])
& fait_à_modifier (F7 , [F9])
==> fait_à_modifier (F3 , [F7 , F9])

fait_à_modifier (F3 , [F7 , F9])
& fait_à_modifier (F9 , [F])
==> fait_à_modifier (F3 , [F7 , F9 , F])

F4 : fait_à_modifier (F4 , [F7 ,F8])
& fait_à_modifier (F7 , [F9])
==> fait_à_modifier (F4 , [F7 , F8 , F9])

fait_à_modifier (F4 , [F7 , F8 , F9])
& fait_à_modifier (F8, [F])
==> fait_à_modifier (F4 , [F7 , F8 , F9 , F])

fait_à_modifier (F4 , [F7 , F8 , F9 , F])
& fait_à_modifier (F9, [F])
==> fait_à_modifier (F4 , [F7 , F8 , F9 , F , F])

fait_à_modifier (F4 , [F7 , F8 , F9 , F , F])
==> fait_à_modifier (F4 , [F7 , F8 , F9 , F])
par élimination des éléments en double

F5 : fait_à_modifier (F5 , [F8])
& fait_à_modifier (F8 , [F])
==> fait_à_modifier (F5 , [F8 , F])

F6 : fait_à_modifier (F6 , [F9])
& fait_à_modifier (F9 , [F])
==> fait_à_modifier (F6 , [F9 , F])

F7 : fait_à_modifier (F7 , [F9])
 & fait_à_modifier (F9 , [F])
 ==> fait_à_modifier (F7 , [F9 , F])

F8 : fait_à_modifier (F8 , [F])

F9 : fait_à_modifier (F9 , [F])

F : fait_à_modifier (F , [ ])

F1 : règle_arrière_avant (F1 , [1]) (7)
règle_arrière_avant (F1 , [1 , 1 , 4]) + 1 et 4 pour F6
règle_arrière_avant (F1 , [1 , 1 , 4 , 4 , 5]) + 4 et 5 pour F9
règle_arrière_avant (F1 , [1 , 1 , 4 , 4 , 5 , 5]) + 5 pour F
règle_arrière_avant (F1 , [1 , 4 , 5]) suppression des doubles

F2 : règle_arrière_avant (F2 , [1])
règle_arrière_avant (F2 , [1 , 1 , 4]) + 1 et 4 pour F6
règle_arrière_avant (F2 , [1 , 1 , 4 , 4 , 5]) + 4 et 5 pour F9
règle_arrière_avant (F2 , [1 , 1 , 4 , 4 , 5 , 5]) + 5 pour F
règle_arrière_avant (F2 , [1 , 4 , 5]) suppression des doubles

F3 : règle_arrière_avant (F3 , [2])
règle_arrière_avant (F3 , [2 , 2 , 4]) + 2 et 4 pour F7
règle_arrière_avant (F3 , [2 , 2 , 4 , 4 , 5]) +4 et 5 pour F9
règle_arrière_avant (F3 , [2 , 2 , 4 , 4 , 5 , 5]) + 5 pour F
règle_arrière_avant (F3 , [2 , 4 , 5]) suppression des doubles

F4 : règle_arrière_avant (F4 , [2 , 3])
règle_arrière_avant (F4 , [2 , 3 , 2 , 4]) + 2 et 4 pour F7
règle_arrière_avant (F4 , [2 , 3 , 2 , 4 , 3 , 5]) + 3 et 5 pour F8
règle_arrière_avant (F4 , [2 , 3 , 2 , 4 , 3 , 5 , 4 , 5])
+ 4 et 5 pour F9
règle_arrière_avant (F4 , [2 , 3 , 2 , 4 , 3 , 5 , 4 , 5 , 5])
+ 5 pour F
règle_arrière_avant (F4 , [2 , 3 4 , 5]) suppression de double

F5 : règle_arrière_avant (F5 , [3])
règle_arrière_avant (F5 , [3 , 3 , 5])
règle_arrière_avant (F5 , [3 , 3 , 5 , 5])
règle_arrière_avant (F5 , [3 , 5])

F6 : règle_arrière_avant (F6 , [4])
règle_arrière_avant (F6 , [4 , 4 , 5])
règle_arrière_avant (F6 , [4 , 4 , 5 , 5])
règle_arrière_avant (F6 , [4 , 5])

F7 : règle_arrière_avant (F7 , [4])
règle_arrière_avant (F7 , [4 , 4 , 5])
règle_arrière_avant (F7 , [4 , 4 , 5 , 5])
règle_arrière_avant (F7 , [4 , 5])

F8 : règle_arrière_avant (F8 , [5])
règle_arrière_avant (F8 , [5 , 5])
règle_arrière_avant (F8 , [5])

F9 : règle_arrière_avant (F9 , [5])
règle_arrière_avant (F9 , [5 , 5])
règle_arrière_avant (F9 , [5])

F : règle_arrière_avant (F , [5])

Seuls F1, F2, F3, F4 et F5 sont des faits de bases **(8)**

Tous les liens calculés sont sauvegardés dans le fichier avec l'extension ".LNK".
En plus, les faits de base qui sont avec l'extension ".FBA"

3.4) Edition de la base de connaissances : APEDIT.

Ce module est appelé à partir du menu principal de APSYST. Il est prévu pour faciliter la mise au point de la base de connaissances, en éditant les objets qui la constituent sous forme lisible et claire.
L'édition peut se faire soit sur écran, soit sur imprimante. Dans les 2 cas, la méthode utilisée est sensiblement la même. La différence majeure étant de rediriger la sortie sur imprimante.

De même que pour les modules précédemment décrits, l'utilisateur doit commencer par choisir la base de règles à charger en mémoire.

Les règles : Les règles sont affichées avec leur numéro dans la base, à partir de leur notation interne. Un groupe de prédicats assure la transformation des règles en notation externe. Ces mêmes prédicats sont utilisés dans le module de justification.
A l'écran, une seule règle est affichée à la fois. Ensuite, le programme attend que l'utilisateur demande la règle suivante, ou la précédente Sur l'imprimante, toute la base de règles est imprimée en une fois.

Les faits : L'ensemble des faits est visualisé dans une fenêtre par le prédicat DISPLAY.
Le prédicat FINDALL (de TURBO-PROLOG) permet de construire la liste de tous les faits déclarés par des "asserts" en database.
Cette liste est ensuite transformée en une chaîne unique.

Ex : Un fait réel FR est déclaré en database par :
assert (REELFACT (FR))
Pour obtenir la liste des faits réels :
FINDALL (X , REELFACT (X) , LF)
X doit être une variable libre.

Les faits sont séparés en 4 parties, chacune d'elle construite par un FINDALL : booléens sans et avec variable, réel sans et avec variables.

Les commentaires : La méthode utilisée est la même que pour les règles : au choix de l'utilisateur, un commentaire à la fois .

Les paramètres : De même que pour les faits le prédicat FINDALL construit la liste des commentaires associés à leur valeur et leur

explication en vue de construire une chaîne unique éditée par un DISPLAY.

Les liens : A cause de la limitation de la taille des chaînes de caractères, il n'est pas possible de construire une liste unique des faits reliés par des liens du graphe de dépendances. La solution retenue a été celle de visualiser pour chaque fait la liste des faits_devant et la liste des règles_arrière_avant un fait à la fois à l'écran. Mais pour l'envoi sur imprimante tout le graphe est imprimé en une fois.
Pour augmenter la lisibilité du graphe, 2 prédicats supplémentaires permettent de calculer la longueur des chaînes de caractères afin de générer des sauts de ligne pour ne pas couper les noms de fait.

4) Le module consultant

Le système APSYST est conçu pour que l'expert n'ait plus à intervenir en mode consultant.

Pour l'écriture des modules qui constituent le mode consultant, certaines directives ont été données par les experts du site de développement. Celles-ci seront expliquées pour chaque module.

4.1) Initialisation

Il existe un autre type de paramètres qui permet de sélectionner les informations qui doivent être capturées lors de la connexion :
les paramètres de contexte.

La formulation d'un paramètre de contexte en database est la suivante :

AUTPAR (NP, EXP, VPD, LV) Où
NP représente le nom d'identification du paramètre
EXP représente l'explication de ce qu'il représente
VPD représente la valeur par défaut
LV représente la liste des valeurs possibles.

Les paramètres de ce type ne peuvent pas être déclarés par l'ingénieur système pendant l'utilisation du mode expert.

En fait, ceux-ci ne diffèrent pas d'un site à l'autre et ne modifient pas l'utilisation d'APSYST. Leur évolution ne peut avoir que 2 origines :

1. - évolution des outils de contrôle du constructeur.

2. - évolution du système APSYST

Ces paramètres sont déclarés dans le code source du module APINIT.

Paramètres intervenant sur la connexion :

- **VIDEO** : connexion oui ou non au moniteur de surveillance du système d'exploitation du DPS 8 : GCOS 8.

- **EXEC, TASK, FILE, CORE, DACQ, SSA** : connexion oui ou non au différents moniteurs de contrôle du TP sur le DPS 8

- **TEMPS DE REPONSE** : connexion oui ou non au TP sur une transaction étalon et calcul des différents temps de réponse

- **TRACE DE CONNEXION** : trace sur l'écran des messages envoyés par le micro-ordinateur au DPS 8. Cette option permet de suivre la connexion du micro-ordinateur au DPS 8.

Autres paramètres :

- **TEMPORISATION** : ce paramètre permet de fixer le temps d'attente entre 2 cycles consécutifs

- **IMPRIMANTE** : présence ou non d'une imprimante reliée au micro-ordinateur. La journalisation est faite sur fichier et sur imprimante

- **JUSTIFICATION** : le module de justification peut être inséré ou non dans le cycle normal augmente au risque d'augmenter le temps de cycle. Mais si personne ne demande de justification pendant les 20 premières secondes de fonctionnement du module, celui-ci cède sa place au module de JOURNALISATION

- **JOURNAL** : ce paramètre n'apparaît que si un fichier journal de fonctionnement du SE est présent sur disque. Trois valeurs sont possibles pour ce paramètre :
  0 - Archivage sous un autre nom
  1 - Impression du journal avant destruction
  2 - Destruction

Voir figure IV.3

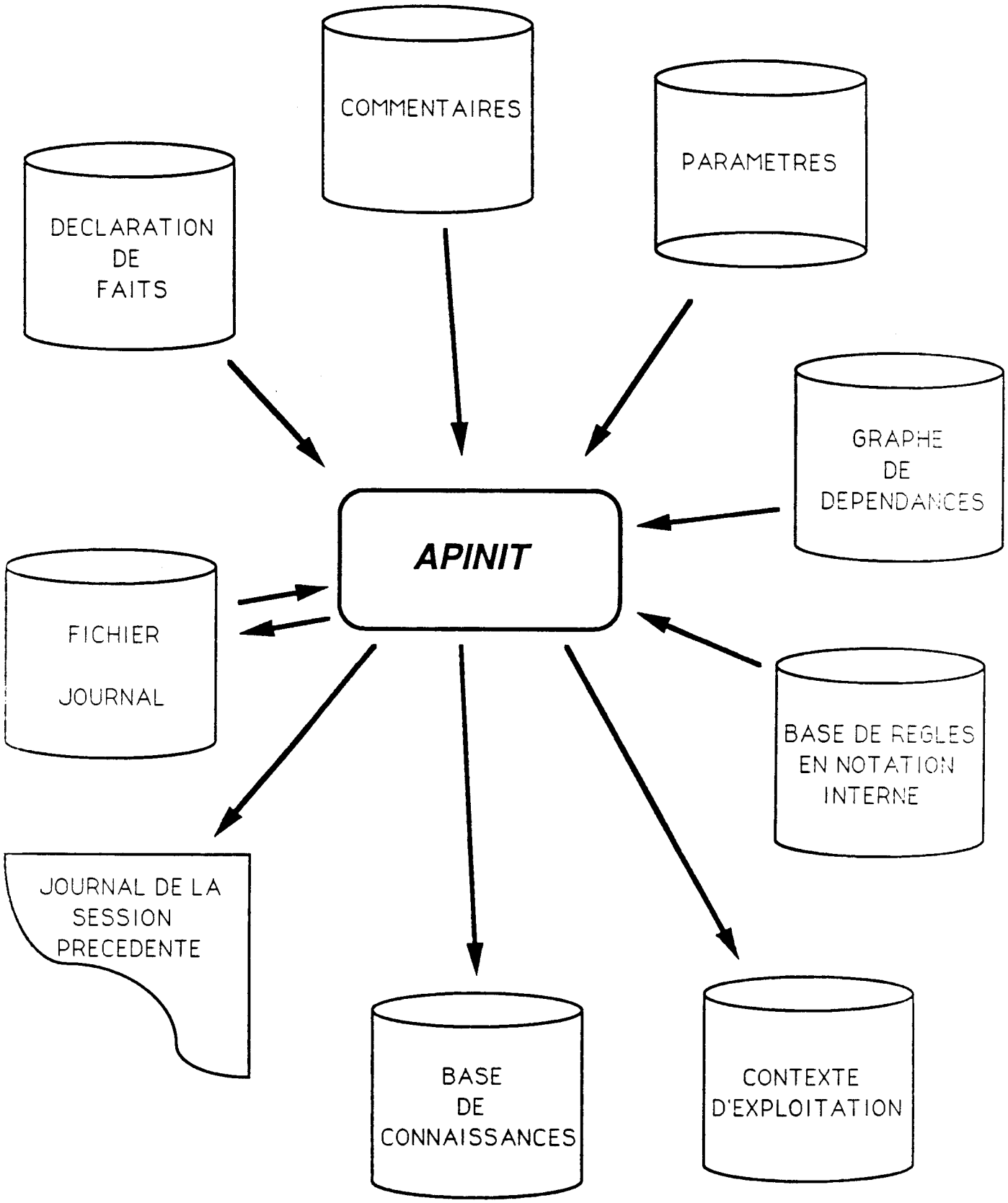


FIGURE IV.3
Initialisation des paramètres

Algorithme de mise à jour de la valeur des paramètres

```
Début
    L1 ← liste des noms de paramètres particuliers
    L2 ← liste des noms des paramètres de la base de connaissances
    L ← concaténation de L1 et L2
    NP ← nombre d'éléments de L
    Partage de l'écran en 3 fenêtres : F1, F2, F3
    Affichage de la liste L dans F1 sous forme de menu déroulant
    N ← 1
    tq l'utilisateur n'a pas appuyé sur ESC faire
        mettre en inversion vidéo la ligne N du menu dans F1
        écrire dans F2 la valeur courante du N-ième paramètre
        écrire dans F3 l'explication du paramètre
        lire au clavier (C)
        si C = " ↑ "et N>1 alors
            N ← N - 1
        sinsi C = " ↓ "et N<NP alors
            N ← N + 1
        sinsi C <> "ESC" alors
            gérer l'entrée au clavier de la nouvelle valeur de paramètre
            dans F2 (avec contrôle de compatibilité avec le type)
        fsi
    ftq

    { le fichier OLDBASE.PRA contiendra toujours le contexte complet
    d'un cycle complet à l'autre }
    mise à zéro du fichier "OLDBASE.TRA"
    Rediriger la sortie vers "OLDBASE.TRA"
    écrire la base de règle en notation interne
    écrire le graphe de dépendances
    écrire la définition des faits
    écrire le nom de la base de connaissances
    écrire la valeur courante de chaque paramètre d'expertise
    écrire tous les commentaires
    fermer le fichier OLDBASE.TRA
    rediriger la sortie vers l'écran

    {traitement des autres paramètres}
    si JOURNAL = 0 alors
        demander le nouveau nom
        copier le fichier journal
    sinsi JOURNAL = 1 alors
        imprimer le journal
    fsi
```

```
Détruire le fichier JOURNAL
Si TRACE DE CONNEXION = "0" alors
| mettre 1 dans le fichier CONNEX.TRA
sinon
| mettre 0 dans le fichier CONNEX.TRA
fsi
mettre à zéro le fichier PILOT.PRA
rediriger la sortie vers PILOT.TRA
si TEMPS DE REPONSE = "0" alors écrire "TEMPREP"
fsi
si EXEC = "0" alors écrire "EXEC"
fsi
si TASK = "0" alors écrire "TASK"
fsi
si CORE = "0" alors écrire "CORE"
fsi
si SSA = "0" alors écrire "SSA"
fsi
si DACQ = "0" alors écrire "DACQ"
fsi
si FILE = "0" alors écrire "FILE"
fsi
si VIDEO = "0" alors écrire "VIDEO"
fsi
fermer le fichier PILOT.TRA
Rediriger la sortie vers l'écran
si IMPRIMANTE = "0" alors
| mettre 1 dans le fichier IMPRIM.TRA
sinon
| mettre 0 dans le fichier IMPRIM.TRA
fsi
si JUSTIFICATION = "0" alors
| mettre 1 dans le fichier JUST.TRA
sinon
| mettre 0
fsi
Mettre TEMPORISATION dans TEMPO.TRA
si IMPRIMANTE = "0" alors
| imprimer l'entête de journal la date, l'heure et la valeur des
| paramètres d'expertise
fsi
Mettre "Ø" dans FROID.TRA
fin
```

Les informations sont échangées entre les modules par l'intermédiaire de fichiers (voir annexe pour un résumé).

4.2) Connexion : APCNX

Ce module est écrit en C.

Il assure la connexion directe entre le micro-ordinateur et le DPS 8. Pour fonctionner, 2 fichiers doivent être présents sur disque en plus de l'exécutable : PILOT.TRA et MSG.TRA.

On a vu précédemment le contenu et l'utilisation de PILOT.TRA. Le programme commence par lire le contenu du fichier PILOT.TRA afin de savoir quels sont les informations qu'il faut aller chercher sur le DPS 8.

Le fichier MSG.TRA contient l'ensemble des messages qui peuvent être échangés entre les 2 ordinateurs. Le protocole de connexion est organisé sous forme d'un automate de mots.

Un message en entrée (PC vers DPS 8) est constitué du caractère ">" suivi d'un numéro d'état et du texte à envoyer.
Un message en sortie (DPS 8 vers PC) est constitué du caractère "<", du numéro d'état correspondant au message en entrée, du numéro d'état suivant suivi du texte à identifier.

Voici un extrait du fichier MSG.TRA (voir annexe pour le fichier complet)

```
>01
<01 02 $*$
<01 99 FUNCTION
<01 01 $$ 0200
<01 03 SNUMB
<01 03 MONITOR

>02 $*$ CN @*@
<02 97 IS NOT KNOWN
<02 01 $$ 0200
<02 02 $$ 1700
<02 01 $$ 0400
<02 99 FUNCTION
<02 03 SNUMB
<02 03 MONITOR

>03 @2@
<03 99 FUNCTION

>04
<04 99 FUNCTION
<04 01 $$ 0200
<04 02 $*$
<04 03 SNUMB
<04 03 MONITOR
```

L'état 1 est l'état d'entrée pour la commande de connexion au TP-MONITOR.

L'état 99 est l'état de sortie pour toute commande.

L'automate a plusieurs points d'entrée. Chacun correspondant à un envoi de message spécial vers le DPS 8.

la figure IV.4 représente l'extrait ci-dessus de l'automate sous forme graphique.

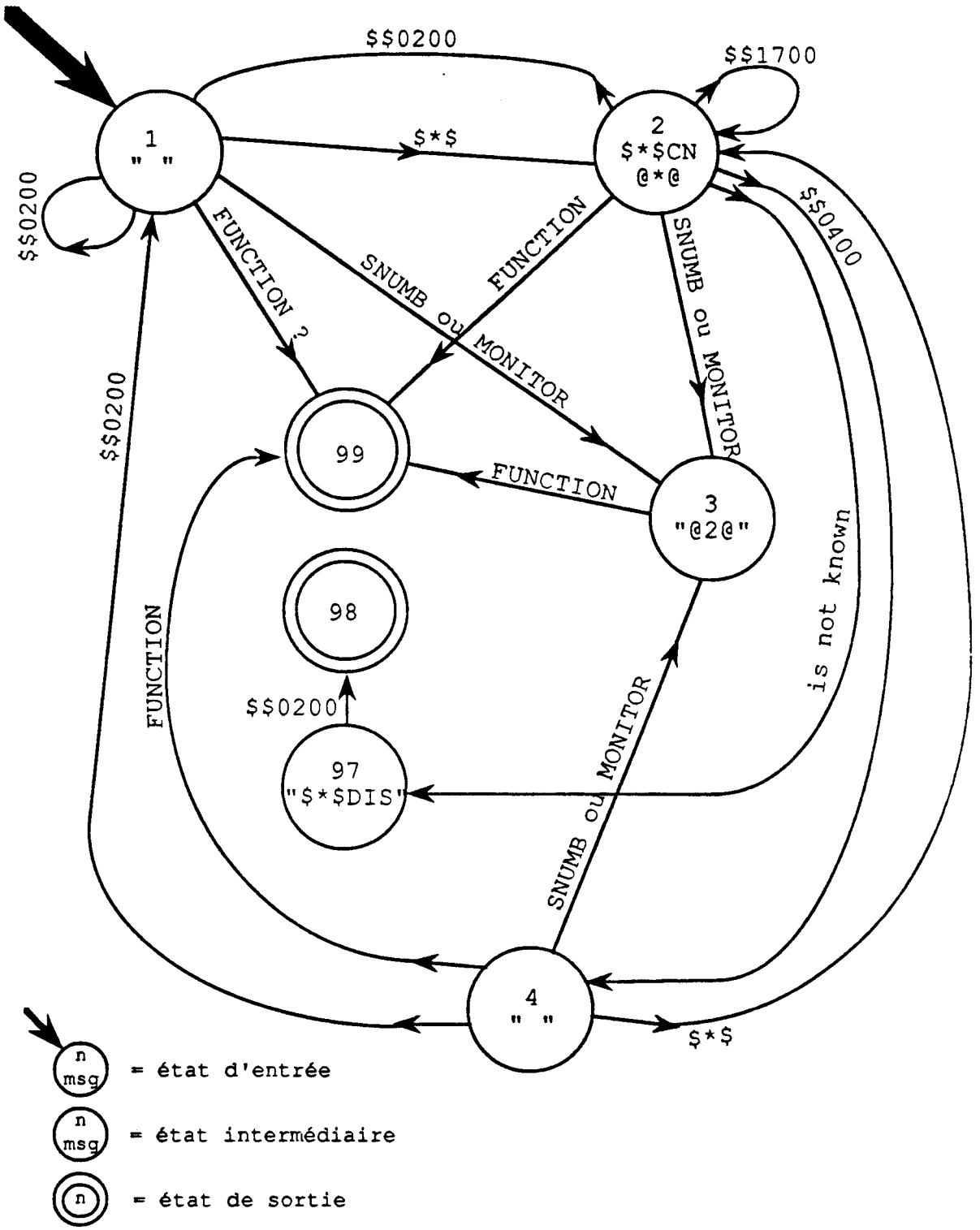


FIGURE IV.4
Extrait de l'automate de connexion

L'entrée dans l'automate pour demander une commande se fait de manière simple :

- Mettre le numéro d'état d'entrée dans une variable globale
- Appeler la procédure de gestion des états

Voir figure IV.5

Algorithme de gestion des états

```
Début
    COMPTEUR ← 500
    tq numéro d'état courant <> 99 et COMPTEUR > 0  faire
        Envoyer le message vers le DPS 8
        COMP_LIT ← 20
        tq COMP_LIT > 0 et COMPTEUR > 0  faire
            LIRE le message venant du DPS 8
            COMPTEUR ← COMPTEUR - 1
            Si le message reçu est connu dans l'automate  alors
                changer le numéro d'état courant
                COMPT_LIT ← 0
            sinon
                COMPT_LIT ← COMPT_LIT - 1
            fin
        ftq
    ftq
fin
```

On a vu au chapitre I comment sont fournies les informations dans TPMON et VIDEO, par écrans défilants. Par défaut, le temps entre 2 rafraîchissements est de 10 secondes. Un message le positionne à 3 secondes pour APSYST.

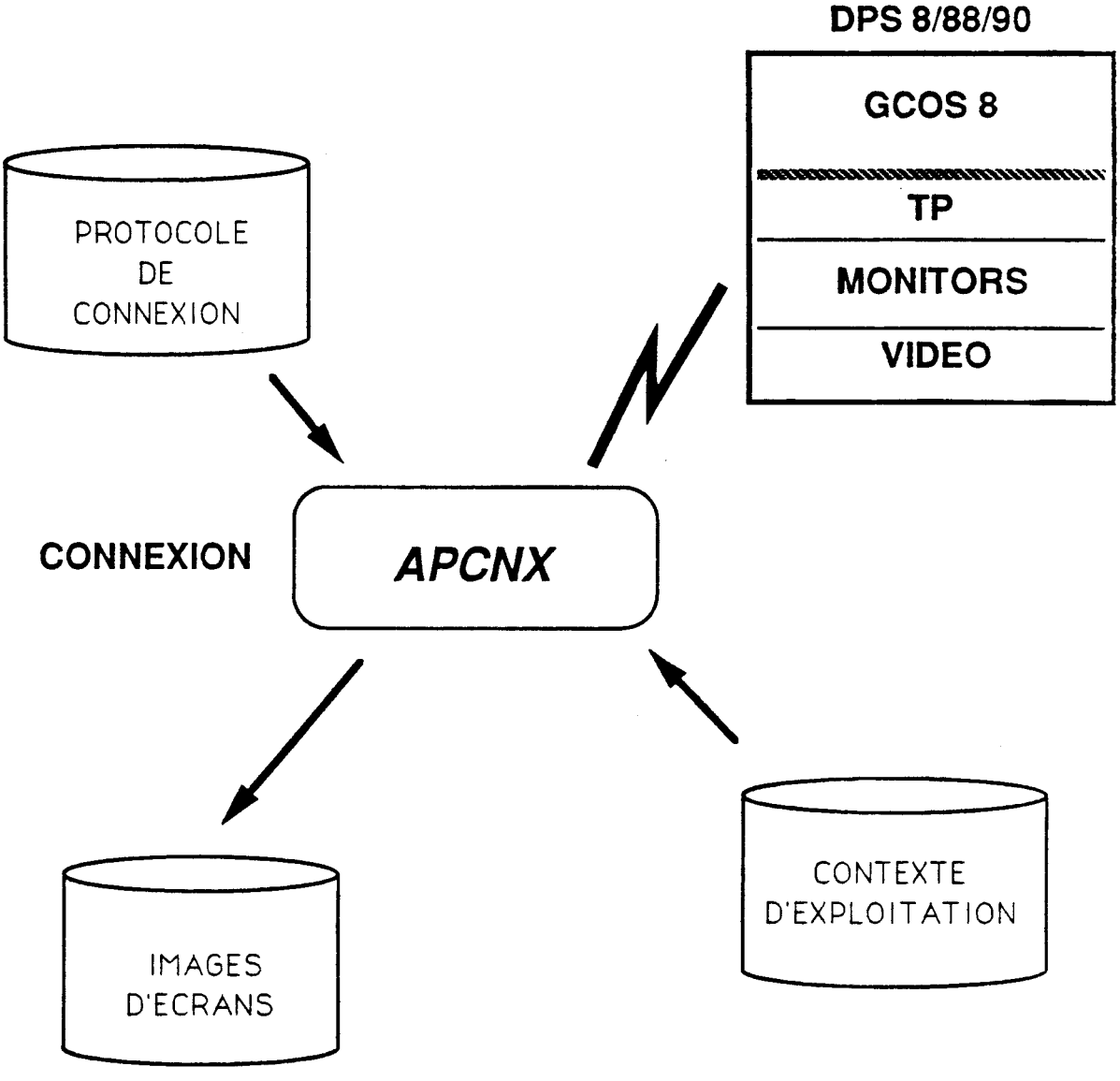


FIGURE IV.5
Module de connexion

L'algorithme précédent n'est pas valable pour effectuer la capture des écrans.

```
Début
    . Lire l'écran envoyé par le DPS 8
    . H ← heure contenue dans la première ligne de l'écran
    . Enlever de l'écran les caractères de contrôle spécifiques au
    terminaux DPS 8
    . Ecrire l'écran sur fichier
    . NB_ECRAN ← 5
    tq NB_ECRAN > 0 faire
        lire l'écran envoyé par le DPS 8
        HH ← heure contenue dans la première ligne de l'écran
        si HH > H alors
            . enlever de l'écran tous les caractères de contrôle
            . écrire l'écran sur fichier
            . H ← HH
            . NB_ECRAN ← NB_ECRAN - 1
        fsi
    ftq
fin
```

Six écrans consécutifs de chaque type sont capturés.
Pour le calcul du temps de réponse, l'heure est relevée sur le micro-ordinateur avant et après l'échange de message et la différence est calculée ensuite. L'algorithme utilisé est celui de la gestion des états de l'automate.

Les écrans sont sauvegardés sur des fichiers différents :

- ECRAN.TSK
- ECRAN.SSA
- ECRAN.FIL
- ECRAN.COR
- ECRAN.DAC
- ECRAN.EXC
- ECRAN.VID

Plus

- TEMPS.TRA pour le temps de réponse.

4.3) Extraction : APEXT

Ce module est écrit en C.

La forme des écrans tels qu'ils sont sauvés par APCNX ne sont pas directement réutilisables par le moteur d'inférences.
Tous les écrans ayant une forme différente, il est nécessaire d'avoir une procédure spécifique à chacun d'eux.

Afin d'éviter des calculs inutiles, ces procédures sont appelées suivant le contenu du fichier PILOT.TRA.

Tous les faits seront présentés au moteur d'inférences à partir du fichier NOUFAIT.TRA, qui est tout d'abord initialisé à vide.

Voici le détail des procédures les plus importantes :

Ecran EXEC de TPMON :

C'est l'écran le plus chargé d'informations, à savoir 61 par écran. Il y a parfois plusieurs valeur à prendre dans une même ligne. Toutes ces valeurs représentent des faits de type réels sans variable. Pour réaliser cette extraction d'informations, on a recours à une structure de données spécifique à cet écran.:

un tableau T d'enregistrements de VARIABLE

Le type VARIABLE étant défini ainsi :

NUMLIGNE :	entier - numéro de ligne dans l'écran EXEC
NOM :	chaîne de caractères - nom de fait à présenter au moteur d'inférences
G :	entier - colonne de gauche de la valeur
D :	entier - colonne de droite de la valeur
VALEUR :	réel - valeur de fait.

Le tableau est trié par ordre croissant sur les numéros de ligne.

Voir figure IV.6

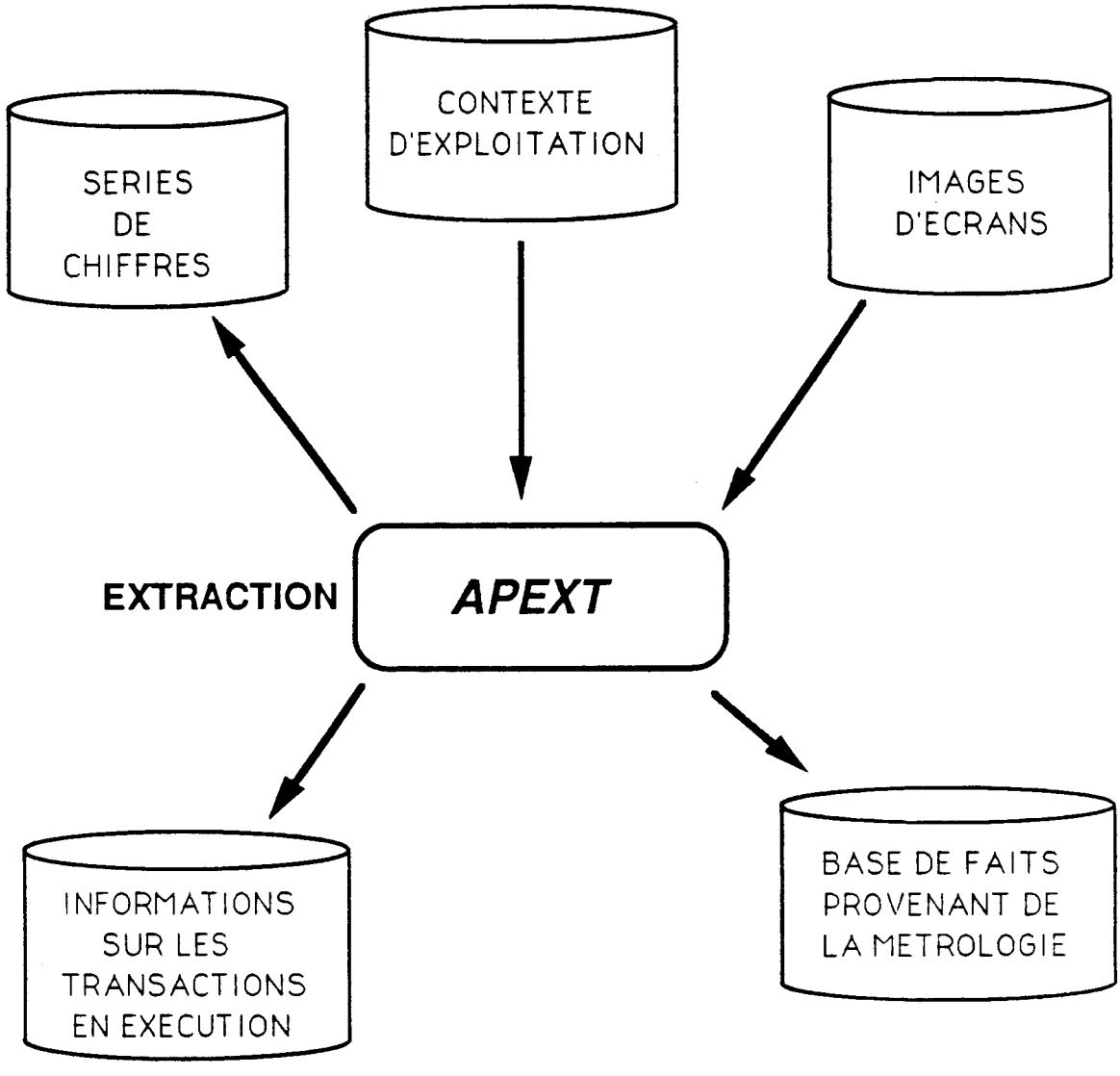


FIGURE IV.6
Module d'extraction

Algorithme de la procédure d'extraction de l'écran EXEC.

LIGNE est une chaîne de caractères contenant la ligne courante.

```
Début
    NB_VAR ← 61 {nombre de faits à prendre}
    pour I de 1 à 6  faire
        positionnement en début de la première ligne d'écran EXEC
        sur le fichier ECRAN.EXEC
        LIGNE ← lire_une_ligne (ECRAN_EXEC)
        NBL ← 1
        pour J de 1 à NB_VAR  faire
            tq T[J].NUMLIGNE <> NBL  faire
                LIGNE ← lire_une_ligne (ECRAN_EXEC)
                NBL ← NBL + 1
            ftq
            T[J].VALEUR ← T[J].VALEUR + valeur comprise entre
            LIGNE [T [J].G] et LIGNE [T [J].D]
        fpour
    fpour
    pour J de 1 à NB_VAR  faire
        t[J].VALEUR ← T[J].VALEUR / 6
        Ecrire à la suite de NOUFAIT.TRA : (T[J].NOM, "=" ,
        T(J).VALEUR)
    fpour
fin
```

Ecran FILE de TPMON

Le traitement de cet écran n'a rien de comparable avec le précédent. L'écran FILE présente des valeurs par fichier. Le nombre de fichiers étant supérieur au nombre de lignes disponibles par écran, et les informations présentées à raison d'un fichier par ligne, la lecture des informations s'étale sur plusieurs écrans FILE. Pour chaque fichiers, 7 informations sont fournies par ligne. Tous les faits de cet écran sont de type réel avec variable.

L'identificateur d'un fichier du TP est composé de 2 caractères. Le nom du premier fichier lu est conservé en mémoire. Les noms des fichiers suivants sont tous comparés avec le premier. La lecture s'effectue sur plusieurs écrans successifs. L'arrêt se fait quand le premier nom de fichier est retrouvé.

Ecran SSA de TPMON

L'écran SSA est divisé en deux parties.
Le haut de l'écran comporte des faits réels sans variable. Ils sont au nombre de 13. Leur lecture se fait par la prise des caractères compris entre un numéro de colonne de gauche et un numéro de colonne de droite.
Ils sont écrit au fur et à mesure à la suite du fichier NOUFAIT.TRA.

Le bas de l'écran présente des informations fichier par fichier. Il y a ici une information par fichier et 5 fichiers par ligne. Il s'agit donc d'un fait avec variable.

Les noms sont gérés par le moteur d'inférences (voir IV.4.4 pour l'explication de leur gestion). Un seul nom de fichier est connu du module APEXT, il s'agit du fichier SWAP du TP dont le nom est SW. L'information de l'écran SSA concernant le fichier SW est écrite sous forme de fait sans variable.

Ecran TASK de TPMON :

Au dire de certains experts, cet écran est le plus important !
C'est aussi celui qui donne le plus de travail.
Voir l'exemple d'écran TASK donné au Chapitre V.
Les lignes prises en compte sont les lignes à partir de la n°3.
Le nombre de lignes par écran est égal au nombre de couloirs ouverts pour le TP.
La mesure est faite sur 6 écrans consécutifs.
A chaque écran, on prend l'heure d'émission.

A chaque transaction du TP est associé un numéro de série (SERIAL NUMBER) attribué dynamiquement par l'exécutif du TP.

Pour chaque SERIAL NUMBER on calcule :

- Le temps pendant lequel la transaction est restée dans un couloir dans un même numéro de phase dans un état autre que *COMPLETE*
- Le numéro de phase en question
- Le nombre de phases exécutées pendant les 6 écrans.

Entre 2 écrans, l'état des couloirs change. Une transaction peut passer d'un couloir à l'autre entre 2 rafraîchissement d'écran. Une même transaction peut être présente dans plusieurs couloirs en même temps, c'est à dire dans le même écran. La seule contrainte est que la transaction ne peut être dans un état différent de *COMPLETE* qu'une seule fois par écran.

Le tri des transactions est réalisé de manière croissante avec
clé primaire : SERIAL NUMBER de la transaction
clé secondaire : PHASE de la transaction
clé tertiaire : HEURE de l'écran
clé quaternaire : Etat des transactions (COMPLET > tous les autres)

Ensuite pour chaque SERIAL NUMBER, on écrit sur le fichier NOUFAIT.TRA sous forme de fait réel variable :

- Le numéro de la phase dans laquelle la transaction est restée le plus longtemps en mémoire (état NOT COMPLETE)
- Le temps pendant lequel elle est restée en mémoire
- Le nombre de phases exécutées (différence entre la valeur maximale et la minimale)
- Les ressources allouées et utilisées
- Le temps processeur consommé
- La place mémoire utilisée

Ecran VIDEO :

Comme pour l'écran EXEC de TPMON, chaque valeur de type sans variable est prise 6 fois de suite dans les écrans de VIDEO. Ensuite, on calcule la moyenne avant de l'écrire sur NOUFAIT.TRA. Pour les faits avec les variables, comme pour l'écran FILE les valeurs s'étendent sur plusieurs écrans consécutifs. Mais il n'y a pas de tests d'arrêt à cause des faits sans variable.

Quatre faits en plus d'être stockés sur NOUFAIT.TRA sont écrits, associés à l'heure de l'écran, sur des fichiers particuliers dans le but de construire des séries qui seront présentées sous forme d'histogrammes (voir IV.4.8 et figure II.15)

A partir de l'écran **SSA** de TPMON
- Consommation des couloirs d'accès au fichier du TP:
  ==> Fichier TIO.JRN
- Consommation processeur du TP
  ==> Fichier.TCP.JRN

A partir de l'écran **TASK** de TPMON
- Nombre de transactions simultanément en mémoire
  ==> Fichier CTLTSK.JRN

A partir de **VIDEO**
- Pourcentage de consommation processeur de TP sur le DPS 8
  ==> Fichier CPDUTP.JRN

Avant de terminer le module APEXT, le micro-ordinateur prend l'heure interne du micro-ordinateur et l'écrit sur le fichier FICHEURE.
C'est cette heure qui sera écrite avec les commentaires dans le but de faire des justifications.

4.4) Moteur d'inférences : APUNFER

Ce module est réalisé en PROLOG.

On distingue 5 phases dans ce module :

1- Lecture du contexte correspondant au cycle précédent
2- Insertion des nouveaux faits dans la base de connaissances
3- Inférence
4- Sauvegarde du contexte
5- Préparation de l'enchaînement.

Voir figure IV.7

Chacun de ces 5 points sera détaillé dans les paragraphes suivants.

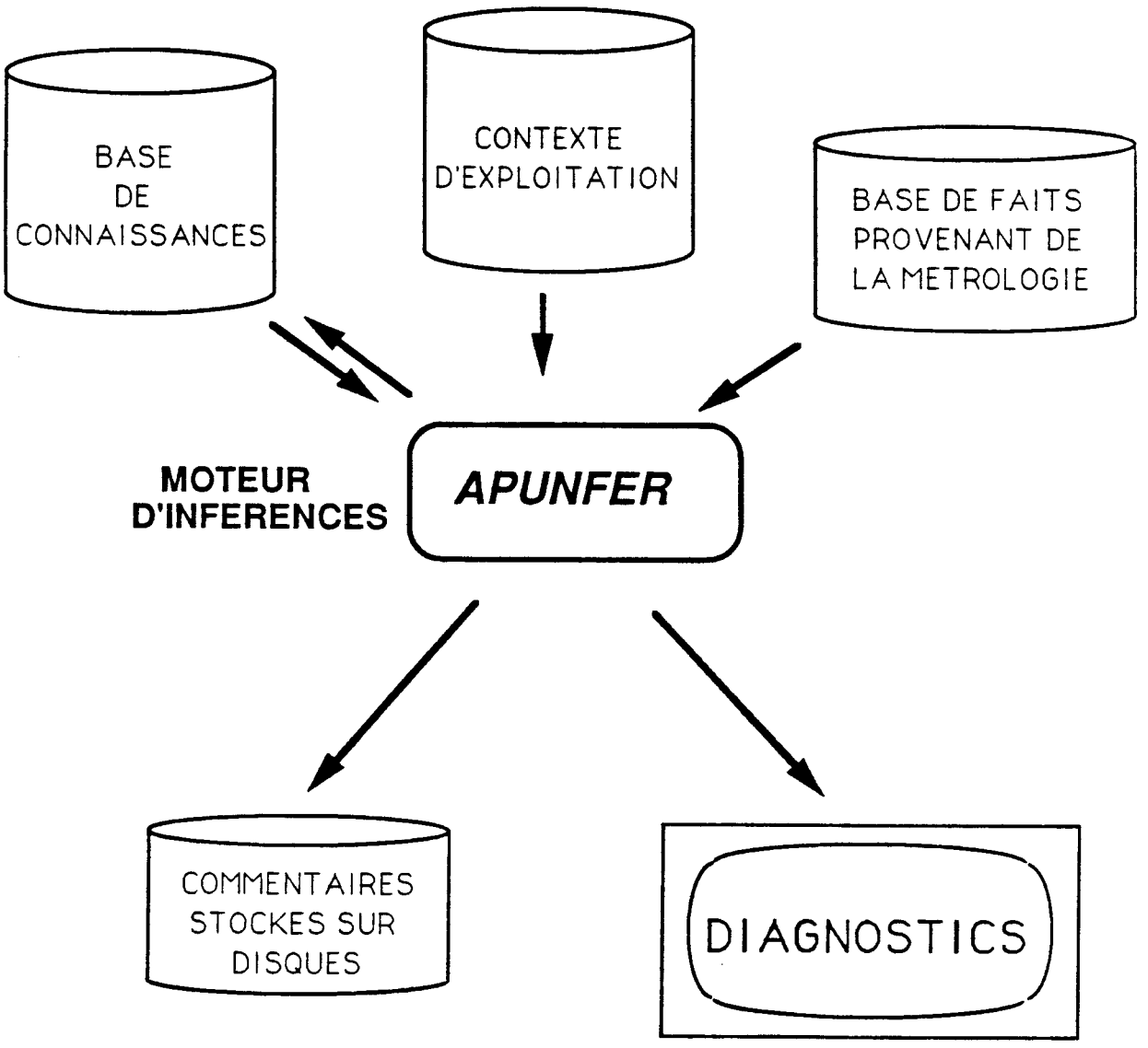


FIGURE IV.7
Moteur d'inférences

4.4.1) Lecture du contexte

Si c'est la première fois que le moteur d'inférence est appelé durant la session, le contexte est celui créé par le module d'initialisation (voir IV.4.1). Sinon, c'est celui sauvegardé lors de la dernière exécution du module APUNFER.
Il s'agit du fichier OLDBASE.TRA. Un fichier journal (JOURNAL.TRA) est initialisé à zéro, dans le but de pouvoir y copier les diagnostics pour les visualiser lors du module de JOURNALISATION.

4.4.2) Insertion des nouveaux faits

Dans la database, l'ordinateur dispose de la base de faits correspondant au cycle précédent. L'insertion des nouveaux faits va introduire dans la base des incohérences et des contradictions. C'est ici que l'on utilise le graphe de dépendances créée par APLIEN. Il se trouve en database sous forme :

FAIT_A_MODIFIER (F , [F1,F2,...,Fp])
REGLE_ARRIERE_AVANT (F , [R1,R2,...,Rq])

Les faits à introduire dans la base sont contenus dans le fichier NOUFAIT.TRA construit par le module APEXT.

Les faits sont présents en database sous forme :

BOOLEENS :
VRAI (F , N)
FAUX (F , N)
VRAI_VAR (F , I , N)
FAUX_VAR (F , I , N)
où F est un nom de fait (type STRING)
I est la valeur de l'indice (type INTEGER)
N est le numéro de la règle d'origine (type INTEGER).

REELS :
VA (F , V , N)
VA_VAR (F , I , V , N)
où F est un nom de fait (type STRING)
I est la valeur de l'indice (type INTEGER)
V est la valeur du fait (type INTEGER)
N est le numéro de la règle d'origine (type INTEGER).

Dans le fichier NOUFAIT.TRA, les faits sont présentés sous forme :

BOOLEENS

F
NON F
F(N)
NON F(N)

où F est un nom de fait (type STRING)
N est un nom d'objet représenté par un indice (type STRING)

REELS :

F = n F(N) = n
F = -n F(N) = -n

où F est un nom de fait (type STRING)
N est un nom d'objet représenté par un indice (type STRING)
n est un nombre (type REEL).

Dans la suite nous parlerons des prédicats database prolog REGALANCER et FAITANNULER, ce qui correspond aux définitions données au chapitre trois : REGLE_ARRIERE_AVANT et FAIT_AVANT.
On remarque la différence entre la représentation database et la présentation dans NOUFAIT.TRA : *les indices.*

Le moteur d'inférences ne travaille que sur des valeurs entières d'indice (on verra l'intérêt dans le prochain algorithme).
Pour cela, il associe dynamiquement un entier à chaque objet représenté par un indice dans les règles.

Les associations sous forme :

ASSOCIATION (nom_indice, entier, nom objet)

Le système retourne, pour chaque fait, l'indice qui lui est associé par l'intermédiaire d'une règle où il apparaît.
Chaque indice possède un domaine de variation correspondant au nombre d'objets qui lui sont associés.

Exemple :

A la lecture sur NOUFAIT.TRA de la ligne :

F (OBJ1) = n

Le système recherche d'abord une règle ou F apparaît pour trouver l'indice associé : IND.

Ensuite, il cherche dans la database s'il existe une association de la forme ASSOCIATION (IND, X , OBJ1) en essayant d'unifier cette expression avec les éléments de la database.

L'unification peut donner 2 résultats :

- **ECHEC** : ceci signifie que le système ne connaît pas encore l'objet OBJ1 associé à IND
- **Affectation d'un entier à X** : cela signifie que le système à déjà rencontré le symbole OBJ1 associé à IND. Le numéro d'ordre de la première rencontre de OBJ1 associé à IND.sera affecté à X.

En supposant que le système connaît "s" associations avec l'indice IND, dans le premier cas, il va faire des ajouts en database:

DOMAINE (IND, s + 1)
ASSOCIATION (IND, S + 1, OBJ1)
VA_VAR (F, S + 1, n)

Dans le deuxième cas, le système va faire un seul ajout en database.
En supposant que le résultat de l'unification est :

ASSOCIATION (IND, OBJ1, r).

L'ajout sera :

VA_VAR (F, r, n).

Les contradictions sont évitées en déterminant un ensemble de faits à sortir de la base avant de réaliser l'inférence. L'ensemble des règles susceptibles de modifier la valeur de certains faits en database est déterminé.

Voici l'algorithme de lecture et d'introduction des faits avec l'utilisation du graphe de dépendances :

```
Début
    Enlever de la base de connaissances tous les "FAITANNULER"
    Enlever de la base de connaissances tous les REGALANCER
    Ouvrir le fichier NONFAIT.TRA en lecture
    tq non (fin de fichier (NOUFAIT.TRA))
      lire une ligne du fichier NOUFAIT.TRA {nouvelle valeur de F}
      suivant le type du fait F, mettre sa nouvelle valeur en database
      prendre le numéro d'indice du fait par ASSOCIATION (_, IF, _)
      unifier avec un élément de la database FAIT_A_MODIFIER (F, LF)
      pour chaque élément FF de LF  faire
        si FF est un fait sans variable  alors
          si l'unification avec un élément de la database de
          FAIT_ANNULER (FF) donne un échec alors
            déclarer FAITANNULER (FF)
          fsi
        sinon
          si l'unification avec un élément de la database de
          FAITANNULER_VAR (FF) donne un échec  alors
            déclarer FAIT_ANNULER_VAR (FF , IF)
          fsi
        fsi
      fpour
      unifier avec un élément de la database
      REGLE_ARRIERE_AVANT(F,LR)
      pour chaque élément R de LR  faire
        si F est un fait sans variable  alors
          si R est une règle sans variable  alors
            si l'unification de REGALANCER (R,1) donne un échec
              alors déclarer REGALANCER (R , A
            fsi
          sinon {R est une règle avec variable IND}
            pour I de 1 à DOMAINE (IND)  faire
                si l'unification de REGALANCER (R , I) donne un
                échec alors déclarer REGALANCER (R , I)
                fsi
            fpour
          fsi
        sinon
          si l'unification de REGALANCER (R , IF) donne un échec
            alors
            déclarer REGALANCER (R , IF)
          fsi
        fsi
      fpour
    ftq
fin
```

4.4.3) Inférence

L'inférence va permettre d'effectuer la saturation de la base de faits tout en assurant sa consistance.

L'algorithme utilisé est :

```
Début
    déclarer "au moins une règle utilisée"
    tq au moins une règle utilisée faire
        enlever de la database "au moins une règle utilisée"
        pour chaque FAITANNULER (F) faire
            si F est un fait sans variable alors
                enlever le fait F de la database
            sinon
                prendre l'indice IND associé à F
                pour I de 1 à IND faire
                    enlever toutes les valeurs de F associé à I
                    {suivant le type de F}
                fpour
            fsi
        fpour
        pour chaque FAITANNULER_VAR (F , I) faire
            enlever la valeur de F associé à I
            {suivant le type de F}
        fpour
        pour chaque REGALANCER (R , I) faire
            si non (utilisée (r , i)) alors
                si la partie condition de R est vraie alors
                    déclarer utilisée (R ,I)
                    déclencher la conclusion
                    déclarer "au moins une règle utilisée"
                fsi
            fsi
        fpour
    ftq
fin
```

Evaluation des parties condition pour une règle :
REG (n , "I" , [...] , [...])

CONDITION	TEST
est_il_vrai (F)	unification avec VRAI (F , _)
est_il_vrai_var (F)	unification avec VRAI_VAR (F , P, _)
est_il_faux (F)	unification avec FAUX (F , _)
est_il_faux_var (F)	unification avec FAUX_VAR (F , P , _)
associe (I , NOM)	unification avec ASSOCIATION (I,P,NOM)
connu (F)	unification avec VA (F , _)
connu_var (F)	unification avec VA_VAR (F,I,_,_)
est_ce_que (comp,F,G)	unification avec VA (F,X,_) et VA (G,Y,_) puis comparaison de X et Y
est_ce_que_var_1 (comp,F,G)	unification avec VA_VAR (F,I,X,_) et VA (G,Y,_) puis comparaison de X et Y
est_ce_que_var_2 (comp,F,G)	unification avec VA (F,X,_) et VA_VAR(G,I,Y,_) puis comparaison de X et Y
est_ce_que_var_3 (comp,F,G)	unification avec VA_VAR (F,I,X,_) et VA_VAR (G,I,Y,_) puis comparaison de X et Y

Déclenchement des conclusions pour une règle :
REG (n , "I" , [...] , [...]) et REGALANCER (n , p).

CONCLUSION	ACTION
doncvrai (F)	assert (vrai (F , n))
doncvrai_var (F)	assert (vrai_var (F , n))
doncfaux (F)	assert (faux (F , n))
doncfaux_var (F)	assert (faux_var (F , n))
callcom (N)	transformation du commentaire (*)
affect (F , post-fix (S))	R ← calcul (S) (**) puis assert (va(F,R,n))
affect_var (F , post-fix (S))	R ← calcul (S) puis assert (va_var(F,I,R,n))

(*) Transformation du commentaire

Les commentaires sont en database sous forme :

COMAF (I , T , LV)
I est le numéro du commentaire
T est le texte du commentaire contenant des #
LV est la liste de variables qui remplaceront les #.

L'algorithme de transformation est :

en supposant que la database contienne
REG(n , "I" , [. . .] , [. . . , callcom (N) , . . .])
REGALANCER (n , p)

```
Début
    unifier avec un élément de la database COMAF (N , T , LV)
    tq non fin de la chaîne (T)  faire
        prendre le caractère suivant CAR de T
        si CAR = "#"  alors
            prendre la variable suivante V de LV
            si VRAI (V)  alors écrire "VRAI"
            sinsi VRAI_VAR (V , p)  alors écrire "VRAI"
            sinsi FAUX (V)  alors écrire "FAUX"
            sinsi FAUX_VAR (V , p)  alors écrire "FAUX"
            sinsi VA (V , X , _)  alors écrire (X)
            sinsi VA_VAR (V , p , Y , _)  alors écrire (Y)
            sinsi ASSOCIATION (V , p , IND)  alors écrire (IND)
            fsi
        sinon
            écrire CAR
        fsi
    ftq
fin
```

A noter qu'un indice est remplacé par l'objet qu'il représente

(**) Calcul en notation post-fixée
S est une expression en notation post-fixée
On commence par remplacer chaque nom de fait de S par sa valeur. On obtient une liste L contenant uniquement des nombres et des opérateurs.

L'algorithme de calcul est :

```
Début
    LL ← liste vide
    tq L non vide faire
        prendre le premier élément L1 de L
        si L1 est un opérateur OP alors
            enlever les 2 premiers éléments LL1 et LL2 de LL
            RES ← LL2 OP LL1
            ajouter RES entête de LL
        sinon
            enlever L1 de L
            ajouter L1 en tête de LL
        fsi
    ftq
fin
```

4.4.4) Sauvegarde du contexte

Tous les commentaires ont été mis sur fichier pendant l'inférence après transformations pour être revus en phase de journalisation.
La sauvegarde du contexte consiste en l'écriture de tous les éléments qui sont en database sur le fichier OLDBASE.TRA.

4.4.5) Exemple de gestion de la base de connaissances

Exemple de base de règles :

R1. si taille_fichier(I)=connu et nb_acces(I)=connu
alors taux_acces(I):=nb_acces(I)/taille_fichier(I) ;

R2. si taux_acces(I)>10
alors fichier_trop_sollicité(I) ;

R3. si taille_mem(J)=connu et taille_machine=connu
alors taux_mem(J):=taille_mem(J)/taille_machine ;

R4. si taux_mem(J)>=0.5
alors trop_consom(J) ;

Cette base de règles contient
2 indices (I et J) dans 4 règles

Extrait du graphe de dépendances

FAIT_A_MODIFIER (taille_fichier , [taux_acces, fichier_trop_sollicité])
FAIT_A_MODIFIER (nb_acces , [taux_acces , fichier_trop_sollicité])
FAIT_A_MODIFIER (taille_mem , [taux_mem , trop_consom])
FAIT_A_MODIFIER (taille_machine , [taux_mem , trop_consom])
REGLE_ARRIERE_AVANT (taille_fichier , [1 , 2])
REGLE_ARRIERE_AVANT (nb_acces , [1 , 2])
REGLE_ARRIERE_AVANT (taille_mem , [3 , 4])
REGLE_ARRIERE_AVANT (taille_machine , [3 , 4])

Exemple de base de faits : contenu dans NOUFAIT.TRA

taille_fichier (F1)=1000
taille_fichier (F2)=5000
nb_acces(F1)=40000
nb_acces(F2)=10000
taille_mem(JOB1)=500
taille_mem(JOB2)=1000
taille_mem(JOB3)=700
taille_machine=4000

En database, les associations sont de la forme :
ASSOCIATION (nom_indice , nom , entier)

Ici : **nom_indice** peut prendre les valeurs **I** et **J**.
nom peut prendre les valeurs **F1, F2, JOB1, JOB2** et **JOB3**.

Le système retrouve pour chaque fait l'indice qui lui est associé.

Ici : à *"taille_fichier"* correspond l'indice **I**
à *"nb_acces"* correspond l'indice **I**
à *"taille_mem"* correspond l'indice **J**
à *"taille_machine"* correspond l'indice **sans**

Avant la première lecture du fichier contenant les faits à introduire dans la base, le système initialise les domaines de tous les indices à 0.

Avant la lecture de NOUFAIT.TRA :

domaine (I , 0)
domaine (J , 0)

vrai (_ , _) : échec
vrai_var (_ , _ , _) : échec
faux (_ , _) : échec
faux_var (_ , _ , _) : échec
va (_ , _ , _) : échec
va_var (_ , _ , _ , _) : échec

faitannuler (_) : échec
faitannuler_var (_ , _) : échec
regalancer (_ , _) : échec

Lecture du fichier NOUFAIT.TRA

Lecture de *taille_fichier (F1) = 1000*
Unification de ASSOCIATION ("I" , "F1" , X) : *échec*
Création de l'association :
ASSERT (ASSOCIATION ("I" , "F1" , 1))
Redéfinition du domaine
ASSERT (DOMAINE ("I" , 1))
Déclaration de la valeur d'un fait
ASSERT (VA_VAR (taille_fichier , 1 , 1000 , 0))
Unification de FAIT_A_MODIFIER (taille_fichier , L) :
Résultat : L = [taux_acces , fichier_trop_sollicité]
Déclaration
ASSERT (FAITANNULER_VAR (taux_acces , 1))
ASSERT (FAITANNULER_VAR (fichier_trop_sollicité , 1))
Unification de REGLE_ARRIERE_AVANT (taille_fichier , LR) :
Résultat : LR [1 , 2]
Déclaration
ASSERT (REGALANCER (1 , 1))
ASSERT (REGALANCER (2 , 1))

Lecture de *taille_fichier (F2) = 5000*
Unification de ASSOCIATION ("I" , "F2" , X) : *échec*
ASSERT (ASSOCIATION ("I" , "F2" , 2))
ASSERT (DOMAINE ("I" , 2))
ASSERT (VA_VAR (taille_fichier , 2 , 5000 , 0))
Unification de FAIT_A_MODIFIER (taille_fichier , L) :
Résultat L = [taux_acces , fichier_trop_sollicité]
Déclaration
ASSERT (FAITANNULER_VAR (taux_acces , 2))
ASSERT (FAITANNULER_VAR (fichier_trop_sollicité , 2))
Unification de REGLE_ARRIERE_AVANT (taille_fichier , LR) :
Résultat LR [1,2]
Déclaration
ASSERT (REGALANCER (1 , 2))
ASSERT (REGALANCER (2 , 2))

Lecture de nb_acces (F1) = 40000
Unification de **ASSERT ("Z" , "F1" , 1)** : ***résultat X=1***
ASSERT (VA_VART (nb_acces , 1 , 40000 , 0))
Unification de **FAIT_A_MODIFIER (nb_acces , L)** :
Résultat : [taux_acces , fichier_trop_sollicité]
Unification de **REGLE_ARRIERE_AVANT (nb_acces , LR)** :
Résultat [1 , 2]

Lecture de *nb_acces (F2) = 1000*
Unification de **ASSERT ("I" , "F2" , X)** : ***résultat X=2***
ASSERT (VA_VAR (nb_acces , 1 , 40000 , 0))
Unification de FAIT_A_MODIFIER (nb_acces , L) :
Résultat [taux_acces , fichier_trop_sollicité]
Unification de **REGLE_ARRIERE_AVANT (nb_acces , LR)** :
Résultat [1 , 2]

Lecture de *taille_mem (JOB1) = 500*
Unification de **ASSOCIATION ("J" , "JOB1" , X)** : ***échec***
ASSERT (ASSOCIATION ("J" , "JOB1" , 1))
ASSERT (DOMAINE ("J" , 1))
ASSERT (VA_VAR (taille_mem , 1 , 500 , 0))
Unification de **FAIT_A_MODIFIER (taille_mem , L)** :
Résultat L = [taux_mem , trop_consom]
ASSERT (FAIT_ANNULER_VAR (taux_mem , 1))
ASSERT (FAIT_ANNULER_VAR (trop_consom , 1)
Unification de ***REGLE_ARRIERE_AVANT (taille_mem , LR)*** :
Résultat LR [3,4]
ASSERT (REGALANCER (3 , 1))
ASSERT (REGALANCER (4 , 1))

Lecture de ***taille_mem (JOB2) = 1000***
Unification de **ASSOCIATION ("J" , "JOB2" , X)** : ***échec***
ASSERT (ASSOCIATION ("J" , "JOB2" , 2))
ASSERT (DOMAINE ("J" , 2))
ASSERT (VA_VAR (taille_mem , 2 , 1000 , 0))
Unification de **FAIT_A_MODIFIER (taille_mem , L)** :
Résultat L = [taux_mem , trop_consom]
ASSERT (FAIT_ANNULER_VAR (taux_mem , 2))

ASSERT (FAIT_ANNULER_VAR (trop_consom , 2)

Unification de **REGLE_ARRIERE_AVANT (taille_mem , LR) :**

Résultat LR [3,4]

ASSERT (REGALANCER (3 , 2))

ASSERT (REGALANCER (4 , 2))

Lecture de *taille_mem (JOB3) = 700*

Unification de **ASSOCIATION ("J" , "JOB3" , X) :** ***échec***

ASSERT (ASSOCIATION ("J" , "JOB3" , 3))

ASSERT (DOMAINE ("J" , 3))

ASSERT (VA_VAR (taille_mem , 3 , 700 , 0))

Unification de **FAIT_A_MODIFIER (taille_mem , L) :**

Résultat L = [taux_mem , trop_consom]

ASSERT (FAIT_ANNULER_VAR (taux_mem , 3))

ASSERT (FAIT_ANNULER_VAR (trop_consom , 3)

Unification de **REGLE_ARRIERE_AVANT (taille_mem , LR) :**

Résultat LR [3,4]

ASSERT (REGALANCER (3 , 3))

ASSERT (REGALANCER (4 , 3))

Lecture de *taille_machine = 4000*

ASSERT (VA (taille_machine , 40000 , 0))

Unification de **FAIT_A_MODIFIER (taille_machine , 2) :**

Résultat [taux_mem , trop_consom]

ASSERT (FAIT_ANNULER (taux_mem))

ASSERT (FAIT_ANNULER (trop_consom))

Unification de **REGLE_ARRIERE_AVANT (taille_machine , LR) :**

Résultat [3 , 4]

ASSERT (REGALANCER (3 , 1))

ASSERT (REGALANCER (3 , 2))

ASSERT (REGALANCER (3 , 3))

ASSERT (REGALANCER (4 , 1))

ASSERT (REGALANCER (4 , 2))

ASSERT (REGALANCER (4 , 3))

Les nouveaux faits sont tous introduits dans la base.
On enlève les **FAITANNULER**

FAITANNULER_VAR (taux_acces , 1) :

retract (VA_VAR (taux_acces , 1 , _ , _))

FAITANNULER_VAR (taux_acces , 2) :

retract (VA_VAR (taux_acces , 2 , _ , _))

FAITANNULER_VAR (fichier_trop_sollicité , 1) :

```
     retract ( VRAI_VAR (fichier_trop_sollicité , 1 , _ , _) )
FAITANNULER_VAR (fichier_trop_sollicité , 2) :
     retract ( VRAI_VAR (fichier_trop_sollicité , 2 , _ , _) )
FAITANNULER_VAR (taux_mem , 1) :
     retract ( VA_VAR (taux_mem , 1 , _ , _) )
FAITANNULER_VAR (taux_mem , 2) :
     retract ( VA_VAR (taux_mem , 2 , _ , _) )
FAITANNULER_VAR (taux_mem , 3) :
     retract ( VA_VAR (taux_mem , 3 , _ , _) )
FAITANNULER_VAR (trop_consom , 1) :
     retract ( VRAI_VAR (trop_consom , 1 , _ , _) )
FAITANNULER_VAR (trop_consom , 2) :
     retract ( VRAI_VAR (trop_consom , 2 , _ , _) )
FAITANNULER_VAR (trop_consom , 3) :
     retract ( VRAI_VAR (trop_consom , 3 , _ , _) )
FAITANNULER (taux_mem) :
     pour K de 1 à domaine(I) faire
          retract ( VA_VAR (taux_mem , K , _ , _) )
     fpour
FAITANNULER (trop_consom) :
     pour K de 1 à domaine(I) faire
          retract ( VRAI_VAR (trop_consom, K , _) )
     fpour
```

On lance l'inférence pour saturer la base de connaissances avec les règles :

R1 : avec I = 1,2
R2 : avec I = 1,2
R3 : avec I = 1,2,3
R4 : avec I = 1,2,3

Lors de l'inférence, des faits vont être ajoutés à la base :

```
VA_VAR (taux_acces , 1 , 40 , 1)
VA_VAR (taux_acces , 2 , 2 , 1)
VA_VAR (taux_mem , 1 , 0,125 , 3)
VA_VAR (taux_mem , 2 , 0,25 , 3)
VA_VAR (taux_mem , 3 , 0,175 , 3)
VRAI_VAR (fichier_trop_sollicité , 1 , 2)
```

Lors du cycle suivant le fichier NOUFAIT.TRA contient :

taille_mem (JOB2) = 1650
taille_mem (JOB3) = 220
taille_machine = 3000

Lecture de *taille_mem (JOB2) = 1650*

Unification de ASSOCIATION ("J" , "JOB2" , X) : *résultat X=2*
ASSERT (VA_VAR (taille_mem , 2 , 1650 , 0))
Unification de FAIT_A_MODIFIER (taille_mem , L) :
Résultat L = [taux_mem , trop_consom]
ASSERT (FAITANNULER_VAR (taux_mem , 2))
ASSERT (FAITANNULER_VAR (trop_consom , 2))
Unification de REGLE_ARRIERE_AVANT (taille_mem , LR) :
Résultat LR [3,4]
ASSERT (REGALANCER (3 , 2))
ASSERT (REGALANCER (4 , 2))

Lecture de *taille_mem (JOB3) = 220*

Unification de ASSOCIATION ("J" , "JOB3" , X) : *résultat X=3*
ASSERT (VA_VAR (taille_mem , 3 , 220 , 0))
Unification de FAIT_A_MODIFIER (taille_mem , L) :
Résultat L = [taux_mem , trop_consom]
ASSERT (FAITANNULER_VAR (taux_mem , 3))
ASSERT (FAITANNULER_VAR (trop_consom , 3))
Unification de REGLE_ARRIERE_AVANT (taille_mem , LR) :
Résultat LR [3,4]
ASSERT (REGALANCER (3 , 3))
ASSERT (REGALANCER (4 , 3))

Lecture de *taille_machine = 3000*

ASSERT (VA (taille_machine , 3000 , 0))
Unification de FAIT_A_MODIFIER (taille_machine , L) :
Résultat L [taux_mem , trop_consom]
ASSERT (FAITANNULER (taux_mem))
ASSERT (FAITANNULER (trop_consom))
Unification de REGLE_ARRIERE_AVANT (taille_machine , LR) :
Résultat LR [3 , 4]
ASSERT (REGALANCER (3 , 1))
ASSERT (REGALANCER (3 , 2))
ASSERT (REGALANCER (3 , 3))
ASSERT (REGALANCER (4 , 1))
ASSERT (REGALANCER (4 , 2))
ASSERT (REGALANCER (4 , 3))

Ensuite on enlève les FAITANNULER

```
FAITANNULER_VAR (taux_mem , 2) :
     retract (VA_VAR (taux_mem , 2 , _ , _) )
FAITANNULER_VAR (taux_mem , 3) :
     retract (VA_VAR (taux_mem , 3 , _ , _) )
FAITANNULER_VAR (trop_consom , 2) :
     retract (VRAI_VAR (trop_consom , 2 , _ , _) )
FAITANNULER_VAR (trop_consom , 3) :
     retract (VRAI_VAR (trop_consom , 3 , _ , _) )
FAITANNULER (taux_mem) :
     pour K de 1 à domaine (I) faire
          retract ( VA_VAR (taux_mem , K , _ , _) )
     fpour
FAITANNULER (trop_consom) :
     pour K de 1 à domaine (I) faire
          retract ( VRAI_VAR (trop_consom , K , _) )
     fpour
```

On lance l'inférence pour saturer la base de connaissances avec les règles :

R3 : avec J = 1,2,3
R4 : avec J = 1,2,3

Lors de l'inférence des faits sont ajoutés dans la base :

```
VA_VAR (taux_mem , 1 , 0,167 , 3)
VA_VAR (taux_mem , 2 , 0,55 , 3)
VA_VAR (taux_mem , 3 , 0,073 , 3)
VRAI_VAR (trop_consom , 2 , 4)
```

L'exemple montre bien les avantages apportés par l'utilisation du graphe de dépendances :

- Eviter les contradictions
- Réduire l'ensemble des règles pour l'inférence

4.5) Journalisation et temporisation : APTEMP

Ce module est écrit en TURBO_PASCAL. Son rôle est d'effectuer la journalisation des commentaires générés lors de l'inférence, de permettre de les revoir, de préparer l'archivage de la métrologie, de modifier le cycle de fonctionnement du système et d'attendre avant le suivant.

4.5.1) Journalisation des commentaires

Tout d'abord le module va prendre l'heure et la date qui se trouvent dans le fichier FICHEURE.TRA créé par APCNX.
Toutes les informations sont écrites sur imprimante et sur le journal du système expert (JRNSE.TRA).
Le module écrit d'abord l'heure et la date avant d'écrire tous les commentaires (contenus dans JOURNAL.TRA) séparés par des lignes de tirets. Ensuite, le fichier LIGNETSK.TRA est recopié à la suite des commentaires (voir figure IV.8).

4.5.2) Revisualisation des commentaires

Pendant que le temps de temporisation s'écoule, l'utilisateur a la possibilité de demander à revoir les commentaires. Ils sont affichés les uns après les autres dans l'ordre de génération par le moteur d'inférences. Sur le fichier JOURNAL.TRA, ils sont séparés par des #.

4.5.3) Préparation de l'enchaînement

Archivage : le module va écrire sur le fichier d'enchaînement les commandes DOS de copie de fichier pour archiver la métrologie du fichier PILOT.TRA.
L'heure est sous forme : **JJMMHHMM**

La commande générale est :
COPY ECRAN. JJMMHHMM.**

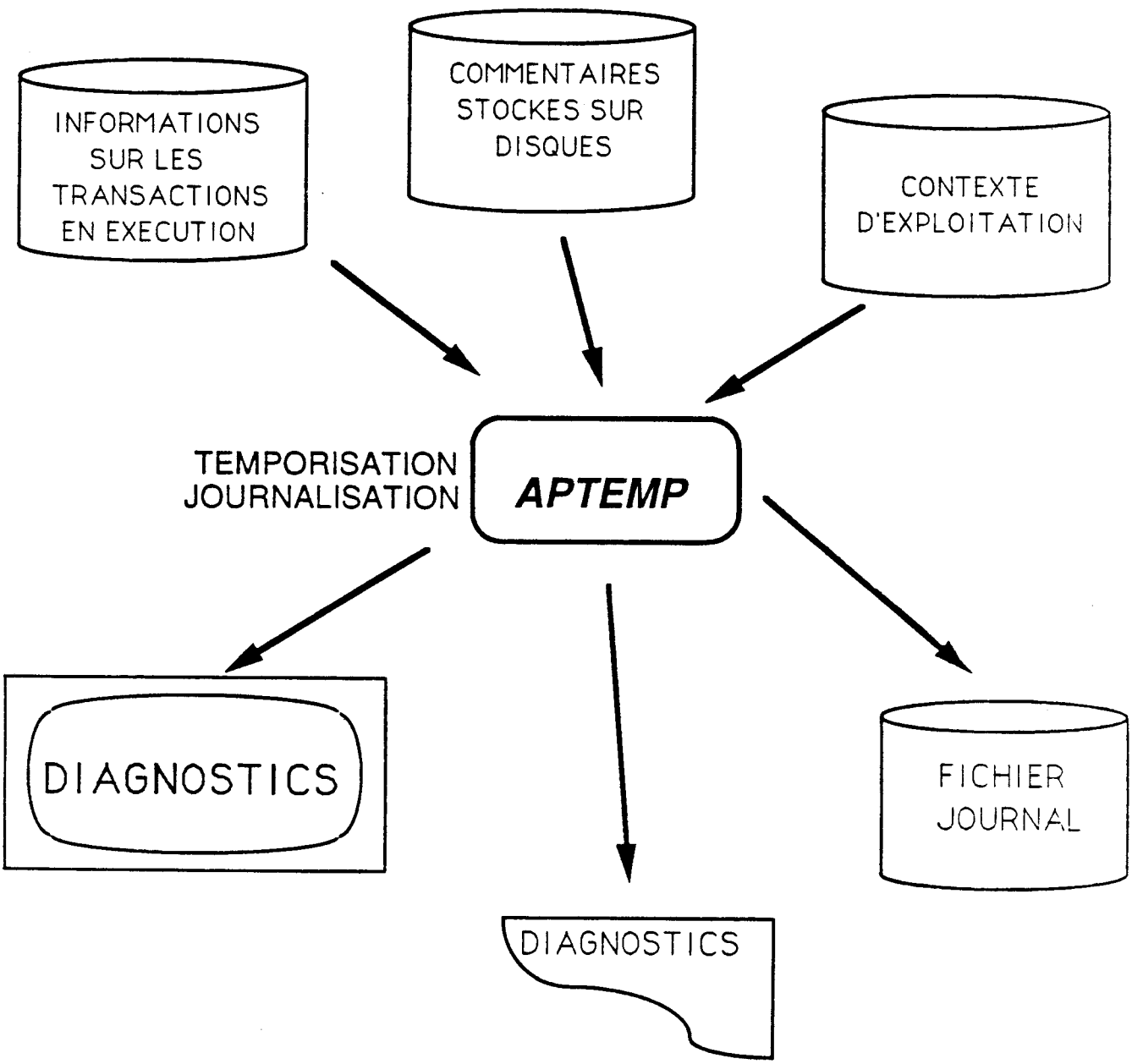


FIGURE IV.8
JOURNALISATION
TEMPORISATION

4.6) Modification de paramètres : APCHANG

Module écrit en PROLOG

Ce module a 2 fonctionnalités :

- Ajustement de la valeur des paramètres
- Simulation

Dans les 2 cas, il s'agit de donner des valeurs nouvelles à des faits de type réel sans variable.
La méthode utilisée pour entrer des valeurs est la même que pour le module d'initialisation : 3 fenêtres sur l'écran. La première contient la liste des faits, la deuxième la valeur courante du fait selectionné dans la liste et la troisième son explication (voir fig.II.10)

Algorithme du module de modification de paramètres :

```
Début
    lire le nom de la base NB dans "NOMBASE.TRA"
    changer la database avec le fichier NB + ".PAR"
    changer la database avec le fichier NB + ".FBA"
    changer la database avec le fichier "OLDBASE.TRA".
    demander à l'utilisateur de choisir entre "Simulation" et
    "Modification de paramètres"
    si choix = "Simulation"  alors
        liste_aff ← liste des X tels que FAIT_BASE (X)
        SIMU ← VRAI
    sinon { choix = "modification" de paramètres" }
        liste_aff ← liste des X tels que PARAM (X , _ , _)
        SIMU ← FAUX
    fsi
    partager l'écran en 3 fenêtres F1 , F2 et F3
    affichage de liste_aff dans F1 sous forme de menu déroulant
    N ← 1
    liste_mod ← [ ]
    tq l'utilisateur n'a pas appuyé sur ESC  faire
        mettre en inversion vidéo la ligne N du menu dans F1
        écrire dans F2 la valeur V du N-ième fait de liste_aff FN
        { VA (Fn , V , _) }
        efface le contenu de F3
```

```
        si PARAM (Fn , _ , T) existe  alors
        | écrire T dans F3
        fsi
        lire (C)
        si C est une touche déplacement du curseur  alors
        | changer la ligne courante N
        sinon
        | gérer l'entrée au clavier de la nouvelle valeur dans F2 (avec
        | contrôle de type)
        | si Fn n'appartient pas à liste_mod  alors
        | | ajouter le fait Fn en tête de liste_mod
        | fsi
        fsi
      ftq
      si NON SIMU alors
      | demander à l'utilisateur de choisir "Inférence" ou "Connexion"
      | si choix = "Inférence"  alors
      | | les nouvelles valeurs doivent être mises dans NOUFAIT.TRA
      | | pour réévaluer le diagnostic
      | | effacer NOUFAIT.TRA
      | | pour chaque fait FM de liste_mod  faire
      | | | unifier VA (FM , X , _)
      | | | écrire dans NOUFAIT.TRA : (FM , "=" , X)
      | | fpour
      | sinon { les nouvelles valeurs doivent être mises en database et
      | | | seront prises en compte lors du cycle suivant}
      | | | enlever de la database tous les PARAM (_ , _ , _)
      | | | enlever de la database tous les FAIT_BASE (_)
      | | | SAUVEGARDER la database dans OLDBASE.TRA
      | fsi
      sinon {pas de choix possible : inférence obligatoire}
      |     effectuer NOUFAIT.TRA
      |     pour chaque fait FM de liste_mod  faire
      |       | unifier VA (FM , X , _)
      |       | écrire dans NOUFAIT.TRA (FM , "=" , X)
      |     fpour
      fsi
fin
```

4.7) Justification des commentaires : APJUST

Le module de justification est écrit en PROLOG. Il s'insère entre le moteur d'inférences et la journalisation. Sa présence dans le cycle de fonctionnement d'APSYST est définie dans APINIT. Son rôle est de

permettre à l'utilisateur de savoir pourquoi tel ou tel commentaire a été généré.

L'exécution du module APJUST commence par le chargement en database du contexte "OLDBASE.TRA".

Tous les faits sont présents en mémoire. De plus, on y trouve le prédicat **COMMENTER (C , A , O)** pour chaque commentaire généré par l'inférence :

C représente le n° du commentaire
A représente la valeur de la variable
O représente le n° de la règle d'origine

Une méthode évidente de justification des commentaires est l'écriture d'un moteur d'inférences en chaînage arrière. Mais elle est coûteuse en place mémoire et en temps de calcul.

Nous avons retenu une autre méthode de conception plus simple , moins coûteuse en ressource mémoire et beaucoup plus rapide.

Chaque fait en database a un argument supplémentaire qui est le numéro de la règle qui a permis de le calculer.
Dans le cas d'un fait de base issu de métrologie ou un paramètre, le dernier argument est mis à zéro par convention.

Voici l'algorithme général de fonctionnement du module APJUST :

```
Début
    changer la database à partir d'OLDBASE.TRA
    pour tous les commentaires de la database
        afficher le commentaire
        CAR ← "F"
        SEC ← 20
        SECCOUR ← prend_seconde
        tq SEC <> 0 faire
            si SECCOUR <> prend_seconde alors
                SECCOUR ← prend_seconde
                SEC ← SEC - 1
            fsi
            si l'utilisateur a frappé quelque chose au clavier alors
                C ← caractère frappé au clavier
                si C appartient à [S , F , P , A] alors
                    CAR ← C
                    SEC ← 0
                fsi
            fsi
        ftq
        traitement (CAR)
    fpour
fin
```

traitement ("S") : *passer au commentaire suivant*
traitement ("P") : *demander une justification immédiate*
traitement ("A") : *justifier un ancien commentaire*
traitement ("F") : *fin du module de justification.*

4.7.1) Commentaire suivant

Cette commande correspond à l'itération dans la boucle "pour" de l'algorithme.

4.7.2) Justification immédiate

Situation : pendant l'exécution du module APUNFER, l'utilisateur vient de voir apparaître à l'écran un commentaire, il souhaite comprendre tout de suite pourquoi.

La justification consiste à prendre et afficher la règle d'origine d'un commentaire. Ensuite, pour chaque règle, on prend et on affiche la règle d'origine d'un fait de la partie condition. Et ainsi de suite, jusqu'aux fait de base.

Tous les faits et règles de justification sont affichés à l'écran et imprimés.

Algorithme de TRAITEMENT ("P")

Au moment de l'appel de TRAITEMENT ("P"), un commentaire est affiché et l'utilisateur a demandé une justification :

COMMENTER (C , A , O)
CAR = "P"

L'algorithme du début de TRAITEMENT ("P") est :

```
Début
    | unifier REG (O , _ , _ , _)
    | afficher la règle en notation externe
    | choix (O , A)
 Fin
```

Algorithme général de TRAITEMENT("P")

```
Début
    | C ← lire_parmi ["S" , "F" , "P"]
    | si C = "P"  alors
    |   | unification avec un élément de la database de REG (O , _ , LC , _)
    |   | si LC ne contient qu'un élément F  alors
    |   |   | FAJ ← F
    |   | sinon
    |   |   | FAJ ← choix d'un élément de LC
    |   | fsi
    |   | prendre le numéro de la règle d'origine de FAJ : ORFAJ
```

```
        si ORFAJ = O alors
        | affichage de la valeur de FAJ
        |         (éventuellement VARIABLE = A)
        sinon
        | unifier avec un élément de la database REG (ORFAJ , _ , _ , _)
        | afficher la règle en notation externe
        | {ce sont les prédicats inverses de APA2}
        | CHOIX (ORFAJ , A)
        fsi
      fsi
      si C = "S" alors
      | itérer la boucle "pour" de l'algorithme d'APJUST
      fsi
      si C = "F" alors
      | sortir de la boucle "pour"
      fsi
fin
```

4.7.3) Justifier un ancien commentaire

Situation : l'utilisateur remarque sur le journal imprimé un commentaire intéressant dont il connaît l'heure et la date. Il aimerait savoir pourquoi ce commentaire a été généré.

On ne dispose plus des faits en mémoire. Pour permettre cette justification, il va falloir recréer la base de connaissances qui est à l'origine de ce commentaire.

Algorithme de TRAITEMENT ("A")

```
Début
    si le fichier FROID.TRA ne contient pas "1"  alors
    | COPY OLDBASE.TRA CHAUDBASE.TRA
    | {sauvegarde du contexte courant}
    | COPY ECRAN.* CHAUD.*
    | {sauvegarde des écrans comptés}
    | mettre "1" dans FROID.TRA
    | {pour exprimer qu'il s'agit d'une justification en temps différé
    | (à froid) }
    fsi
    demander la date et l'heure du commentaire à justifier
    lire (HCOM)
    {les écrans archivés ont pour nom : JJMMhhmm}
    si HCOM = "ESC" alors
    | revenir au début de APJUST
    sinon
    | si il existe au moins un écran de nom HCOM  alors
    | | COPY HCOM.* ECRAN.*
    | sinon
    | | tq (il n'existe pas d'écran de nom HCOM) et (HCOM<>"ESC")
    | | faire
    | | | lire (HCOM)
    | | ftq
    | fsi
    fsi
    sortir de la boucle "pour" de l'algorithme général d'APJUST
Fin
```

Le passage par le module APCHANG permet de donner aux paramètres leur valeur lors de la génération du commentaire à justifier. Ensuite, le module APEXT est rappelé. L'extraction sera refaite. Les mêmes faits seront mis dans NOUFAIT.TRA et lors de l'inférence, le même commentaire sera regénéré. Il sera facile ensuite de le justifier comme s'il s'agissait d'une justification immédiate.

Le fichier FROID.TRA permet de savoir, avant de sauvegarder les contextes, si cela n'a pas déjà été fait auparavant. Dans ce cas, le contexte courant serait un ancien contexte.

4.7.4) Fin de l'algorithme de justification

Dans le cas où l'on a fait de la justification différée, il est nécessaire de récupérer les contextes avant de continuer.
Les commandes sont :

COPY CHAUDBASE.TRA OLDBASE.TRA
COPY CHAUD.* ECRAN.*
mettre "O" dans FROID.TRA.

4.8) Histogrammes : APHIST

Ce module est écrit en PASCAL

A partir des 4 séries de valeurs stockées à chaque exécution du module d'extraction, le module APHIST trace 4 histogrammes.

Pour cela, on utilise les sous-programmes et déclaration de type du TURBO GRAPHIX TOOLBOX :

Définition de l'écran graphique CGA
Définition de fenêtres graphiques
Gestion de fenêtres graphiques
Tracés d'axes
Tracés d'histogrammes.

La forme des fichiers contenant les séries est :
heure de lecture valeur

Il faut constater qu'avec cette méthode, l'échelle des heures est calculée dynamiquement. En dessinant des barres toutes les "T" heures, il se peut qu'une valeur n'existe pas. De même, en "T" heures, on peut avoir plusieurs valeurs. L'histogramme étant une fonction échelle, on a choisi de ne pas laisser de barres nulles et d'aligner avec la valeur précédente, et de prendre la première valeur de chaque tranche horaire. Le nombre de tranche est limité à 100.

Algorithme de lecture des séries dans le tableau A : lire_série (NOM)

```
Début
    ouvrir le fichier (NOM) en lecture
    lire sur le fichier (X , Y)
    MAX ← Y
    DEBUT ← X
    A[1] ← Y
     I ← 2
    lire sur le fichier (X , Y)
    tq non fin (NOM) et I<100  faire
      INF ← DEBUT+ ( (I - 1) * T)
      SUP ← DEBUT + (I * T)
      si X > INF et X <= SUP  alors
        A [I] ← Y
        si Y>MAX  alors
          MAX ← Y
        fsi
      sinon
        si X>SUP  alors
          tq non ( ( X>INF) et (X>=SUP) )  faire
            A[I] ← A[I - 1]
            I ← I + 1
            INF ← DEBUT+ ( (I - 1) * T)
            SUP ← DEBUT + (I * T)
          ftq
        fsi
      fsi
    lire sur le fichier (X , Y)
    ftq
  Fin
```

Pour obtenir une version papier des histogrammes, il suffit de faire une hardcopy d'écran graphique.

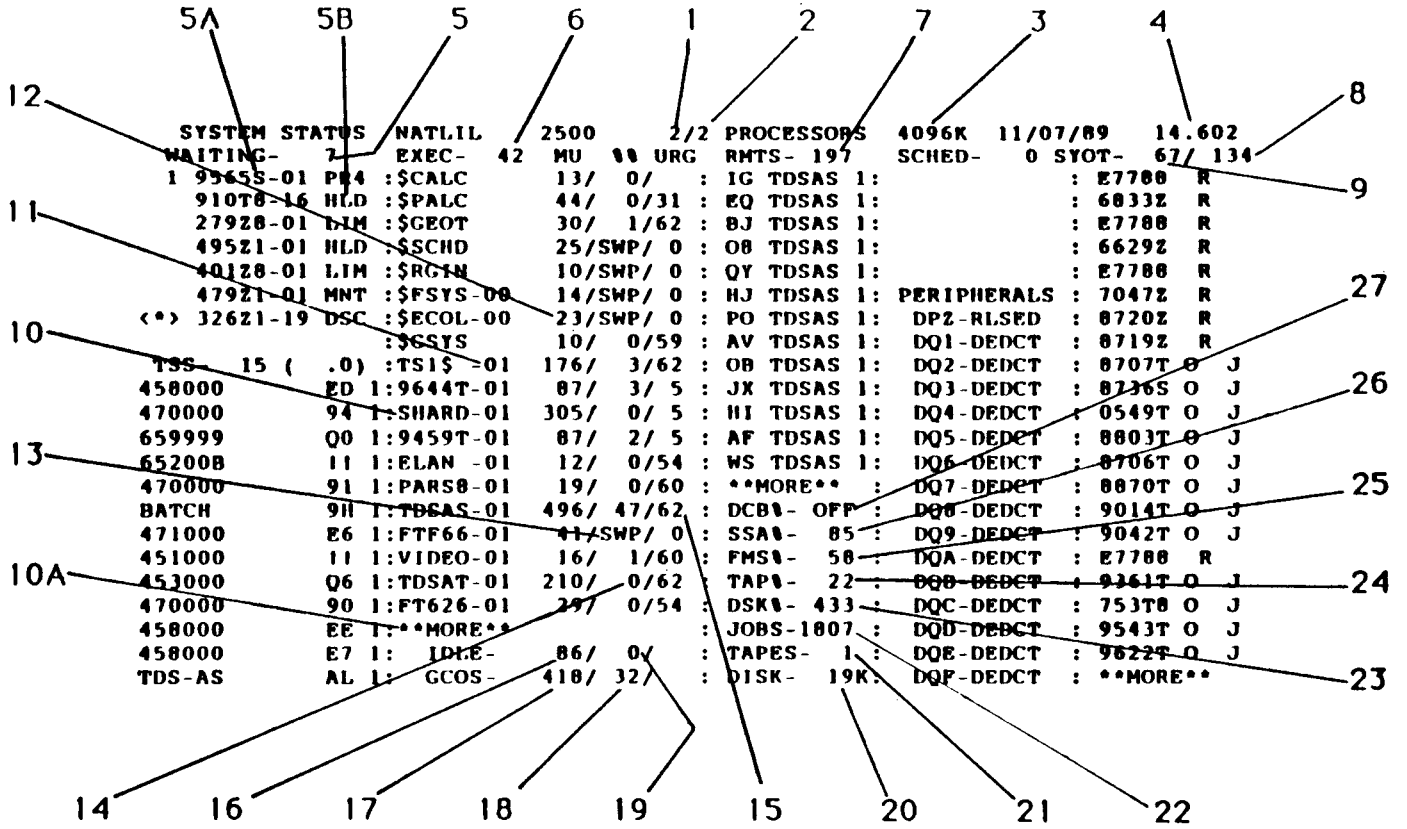
V L'expertise et la base de connaissances

1) Les faits de bases issus de métrologie

Les écrans de chaque moniteur temps réel seront expliqués fait par fait.

1.1) Ecran VIDEO (Voir figure V.1)

N° de valeur	Explication
1	Nombre de processeurs actifs sur le DPS 8
2	Nombre de processeurs présents sur le DPS 8
3	Taille mémoire du DPS 8 Cette valeur est donnée en "mots machine" 1 mot = 36 bits = 4 octets + 4 bits ici 4096 k mot = 4096 2^{10} mots
4	Heure décimale
5 5A 5B	Nombre de JOBS en attente d'exécution Noms des JOBS en attente Cause de l'attente
6	Nombre de JOBS en exécution
7	Nombre de stations connectées au DPS8
8	Nombre de "SYSOUT" à sortir sur imprimante Une "Sysout" correspond au résultat papier de l'exécution d'un JOB
9	Nombre de "SYSOUT" non encore sortis
10 10A	Nom des JOBS en exécution **MORE** indique, que la liste des jobs continue sur l'écran suivant
11	Numéro de l'activité des jobs en exécution



SYSTEM STATUS NATLIL 2500 2/2 PROCESSORS 4096K 11/07/89 14.602
WAITING- 7 EXEC- 42 MU %% URG RMTS- 197 SCHED- 0 SYOT- 67/ 134
1 95655-01 PR4 :$CALC 13/ 0/ : IG TDSAS 1: : E7700 R
910T8-16 HLD :$PALC 44/ 0/31 : EQ TDSAS 1: : 6033Z R
27928-01 LIM :$GEOT 30/ 1/62 : BJ TDSAS 1: : E7700 R
495Z1-01 HLD :$SCHD 25/SWP/ 0 : OB TDSAS 1: : 6629Z R
401Z8-01 LIM :$RGIN 10/SWP/ 0 : QY TDSAS 1: : E7700 R
479Z1-01 MNT :$FSYS-00 14/SWP/ 0 : HJ TDSAS 1: PERIPHERALS : 7047Z R
(*) 326Z1-19 DSC :$ECOL-00 23/SWP/ 0 : PO TDSAS 1: DP2-RLSED : 8720Z R
:$GSYS 10/ 0/59 : AV TDSAS 1: DQ1-DEDCT : 8719Z R
TSS- 15 (.0) :TS1$ =01 176/ 3/62 : OB TDSAS 1: DQ2-DEDCT : 8707T O J
458000 ED 1:9644T-01 87/ 3/ 5 : JX TDSAS 1: DQ3-DEDCT : 8736S O J
470000 94 1:SHARD-01 305/ 0/ 5 : HI TDSAS 1: DQ4-DEDCT : 0549T O J
659999 Q0 1:9459T-01 87/ 2/ 5 : AF TDSAS 1: DQ5-DEDCT : 8803T O J
65200B 11 1:ELAN -01 12/ 0/54 : WS TDSAS 1: DQ6-DEDCT : 8706T O J
470000 91 1:PARS8-01 19/ 0/60 : **MORE** : DQ7-DEDCT : 8870T O J
BATCH 9H 1:TDSAS-01 496/ 47/62 : DCB%- OFF : DQ8-DEDCT : 9014T O J
471000 E6 1:FTF66-01 41/SWP/ 0 : SSA%- 85 : DQ9-DEDCT : 9042T O J
451000 11 1:VIDEO-01 16/ 1/60 : FMS%- 58 : DQA-DEDCT : E7700 R
453000 Q6 1:TDSAT-01 210/ 0/62 : TAP%- 22 : DQB-DEDCT : 9161T O J
470000 90 1:FT626-01 297 0/54 : DSK%- 433 : DQC-DEDCT : 753T8 O J
458000 EE 1:**MORE** : JOBS-1807 : DQD-DEDCT : 9543T O J
458000 E7 1: IDLE- 86/ 0/ : TAPES- 1 : DQE-DEDCT : 9622T O J
TDS-AS AL 1: GCOS- 418/ 32/ : DISK- 19K: DQF-DEDCT : **MORE**

FIGURE V.1
Ecran VIDEO

12	Consommation mémoire des jobs en mots machine
13	SWP : indique que le job est présent en mémoire mais qu'il est "SWAPPE" car personne ne l'utilise Ex : FTF66 correspond au job de transfert de fichier Il est disponible mais aucun transfert n'est en cours
14	Pourcentage d'utilisation des processeurs Le maximum étant : 100 x nb processeurs actif
15	Priorité des jobs
16	Mémoire disponible en mots machine
17	Mémoire utilisée par le système d'exploitation GCOS
18	Pourcentage processeur utilisé par GCOS
19	Pourcentage processeur disponible
20	Place disponible sur disque en "Little Link" (LL) Un LL correspond à 1920 mots
21	Nombre de bandes utilisées
22	Nombre de jobs exécutés
23	Pourcentage d'utilisation des disques Maximum = 100 x nb de contrôleur Ici maximum = 600
24	Peu intéressant pour APSYST
25	Peu intéressant pour APSYST

26 Pourcentage d'utilisation de la "SSA cache"
La "SSA cache" est une zone mémoire où le système GCOS recopie le code exécutif de certains programmes afin d'augmenter ses performances lors d'une prochaine utilisation
Cette valeur correspond au taux de programmes actifs qui sont exécutés à partir de la SSA cache plutôt qu'à partir des disques

27 Pourcentage d'utilisation du système d'optimisation des accès disque : SYSTEM DISK CACHE BUFFER
La valeur est à OFF si le SDCB n'est pas utilisé

1.2) Ecran EXEC de TPMON (Voir figure V.2)

1 Nombre de transactions en exécution

2 Nombre total de transactions exécutées depuis le début de la session du TP

3 Nombre total de transactions de type BIBO (Batch In Batch Out) exécutées depuis le début de la session TP

4 Nombre de TPR dont l'exécution a nécessité une lecture du code exécutable sur disque

5 Nombre de TPR exécutées

6 Nombre de messages erronés envoyés par les utilisateurs

7 Nombre d' "aborts" TPR

8 Nombre de vidages de TPR (à rapprocher de 15)

9 Durée de la session TP sur le DPS

10 Nombre moyen de transactions exécutées par seconde

11 Nombre maximum de transactions exécutées par seconde depuis le début de la session TP

12 Nombre total de messages qui se sont trouvés en attente depuis le début de la session TP

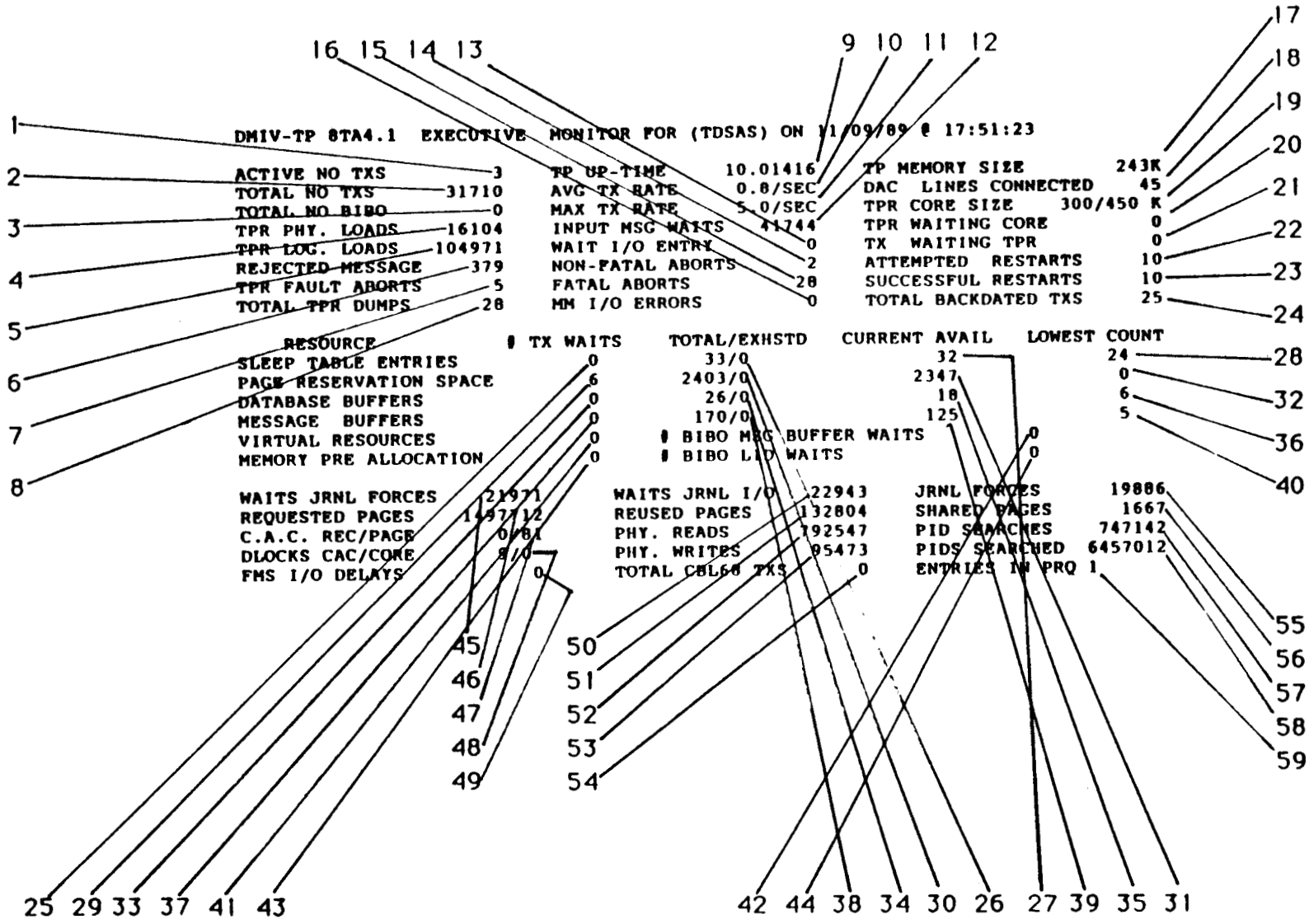


FIGURE V.2
Ecran EXEC de TPMON

13 Non utilisé par APSYST

14 Nombre "d'aborts" TPR que le système TP a pu réparé seul

15 Nombre "d'aborts" TPR que le système TP n'a pu réparé seul et qui ont généré un "dump" (voir 8)

16 Non utilisé par APSYST

17 Taille mémoire de l'exécutif TP

18 Nombre de communications "physiques" établies entre le Datanet et le TP
En mode DACQ, ce nombre correspond au nombre de gens connectés au TP
En mode DACQ QUEUED, ce nombre est fixe et donc peu significatif
Voir II.2.1 pour l'explication de ces modes

19 Taille de la zone mémoire réservée pour mettre les exécutifs de TPR
La valeur de gauche correspond à la zone occupée
La valeur de droite correspond à la zone maximale

20 Nombre de TPR qui se sont retrouvées en attente de place mémoire

21 Nombre de transactions qui se sont retrouvées en attente d'une TPR disponible

22 Nombre de tentatives de relances de transactions (suite à un abort par exemple)

23 Nombre de relances de transactions réussies

24 Non utilisé par APSYST

25 Nombre de transactions qui se sont retrouvées en attente d'une place disponible dans la table de "SLEEP"

26 Gauche : nombre d'entrées de la table des "SLEEP"

27 Nombre d'entrées disponibles dans la table des "SLEEP"

28	Valeur minimale du nombre d'entrée dans la table des " SLEEP" depuis le début de la session TP
29	Nombre de transactions qui se sont retrouvées en attente de PAGE RESERVATION SPACE
30	Gauche : nombre total de PAGE RESERVATION SPACE
31	Nombre de PAGE RESERVATION SPACE disponible
32	Valeur minimale du nombre de PAGE RESERVATION SPACE depuis le début de la session TP
33	Nombre de transactions en attente de DATABASE-BUFFERS
34	Nombre total de DATABASE-BUFFERS
35	Nombre de DATABASE-BUFFERS disponibles
36	Nombre minimal des DATABASE-BUFFERS disponibles depuis le début de la session TP
37	Nombre de transactions qui se sont retrouvées en attente de messages buffers depuis le début de la session
38	Nombre total de messages buffers de TP
39	Nombre de messages buffers disponibles
40	Valeur minimale du nombre de messages buffers disponibles depuis le début de la session TP
41	Nombre de transactions qui se sont retrouvées en attente de ressources virtuelles
42	Nombre de transactions de type BIBO qui ont été en attente de messages buffers
43	Nombre de transactions qui se sont retrouvées en attente de PREALLOCATION mémoire
44	Nombre de transactions de type BIBO qui ont dû attendre une ressource

45 Nombre de journalisations forcées qui ont dû attendre avant d'être exécutées

46 Nombre de pages base de données qui ont été demandées

47 Nombre de problèmes d'accès concurrents que le système a dû gérer au niveau :
(gauche) des enregistrements ou
(droite) des pages BD

48 Nombre de deadlocks que le système a dû débloquer au niveau
(gauche) des contrôles d'accès concurrents
(droite) des zones mémoires

49 Nombre de retards survenus dans les accès physiques aux fichiers

50 Nombre de journalisations qui ont dû attendre l'accès entrées/sorties.

51 Nombre de pages BD que le système a pu réutiliser (c'est-à-dire, qu'il a évité un accès physique)

52 Nombre de lectures physiques sur les fichiers BD

53 Nombre d'écritures physiques (mise à jour BD)

54 Nombre de transactions exécutées qui ont été écrites en COBOL 68

55 Nombre d'écritures journaux forcées

56 Nombre de pages BD partagées

57 Nombre d'appels de Page IDentifier

58 Nombre de Page IDentifier effectivement accédée par le système

59 Nombre de transactions prêtes à être exécutées et qui sont en attente d'un couloir dans l'exécutif (CONTROL TASK)
PRQ = Pending Request Queue

1.3) Ecran FILE de TPMON (Voir figure V.3)

1	Voir 46 de l'écran EXEC
2	Voir 51 de l'écran EXEC
3	Voir 56 de l'écran EXEC
4	Voir 52 de l'écran EXEC
5	Voir 53 de l'écran EXEC
6	Voir 49 de l'écran EXEC
7	Voir 33 de l'écran EXEC
8	Voir 34 de l'écran EXEC
9	Voir 47 droite de l'écran EXEC
10	Voir 48 gauche de l'écran EXEC
11	Nom logique du fichier (File Code sur 2 caractères)
12	Nombre d'accès physiques en lecture
13	Nombre d'accès logiques en lecture
14	Nombre d'accès physiques en écriture
15	Nombre d'accès logiques en écriture
16	Nombre de conflits d'accès
17	Nombre de "restarts" de transactions dûs à un DEADLOCK
18	Nombre de "restarts" de transactions

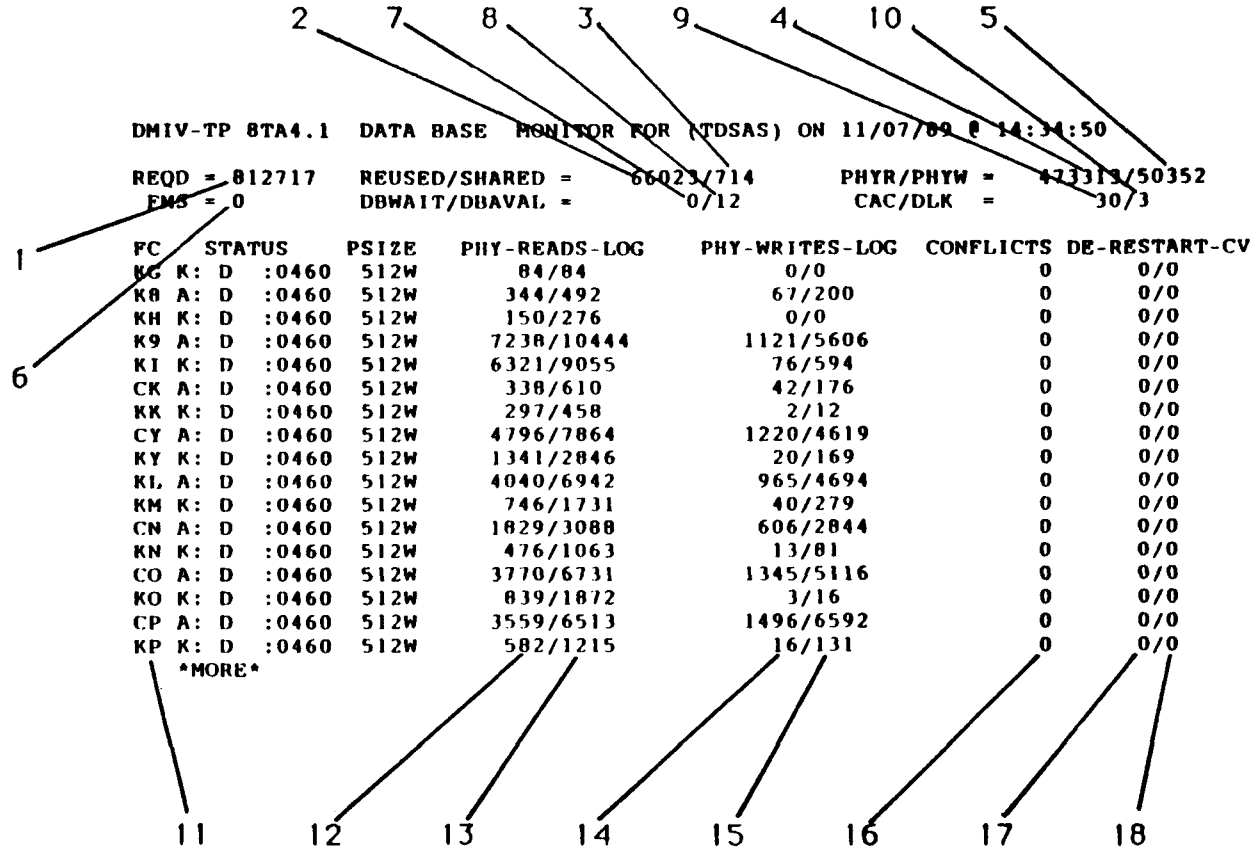


FIGURE V.3
Ecran FILE de TPMON

Explications des points 12 à 15 :

Une opération de lecture écriture sur fichier est d'abord effectuée en mémoire avant d'être réalisée sur le disque. Quand un utilisateur demande la consultation d'une page base de données, le système regarde si la page est présente en mémoire dans un data-base buffer. Dans ce cas, le compteur des lectures logiques est incrémenté du 1. Si elle n'est pas présente en mémoire, il y a accès physique sur le disque et les compteurs logiques et physiques sont incrémentés.

1.4) Ecran SSA de TPMON (Voir figure V.4)

1	Heure de démarrage du TP
2	Temps écoulé depuis le démarrage du TP
3	Consommation CP total du TP
4	Consommation IO du TP
5	Niveau de priorité du TP
6	Non utilisé par APSYST
7	Temps d'occupation des couloirs IO pour le TP (sensiblement égal à la valeur n° 4)
8	Voir le II pour l'explication des types de priorité
9	Taille mémoire du TP
10	"File code" du fichier (nom logique sur 2 caractères)
11	"Channel-time" pour le fichier spécifié Temps d'occupation des couloirs IO pour des accès au fichier

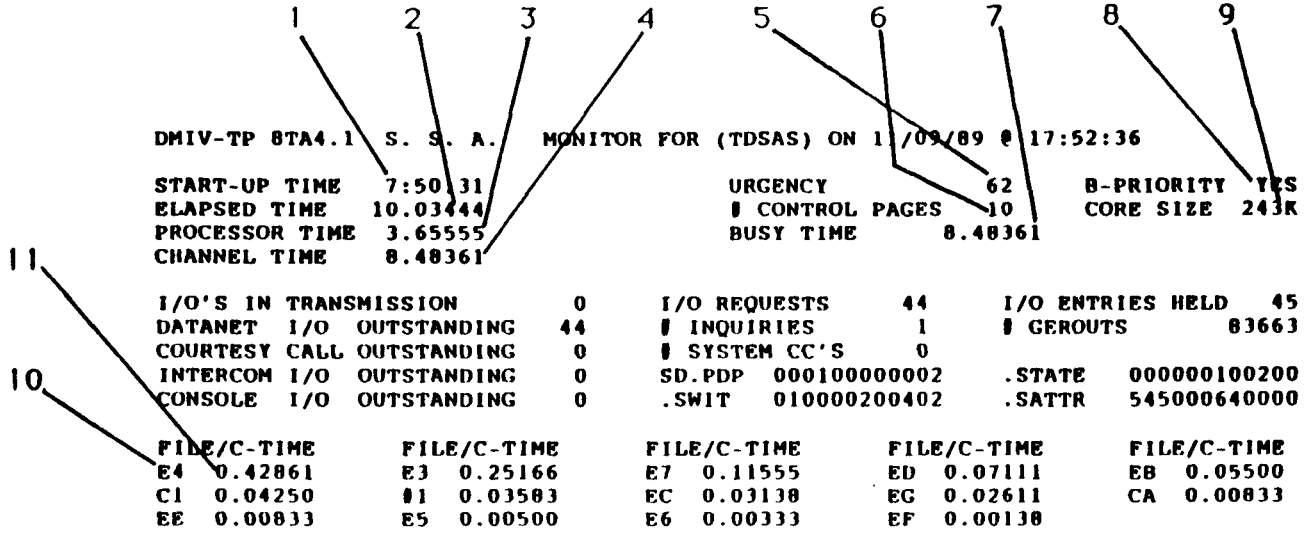


FIGURE V.4
Ecran SSA de TPMON

1.5) Ecran TASK de TPMON (Voir figure V.5)

N.B Cet écran est le plus important de TPMON car il permet de suivre le TP de près.

Grâce à lui, on sait : **QUI** fait **QUOI** à **QUELLE HEURE**

1	Numéro d'identification interne au TP pour une transaction Ce numéro chronologique est appelé SERIAL NUMBER
2	Numéro de la phase pour la transaction identifiée en 1 Chaque changement de phase correspond à un point de reprise pour le TP en cas de problème
3	Identification logique de l'utilisateur. Quand la transaction est présente mais non active (status "complete" - voir 15) cette zone est laissée en blanc
4	Identification de la transaction pour le TP
5	Nom de la TPR active dans le couloir ou de la dernière TPR activée dans le couloir
6	Taille mémoire de la TPR
7	Nombre de messages buffers alloués à la transaction dans le couloir (plus précisément à son occurrence représentée par son SERIAL NUMBER)
8	Nombre de messages buffers utilisés par cette transaction
9	Nombre de Data Base buffers alloués à la transaction
10	Nombre de Data Base buffers utilisés par la transaction
11	Nombre de Page IDentifier alloués à la transaction
12	Nombre de Page IDentifier utilisés par la transaction
13	Temps processeur consommé par la transaction
14	Temps processeur limite restant pour la transaction

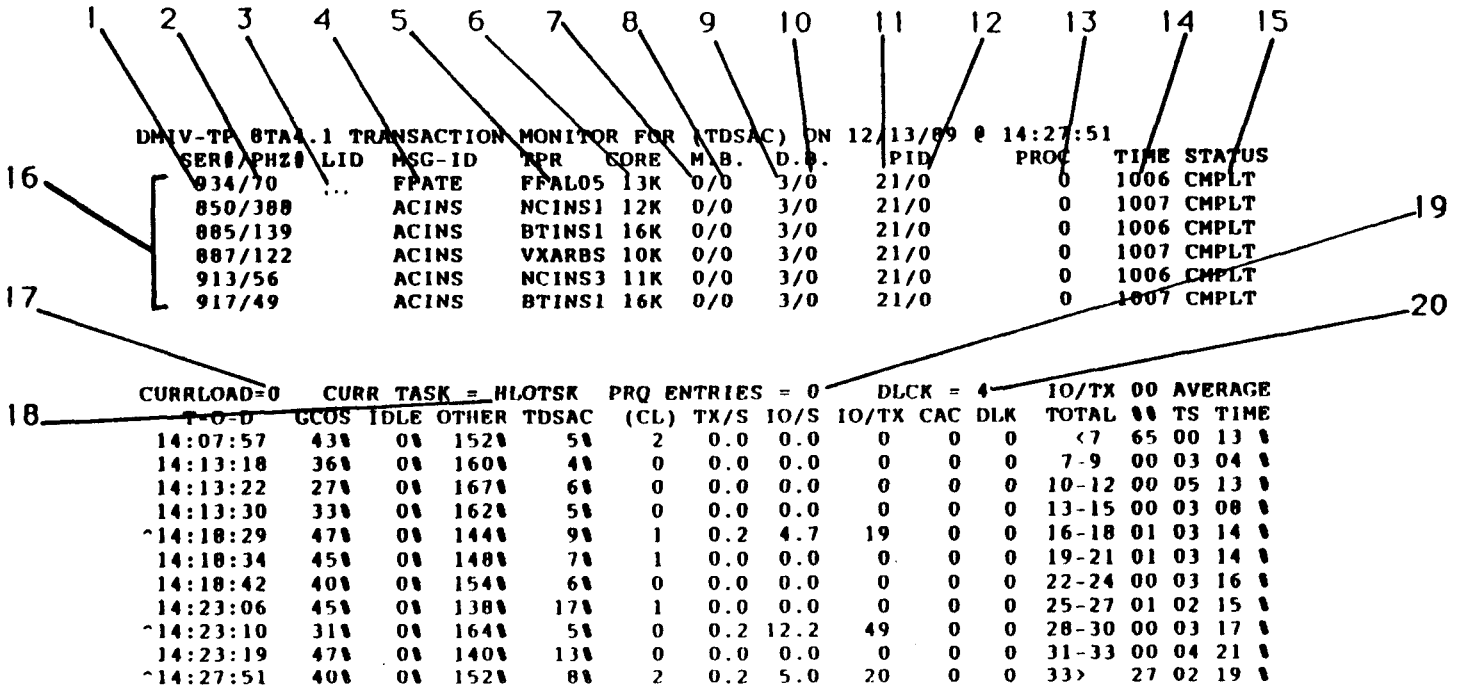


FIGURE V.5
Ecran TASK de TPMON

15 Status de la transaction ou de la TPR
Ex : COMPLETE - la TPR est terminée
PG (K3) - en lecture écriture sur le fichier K3
JRN I/O - en plan de journalisation

16 Valeur du Normal Load (6 dans l'exemple)
Cette valeur correspond au taux de simultanéité du TP à savoir le nombre de transactions qui peuvent disposer d'un couloir pour être actives simultanément

17 Valeur courante du Normal Load
Dans l'exemple cette valeur vaut zéro toutes les transactions sont dans un état COMPLETE

18 Numéro de la transaction effectivement active
Même si le taux de simultanéité est supérieur à 1 en réalité une seule transaction ne peut être effectuée à la fois

19 Voir 59 de l'écran EXEC

20 Voir 48 gauche de l'écran EXEC

2) Utilisation des faits

2.1) Extraction

Comme on a vu aux chapitres précédents, la métrologie est effectuée en deux étapes :

1 - Connexion et capture d'écrans

2 - Préparation des faits

A la fin de la première étape, les écrans, tels qu'ils ont été décrits en V.1., sont stockés sur fichier.

A noter que pour la préparation des faits, plusieurs opérations sont réalisées, et qu'il ne s'agit pas seulement d'une remise en forme.

On peut distinguer 3 types de préparation des faits :

1 - Remise en forme

Les faits sont présentés au moteur d'inférences de la manière la plus simple possible, à raison d'un fait par ligne :

Exemple :

fait booléen vrai	⇒	NOM_DE_FAIT
fait booléen faux	⇒	NON NOM_DE_FAIT
fait booléen variable vrai	⇒	NOM_DE_FAIT (I)
fait booléen variable faux	⇒	NON NOM_DE_FAIT (I)
fait réel	⇒	NOM_DE_FAIT = valeur
fait réel variable	⇒	NOM_DE_FAIT (I) = valeur

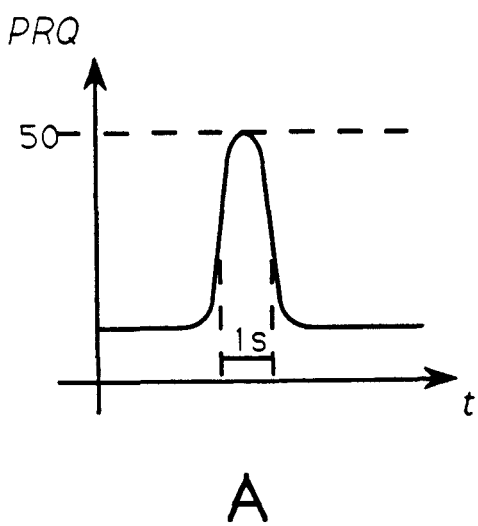
Cette simple remise en forme est valable pour les faits dont la valeur correspond à un cumul sur toute la session TP. Il s'agit de valeurs croissantes.

2 - Calcul de moyenne

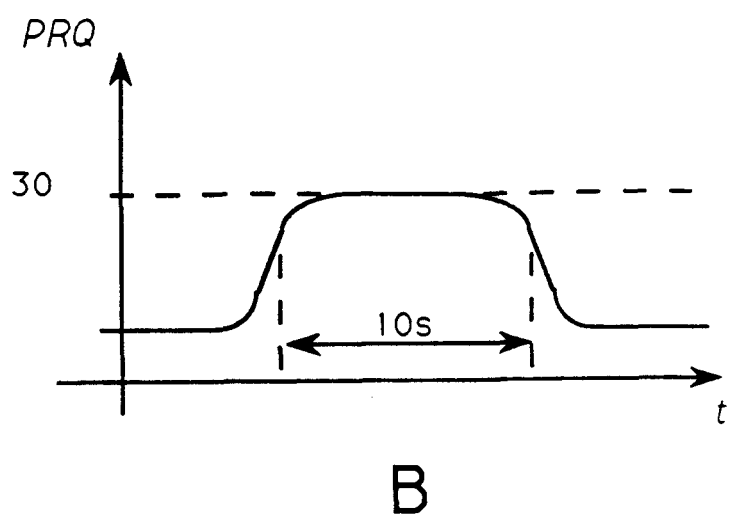
Certains faits représentent une valeur "temps réel" et fluctuante.

Pour ces faits, la préparation consiste en un calcul de moyenne sur une durée courte (quelques secondes) mais suffisante pour absorber des pointes non significatives.

Pour illustrer cela on peut dessiner deux graphiques représentant l'évolution de la PRQ pendant le temps de la métrologie
(voir V.1 Ecran EXEC n°59)



A



B

Commentaires :

A
Le fait possède une valeur très élevée pendant un court laps de temps : 50 pendant 1 seconde

Cela correspond à une surcharge ponctuelle forte mais que le système transactionnel absorbe sans aucun problème. Il s'agit d'un fonctionnement normal du TP.

B
La valeur maximale est plus faible qu'en A, mais elle dure plus longtemps. Cela prouve que le système transactionnel n'arrive pas à satisfaire les demandes. Il est nécessaire dans ce cas de sortir un diagnostic.

3 - Tri & Calcul de maxima

On a vu dans l'écran TASK de TPMON (voir fig. V.5) qu'un certain nombre de lignes correspondent au taux de simultanéité ou Normal Load. Ces lignes représentent les couloirs du TP qui contiennent une transaction active (valeurs 1 à 15).
Dès qu'il y a échange de message, la transaction concernée, est SWAPPE ou tout au moins démunie de ses ressources. Il est possible que la TPR suivante dans la même transaction soit exécutée dans un autre couloir. Si le TP n'est pas chargé, il se peut même qu'une TPR qui a fini d'être exécutée se trouve dans un autre couloir dans un état *COMPLETE*.
Pour donner des informations significatives sur une transaction, il est important de savoir pendant combien de temps elle est restée active en mémoire sans libérer ses ressources.
Pour cela, il faut trier les occurrences de transaction par SERIAL NUMBERS, puis PHASES puis STATUS.

En résumé : Par cette méthode, on sait combien de temps chaque transaction est restée active et quelles sont les ressources qu'elle a consommées.

2.2) L'exploitation des faits par le moteur

Il existe plusieurs bases de règles, ayant des finalités différentes. (contrôle des fichiers, contrôle des transactions, contrôle général de la machine, suivi de nouvelles applications, etc). La taille de ces bases varie de 30 à 150 règles avec variables. La taille des bases de faits correspondantes varie de 100 à 1000 faits.

Le temps d'un cycle est aussi variable suivant le travail demandé au système expert. Un simple contrôle des activités du DPS 8 avec une métrologie limité au seul VIDEO permettra de faire plus de 30 cycles par heures (c'est-a-dire environ 1 minute pour la connexion et 1 minute pour le moteur d'inférences et l'archivage). Par contre, si l'on prend la base de règles standard, (pour surveiller le TP uniquement) le temps de cycle est d'environ 6 minutes qui sont réparties en 2/3 pour la connexion et 1/3 pour le moteur d'inférences et l'archivage, pour une base de d'environ 100 règles et 800 faits.

Il est important de préciser que le système APSYST a déjà fonctionné pendant plus de 1500 heures sur le site de CdFi-NATEL, ce qui représente plus de 20000 cycles.

La base de règles standard est relativement large et peu profonde. Elle est facilement perfectible avec le mode EXPERT.

Dans son état actuel, elle permet surtout de faire un contrôle du fonctionnement du TP.

L'avantage principal d'APSYST est de faire ce contrôle en temps réel de manière répétitive fréquente, à raison de 10 à 20 cycles par heure.

Un site ou un TP est caractérisé par des seuils. Ceux-ci représentent une normalité d'utilisation des applications.

Exemple 1 :

Pour un TP, on calcule le rapport entre la consommation processeur et la consommation Entrées/Sorties.
Sur le site, on sait que ce rapport vaut a peu près 1 pour 3 quand le TP fonctionne normalement.
APSYST contrôle que ce rapport est compris entre 0,25 et 0,40 ou entre 0,30 et 0,35 suivant le degré de précision exigée.

Exemple 2 :

Pour chaque fichier, APSYST contrôle le taux d'accès physique par rapport aux accès logiques. Dans le meilleur des cas, ce rapport tendra vers 0 : c'est à dire que chaque article de base de données se trouve dans une page présente en Database Buffers et donc il n'y a qu'un seul accès physique à la première lecture. Dans le cas d'un fichier désorganisé, chaque demande d'article de Database fera l'objet d'un accès physique. Dans le pire des cas, ce rapport tendra vers 1.
Dans la pratique, quand ce rapport dépasse 80%, il est demandé à l'administrateur de base de données de réorganiser le fichier.

Quel que soit le site sur lequel on installe le système expert APSYST, les règles sont valables. Seuls les seuils sont à adapter et leurs valeurs peuvent.être très variable.

Exemple :

Le temps limite de consommation processeur pour une TPR ou une transaction est variable selon le type d'application.

Le temps processeur pour une mise à jour en base de données est de quelques milli-secondes, alors que pour la construction d'une liste de noms par similitudes phonétiques à partir de la consultation d'une base de données peut demander plusieurs secondes.

Malgré ces différences sensibles, il faut pouvoir surveiller les deux types de transaction.

3) Structuration de la base de connaissances

Dans l'expertise qui nous concerne, la base de faits peut être découpée en 4 ensembles.

I Les faits concernant les FICHIERS base de données (ou areas)
II Les faits relatifs au JOBS
III Les faits relatifs aux TRANSACTIONS
IV Les faits donnant des informations d'ordre général sur le fonctionnement du DPS8 et du TP

Ces 4 ensembles sont disjoints, mais les informations qu'ils contiennent ne sont pas indépendantes :

I, II et III sont indépendants.
I et IV sont dépendants.
II et IV sont dépendants.
III et IV sont dépendants.

Indépendant doit être compris comme:

- Deux faits appartenant à deux ensembles indépendants ne peuvent apparaître dans une même règle.
- Deux faits apparaissant dans une même règle sont dépendants.

La relation de dépendance est transitive.

On obtient facilement 4 ensembles de règles:

A Les règles correspondant aux faits de l'ensemble I.
B Les règles correspondant aux faits de l'ensemble II.
C Les règles correspondant aux faits de l'ensemble III.
D Les règles correspondant aux faits de l'ensemble IV.

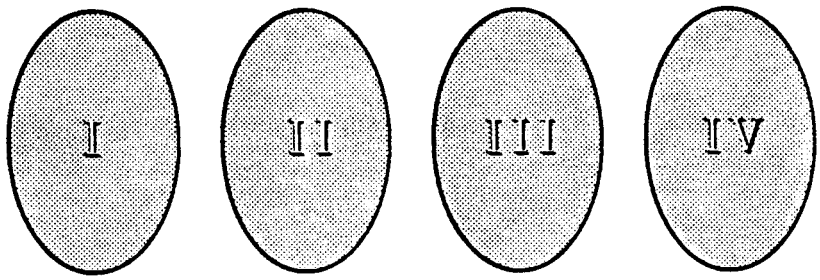
On a entre ces 4 ensembles les relations :

$A \cap B = \emptyset$ $B \cap C = \emptyset$ $A \cap C = \emptyset$

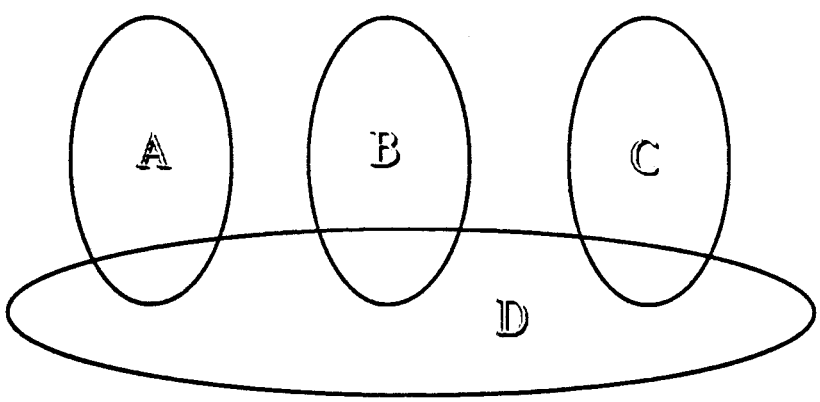
$A \cap D \neq \emptyset$ $B \cap D \neq \emptyset$ $C \cap D \neq \emptyset$

On peut représenter ces ensembles sous forme de diagrammes

Les ensembles de faits:

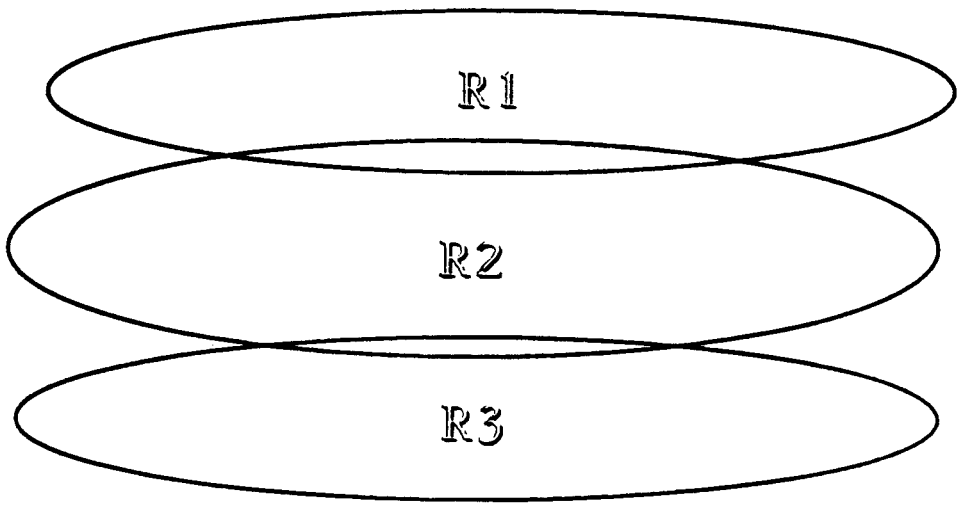


Les ensembles de règles:



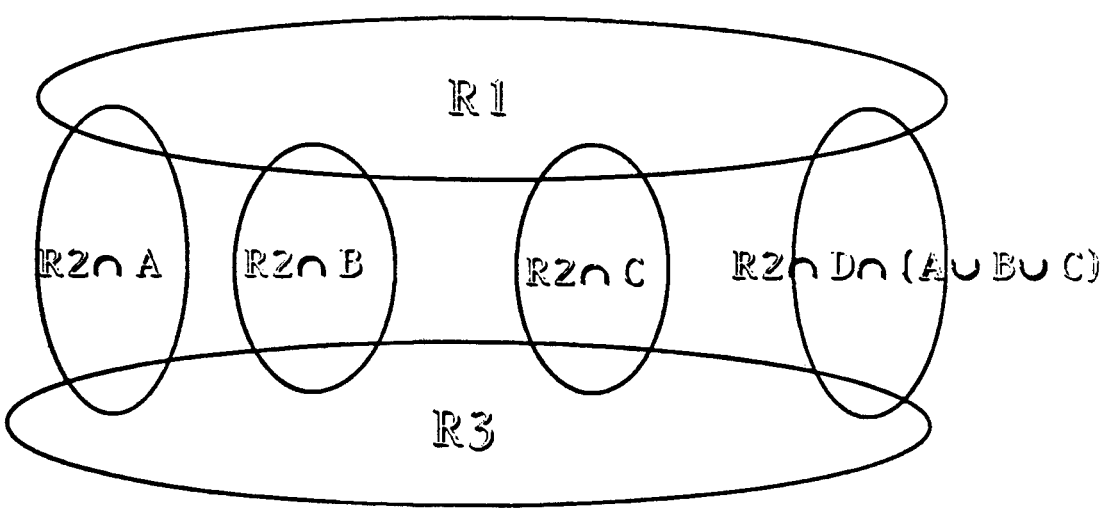
De plus, la base de règles peut être découpée en 3 groupes.

R1 Règles de calculs.
R2 Règles "expertes"
R3 Règles résultats.



Remarque : R1 et R3 ne sont pas obligatoirement disjoints

Sur l'ensemble R1\R2, il n'est pas nécessaire de faire de la saturation, "un seul passage" suffit.
De même pour l'ensemble R3\R2.
Seul sur l'ensemble R2, la saturation est nécessaire.
Il est aussi possible de combiner R1, R2, R3 et A, B, C et D:



Afin d'optimiser le fonctionnement du moteur d'inférences, en tenant compte des propriétés intéressantes qui sont exposées ci-dessus. Le chaînage peut s'effectuer de la manière suivante :

Lecture des nouveaux faits.

Application de toutes les règles de l'ensemble R1 en les exécutant chacune une seule fois.

Saturation de la base en chaînage avant sur les règles de l'ensemble $R2 \cap A$.

Saturation de la base en chaînage avant sur les règles de l'ensemble $R2 \cap B$.

Saturation de la base en chaînage avant sur les règles de l'ensemble $R2 \cap C$.

Saturation de la base en chaînage avant sur les règles de l'ensemble $R2 \cap D \cap (A \cup B \cup C)$.

Application de toutes les règles de l'ensemble R3 en les exécutant chacune une seule fois.

VI APSYST et l'entreprise
du prototype au progiciel

Le prototype d'APSYST, tel qu'il a été décrit dans les chapitre I à V a été utilisé sur le site de CdFi-NATEL pendant plusieurs mois.
Les premiers résultats ont été satisfaisants et certaines applications ont été améliorées grâce à APSYST .

Voir des exemples dans le l'annexe 2.

Cependant, il était nécessaire de revoir l'aspect convivial du prototype APSYST pour avoir un progiciel commercialisable.

La réorganisation d'APSYST a été basée sur des remarques des utilisateurs concernant l'utilisation du prototype :

> *" Le but de l'utilisation d'APSYST est de surveiller en continu le fonctionnement d'applications TP sur un DPS 8. Il faut pour cela prendre le maximum d'informations le plus souvent possible. Il faut donc limiter les possibilités d'interruption du système expert afin de ne pas manquer un événement important qui se déroulerait sur le DPS 8."*

Dans la version prototype présentée aux chapitres précédents, le même menu général permet l'exploitation du système expert, la justification de diagnostics et la construction des bases de connaissances.

Les deux dernières possibilités sont bien évidemment en contradiction avec la remarque précédente, et sont pourtant fondamentales pour l'utilisation du système expert.

A partir de là, on est passé du *prototype **APSYST*** au *progiciel **SEAT***.

Les choix techniques d'APSYST ont été validés et fiabilisés par son utilisation pendant plusieurs mois et par les résultats obtenus.

Le passage du prototype APSYST au progiciel SEAT consistait en l'ajout des quelques fonctionnalités et en une réorganisation des différents modules développés.

Fonctionnalités nouvelles :

Enrichissement de la grammaire :

<une_cond> ::= <indice> ## <nom_indice>

Cette nouvelle condition complète les possibilités d'utilisation des variables. Il y a donc trois utilisations possibles des règles avec variable :

1 - Si la partie condition contient une variable, et aucune prémisse du type
<une_cond> ::= <indice> == <nom_indice> ou
<une_cond> ::= <indice> ## <nom_indice>,
cette règle sera appliquée sur tous les éléments du domaine représenté par la variable*<indice>* .

2 - Si la partie condition contient une variable et une prémisse du type *<une_cond> ::= <indice> == <nom_indice>* ,
cette règle sera appliquée seulement sur l'élément du domaine identifié par *<nom_indice>* .

3 - Si la partie condition contient une variable et une prémisse du type *<une_cond> ::= <indice> ## <nom_indice>*,
cette règle sera appliquée sur tous les éléments du domaine représenté par la variable*<indice>* sauf ceux cités par *<nom_indice>*.

Augmentation du nombre de séries :

Un certain nombre de faits ont été sélectionnés pour être mis sous forme de séries. Cette sélection a été faite en collaboration avec des experts du site. L'objectif de cette opération est de pouvoir évaluer un profil de charge et d'activité du DPS 8 sur une journée.

Les nouvelles séries sont :

- les temps de réponse du TP et du DPS 8
- les ressources disponibles du TP
- les nombres de transactions et TPR exécutées
- les statistiques d'accès aux bases de données
- le nombre d'utilisateurs connectés au TP
- la PRQ

Fonction d'archivage

Pour conserver les données captées par métrologie, et pouvoir disposer d'un historique des sessions du TP et du système expert, un module de clôture de session avec un archivage automatique a été développé.
(Voir SEAT mode CONSULTANT pour l'utilisation des archives.)

Synthèse graphique multi-session

Dans le prototype, seules les séries d'une même journée pouvaient être visualisées sous forme graphique. Dans le progiciel, il est prévu d'avoir sur le même écran des séries correspondant à des journées différentes.

La figure VI.1 illustre tous les modules constituants le prototype.

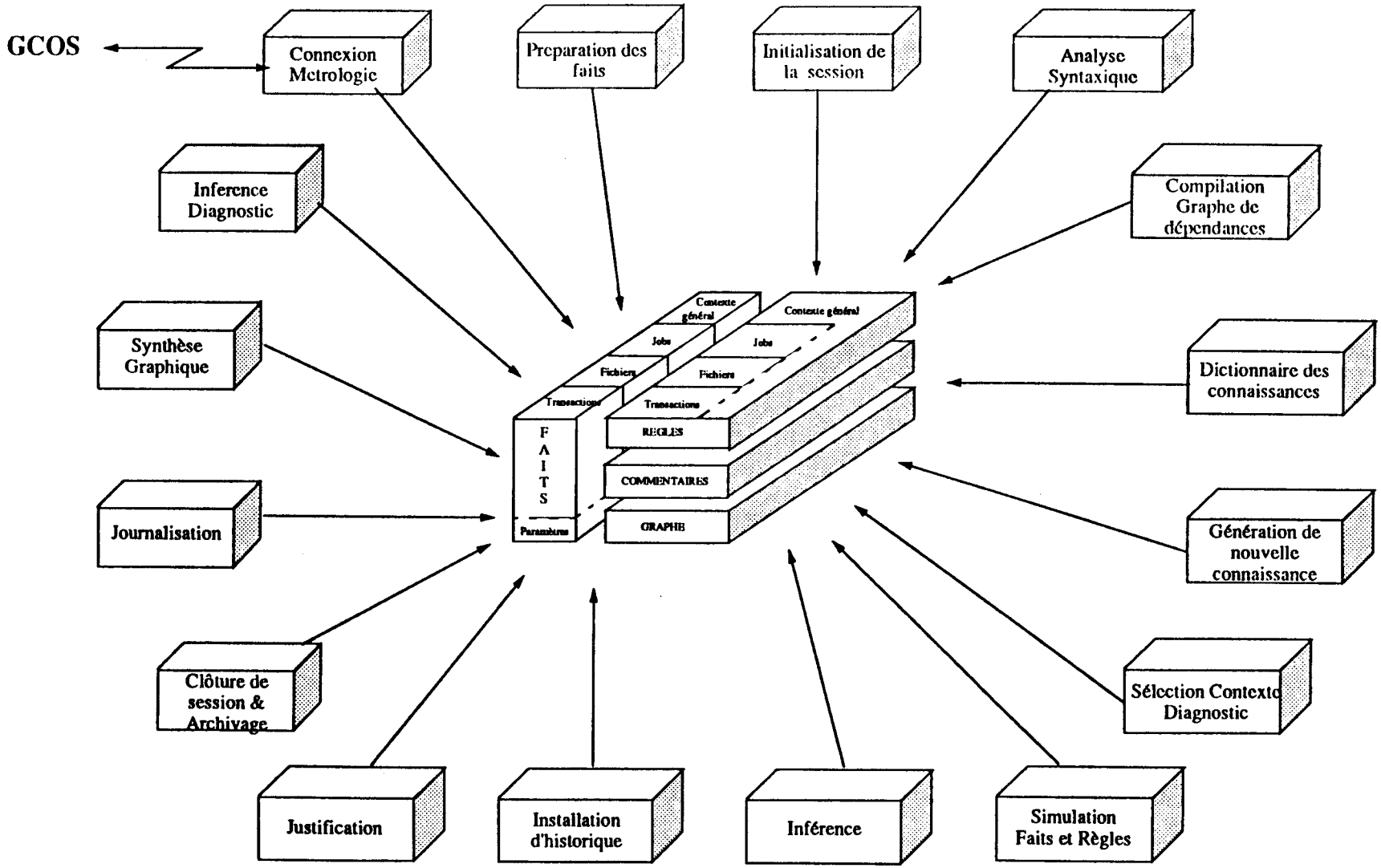


FIGURE VI.1
Des développements autour d'une base de connaissances

1) Le logiciel SEAT V1

Les besoins du progiciels étant définis, il a été décidé de découper APSYST en trois parties qui allaient constituer les trois modes de SEAT.

1.1) Le mode PRODUCTION

Ce mode correspond à l'utilisation du système expert.

L'utilisateur de SEAT PRODUCTION n'est pas obligatoirement un expert en TP ou un ingénieur système DPS 8. Il dispose d'une base de connaissances standard comportant un certain nombre de paramètres suffisant pour adapter la base de connaissances au site d'implantation.

La finalité de SEAT PRODUCTION est de contrôler le fonctionnement d'applications TP sur un DPS 8, de sortir des diagnostics en cas de dysfonctionnement et de produire un journal relatif au TP. Ce journal peut être examiné par un non spécialiste du TP ou du DPS 8.

Cependant SEAT PRODUCTION dispose d'une fonction d'archivage pour permettre d'utiliser les résultats de manière plus fine par un expert TP ou DPS 8 par exemple.

Voir figure VI.2 et VI.3

1.2) Le mode EXPERT

Le mode expert est destiné à être utilisé par des experts TP ou DPS8.

Il permet d'écrire ou de modifier la base de règles standard ou de créer des nouvelles bases de règles.

Un module a été ajouté pour ce mode :
Installation d'une nouvelle base de connaissances.

Voir figure VI.4 et VI.5

Observation, Diagnostic,Alarme,Synthese,Historique

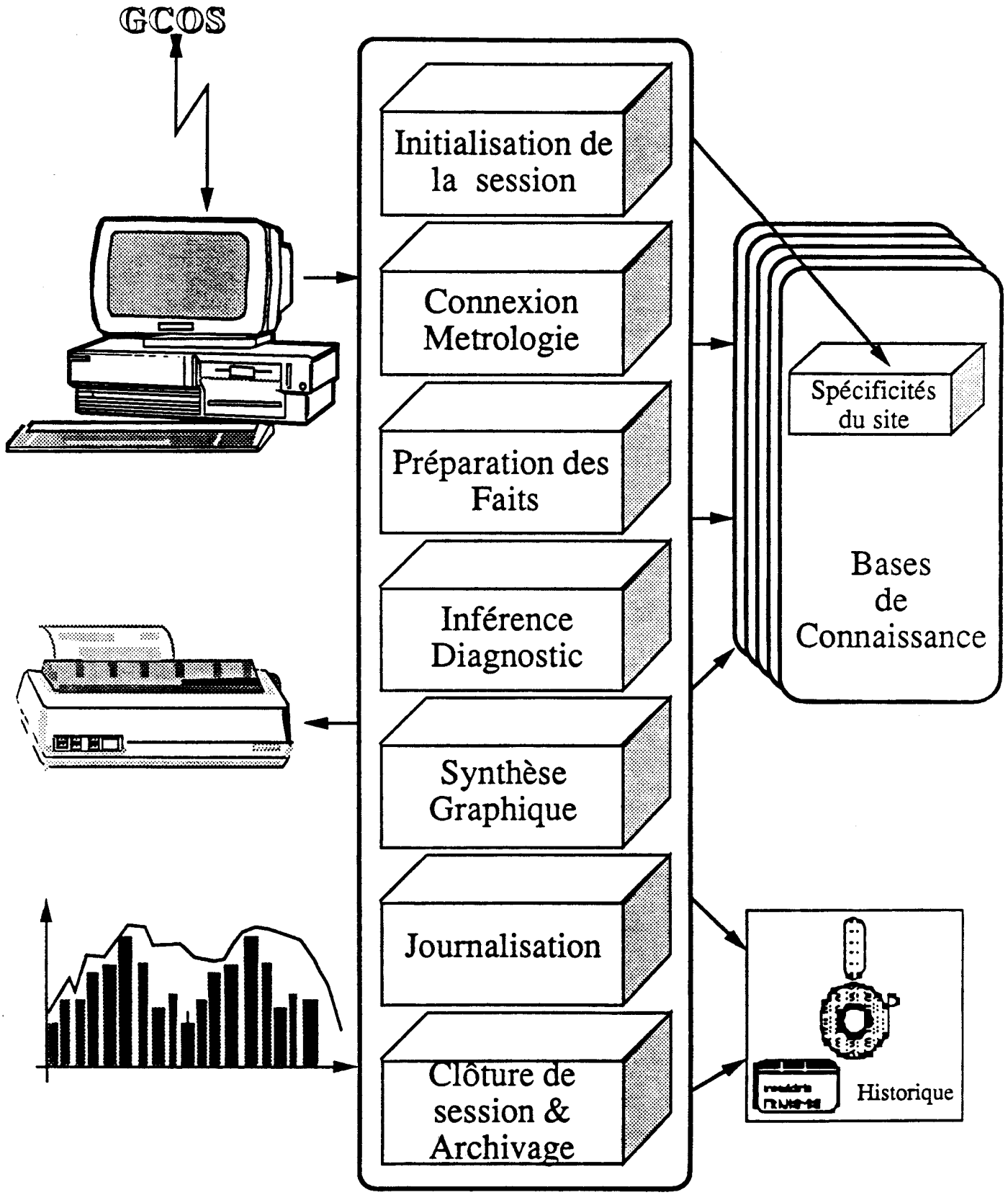


FIGURE VI.2
SEAT : le mode PRODUCTION

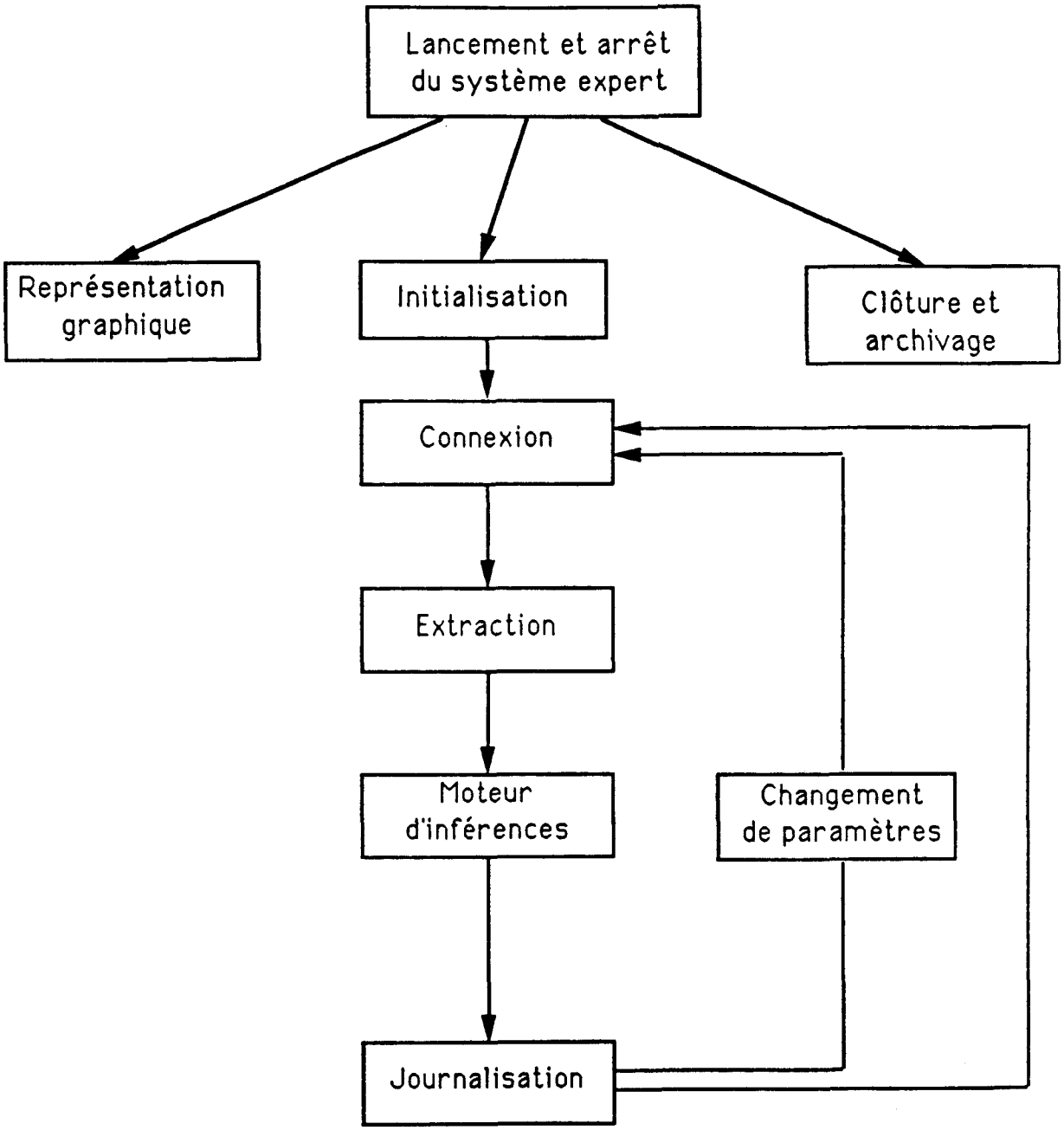


FIGURE VI.3
SEAT PRODUCTION
Version 1

Structurer,Gerer,Capitaliser le SAVOIR du SITE

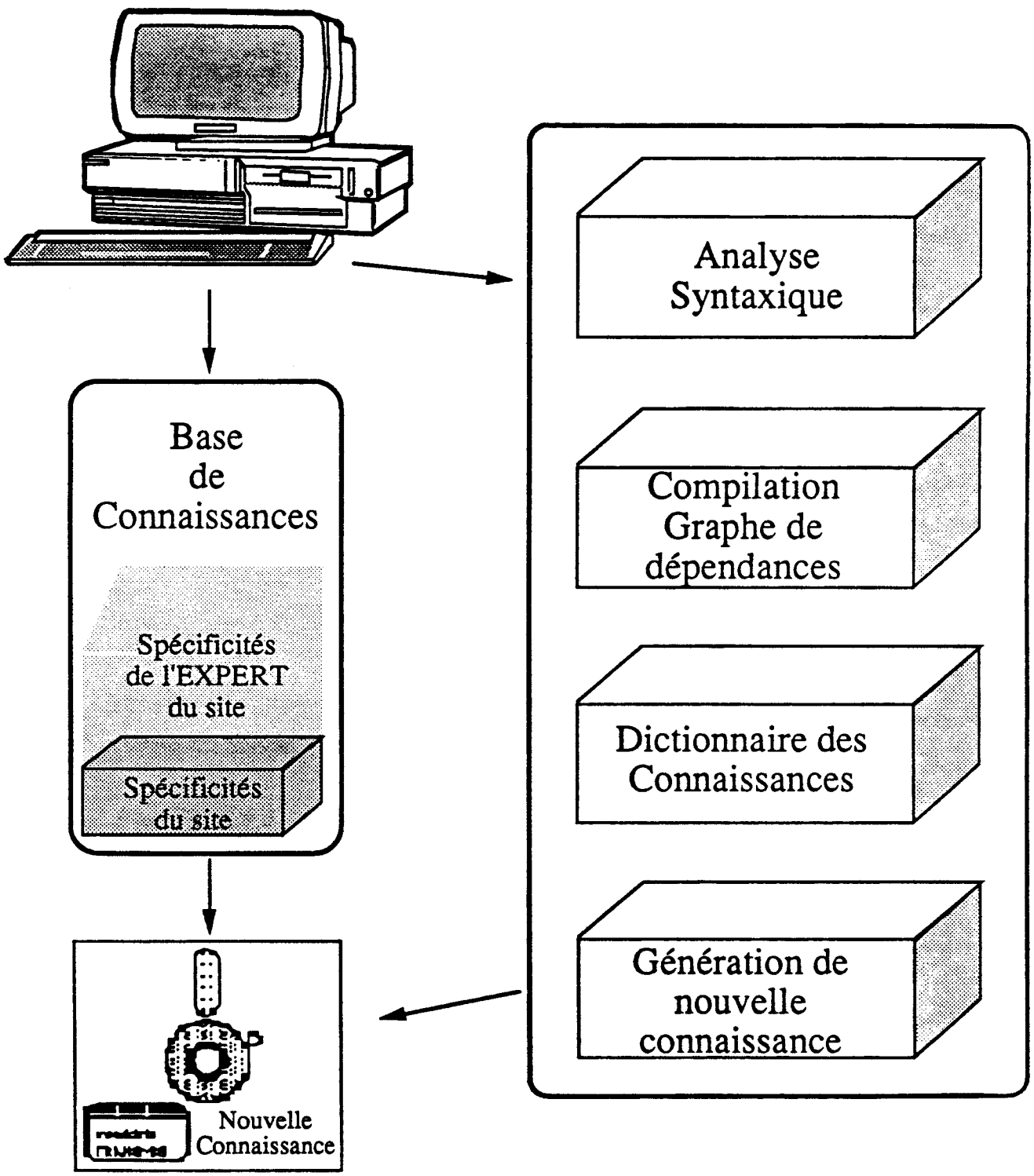


FIGURE VI.4
SEAT : le mode EXPERT

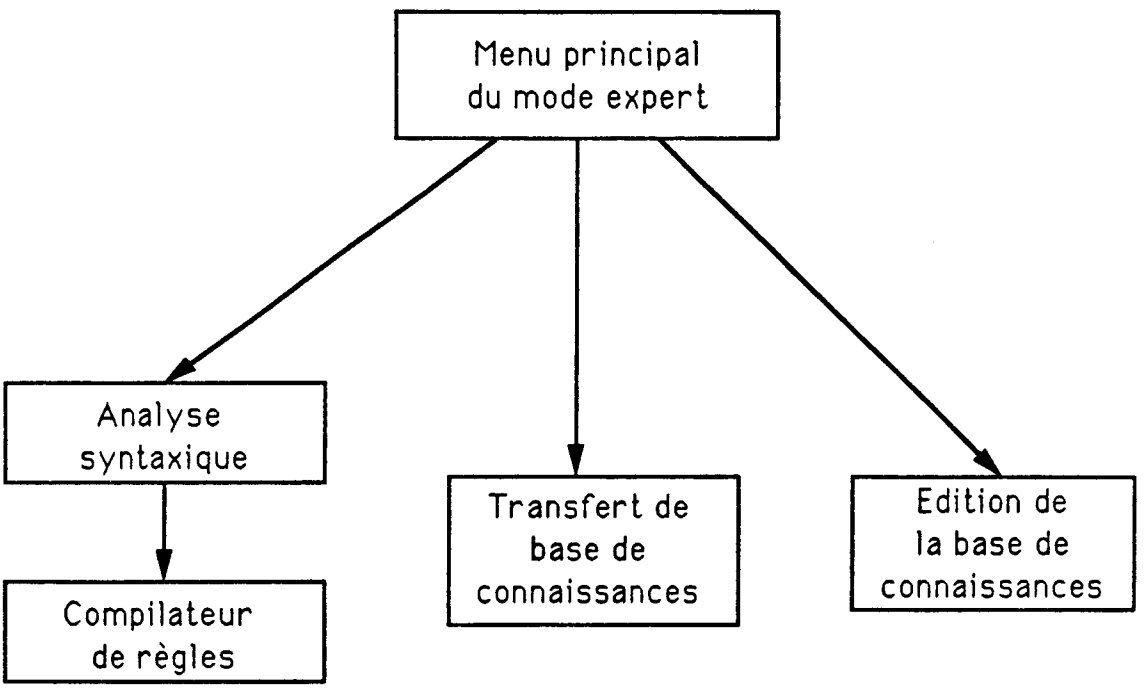


FIGURE VI.5
SEAT EXPERT
Version 1

1.3) Le mode CONSULTANT

Il s'agit d'un complément aux deux autres modes, qui est destiné à être utilisé par des experts.

Complément au mode PRODUCTION :

En utilisant les archives créées par le mode production, il est possible de reprendre les diagnostics en différé et de demander leur justification. Grâce à cette possibilité, l'expert peut analyser le raisonnement qui a engendré tel ou tel diagnostic.

C'est dans ce mode qu'a été mis le module de synthèse graphique multi-session. Il s'agit ici de gérer des séries archivées.

Complément au mode EXPERT :

Ce mode permet d'utiliser des nouvelles bases de connaissances créées en mode expert sur des métrologies réelles.
Cela offre la possibilité à un expert de tester des nouvelles bases de règles en simulant leur utilisation sur des contextes archivés.

Voir figure VI.6 et VI.7

2) De SEAT V1 à SEAT V2

La version 1 de SEAT a été utilisée pendant plusieurs mois sur le site de NATEL.

D'un point de vue statistique, SEAT PRODUCTION V1 a fonctionné pendant plus de 1000 heures et a réalisé plus de 10000 cycles.

De l'utilisation de la V1 sont nées plusieurs remarques qui ont donné lieu aux améliorations développées pour la V2.

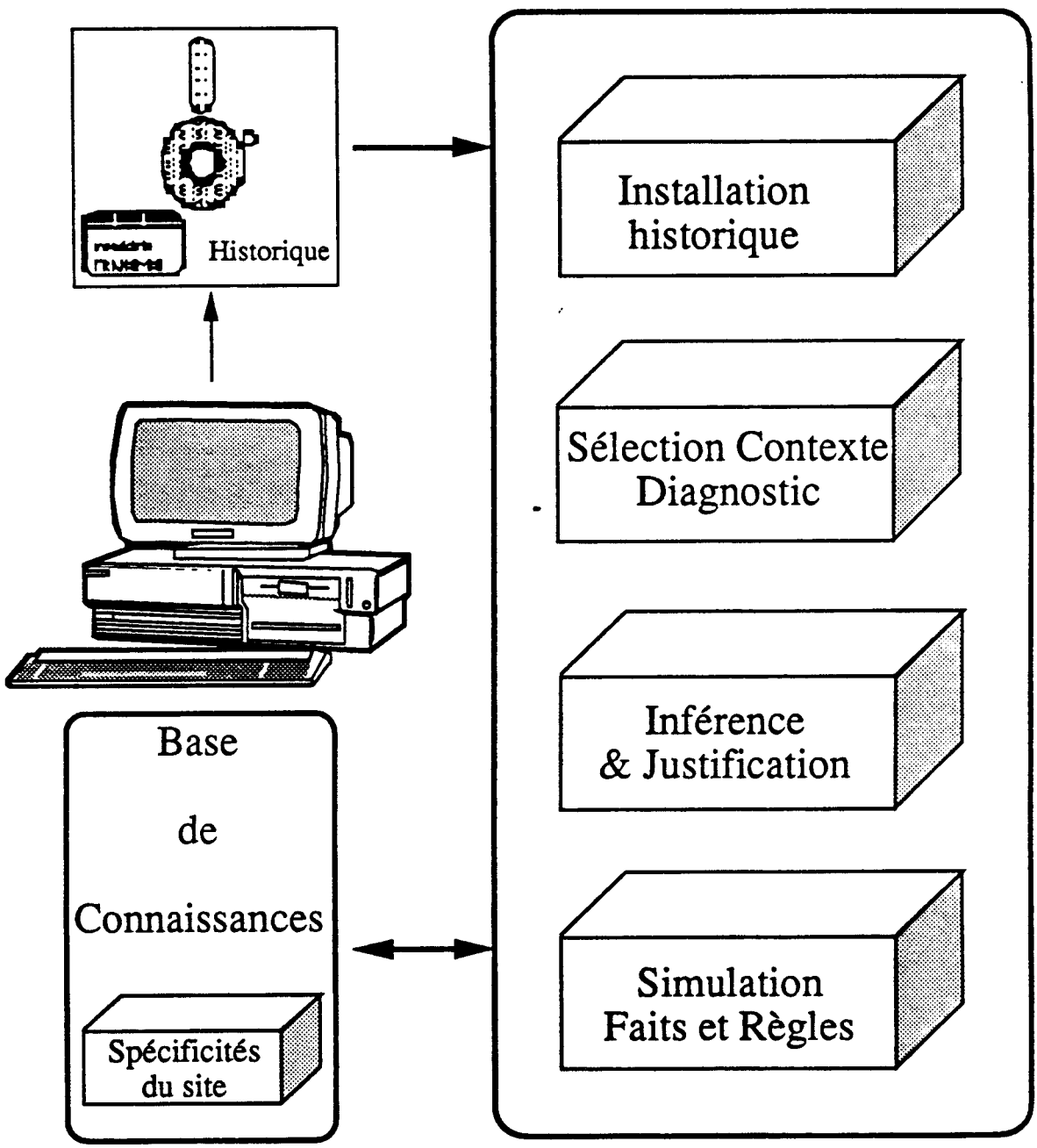


FIGURE VI.6
SEAT : le mode CONSULTANT

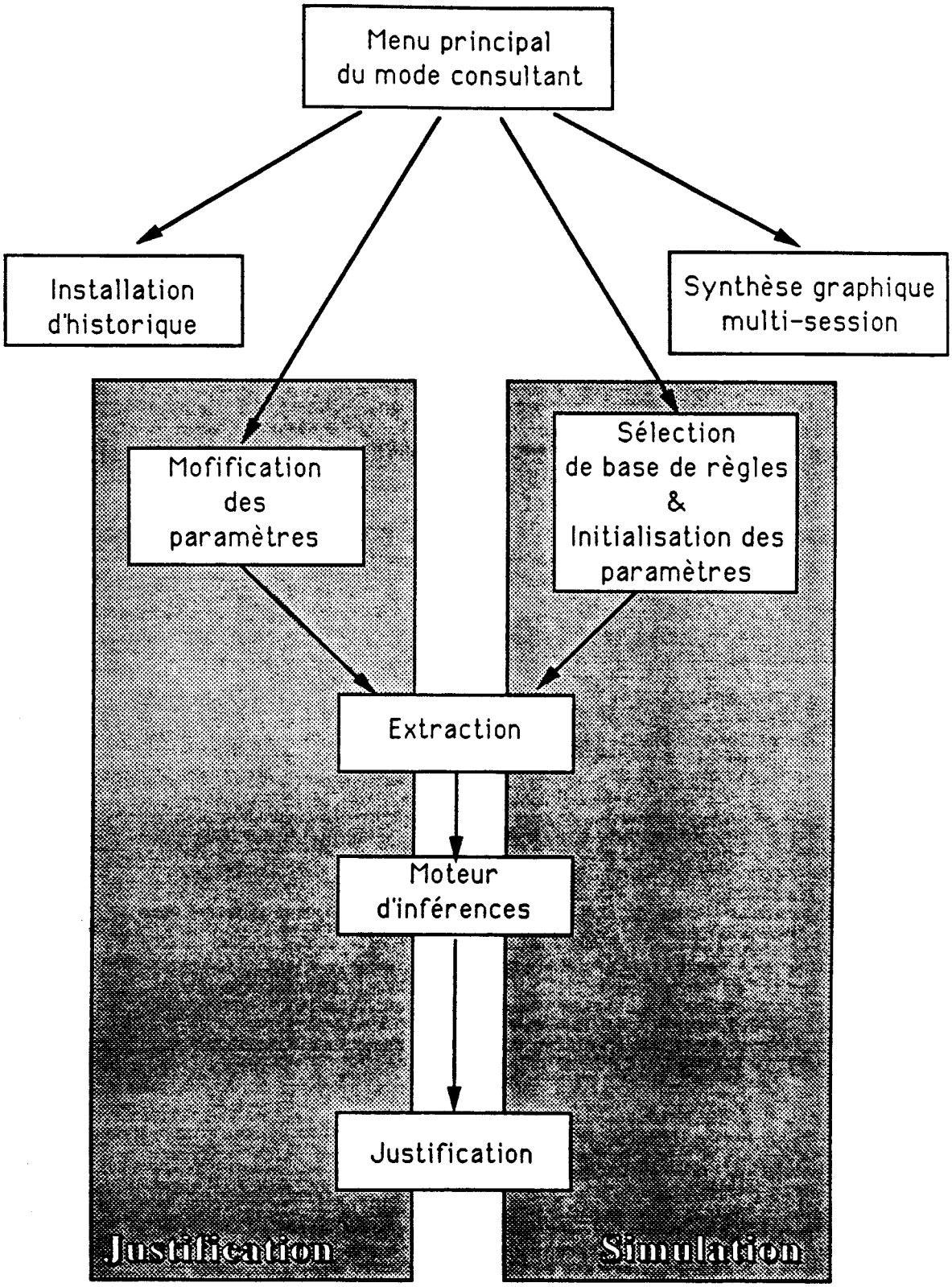


FIGURE VI.7
SEAT CONSULTANT
Version 1

Mode EXPERT

La structure du mode EXPERT reste inchangée par rapport à la V1.

Les améliorations apportées concernent l'enrichissement du langage d'expression des connaissances et portent sur les points suivants :

Les paramètres spéciaux

Il s'agit de paramètres de type booléen avec variable, qui permettre de sélectionner ou d'exclure des éléments du domaine.

Le but de cet enrichissement de la base de connaissances est de permettent à l'expert de définir des sélections et des exclusions d'éléments de domaines, mais sans nommer les éléments dans la base de règles, comme il est expliqué dans la présentation des fonctionnalités nouvelles du progiciel SEAT par rapport au prototype.
L'avantage de l'utilisation des paramètres spéciaux est que les éléments exclus ou sélectionnés dans les règles ne sont nommés qu'à l'initialisation du mode PRODUCTION et de manière dynamique.

La fonction cumulée

Il est possible dans la Version 2 de SEAT, de cumuler des valeurs au niveau de l'expertise, c'est-à-dire, de faire la somme des valeurs de faits relatifs à un même domaine ou sous-ensemble de domaine.
Le sous-ensemble de règles peut être déclaré par un paramètre spécial, et n'être construit qu'à l'initialisation du mode PRODUCTION.
Dans la grammaire on trouve en plus :

<une_concl> ::= <fait> += <fait>

Exemple :

Règle : SI T[J] ET VAL[J]=CONNU
ALORS SOM+=VAL[J]

Paramètres spéciaux
T[JOB1]
T[JOB2]

Base de faits
VAL[JOB1]=12
VAL[JOB2]=25
VAL[JOB3]=155

L'application de la règle pour chaque "J" donnera :

Base de faits
VAL[JOB1]=12
VAL[JOB2]=25
VAL[JOB3]=155
SOM=37

Les variations de faits

Tout fait réel, avec ou sans variable, dont le nom commence par **D_** est reconnu comme une variation de fait, c'est-à-dire la différence entre les valeurs d'un fait entre deux cycles consécutifs.

Exemple :
TPR_LOG_LOAD : correspond au nombre de TPR exécutées depuis le début de la session TP.
D_TPR_LOG_LOAD : correspond au nombre de TPR exécutées pendant le dernier cycle.

D'un point de vue technique, quand l'analyseur syntaxique trouve un fait ayant un nom du type **D_<nom de fait>**
il déclare **D_<nom de fait>** et **<nom de fait>** sous le même type.

Exemple :
La règle contient le fait **D_TPR_LOG_LOAD.**
L'analyseur syntaxique déclare automatiquement les fait **D_TPR_LOG_LOAD** et **TPR_LOG_LOAD** comme faits réels.

La mise sous forme de série

Tout fait réel sans variable dont le nom commence T_ est reconnu comme un fait à sérier.
C'est-à-dire qu'il faut conserver sa valeur au cours de toute la session du système expert sous la forme d'une série. Il sera possible ainsi de visualiser l'évolution de ce fait sous forme graphique.

Mode CONSULTANT

Le structure du mode consultant reste inchangée par rapport à la V1.

Les modules constituant le mode CONSULTANT ont été modifiés de façon à profiter des améliorations apportées dans les modes EXPERT et PRODUCTION.

Mode PRODUCTION

Voir figure VI.8 pour la structure du mode PRODUCTION Version 2

Les modifications concernent les points suivants :

Les variations de faits

Le module contenant la lecture des faits de métrologie et le moteur d'inférences a été adapté. Lors de l'introduction d'un fait dans la base de connaissances, dans le cas où le fait possédait une valeur lors du cycle précédent le module calcule, la différence et l'introduit dans la base de connaissances.

Les paramètres spéciaux

La liste d'éléments associés à un paramètre spécial peut être initialisée lors du lancement du mode PRODUCTION et modifiée en cours de session.

La gestion des écrans en piles

La métrologie des cycles est conservée et gérée sous forme d'une pile dont la hauteur *'n'* (0 <= *'n'* <=20) est définie lors de l'initialisation de mode PRODUCTION.
Grâce à cela, l'archivage ne contient plus seulement la métrologie relative à un dysfonctionnement décelé du TP, mais à la métrologie des *'n'* cycles précédents.

L'archivage sélectif

Le module de clôture de la session propose un menu avec plusieurs options d'archivage :

archivage total de la session
archivage partiel des métrologies
archivage des séries seulement
sauvegarde des séries

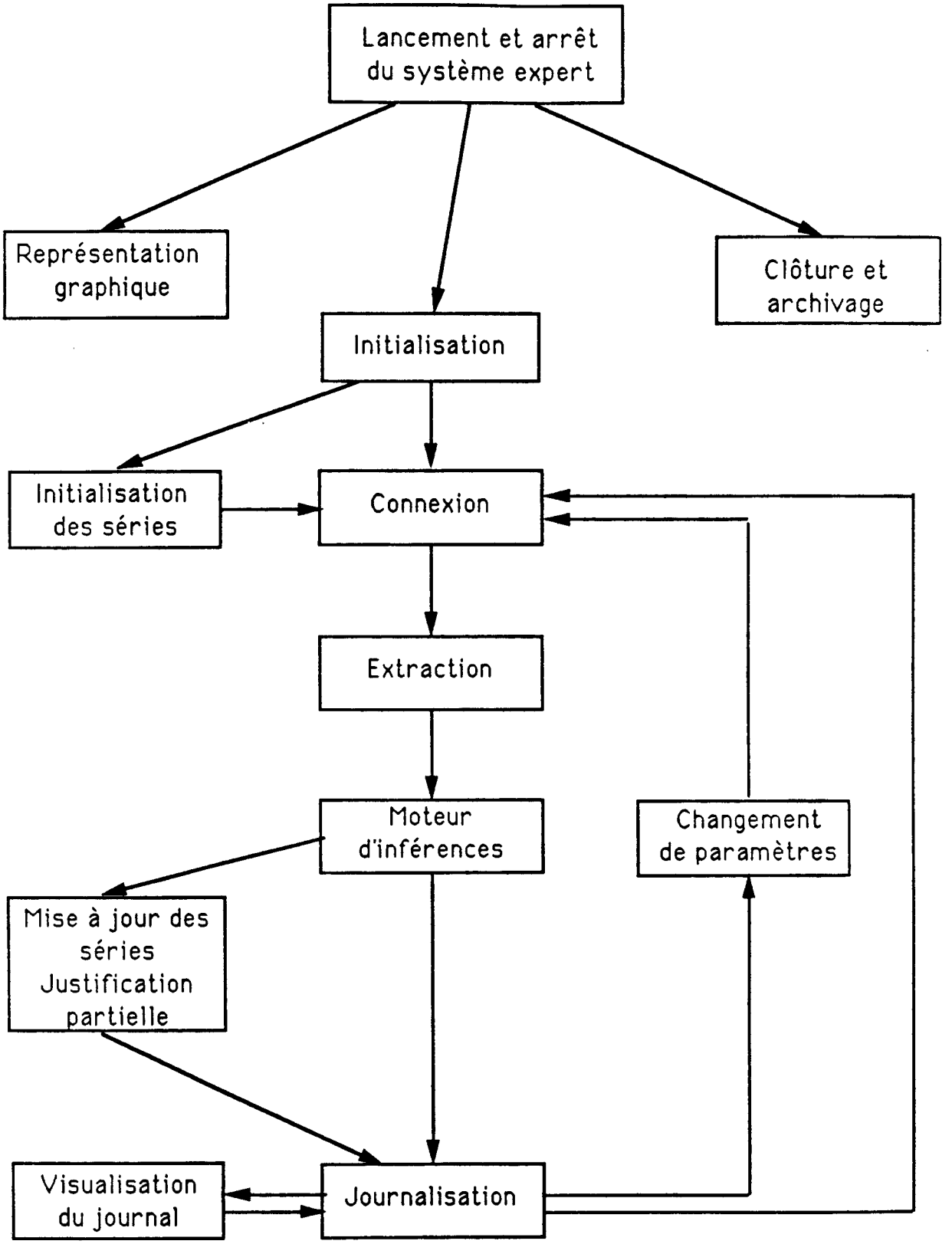


FIGURE VI.8
SEAT PRODUCTION
Version 2

Différence entre archivage et sauvegarde des séries :

- Les séries archivées doivent être réinstallées avant d'être utilisées
- Les séries sauvegardées peuvent être utilisées directement à partir d'un disquette

La visualisation du journal de SE

Un module supplémentaire a été développé pour cette fonctionnalité. Il permet d'interrompre le fonctionnement du système expert, pour visualiser l'ensemble des commentaires engendrés depuis le début de la session.
Les commentaires sont horodatés et numérotés par cycle pour faciliter la consultation du journal du SE.

Le mode "training"

C'est une option du mode PRODUCTION. L'utilisateur est libre de la choisir ou non lors de l'initialisation de la session.

Le mode training a deux rôles principaux :

- Il permet de faire une justification partielle des commentaires engendrés pendant le dernier cycle, en ne fournissant comme justification que les faits de base à l'origine du commentaire

- Il permet de sélectionner des faits que l'on désire sérialiser. L'utilisateur effectue son chois lors de l'initialisation de la session. Les séries sont complétées à chaque cycle.
A noter que tous les fait commençant par **T_** seront séries par défaut.

3) SEAT et son avenir

Le moniteur transactionnel DMIV-TP arrive en fin de vie et il faut considérer qu'aucun site DPS 8 ne fera l'acquisition d'un outil d'assistance au pilotage des systèmes transactionnels sous DMIV-TP.

Certains sites ont déjà migré de DMIV-TP à TP8, mais ils représentent moins de 30% du parc national. Par contre, la majorité des autres sont en phase de préparation à la migration vers TP8.
Actuellement, le logiciel SEAT est en version 2. Il est destiné à aider les ingénieurs système d'un site DPS 8 à préparer la migration de leur TP sous TP8.

Le concept de système expert sur micro-ordinateur permettant de surveiller un mainframe n'est pas seulement valable sur DPS 8.

Nous étudions actuellement l'utilisation de l'enveloppe du produit SEAT pour réaliser des produits du même type sur d'autres mainframes ou midframes. Les seuls modules à modifier sont la connexion et l'extraction.
Et bien sûr il est nécessaire de réécrire des bases de règles adaptées à chaque environnement.

SEAT pour IBM 30XX ?

SEAT pour IBM 43XX ?

SEAT pour AS400 ?

SEAT pour DPS 7 ?

SEAT pour . . . ?

CONCLUSION

Le produit APSYST présenté dans ce mémoire répond à des besoins dans le domaine de la production informatique. Un certain nombre de ces besoins sont exprimés tout au long de ce mémoire. On retiendra parmi les plus importants la complexité de l'environnement d'un grand ordinateur, la mouvance de cet environnement, la croissance des sollicitations (de la charge machine), les taux et qualités des services de plus en plus élevés, les problèmes de disponibilité des hommes, particulièrement des experts.

Le produit apporte des solutions à certains besoins :

- il analyse de manière systématique, et avec un "œil d'expert", le fonctionnement du DPS 8 et surtout de la production transactionnelle
- il fournit en temps réel des diagnostics concernant l'état du T.P.
- il enregistre tous les contextes sujets à diagnostics
- il crée des séries de chiffres permettant de faire des graphiques qui donnent une vue globale journalière de certains indicateurs.
- il permet de capitaliser l'expertise du site

Pour renforcer l'idée qu'un tel produit est un besoin sur le marché des sites grands ordinateurs, il faut remarquer qu'un produit aux caractéristiques fonctionnelles identiques a été réalisé aux Etats-Unis (concernant les grand ordinateurs IBM).
Il s'agit de Mindover-MVS [ADR 88]

Un système expert d'assistance au pilotage de la production transactionnelle sur DPS 8 est une originalité. Il n'existe pas à ce jour de produits concurrents.

Le produit, dans sont état en Octobre 1990, permet de faire dix à vingt cycles par heure. Ce nombre paraît relativement faible, mais il est suffisant d'après les résultats obtenus jusqu'ici (d'ailleurs aucun expert ne peut surveiller complètement son TP dix fois par heure et huit heures par jour).

Cependant, il faut relativiser ces chiffres : le système passe près de 70% du temps en connexion.

Cela est dû au temps qu'il faut pour se connecter et se déconnecter de TPMON ou de VIDEO. Il faut, pour augmenter l'efficacité du S.E., que les cycles ne dépassent pas 3 minutes : pour cela, dans la version TP8 à venir, la connexion aux différents écrans de métrologie est effectuée de manière simultanée sur une ligne mono-poste multi-session.

Pour diminuer encore le temps de cycle, une autre solution est à l'étude : ne faire qu'un seul module qui réalise la métrologie et l'inférence (en langage C).

L'évolution technologique des micro-ordinateurs améliore encore les performances du système expert. Par exemple, pour certains modules, les temps d'exécution varient d'un facteur de 1 à 6 entre un microprocesseur Intel 80286 et un 80486.

Toutes ces améliorations sont d'ordre technique sur l'implémentation. Aucun des choix théoriques retenus, sur la gestion des contradictions et l'optimisation de l'inférence n'est à remettre en cause, et les particularités de gestion de la base de connaissances que nous avons implémentées sont suffisantes pour ce produit.

Il faut ajouter que la limite du produit n'a pas encore été atteinte en matière d'expertise. L'ouverture du produit sur tous les sites par le mode expert permet à chaque site d'introduire sa propre expérience dans la base de connaissances.

L'architecture du produit été élaborée dans le but de pouvoir réutiliser la majeure partie des modules pour d'autres applications du même type, à savoir la gestion de production en général (seule la métrologie est à refaire).

La caractéristique principale du produit est la surveillance en temps réel avec génération de diagnostics avec un temps de réponse de l'ordre de quelques minutes (par opposition à la gestion de processus industriels qui demande des temps de réponse de l'ordre de la seconde).

En résumé, nous sommes satisfaits du travail présenté dans ce mémoire pour plusieurs raisons :

- Ce travail démontre que l'approche système expert temps réel fonctionnant au rythme de quelques cycles par heure est tout à fait satisfaisante et adaptée pour résoudre les problèmes de suivi de la production informatique.

- Comme nous l'avions défini au départ du projet, un moteur d'inférences en chaînage avant est suffisant et tout à fait adapté à notre problème. L'idée d'une réévaluation partielle de la base de connaissances telle que nous l'avons définie dans un souci d'optimisation et de gestion des contradictions s'est révélée efficace.

- Le choix de développer un système expert temps réel sur un micro-ordinateur externe au grand ordinateur (pour ne pas le surcharger dans les moments critiques) convient parfaitement. Cette solution peut d'ailleurs être retenue pour d'autres problèmes industriels (par exemple pour les contrôles de certains processus industriels).

- Le choix du langage Turbo-Prolog est validé par les résultats obtenus. C'est un langage de développement performant et adapté pour des réalisations industrielles importantes même sur micro-ordinateur.

BIBLIOGRAPHIE

[ADR 88] ADR - Mindover MVS
Expert System Performance Management
Plaquette de présentation 1988

[BAR 81] A. BARR - FEIGENBAUM
The handbook of Artificial Intelligence
Pitman LONDRES 1981
Traduction française - Edition Eyrolles 1986

[BEN 89] A. BENICOURT - V. CROMMELYNCK
CENTAURE : Le système expert de surveillance du réseau national de téléinformatique de la SNCF
AVIGNON 89 - Les Systèmes Experts et leurs applications

[BOY 86] G.A. BOY, NASA Amcs Research Center
An Expert System for fault diagnosis in orbital refueling
AIAA 24th Aerospace Science Meeting 1986

[BRA 86] Yvan BRATKO
Programmation en PROLOG pour l'Intelligence Artificielle
Interedition 1988

[BULL 80] DPS 8 - Data Management IV
Data Base Administrator
Bull 1980 Volume DF 77

[BULL 83] DPS 8 - Data Management IV
Concepts and characteristics
Bull 1983 Volume DP 33

[BULL 85a] DPS 8 - Data Management IV
TP Transaction Processor Reference Manual
BULL 1985 Volume DH 44

[BULL 85b] DPS 8 - Data Management IV
TP Transaction Processor Site Manual
BULL 1985 Volume DH 45

[BULL 86] DPS 8 / DPS 88 / DPS 90
GCOS 8 - Concepts and characteristics
Bull 1986 Volume DH 15

[BULL 88a] DPS 8 / DPS 88 / DPS 90
TP8 - Administrator's guide
Bull 1988 Volume DH 36

[BULL 88b] DPS 8 / DPS 88 / DPS 90
DMIV-TP to TP8 System migration
Bull 1988 Volume DH 38

[BULL 88c] DPS 8 / DPS 88 / DPS 90
TP8 - System overview
Bull 1988 Volume DH 39

[CHA 86] K.H. CHAN
Representation of negative and incomplete information in PROLOG
Sixth Canadian Conference on Artificial Intelligence
Montreal 21-23 May 1986

[CHE 86] L.CHEVALLIER - S. LE HUITOUZE - O. RIDOUX
IRISA
Style de programmation pour une machine de programmation logique munie d'un récupérateur de mémoire
Séminaire de programmation logique du CNET - 1987

[CLO 81] W.S. CLOCKSIN - C.S. MELLISH
Programming in prolog
Springer-Verlag, BERLIN

[DEA 87] T.L. DEAN - D. McDERMOTT
Temporal data base management
Artificial Intelligence Avril 1987

[DEL 87] Jean-Paul DELAHAYE - L.I.F.L. - Université de LILLE I
Organisation et programmation des bases de connaissances en calcul propositionnel
Edition Eyrolles 1987

[FAR 85] H. FARRENY
Les systèmes-experts, principes et exemples
Editions CEPADUES, 1985

[GHA 88] Malik GHALLAB
L.A.A.S., C.N.R.S., TOULOUSE
Compilation des bases de connaissances 1988

[GUP 89] Anoop GUPTA - Charles FORGY - Allen NEWELL
High speed implementation of rule-based systems
A.C.M. Transactions on Computer Systems
Vol 7 - N°2 - May 89 - p.119-146

[HAY 85] F. HAYES-ROTH
Rule-based systems
Communication ACM, 28, 9, 921-932, 1985

[KLE 86a] J. de KLEER
An assumption-based truth maintenance system
Artificial Intelligence 28(1), 1986

[KLE 86b] J. de KLEER
Extending the ATMS
Artificial Intelligence 28(1), 1986

[KLE 86c] J. de KLEER
Problem Solving with the ATMS
Artificial Intelligence 28(1), 1986

[KOW 79] R. KOWALSKI
Logic for problem solving
Elsevier Notrh Holland inc- New-York, 1979

[LAU 86] J.L. LAURIERE
Intelligence Artificielle
Résolution de problèmes par l'Homme et la machine
Edition Eyrolles - 1986

[LEC 88a] C. LECLERCQ CdFi - NATEL
APSYST - Transaction processor tuner assistant
BERLIN 88 - Congrès HLSUA

[LEC 88b] C. LECLERCQ - L. OSSART
Documentation technique d'APSYST
CdFi NATEL 1988

[LEC 89] C. LECLERCQ CdFi - NATEL
Pilotage des moniteurs transactionnels de DPS 8/88/90 par système expert
AVIGNON 89 - Les Systèmes Experts et leurs applications

[MOO 84] R. MOORE - L.B. HAWKINSON - C.G. KNICKERBOCKER
L CHURCHMAN
A real-time Expert System for process control
1984 - IEEE proceedings 1° conference on AI applications

[MOO 85] R. MOORE - L.B. HAWKINSON - C.G. KNICKERBOCKER
L CHURCHMAN
The PICON Expert System for process control .
AVIGNON 85 - Les Systèmes Experts et leurs applications

[NEM 87] "NEMO" un outil de developpement de Systèmes Experts temps réels
S2O Developpement Document 87/067

[RAS 87] J. RASMUSSEN, K. VINCENTE
The cognitive architecture of decision support systems for industrial process control.
1987 MARCOUSSIS

[RIC 85] E. RICH
Artificial Intelligence
Mac Graw Hill Inc. , New York, 1985

[ROS 86] Gianfranco ROSSI
Uses of PROLOG in implementation of Expert Systems
New Generation Computing 1986

[SAC 86] P.A. SACHS - A.M. PATERSON - M.H.M. TURNER
ESCORT
An Expert System for complex operation in real time
Expert System 86

[SOU 87] SOURCE INFORMATIQUE
Plaquette de présentation du logiciel Super 7 - 1986

[WIN 84] P.H. WINSTON
Artificial Intelligence
Addison Wesley Company, Ready 1984

Annexe A

Exemples de résultats

TDS-SD

TP de gestion et facturation des consommations d'eau

Le 12 mars 1990

Le journal du Système Expert fait ressortir un certain nombre d'aborts de type "FATAL ABORT". L'explication est la non allocation d'un fichier base de données au TP causant des aborts de TPR
L'esplication n'est pas disponible pour le Système Expert car elle ne peut être obtenue qu'avec des priorités particulières d'accès.
A 15h09, 15h40 et 15h45, un diagnostic est sorti par le SE. Il s'agit là d'une surconsommation processeur par la transaction CEPHO.

Le 15 mars 1990

Le même phénomène que le 12 mars est mis en évidence : la transaction CEPHO consomme trop de processeur

798 ms à 09h41
675 ms à 14h24
692 ms à 14h36
735 ms à 14h42
677 ms à 14h42
155 ms à 16h21

Conclusion

La transaction CEPHO apparaît régulièrement durant la session TP tous les jours. Il y a donc un problème avec cette transaction.
Il est facile de vérifier que le phénomène de surconsommation n'est rencontré qu'avec certains mots de passe (c'est à dire certains utilisateurs). Sachant que la transaction CEPHO est un outil fourni aux clients permettant de faire de la recherche d'abonné par similitude phonétique ou par nom tronqué, l'explication probable était l'utilisation abusive de cet outil.
Elle a été vérifiée rapidement : un client faisait de la recherche d'abonnés en ne donnantr que les deux premiers caractères du nom. Ce qui engendrait une consommation excessive de processeur pour le balayage complet d'un base de données de plus de 200000 clients.

```
*----------------*
ù  A P S Y S T  ù
*----------------*

 Le 12/3/1990  8:37

 Parametres:
  NB_CANAUX_DISK = 6
  PAR_MIN_TCP_TIO = 0.25
  PAR_MAX_CONS_BASE = 0.3
  PAR_MAX_JOURN = 0.3
  PAR_MAX_SW = 0.8
  PAR_MAX_L1 = 0.15
  PAR_MAX_MSG_ATT = 0.3
  PAR_MAX_TX_ATT_TPR = 0.05
  PAR_MAX_TPR_ATT = 0.05
  PAR_MAX_TCP_TIO = 0.5
  PAR_MIN_REUSED = 0.05
  PAR_MIN_PID = 0.01
  PAR_MAX_CAC_DDLK = 15
  PAR_MAX_NON_FATAL = 15
  PAR_MAX_FATAL = 15
  PAR_MAX_RAP_PHYLOG = 0.8
  PAR_MAX_CONFLICTS = 0.01
  PAR_MAX_DDLY_REST = 0.01
  PAR_MAX_LAPSE = 10
  PAR_MAX_PROC = 100
  PAR_MAX_MSG_BUF = 3
  PAR_MAX_PRQ = 10
  PAR_MIN_IDLE = 5
  PAR_MAX_DSK = 80
  PAR_MAX_NON_CHARGE = 75
  PAR_MIN_DCB = 80
  PAR_MAX_CP_JOB = 25
  PAR_NB_BASES_MIN = 1000
  PARAM_TPS_REP_MAX_DN = 5
  PARAM_TPS_REP_MAX_TP = 5
  PARAM_TPS_REP_MAX_LID = 5
  PARAM_TPS_REP_MAX_MID = 5
  PAR_MAX_RC = 0.3
  PAR_MAX_MU_JOB = 5
  PAR_MAX_PHASE = 10
  PAR_MAX_PID = 162
  PAR_MIN_SSA = 70


-------------------------------------------------------
 Le 12/03  08:47
-------------------------------------------------------
 LE JOB TDSED CONSOMME 6.20% DE LA MEMOIRE DISPONIBLE.

-------------------------------------------------------

 LE JOB TDSMA CONSOMME 5.59% DE LA MEMOIRE DISPONIBLE.

-------------------------------------------------------

 LE JOB TDSSD CONSOMME 9.88% DE LA MEMOIRE DISPONIBLE.

-------------------------------------------------------

 LE JOB TDSAC CONSOMME 8.52% DE LA MEMOIRE DISPONIBLE.

-------------------------------------------------------

 LE JOB TDSAT CONSOMME 5.59% DE LA MEMOIRE DISPONIBLE.

-------------------------------------------------------

 LE JOB TDSAS CONSOMME 10.62% DE LA MEMOIRE DISPONIBLE.

-------------------------------------------------------


-------------------------------------------------------
 Le 12/03  09:56
-------------------------------------------------------
 LE JOB TDSAS CONSOMME 32.5% DU TEMPS CP TOTAL DE LA MACHINE.
```

Le 12/03 10:13

BASE DM : 101 CONFLITS SUR 312 ACCES LOGIQUES

Le 12/03 11:04

LE JOB TDSAS CONSOMME 32.5% DU TEMPS CP TOTAL DE LA MACHINE.

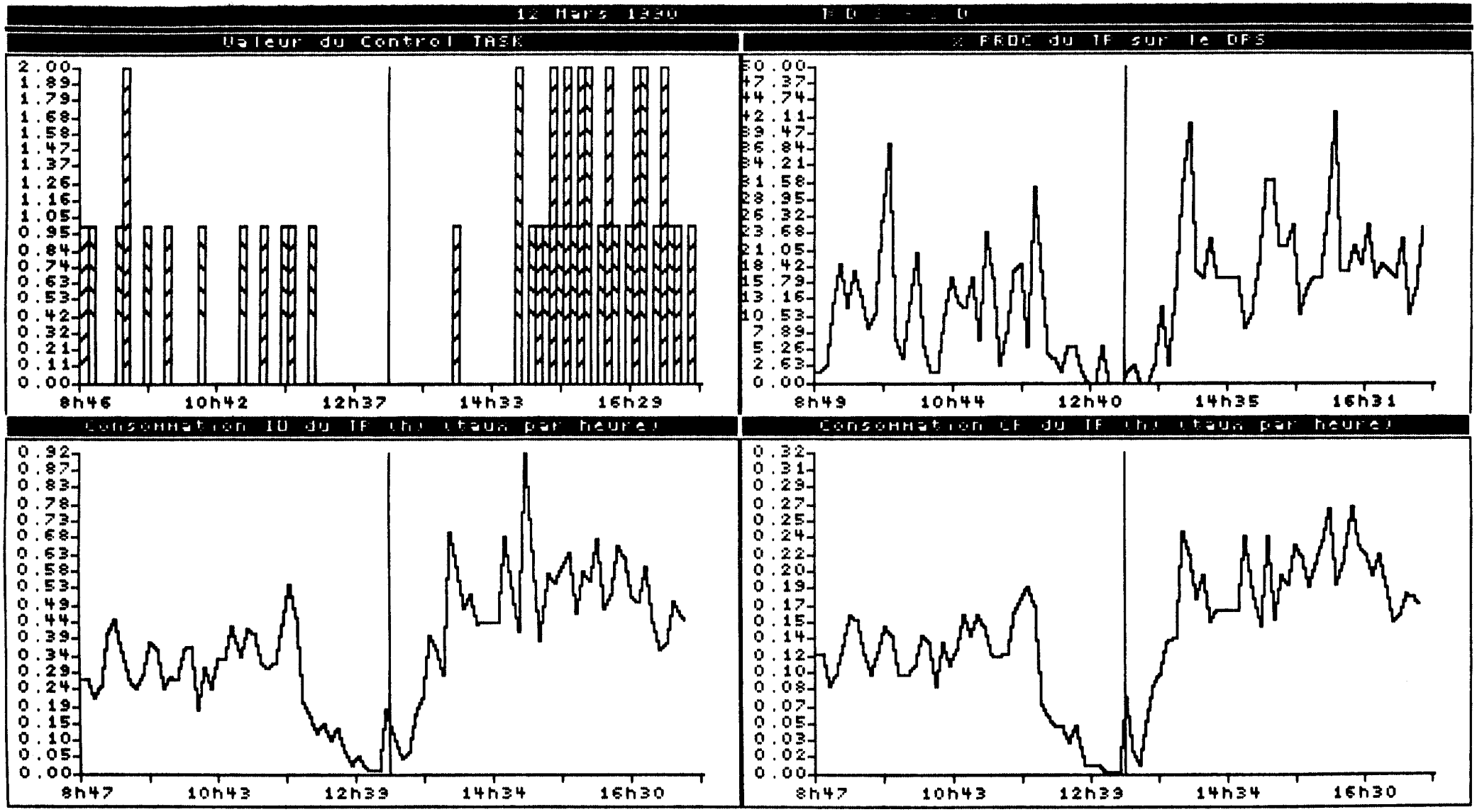
Le 12/03 11:33

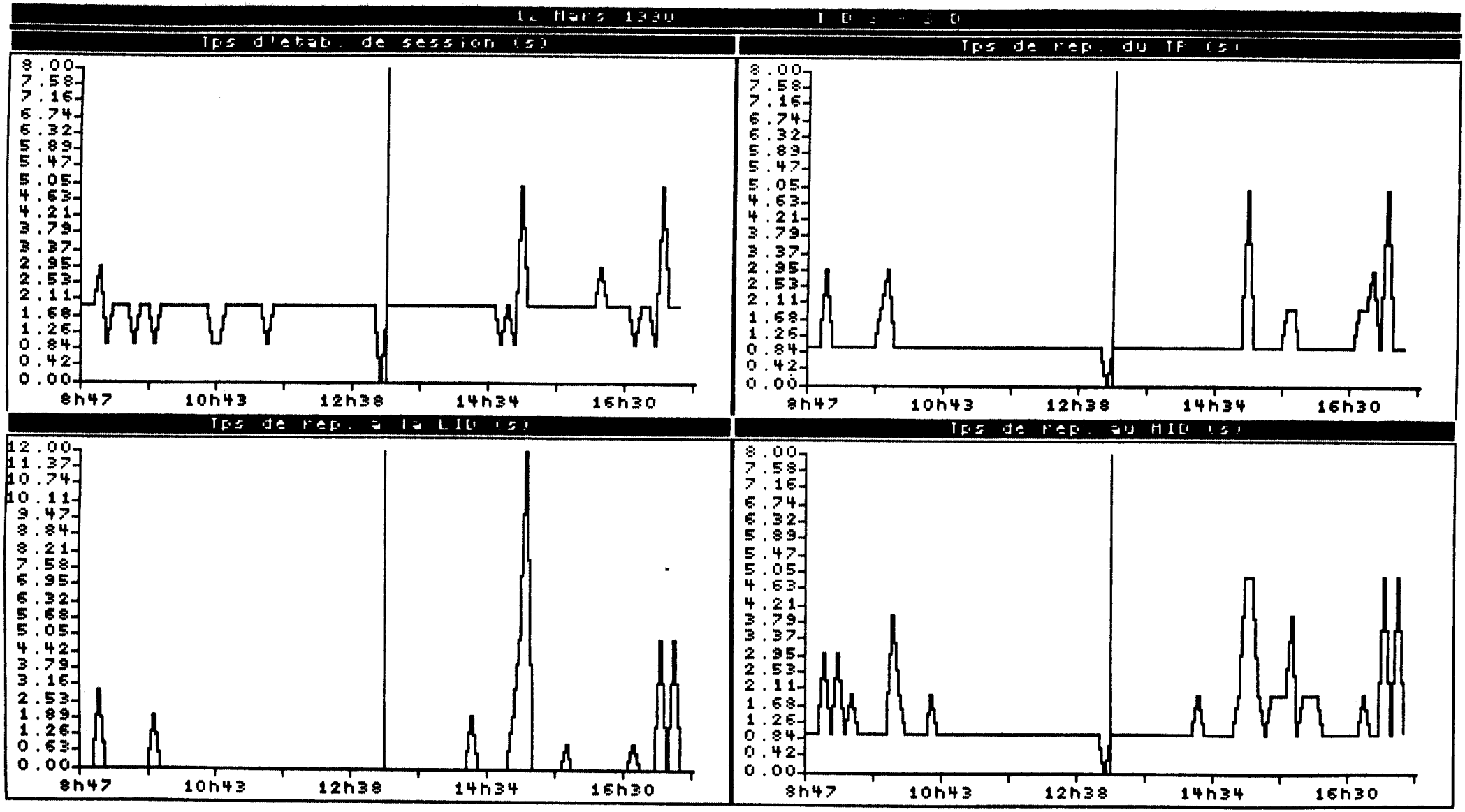
LE NOMBRE D'ABORT EST IMPORTANT :
FATAL ABORT = 16
NON FATAL ABORT = 0

Le 12/03 15:40

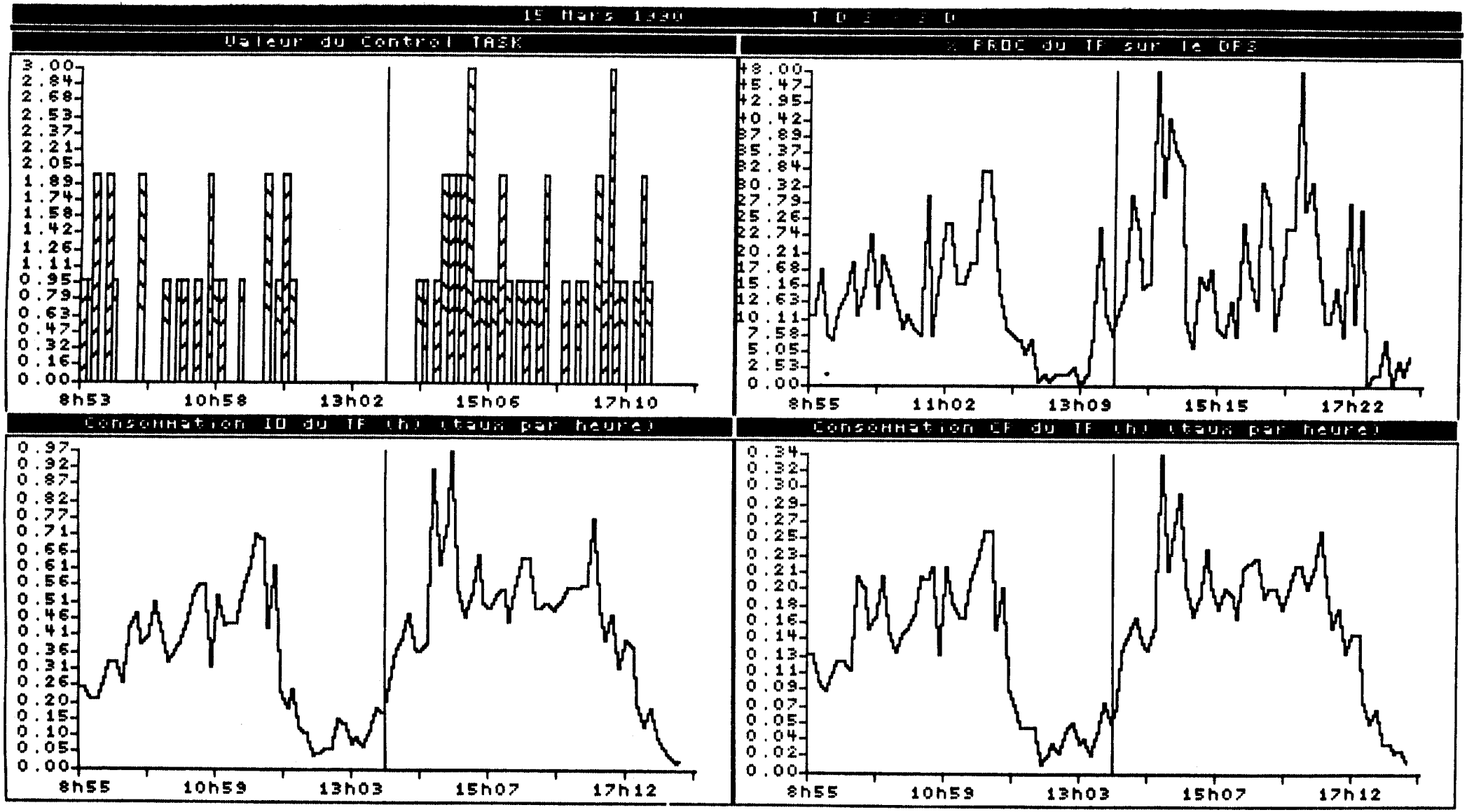
T
LA TRANSACTION 10055 EST A SURVEILLER :
ELLE CONSOMME BEAUCOUP DE PROCESSEUR (779 MS)

```
15:38: 0
   SERf/PHZf LID  MSG-ID   TPR    CORE  M.B.  D.B.    PID      PROC   TIME STATUS
> 10055/0    QJRU^CEPHO   CEPHO1  8K  0/1   4/4   24/23      706   1309 PG(E2)
  10023/3         CETRI   CETRI1  6K  0/0   4/0   25/0         0   2015 CMPLT
  10107/0         CEADM   CEABO1  7K  0/0   4/0   25/0         2   2013 CMPLT
  PRQ ENTRIES = 0
15:38: 1
> 10055/0    QJRU^CEPHO   CEPHO1  8K  0/1   4/4   24/23      706   1309 PG(E2)
  10023/3         CETRI   CETRI1  6K  0/0   4/0   25/0         0   2015 CMPLT
  10107/0         CEADM   CEABO1  7K  0/0   4/0   25/0         2   2013 CMPLT
  PRQ ENTRIES = 0
15:38: 2
> 10055/0    QJRU^CEPHO   CEPHO1  8K  0/1   4/4   24/23      732   1283 PG(E2)
  10104/1         CEFIN   CEFIN1 11K  0/0   4/0   25/0         7   2015 CMPLT
  10108/0    QJE8 CEADM   CEABO1  7K  0/1   4/4   24/11        1   2014 PG(E7)
  PRQ ENTRIES = 0
15:38: 3
> 10055/0    QJRU^CEPHO   CEPHO1  8K  0/1   4/4   24/23      739   1276 PG(E2)
   9988/4    QJF8 CEABO   CETEC1 12K  0/1   4/4   24/10        1   2014 PG(E3)
  10034/1    QJE4 CEFAC   CEFAC1 10K  0/1   4/3   24/6         2   2014 PG(E7)
  PRQ ENTRIES = 0
15:38: 6
  10055/0    QJRU^CEPHO   CEPHO1  8K  0/1   4/4   24/23      742   1273 RDY(6)
> 10109/0    QJR2 CEFIN   CEFIN1 11K  0/0   4/0   24/0         0   2013 SYS IO
  10034/1    QJE4^CEFAC   CEFAC4  9K  0/1   4/4   24/17       15   2003 PG(E7)
  PRQ ENTRIES = 2
15:38: 8
> 10055/0    QJRU^CEPHO   CEPHO1  8K  0/1   4/4   24/19      779   1236 PG(E2)
  10086/5    QJX1 CEFAC   CEFAC1 10K  0/1   4/3   24/6         1   2014 PG(E4)
  10111/0         CEBIB   CEBIB1  7K  0/0   4/0   25/0         0   2014 CMPLT
  PRQ ENTRIES = 0
```


12 Mars 1330
Valeur du Control TASK
2.00
1.89
1.79
1.68
1.58
1.47
1.37
1.26
1.16
1.05
0.95
0.84
0.74
0.63
0.53
0.42
0.32
0.21
0.11
0.00
8h46
10h42
12h37
14h33
16h29
% FREQ du TF sur le DFS
50.00
47.37
44.74
42.11
39.47
36.84
34.21
31.58
28.95
26.32
23.68
21.05
18.42
15.79
13.16
10.53
7.89
5.26
2.63
0.00
8h49
10h44
12h40
14h35
16h31
Consommation IU du TF (hr Utaun par heure)
0.92
0.87
0.83
0.78
0.73
0.68
0.63
0.58
0.53
0.49
0.44
0.39
0.34
0.29
0.24
0.19
0.15
0.10
0.05
0.00
8h47
10h43
12h39
14h34
16h30
Consommation CF du TF (hr Utaun par heure)
0.32
0.31
0.29
0.27
0.25
0.24
0.22
0.20
0.19
0.17
0.15
0.14
0.12
0.10
0.08
0.07
0.05
0.03
0.02
0.00
8h47
10h43
12h39
14h34
16h30


12 Mars 1990
Tps d'etab. de session (s)
Tps de rep. du TF (s)
Tps de rep. a la LID (s)
Tps de rep. au HID (s)
8h47
10h43
12h38
14h34
16h30

```
*----------------*
ù   A P S Y S T   ù
*----------------*

 le 15/3/1990  8:45

 Parametres:
  NB_CANAUX_DISK = 6
  PAR_MIN_TCP_TIO = 0.25
  PAR_MAX_CONS_BASE = 0.3
  PAR_MAX_JOURN = 0.3
  PAR_MAX_SW = 0.8
  PAR_MAX_L1 = 0.15
  PAR_MAX_MSG_ATT = 0.3
  PAR_MAX_TX_ATT_TPR = 0.05
  PAR_MAX_TPR_ATT = 0.05
  PAR_MAX_TCP_TIO = 0.5
  PAR_MIN_REUSED = 0.05
  PAR_MIN_PID = 0.01
  PAR_MAX_CAC_DDLK = 15
  PAR_MAX_NON_FATAL = 15
  PAR_MAX_FATAL = 15
  PAR_MAX_RAP_PHYLOG = 0.8
  PAR_MAX_CONFLICTS = 0.01
  PAR_MAX_DDLY_REST = 0.01
  PAR_MAX_LAPSE = 10
  PAR_MAX_PROC = 100
  PAR_MAX_MSG_BUF = 3
  PAR_MAX_PRQ = 10
  PAR_MIN_IDLE = 5
  PAR_MAX_DSK = 80
  PAR_MAX_NON_CHARGE = 75
  PAR_MIN_DCB = 80
  PAR_MAX_CP_JOB = 25
  PAR_NB_BASES_MIN = 1000
  PARAM_TPS_REP_MAX_DN = 5
  PARAM_TPS_REP_MAX_TP = 5
  PARAM_TPS_REP_MAX_LID = 5
  PARAM_TPS_REP_MAX_MID = 5
  PAR_MAX_RC = 0.3
  PAR_MAX_MU_JOB = 5
  PAR_MAX_PHASE = 10
  PAR_MAX_PID = 162
  PAR_MIN_SSA = 70

------------------------------------------------------
 Le 15/03  09:41
------------------------------------------------------
*T*
 LA TRANSACTION 1519 EST A SURVEILLER :
 ELLE CONSOMME BEAUCOUP DE PROCESSEUR ( 798 MS )


------------------------------------------------------

 9:41:37
    SERE/PHZE LID  MSG-ID   TPR    CORE  M.B.  D.B.    PID      PROC   TIME STATUS
    1529/2     QJX9"CETAR    CETAR3 10K  0/0   8/0   161/0       0   2003 SYS IO
    1519/0     QJXG"CEPHO    CEPHO1  8K  0/1   4/4    24/19    713   1302 PG(E4)
>   1532/1     QJTJ CEVAS    CEVAS4  7K  0/1   4/1    24/2       0   2015 PG(CU)
  PRQ ENTRIES =  0
 9:41:38
    1529/2     QJX9"CETAR    CETAR3 10K  0/0   8/0   161/0       0   2003 SYS IO
    1519/0     QJXG"CEPHO    CEPHO1  8K  0/1   4/4    24/17    714   1301 PG(E4)
>   1532/1     QJTJ"CEVAS    CEVAS4  7K  0/0   4/1    24/2        4   2011 PG(CU)
  PRQ ENTRIES =  0
 9:41:39
    1534/1           CETRI    CETRI3  6K  0/0   4/0    25/0        0   2000 CMPLT
>   1519/0     QJXG"CEPHO    CEPHO1  8K  0/1   4/4    24/17    735   1280 PG(E4)
    1532/1           CEVAS    CEVAS4  7K  0/0   4/0    25/0        0   2011 CMPLT
  PRQ ENTRIES =  0
 9:41:40
    1539/0     QJX8 CETEC    CEAB01  7K  0/1   4/1    24/1        0   2015 PG(CL)
>   1519/0     QJXG"CEPHO    CEPHO1  8K  0/1   4/4    24/21    750   1265 PG(E4)
    1524/2           CEFAC    CEFAC2  9K  0/0   4/0    25/0       14   2001 CMPLT
  PRQ ENTRIES =  0
 9:41:42
    1529/3           CETAR    CETAR3 10K  0/0   8/0   161/0       0   2011 CMPLT
    1519/0     QJXG"CEPHO    CEPHO1  8K  0/1   4/4    24/19    764   1251 PG(E4)
>   1534/2     QJX9 CETRI    CETRI3  6K  0/1   4/3    24/6        5   2010 PG(E4)
  PRQ ENTRIES =  0
 9:41:43
    1525/1           CEFAC    CEFAC1 10K  0/0   4/0    25/0        0   2015 CMPLT
>   1519/0     QJXG"CEPHO    CEPHO1  8K  0/1   4/4    24/23    798   1217 PG(E4)
    1540/0           CEFIN    CEFIN1 11K  0/0   4/0    25/0        6   2015 CMPLT
  PRQ ENTRIES =  0
```


15 Mars 1990
Valeur du Control TASK
3.00
2.84
2.68
2.53
2.37
2.21
2.05
1.89
1.74
1.58
1.42
1.26
1.11
0.95
0.79
0.63
0.47
0.32
0.16
0.00
8h53
10h58
13h02
15h06
17h10
% PROC du TF sur le DFS
8h55
11h02
13h09
15h15
17h22
Consommation ID du TF (h) (taux par heure)
0.97
0.92
0.87
0.82
0.77
0.71
0.66
0.61
0.56
0.51
0.46
0.41
0.36
0.31
0.26
0.20
0.15
0.10
0.05
0.00
8h55
10h59
13h03
15h07
17h12
Consommation CF du TF (h) (taux par heure)
0.34
0.32
0.30
0.29
0.27
0.25
0.23
0.21
0.20
0.18
0.16
0.14
0.13
0.11
0.09
0.07
0.05
0.04
0.02
0.00
8h55
10h59
13h03
15h07
17h12

TDS-MA

Le TDS-MA est un moniteur transactionnel de type VIDEOTEX contenant une base de données professionnelle à consulter concernant l'exportation.

Le 9 février 1990

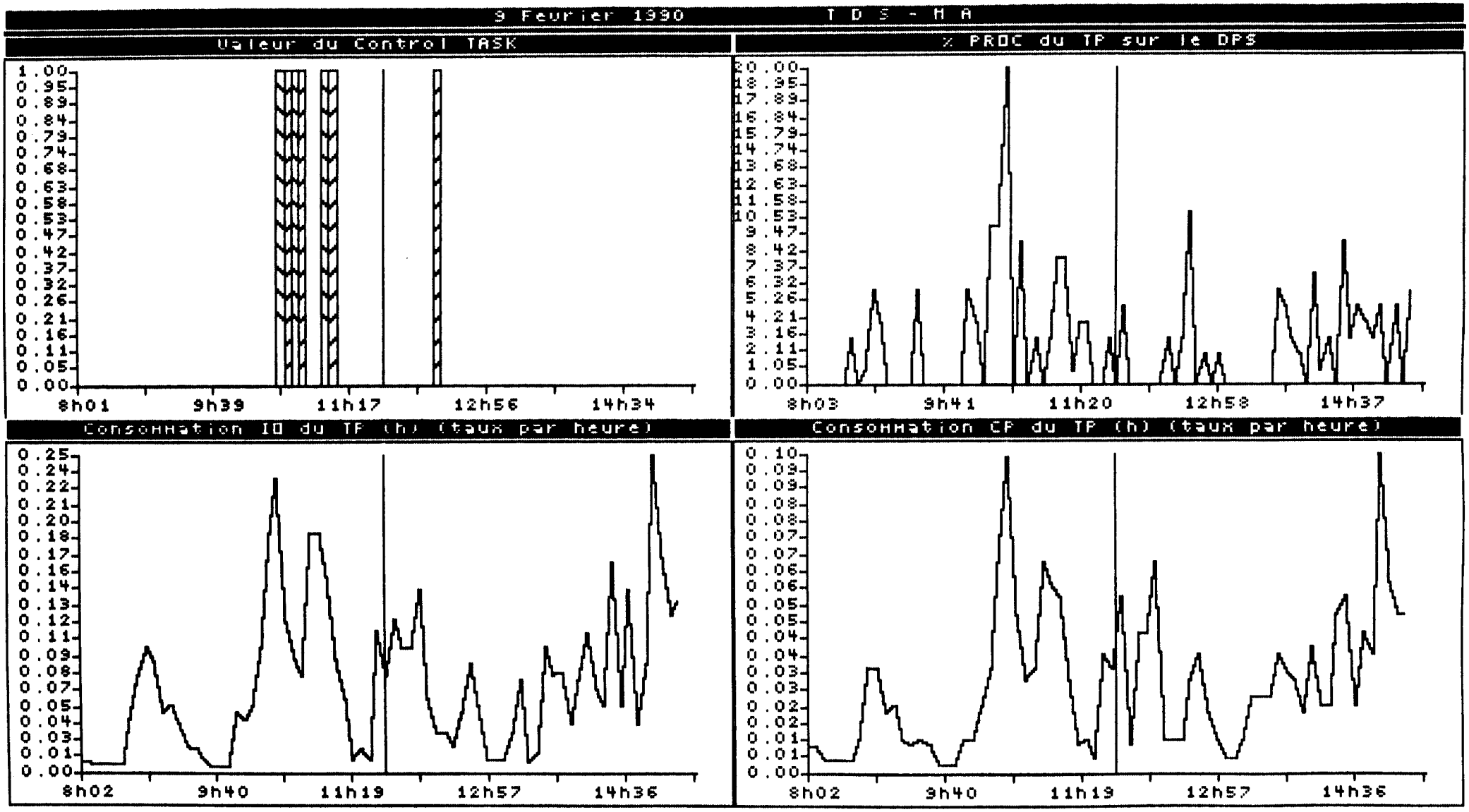
Le journal du Système Expert met en évidence des problèmes de temps de réponse durant toute la journée.

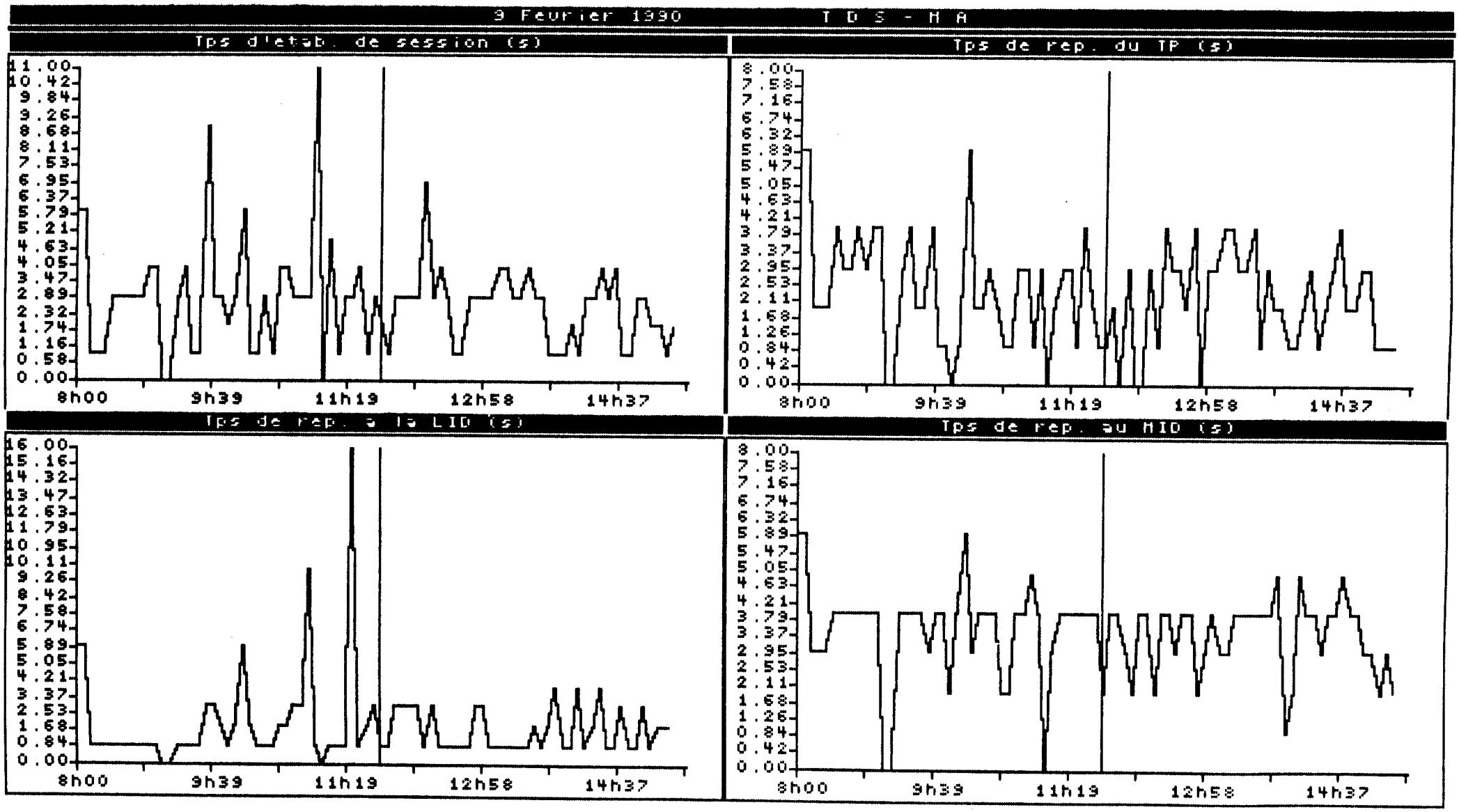
A 11h06, une TPR apparaît comme fortement consommatrice de pages Base de Données (TX=EXPOR & TPR=EX6750)

Ce moniteur transactionnel a été surveillé tous les jours pendant un mois.
Régulièrement un diagnostic était donné par le Système Expert, concernant la transaction EXPOR et la TPR EX6750 relatant une forte consommation de pages Base de données.

Suite à ces diagnostics, une étude a été réalisée sur le source de cette TPR et a révélé des erreurs de programmation.

```
+----------------+
| APSYST ù
+----------------+

le 8/2/1990  17:50

Parametres:
 NB_CANAUX_DISK = 6
 PAR_MIN_TCP_TIO = 0.25
 PAR_MAX_CONS_BASE = 0.3
 PAR_MAX_JOURN = 0.3
 PAR_MAX_SW = 0.8
 PAR_MAX_L1 = 0.15
 PAR_MAX_MSG_ATT = 0.3
 PAR_MAX_TX_ATT_TPR = 0.05
 PAR_MAX_TPR_ATT = 0.05
 PAR_MAX_TCP_TIO = 0.45
 PAR_MIN_REUSED = 0.05
 PAR_MIN_PID = 0.02
 PAR_MAX_CAC_DDLK = 15
 PAR_MAX_NON_FATAL = 15
 PAR_MAX_FATAL = 15
 PAR_MAX_RAP_PHYLOG = 0.85
 PAR_MAX_CONFLICTS = 0.01
 PAR_MAX_DDLY_REST = 0.01
 PAR_MAX_LAPSE = 10
 PAR_MAX_PROC = 100
 PAR_MAX_MSG_BUF = 3
 PAR_MAX_PRQ = 10
 PAR_MIN_IDLE = 5
 PAR_MAX_DSK = 80
 PAR_MAX_NON_CHARGE = 75
 PAR_MIN_DCB = 80
 PAR_MAX_CP_JOB = 30
 PAR_NB_BASES_MIN = 1000
 PARAM_TPS_REP_MAX_DN = 5
 PARAM_TPS_REP_MAX_TP = 5
 PARAM_TPS_REP_MAX_LID = 7
 PARAM_TPS_REP_MAX_MID = 5
 PAR_MAX_RC = 0.3
 PAR_MAX_MU_JOB = 25
 PAR_MAX_PHASE = 10
 PAR_MAX_PID = 100
 PAR_MIN_SSA = 70


-------------------------------------------------------
 Le 09/02  11:06
-------------------------------------------------------
*T*
 244 PAGES BASE DE DONNEES ONT ETE ALLOUEES ET
 244 PAGES ONT ETE UTILISEES PAR LA
 TRANSACTION DE SERIAL NUMBER 105

 LE FONCTIONNEMENT DE LA TRANSACTION ET DE LA BASE DE DONNEES
 SONT A SURVEILLER.


-------------------------------------------------------

11: 2: 1
   SERL/PHIL LID  MSG-ID   TPR    CORE  M.B.  D.B.    PID      PROC   TIME STATUS
>   118/0    MA29 AUDIT    EX0010 13K   0/0   5/4    80/4        0   2015 SYS IO
    110/28        AUDIT    EX2000 32K   0/0   5/0    81/0        0   1876 CMPLT
  PRQ ENTRIES = 0
11: 2: 2
>   118/0    MA29*AUDIT    EX0010 13K   0/0   5/4    80/4        0   2015 SYS IO
    110/28        AUDIT    EX2000 32K   0/0   5/0    81/0        0   1876 CMPLT
  PRQ ENTRIES = 0
11: 2: 3
     96/65        AUDIT    EX0501 20K   0/0   5/0    81/0        0   2015 CMPLT
    105/68   ME12 EXPOR    EX6750 20K   0/1   5/5    80/20      20   1995 PG(PR)
  PRQ ENTRIES = 0
11: 2: 6
     96/65        AUDIT    EX0501 20K   0/0   5/0    81/0        0   2015 CMPLT
>   105/68   ME12*EXPOR    EX6750 20K   0/1   5/5    80/46      25   1990 PG(GE)
  PRQ ENTRIES = 0
11: 2:10
     96/65        AUDIT    EX0501 20K   0/0   5/0    81/0        0   2015 CMPLT
>   105/68   ME12*EXPOR    EX6750 20K   0/1   5/5   106/106     25   1990 PG(GE)
  PRQ ENTRIES = 0
11: 2:12
     96/65        AUDIT    EX0501 20K   0/0   5/0    81/0        0   2015 CMPLT
>   105/68   ME12*EXPOR    EX6750 20K   0/1   5/5   244/244     25   1990 PG(GE)
  PRQ ENTRIES = 0
```


9 Fevrier 1990
T D S - M A
Valeur du Control TASK
1.00
0.95
0.89
0.84
0.79
0.74
0.68
0.63
0.58
0.53
0.47
0.42
0.37
0.32
0.26
0.21
0.16
0.11
0.05
0.00
8h01
9h39
11h17
12h56
14h34
% PROC du TP sur le DPS
20.00
18.95
17.89
16.84
15.79
14.74
13.68
12.63
11.58
10.53
9.47
8.42
7.37
6.32
5.26
4.21
3.16
2.11
1.05
0.00
8h03
9h41
11h20
12h58
14h37
Consommation ID du TP (h) (taux par heure)
0.25
0.24
0.22
0.21
0.20
0.18
0.17
0.16
0.14
0.13
0.12
0.11
0.09
0.08
0.07
0.05
0.04
0.03
0.01
0.00
8h02
9h40
11h19
12h57
14h36
Consommation CP du TP (h) (taux par heure)
0.10
0.09
0.09
0.08
0.08
0.07
0.07
0.06
0.06
0.05
0.05
0.04
0.04
0.03
0.03
0.02
0.02
0.01
0.01
0.00
8h02
9h40
11h19
12h57
14h36


9 Fevrier 1990
T D S - M A
Tps d'etab. de session (s)
11.00
10.42
9.84
9.26
8.68
8.11
7.53
6.95
6.37
5.79
5.21
4.63
4.05
3.47
2.89
2.32
1.74
1.16
0.58
0.00
8h00
9h39
11h19
12h58
14h37
Tps de rep. du TP (s)
8.00
7.58
7.16
6.74
6.32
5.89
5.47
5.05
4.63
4.21
3.79
3.37
2.95
2.53
2.11
1.68
1.26
0.84
0.42
0.00
8h00
9h39
11h19
12h58
14h37
Tps de rep. a la LID (s)
16.00
15.16
14.32
13.47
12.63
11.79
10.95
10.11
9.26
8.42
7.58
6.74
5.89
5.05
4.21
3.37
2.53
1.68
0.84
0.00
8h00
9h39
11h19
12h58
14h37
Tps de rep. au MID (s)
8.00
7.58
7.16
6.74
6.32
5.89
5.47
5.05
4.63
4.21
3.79
3.37
2.95
2.53
2.11
1.68
1.26
0.84
0.42
0.00
8h00
9h39
11h19
12h58
14h37

TDS-ED

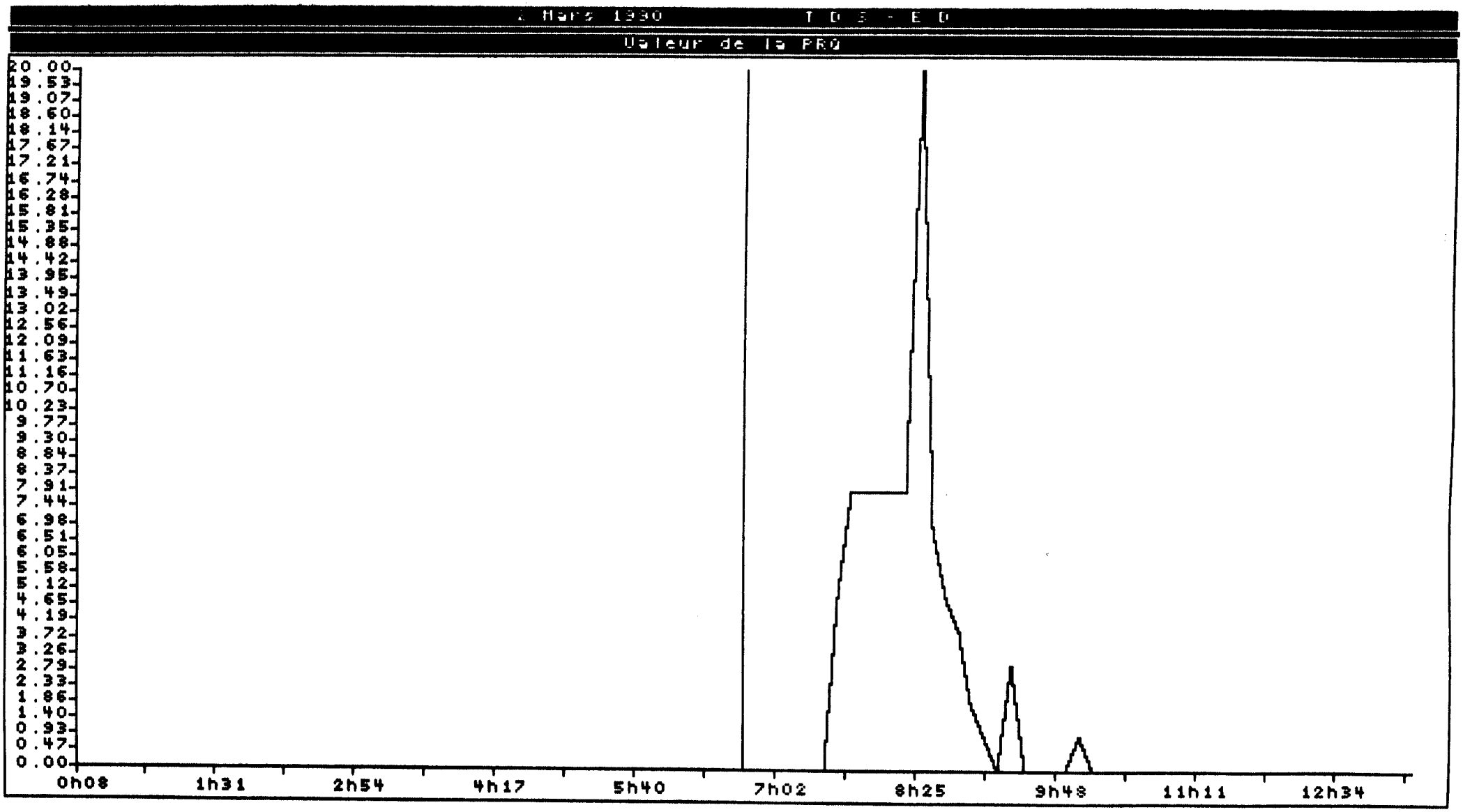
Il s'agit ici d'un TP dit "mécanique". Ses deux activités principales sont l'impression d'états et les transferts de fichier entre DPS 8 et micro-ordinateurs.

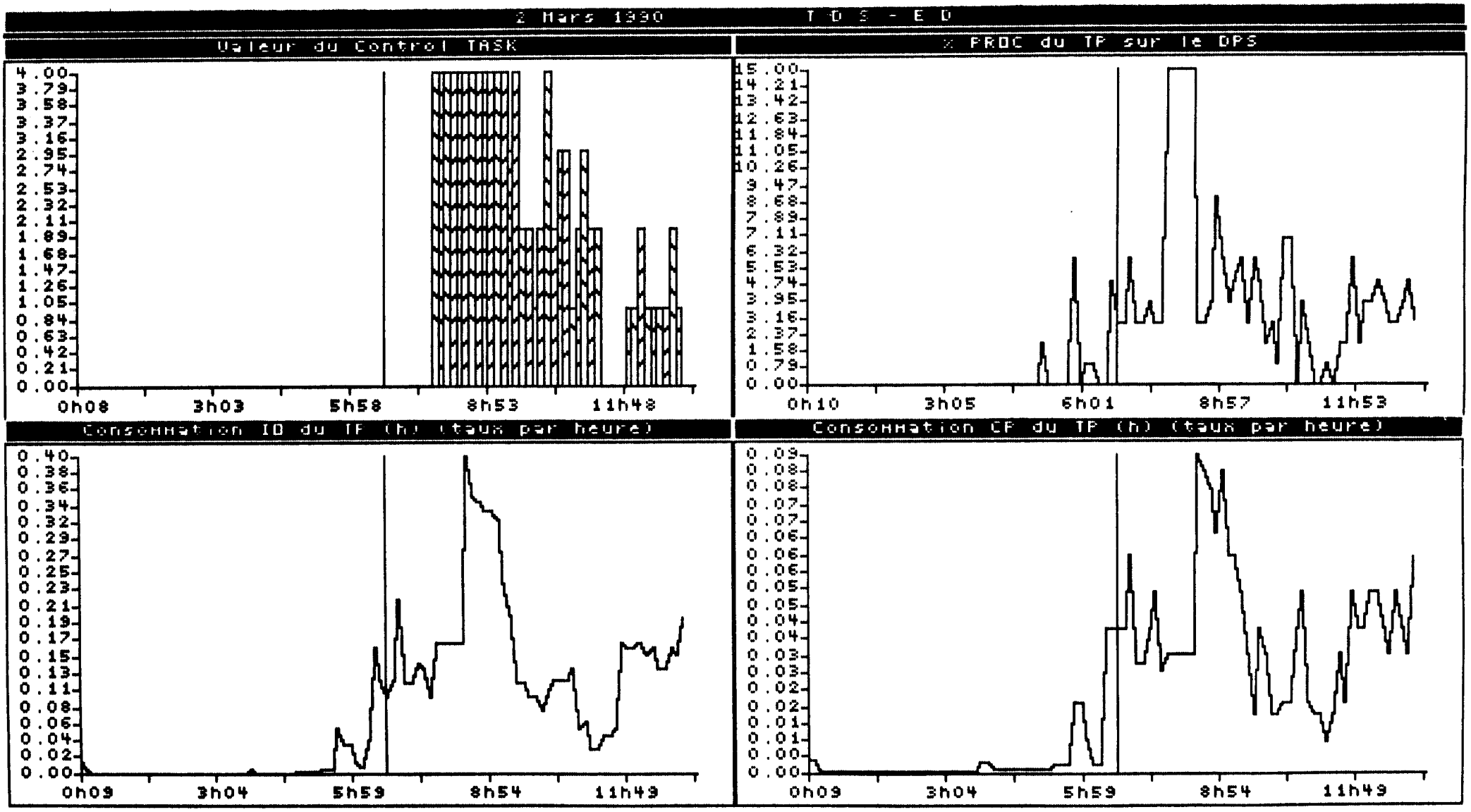
La sollicitation de ce TP est très forte en période de pointe.
L'utilisateur est soit un micro-ordinateur, soit une imprimante, et la quantité d'informations échangées est considérable.

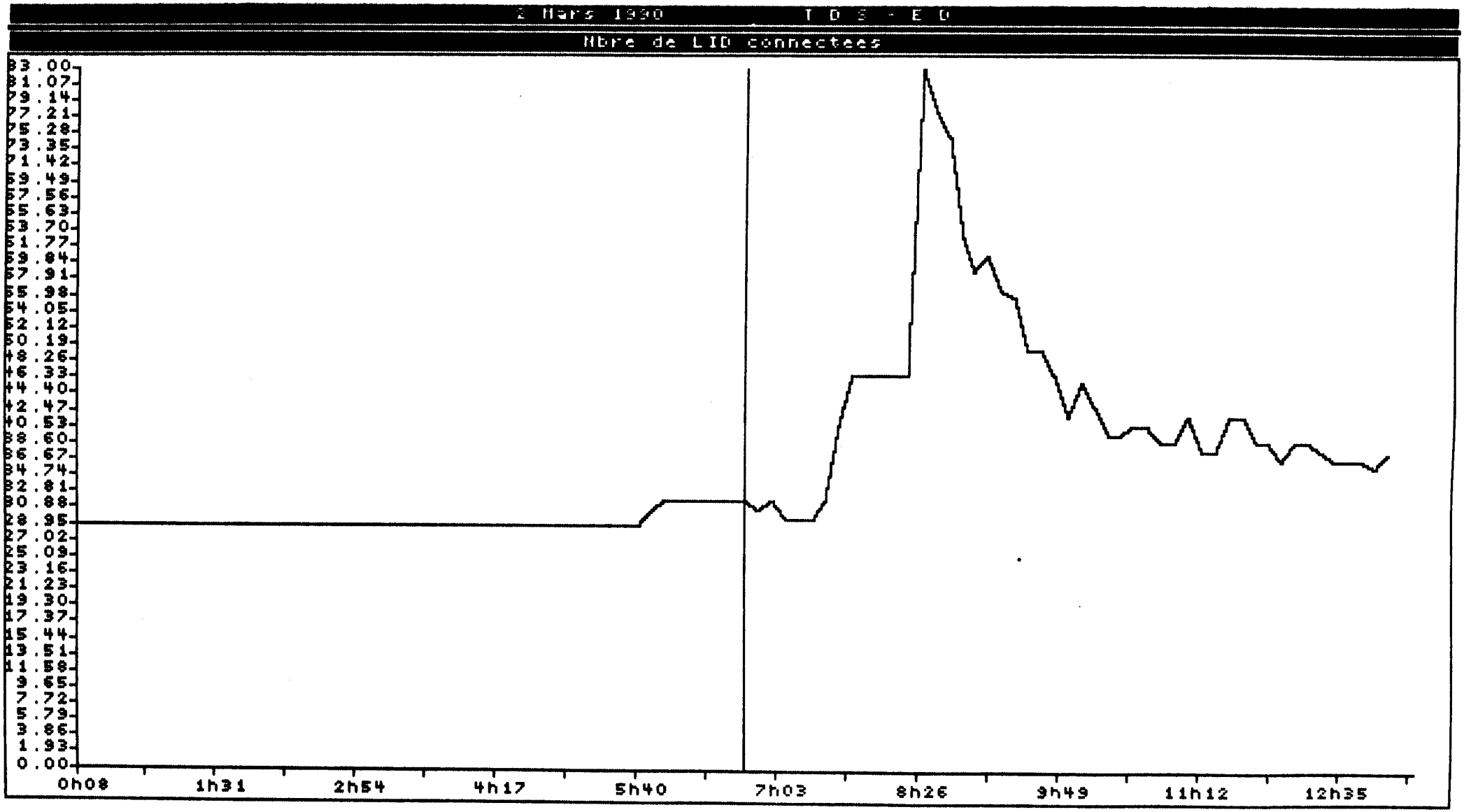
Le 1 mars 1990 (départ différé le 2 mars à 0h00)

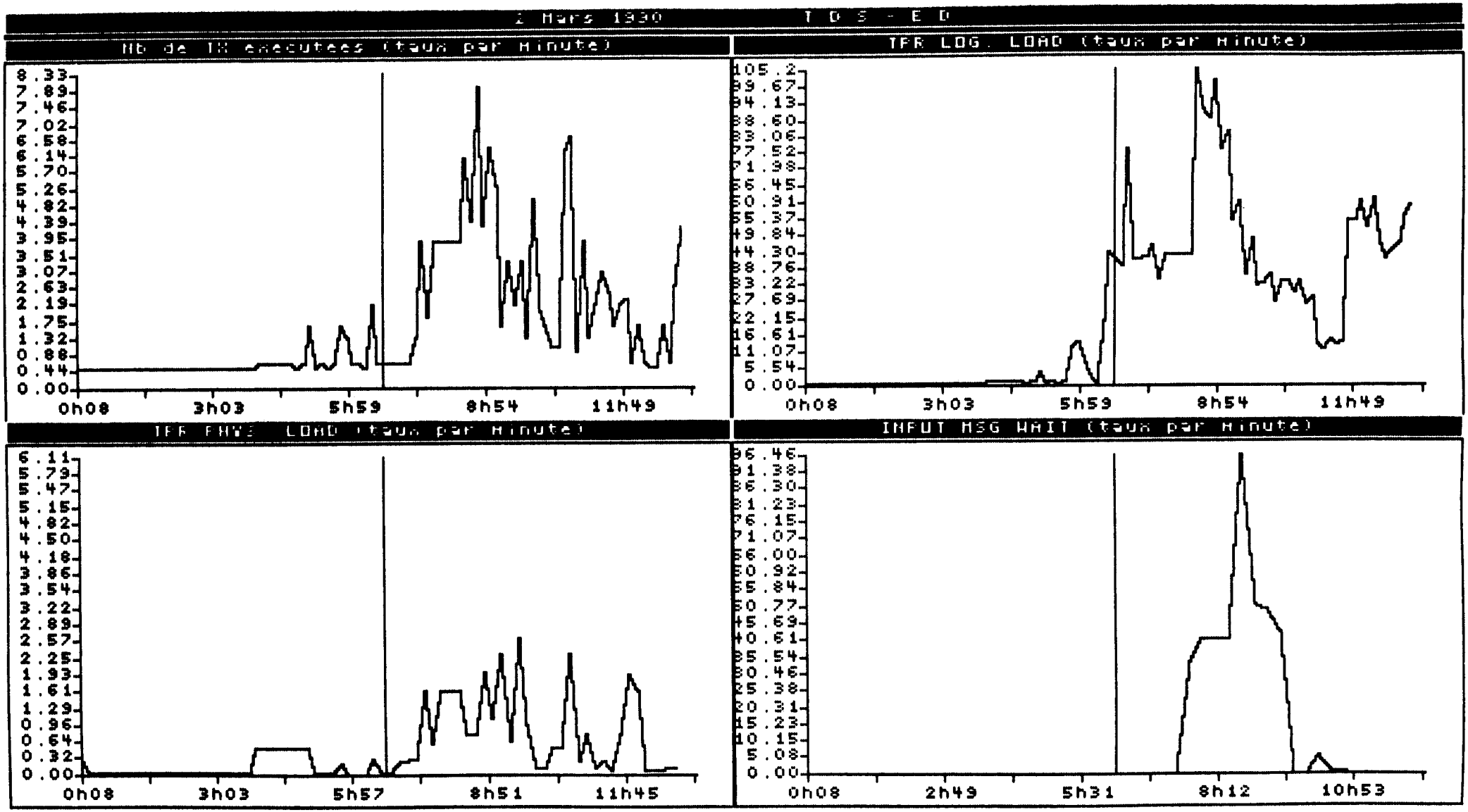
Le journal montre bien que la charge TP commence dès 7h30, là où les temps de réponse se dégradent considérablement, et diminue à partir de 9h00.
Les graphiques décrivent bien le même phénomène, particulièrement la courbe de la PRQ (les transaction en attente d'exécution).

Cette dégradation des temps de réponse n'est pas dûe à un mauvais fonctionnement du TP, mais à une très forte montée en charge du TP.


4 Mars 1990
T D E - E D
Valeur de la PRO
20.00
19.53
19.07
18.60
18.14
17.67
17.21
16.74
16.28
15.81
15.35
14.88
14.42
13.95
13.49
13.02
12.56
12.09
11.63
11.16
10.70
10.23
9.77
9.30
8.84
8.37
7.91
7.44
6.98
6.51
6.05
5.58
5.12
4.65
4.19
3.72
3.26
2.79
2.33
1.86
1.40
0.93
0.47
0.00
0h08
1h31
2h54
4h17
5h40
7h02
8h25
9h48
11h11
12h34


2 Mars 1990
T D S - E D
Valeur du Control TASK
4.00
3.79
3.58
3.37
3.16
2.95
2.74
2.53
2.32
2.11
1.89
1.68
1.47
1.26
1.05
0.84
0.63
0.42
0.21
0.00
0h08
3h03
5h58
8h53
11h48
% PROC du TP sur le DPS
15.00
14.21
13.42
12.63
11.84
11.05
10.26
9.47
8.68
7.89
7.11
6.32
5.53
4.74
3.95
3.16
2.37
1.58
0.79
0.00
0h10
3h05
6h01
8h57
11h53
Consommation ID du TP (h) (taux par heure)
0.40
0.38
0.36
0.34
0.32
0.29
0.27
0.25
0.23
0.21
0.19
0.17
0.15
0.13
0.11
0.08
0.06
0.04
0.02
0.00
0h09
3h04
5h59
8h54
11h49
Consommation CF du TP (h) (taux par heure)
0.09
0.08
0.08
0.07
0.07
0.06
0.06
0.06
0.05
0.05
0.04
0.04
0.03
0.03
0.02
0.02
0.01
0.01
0.00
0.00
0h09
3h04
5h59
8h54
11h49


2 Mars 1990
T D S - E D
Nbre de LID connectees
83.00
81.07
79.14
77.21
75.28
73.35
71.42
69.49
67.56
65.63
63.70
61.77
59.84
57.91
55.98
54.05
52.12
50.19
48.26
46.33
44.40
42.47
40.53
38.60
36.67
34.74
32.81
30.88
28.95
27.02
25.09
23.16
21.23
19.30
17.37
15.44
13.51
11.58
9.65
7.72
5.79
3.86
1.93
0.00
0h08
1h31
2h54
4h17
5h40
7h03
8h26
9h49
11h12
12h35


2 Mars 1990
T O S - E D
Nb de IX executees (taux par minute)
8.33
7.89
7.46
7.02
6.58
6.14
5.70
5.26
4.82
4.39
3.95
3.51
3.07
2.63
2.19
1.75
1.32
0.88
0.44
0.00
0h08
3h03
5h59
8h54
11h49
TFR LOG. LOAD (taux par minute)
105.2
99.67
94.13
88.60
83.06
77.52
71.98
66.45
60.91
55.37
49.84
44.30
38.76
33.22
27.69
22.15
16.61
11.07
5.54
0.00
0h08
3h03
5h59
8h54
11h49
TFR PHYS LOAD (taux par minute)
6.11
5.79
5.47
5.15
4.82
4.50
4.18
3.86
3.54
3.22
2.89
2.57
2.25
1.93
1.61
1.29
0.96
0.64
0.32
0.00
0h08
3h03
5h57
8h51
11h45
INPUT MSG WAIT (taux par minute)
96.46
91.38
86.30
81.23
76.15
71.07
66.00
60.92
55.84
50.77
45.69
40.61
35.54
30.46
25.38
20.31
15.23
10.15
5.08
0.00
0h08
2h49
5h31
8h12
10h53

Annexe B

Grammaire BNF des règles

GRAMMAIRE BNF des REGLES en NOTATION EXTERNE

<règle> ::= SI <cond> ALORS <concl>

<cond> ::= <une_cond>
::= <une_cond> ET <cond>

<concl> ::= <une_concl>
::= <une_concl> ET <concl>

<une_cond> ::= <fait>
::= "NON" <fait>
::= <fait> = "CONNU"
::= <fait> <comparateur> <valeur>
::= <fait> <comparateur> <fait>
::= <indice> == <nom_indice>

<une_concl> ::= <fait>
::= "NON" <fait>
::= <fait> := <calcul>
::= "COMMENTAIRE" <entier>

<comparateur> ::= "<" | ">" | "=" | "<>" | "<=" | ">="

<calcul> ::= <fait>
::= <valeur>
::= <fait> <opérateur> <fait>
::= <fait> <opérateur> <calcul>
::= <fait> <opérateur> "(" <calcul> ")"
::= "(" <calcul> ")" <opérateur> <fait>
::= "(" <calcul> ")" <opérateur> <calcul>
::= "(" <calcul> ")" <opérateur> "(" <calcul> ")"

<fait> ::= <fait_sans_var>
::= <fait_avec_var>

<fait_sans_var> ::= "chaîne de caractères ne commençant pas par un chiffre"

<fait_avec_var> ::= <fait_sans_var> "(" <indice> ")"

<indice> ::= "I" | "J" | "K" | "L" | "M" | "N"

<nom_indice> ::= "chaîne de caractères"

<valeur> ::= nombre entier ou réel

Remarque sur les indices dans les règles :

Un seul indice est pris en compte par règle. Si plusieurs indices différents apparaissent dans la même règle, le domaine de variation considéré pour cette règle sera celui du premier indice apparaissant, et tous les indices auront la même valeur en même temps lors de la résolution.

Titre : UN PROBLEME DE SYSTEME EXPERT TEMPS REEL
LA GESTION DE CENTRES INFORMATIQUES

Résumé : Le pilotage des centres informatiques est une tâche délicate qui ne peut être confiée qu'à un ingénieur système. Sa compétence est basée sur trois qualités : une bonne connaissance technique, une longue expérience et une capacité à trouver rapidement des solutions efficaces aux problèmes qui lui sont posés. Il est ce qu'on appelle un "expert". Au vu de ces caractéristiques, nous avons décidé de développer sur micro-ordinateur un système expert d'assistance au pilotage des centres informatiques, dont le but est de surveiller le fonctionnement d'un grand ordinateur. La caractéristique originale de ce système expert est de fonctionner en temps réel. Ceci pose des problèmes de variables à affectations multiples qui peuvent être à l'origine d'inconsistances dans la base de faits. De plus, la quantité d'informations gérées aurait été incompatible avec la contrainte du temps réel si nous n'avions pas eu recours à une méthode particulière de gestion des inconsistances et d'optimisation de l'inférence. Cette méthode, telle qu'elle a été implémentée dans le logiciel, fait l'objet d'une étude théorique détaillée dans le mémoire. Le fruit de ce travail est un progiciel commercialisé : "SEAT", dont l'utilisation a déjà fourni de bons résultats.

Mots clés : Système Expert, Temps Réel, Prolog, Intelligence Artificielle, Moteur d'inférences, Système d'exploitation, Grand ordinateur