MÉTHODOLOGIE DIABC RECENSEMENT DES CAS

TABLE DES MATIÈRES

I INTRODUCTION

Il est maintenant admis que le principal goulet d'étranglement rencontré lors du développement d'un système à base de connaissance, représentée sous la forme de règles de production, réside dans l'élucidation de cette connaissance, malgré l'aspect déclaratif "en langage naturel" de ces règles, avantage souvent mis en avant pour ces dernières, et bien que plusieurs méthodes aient été proposées (KADS, KOD, …).

Peu de choses ont été écrites sur ce sujet dans le domaine du Raisonnement Basé sur les Cas (RBC), la principale raison étant que, parmi les promoteurs de cette technique de raisonnement, certains ont été jusqu'à considérer que l'étape de l'analyse de la connaissance n'était pas nécessaire, l'expert s'exprimant "naturellement", d'après eux, en terme de cas, cas qu'il suffirait simplement de recenser.

En fait certaines questions demeurent sans réponse. Il est assez facile de répondre aux questions suivantes:

Etant donnés le domaine et la tâche de l'application,

- Quelle taille doit avoir un cas ?
- Que doit-il contenir ?
- Quels doivent être les index permettant de le retrouver ?

Les réponses à ces questions permettent de définir précisément la tâche à accomplir lors de la collecte de la Base de Cas (BC): Que doit-on collecter ?

Par contre, mises à part quelques présentations d'heuristiques exprimant simplement des règles de bon sens, peu de réponses ont été proposées pour les deux questions fondamentales suivantes:

- Comment collecter un cas ? et surtout
- Comment collecter une base de cas ?

Le propos ici est d'étudier s'il est possible de transposer une méthode issue du génie logiciel orienté objet, et en quoi cette transposition permettrait d'atteindre une "bonne couverture" du domaine par la base de cas recensés.

En introduction il est présenté un rappel de définitions liminaires, des structures de données et des résumés des algorithmes tels qu'ils avaient été implémentés dans la méthodologie DIABC (DIAgnostic Basé sur les Cas) en 1992. Ensuite des réponses aux premières questions simples sont proposées. Finalement, une méthode est préconisée pour le recensement d'une base de cas, méthode correspondant à une transposition de la méthode OOSE (Object Oriented Software Engineering) de Ivar Jacobson [JAC, 93].

I.1 Rappels et Définitions

Le Raisonnement par Analogie (RA) et le Raisonnement Basé sur les Cas (RBC) sont deux approches utilisant des solutions de problèmes résolus antérieurement. Ce sont des procédés dont le but est d'élaborer la solution d'un nouveau problème (le problème cible) en:

- retrouvant un problème antérieur (le problème base) accompagné de sa solution,
- identifiant et mesurant la similitude globale entre le problème cible et le problème base,
- adaptant éventuellement la solution du problème base pour satisfaire le problème cible.

La mesure de similitude dépend des correspondances individuelles entre les parties, caractéristiques ou aspects de la base et de la cible.

Le RBC, manipulant des cas relatifs à un seul et même domaine, est souvent considéré comme une forme plus restreinte du RA, ce dernier se caractérisant par des applications inter domaines: l'exemple en général cité est celui de la recherche de la solution d'un problème médical à partir de la solution d'un problème militaire analogue [DUN, 45].

La première implémentation du RA remonte au programme d'Evans [EVA, 68] en 1968, programme résolvant des tests d'intelligence basés sur des analogies entre figures géométriques.

Mais l'une des premières descriptions exhaustives des différentes étapes du processus n'a été publiée qu'en 1989-90 par Campbell et Wolstencroft [WOL, 89] [CAM, 90].

Les 7 étapes qu'ils ont identifiées sont les suivantes (cf. fig. 1):

1 Identification que le RA peut aider à la résolution du problème cible courant.

2 Recherche d'un problème base analogue au problème cible.

3 Elaboration ou élimination des aspects non pertinents du problème de base.

4 Mapping ou mise en correspondance des parties du problème de base avec les aspects similaires du problème cible.

5 Inférence ou déduction de caractéristiques (solutions) attribuées au problème cible à partir de celles présentes dans le problème de base.

6 Justification ou vérification du bien fondé des conclusions inférées.

7 Apprentissage, c'est à dire mémorisation du problème cible enrichi de sa solution.

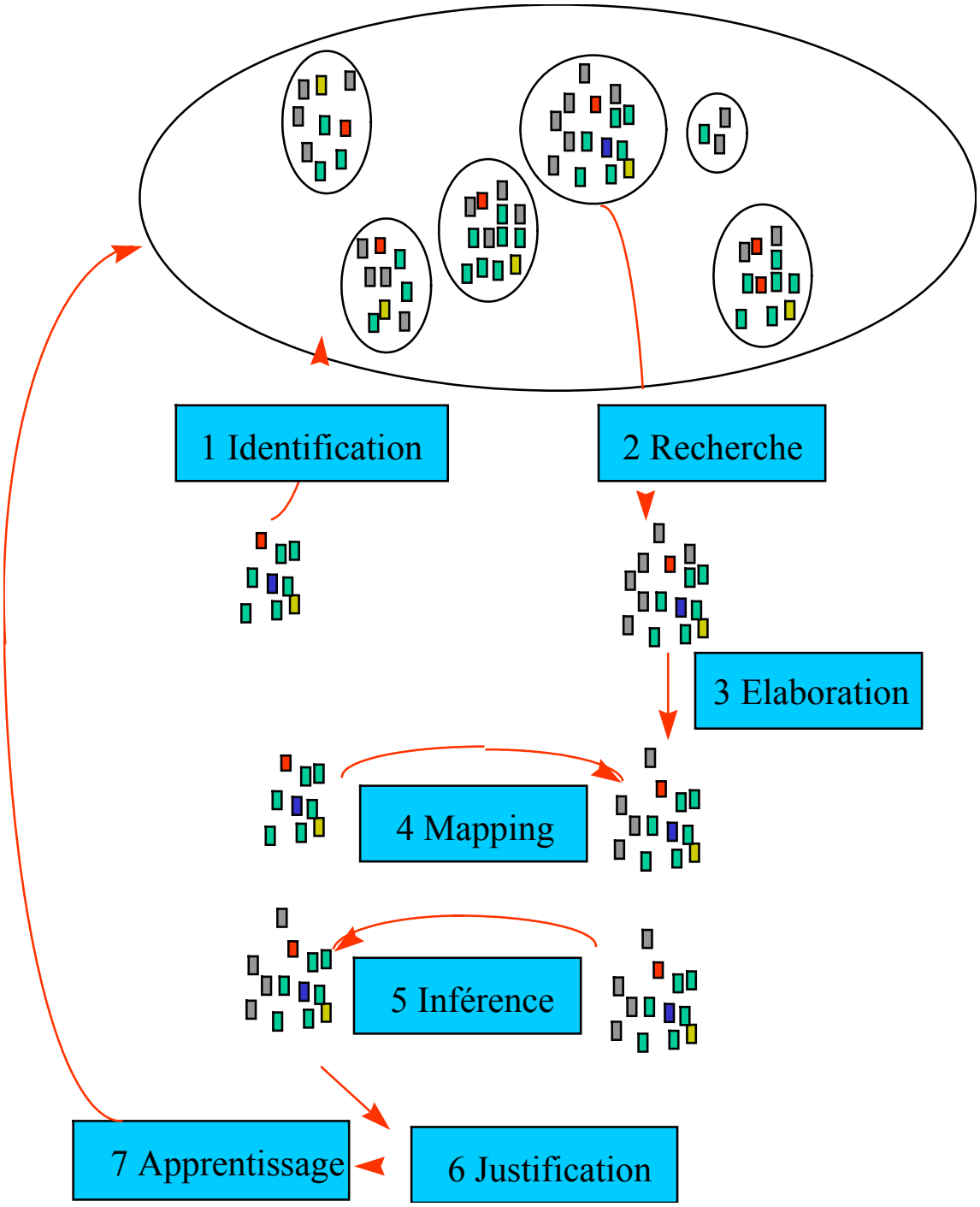


fig1: Les 7 étapes du raisonnement par analogie d'après Campbell et Wolstencroft.

I.2 La Méthodologie DIABC

I.2.A Les Paramètres et les Symptômes

Un **paramètre** est une caractéristique ou indicateur mesurable de la réponse ou sortie du système. Un **symptôme** est un couple **<paramètre, valeur ou tendance hors norme>**. A un même paramètre il peut donc correspondre plusieurs symptômes. Selon le Robert on peut distinguer plusieurs types de symptômes: "les symptômes **subjectifs** perçus et signalés par le patient, les symptômes **objectifs**, découverts par le médecin, et les symptômes **diacritiques**, liés d'une façon sûre à une maladie déterminée, qu'ils permettent de diagnostiquer". Cette distinction est reprise dans l'implémentation informatique de la méthodologie DIABC, une distinction étant faite entre:

- les symptômes facilement observables, correspondant à des paramètres directement perceptibles par tout utilisateur, dont la détermination de leur présence ou non peut ainsi être considérée comme nécessaire avant toute élaboration de diagnostic;
- les symptômes dont la présence peut être difficile et/ou coûteuse à déterminer, ces symptômes pourront être recherchés dans un second temps.

I.2.B Les Cas

Un cas (déjà rencontré ou simulé) est une entité ou objet, manipulé dans son intégralité. Les principaux champs d'un cas de diagnostic / thérapie sont les suivants:

- **une liste d'index**: c'est ici que l'on place les symptômes subjectifs et également ici que cette notion prend toute son importance, ces symptômes servent à indexer les cas et à les retrouver facilement, ces symptômes étant immédiatement déterminés par l'utilisateur;
- **une fonction test:** fonction à développer (elle doit retourner une valeur comprise entre 0 et 1, par défaut cette fonction retourne simplement 1); c'est elle qui va permettre si nécessaire de prendre en compte le contexte, la présence de symptômes infirmant le cas, etc.;
- **une fonction action**: c'est la thérapie à effectuer, pouvant consister en un simple message à afficher pour l'utilisateur ou en une fonction plus complexe à développer;
- **un champ d'explications:** destinées à l'utilisateur, elles peuvent l'aider par exemple à adapter plus précisément la thérapie au cas courant.

A la place de sa fonction action, un cas peut contenir une liste de sous-cas. Ceux-ci ont la même structure que les cas, mais c'est au niveau de leur liste d'index que l'on peut placer les symptômes objectifs et/ou diacritiques, le diagnostic nécessitant un examen plus approfondi.

Les paramètres sont également des objets contenant, outre leur nom, leur valeur normale, leurs valeurs hors norme (exprimées éventuellement sous la forme d'ensembles flous), celles qui peuvent s'annuler ou s'ajouter, permettant la détection des cooccurrences de dysfonctionnements. On peut également spécifier, au niveau du cas, un degré de nécessité de présence ou un délai d'apparition (également exprimé éventuellement sous la forme d'un ensemble flou) pour chacun des symptômes.

I.2.C Les Principaux Algorithmes

Les algorithmes sont répartis en deux groupes: le groupe pré-diagnostic, algorithmes qui ne doivent être exécutés que lors de la création de la base de cas ou lors d'une modification profonde de celle-ci, et le groupe diagnostic/thérapie.

Lors de la phase de pré-diagnostic des dictionnaires de similitude vont être créés, dictionnaires composés de toutes les combinaisons possibles des représentations internes des paramètres correspondant aux symptômes subjectifs. Aussi, afin d'éviter toute explosion combinatoire, il est nécessaire de minimiser le nombre de ces paramètres. Ceci est réalisé à l'aide d'un premier algorithme du type ID3 qui va déterminer leur "pouvoir classifiant": il est inutile de conserver un paramètre qui apparaît dans presque tous les cas ou au contraire dans presque aucun cas (un tel paramètre n'est que peu discriminatoire et ne sera que peu utile pour la recherche de cas). Un deuxième algorithme va étudier l'évolution de la répartition statistique de la base de cas (les cas présentant la même combinaison de représentations internes de paramètres sont regroupés dans un même ensemble, une classe statistique, dans un premier dictionnaire dico1) lorsque l'on enlève un de ces paramètres.

Le dictionnaire principal (dico2) est alors initialisé par la liste de toutes les combinaisons possibles des représentations internes (une lettre) des paramètres restants. Pour chaque combinaison, des liens sont établis avec les cas ou couples de cas suivants:

- les cas correspondant à cette combinaison de paramètres,
- les cas correspondant à une sous ou sur-combinaison (au sens de l'inclusion d'ensembles) de paramètres,
- les couples de cas dont la cooccurrence conduit à cette combinaison de paramètres.

Le diagnostic en lui même est composé des étapes suivantes:

- détermination des symptômes subjectifs courants, construction du mot correspondant;
- recherche dans le dictionnaire des données concernant ce mot, sélection de l'ensemble des cas pertinents à étudier;
- sélection du cas (ou du couple de cas) correspondant le mieux à la situation courante (dont le degré de correspondance avec la situation courante, ou similarité, est le plus proche de 1, similarité prenant en compte le degré de présence des symptômes, des délais d'apparition, des fonctions test et des sous-cas éventuellement définis avec leurs symptômes objectifs);
- sauvegarde du cas retenu finalement (en accord éventuel avec l'utilisateur), cette sauvegarde complète une base historique dans un souci de maintenance. Elle a une autre utilité: une thérapie ne fait pas forcément disparaître tous les symptômes immédiatement. Il est possible de préciser le temps d'action et un délai d'effet pour celle-ci: il ne faut pas alors lancer une autre recherche si les symptômes que l'on vient d'observer correspondent au fait que l'action corrective précédente n'a pas encore eu le temps d'agir totalement.

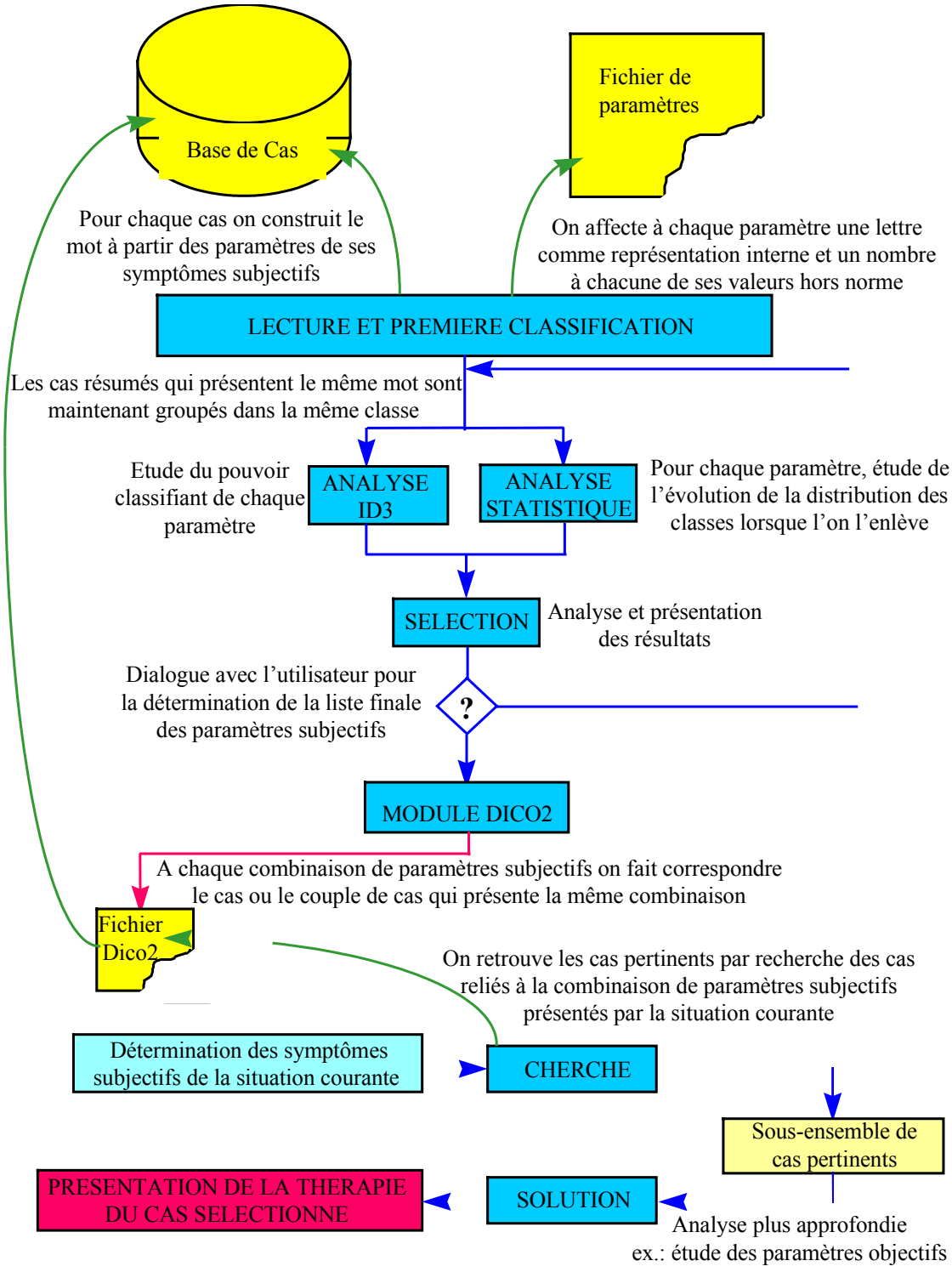


fig2. Les modules de la méthodologie DIABC

II RÉPONSE AUX PREMIÈRES QUESTIONS

II.1 Qu'est ce qu'un cas ?

Un cas est une représentation structurée d'une histoire (passée ou imaginée) composée de trois parties principales: les prémisses, le contexte et le développement avec sa conclusion.

Cette histoire doit bien sûr être lisible et affichable par la machine. De plus elle doit paraître intéressante pour l'utilisateur: d'une part elle doit lui permette d'apprendre quelque chose et d'autre part elle doit lui être présentée sous un format et une écriture qu'il comprenne et apprécie.

Le raisonnement basé sur les cas (RBC) est une technique de raisonnement utilisée principalement dans les domaines du contrôle de système (diagnostic/thérapie, surveillance, maintenance, ...), de la conception et de l'élaboration de plan. Les systèmes développés à partir de cette technique sont destinés à apporter une aide à l'utilisateur dans ces différents domaines.

Ces systèmes permettent également de **capitaliser l'expertise d'une entreprise** . Ils mettent à la disposition de toute personne habilitée l'expertise accumulée par cette entreprise.

Il faut noter également qu'un cas peut être la représentation d'une connaissance superficielle (par exemple le résultat d'une compilation d'autres connaissances) ou au contraire d'une connaissance profonde (il se peut par exemple que pour le domaine étudié ce soit le seul type de connaissance disponible: il n'existe aucun modèle mathématique, qualitatif, etc.).

II.2 Que doit-il contenir ?

La base de cas (BC) peut être considérée comme une bibliothèque dans laquelle il n'y aurait pas nécessairement de rangement ni de classement thématique ou autre. Les cas doivent donc inclure le moyen de les retrouver facilement. C'est la contrainte principale concernant la constitution d'un cas: il doit contenir des index.

Plus généralement un cas doit être composé au minimum des trois parties suivantes:

- **la description de la fonction à assumer, du problème à résoudre, ou du but à atteindre**. C'est cette partie qui doit être enregistrée dans un format tel qu'elle serve d'index. Dans le cas de systèmes dédiés au diagnostic / thérapie au sens large, cette partie ne contient généralement que des symptômes, symptômes apparaissant lors de la présence du dysfonctionnement décrit par le cas. La fonction suivante est sous-entendue: "Agir sur le système (sous contrôle) de manière à faire disparaître la liste des symptômes décrits".
- **une prise en compte de l'état initial ou du contexte** (dont le système RBC devra également vérifier la similitude avec la situation courante). Toujours dans le domaine du diagnostic, ce contexte permet de préciser les valeurs des paramètres extérieurs qu'il faudra vérifier. Appliqué au diagnostic médical il sera ainsi possible de décrire les antécédents ou le contexte familial, économique ou social. Concernant le domaine de la conception architecturale, cette partie permettra de prendre en compte différentes contraintes esthétiques, financières ou légales.
- **une proposition de solution** qui devrait être applicable au problème rencontré. Ce champ peut également contenir une présentation des risques éventuels de cette solution,

une prévision des obstacles pouvant survenir avant, après et/ou au cours de son application.
Il se peut aussi que le problème représenté par ce cas n'ait jamais été résolu de manière satisfaisante: il faut alors l'indiquer en précisant les actions qui ont été entreprises et les résultats obtenus jusqu'alors.
Puisque l'on accepte (et/ou que l'on est obligé d'accepter) qu'il n'y ait pas parfaite concordance entre le cas mémorisé et retrouvé par le système, et le problème courant à résoudre, il peut être nécessaire de modifier, d'adapter la solution, figurant dans le cas retrouvé à la situation courante. Cette adaptation peut être automatique lorsque l'on peut utiliser les techniques du raisonnement approximatif [KOS, 92] (ex.: utilisation d'inférences graduelles: plus .... plus ... [LAU, 90]) ou un ensemble de règles de production. Lorsque cette automaticité ne peut pas être atteinte, il est souhaitable d'inclure dans ce champ une aide pour guider l'utilisateur dans cette adaptation, gardant à l'esprit la complémentarité entre les capacités respectives de mémorisation de l'ordinateur et d'adaptation de l'être humain.

Pour les systèmes automatiques ou non interactifs, il est possible de limiter la composition des cas à ces trois parties, les cas étant indexés par la première partie ou fonction à assumer. Pour les systèmes interactifs d'aide à la décision, il est préférable d'adjoindre une quatrième partie narrative: l'explication destinée à l'utilisateur.

En effet la résolution de problème fondée sur le RBC n'inclut pas le déclenchement d'un enchaînement de règles de productions. La génération automatique d'une explication, par sélection intelligente des règles utilisées, est donc impossible. Cette explication, nécessaire si l'on veut préserver l'aspect formateur ou pédagogique, doit être incluse dans le cas.

D'autre part cette explication devrait être adaptable à la demande de l'utilisateur ou à son niveau d'expérience, et pouvoir consister soit en un simple rappel théorique ou pratique, soit en une description précise, complète et détaillée.

Contraintes sur le domaine et les index

L'histoire racontée par un cas décrit généralement la résolution d'un problème déjà rencontré. Pour que le système développé présente un intérêt, la première condition portant sur le domaine est que les problèmes, mis sous forme de cas et mémorisés, surviennent de nouveau.

Il faut également que le domaine présente une continuité dans le temps et l'espace. En effet deux problèmes similaires (par exemple un ancien et un nouveau problème) doivent accepter pour solutions des solutions similaires.

Ce sont les index qui doivent permettre une mesure fiable de la similitude entre cas. Ces index ne doivent ni être trop stricts (le "matching" ne sera que rarement obtenu) ni au contraire trop abstraits (ils représentent alors une contrainte de similitude trop lâche).

Un exemple d'étude de constitution des cas

Strube & al [STR, 95] ont mené une étude intéressante permettant la compréhension de la composition des cas dans le domaine de la conception architecturale, étude résumée ci-après.

Strube & al ont demandé à deux groupes d'architectes de concevoir un centre de conférences. Le premier groupe (practioner group), appartenant à un cabinet d'architectes, était constitué de deux architectes qui possédaient une expérience de plusieurs dizaines d'années dans la conception de maisons et de bâtiments, et d'un plus jeune architecte moins expérimenté. Le deuxième groupe (academic group) était constitué quant à lui de trois architectes universitaires dont la principale expérience était plus théorique que pratique, et orientée vers la conception assistée par ordinateurs.

Par observation des étapes et des techniques étudiées par les deux groupes, Strube & al ont abouti au tableau de résultats suivant:

Input category	*Practioner group*	*Academic group*
examples	2	18
cases	16	0
schemata	20	14
scripts	20	16
rules	11	12
norms	10	20
principles	7	29
requirements	40	67
prior results	53	99

Dans ce tableau, *examples* signifie solutions d'autres personnes (ex. catalogues), *cases* sont des solutions imaginées antérieurement par les mêmes architectes, *schemata* sont des structures de cas général, qui représentent une connaissance abstraite et complexe, *scripts* sont des *schemata* représentant une connaissance de séquences d'actions et stratégies, *rules* sont des prescriptions abstraites et indépendantes du contexte, *norms* réfèrent aux lois, standards, ou prescriptions administratives, *principles* désignent des valeurs personnelles, des standards et des principes d'esthétisme, *requirements* sont les demandes figurant dans le cahier des charges, *prior results* sont des solutions d'étapes de travail précédentes.

Leur conclusion est la suivante: "We may conclude that cases do not have the significance which CBR theory suggests. Compared to other kinds of input they are used rather infrequently and mostly in connection with schemata and norms. However, even a single case may be used extensively, and guide much of the design process."

On peut tout de suite remarquer concernant cette conclusion que si les deux groupes avaient eu à leur disposition un système RBC, particulièrement le groupe d'académiciens, le nombre d'appels à ce type d'aide aurait été certainement beaucoup plus important.

Mais on peut surtout déduire de ce tableau que, dans ce domaine de la conception, plus synthétique qu'analytique, certains cas doivent contenir des connaissances abstraites pour répondre aux utilisations de *schemata*, *rules*, *norms* et *principles*, et d'autres des propositions de *startegies*.

II.3 Quelle doit être sa taille ?

La troisième question évoquée à laquelle il est relativement aisé d'apporter une réponse concerne la taille que doit avoir un cas.

Un cas doit représenter un sous problème auquel l'utilisateur est typiquement confronté. Par exemple dans le domaine de la conception de bâtiments, si le cas représente tout le bâtiment, sa réutilisabilité et son intérêt seront beaucoup plus faibles que si le bâtiment est décomposé en l'ensemble de ses parties et de ses fonctions.

Une réponse pertinente à cette question proposée entre autres par Kolodner [KOL, 93] consiste à attribuer au cas la taille d'une leçon, concernant le domaine, telle que celle-ci soit aisée à enseigner et à apprendre.

II.4 Comment Collecter un Cas ?

Qu'est-ce qu'un bon cas ?

En effet il faut commencer par déterminer ce que peut être un "bon cas" avant de chercher comment le collecter. Un bon cas est un cas qui satisfait aux deux conditions suivantes:

- être retrouvé par le système RBC quand il correspond au problème posé par l'utilisateur, et uniquement à ce moment là;
- proposer la meilleure solution possible à ce problème.

Ces conditions correspondent en fait aux activités de test habituelles du génie logiciel (remarque: c'est le premier parallélisme entre l'activité de recensement d'une base de cas et les activités du génie logiciel):

- **validation**: le cas sélectionné est-il le cas correct ? (le système répond-il aux spécifications de l'utilisateur ?);
- **vérification**: le cas est-il correct ? (le système répond-il correctement ?)

Ces tests peuvent être effectués en faisant entrer par l'expert des symptômes simulés, en lui présentant les résultats, en lui demandant d'étudier les réactions d'un utilisateur non expert placé devant la même situation, et en lui demandant de certifier le cas, ou de le modifier.

Comment le Collecter ?

Un cas peut être obtenu par extraction et structuration de données déjà enregistrées sous une autre forme, et/ou le plus souvent par dialogue avec l'expert (les techniques d'interview sont rappelées au paragraphe suivant).

Il peut être intéressant d'interviewer plusieurs experts, en même temps ou successivement, mais on risque de se retrouver face à des désaccords entre experts, désaccords concernant généralement une interprétation de symptômes ou de contexte, et désaccords qu'il faudra résoudre (ex.: utilisation des techniques de raisonnement approximatif pour tenter de remédier aux différences d'interprétation, nomination de l'expert "final", etc.).

III LE RECENSEMENT DES CAS

III.1 Des exemples de ce que l'on peut lire dans la littérature

- Pour Strube & al [STR, 95] un cas est une instance d'une sous tâche réoccurrente. Ils proposent donc de décomposer le problème complet en sous tâches.

  "This can be done along the physical components involved in the overall domain task. The task-structure may reflect the structure of the physical components that occur in planning or design. Another way of factoring an overall task into sub-tasks is by focusing on reoccurring problems that are separated from other problems by having unique input or output".

La première méthode proposée ne peut être appliquée, comme nous le répéterons plus loin, que lorsque le nombre de composants est faible. La deuxième méthode est un meilleur point de départ, mais n'est pas assez développée: ils rappellent simplement plus loin dans leur texte l'existence des techniques d'élucidation de la connaissance.

- Fuchs & al [FUC, 95] proposent une représentation intéressante de la connaissance, connaissance décomposée en structures, topologie, fonctions, événements et concepts. Mais leur simple assertion:

  "There are several advantages of the CBR approach:
  - It is an alternative to other methods when the domain knowledge is weak or when we don't want to spend a lot of time for knowledge acquisition."

représente un escamotage du problème réel.

- Kolodner [KOL, 93] affirme:

  "Knowledge acquisition for a case based system is natural."

Si cette affirmation semble assez proche de la réalité par exemple dans le domaine du diagnostic médical (un médecin va assez "naturellement" évoquer le cas d'un patient avec un confrère), elle paraît un peu trop optimiste pour de nombreux autres domaines non analytiques (ex.: les architectes ne s'expriment pas naturellement en terme de cas).

Bien que dans cet article figure un paragraphe ayant pour titre "Collecting Cases: How ?", les conseils sont plutôt axés sur la question "Comment collecter un "bon" cas ?"

III.2 Les recensements simples: exemple du diagnostic

III.2.A Décomposition hiérarchique du système

L'opération de diagnostic [CHA, 83] consiste en l'identification d'une description (partielle) d'une situation avec un noeud spécifique de la hiérarchie représentant le système diagnostiqué: les niveaux les plus élevés de cette hiérarchie correspondent aux concepts les plus généraux (niveau fonctionnel), les niveaux spécifiques (niveau des composants) se trouvant à la partie inférieure. Le raisonnement affine progressivement le problème posé par une analyse

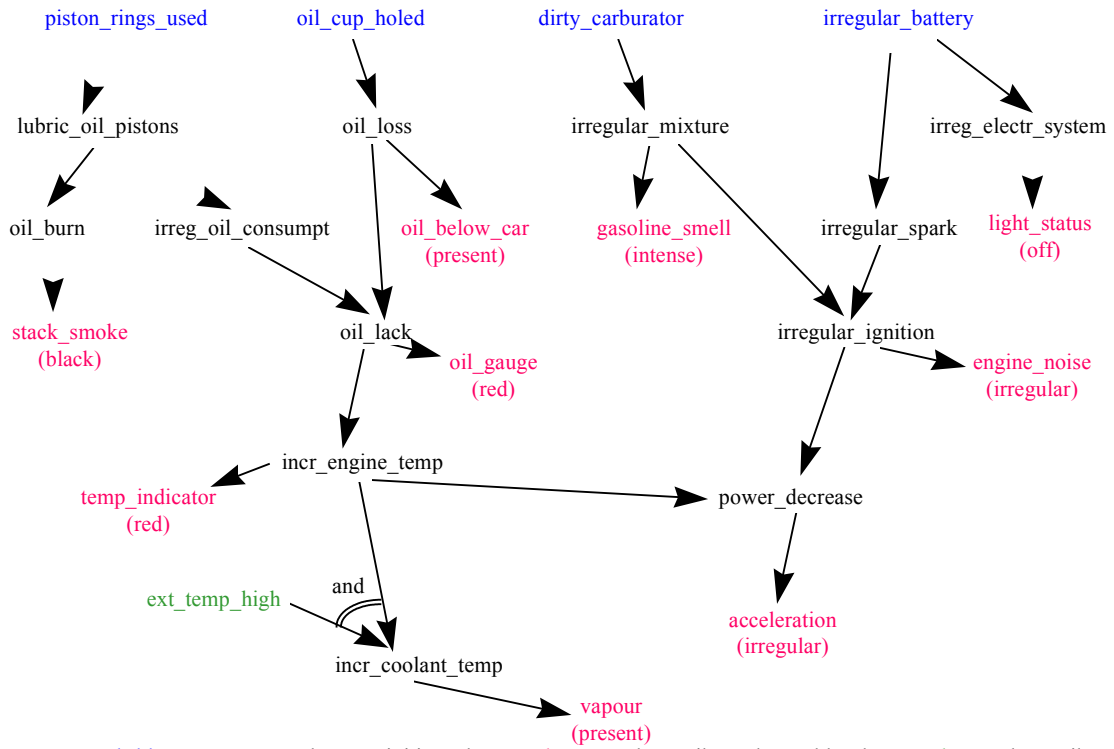
globalement top-down pour déterminer où se situe dans la hiérarchie le dysfonctionnement, origine des symptômes observés.

Partant de cette définition, il est possible de décomposer le système selon plusieurs niveaux d'abstraction, la plupart des cas seront recensés au niveau le plus bas, les niveaux supérieurs correspondront aux dysfonctionnements qui font intervenir plusieurs composants élémentaires, ou auxquels il est difficile d'attribuer un composant précis.

III.2.B Utilisation d'un réseau causal

Un diagnostic peut également être établi grâce à un modèle causal décrivant les possibilités de comportement anormal du système sous contrôle. Console & al [CON, 91] décrivent une approche "focusing (model based) diagnosis". L'opération de diagnostic est composée de deux processus:

- le premier (backward) part des observations et tente de déterminer les états initiaux ou intermédiaires qui impliquent ces observations,
- le deuxième (forward) vérifie que les états déduits ci-dessus n'introduisent pas d'incohérence avec l'ensemble des observations.



Les noeuds bleus représentent des états initiaux, les noeuds rouges des attributs observables, les noeuds verts les attributs externes contextuels, les autres noeuds représentent les états intermédiaires.

fig3. d'après Console et al

Les auteurs proposent de compiler le réseau causal afin de déterminer pour chaque noeud du réseau l'ensemble de ses conditions nécessaires (l'ensemble des observations nécessaires pour que le noeud corresponde au dysfonctionnement courant). Cet ensemble de conditions

nécessaires permet d'éviter l'étude de branches d'alternatives. Par exemple pour le noeud "incr_engine_temp" cet ensemble sera:

temp_indicator (red)
& acceleration (irregular)
& **[**vapour (present) **V** ⌉ext_temp_high**]**
& oil-gauge (red)
& **[**oil_below_car (present) **V** stack_smoke (black)**]**

En prolongeant cette compilation, on aboutit à quatre cas représentant l'ensemble du réseau causal de la figure 3:

cas1

nom	piston_rings_used	
symptômes	stack_smoke (black)	
	temp_indicator (red)	
	oil_gauge (red)	
	acceleration (irregular)	
contexte	vapour (present)	
	V	⌉ext_temp_high
thérapie	....	

cas3

nom	dirty_carburator
symptômes	acceleration (irregular)
	gazoline_smell (intense)
	engine_noise (irregular)
contexte	1
thérapie	....

cas 2

nom	oil_cup_holed	
symptômes	temp_indicator (red)	
	oil_gauge (red)	
	acceleration (irregular)	
	oil_below_car (present)	
contexte	vapour (present)	
	V	⌉ext_temp_high
thérapie	...	

cas 4

nom	irregular_battery
symptômes	acceleration (irregular)
	engine_noise (irregular)
	light_status (off)
contexte	1
thérapie	...

III.3 Critique de ces Solutions

On constate que les différentes méthodes proposées pour le recensement des cas tentent de partir de deux points distincts:

- décomposition hiérarchique du domaine pour recenser les cas à partir des composants (exemple du diagnostic par réseau causal),
- décomposition du système en ses sous tâches, pour identifier les cas à partir de celles-ci (exemple de la conception architecturale).

La démarche consistant à identifier les cas après énumération des composants peut paraître plus rassurante dans la mesure où elle semble aboutir à une collection de cas présentant une meilleure couverture du domaine (il semble aisé d'identifier tous les composants susceptibles d'être défectueux). Mais elle présente deux défauts principaux:

- l'identification des cas (des composants défectueux pour les systèmes de contrôle, des éléments à insérer dans le bâtiment pour les systèmes de conception, etc.) ne prend en compte qu'un aspect **statique**, négligeant les aspects **dynamique** et/ou de **contrôle**. Soit le système représenté par la figure 4a: un composant à découpage de phase R permet de réguler la tension appliquée aux bornes d'une lampe en fonction de l'éclairement mesuré par une cellule photométrique CP et contrôlé par un analyseur A qui commande le régulateur par l'intermédiaire du composant C.

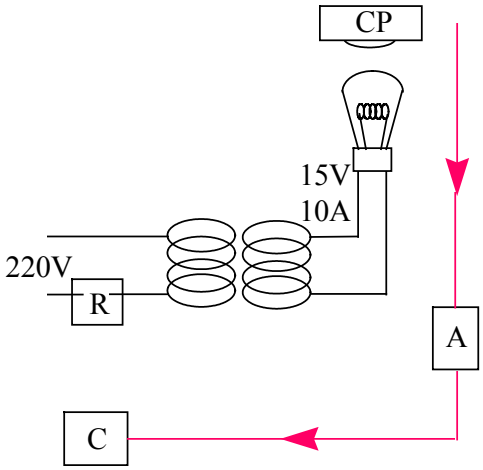


fig. 4a

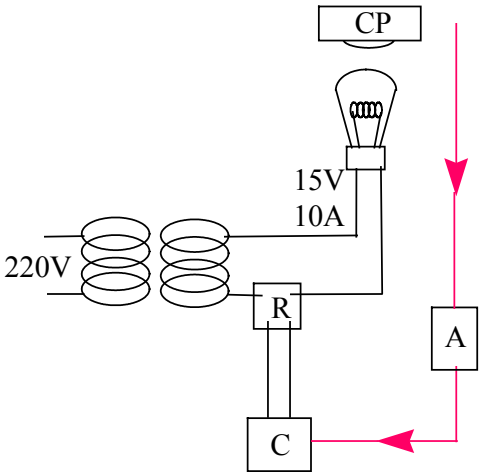


fig. 4b

En fait le montage représenté par la figure 4a n'assure pas un fonctionnement correct du régulateur R (placé avant le transformateur, il fonctionne sous 220V et sous trop faible intensité). Le montage correct est représenté par la figure 4b. Aussi, sans qu'aucun des composants ne soit (statiquement) défectueux, on aboutit avec le montage de la figure 4a à une mauvaise régulation de la lumière (dysfonctionnement dynamique et de contrôle) et le transformateur bourdonne jusqu'à ce qu'il "claque". A ce moment on sera en présence d'un dysfonctionnement statique. Ce problème semble simple à résoudre parce que seuls les câbles dédiés à la fonction de régulation ont été représentés. Il n'en est pas du tout de même lorsque tous les câbles assurant les différentes fonctions sont rassemblés et plus ou moins enchevêtrés.

- lorsque le système devient quelque peu complexe, le nombre de composants augmente, et le nombre d'interconnexions entre ceux-ci devient très important. Un dysfonctionnement ne pourra que très rarement être rattaché à un seul composant. La décomposition hiérarchique du système sera bloquée au niveau des sous fonctions de ce système.

Il semble donc souhaitable de suivre l'approche par décomposition fonctionnelle en sous tâches. Mais d'une part l'identification des composants demeure nécessaire, d'autre part aucun conseil méthodique n'est présenté dans la littérature concernant cette approche par décomposition.

Conclusion

Il serait souhaitable de disposer d'une méthode permettant une analyse décomposée en trois parties:

- **analyse fonctionnelle**: identification des utilisateurs du système permettant l'identification des interfaces entre ces utilisateurs et ce système. D'où la possibilité de l'étude de la transmission des ordres (les fonctions assumées par le système) et des réponses, et l'identification des différents indicateurs, voyants, instruments de mesure, ... cette dernière conduisant également à l'identification des différents paramètres des symptômes.
- **analyse statique**: identification des différents composants physiques et des liens statiques entre eux. Que se passe-t-il lorsque tel composant est défectueux ? Comment affirmer si ce composant fonctionne ou non ?
- **analyse dynamique**: identification des interactions de contrôle / commande entre composants. Quels composants jouent ce rôle vis à vis de quels autres composants ? Que se passe-t-il lorsque cette commande n'est pas correcte et pourquoi ? Pourquoi cette commande est-elle incorrecte ?

III.4 Pourquoi la Méthode OOSE de Jacobson ?

III.4.A Pourquoi une Méthode de Génie Logiciel ?

Trois principales raisons ont amené à considérer l'intérêt des techniques du génie logiciel pour l'analyse d'un domaine en vue du recensement d'une base de cas:

- on oublie parfois qu'un système expert est avant tout un logiciel (ou un progiciel). Bien que le cheminement du raisonnement (par exemple le déclenchement d'une série de règles de production) ne semble pas prévisible a priori, l'ensemble, composé du moteur d'inférences et des règles, est traduit sous la forme de 0 et de 1 pour pouvoir être exécuté sur une machine.
- la connaissance incluse peut être exprimée sous différentes formes (règles, cas, contraintes, ...). La forme retenue correspond ni plus ni moins à un langage de programmation, et comme pour tout développement de logiciel, la phase d'analyse doit demeurer indépendante du langage de programmation, du type de SGBD retenu, etc.
- la réalisation d'un système RBC suit le même cycle de développement que celui d'un autre logiciel: la spirale traversant de manière cyclique les phases d'analyse, de

construction et de test. En effet les cas doivent être recensés, intégrés à la base de cas, testés, puis affinés (nouvelle analyse), ...

On pourrait dire que dans le domaine du diagnostic, on souhaite recenser les dysfonctionnements, alors que ce n'est pas la fonction première d'une analyse classique d'un logiciel. Il est facile de répondre qu'une bonne analyse doit identifier les exceptions, celles-ci n'étant rien d'autre que des fonctionnements anormaux du système.

III.4.B Pourquoi une Méthode Objet ?

Les principales origines de l'approche objet sont au nombre de quatre:

- la simulation (langage Simula 1967, développement de simulations de processus parallèles);
- la communication homme - machine (langage Smalltalk 1976, développement d'environnement graphiques);
- les systèmes embarqués (langage Ada 1980, développement de systèmes temps réel);
- et l'intelligence artificielle (différents langages de représentation de données, de connaissances complexes).

C'est cette dernière origine qui a tout naturellement conduit à étudier les approches objet pour l'analyse des systèmes RBC.

Parallélisme entre Cas et Objet

Il existe en effet un parallélisme certain entre la composition d'un cas et la définition d'un objet telle que présentée par Bouché [BOU, 94], suivant le courant de pensée de Booch:

Un objet se définit comme tout ce qui possède	**Un cas** est constitué de champs représentant
- une identité	- les index
- un état	- le contexte
- un comportement	- l'action ou thérapie

Finalement, un "bon cas" doit satisfaire aux mêmes contraintes ou critères que ceux auxquels doit satisfaire un "bon objet", notamment en terme de maintenance:

- maintenance corrective: qui corrige les erreurs,
- maintenance adaptative: qui adapte le système à un nouvel environnement,
- maintenance évolutive: qui permet l'ajout de nouvelles fonctions.

Le cas est à la représentation des connaissances ce qu'est l'objet aux langages de programmation. Un cas est une entité qui reste indépendante des autres cas. La lisibilité est meilleure et du fait de cette indépendance, la maintenance est facilitée.

L'Analyse Orientée Objet

Les raisons précédentes sont des raisons qui nous font préférer une représentation des cas par des objets, une analyse de la base de cas en suivant une méthode objet.

Mais ici le premier objectif est d'analyser le système (sous contrôle, de conception, de planning, etc.) pour pouvoir recenser la base de cas, et non d'analyser la base de cas.

Bouché [BOU, 94] situe cette étape: "l'analyse permet de modeler un environnement en identifiant les classes et objets du vocabulaire du domaine du problème" et rappelle une suggestion de Shlaer et Mellor pour l'identification des classes et objets: "les choses tangibles, les rôles, les événements, les interactions".

Un objet doit posséder, suivant la définition présentée ci-dessus, un état et un comportement. Ainsi on retrouve les souhaits exprimés plus haut à savoir une analyse statique et dynamique.

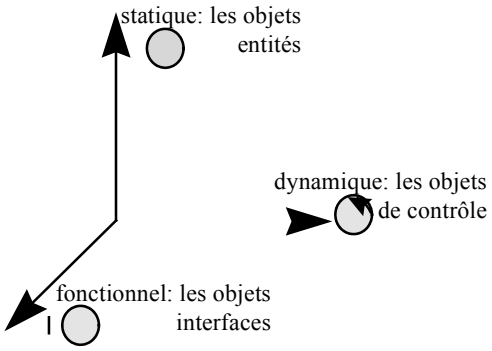
Et surtout l'approche objet permet une grande cohérence entre les objets et leur dynamique, contrairement à l'approche systémique classique où l'on modélise séparément les données et les traitements, séparation risquant d'introduire des incohérences, des accès concurrents aux données, etc.

III.4.C Pourquoi la Méthode OOSE d'Ivar Jacobson ?

Comme le reconnaît Bouzeghoub [BOUZ, 94], OOSE est une méthode objet originale, dont le cycle de développement est l'un des plus complets. Elle suit une démarche à la fois structurée et systémique tout en intégrant l'approche objet. Ses deux principales caractéristiques et avantages sont:

- l'identification des cas d'utilisation, scénarios fonctionnels ou vues concrètes du système, permettant une analyse modulaire et incrémentale;

- l'introduction de trois types distincts d'objets: les objets entité (aspect statique), les objets de contrôle (aspect dynamique) et les objets interfaces (aspect fonctionnel).



On constate immédiatement que ces deux caractéristiques, présentées comme des avantages, satisfont pleinement les souhaits formulés en conclusion du paragraphe précédent.

III.4.D La Méthode OOSE

Description de l'exemple

Une analyse chromatographique en phase liquide repose sur le principe suivant: les composés chimiques d'un mélange, mélange introduit dans le système à l'aide d'un injecteur, sont plus ou moins retenus par une phase stationnaire solide (contenue dans une colonne) lorsqu'ils sont entraînés par un mélange de solvants, mélange poussé par une pompe haute pression. En sortie de colonne, ils seront détectés les uns après les autres, et la surface de chaque pic (calculée par un intégrateur) du chromatogramme résultat est proportionnelle à leur concentration dans le mélange. Il est ainsi possible de déterminer la composition inconnue d'un mélange donné de composés en comparant les surfaces de ses pics à celles obtenues avec un mélange de composition connue. Le schéma de principe du système est représenté figure 5.

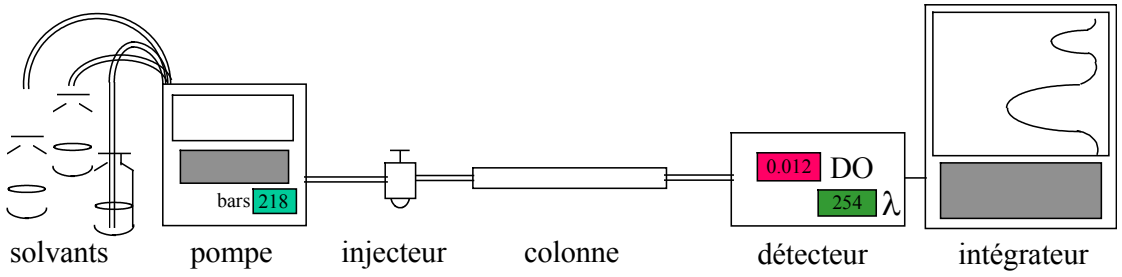


fig. 5 Schéma de principe d'un système d'analyse chromatographique en phase liquide

La Phase d'Analyse de OOSE

Cette phase construit deux modèles: le **modèle des besoins** et le **modèle d'analyse** (remarque: dans tout ce paragraphe, le terme système fait référence au système sous contrôle, de conception, ou de planning et non au système RBC, sauf expressément stipulé)

Le Modèle des Besoins

Jacobson indique que ce modèle doit préciser les acteurs, les cas d'utilisation, les interfaces et les objets du domaine.

Les acteurs

Jacobson fait une distinction entre un utilisateur (instance d'une personne) et un acteur (instance du rôle que cet utilisateur joue). Cette distinction n'est pas nécessaire pour le recensement de la BC, les deux notions seront regroupées sous le terme d'utilisateur. Il en sera de même pour la distinction entre acteurs primaire (utilisateur direct du système) et secondaire (superviseur du système). Ces simplifications sont la conséquence du fait qu'une base de cas est dédiée à un type d'utilisateur précis.

Les cas d'utilisation

Un cas d'utilisation est une manière spécifique d'utiliser le système en faisant appel à une partie de sa fonctionnalité.

Les utilisateurs du système étant identifiés, il est facile de recenser les cas d'utilisation, les questions suivantes proposées par Jacobson pouvant servir de guide:

- quelles sont les principales tâches de chaque acteur ?
- l'acteur aura-t-il à lire/écrire/modifier une information du système ?
- l'acteur devra-t-il informer le système des modifications extérieures ?
- l'acteur souhaite-t-il être informé des modifications inopinées ?

Les interfaces

Une interface utilisateur reflète le point de vue logique de l'utilisateur sur le système. Mais il faut aussi identifier les interfaces entre sous-systèmes, opération plus complexe: certaines s'avéreront peut-être inutiles, mais dans ce cas les deux sous-systèmes devront être considérés comme un seul, au contraire un sous-système pourra être décomposé et une interface ajoutée.

Les objets du domaine

Leur identification et définition permet les dialogues avec l'expert et les utilisateurs. L'objectif ici est de simplement constituer un dictionnaire des concepts du domaine. Dans le cas d'une application complexe et/ou importante, l'analyste doit en effet tout d'abord acquérir une connaissance et une expérience du domaine:

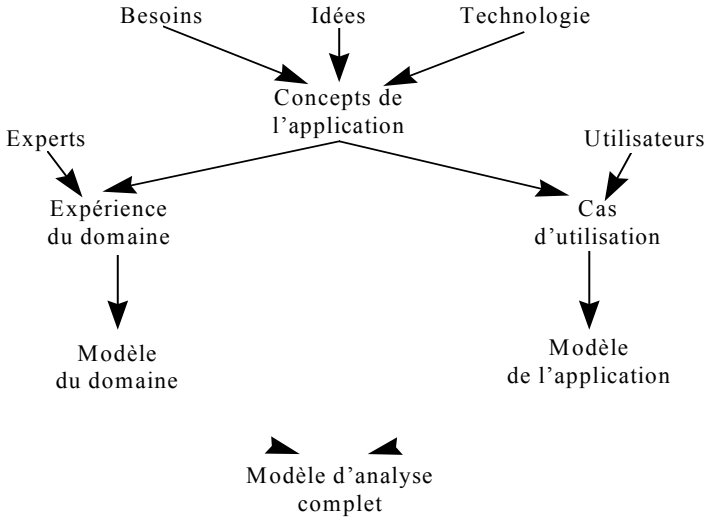


fig. 6 L'élaboration du Modèle d'analyse d'après Rumbaugh [RUM, 94]

Le Modèle d'Analyse

Le modèle d'analyse structure le modèle des besoins en trois types d'objets: les objets interfaces, les objets entités et les objets de contrôle. Durant cette phase les comportements décrits dans les cas d'utilisation sont répartis entre ces trois types d'objets.

Remarque: La phase d'analyse de OOSE **doit recenser les objets au sens langage informatique**. Dans la phase d'analyse du système (domaine du diagnostic, de la conception, du planning, etc.) on veut **recenser les objets au sens composants**, matériels ou logiciels, et déterminer leur rôle (interface, contrôle, liaisons statique ou dynamique).

Cette confusion concernant les deux significations du concept d'objet est volontairement maintenue. Le soin d'identifier le sens qu'il faut donner au mot "objet" est laissé au lecteur, afin de maintenir le parallélisme entre les deux phases d'analyse.

Les Objets Interfaces

Le rôle des objets interfaces utilisateur est de traduire les actions d'un utilisateur en événements dans le système et d'informer l'utilisateur des événements du système qu'il doit connaître. Le chromatogramme résultat est un objet interface, constitué de ("constitué de" est une liaison statique) pics, autres objets interfaces. La forme de ceux-ci constitue un paramètre subjectif (pic symétrique fig. 7a, ou présentant une traînée fig. 7d).

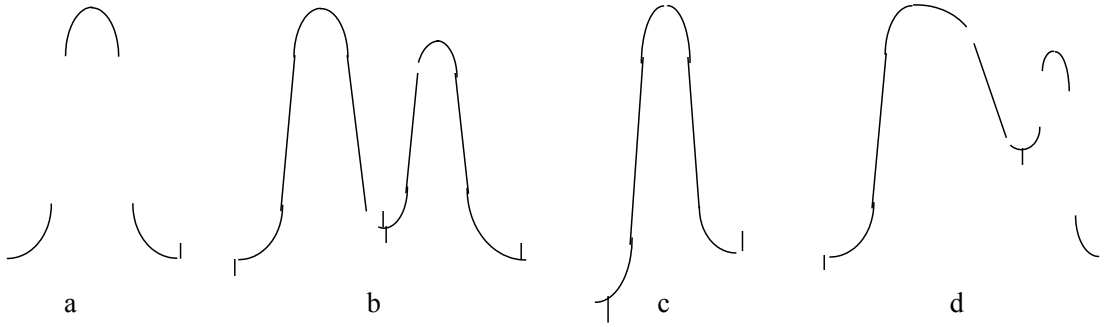


fig. 7 Différentes configurations de pics

Le rôle des objets interfaces entre sous-systèmes est souvent plus complexe: par exemple le détecteur envoie un signal continu à l'intégrateur. L'objet interface entre eux doit échantillonner le signal et transmettre ces données discrètes à un objet de contrôle.

N.B: On s'aperçoit que les objets interfaces permettent d'identifier les paramètres subjectifs.

Les Objets Entités

Ils sont destinés à modéliser l'information que le système traite, et le comportement naturellement associé à cette information.

Par exemple le calculateur de surface de pic est un objet entité.

N.B.: Ils permettent d'identifier les composants et les liaisons statiques.

Les Objets de Contrôle

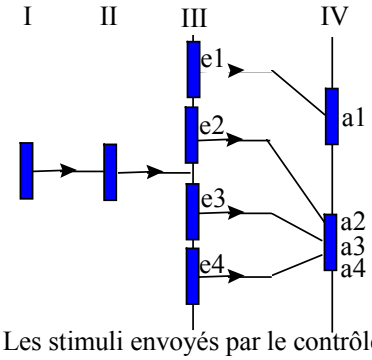
Ils sont destinés à recevoir les comportements qui ne seraient pas **naturellement** associés aux deux autres types d'objets. Typiquement ces comportements sont du type transaction, ou cloisonnement entre les objets interfaces et les objets entités.

Par exemple l'objet de contrôle entre l'interface et le calculateur de surface de pics : ce dernier doit actionner une alarme lorsque la pente de la ligne de base, calculée à partir des données transmises par l'interface, a dépassé en augmentant un certain seuil depuis un certain temps (début de pic), est passé en dessous d'un certain seuil depuis un certain temps (fin de pic) ou a changé de sens (sommet du pic ou début d'un pic non séparé du précédent). Sur la figure 7 les débuts de pic sont indiqués par l'intégrateur par un petit trait vertical vers le bas, les fins de pic par un petit trait vertical montant; ces marques fonctionnent comme des parenthèses imbriquées.

Cet objet de contrôle utilise deux paramètres principaux pour détecter les débuts et fins de pic:

- PT (pour Pic Threshold) permet de les différencier du bruit de fond et doit être adapté à la sensibilité réglée au niveau du détecteur;
- PW (pour Pic Width) permet de différencier les traînées de pic, les plateaux, etc. Ce paramètre doit être adapté à la largeur des pics de l'analyse.

L'observation des marques de début et de fin de pic permet de vérifier si ces paramètres ont bien reçu des valeurs correctes, ou si ces valeurs doivent être modifiées.



I: détecteur, envoie un signal continu
II: interface reçoit signal continu, envoie un signal discret
III: contrôle reçoit signal discret, calcule pente, envoie un stimulus
IV: calculateur de surface de pic

Les stimuli envoyés par le contrôle en fonction de l'événement et de son état, et les actions qui en découlent au niveau du calculateur de surface de pic, sont:

e1: événement dépassement de seuil, état : pas de pic déjà détecté => a1: début de pic

e2: événement dépassement de seuil, état : pic déjà détecté => a2: on incrémente la surface du pic

e3: événement en dessous du seuil, état : pic déjà détecté => a3: fin de pic

e4: événement changement de pente, état : pic déjà détecté => a4: début d'un deuxième pic (négatif -> positf)

événement en dessous du seuil, état : pas de pic déjà détecté => rien à faire

N.B.: Les objets de contrôle permettent d'identifier les liaisons dynamiques entre objets.

III.4.E Rappel: Les Techniques d'interview

Ces techniques doivent être employées de la même manière qu'elles le sont pour l'acquisition des connaissances. Cette étape nécessite de la part de l'analyste psychologie et pédagogie, une préparation appropriée, un enregistrement et une documentation des données (verbales, graphiques, observées, qualitatives et quantitatives), et demeure un processus plus complexe qu'un "bon" ordonnancement de "bonnes" questions. Comme indiqué plus haut, l'analyste doit tout d'abord acquérir une bonne connaissance des concepts du domaine, il pourra ensuite s'aider des schémas suivants:

- **Interview centré**

  Phase initiale normale la plus employée, comportant trois parties: le but ou sujet, les questions / réponses et un résumé; les réponses les plus utiles proviennent plus de questions précises que de questions générales, questions pouvant se référer à: - l'application dans son environnement, - comment est-elle réalisée, - les acteurs et les objets, - les caractéristiques des différents utilisateurs.

- **Interview structuré**

  Permet d'analyser peu de thèmes en profondeur plutôt que beaucoup superficiellement, demandant continuellement des éclaircissements, des explications, des conséquences, des justifications, des exemples et des contre-exemples, orienté vers l'aspect statique de l'application.

- **Introspection**

  On demande au spécialiste comment il résout tel problème, orienté vers les stratégies, les décisions, justifications; utilisation de scénarios de simulation

- **Enonciation**

  Cheminement à l'intérieur de l'application; énoncé à voix haute; sélection de sujets ni trop complexes ni trop évidents; obtention d'information concernant les aspects de contrôle, les enchaînements.

- **Dialogues spécialiste <-> utilisateur**

  Identification des objets interfaces; caractéristiques des utilisateurs; fonctionnalité du système.

- **Vérification**

  Vérification des données et de leur interprétation; aspects non résolus.

IV EXEMPLE

IV.1 Identification des Utilisateurs, des Cas d'Utilisation, des Interfaces et des Objets du Domaine

IV.1.A Les Utilisateurs

Le système RBC envisagé est conçu pour apporter une aide à l'utilisateur, technicien de laboratoire, et non au technicien spécialiste, responsable de la maintenance. Il n'y a donc qu'un seul utilisateur.

IV.1.B Les Cas d'Utilisation

Ici peut se poser une première question: doit-on considérer un seul cas d'utilisation, par exemple "Réaliser une analyse chromatographique de dipyridamole", ou le décomposer en plusieurs étapes ? L'option prise consiste à considérer que tout scénario ou sous scénario lancé par un événement extérieur au système, et non par un événement interne, est un cas d'utilisation.

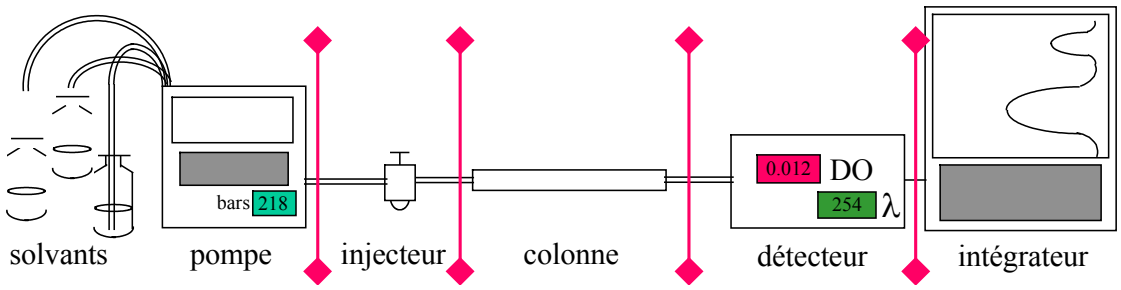
Un seul utilisateur donc, celui-ci doit:

1. Préparer les solvants prévus pour l'analyse: les solvants (après filtration si nécessaire) doivent être dégazés dans un bain à ultra sons, puis par barbotage d'Hélium, avec un débit au début assez élevé puis ensuite maintenu faible.
2. Installer la colonne prévue pour l'analyse.
3. Allumer la pompe et programmer l'analyse : réglage de la composition du mélange d'élution (réglage du pourcentage de chacun des solvants), réglage du débit.
4. Allumer et régler la longueur d'onde du détecteur.
5. Allumer l'intégrateur.
6. Laisser stabiliser l'ensemble et vérifier la conformité de la pression du solvant d'élution dans la colonne, pression indiquée sur la pompe.
7. Régler le zéro du détecteur.
8. Injecter le mélange de référence. Observer le chromatogramme obtenu (forme des pics, leur temps de rétention, allure de la ligne de base).
9. Programmer l'intégrateur.
10. Injecter le mélange à analyser. Observer le chromatogramme obtenu (cf. ci-dessus) et les résultats.

IV.1.C Les Interfaces

Les différentes interfaces utilisateur correspondent aux différents claviers et écrans.

Quant aux interfaces système, celles que l'expert retient dans un premier temps sont représentées en rouge dans la figure ci-dessous, les solvants et la pompe ne constituant à ce niveau de l'analyse qu'un sous-système. Il est intéressant de noter que certaines interfaces ne semblent correspondre qu'à un "bout de tuyau" et servent surtout de séparations logiques entre sous-systèmes.



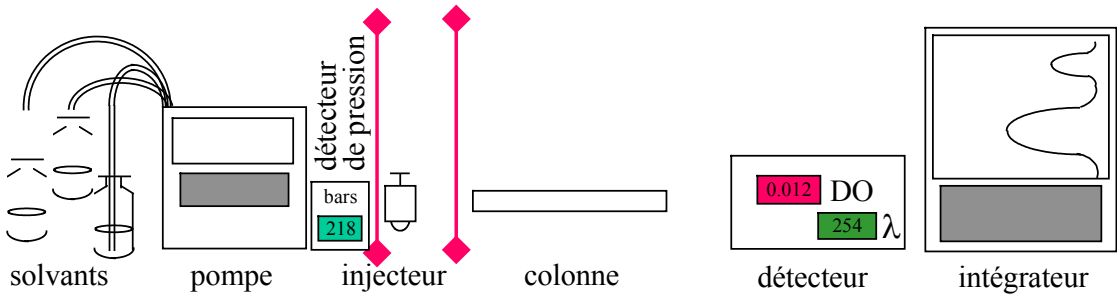
IV.1.D Identification des Objets du Domaine

Nous allons nous intéresser pour la suite uniquement au 6ème cas d'utilisation.

Le mélange de solvants, spécifié pour l'analyse en cours, présente, à une température donnée, une viscosité donnée. La colonne de l'analyse contient des particules de taille donnée. La pression en tête de la colonne, pour l'analyse en cours, doit donc toujours être la même, aux conditions de température près, exemple: pour une colonne de 25cm de long, de diamètre 4mm, remplie de particules de 5µm, traversée par un mélange 85% méthanol / 15% tampon phosphate aqueux, à un débit de 2ml/mn, la pression doit être de l'ordre de 280 bars.

De nombreux concepts ont certainement été identifiés lors de l'analyse des cas d'utilisations précédents, et seulement certaines précisions sont apportées, comme par exemple la viscosité pour le solvant d'élution. Par contre un nouveau paramètre apparaît pour la première fois : la pression. Il y correspond un voyant et une valeur normale, c'est un paramètre subjectif. En interrogeant l'expert on apprend que les valeurs hors normes sont "faible", "forte" et "instable".

De plus on peut identifier un nouveau sous-système, et représenter pour ce cas d'utilisation le système complet de la manière suivante: groupement des solvants et de la pompe en un seul sous-système (comportant le sous-système de détermination de la pression), groupement de la colonne, du détecteur et de l'intégrateur en un seul sous-système (ces deux derniers composants n'ont pas d'influence sur la pression).



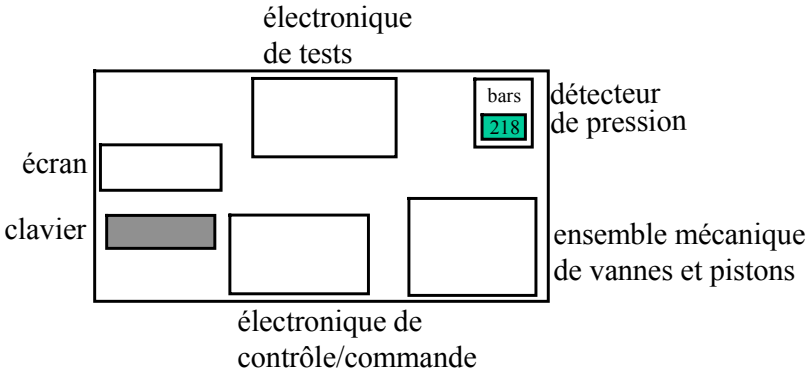
IV.2 Identification des Objets Interfaces, des Objets de Contrôle et des Objets Entités

On peut commencer cette étape avant que la précédente ne soit complètement terminée (notamment le recensement complet des objets du domaine). Des itérations cycliques (indiquées par Ii) vont avoir lieu entre les deux étapes pour compléter, au fur et à mesure de l'avancement de l'analyse, les ensembles des différents types d'objets.

IV.2.A Les Objets Entités

I1: L'injecteur comprend une boucle de volume fixe (le volume d'injection) et de nombreux raccords (les interfaces avec les autres sous-systèmes). Chacun de ces différents raccords peut être le siège d'une fuite.

I2: La pompe est en fait composée de plusieurs sous-systèmes: le clavier, l'écran de visualisation, le sous-système de détermination/visualisation de la pression, l'électronique de contrôle/commande de la composition du solvant d'élution et du débit, les vannes ternaires d'entrée et de sortie (permettant le mélange des trois solvants) et les pistons, et une électronique de tests.



IV.2.B Les Objets Interfaces

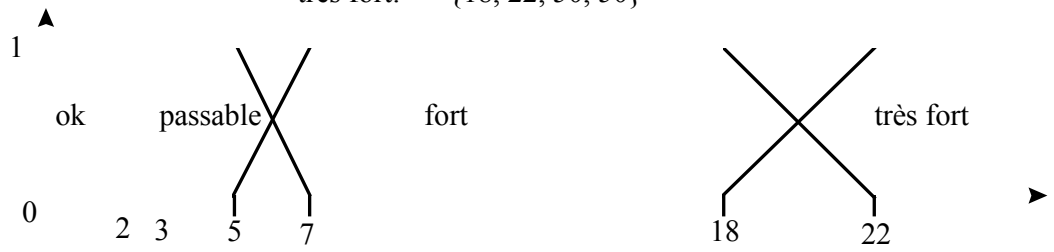
I1: La colonne, remplie de particules de 10, 7, 5 ou même 3µm, contient à ses deux extrémités des filtres pour retenir les particules les plus fines et ces filtres peuvent être obstrués par les impuretés du solvant d'élution, de l'échantillon, ou du solvant de dilution de celui-ci.

L'interface entre l'injecteur et la colonne comprend d'un côté le raccord de sortie de l'injecteur et de l'autre le raccord à la colonne (pouvant tous deux occasionner des fuites), cette dernière comportant un filtre en entrée et en sortie (toujours du point de vue de l'analyse de la pression).

I2: Le clavier et l'écran servent d'interface utilisateur au module électronique de test. En lançant la procédure de test, l'opérateur va recevoir sur l'écran les valeurs numériques, symptômes objectifs, concernant les vannes d'entrée et de sortie, et le contrôle de flux. Ces valeurs sont exprimées dans une échelle arbitraire, les valeurs normales et hors normes correspondant aux ensembles flous représentés sur la figure 8.

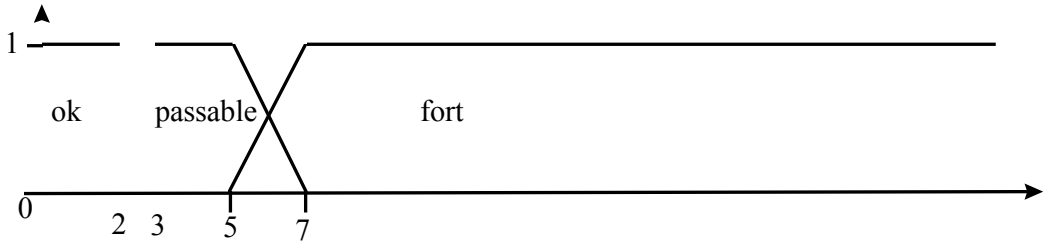
vanne d'entrée

valeur normale: {0, 0, 2, 3}
valeurs hors norme: passable: {2, 3, 5, 7}
fort: {5, 7, 18, 22}
très fort: {18, 22, 50, 50}



vanne de sortie

valeur normale: {0, 0, 2, 3}
valeurs hors norme: passable: {2, 3, 5, 7}
fort: {5, 7, 50, 50}



contrôle flux

valeur normale: {0, 0, 4, 5}
valeur hors norme: fort: {4, 5, 10, 10}

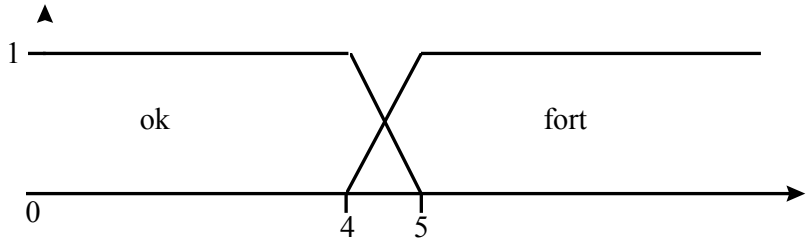


fig. 8: les différents ensembles flous représentant les valeurs concernant les vannes d'entrée et de sortie, et le contrôle de flux.

Rappel: Pour un ensemble classique, le degré d'appartenance d'un élément x à cet ensemble est une fonction qui retourne une valeur de l'ensemble {0, 1}; pour un ensemble flou le degré d'appartenance est une fonction qui retourne une valeur de l'intervalle [0, 1]. Ainsi, sur la figure 9 on voit que la personne de taille x appartient au degré m à l'ensemble "moyen" et au degré p à l'ensemble "petit".

$\mu_{moyen}(x) = m$ (la personne x est "moyen" au degré m)

$\mu_{petit}(x) = p$ (la personne x est "petit" au degré p).

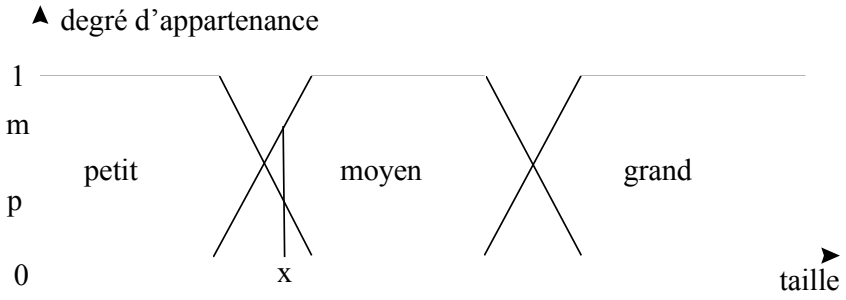


fig. 9: les ensembles flous représentatifs des valeurs "petit", "moyen" et "grand"

IV.2.C Les Objets de Contrôle

I1 La pompe est une pompe à débit constant (il existe des pompes qui maintiennent la pression constante en faisant varier le débit). L'électronique de régulation du débit est très fiable (aucun dysfonctionnement détecté en plusieurs années d'utilisation) et le débit peut être considéré comme exact.

I2: L'électronique de test est un objet de contrôle (entre autres) du bon fonctionnement des vannes.

IV.2.D Le Recensement des Cas

I1: Les valeurs hors normes de la pression sont: "faible", "forte" et "instable". Etant donnée la fiabilité de la régulation du débit, cette partie n'est pas mise en cause.

Deux cas peuvent être identifiés:

C1: Symptôme: la pression est faible
Contexte: conditions normales d'analyse
Thérapie: Examiner les différents raccords pour vérifier l'absence de fuite

C2: Symptôme: la pression est forte
Contexte: conditions normales d'analyse
Thérapie: Changer les filtres d'entrée et de sortie de la colonne

I2: Un cas est identifié correspondant à une instabilité de la pression, mais il comporte des sous cas faisant intervenir des symptômes objectifs, correspondant aux valeurs hors norme des résultats des tests des vannes d'entrée et de sortie, et du contrôle de flux:

C3: Symptôme: la pression est instable
Contexte: conditions normales d'analyse

Sous Cas SC1: Symptômes: test de la vanne d'entrée retourne fort
test de la vanne de sortie retourne fort
test du contrôle de flux retourne fort
Thérapie: Il y a certainement des bulles dans les vannes.
Il faut purger le circuit.

Sous Cas SC2: Symptômes: test de la vanne d'entrée retourne très fort
test du contrôle de flux retourne fort
Thérapie: Il y a des bulles au niveau de la vanne d'entrée.
Purgez la.

Sous Cas SC3: Symptômes: test de la vanne d'entrée retourne fort
test du contrôle de flux retourne fort

	Thérapie:	La vanne d'entrée est défectueuse. Il faut la changer.
Sous Cas SC4:	Symptômes:	test de la vanne de sortie retourne fort test du contrôle de flux retourne fort
	Thérapie:	La vanne de sortie est défectueuse. Il faut la changer.
Sous Cas SC5:	Symptômes:	test de la vanne d'entrée retourne passable, test de la vanne de sortie retourne passable test du contrôle de flux retourne fort
	Thérapie:	Les deux vannes semblent encore utilisables. Mais si vous observez d'autres symptômes (ligne de base dérive, les temps de rétention varient, Présence de pics fantômes), il faudra les remplacer.

V CONCLUSION

Le recensement d'une base de cas doit être mené avec rigueur. C'est une opération réalisée par des itérations cycliques.

Ce recensement peut être obtenu plus aisément après une analyse du système (de diagnostic, de conception ou de planification) suivant la méthode OOSE de Jacobson.

L'analyse fonctionnelle est facilitée par l'identification des utilisateurs, des cas d'utilisation (la fonctionnalité du système) et des interfaces, ces dernières permettant l'identification des symptômes subjectifs.

La décomposition des cas d'utilisation en objets entités, objets interfaces et objets de contrôle permet de déterminer le rôle de chaque composant de chaque fonction ou sous tâche du système, et d'identifier les différents cas.

BIBLIOGRAPHIE

[BOU, 94]: M. Bouché, "La démarche objet. Concepts et outils", AFNOR, 1994.

[BOUZ, 94]: M. Bouzeghoub & al, "OBJETS, Concepts, Langages, Bases de données, Interfaces", Eyrolles, 1994.

[CAM, 90]: J.A. Campbell and J. Wolstencroft, "Structure and Significance of Analogical Reasoning", AI in Medicine, April 1990.

[CHA, 83]: B. Chandrasekaran, "Towards a Taxonomy of Problem Solving Types", The AI Magazine, Winter/Spring 1983, pp 9-17.

[CON, 91]: L. Console, L. Portinale, D.T. Dupré, "Focusing Abductive Diagnosis", AI Communications, Vol. 4, Nr. 2/3, June/December 1991

[DUN, 45]: K. Duncker, "on Problem Solving", Psychological Monographs, 58., N° 270, 1945.

[EVA, 68]: T.G. Evans, "A Program for the solution of a class of geometric analogy intelligence test questions", In M. Minsky editor, Semantic Information Processing, MIT press, Cambridge, Mass., 1968.

[FUC, 95]: B. Fuchs, A. Mille, B. Chiron, "Operator decision Aiding by Adaptation of Supervision Strategies", ", In M. Veloso & A. Aamodt editors, Case Based reasoning Research and Development, First International Conference, Springer, October 1995.

[JAC, 93]: I. Jacobson, "Le génie logiciel orienté objet", Addison-Wesley France, 1993.

[KOL, 93]: J. Kolodner, "Case-Based Reasoning", Morgan Kaufmann Publishers, 1993.

[KOS, 92]: B. Kosko, "Neural Networks and Fuzzy Systems. A Dynamical System Approach to Machine Intelligence", Prentice Hall, 1992.

[LAU, 90]: J-M. Laurent, "Raisonnement Approximatif, une extension du Modus Ponens Stricto Sensu par les modificateurs", Proceedings des 10èmes Journées internationales, Les Systèmes Experts et leurs Applications, Avignon 1990.

[RUM, 94]: J. Rumbaugh, "Modelling & Design", Journal of Object Oriented Programming, Vol. 7, N°5, September 1994.

[STR, 95]: G. Strube, A. Enzinger, D. Janetzko & M. Knauff, "Knowledge Engineering for CBR systems from a Cognitive Science Perspective", In M. Veloso & A. Aamodt editors, Case Based reasoning Research and Development, First International Conference, Springer, October 1995.

[WOL, 89]: J. Wolstencroft, "Restructuring, Reminding and Repair: What is missing from models of Analogy", AI Communications, Vol. 2, N°2, June 1989.