Chapitre 1 : Contexte-Préliminaires-Choix effectués

Devant la complexité du problème posé par le développement d'un Environnement de Simulation Intelligent, il s'agit tout d'abord de définir le cadre de travail de thèse (paragraphe 1-1) et d'opérer des restrictions quant à nos objectifs. Il s'agit pour nous plutôt de raisonner sur un cas particulier d'ESI, c'est à dire sur une instance d'ESI, puis à essayer de tirer des généralisations du résultat obtenu. Cette démarche nous permet de choisir un niveau d'abstraction où les concepts qui seront mis en œuvre seront plus simples à manipuler. Pour ce faire, il s'agit tout d'abord de rappeler les différentes étapes du processus de conception d'un logiciel de simulation (paragraphe 1-2). La connaissance des aspects principaux de ce processus va influencer la conception de l'application interactive. Il s'agit ensuite de dégager les critères de qualité auxquels doit répondre un produit logiciel (paragraphe 1-3), ce qui va nous permettre de définir les caractéristiques que doit posséder l'environnement informatique du développement de l'ESI : choix du système d'exploitation, du type de langage de développement. Cependant, l'interface n'est pas tout, le produit interfacé doit être aussi une bonne application. Une étude des codes de calcul nous a permis d'opérer un choix parmi les codes existants (paragraphe 1-4). Il nous est aussi apparu qu'un Environnement de Simulation Intelligent doit incorporer en son sein des mécanismes d'aide. En effet, dans le cas d'une application qui entre dans le cadre de la modélisation/simulation, l'utilisateur non expert ne connait pas, a priori, la procédure à suivre au sein du système informatique : l'utilisateur est généralement démuni pour contrôler le processus de modélisation/simulation et pour en apprécier les résultats (multiples, méconnus et qui ne sont pas prédéterminés). Le paragraphe 1-5 dresse une première liste d'applications possibles dont l'objectif est l'assistance de l'utilisateur.

1-1 Positionnement

Le projet ESI a pour objectif de développer un environnement de simulation générique permettant l'intégration de logiciels de simulation pour des domaines d'applications variés (le bâtiment pour ce qui nous concerne). Il s'agit d'autre part de réaliser des applications, qui se situent à plusieurs niveaux et qui concernent des publics différents [PELLETRET 1-92].

Les travaux réalisés dans le cadre de cette thèse consistaient à contribuer au développement du concept d'ESI, notamment par la réalisation et le test d'une instance de ce concept. Pour ce faire, nous avons réalisé un environnement fondé sur le principe de la modélisation par assemblage, cet environnement est appelé IISIBât (Interfaces Intelligentes pour la Simulation dans le Bâtiment). Dans la suite de ce document, nous ferons référence essentiellement à cette application du concept générique d'ESI.

Les travaux menés dans le cadre du projet IISIBât portent principalement sur le développement d'un certain nombre de fonctions d'aide à la modélisation, à la simulation et à l'analyse des résultats ; ces fonctions utilisent les informations relatives aux modèles (consignées sous forme de PROFORMA [DUBOIS 90]). Dans ce cadre, une des tâches du projet IISIBât consiste à envisager le couplage avec les banques de données de composants manufacturés. Par ailleurs, IISIBât a pour objectif d'intégrer un modèle de données des projets de bâtiments, adapté aux "points de vue" des différents évaluateurs incorporés dans cet environnement. Une tâche importante du projet IISIBât est de faire le lien entre le mode standardisé d'archivage des connaissances sur les modèles (PROFORMA) et le modèle de données interne à IISIBât. Une autre tâche importante est de démontrer la pertinence de l'utilisation couplée de divers logiciels, ce qui, a priori,

permet d'accroître les potentialités de l'environnement de simulation en termes d'objectifs et de précision des calculs [PELLETRET 1-92].

Les applications en développement actuellement sont :

- l'application **IISIBât/TRNSYS**[1] qui intéresse principalement les laboratoires de recherche ; elle constitue elle-même un nœud à partir duquel d'autres applications spécialisées peuvent être développées (**IISIBât/CA-SIS**[2]).

- l'application **IISIBât/COMIS**[3] qui s'adresse également aux centres de recherche. Comme pour IISIBât/TRNSYS, il est tout à fait possible de dériver des sous-applications spécialisées.

Les applications IISIBât/TRNSYS et IISIBât/COMIS peuvent être aussi couplées avec des banques de données "constructeurs", de façon à les rendre opérationnelles dans des bureaux d'études.

D'autres applications sont envisagées, à moyen terme :

- l'application **IISIBât/TRNSYS/COMIS** s'inscrit dans l'idée d'un couplage entre logiciels de calcul, ce couplage permettant de couvrir différents aspects des performances d'un bâtiment à partir d'une même application interactive.

- L'application **IISIBât/SPARK**[4] va permettre d'utiliser un solveur numérique performant (SPARK [NATAF-WINKELMANN 91]). Ce solveur permet une distinction claire entre modèle physique et résolution numérique. Cette séparation entre objets représentant des composants technologiques ou des phénomènes physiques, et des méthodes de résolution confère à SPARK une grande modularité et de réelles capacités d'adaptation à n'importe quel domaine d'application. Il est donc possible de dériver aisément l'application IISIBât/SPARK vers n'importe quel domaine d'application.

- L'application **IISIBât/CSTBât**[5] va permettre de valoriser des modèles complexes développés au sein de la structure du code de calcul TRNSYS [LARET 1-89].

---

1 TRNSYS est un logiciel de calculs thermiques pour le bâtiment ; il a été développé à l'Université de Wisconsin-Madison [TRNSYS 90].

2 CA-SIS est un logiciel dérivé de TRNSYS, centré sur les performances des installations de climatisation, développé par EDF.

3 COMIS est un logiciel de calculs aérauliques, développé par le Lawrence Berkeley Laboratory [COMIS 91].

4 SPARK est un solveur numérique développé par le Lawrence Berkeley Laboratory.

5 CSTBât est un logiciel dérivé de TRNSYS, développé par le CSTB (Centre Scientifique et Technique du Bâtiment) [LARET 1-89].

Le projet IISIBât peut se résumer par la figure 1-1. Ce travail de thèse s'inscrit principalement dans le cadre du développement de l'application IISIBât/TRNSYS.

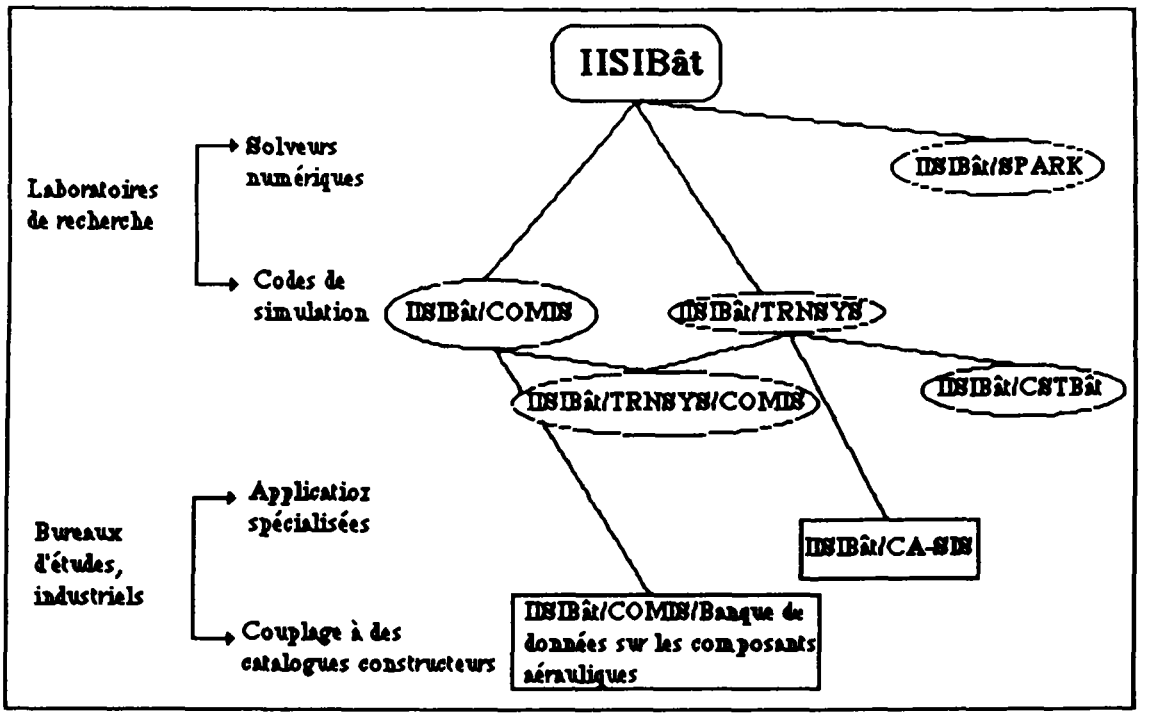

Figure 1-1 : Exemples des différents niveaux d'application du projet IISIBât

1-2 La simulation

L'objectif général d'un programme de simulation est d'analyser et de prédire le comportement d'une réalité externe, appelée aussi système physique. Beaucoup de ces systèmes physiques se prêtent à la simulation par événements discrets. Celle-ci consiste à décrire des systèmes physiques dont les états changent à la suite d'événements individuels, qui se produisent à des instants précis. En entrée de la simulation, il y a une suite d'événements ou de données avec leurs date de réalisation. Cette suite est obtenue en général grâce à des mesures sur de vrais systèmes physiques (lorsque la simulation est utilisée pour analyser des phénomènes passés). Elle est aussi obtenue grâce à des générateurs de nombres aléatoires conformément à certaines lois statistiques.

Un modèle à événements discrets doit notamment observer le temps du système physique, appelé *temps simulé*. Ce temps simulé ne doit pas être confondu avec le temps de calcul nécessaire pour exécuter le code de simulation. En général, le temps simulé est pour le programme de simulation une variable qui augmente par sauts discrets (variable *time* pour le code TRNSYS).

1-2-1 Objectifs de la Simulation

La simulation numérique donne à l'utilisateur la possibilité de faire varier différents paramètres d'un système en vue d'observer son comportement. La simulation numérique est généralement utilisée pour des phénomènes dont l'expérimentation réelle est dangereuse, impossible ou coûteuse.

Les objectifs peuvent être multiples :

- l'étude du comportement d'un système dans différentes circonstances ;
- l'évaluation de diverses stratégies pour le fonctionnement d'un système ;
- la détermination des limites de fonctionnement d'un système ;
- la prévision ;
- la vérification de théorie ;
- la mise au point de systèmes ;
- la compréhension des phénomènes physiques ;
- la compréhension du fonctionnement des systèmes physiques.

Ces différents objectifs vont avoir une incidence directe sur la conception d'une interface pour un outil de simulation.

Un utilisateur qui s'intéresse aux données et à leur interprétation doit avoir à sa disposition des outils pour l'aider dans sa tâche : traceur de courbes, logiciels de mise en forme de données statistiques. Par contre, un utilisateur dont l'objectif est la modélisation peut avoir besoin d'accéder aux fichiers sources, pouvoir en modifier certains et tester les incidences de sa modification. Dans ce cas, l'interface doit proposer à l'utilisateur des outils qui lui permettront d'accélérer la mise au point de ses programmes.

1-2-2 La conception d'un logiciel de simulation

Deux grandes tâches sont à prévoir lors de la réalisation d'un outil de simulation :

- la modélisation ; toute conception d'un logiciel de simulation passe par un travail de modélisation ; celle-ci consiste à construire des modèles qui sont des représentations abstraites des phénomènes-objets réels ; la modélisation est un travail qui comprend deux étapes, la modélisation mathématique et la modélisation numérique ;
- la mise en place d'une structure logicielle.

La modélisation mathématique consiste, après avoir fait une analyse physique du système à étudier, à dégager les flux jugés essentiels dans le cadre du problème à étudier et à les traduire sous forme d'équations (algébriques ou différentielles). Cette étape est délicate, il s'agit souvent de générer le modèle le mieux adapté au problème posé. Cette difficulté peut être contournée en proposant plusieurs modèles pour un même système, d'où la notion de bibliothèques de modèles (on parle de *modèlothèque*). Un modèle reste une représentation approchée, il représente certains comportements d'un objet ou système physique, avec un certain nombre d'approximations. Au cours d'une modélisation par assemblage, ces approximations peuvent expliquer des résultats de simulation inattendus : un effet indésirable dû à une approximation pouvant être propagé ou amplifié dans l'assemblage au cours d'une simulation.

La conception d'une interface doit tenir compte du grand nombre d'objets existant dans une modèlothèque ; elle doit permettre à l'utilisateur l'accès rapide aux modèles, ainsi qu'une aide pour leur choix dans le cadre d'un projet. A cet effet, une attention toute particulière doit être accordée aux fonctions relatives à l'archivage des composants dans une modèlothèque (cf. § 2-3-1-1, § 2-4-3).

La modélisation numérique consiste à traduire les équations définies lors de la modélisation mathématique en algorithmes de calcul. La modélisation numérique va elle

aussi introduire une approximation par rapport à la modélisation mathématique. De plus, l'algorithme de calcul doit être adapté aux possibilités de l'ordinateur.

Dans le monde de la thermique, les systèmes différentiels linéaires occupent une place importante ; ils sont généralement résolus à l'aide de méthodes qui discrétisent le temps (méthode d'Euler). D'autres méthodes de résolution discrétisent et le temps et l'espace : méthodes aux différences finies et méthodes aux éléments finis.

Remarque : il n'est pas possible de tout résoudre numériquement ; certaines équations instables sont difficiles, voire impossibles à résoudre numériquement (cf Annexe 2).

Principes de la modélisation mathématique

Dans le monde du bâtiment, la modélisation mathématique des phénomènes repose essentiellement sur des principes de conservation et sur des équations de transferts de deux types [LEBRUN 85] :

- transfert des flux d'énergie échangés entre sous-systèmes. On distingue les transferts qui ont lieu sans transfert de fluide, sous la forme de chaleur (modélisation de la conduction) ou de travail, et les transferts qui s'opérent avec transfert de fluide sous forme d'énergie interne, d'énergie cinétique, d'énergie potentielle, etc...

- transfert des flux de matière, comme les débits d'air, les débits d'eau.

Dans la modélisation mathématique, ce sont principalement les équations de transfert qui introduisent le plus grand nombre d'incertitudes et d'imprécisions. D'autres imprécisions sont introduites par les définitions de quantités telles que la chaleur massique, la conductivité thermique d'un corps. Dans la plupart des cas, on se contente de fabriquer des modèles simplifiés, à base d'hypothèses simplificatrices dont certaines reviennent souvent :

- isotropie et homogénéité d'un milieu ;
- régime permanent ;
- conduction unidimensionnelle ;
- tout transfert thermique peut s'exprimer linéairement en fonction des températures d'air (intérieures, extérieures), et des températures de parois ;
- les propriétés thermophysiques des matériaux varient très peu ;
- les échanges convectifs et radiatifs, bien que non linéaires, peuvent se linéariser.

La connaissance d'informations concernant la modélisation mathématique peut aider l'utilisateur à opérer des choix dans une bibliothèque de composants, et à interpréter des résultats. Ces informations peuvent être contenues dans des fiches, appelées fiches Proforma (cf. § 1-5-3). Si l'utilisateur sait que la modélisation d'un composant s'est faite à partir d'équations de transferts, il peut effectuer des déductions sur ce modèle quant à sa précision et au degré d'incertitude qui l'entoure.

1-2-3 Modularité et structure informatique des codes de simulation

Il existe dans le monde de la thermique du bâtiment un certain nombre de logiciels qui offrent la possibilité de simuler des phénomènes physiques. Ces outils diffèrent par les techniques numériques et logicielles employées, le niveau de détail des systèmes thermiques décrits et la convivialité des interfaces. Ces outils sont généralement

incompatibles entre eux, et l'étude complète d'un problème complexe de bâtiment nécessite l'emploi de plusieurs logiciels ; le chercheur est même parfois obligé d'insérer au sein d'un programme principal des morceaux de code spécifiques, ce qui a pour effet de rendre rapidement les programmes difficiles à maintenir.

Par ailleurs, les outils de simulation sont caractérisés par le domaine traité : on parle d'outil "général" et d'outil "dédié". Les développeurs sont confrontés au dilemne suivant : un outil "général" est mal adapté à une application particulière, un outil "dédié" ne permet pas de traiter de problèmes, qui relevent d'un autre domaine que celui pour lequel il a été conçu. Le principe de la modularité permet de contourner cette difficulté : il est possible au sein d'une architecture modulaire d'insérer des modules écrits pour des applications a priori différentes ; certains modules peuvent être polyvalents ; d'autres peuvent être très généraux, d'autres encore peuvent être spécifiques (modules "dédiés"). Ainsi, la modularité permet d'avoir "au sein d'un seul logiciel plusieurs logiciels".

On peut donc distinguer les codes de simulation par leur structure informatique :

- certains possèdent une structure modulaire ;
- d'autres possèdent une "structure globale".

Ces derniers permettent en général une simulation globale du bâtiment et de ses équipements ; l'utilisateur doit décrire obligatoirement les caractéristiques de son bâtiment et de ses équipements pour pouvoir simuler. Ces logiciels opèrent des restrictions importantes quant aux variantes du système, et dans la prise en compte des interactions entre le bâtiment et ses équipements.

Par opposition à ces codes de simulation, il existe des logiciels permettant une simulation à *la carte*. Ces logiciels ont la particularité de posséder une importante bibliothèque de composants et de sous-systèmes, qui doivent être assemblés par l'utilisateur pour effectuer une simulation. Cet assemblage est entièrement dirigé par l'utilisateur, à ses risques et périls. Ces logiciels demandent donc un peu de dextérité de la part de l'utilisateur pour leur utilisation.

Ces logiciels sont généralement de type *modulaire* et sont structurés en *routines*, celles-ci modélisent les processus de façon autonomes. Chaque processus se caractérise par un cycle de vie bien défini : une routine appelle une autre, qui s'exécute complètement et renvoie le contrôle à l'envoi de l'appel. Ainsi chaque processus suit sa propre vie et s'interrompt pour fournir ou recevoir de l'information d'un autre. Le schéma global de l'architecture informatique est complètement décentralisé : chaque processus s'occupe de sa propre tâche, avec une interférence limitée avec les autres.

1-3 Eléments de base pour la conception d'une application interactive

Avant de concevoir une interface, il paraît important de répertorier les exigences que doit satisfaire un produit logiciel. Ces exigences vont permettre de donner une appréciation sur sa qualité.

On peut définir neuf qualités essentielles d'un logiciel [MEYER 90] :

- la *validité* ; c'est l'aptitude du logiciel à réaliser exactement les tâches définies par sa spécification ;
- la *robustesse* ; c'est l'aptitude du logiciel à fonctionner même dans des conditions anormales. Il faut s'assurer que le logiciel, pour des tâches non spécifiables, puisse

quand même fonctionner, ou du moins ne pas déclencher des actions irréparables (destruction de fichiers de données par exemple).
- *L'extensibilité* ; c'est la facilité d'adaptation d'un logiciel aux changements de spécification ;
- la *réutilisabilité* ; c'est l'aptitude d'un logiciel à être réutilisé en tout ou partie pour de nouvelles applications ;
- la *compatibilité* ; c'est l'aptitude d'un logiciel à pouvoir être compatible avec d'autres logiciels ;
- *l'efficacité* ; c'est l'aptitude d'un logiciel à utiliser de façon optimale les ressources du matériel ;
- la *portabilité* ; c'est l'aptitude d'un logiciel à s'adapter à différents environnements ;
- *l'intégrité* ; c'est l'aptitude d'un logiciel à protéger ses composantes contre des modifications non autorisées ;
- la *facilité d'utilisation* ; c'est la facilité avec laquelle les utilisateurs vont apprendre à utiliser un logiciel.

En pratique, il est impossible de respecter simultanément tous les critères cités précédemment. A titre d'exemple, une efficacité optimale qui exige l'adaptation parfaite à un certain environnement est en contradiction avec le principe de la portabilité. Il faut donc effectuer un compromis.

Dans le cadre du projet IISIBât, la continuité est une préoccupation importante. Il ne s'agit pas de ne réaliser qu'une première version, il s'agit tout au contraire de penser à son cycle de vie qui peut s'étendre sur différentes phases d'adaptation et d'évolutions. Cette remarque conduit à accorder une attention particulière à l'extensibilité et à la réutilisabilité.

Il faut retenir deux principes qui permettent d'assurer l'extensibilité à un logiciel :

- la simplicité de l'architecture informatique ;
- la modularité de l'architecture informatique ; plus les modules de l'architecture logicielle sont autonomes, plus il est probable qu'une modification n'affectera qu'un nombre restreint de modules.

La compatibilité est à prendre en compte. En effet, les interfaces produites dans le cadre d'IISIBât doivent pouvoir communiquer avec le monde des outils de simulation (monde de TRNSYS) et interagir avec eux.

La facilité d'utilisation est un critère important dans le cadre d'IISIBât : il s'agit d'offrir aux utilisateurs une interface simple à appréhender et conviviale.

En résumé, les qualités essentielles à respecter dans le cadre d'IISIBât sont les suivantes : validité, robustesse, extensibilité, réutilisabilité, compatibilité, facilité d'utilisation.

1-3-1 Extensibilité, réutilisabilité et approche orientée objet

Pour la conception de l'interface, l'approche orientée objet répond aux critères énoncés plus haut. Elle répond notamment aux soucis de ***réutilisabilité et d'extensibilité***. Les programmes sont de nature modulaires grâce à la création d'entités cohérentes et facilement manipulables : les objets.

La question essentielle à laquelle est confrontée un développeur est la suivante "Quels objets sont concernés ? ". Cette question touche deux aspects ; d'un côté, elle concerne les objets appartenant à la réalité physique environnante (pompes, régulateurs, bâtiments...) ; d'un autre côté elle concerne les représentations informatiques appropriées, appelées aussi objets. On parlera dans la suite de ce document d'objets externes et internes. Au stade de la conception, il s'agit plutôt de passer en revue les classes d'objets externes dont le système veut modéliser le comportement ; au stade de l'implémentation, les systèmes à modéliser doivent être écrits comme des objets internes.

Chaque objet appartient à une classe qui détermine son fonctionnement et les attributs statiques le caractérisant. Une classe est un objet "abstrait" capable d'engendrer d'autres objets appelés *instances*. Une instance est un objet "concret" ; l'instanciation se fait par duplication de toute la structure de l'objet abstrait : les champs ou méthodes définis au niveau d'une classe se retrouvent dans tous les objets concrets instanciés à partir de cette classe. Les objets appartenant à une même classe ont la même forme, ont un comportement commun, mais sont différents par la valeur de leurs champs. Appliquer une opération à une instance consiste à lui envoyer un *message* contenant le nom de la méthode devant être activée.

Associée à ce concept de classe se rattache une notion importante : l'héritage de classe. Beaucoup d'objets peuvent être définis par rapport à d'autres, seules quelques caractéristiques les différencient. Une classe peut dériver d'une ou plusieurs classes appelées *superclasses*, dont elle hérite les variables et les méthodes. L'ensemble des classes forment ce qu'on appelle un *graphe d'héritage*. En général, l'héritage des variables est statique : les variables héritées sont recopiées dans la structure représentant la classe. Par contre, l'héritage des méthodes est dynamique : les méthodes héritées ne sont pas recopiées et la méthode à appliquer lors d'un envoi de message est recherchée dynamiquement dans le graphe d'héritage.

Une application est utilement formulée par un langage orienté objet quand elle en met à profit les caractéristiques. La structuration en objets se justifie lorsqu'on a à représenter des entités ayant des données propres en quantité suffisante et des comportements propres. L'instanciation est particulièrement utile dans des applications ayant de nombreux objets du même type. L'héritage trouve son utilité dans un environnement qui nécessite la construction de nouveaux types d'objets à partir de ceux existants. Ceci fait gagner en productivité en évitant la création inutile de code similaire.

La conception d'une hiérarchie de classes constitue le problème central. Dans le cadre d'un outil de simulation, la conception par objets est naturelle : il suffit de décrire les objets à simuler, et par conséquent la modularité d'un langage orienté objet sera mise à profit. L'instanciation et l'héritage seront également mis à profit, ainsi que les interactions entre les objets.

L'approche orientée objets n'est pas seulement utile pour le développeur de l'environnement de simulation mais elle peut aussi être utile pour celui qui développe et intègre de nouveaux modèles au sein de l'environnement de simulation. En effet, il existe au sein de l'ESI des fonctions d'aide à la modélisation et la simulation qui utilisent des propriétés et des informations définies au niveau des modèles. En utilisant la programmation orientée objet (POO), on crée des bibliothèques de modèles qui se présentent sous forme arborescente (cf. § 2-4-3) ; les nœuds de l'arbre constituent des classes auxquelles il est possible de rattacher des propriétés, communes à plusieurs modèles. Les modèles attachées à une classe peuvent hériter des propriétés définies pour cette classe (c'est le principe d'héritage propre à la POO). Ainsi, s'il existe une ou des

bibliothèques de modèles déjà structurées et implémentées sous formes d'objets, le concepteur d'un nouveau modèle pourra, en rangeant son modèle dans une bibliothèque, c'est à dire en rattachant ce modèle à une classe de modèle, bénéficier des propriétés génériques définies au niveau de la classe ou des éventuelles classes-mères.

Par exemple, supposons qu'il existe une classe "Pompe de Circulation" sous laquelle sont déjà rattachés des modèles de pompe de circulation. Admettons que nous ayons défini, parmi d'autres, comme propriété générique d'une pompe de circulation la propriété "possède en sortie un débit" et que cette propriété soit utilisée par un module expert pour réaliser des connexions automatiques entre modèles. Supposons alors qu'un créateur de modèle développe un nouveau modèle de pompe de circulation qui, par exemple, fasse le distinguo entre les pertes thermiques du moteur vers le fluide caloporteur et vers l'ambiance. Une fois son modèle créé, il le rattache à la classe "Pompe de Circulation" ; cette action induit l'héritage par le nouveau modèle des propriétés préalablement définies au niveau de la classe "Pompe de Circulation". Le nouveau modèle hérite du fait qu'il possède un débit en sortie. De plus, toutes les fonctions de l'environnement de simulation qui utilisent ce champ dans leur raisonnement deviennent opérationnelles pour le nouveau modèle.

1-3-2 Plate-forme logicielle retenue pour la conception de l'interface

Pour notre étude, l'environnement informatique est constitué du système d'exploitation UNIX. Celui-ci autorise la communication entre programmes par fichiers interposés, ce qui assure la compatibilité entre les différents logiciels ; de plus, il offre la possibilité de concevoir des interfaces graphiques conviviales, à base de menus déroulants, d'icônes, de fenêtres, pilotées par une simple souris, grâce à l'environnement X-Window (appelé aussi X11 ou tout simplement X).

Celui-ci peut être qualifié de standard "de fait", car implicitement adopté à la fois par les constructeurs de stations de travail et par les programmeurs. En général, X-Window est livré avec un gestionnaire de fenêtres, chargé d'habiller les fenêtres et de les contrôler ; on peut citer : UWM (Unix Window Manager), TWM (Tiled Window Manger), MWM (Motif Window Manager), OWM (OpenLook Window Manager).

Les avantages présentés par X-Window sont nombreux ; il ne privilégie aucun environnement de programmation, il est extensible et reconfigurable. Seule son implantation diffère d'une machine à l'autre.

X-Window s'appuie sur une librairie de fonctions (XLIB pour le langage C), mais celle-ci n'est pas simple à mettre en œuvre directement. Un certain nombre de langages de haut niveau qui facilitent le travail pour le programmeur sont apparus ces dernières années ; il s'agit principalement de langages interprétés et compilés (langage LISP[6]) et de langages compilés (langage C). Ils permettent de définir des objets graphiques et leurs comportements. Les langages interprétés permettent de tester rapidement des morceaux d'interface, les langages compilés sont plus rapides à l'exécution mais plus difficiles à implanter.

---

6 Dans un environnement LISP, il s'agit d'une compilation "au sens LISP", c'est à dire une transformation en un langage intermédiaire plus rapide (LLM3 pour Le-LISP).

Dans le cadre de ce travail, nous avons opté pour l'utilisation du langage Le-Lisp[7], enrichi d'une extension capable de gérer des objets graphiques : la boîte à outils Aïda[8].

Le-Lisp est un descendant du langage LISP, qui fut développé au laboratoire d'Intelligence Artificielle du Massachussets of Technology (MIT). Celui-ci donna lieu à la production de nombreux dialectes, Le-Lisp s'en distinguant par le fait qu'il s'appuie sur une machine virtuelle indépendante du système d'exploitation. De plus, Le-Lisp (version 15-25) est muni d'une extension objet (Ceyx), permettant une programmation orientée objet.

La boîte à outils Aïda est liée à la bibliothèque X-Window ; elle offre la possibilité de programmer des interfaces graphiques à partir de composants élémentaires, en utilisant les techniques d'héritage. Elle se compose d'une bibliothèque de plus de 100 classes de composants graphiques prédéfinis simples (boutons poussoirs, boutons radio, cases à cocher) ou complexes (barres de défilement, éditeurs d'arbres, éditeurs de texte, éditeurs de courbes).

L'utilisation de la boîte à outils Aïda (version 1.65) consiste à définir des structures, sous-structures des composants graphiques cités plus haut. La technique de l'héritage permet d'éviter de redéfinir les comportements des nouvelles applications.

Le-Lisp fournit des mécanismes puissants pour appeler des programmes écrits en d'autres langages (comme C, fortran ou pascal) ; cela permet de séparer de façon claire le programme (écrit en Le-Lisp, en C, ou en fortran) de l'interface graphique construite à partir de l'environnement Aïda.

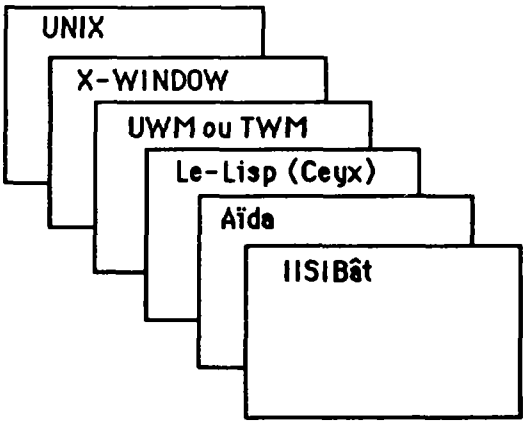

Figure 1-2 : Couches logicielles pour IISIBât

Remarque : **l'extension objet Ceyx sera probablement modifiée, au profit d'une autre extension objet (TELOS) ; certaines fonctions développées au cours de ce travail devront probablement être réécrites (une autre solution consiste à réécrire l'ensemble du code dans un langage indépendant de la couche objet -plate-forme MIPS [POYET 1-92]-).**

---

7 **Le-Lisp est un système LISP, développé à l'INRIA.**

8 **Aïda est un système de construction d'interfaces graphiques pour le système Le-Lisp ; il a été développé par la société ILOG.**

1-4 Code de calcul choisi

Notre choix s'est porté sur l'exploitation du code simulation TRNSYS [TRNSYS 90]. Plusieurs avantages sont présentés par ce logiciel :

- sa portabilité ;
- sa modularité ;
- sa bibliothèque de modules ;
- sa place dans le domaine des logiciels d'études thermiques qui en fait un outil de référence dans le monde de la thermique du bâtiment

Réalisé en Fortran de base, le code TRNSYS est un logiciel modulaire : il est composé d'un programme principal et d'un certain nombre de sous-programmes (ou modules). On distingue :

- les programmes qui modélisent un composant thermique ou un sous-système thermique ;
- les programmes utilitaires qui assurent la lecture des données, l'intégration numérique, l'impression des résultats, etc...

1-4-1 Principe de fonctionnement de TRNSYS

Pour simuler le fonctionnement d'un système global, l'utilisateur doit déterminer quels composants sont présents, à quels modules ils correspondent et quelles relations existent entre eux. La simulation s'effectue à partir du système ainsi reconstitué par modules et connexions.

Un module décrivant un composant du système est caractérisé par un certain nombre de paramètres et d'entrées-sorties connectées sur des entrées-sorties d'autres modules. Ainsi, le module "capteur solaire" est symbolisé de la façon suivante :

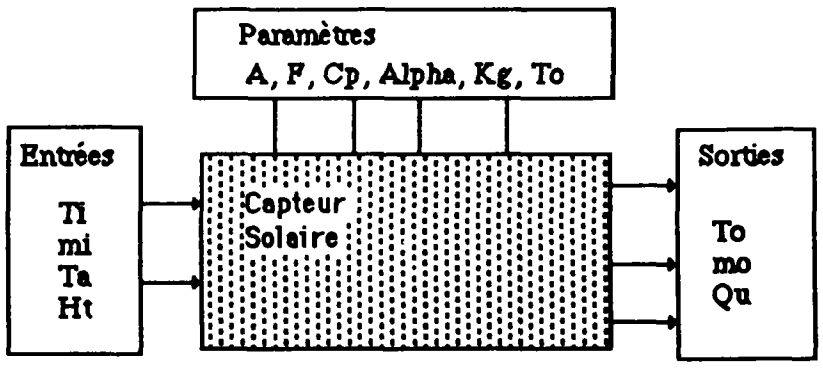

Figure 1-3 : module "capteur solaire" de TRNSYS

Les paramètres pour le module "capteur solaire" de TRNSYS sont :

- la surface (A) ;
- l'efficacité d'absorbeur (F) ;
- la chaleur spécifique du fluide (Cp) ;
- l'absorptivité de la plaque (Alpha) ;
- le coefficient global de déperditions (Kg) ;
- la transmittivité de la plaque (To).

Les entrées pour le module "capteur solaire" de TRNSYS sont :

- la température d'entrée du fluide (Ti) ;
- le débit du fluide (mi) ;
- la température ambiante (Ta) ;
- le rayonnement total incident (Ht).

Remarque : pour une simlation spécifique, une entrée peut être déclarée constante.

Les sorties pour le module "capteur solaire" de TRNSYS sont :

- la température de sortie du champ de capteurs (To) ;
- le débit du fluide (mo) ;
- l'énergie utile recueillie par le fluide (Qu).

Remarque : ces trois sorties (To, mo, et Qu) vont servir à leur tour d'entrées à des modules divers.

Pour certains modules, il faut ajouter aux entrées, sorties, paramètres la notion de "dérivatives" ; les "dérivatives" permettent de spécifier les valeurs initiales des dérivées premières par rapport au temps de certaines variables d'état.

La simulation d'un système complet nécessite la résolution numérique d'équations différentielles du premier ordre couplées. L'algorithme d'intégration utilisé est une méthode d'Euler modifiée, qui introduit un critère de convergence. S'il n'y a pas convergence au bout d'un certain nombre d'itérations au moins sur une variable, la simulation est avortée. Il faut noter que c'est l'utilisateur qui fixe le pas de temps et la tolérance de convergence.

1-4-2 Le fichier DECK

Une simulation est décrite par le fichier de simulation appelé "DECK". Ce fichier est composé de cartes alphanumériques : les cartes de contrôle et les cartes "Unit".

Chaque module mis en jeu au cours d'une simulation est décrit par une carte "Unit". Celle-ci contient les informations suivantes :

- n° d'unité dans la simulation et n° du type du module correspondant ;
- déclaration des paramètres (nombre, valeurs) ;
- déclaration des entrées (nombre, provenance, valeurs initiales) ;
- déclaration éventuelle des "dérivatives" ;
- déclaration éventuelle de cartes spéciales (par exemple la spécification du format de lecture et d'écriture des variables).

Les cartes de contrôle concernent les aspects généraux de la simulation. Les cartes de contrôle les plus importantes sont :

- **la simulation card** (définit le début, la fin et le pas de temps de la simulation) ;
- **la tolérance card** (définit la tolérance sur la convergence) ;
- **la limits card** (définit le nombre maximum d'itérations à chaque pas de temps).

1-4-3 Analyse de TRNSYS

La structure modulaire est le point fort de TRNSYS : celle-ci assure l'extensibilité de la bibliothèque de modules. Il est assez aisé de modifier un module ou même d'en créer de nouveaux.

L'autre point fort de TRNSYS est sa portabilité.

Par contre, l'exécution d'une simulation nécessite des va et vient permanents d'un sous-programme à un autre, faisant ainsi de TRNSYS un code gourmand en temps de calcul.

TRNSYS pose des problèmes de convergence : l'utilisateur doit choisir le pas de temps de la simulation, et la tolérance de convergence de manière à minimiser le temps de calcul tout en gardant une précision acceptable pour les résultats.

1-5 Aide à la Modélisation/Simulation

Le concept d'aide à la modélisation passe par le développement des axes suivants :

- mise en place d'une interface conviviale, qui tient compte de l'efficacité du dialogue homme / machine (cf. § 1-5-1) ;
- mise en place d'un système d'aide concernant le mode de fonctionnement de l'application interactive, et l'outil de simulation (cf. § 1-5-2) ;
- mise en place d'un système d'aide concernant l'utilisation des composants des bibliothèques (cf. § 1-5-3),
- mise en place de mécanismes intelligents qui guident l'utilisateur vers des solutions satisfaisantes, grâce notamment à la méthodologie des systèmes à base de connaissances (cf. § 1-5-4). Ces mécanismes concernent :

  - la mise en place d'un système d'aide à l'introduction, dans une bibliothèque de composants, de nouveaux modèles ;
  - la mise en place d'un système d'aide à la modification des modèles ;
  - la mise en place d'un système d'aide au choix de modules adaptés aux besoins des utilisateurs, en fonction d'un assemblage préexistant ou en fonction de l'objectif de la simulation ;
  - la mise en place d'un système d'aide au choix de méthodes numériques adaptées au problème posé par l'utilisateur ;
  - la mise en place d'un système d'aide à l'analyse des résultats de simulation.

1-5-1 Efficacité du dialogue Homme/Machine

Tout système d'interface est déformant vis à vis de l'information qu'il véhicule. Il est indispensable de développer autour du clavier et de l'écran des systèmes de dialogue qui soient de nature à favoriser le dialogue Homme/Machine. Ceci doit être réalisé dans des conditions qui respectent le plus le mode naturel de langage ou de concept mis en œuvre par l'homme avec les possibilités du système informatique.

L'homme a mis en œuvre la communication, caractérisée par une relative ambiguïté et une certaine redondance. Les systèmes informatiques, quant à eux, utilisent leurs propres langages où redondance et ambiguïté sont absents. Cette distance qui sépare ces deux systèmes d'informations pose problème : qui doit s'adapter à l'autre ?

1-5-1-1 Modélisation de l'interaction Homme/Machine

Plusieurs approches ont été développées, avec pour objectif la modélisation de l'interaction Homme/Machine. Dans le cadre de cette étude, nous rappellerons les aspects principaux des deux modèles suivants : le modèle STI (Système de Traitement de l'Information [BONNET 85]) et les modèles centrés objet [GOLDBERG 84].

Le modèle STI introduit la notion de mémoires. Il distingue :

- la mémoire à court terme, appelée aussi mémoire de travail, qui contient les résultats intermédiaires du raisonnement ;
- la mémoire à long terme qui stocke les connaissances pour une utilisation future, sous forme d'unités de connaissances.

L'augmentation de la productivité d'un utilisateur passe par une utilisation rationnelle de ces deux types de mémoires. Il faut retenir les principes suivants pour la conception d'une interface :

- éviter de surcharger la mémoire de travail de l'utilisateur ; cela consiste à le soulager, par exemple, de la manipulation d'informations telles que les domaines de validité des variables utilisées dans le cadre d'une simulation ; la solution consiste en général à introduire ces données dans une mémoire permanente sous forme d'unités de connaissances ; il s'agit ensuite d'associer à ces unités des mécanismes automatisés de *remémoration* ;

- proposer à l'utilisateur un environnement connu. L'utilisateur doit pouvoir poser ses problèmes et retrouver ce qui est nécessaire à son processus de réflexion, tout en utilisant des formalismes correspondant aux concepts qu'il a l'habitude de manipuler. Ainsi l'utilisateur consacrera l'essentiel de ses efforts à résoudre le problème physique posé et non pas à essayer de dialoguer avec la machine.

Les utilisateurs de TRNSYS travaillent généralement en réalisant un diagramme des flux (établissement des connexions entre les modules) avant d'en faire une traduction informatique. Au niveau de l'interface, l'utilisateur doit pouvoir constituer ce diagramme en manipulant des graphismes. Ceci permettra d'établir un dialogue purement conceptuel entre l'utilisateur et le système.

Les modèles centrés objets introduisent, quant à eux, le principe de la conception modulaire des composants des applications interactives. Cette approche sépare les deux notions suivantes :

- l'application qui rassemble les fonctions propres au domaine ;
- l'interface qui assure la communication entre l'utilisateur et l'application interactive.

La fonction de l'interface consiste à gérer les images à l'écran et à lire les données d'entrée. L'application quant à elle gère le contexte des actions de l'utilisateur.

L'aspect le plus important de cette approche est la répartition du dialogue Homme/Machine, chaque composant dialogue avec l'utilisateur suivant un langage et un formalisme qui lui est propre ; cette répartition du dialogue assure à l'interface extensibilité et réutilisabilité. Dans le cadre d'IISIBât, le dialogue est réparti entre les différents types de fenêtres, chaque type de fenêtre permet de gérer une activité ; malgré

tout, des efforts ont été menés pour rapprocher les différents formalismes associés aux activités en cours au sein de chaque type de fenêtre (cf. § 2-3-3).

1-5-1-2 Les différents types de dialogue Homme/Machine

Les premieres interfaces conçues étaient en général des interfaces textuelles, où l'utilisateur dialoguait avec le système via un système de questions/réponses prédéfini ; ce type d'interface est mal adapté à la description rapide et précise d'un bâtiment, du fait du grand nombre des informations à introduire lors de la conception des projets de bâtiment, de leur caractère hétérogéne et de la difficulté de description des systèmes techniques en langage naturel (il est difficile par exemple de décrire en une seule phrase un pont thermique). Ces dix dernières années, on a vu apparaître des interfaces, pour ce qui concerne le secteur bâtiment, où le dialogue Homme/Machine s'effectue via le dessin du bâtiment et de ses équipements : il s'agit des outils de CAO (Conception Assitée par Ordinateur). On peut penser qu'ils sont bien adaptés au mode de travail des différents intervenants du bâtiment ; en effet, à tous les niveaux de l'étude d'un projet de bâtiment est produit un plan graphique (structure, architecture, électicicité, ...). Cependant, la plupart de ces outils véhiculent une sémantique pauvre, ils ne permettent de manipuler que des données dimensionnelles, et des données touchant à la composition des éléments entrant dans la conception du bâtiment. Ils ne permettent pas de définir tous les comportements (thermiques, acoustiques...) des "objets" constituant le bâtiment, et les interactions de ces "objets" entre eux. Pour y arriver, il faudrait disposer d'un modeleur 3D, capable de générer un modèle de données intégré du bâtiment et de ses équipements (un tel modeleur est en cours de développement pour complèter la plate-forme MIPPE POYET 2-92]).

Dans l'attente de la mise au point d'interfaces graphiques basés sur des outils de CAO qui véhiculeraient une sémantique riche, construits à partir d'une approche orientée objets, il convient de développer des solutions intermédiaires pour le dialogue Homme/Machine ; dans le domaine qui nous intéresse, nous retiendrons comme possibilités :

- le dialogue fondé sur l'assemblage de boîtes noires ;
- le dialogue fondé sur l'assemblage de composants électroniques (résistances, capacités,...) ;
- le dialogue fondé sur l'écriture d'un système d'équations, selon une certaine syntaxe.

Chacun de ces types de dialogue présente avantages et inconvénients, en fonction des utilisateurs, des domaines d'application, et des codes de calcul. Par exemple, pour un mathématicien chevronné, s'exprimer au travers d'équations ne pose guère de difficultés ; quand on traite de modélisation de circuits électriques, décrire le problème par assemblage de composants électroniques est tout naturel.

D'une façon générale, pour appréhender un problème, l'être humain utilise couramment la technique de décomposition d'un système complexe en sous-systèmes plus simples, jusqu'à arriver à des briques élémentaires qu'il sait maîtriser. La description d'un problème par assemblage de composants épouse cette logique.

Il est parfois utile au sein d'un environnement de simulation de combiner divers modes de dialogue. S'il nous paraît aujourd'hui naturel de dialoguer par assemblage de composants, ce type de dialogue étant connu et facile à appréhender par les utilisateurs, il peut être parfois utile de mettre à la disposition de l'utilisateur des outils qui permettent par exemple de décrire une partie précise du problème, sous la forme d'un réseau RC.

Prenons un exemple concret pour étayer notre propos ; supposons la modélisation thermique d'une pièce sous la forme de l'assemblage suivant (figure 1-4-a).

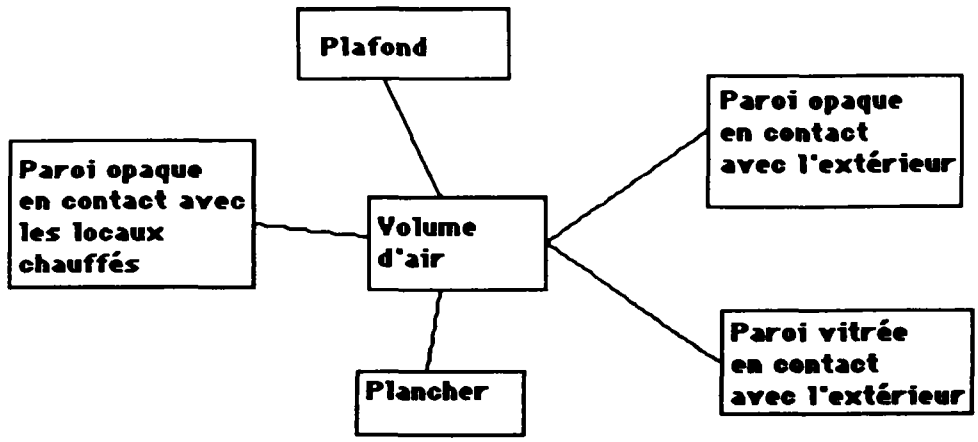

Figure 1-4-a : Modélisation thermique d'une pièce sous la forme d'un assemblage

Il est tout à fait possible que le modèle "Paroi extérieure", ait été conçu de façon à permettre une discrétisation aussi fine que souhaitée. On peut alors imaginer que, pour décrire cette discrétisation, on mette à la disposition de l'utilisateur une interface spécialisée qui lui permette d'assembler des conductances, des capacités, des générateurs de courant, des générateurs de tension...(figure 1-4-b)

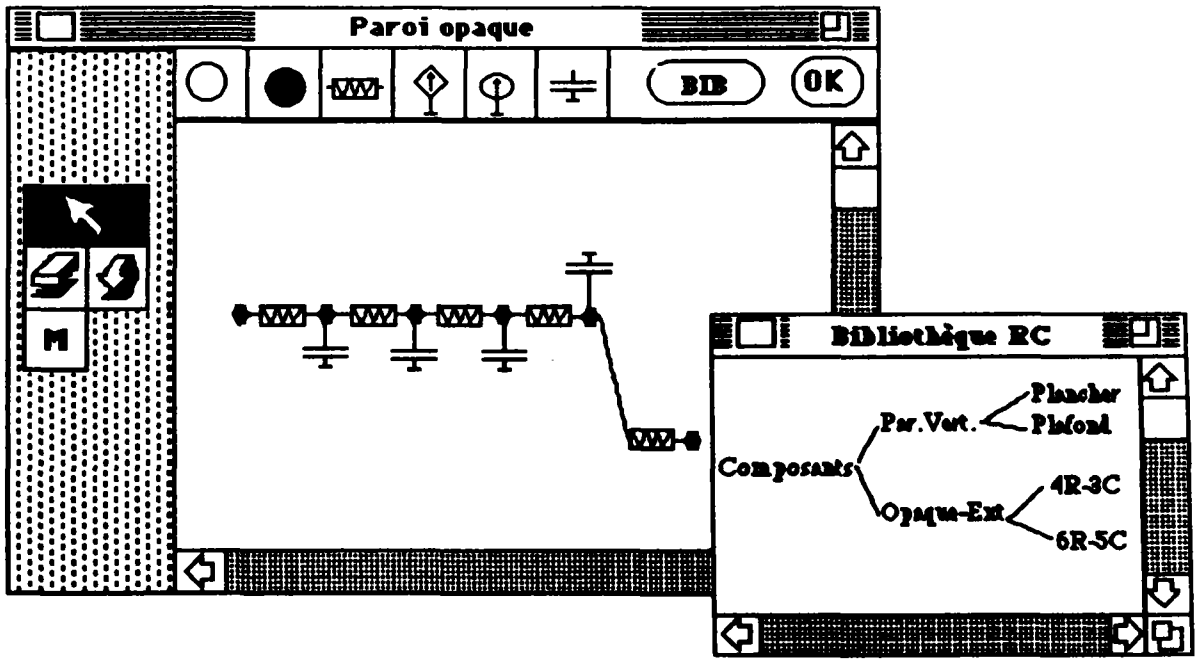

Figure 1-4-b : Sous-interface "Réseau RC"

La sous-interface "Réseau-RC" est constituée :

- d'un bouton, en haut à gauche, qui permet de quitter la session en cours ;
- d'un bouton "OK" qui permet de valider le réseau ;

- d'un bouton "BIB" qui permet d'ouvrir la bibliothèque de réseaux RC prédéfinis ;
- de composants élémentaires. Ils sont au nombre de six : le nœud qui représente une condition limite (cercle vide), le nœud sans inertie (cercle plein), la conductance, le générateur de courant, le générateur de tension et la capacité ;
- d'une boîte à outils ;
- d'une zone de travail.

Les outils qui permettent d'exploiter cette interface spécialisée sont :

-  permet d'instancier l'objet couramment sélectionné (en inverse-vidéo) qui peut être soit un composant élémentaire, soit un réseau prédéfini dans la bibliothèque RC ; il permet aussi de déplacer un objet dans la zone de travail, et de spécifier les valeurs numériques (double-clic) ;
-  permet d'effacer un composant ;
-  permet de revenir à l'état précédent ;
-  permet de stocker dans la bibliothèque RC le réseau qui se trouve dans la zone de travail.

1-5-2 Mise en place d'un système d'aide

Dans un outil logiciel, l'aspect "aide" est important ; il va permettre à un utilisateur de "mieux utiliser" le logiciel. Pour y arriver, l'aide doit concerner :

- l'interface et ses fonctionnalités ; l'utilisateur doit notamment être informé, sur la façon de manipuler les différentes fenêtres, sur les objets manipulables au sein de ces fenêtres, et sur la manière d'utiliser les différents outils. Les informations concernant l'interface, dans le cas idéal, n'ont pas lieu d'être (interface si simple que son utilisation est immédiate) ; dans le cadre d'IISIBât, l'une des exigences à respecter est la facilité d'utilisation de l'interface ; par conséquent, un effort tout particulier doit être fourni pour concevoir une interface simple d'utilisation, qui n'aurait donc pas besoin d'un support documentaire important ;
- l'outil de simulation ; l'utilisateur a besoin à ce niveau d'informations concernant le mode d'emploi du code de calcul. Ce mode d'emploi doit notamment faire apparaître des informations sur la manière de constituer de la façon la plus productive possible le fichier d'entrée pour la simulation (fichier DECK pour TRNSYS), en mettant l'accent sur les options existantes et sur les "subtilités" du code de calcul (par exemple les astuces pour faire converger plus vite les calculs...) ;
- la physique, c'est à dire les modèles existants au sein de l'environnement du code de calcul, grâce notamment à une description standardisée des modèles (fiches Proforma).

Par ailleurs, nous défendons l'idée d'un système d'aide intégré au logiciel, le support papier devant être à terme inutile. L'aide peut apparaître sous plusieurs formes, elle peut être :

- **ciblée ou globale** ;
- **statique ou interactive** ;

- **adaptée** à l'utilisateur ou non.

L'aide ciblée consiste pour le système à donner des informations qui ne concernent que l'objet de la demande de l'utilisateur, par exemple lorsque celui-ci déplace son dispositif de pointage (souris) sur des objets graphiques. L'aide globale consiste quant à elle à afficher un document complet plus ou moins astucieusement administré (structuration en chapitres, sous-chapitres, fonctions de requête), et qui concerne tout le domaine d'étude. L'aide globale posséde deux inconvénients : d'une part, elle oblige l'utilisateur à faire le tri entre les informations qui concernent son problème, et les autres. D'autre part, elle demande à l'utilisateur d'établir la correspondance entre la terminologie employée au sein de l'aide, et l'objet de son problème (qui peut être un bouton, un menu, une icône, une action -dupliquer, gommer-).

L'aide statique est une aide invariante, ceci quel que soit le contexte d'utilisation. Elle peut apparaître sous une forme textuelle ou graphique, pouvant comporter plusieurs niveaux (Hypertext). Pour ce qui concerne l'aide interactive, il s'agit dans la plupart des cas de mini-séquences didactiques, qui permettent d'expliquer le fonctionnement de tel ou tel outil ; l'utilisateur s'entraîne à "blanc", pendant qu'un module expert espionne ses faits et gestes, et lui explique ses erreurs éventuelles. Pour cela, le module expert doit connaître la façon de manipuler les différents outils, et doit être capable, d'une part d'analyser les actions de l'utilisateur, et d'autre part de générer des commentaires adaptés aux erreurs commises, et aux succès de manipulation.

Une aide adaptée à l'utilisateur est fondée sur le principe suivant : "à utilisateur différent aide différente". Il faut donc disposer d'un modèle performant de l'utilisateur, qui puisse évaluer son niveau de compétence. Illustrons ces propos par un exemple ; supposons que l'objectif de l'utilisateur soit l'accomplissement de l'action "Tout Effacer" ; une façon optimale de réaliser cette action consiste à combiner l'action "Tout Sélectionner" et l'action "Gommer". Si le système d'aide estime que l'utilisateur maîtrise l'action "Tout Sélectionner" (parce que le système sait que l'utilisateur en question a utilisé plusieurs fois déjà "Tout Sélectionner"), le message d'aide insistera plutôt sur l'explication du fonctionnement de l'outil "Gomme". La mise en place à l'heure actuelle d'un modèle de l'utilisateur est difficile, d'autant plus qu'il faut tenir compte de certains facteurs psychologiques tel que la notion de dégradation de compétence entre deux manipulations, distantes dans le temps.

Dans le cadre de ce travail, pour les raisons invoqués ci-avant, nous avons opté pour un système d'aide non adapté à l'utilisateur, tous les autres types d'aide (ciblé, global, statique et intéractif) peuvent être insérés au sein de l'environnement de simulation.

1-5-3 La fiche PROFORMA

Dans le cadre des travaux liés à l'analyse des phénomènes physiques, la documentation sur les modèles utilisés est de qualité variable et de présentation différente. Elle se retrouve parmi des documents hétéroclytes tels que des manuels d'utilisateurs, des thèses, des articles, des ouvrages spécialisées. En 1984, sous l'appellation de "Data Processor Proforma", le laboratoire de Physique du Bâtiment de LIEGE (B), avait proposé, avec le groupe ABACUS de l'Université de Stathclyde (GLASCOW-UK), une fiche descriptive des modèles de thermique. Cette première version a été appliquée dans les travaux de l'Agence Internationale de l'Energie, en liaison avec le CSTB. Ce dernier au sein du GER Almeth reprenait et développait le concept, en collaboration avec la FNB. Un projet, testé notamment à l'INSA de Lyon en 1985 et au CSTB, a abouti à la proposition du Proforma, cadre permettant à tous d'archiver de façon standard les

connaissances relatives aux modèles. Depuis, les réflexions sur le Proforma continuent au sein du GER-Almeth ; par ailleurs, les idées à la base du Proforma ont été reprises par différents laboratoires qui développpent des modèles (DETN-GDF, département ADE à EDF).

L'idée initiale du Proforma est d'offrir un outil pour faciliter la compréhension, et l'échange des modèles, grâce à une standardisation de la documentation associée aux modèles. Cette standardisation va permettre aussi d'assurer une certaine survie du modèle à son auteur. Depuis 1990, dans le cadre du développement du concept d'ESI, le Proforma est utilisé pour documenter sous forme de fichiers informatiques les modèles. Les informations contenues dans les Proformas sont utilisées pour divers processus de raisonnements intégrés dans les ESI.

Remarque : la nécessité de disposer de certaines informations spécifiques pour conduire des raisonnements au sein de l'ESI peut amener des modifications du Proforma.

A l'heure actuelle, les fiches Proforma sont utilisées pour :

- la documentation. Il s'agit de renseigner les utilisateurs sur les bases théoriques des modèles, sur les principales hypothèses qui ont guidé la modélisation, sur les méthodes de traitement utilisées,...
- l'aide à l'utilisation des modèles au sein des environnements de simulation ; une description simple et claire de ce que fait le modèle, de ses entrées, de ses sorties, de ses paramètres figure au sein de la fiche Proforma. Cette description permet à l'utilisateur de faire le choix du modèle le mieux adapté à son problème ;
- l'aide à l'interprétation des résultats ; les informations qui concernent les applications possibles du modèle, les limites de validité, les problèmes rencontrés permettent d'expliquer des anomalies éventuelles constatées au cours de l'utilisation. Elles permettent par exemple de comprendre pourquoi des valeurs calculées sortent de l'intervalle de validité donné dans les fiches Proforma.

Dans le cadre de notre travail, nous avons choisi de documenter les modèles de TRNSYS en gardant l'essentiel de la structure du Proforma. Les informations essentielles pour l'aide à l'utilisation des modèles sont :

- les hypothèses physiques sous-jacentes à la modélisation (linéarité, isotropie, etc...) ;
- le type du modèle (équationnel, algorithmique, expérimental, logique, qualitatif) ;
- les limites de validité (elles sont décrites essentiellement dans les domaines de valeur des variables) ;
- la liste des applications (un même modèle peut par exemple aussi bien représenter un refend, une paroi extérieure ou un plancher) ;
- les modèles amonts (modèles connectables en amont du modèle considéré) ;
- la liste des variables utilisées avec leurs attributs (nom, valeurs par défaut, unité, ...).

La fiche Proforma est structurée en cinq chapitres :

- le premier donne une information synthétique à l'aide d'un nom, d'un résumé et de quelques données sur les méthodes utilisées ;
- le second apporte la description formelle du modèle ;
- le troisième décrit le champ d'application du modèle par le biais de conditions de validité et de règles d'usage ;

- tout ce qui touche aux essais de validation est rassemblé dans le quatrième chapitre ;
- le cinquième rassemble les références bibliographiques.

Nous nous sommes posés les questions suivantes :

- Les rubriques sont-elles remplies et correctement remplies ?
- Que faut-il faire pour inciter les utilisateurs à remplir toutes les rubriques ?

Le travail de documentation est un travail pénible, la description d'un modèle dans le formalisme des fiches Proforma demande un effort non négligeable. Le temps de rédaction d'une fiche Proforma peut varier entre une journée et trois semaines, selon que le rédacteur possède ou non une bonne connaissance de la méthodologie du Proforma, selon qu'il soit ou non l'auteur du modèle et selon la complexité du modèle lui-même. Ainsi, il n'est pas rare de trouver des Proformas incomplets ou incorrectement remplis. Pour aider un utilisateur à écrire des Proformas, il serait souhaitable de disposer d'un logiciel ; un premier travail en ce sens a été mené au CSTB, il s'agit du logiciel PROFSYS [ESCUDIER 92] (PROFSYS est un prototype logiciel, implanté sur MacIntosh ; il utilise l'application HyperCard). Ce travail a permis une certaine standardisation du vocabulaire : on y retrouve des listes préétablies d'hypothèses, d'objets physiques, de phénomènes rencontrés en thermique du bâtiment. Il permet, d'une part de réaliser un gain de temps dans l'écriture des fiches Proforma (certaines tâches sont automatisées, comme par exemple la génération automatique des schémas-blocs), et d'autre part d'homogénéiser la présentation des documents papiers et du vocabulaire.

Il serait possible d'améliorer cette version, notamment en introduisant des vérificateurs :

- vérificateurs de formules, du point de vue des unités (équations aux dimensions) ;
- vérificateurs de cohérence ; par exemple déclarer des modèles amonts et pas de variables d'entrée.

Cependant, la version sur Macintosh n'est pas directement utilisable au sein de l'environnement Aïda, il serait plus pertinent de développer une version compatible avec le modèle abstrait des fiches Proforma utilisées au sein de l'ESI (les fiches Proforma des modèles incorporées dans les environnements de simulation sont des instances de ce modèle abstrait).

Remarque : quand on instancie le modèle abstrait de fiche Proforma, on l'adapte aux spécifités de description des modèles dans les différents codes de calcul considérés ; par exemple, le logiciel TRNSYS nécessite la définition de dérivatives (cf § 1-4-1) ; COMIS ignore cette notion, tout comme il ignore la notion de variables d'entrée/sortie.

Pour inciter un auteur à remplir une rubrique et à la remplir correctement, il faut non seulement lui donner des aides, mais lui expliquer aussi à quoi vont servir les informations. Si l'on explique au développeur de modèles que les informations contenues dans les Proformas seront manipulées dans le cadre d'un système expert (connexions automatiques, aide au choix de modules, aide à l'introduction des cartes de contrôle, aide au choix des valeurs des paramètres), cela l'incitera à remplir les fiches Proforma, car il prendra conscience de la valorisation qu'apporte à son modèle cet ensemble d'informations.

Un ensemble de fiches Proforma constitue une base de données à partir de laquelle il est possible d'élaborer des bases de connaissances, et donc des systèmes experts. Il faut

pour cela exploiter les informations contenues dans les fiches pour élaborer des règles de production. L'exploitation informatique des fiches Proforma n'est donc pas réduite à une simple manipulation de document. Elle peut s'effectuer autour des axes suivants :

- les connaissances qui donnent lieu à l'expression de règles de production ;
- les connaissances qui permettent d'automatiser certaines tâches, par exemple celles relatives à l'introduction de valeurs numériques (valeurs par défaut) ;
- les connaissances qui permettent de réaliser des tests de vérification (comparaison des valeurs des paramètres avec leur domaine de validité).

1-5-4 Mise en place d'une couche experte

L'étude bibliographique concernant les systèmes experts, en particulier centrée sur l'utilisation des systèmes experts dans le domaine de la modélisation/simulation a montré :

- qu'il s'agit encore de travaux de recherche ;
- que les applications sont peu nombreuses et très focalisées [ROBIN 91], [BAER 92] ;
- que le problème essentiel se situe au niveau de la constitution de la base de connaissances ; il existe peu de connaissances expertes correctement formalisées pour l'aide à la modélisation des bâtiments et pour l'aide à la simulation de leurs comportements.

D'une façon générale, la mise en place d'une couche experte a pour rôle de prendre en charge tout ou partie des mécanismes de raisonnement, qui sont utilisés habituellement dans le cadre d'un outil de simulation. Ces mécanismes font souvent appel à des connaissances éparses, parcellaires souvent d'origine expérimentale.

Ces différents raisonnements peuvent être reproduits en faisant appel à la méthodologie des systèmes experts. En effet, leur structure et leur fonctionnement se prêtent bien à la reproduction des facultés humaines de décision.

Le concept de système expert regroupe deux notions qui sont les connaissances nécessaires pour résoudre un problème et les mécanismes de raisonnement exploitant ces connaissances, appelés aussi moteurs d'inférences.

Les systèmes à base de connaissances peuvent être utiles dans le cadre d'un ESI. Illustrons nos propos : un utilisateur de TRNSYS doit, pour pouvoir simuler, construire un diagramme de flux, c'est à dire qu'il doit choisir les modules constituant son assemblage ainsi que les connexions à réaliser ; ces connexions obéissent à des règles, qui sont implicites pour un utilisateur chevronné. Par exemple, l'objet "capteur solaire" possède comme entrées :

- la température extérieure "Te" ;
- le rayonnement total incident "Ht" sur la surface du capteur ;
- le débit du fluide "mi".

L'utilisateur chevronné sait que la variable "Te" est issue directement d'un fichier météorologique, que le rayonnement total incident "Ht" est soit issu d'un fichier météo, soit issu d'un objet de type "processeur météo" et que le débit du fluide "mi" est une des sorties d'un objet de type "pompe". Ce savoir peut être formalisé par des règles de production qui permettront, à tout le moins des vérifications concernant l'exactitude et la

complétude des assemblages, et à terme d'automatiser la construction de schémas (cf. Chapitre 3).

1-5-4-1 Connaissances dans un système expert

Les connaissances peuvent être permanentes, temporaires ou une hypothèse émise au cours d'un raisonnement. Dans de nombreux systèmes experts, une distinction est faite entre la base de connaissances contenant la connaissance permanente et la base de faits contenant des connaissances et hypothèses initiales.

La connaissance peut être qualifiée de différentes façons :

- par sa précision ;
- par la certitude avec laquelle elle est donnée ;
- par son évolution au cours du temps ;
- par la finesse de détail de la description du problème à traiter.

En Intelligence Artificielle, il est indispensable de préciser clairement les limites du domaine d'application auquel se rattache les connaissances manipulées, on parle de *monde clos*.

Le mode de représentation des connaissances doit posséder un certain nombre de propriétés :

- celui-ci doit rester simple ;
- le temps de calcul doit rester un facteur à ne pas négliger.

Notons qu'il existe deux grands types de représentation de connaissances :

- les représentations déclaratives dans lesquelles les connaissances sont décrites indépendamment de leur exploitation ultérieure ;
- les représentations procédurales, qui contiennent la façon dont les connaissances doivent être utilisées.

1-5-4-2 Le raisonnement en Intelligence artificielle

Il existe une grande diversité de modes de raisonnement en IA, on peut citer :

- le raisonnement formel fondé sur la manipulation syntaxique de structures symboliques à l'aide de règles ;
- le raisonnement procédural dans lequel toutes les connaissances, la façon de les utiliser et la façon de conduire le raisonnement sont entièrement figés ;
- le raisonnement par analogie consistant à mettre en correspondance deux ou plusieurs situations ; on peut introduire à ce niveau de la notion de raisonnement par cas ;
- le raisonnement qualitatif basé sur la description physique d'un système.

Objectifs et conduite du raisonnement

L'objectif du raisonnement reste la résolution de problèmes. Raisonner nécessite le plus souvent l'examen d'un ensemble de solutions possibles. Ce parcours s'effectue à l'aide des méthodes classiques : en largeur, en profondeur ...

Il existe deux grands types de démarche en résolution de problèmes pour conduire un raisonnement face à un certain but recherché :

- partir des données disponibles sur l'état courant du problème à résoudre et utiliser les connaissances pour progresser vers une ou plusieurs solutions. Cette démarche est qualifiée d'ascendante, elle correspond au chaînage avant des règles ;
- partir du but et des connaissances disponibles pour transformer ce but en sous buts pour vérifier une situation ou une hypothèse sur les données du problème. Cette démarche est quant à elle qualifiée de descendante, elle correspond au chaînage arrière des règles.

Il est possible d'adopter une démarche bidirectionnelle combinant les deux démarches précédentes (chaînage mixte).

Difficultés dans la conduite d'un raisonnement

Un des problèmes auquel chacun est confronté reste l'imprécision des connaissances et des données sur lesquelles il raisonne. La conception des systèmes à base de connaissances performants nécessite donc de disposer de mécanismes définissant l'imprécision, et qui en tienne compte lors de sa propagation au cours des étapes de raisonnement.

Ce problème est résolu partiellement en faisant des suppositions sur les connaissances imprécises ou manquantes. La source la plus habituelle des suppositions est constituée de *valeurs par défaut* et de *domaines de validité* provenant de nombreuses origines :

- l'hypothèse du monde clos conduit à ne pas initialiser ou à ne pas accepter par exemple dans le cadre d'un projet ayant pour monde clos un bâtiment une température de zone de 150 °C ;
- le contexte du raisonnement peut permettre de deviner ou de calculer des informations manquantes (par exemple, le site d'implantation du bâtiment, centre urbain, rase campagne, bord de mer, permet de calculer le coefficient d'exposition au vent, lequel permet de calculer le débit de perméabilité à l'air des façades).

En plus de l'imprécision qui les entachent, les connaissances sont souvent caractérisées par leur évolution au cours du temps. Il est courant d'avoir à réviser au cours d'un raisonnement une décision prise antérieurement. Cela implique de disposer de mécanismes de gestion des hypothèses selon l'évolution des choses. Cela introduit donc les concepts suivants [HATON 91] :

- le concept d'évolution ; des faits nouveaux viennent contredire des faits acquis ;
- le concept de non-monotonie ; l'apport de nouvelles connaissances vient modifier des hypothèses formulées précédemment.

1-5-4-3 Réalisation pratique d'un système expert

La réalisation pratique d'un système IA sur un ordinateur pose des problèmes d'architecture matérielle et logicielle. L'un des problèmes reste la lenteur du moteur d'inférence interprétant une base de règles. Cette lenteur peut être gênante lorsque les contraintes de temps deviennent fortes (cas des interfaces). Dans le cycle de fonctionnement d'un moteur d'inférence, une part majeure du temps est consacrée à l'étape de recherche des régles applicables (parfois plus de 90%).

Système à base de règles

Le noyau d'un système à base de connaissances utilisant des règles comprend trois parties :

- une base de règles constituant la connaissance permanente ;
- un moteur d'inférence qui est le mécanisme de raisonnement exploitant ces règles ;
- une base de faits qui représente la mémoire de travail du système.

Une règle est généralement rédigée de la façon suivante :

[Si **Conditions** alors **Conclusions** (ß)]

La partie *conditions* doit être vérifiée pour que la régle s'applique. La partie *conclusions* correspond à l'action déclenchée lors de l'activation de la règle. Le coefficient ß traduit l'incertitude sur la connaissance exprimée par la règle, appelé degré de vérité ou coefficient de vraisemblance.

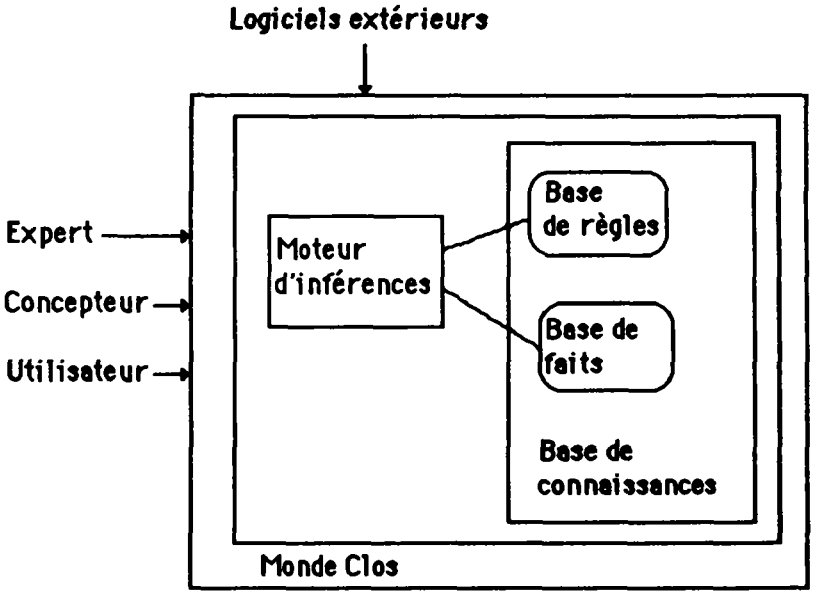

Figure 1-5 : Organisation d'un système à base de connaissances

Fonctionnement d'un moteur d'inférence

Le fonctionnement d'un moteur d'inférence comprend deux phases : la phase d'évaluation et la phase d' exécution. La phase d'évaluation comprend trois étapes : la sélection, le filtrage et la résolution de conflits.

La sélection détermine à partir d'un état présent ou passé de la base de faits et d'un état présent ou passé de la base de règles les faits et les règles qui méritent d'être comparées lors de l'étape de filtrage.

Le filtrage détermine les règles dont les conditions de déclenchement ont été jugées satisfaisantes.

La résolution de conflits détermine les règles qui doivent être effectivement déclenchées, ainsi que l'ordre de leur déclenchement, dans la phase d'éxécution

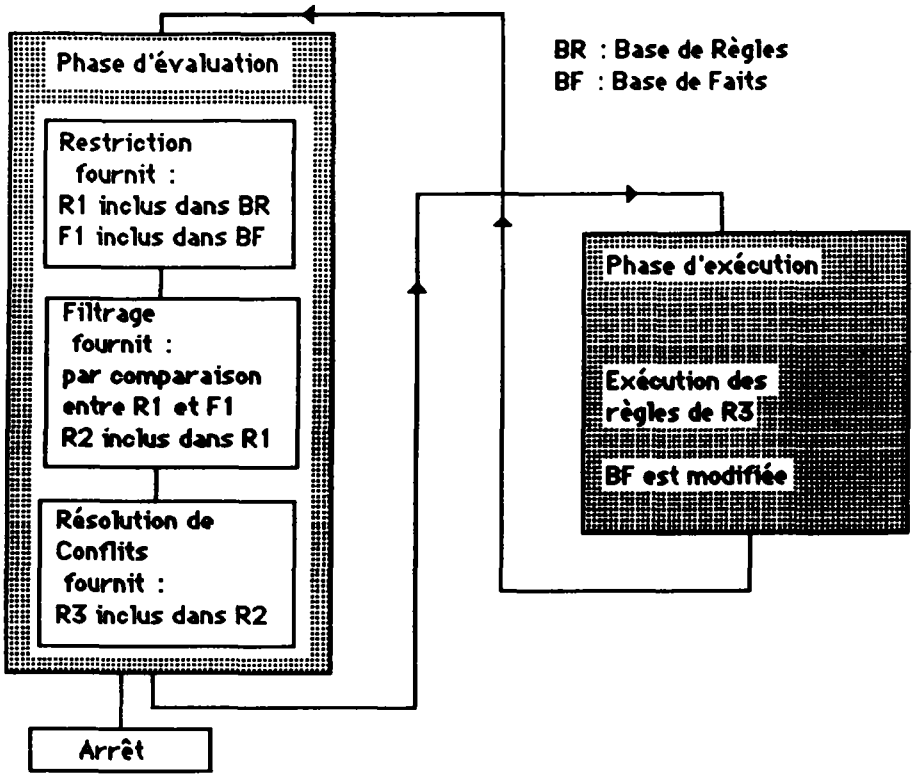

Figure 1-6 : Principe de fonctionnement des moteurs d'inférence [FARRENY 87]

Conclusion du chapitre 1

Nous avons définis au paragraphe 1-1 les grandes lignes (objectifs) des environnements de simulation que nous entendons développer. Pour que ces environnements soient à la fois faciles à faire évoluer, et qu'ils permettent d'offrir à l'utilisateur final toute une panoplie d'aides au processus de Modélisation/Simulation/Analyse des résultats, la programmation orientée objet nous est apparue comme la solution logicielle la mieux adaptée (cf. § 1-3-1). Elle est atout pour le développeur au sens où elle permet une programmation très bien structurée (définition de classes et d'un graphe d'héritage). Cette structuration rend le logiciel particulièrement facile à modifier et à faire évoluer. Pour l'utilisateur final, outre le fait qu'il peut bénéficier à moindre coût de versions améliorées, il peut éventuellement tirer parti des propriétés d'héritage. Les bibliothèques de modèles peuvent (doivent) être structurées selon une approche orientée objets ; ainsi, les propriétés génériques (attributs, méthodes) définies pour une classe seront automatiquement héritées par tout nouveau composant créé par un utilisateur final, et rangé par lui sous cette classe.

Nos réflexions ont ensuite porté sur le choix d'un environnement logiciel : nos analyses nous ont conduit à choisir le langage Le_Lisp, enrichi de la boîte à outils Aïda, au sein de l'environnement UNIX/X-Window (cf. § 1-3-2).

Un de nos objectifs est de démontrer à travers une application concrète la pertinence des concepts génériques que nous développons dans le cadre du projet ESI. Pour réaliser cette première application, nous avons recherché un logiciel modulaire, qui possédait déjà une bibliohèque de composants relativement étendue ; nous avons choisi le code de calcul TRNSYS (cf. § 1-4), qui est largement utilisé dans le monde de la recherche. Cette application spécifique dénommée IISIBât (Interfaces Intelligentes pour la Simulation dans le Bâtiment) est fondée sur la modélisation de systèmes complexes par assemblage de composants élémentaires. Les travaux relatifs au développement de cette application sont exposés au chapitre 2.

Cette application (IISIBât) doit permettre l'amélioration de la productivité des utilisateurs des codes de simulation ; nous pensons que cette amélioration doit passer par :

- la mise en place d'une interface conviviale, où l'efficacité du dialogue Homme/Machine est prise en compte (cf. § 1-5-1) ; pour cela, nous avons retenu deux approches qui permettent de modéliser l'interaction Homme/Machine, il s'agit du modèle STI (Système de Taitement de l'Information) et des modèles centrés objets ;

- la mise en place d'un système d'aide, intégré à l'application interactive, qui concerne le fonctionnement de l'application interactive et l'outil de simulation ; nous pensons qu'un système qui comporte en son sein plusieurs types d'aide (ciblé, global, statique, interactif, et non adapté à l'utilisateur -cf. § 1-5-2-) permet de répondre de façon efficace aux attentes des utilisateurs ;

- la mise en place d'un système d'aide qui concerne l'utilisation des composants des bibliothèques ; nous pensons que la structure du Proforma (cf. § 1-5-3) est bien adaptée à la documentation des composants des bibliothèques ;

- la mise en place de mécanismes fondés sur la manipulation de règles expertes qui guident l'utilisateur vers des solutions satisfaisantes ; ces règles expertes peuvent être écrites à partir des informations contenues dans les fiches Proforma ; en effet, un ensemble de fiches Proforma constitue une base de données à partir de laquelle il est possible d'élaborer des bases de connaissances, et donc des systèmes experts. Ces systèmes pourraient concerner :

    - l'aide au choix des systèmes, en fonction par exemple des objectifs de la simulation ;
    - l'aide au choix des modules, en fonction des objectifs de la simulation ou en fonction de modules constituant un début de solution ;
    - l'aide à la connexion des variables entre modules ;
    - l'aide à la manipulation des données numériques (vérification de validité, aide au choix de valeurs manquantes) ;
    - l'aide à l'introduction des paramètres de la simulation (pas de temps, début et fin de la simulation, tolérance de convergence) ;
    - l'aide à l'analyse des résultats (aide au choix de valeurs sensibles).

A priori, pour le type de connaissances que nous serons amenés à manipuler, et pour le type de raisonnement que nous comptons éventuellement introduire, la méthodologie des systèmes à base de connaissances semble bien adaptée (cf. § 1-5-4). Ainsi au chapitre 3, nous développons les divers types de raisonnement qui peuvent aider les utilisateurs dans leurs démarches de Modélisation/Simulation/Analyse des résultats.