الجمهوريــة الجزائريــة الديمقراطيــة الشعبيــة

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليـم العالـي و البحـث العلمـي

جامعـــــة قسنطينـــــة 1

كليــة علوم التكنولوجية

قسم هندسة التكييف

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE CONSTANTINE I

Faculté des Sciences de la Technologie

Département Génie Climatique

N° d'ordre : . . . . . .

N° de série : . . . . . .

THÈSE

Pour obtenir le diplôme de

DOCTORAT EN SCIENCES

Présentée et soutenue par

Abdeldjebbar Bachkhaznadji

Le ……..

CONDITIONNEMENT D'AIR : CONCEPTION OPTIMALE

Jury :

Mr. Zine Labidine Mahri Professeur Président (Université Constantine 1)

Mr. Azeddine Belhamri Professeur Directeur des études (Université Constantine 1)

Mr. Abdessalem-Hassan Meniai Professeur Examinateur (Université Constantine 3)

Mr. Abdelhamid Ayadi Professeur Examinateur (Université Oum El Bouaghi)

Mr. Abdelmalik Bachtarzi M. de Conférence Examinateur (Université Constantine 1)

بسم الله الرحمان الرحيم

Déclaration

Aucune partie de ce travail présenté dans cette thèse n’a été présentée à l'appui d'une demande d’un autre diplôme, d’un autre titre professionnel, d’un autre titre universitaire ou pour un autre établissement d'apprentissage.

Abdeldjebbar Bachkhaznadji

Remerciements

Le travail présenté dans cette thèse a été réalisée sous la supervision du professeur Azeddine Belhamri, à qui l'auteur est reconnaissant pour ses précieux conseils et assistance. Remerciements vont aussi au professeur Abdessalem-Hassan Menai pour son aide tout au long de ce travail.

"La correction de l'erreur est peut-être l'événement le plus sublime de la vie intellectuelle, le signe suprême de notre nécessaire soumission à une réalité plus vaste et de notre incapacité à construire le monde selon notre désir"

S. J. Gould

Liste des symboles

IBPSA: International Building Performance Simulation Association

DOE: Department of Energy

EKS: Energy Kernel System

IU: User Interface

IDD: Input Data Definition

UML: Unified Modeling Language

RDPs: Réseau de Pétri

ICG : Interface Conviviale Graphique

GES : Gaz à Effet de Serre

DOD : Département américain de la Défense

CERL: U. S. Army Construction Engineering Research Laboratories

LBNL : Laboratoire National Laurent Berkeley

OSU : Université de l'état d'Oklahoma

I/O : Input/Output

CVC : Chauffage, ventilation et conditionnement d'air

RE : Réingénierie Evolutive

CFD : Computational Fluid Dynamics

EMPD : Effective Moisture Penetration Depth

CTA : Centrale de Traitement d'Air

CAO : Conception Assistée par Ordinateur

IFC : Industry Foundation Classes

OMG : Object Management Group

MVC: modèle d'architecture(Model View Controlor)

GOMS: Goal-Operator-Method-Selection

UIMS : User Interface Management Systems

MOO : méthodes orientées objet

OOUI : Object-Oriented User Interface

AOO : Analyse Orientée Objet

COO : Conception Orientée Objet

POO : Programmation Orientée Objet

IDDClassD: Input Data Definition Class Diagram

IDDUseCaseD: Input Data Definition Use Case Diagram

UC: Use Case (cas d'utilisation)

IDDSequenceD: Input Data Definition Sequence Diagram

Mes: message

PN: Réseaux de Pétri

P: Place

T: transition

RdPC : Réseaux de Pétri colorés

CU : Cas d'utilisation

GOA : Goal Oriented Approach

CPN : Colored Petri net

S: Place dans le réseau de Pétri

[*rc, ec*] : couleurs composite

GT : graphe de transitions

IB : sous-graphe du graphe GT

GIB : graphe de blocs d'interface

Liste des figures

Chapitre I

Chapitre II

Chapitre III

Chapitre IV

Résumé

Les outils de simulation numérique occupent une place prépondérante dans les études d'optimisation en physique du bâtiment ainsi que dans les systèmes d'air conditionné. La démarche de la conception énergétique optimale des bâtiments et des systèmes d'air conditionné adhérents résulte de l'épuisement des ressources énergétiques fossiles et du souci de préservation de l'environnement qui n'a fait qu'amplifier leur utilisation. Dans beaucoup de pays, plus particulièrement les pays occidentaux, ces outils si répondus dans la communauté scientifique, n'ont pas encore trouvé leur place parmi la communauté scientifique algérienne et plus particulièrement chez les professionnels. L'enjeu est donc de faire un transfert technologique de ces outils. Contrairement à l'utilisation d'un produit fini (Black box), on devrait plutôt initier une méthodologie de développement d'outils de simulation. Cela permettra aux scientifiques algériens, brasés dans le domaine du génie climatique, l'utilisation des mécanismes de conception qui tiennent compte des nouvelles technologies informatiques et d'assurer leur formation dans le domaine de la simulation.

Un large éventail d'outils de simulation des performances énergétiques des bâtiments et des systèmes d'air conditionné, validés scientifiquement, sont disponibles de part le monde. On recense 360 outils de simulation. Les utilisateurs de ces outils sont principalement des chercheurs, des physiciens et des experts. Cependant, la littérature et les études comparatives montrent que la plupart des praticiens sont beaucoup plus intéressés par (1) l'intégration de la conception intelligente basée sur l'information et la connaissance, (2) la convivialité et la facilité de l'utilisation des interfaces usagers proposées avec ces outils de simulation. Ces deux points sont les principaux facteurs d'identification d'un programme de simulation comme : '' *interface conviviale graphique (ICG)*''.

La démarche de modélisation classique se focalise dans la plupart des cas, sur la description d'un problème, sa représentation mathématique et sa résolution numérique. La méthode décrite ici intègre la notion de compréhension du modèle par un utilisateur. Dans cette démarche, le modélisateur doit garder à l'esprit le profil des utilisateurs potentiels de son modèle du système et d'évaluer leur niveau d'aptitude.

Modéliser un système avant sa réalisation permet de mieux comprendre son fonctionnement. C'est également un bon moyen de maîtriser sa complexité et d'assurer sa cohérence. Un modèle est un langage commun, précis et connu par tous les membres de l'équipe de développement. C'est donc un vecteur privilégié pour communiquer. Cette communication est essentielle pour aboutir à une compréhension commune et précise aux différentes parties prenantes d'un problème donné. Dans le domaine de l'ingénierie du logiciel, le modèle permet de mieux répartir les tâches et d'automatiser certaines d'entre elles. C'est également un facteur de réduction de délais.

La description du modèle accorde une grande importance à l'implémentation informatique du système bâtiment- air conditionné. Elle fait appel aux techniques de la conception orientée-objets décrites par les normes et les spécifications en notation UML 2.0 (Unified Modelling Language, version 2006) et à laquelle se sont rangés tous les grands acteurs du domaine, acteurs qui ont d'ailleurs contribué à son élaboration (IBM, Microsoft, Oracle, DEC, HP, Rational, Unisys etc.). Le concept UML est un langage graphique qui permet de représenter et de communiquer les différents aspects d'un système d'information.

La mise en œuvre des ICG apporte toute fois une nouvelle complexité dans la tache de développement. Cette complexité réside dans la description du dialogue multitâche qui est exécuté par l'utilisateur afin de mener plusieurs taches à la fois. Au niveau de la spécification de l'interface utilisateur, les développeurs peuvent dessiner l'interface. Mais il est souvent difficile de spécifier leurs comportements dynamiques en raison de la complexité du dialogue. Des méthodes formelles telles que les réseaux de Pétri ou les diagrammes d'états de Harel, seront d'une grande utilité dans la spécification du comportement dynamique de l'interface avant sa mise en œuvre. Cela permettra de réduire le temps de tests et d'améliorer la qualité de l'interface.

Le principal apport dans ce travail, outre le cadre méthodologique lié au processus itératif proposé, réside dans la mise en œuvre d'une nouvelle plateforme qui permet la création de prototype d'interface usager spécifique au fichier IDD d'entrée des données du programme de simulation des performances énergétique des bâtiments et des systèmes CVC, *EnergyPlus*. La notation UML et les réseaux de pétri de haut niveaux sont utilisés pour la première fois dans le processus de développement d'interface usager pour le programme *EnergyPlus*. Cette méthodologie permet la génération des aspects statique et dynamique de l'interface graphique à partir des spécifications de comportement et des données.

Mots Clés : Génie Climatique ; Modélisation UML ; Réseaux de Pétri ; Interface conviviale graphique; Simulation ; Bâtiment ; Air Conditionné, Performance Energétique, Conception Optimale

Sommaire

Page

Introduction

Chapitre 1

Chapitre 2

Page

Page

Introduction & Objectifs

I. En ce début du 21eme millénaire qualifié de technologique, l'heure est propice aux bilans. Des termes surgissent couramment dans la littérature tant profane que spécialisée, notamment :

- Le développement économique et sa disparité mondiale.
- Les ressources énergétiques et leur pérennité.
- L'environnement et son devenir.
- Conception optimale.

Il n'est pas étonnant de regrouper ces mots car le développement économique, les ressources énergétiques et l'environnement sont inextricablement liés. En effet le développement économique entraîne des exigences grandissantes de la société dans sa course vers l'amélioration de son bien-être. En parallèle, une sensibilisation sur l'épuisement des ressources énergétiques doit être développée dans cette même société. Le développement économique, avec une utilisation intensive des énergies est souvent mal gérée, et met en péril notre écosystème. La conférence de Kyoto, par exemple, a tiré la sonnette sur les gaz à effet de serre (GES) et un protocole a été signé pour la réduction des GES. L'implication de l'utilisation rationnelle de l'énergie est directe. Elle ne fait que renforcer la démarche de maîtrise énergétique entamée suite aux crises pétrolière mais cette fois ci avec une forte composante environnementale.

La diminution de la consommation d'énergie primaire est toujours d'actualité. On se retourne donc vers les consommateurs d'énergie. Un des gros consommateurs, identifié depuis les années 70 est le secteur du bâtiment. Ce constat a engendré un développement de la recherche en énergétique afin de tendre vers une conception optimale du bâtiment. Les axes de recherche s'étendent de l'enveloppe (isolation, matériaux, vitrage) jusqu'aux différents type d'installations techniques (ventilation, air conditionné etc.) en incluant les systèmes de régulation et gestion technique.

Pour mieux comprendre et prédire les phénomènes intervenant dans le bâtiment, la communauté scientifique fait appel aux techniques de la modélisation et de la simulation numérique. Il en découle que les outils de simulation sont devenus incontournables pour toute étude en énergétique du bâtiment liée aux exigences de confort et de santé ainsi qu'aux questions économiques et impactes environnementaux.

L'utilité grandissante des outils de simulation dans la démarche de la conception optimale des bâtiments et des systèmes de ventilation et d'air conditionné se fait de plus en plus sentir dans notre pays, même si le problème de l'énergie ne se pose pas actuellement en Algérie. La conception optimale met en évidence un nouveau défi : le transfert technologique des outils de simulation de la communauté scientifique vers des professionnels. Cette idée s'appuie en outre sur les avancées technologiques en informatique et électronique qui ont rendu les ordinateurs de plus en plus accessibles à de faible coût.

Un premier constat dévoile que dans les différents instituts spécialisés dans le génie climatique et ses applications dans le bâtiment n'ont jusqu'à ce jour, pas eu recours à ce type d'outils de simulation, ni entrepris un programme sérieux de recherche et de développement pour l'élaboration d'outils de simulation qui tiennent compte de la réalité de la communauté professionnelle algérienne dans le secteur du bâtiment et de la construction. Par voie de conséquence, aucune expertise ni transfert technologique vers ce secteur ne peut être engagé pour le moment.

Le travail présenté dans ce mémoire entre dans le cadre de la recherche et développement et a pour objectifs de faire :

- montrer l'importance de l'utilisation de la simulation dans l'usage du génie climatique qui est le notre.

- Introduire une nouvelle démarche pour initier une nouvelle méthodologie de développement d'Interface usager basée sur la modélisation UML, l'approche scénario, et la description du comportement dynamique de l'interface graphique par les réseaux de Pétri. Pour cela, le programme de simulation des performances énergétiques du bâtiment et les systèmes de chauffage, ventilation et conditionnement d'air (CVC) ''*EnergyPlus*'', du département d'énergie américain, a été adopté. Ce programme est conçu principalement en tant que moteur de simulation, le développement d'interface usager est laissé à une tierce-partie.

- Permettre à la communauté scientifique algérienne, versée dans le domaine du génie climatique, une meilleure compréhension des concepts cités ci-dessus, facilité leur utilisation

et leur donner un moyen de développement qui s'adaptent au mieux à notre environnement, marché et législation.

- On cherche aussi à initier une nouvelle plateforme qui s'appuie sur de nouvelles technologies de développement informatiques tout en profitant des avancées considérables réalisées par la communauté internationale dans le cadre de la modélisation et des techniques de simulation dans le domaine du génie climatique.

- Dans ce travail, nous essayons d'apporter des éléments de réponse aux problèmes de conception d'IU, dans le cadre du développement d'interface usager pour le programme EnergyPlus, par la proposition d'un cadre méthodologique supporté par des techniques formelles. Notre intérêt porte particulièrement sur la conception d'interfaces dans l'approche orientée objet. Ainsi nous avons basé nos travaux sur la méthode unifiée (Unified Modeling Language: UML) qui émerge comme un consensus des méthodes orientées objet et qui supporte bien l'approche scénario.

II. Organisation du mémoire

Le mémoire est constitué de 4 grands chapitres. La problématique est abordée au début du premier chapitre en faisant une revue des travaux en cours dans le processus de recherche et développement des outils de simulation. Certaines notions fondamentales qui sont utilisées par la suite sont décrites.

Le chapitre II traite le programme de simulation des performances énergétiques du bâtiment et des systèmes d'air conditionnés « EnergyPlus, version 3.0 et plus » dans sa globalité tout en focalisant sur l'aspect de l'interface usager qui vient avec le programme.

Au chapitre III, la notation UML et l'utilisation des scénarios sont introduites afin de faciliter la compréhension de l'approche permettant la génération du prototype de l'IU.

Ce chapitre traite aussi de la vision systèmes et formalisation des scénarios à l'aide des réseaux de Pétri. L'approche introduit la modélisation des systèmes interactifs qui utilise les réseaux de Pétri à haut niveau comme formalisme de spécification. Cette approche propose un processus d'ingénierie des exigences permettant de produire un prototype d'interface usager, ainsi que la spécification du comportement global du système à partir des scénarios.

Enfin ce travail se termine par la présentation de la méthodologie (chapitre IV) qui permet la génération de prototype d'interface graphique du fichier IDD du programme de simulation des performances énergétiques *EnergyPlus* à partir des spécifications comportementales du système. Cependant, nous utiliserons également la spécification des données pour déterminer le type de l'aspect présentation de l'interface (Widget) approprié pour une interaction donnée.

Chapitre 1

Contexte

Notions de base

Ce chapitre présente l'évolution des outils de simulation employés en thermique du bâtiment et les installations de ventilation et d'air conditionné. Le cadre dans lequel se situe le présent travail est aussi exposé. Les notions de base, à l'exclusion de la thermique, nécessaire pour mener a bien cette étude sont mises en évidence. Ensuite on aborde le processus de modélisation classique pratiqué dans le domaine de la thermique du bâtiment et de ces installations et les grandes tendances lors du développement des outils de simulation numérique. L'introduction des techniques de programmation graphique dans la suite comme elles sont utilisées dans les environnements de simulation. La genèse de la modélisation graphique UML™ est mise au clair.

1.1. **CONTEXTE**

Le développement des outils de simulation se fait depuis plus d'une vingtaine d'années dans le domaine de la physique du bâtiment et ces installations techniques et en particulier en thermique. Il existe actuellement une multitude d'outils de simulation de par le monde. Quelques outils sont à caractère général alors que d'autres sont développés pour des études spécifiques (adhoc). Une partie de ces outils atteignent de nos jours une certaines maturité (robustesse, fiabilité, champ d'utilisation…).

Cette nécessité d'outils de simulation s'est fait dans un premier temps principalement parmi la communauté scientifique (universités, centre de recherche privés et publics) pour les raisons qui suivent :

- La course vers la réduction du coût suite aux crises pétrolières, plus particulièrement dans les pays occidentaux.

- Les exigences croissantes des occupants au niveau de la qualité de vie dans les bâtiments.

- Les progrès techniques nécessitant des études de plus en plus délicates pour la compréhension des phénomènes. Par exemples, on ne se contente plus de travailler en fonctionnement stationnaire des systèmes thermiques mais on s'intéresse aux régimes non stationnaires.

Vu l'utilité des outils de simulation sur le plan de la recherche, l'idée d'étendre leur utilisation dans le milieu professionnel a fait jour. Cette idée s'est appuyée sur deux faits importants :

- pendant les années 80 et surtout au début des années 90, on a assisté à des progrès considérables dans les méthodes numériques et la puissance de calcul des outils informatiques. Les chercheurs en ont fait bon usage pour apporter des améliorations importantes sur les codes de calcul (temps de calcul, finesse de résolution, portabilité …) et ont donc augmenté les champs d'utilisations des outils de simulation.

- l'utilisation des ordinateurs s'est fortement banalisée. En effet, l'ordinateur se fait son chemin dans tous les secteurs d'activités. Les ordinateurs ont leur place chez les architectes, les bureaux d'ingénierie, les industriels et autres praticiens dans le secteur du bâtiment. De plus ils disposent d'une puissance largement acceptable pour accueillir les outils de simulation développés dans les centres de recherche. Ce transfert d'outils informatiques a débuté avec les outils de CAO et DAO. Ensuite des outils analytiques dans le bâtiment pour des calculs statiques (dimensionnement, résistance des matériaux, etc.) sont apparus. Maintenant, on arrive aux outils de simulation dynamique. Néanmoins, l'utilisation des outils de simulation dynamique se limite encore à des bureaux d'études très spécialisés ou des centres de recherche et développement de grands industriels

La simulation devrait, à priori, apporter une aide lors des calculs ou de la conception optimale des équipements ou installations et réduire les coûts d'études. Or le praticien

perçoit la simulation complexe, peu fiable et de surcroîts coûteux. En bref, le transfert des outils de simulation se fait très difficilement vers les praticiens [Crawley et al. 99].

Enormément d'efforts sont fournis actuellement pour favoriser ce transfert d'outils de simulation. La conférence internationale de l'IBPSA (International Building Performance Simulation Association) tenue en 1997 était essentiellement consacrée à ce sujet.

Les recherches menées pour faciliter le transfert de ces outils de simulation peuvent être regroupées selon les matières suivantes :

- création d'interfaces graphiques conviviales,
- communication et transfert d'information et de données entre outils,
- actions à caractère pédagogique,
- analyse des besoins des utilisateurs d'outils de simulation.

La liste précédente est écrite suivant un ordre chronologique des thèmes de recherche. Ces derniers sont développés ci-dessous :

1.1.1. Création d'interfaces homme - machines conviviales.

Afin de faciliter l'utilisation des outils de simulation, des interfaces graphiques sont ajoutées aux codes de calculs. Les interfaces servent d'une part à définir des problèmes (données d'entrées) et d'autre part à récupérer les résultats de la simulation pour des analyses. Souvent les résultats sont sous forme graphique ou bien sont contenus dans des tableurs tels qu'Excel, etc.

C'est la première solution envisagée par les développeurs d'outils de simulation. Leur tache est de faciliter par le développement des langages avec forte prédominance pour l'aspect Orienté-Objets et visuel (Java, C++, Visual Basic, Delphi, HTML etc.). On peut cités par exemple, deux interfaces conviviales, CA-SIS [Tabary 97] et IISIBAT pour le logiciel TRNSYS [TRNSYS 96], le logiciel ENERGY-10 [Balcomb 97] développé en Visual C++ ou encore le logiciel RESVENT avec une interface HTML accessible par Internet [Web 1].

1.1.2. Amélioration de transferts d'information entre outils

La création de liens entre différents logiciels permet aux praticiens de conserver leurs outils de dessin ou de calcul et d'y intégrer des modules pour faire des études en simulation. Ainsi ils ne perdent pas l'expertise acquise en utilisant leurs logiciels.

Ils n'ont pas à recommencer l'apprentissage d'un nouvel logiciel qui impliquerait une formation et donc un coût supplémentaire.

De plus, en normalisant les formats de données des logiciels, on favorise les échanges d'informations par le biais informatique entre les intervenants dans le secteur du bâtiment. On voit tout de suite ici l'intérêt pour la communication entre le concepteur, l'architecte, le bureau d'ingénierie, etc. sur un projet immobilier par exemple.

Dans ce domaine, le projet COMBINE [Web 2] s'est préoccupé de la normalisation des modèles de données lors du développement des outils en utilisant la norme STEP [Arbouy et

al 94]. Actuellement l'International Alliance for Interoperability (IAI) [Web 3] à commencé un travail colossal pour définir un langage commun dans le secteur du bâtiment.

Un exemple de compatibilité entre différents outils a été réalisé par le bureau d'ingénierie finlandais Olof Granlund. Il faut noter que l'outil de simulation développé, RIUSKA/SMOG [Web 4] s'inspire du projet COMBINE.

1.1.3. Action à caractère pédagogique

On cherche ici à fournir des méthodes pour une utilisation efficace des outils de simulation. Les travaux s'adressent aussi bien aux développeurs qu'a des utilisateurs d'outils de simulation, Annexe 21 de l'AIE [Annexe21 98], Annexe 30 de l'AIE [Annexe30 98], CIBSE Application Manual [Bartholomew et al. 97].

L'annexe 21 s'est préoccupée de l'assurance qualité sur les calculs des performances énergétiques et environnementales des bâtiments. Elle propose un format de documentation des méthodes d'évaluation des performances – les PAMDOCS. Elle a abouti à des conseils aux développeurs et aux utilisateurs pour promouvoir la qualité de l'utilisation des outils de simulation.

Au cours de cette annexe et en collaboration avec l'international Energy Agency Solar Heating and Cooling Task 12, la question de l'évaluation des outils de simulation a été abordée dans le souci de rendre ces derniers plus crédibles vis-à-vis des utilisateurs et aussi de donner aux développeurs un moyen de tester leurs outils. Il existe maintenant un banc d'essai virtuel, le BESTEST [Judkoff 95] qui propose des cas types pour tester les performances d'un logiciel pour la modélisation du comportement dynamique de l'enveloppe du bâtiment et de méthode de diagnostique d'équipements utilisés dans les espaces conditionnés.

L'annexe 30 est orientée vers l'utilisation des outils de simulation au cours des différentes phases de cycle de vie d'un bâtiment par différent acteurs. Les travaux de l'annexe 30 montrent comment le concepteur, l'architecte, le bureau d'ingénierie en génie climatique utilisent la simulation lors de la phase de conception du bâtiment pour faire la sélection et le dimensionnement des installations techniques.

L'utilisation de la simulation pour la mise au point, la gestion technique et aussi lors de la réhabilitation est expliquée. Des études de cas sont fournies à titre d'exemples pour les lecteurs.

L'Application Manuel est réalisé sous l'égide du Chartered Institute of Building Service Engineers (UK). Il a pour but de guider les praticiens pour choisir et utiliser les outils de simulation en génie climatique.

Il ne faut pas oublier les efforts plus fondamentaux sur les concepts de modélisation et de simulation et leurs applications destinées à des étudiants en ingénierie ou à la formation continue. A travers le projet européen TEMPUS, un tel cours est disponible sur Internet en accès libre [Web 5].

Un autre moyen d'amener la simulation sur le terrain est d'avoir un organisme de conseil en matière d'outils de simulation. Un tel organisme indépendant de tout développeur de logiciels ou de bureaux d'études spécialisés existe au royaume uni, Energy Design Advice Scheme (EDAS) [Mc Elroy 97], pour favoriser la communication entre spécialistes en simulation et les concepteurs, architectes, ingénieurs ou autres dans le bâtiment. Ils interviennent tant au niveau de la conception que lors de la réhabilitation des bâtiments. Le projet, financé par l'UK Department of Trade and Industry a duré pendant 8 ans. Plus de 100 études ont été réalisées. Néanmoins, la généralisation d'un tel organisme nécessiterait l'appui de fonds publics que ce soit dans les pays industrialisés ou dans les pays en voie de développement pour pouvoir fonctionner.

1.1.4. Analyse des besoins des utilisateurs d'outils de simulation

Après une période où les développeurs d'outils travaillaient en fonction de leurs besoins propres, on note aujourd'hui un début de structuration des besoins des utilisateurs. Ceci devient possible du fait de l'existence d'un nombre croissant d'utilisateurs et de développeurs. Des ateliers de travail et forum regroupant des utilisateurs, utilisateurs potentiels et de concepteurs sont organisés, des enquêtes auprès des utilisateurs sont lancées. Il est intéressant de mentionner quelques résultats d'un atelier de travail réunissant des développeurs et des utilisateurs pour définir les besoins des nouvelles générations d'outils de simulation. Cet atelier montre que les développeurs sont conscients des difficultés rencontrées par les utilisateurs alors qu'auparavant ils considéraient la convivialité des outils comme élément secondaire et sans grand intérêt. Les mêmes sujets prioritaires ont été identifiés [Crawley et al. 97] :

- Les développeurs s'intéressent au pré et post traitement des outils de simulation (processeurs de données, traitements des résultats, graphismes, etc.). Ces idées sont similaires à la capacité d'interopérabilité et celle d'intégration soulevées par les utilisateurs.

- Les développeurs et les utilisateurs donnent une priorité à l'utilisation de la simulation lors de la phase de conception de bâtiment et des installations.

- Ils considèrent que les outils doivent en premier lieu inclure des modèles robustes des phénomènes physiques (il y a encore des améliorations à apporter dans les modèles existants) et ensuite par exemple des modèles d'installations avec les systèmes de régulation.

Parmi les enquêtes sur la simulation numérique, on note celle réalisée par [Donn 97] auprès des utilisateurs d'outils de simulation en 1996. L'étude a été réalisée en Nouvelle Zélande (80 réponses sur 82) et aux États-Unis (44 sur 421). Ici aussi on remarque que les utilisateurs demandent des améliorations de l'interface Homme -Machine :

- Pour décrire le bâtiment et les éléments constituant les installations.

- Choisir un modèle de conditions climatiques approprié pour la conception.

- Pour communiquer les performances des bâtiments et installations aux clients.

1.2. **NOTIONS DE BASE**

Dès que l'on parle d'outils de simulation, on introduit les notions de système, de modèle, de simulation, d'environnement de simulation ou encore environnement de programmation graphique. Ces termes ont souvent des significations différentes parmi les praticiens, les scientifiques, les chercheurs etc… Cette partie permet de préciser le sens donné à ces termes au sein de ce travail lié au domaine général de la thermique du bâtiment et de ses installations. Les concepts utilisés sont tirés de la littérature sur la systémique, modélisation et à la programmation graphique. On introduit les démarches qu'une personne peut adopter pour analyser ou résoudre des problèmes par le biais de la modélisation et la simulation. Ce chapitre se termine par une brève présentation de la programmation graphique basée sur les spécifications UML™.

1.2.1. QU'EST-CE QU'UN SYSTEME ?

Un système est un ensemble d'objets en fonction d'un but et immergé dans un environnement.

- le système est un outil conceptuel,
- l'ensemble doit former une identité ou une unité cohérente et autonome,
- les objets peuvent être réels ou conceptuels,
- la notion d'organisation implique des relations, interrelations, interactions dynamiques entre les objets.

(Définition de J. L le Moigne dans [Le Gallou 94]).

1.2.2. L'APPROCHE SYSTEMIQUE

Le concept de système est né de la cybernétique pour mieux arrêter la complexité organisée et est le cœur d'une approche maintenant appelée, la systémique. Cette dernière a étendu son application en économie, sociologie, physique, ingénierie, biologie… Cette approche s'est développée suite aux limites de l'analyse cartésiennes classiques, pour décrire des systèmes fortement couplés où on ne peut isoler des parties pour les résoudre séparément. L'analyse thermique du bâtiment est un exemple flagrant.

L'approche cartésienne classique s'appuie sur la décomposition de tout ensemble en éléments essentiels. Les lois de comportement de chaque élément sont étudiées avant même de déduire celui de l'ensemble. Dès que le couplage entre les éléments devient complexe, l'analyse de comportement élémentaire ne permet pas de comprendre celui de l'ensemble. C'est à ce niveau qu'intervient l'approche systémique. Elle met l'accent sur les relations, les fonctionnalités et les notions très

utilisées en automatisme telles que les boucles de rétroaction, contrôle et régulation appliqués en génie climatique.

La Figure 1.1 issue de [Gicquel 92, Durand 98], oppose aux quatre préceptes fondamentaux de la démarche scientifique classique issus du discours de la méthode de Descartes, les principes proposés d'un point de vue systémique.

L'approche systémique parle du principe de pertinence car notre intelligence s'exerce par rapport à des finalités explicitables. Nous cherchons le coté opérationnel d'un concept plutôt que l'aspect d'évidence dans l'absolu. De même, Descartes propose, à juste titre, la décomposition des difficultés en fragments pour les résoudre.

Néanmoins, il n y a aucune indication quant à la façon de procéder et une mauvaise décomposition peut augmenter la difficulté du problème. La systémique procure une

démarche plutôt ascendante où on favorise les relations entre les éléments de l'ensemble étudié.

La causalité est valable pour les systèmes qui suivent des lois déterministes, mais les systèmes, tels que systèmes experts, peuvent rompre les chaînes causales pour atteindre leurs objectifs. Le principe téléologique propose de réfléchir sur les finalités du système à connaître.

Enfin le précepte d'exhaustivité qui est difficilement applicable dans la réalité est confronté aux principes d'agrégativité et de pertinence. Toute représentation est subjective et il est judicieux de définir les règles pour sélectionner de manière pertinente les agrégats à considérer plutôt que de s'élancer dans la tache souvent herculéenne d'un recensement exhaustif.

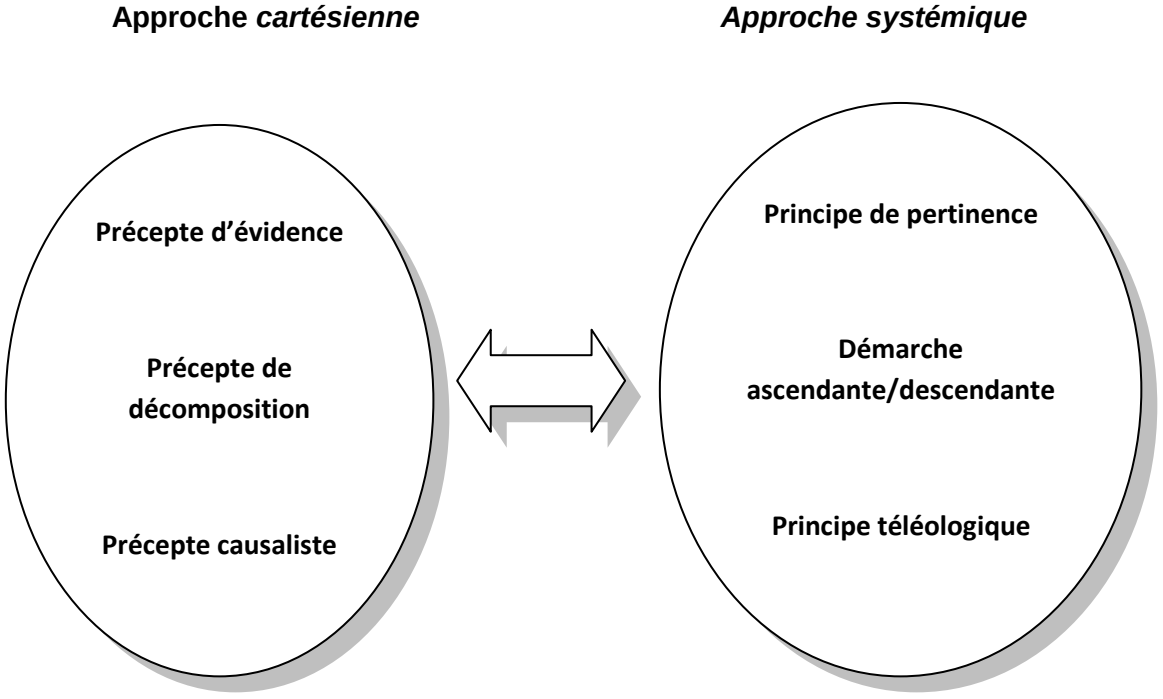


Figure 1.1 : Parallèle entre l'approche Cartésienne et l'approche Systémique

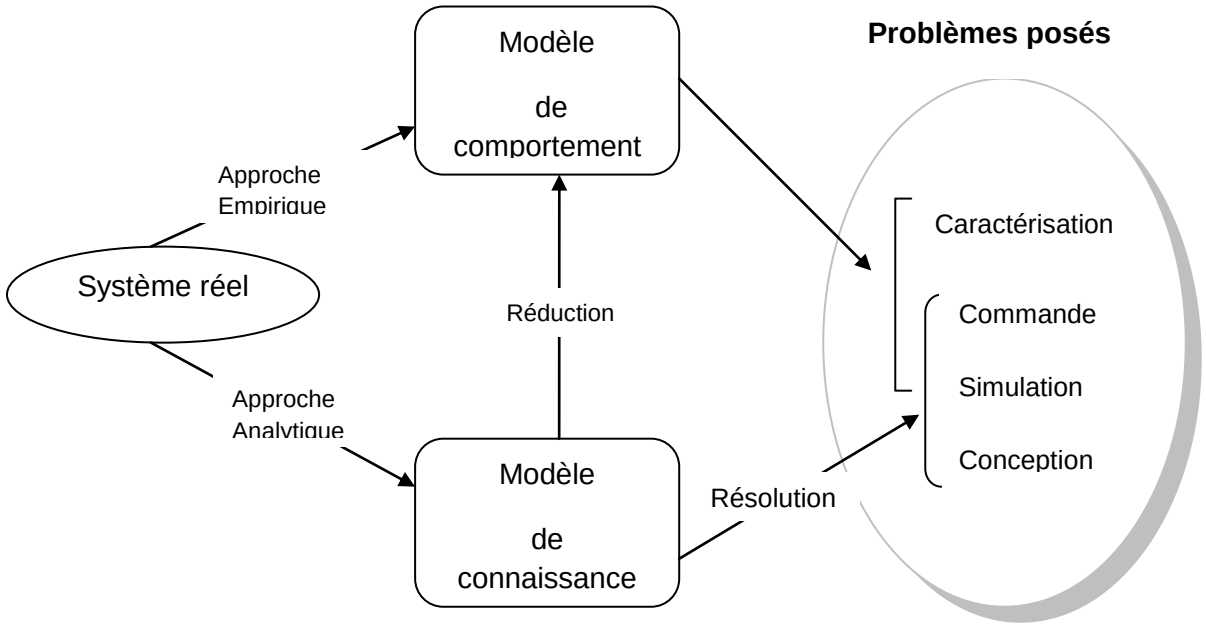


Figure 1.2 : les techniques de modélisation

1.3. LE MODÈLE

Le modèle est une représentation d'une entité réelle. A travers on a une perception partielle des phénomènes impliqués dans l'emplacement réel qui est plus complexe. On cherche à extraire des informations pertinentes de la réalité. La modélisation est utilisée pour prédire (notion de temps) ou pour comprendre. La modélisation commence par définir des objectifs et de la frontière du modèle. Selon le type d'étude, les phénomènes de bases seront abordés différemment. Par exemple, la modélisation d'un brûleur d'une chaudière pour une étude fine de flamme afin d'améliorer la qualité de la combustion sera différente de celle utilisée pour un calcul de rendement saisonnier d'une chaudière dans la détermination d'une stratégie optimale de régulation de chauffage. On doit aussi évaluer les ressources : « ce dont on dispose et ce qu'on l'on donne ».

En général la modélisation peut être analytique (modèle de connaissance) faisant intervenir des lois de la physique ou empirique (modèle de comportement) basé sur le concept de la boite noire issu de la systémique. La figure 1.2 donne une vision d'ensemble de la modélisation dont les termes sont expliqués par la suite.

- Le modèle de connaissance (modèle théorique)

  Les phénomènes physiques (transfert de chaleur et de masse, conservation d'énergie, température, gradient de pression, etc.) intervenant dans le système sont analysés de manière rigoureuse. Les modèles sont souvent détaillés et très lourds à mettre en œuvre.

Ils sont les plus génériques et donc intéressants dans des études de conception et d'optimisation. Le développement de ces modèles est limité par le domaine d'application des théories.

- Le modèle de comportement (modèle empirique)

On fait une analyse globale du comportement entrée-sortie du système. On n'a pas besoin des informations sur les interactions des divers composants dans le modèle, des dimensions physiques, des phénomènes physiques déterminent le comportement du système. Le modèle peut être perçu comme une boite noire. Des méthodes statistiques ou issues de l'automatique (régressions, variance, corrélations, identification …) ou le bon sens, sont utilisées pour sélectionner les données d'entrées représentatives du problème à étudier. Le modèle peut être utilisé dans le domaine où des données expérimentales existent. Ces modèles sont peu génériques et sont utilisés pour caractériser, commander ou simuler un système. Il faut déterminer les paramètres pour chaque problème considéré.

- Le modèle adapté

Il est évident qu'on peut retrouver des modèles empiriques avec un complément de lois physiques caractérisant quelques relations entre les entrées et les sorties. De même, les modèles basés sur les lois physiques font souvent appel à des techniques statistiques pour déterminer quelques paramètres. Ou encore les modèles à la physique simplifiée et des corrélations ; ils sont communément appelés des modèles adaptés [Dubois 91]. Ils sont les plus fréquemment utilisés car il existe des méthodes algorithmiques et des environnements logiciels qui permettent de passer directement au niveau simulateur.

1.3.1. Les niveaux d'abstraction dans l'élaboration du modèle

Etant donné que chacun peut voir sa propre image d'une situation réelle et donc sa façon de l'exprimer, il est donc indispensable de définir une démarche commune pour pouvoir partager des informations. La démarche peut se décomposer en six paliers d'abstractions [Garnier 95]. Ils sont : le palier technique, le palier physique, le palier mathématique, algorithmique, numérique et informatique.

- Le palier technique et le plus proche de la réalité. On définit le système comme un objet technologique ou comme de la matière qui est plus ou moins autonome. un système n'est pas isolé de son environnement. Les objectifs de la modélisation servent entre autres à définir les frontières du modèle. Elle doit représenter l'image qui entoure le système à l'étude.

- Au palier physique, on étudie les phénomènes physiques impliqués dans les éléments technologiques (bilan thermique, convection, gradient de pression, etc.) On établit ici les variables. La définition du problème et les objectifs collaborent à la distinction des différentes natures des variables (entrées, sorties, paramètres) et les conditions aux limites.

➢ Ayant identifié les phénomènes physiques, il faut les exprimés en termes mathématique en utilisant des lois de physique connues ou en faisant des corrélations entre les paramètres, les variables d'entrées et de sorties.

➢ Le palier algorithmique sert à traduire les expressions mathématiques en des algorithmes avant de procéder à la résolution. Les algorithmes définissent les méthodes numériques à utiliser pour la résolution.

➢ Le choix des outils de résolution tels des solveurs ou l'écriture d'un programme, est établi au niveau numérique suivant la complexité de l'algorithme.

➢ Finalement, vu l'importance croissant de l'informatique, il ne faut pas négliger l'influence du matériel informatique disponible. La puissance de calcul nécessaire pour la résolution de certains algorithmes dépend beaucoup de l'environnement informatique utilisé.

1.3.2. Le modèle et le temps

Lors de la simulation, le comportement d'un système est souvent étudié en fonction du temps, ou sinon l'activité du système est analysée sur une période donnée. Le développement d'un modèle tient donc compte de ce facteur temporel.

Donc, si on peut classer les modèles en fonction de la dépendance du temps, on en dénombre trois catégories :

➢ Modèles statiques : les calculs à un pas de temps donné ne dépendent pas des valeurs des entrées à des pas de temps antérieurs. Par contre les entrées peuvent changer d'un pas de temps à un autre.

➢ Modèles quasi-statiques : le modèle est essentiellement statique mais la valeur d'une variable du pas de temps précédent est utilisée comme point de départ. Ce type de modèle est utilisé pour introduire des termes glissants ou encore un régime harmonique.

➢ Modèles dynamiques : on décrit la dépendance du temps du phénomène modélisé. Les variables à un pas de temps donné vont inclure des valeurs des pas de temps précédents.

1.4. La simulation

Elle se définit comme l'utilisation ou la résolution de modèles correspondant à un système donné pour étudier le comportement de ce dernier dans un contexte précis. Elle est la suite logique de la modélisation et on la retrouve d'ailleurs dans la description de la démarche de modélisation de [Gicquel 92] et de [Soubra 92].

La modélisation et la simulation interviennent pour :

- la compréhension de la structure et d'interactions à l'intérieur d'un système (déterminer le rendement, la performance...),

- l'étude du comportement du système par rapport à son environnement extérieur (consommation énergétique/coût …),

- la prédiction du comportement d'un système pour des situations nouvelles ou extrêmes,

- la conception de nouveaux dispositifs/composants. Etude de système (composant) avant la création de prototype et mise en œuvre de procédés nouveaux (stratégies et algorithmes de contrôle),

- l'optimisation des solutions lors de la conception.

La simulation n'écarte pas l'expérimentation mais la complète. La simulation permet d'effectuer l'analyse de problèmes dans des conditions réalistes, reproduire des tests

que l'on fait en expérimentation pour mieux les comprendre et à moindre coût ou au contraire dans des conditions d'essais extrêmes/marginales climats extrêmes, défauts d'installations… Ces dernières ne peuvent être faites par l'expérimentation (raison de coûts …). Ainsi, la simulation accroît le champ des tests pour un système.

A travers la simulation, le système étudié devient plus flexible. On peut facilement faire des études paramétriques. L'utilisateur peut aussi faire varier l'échelle de temps pour une étude, ce qui est possible par ailleurs.

La simulation se présente presque toujours sous forme d'un programme ou d'outils informatiques. Ces derniers sont couramment appelés des environnements de simulation.

1.5. Les outils de simulation

L'activité de modélisation/simulation avec un support informatique en physique du bâtiment existe depuis plus d'une trentaine d'années. Elle résulte en une prolifération d'outils de simulation. Ces outils sont développés par des universités, centres de recherches, bureaux d'études, industriels pour usage interne ou pour être commercialisés.

Une classification des outils ne sera pas nécessaire dans ce travail. Il existe un répertoire d'outils de simulation destinés à l'énergétique du bâtiment sur Internet [Web 6]. Le dénominateur commun des outils listés est leur contribution pour l'utilisation rationnelle de l'énergie ou l'intégration des concepts d'énergie renouvelables dans le bâtiment. Des descriptions d'outils de simulations et quelques comparaisons ou classifications sont effectués dans [Ebert 93], [Garnier 95], [Keilholz 96] et [Crawley et al. 2001].

On analyse plutôt ce que fera une personne face à une application donnée pour exploiter ces modèles sur un support informatique. Si l'on reprend les niveaux d'abstraction de

[Garnier 95], cela correspond au passage de niveaux 4 (algorithmique) aux paliers 5 et 6 (numérique et informatique) pour un modélisateur.

On peut distinguer quatre cas différents :

CAS 1 : *Elaborer un outil entièrement*

La personne aura recours à des langages de programmation de base tels que :

JAVA, C++, FORTRAN, LISP, etc.… pour :

- décrire ses modèles spécifiques,
- développer un solveur pour résoudre ses systèmes d'équations,
- ajouter, éventuellement, une interface graphique afin de faciliter l'utilisation des outils pour des pré ou post traitements.

Au début de l'utilisation du support informatique, le seul choix possible était d'élaborer l'outil ex-nihilo (développement de TRNSYS, TAS, IDA, ESP, HVACSIM+ …). C'est une tache très lourde est coûteuse qui demande des compétences en tant que thermicien/physicien, numéricien, informaticien voire même de cognition pour le développement d'interfaces utilisateurs ergonomiques. De nos jours, avec les panoplies d'outils disponibles, cette option ne présente guère d'intérêt surtout pour le praticien. Elle existe toujours dans le milieu de la recherche pour l'étude de nouvelles techniques. La fusion de deux outils BLAST et DOE-2 pour le développement d'une nouvelle génération d'outils, ENERGYPLUS est un exemple de cette démarche actuellement. Cette tache demandera le recodage quasiment complet.

L'approche Orienté-Objets dans le développement des outils de simulation a ouvert de nouvelles perspectives dans la modélisation des installations de chauffage et d'air conditionné dans le contexte bâtiment. Cette nouvelle méthodologie permet aux développeurs de maximiser l'étendue de la réutilisation du code et la facilité de la maintenance du système modélisé [Jacobson et al. 92].

Quelques logiciels de simulation utilisent cette approche technique permettant en conséquence de créer une taxonomie d'objets qui représente le concept de classes. Une classe est un type de données abstrait, caractérisé par des propriétés (attributs et méthodes) communes à toutes une famille d'objets et permettant de créer (instancier) des objets possédants ces propriétés et fournie une représentation informatique d'une entité particulière. D'autres concepts importants qui établissent des relations entre classes sont l'encapsulation, l'agrégation et l'héritage.

Le mécanisme de l'héritage donne un moyen à étendre les propriétés d'une classe en définissant des classes de rangs supérieurs dans la hiérarchie héritent des propriétés (attributs et méthodes) des classes de rangs inférieurs dans la hiérarchie. Les classes de

rangs inférieurs peuvent représenter un niveau plus général et plus abstrait que celles dans le rang supérieur. Donc, une représentation spécialisée de classes peut être marquée.

Des outils de simulation tels qu'ESP-r [Tang et Clark, 93], et ZEBRA [Marshallsay et al, 97] développés à la base de la méthodologie Orienté Objets. L'environnement EKS [Clark et Randal, 93] offre des classes utilitaires telles que des classes solveurs, des classes intrinsèque (qui annexent des données climatiques, des propriétés des matériaux…). Des métas classes et classes templates pour la création et la manipulation. Le mécanisme de l'héritage est clairement illustré dans l'élaboration de la taxonomie de classes dans l'environnement EKS permettant donc, la réutilisation du code même entre classes différentes. La plus part des outils de simulation cités précédemment sont écrient à la base du langage C++ ; par contre EnergyPlus [Crawley et al. 99], le FORTRAN 90 est le langage de base sur lequel le noyau du programme informatique a été construit.

EnergyPlus qui tient ses origines de BLAST et DOE-2, sa construction et sa structure sont misent à la disposition des développeurs pour d'éventuelles améliorations.

Le programme est conçu en premier lieu, en tant que moteur de simulation, sans interfaces graphiques officielles, possède une structure modulaire qui permet de faciliter l'intégration d'autres modules et facilite le lien avec d'autres programmes permettant de ce fait une configuration de données de type C++ [Fischer et al. 99].

CAS 2 : *Elaborer un outil couplé à un solveur existant*

Ici on s'affranchit du développement et codage des méthodes numériques en utilisant des solveurs génériques. La démarche est modulaire comme on sépare la partie description des modèles de solveurs numériques. Ensuite il y aura la partie interface utilisateur à mettre en œuvre. Les premiers solveurs faisant leur apparition dans le domaine du bâtiment sont NEPTUNIX, ASTEC, ESACAP. Les deux premiers utilisent une analogie thermique-électrique.

Cette démarche a été adoptée lors du développement de l'environnement ALLAN par GDF en utilisant un solveur Neptunix de CISI ingénierie ou ASTEC 4 et pour l'environnement CLIM2000 de EDF qui est lié au solveur ESACAP. Ces deux outils offre un environnement graphique à l'utilisateur. Actuellement, il y a aussi les solveurs fournis dans des outils mathématiques généraux qui sont suffisamment puissants pour le développement d'outils de simulation. On peut citer : MATLAB, MATRIXx, MAPLE, MACSYMA, MATHEMATICA. On peut ajouter EES (Engineering Equation Solver) développé par l'université de Wisconsin à Madison, qui possède une bibliothèque d'outils en thermodynamique. Ce solveur est le moteur de calcul de TRNSYS.

CAS 3 : Avoir recours à un environnement de simulation

Dans ce cas l'utilisateur se préoccupe uniquement de développer ses modèles spécifiques. L'environnement de simulation et le solveur sont à la disposition. ALLAN/Neptunix et CLIM2000 en sont des exemples et ont l'avantage d'être orientés vers les besoins en thermique vu leur origine.

On peut aussi avoir recours à des environnements de simulation commerciaux généraux qui sont devenus commun auprès de la communauté scientifique et des praticiens depuis les années 90 dans des domaines essentiellement liés à l'automatisme (GPSS, MatLab/SimuLink, ModSim/SimPress, etc.). L'utilisateur :

- à accès à des bibliothèques d'utilitaires mathématiques, logique …,

- utilise des langages de programmation évolués qui sont plus faciles à mettre en œuvre.

- et dispose aussi d'un environnement de programmation graphique

CAS 4 : *Avoir recours à un environnement spécifique*

Dans ce cas, on doit faire la distinction entre la personne qui ne souhaite qu'utiliser un environnement de simulation de celui qui sera amené à modifier l'environnement de simulation. Le deuxième devra choisir un environnement de simulation dit « ouvert » ou il pourra faire ses développements spécifiques.

Si l'utilisateur se contente d'utiliser un environnement de simulation,

- Il utilise les modèles disponibles dans l'outil. En thermique du bâtiment il aura un choix énorme d'outils, les plus connus étant TRNSYS, TAS, HVACSIM+, ESP-r …

- Dans l'étude de système de génie climatique, l'utilisateur aura à assembler son système à partir de composants d'une bibliothèque s'il choisit un environnement modulaire (TRNSYS, TAS, ESP-r, HVACSIM+ …). Des interfaces graphiques (IISIBAT pour TRNSYS) faciliteront la tache.

- D'autres environnements (DOE-2 par exemple) proposent une bibliothèque de systèmes de génie climatique les plus courants et l'utilisateur aura à les paramétrer pour se rapprocher de sa configuration d'étude.

Si, par contre, l'utilisateur veut avoir la possibilité de modification,

- Il cherche à augmenter les ressources du système qui existe. Il le fera essentiellement en enrichissant la bibliothèque de modèles mais il pourra aussi compléter les méthodes numériques ou améliorer l'interface utilisateur. Le logiciel CSTBât [Laret 89] est basé sur TRNSYS et contient une bibliothèque de modèles adaptés pour les études des réseaux hydrauliques en chauffage. Il a fallu aussi élaborer une interface pour établir une connexion avec les catalogues de données de composants.

- Un autre exemple est l'outil de simulation RIUSKA, interface pour DOE-2 et ayant une passerelle avec un logiciel de dessin SMOG basé sur AUTOCAD.

- Les logiciels tels que HVACSIM+ permettent l'addition de modèles dans la librairie de modèles de composants.

Les quatre cas expliqués ci-dessus montrent qu'un utilisateur a plusieurs alternatives pour passer de la modélisation à l'application informatique pour la simulation. Tout dépendra :

- des besoins de l'utilisateur,
- des compétences de l'utilisateur,
- des moyens à la disposition de l'utilisateur.

Il faut préciser que la majorité des options citées concerne les utilisateurs qui sont experts ou des développeurs.

Les praticiens s'intéresseront principalement à la première possibilité de l'option 4. Ceux qui ont une activité de recherche et développement sont très souvent impliqués dans la deuxième possibilité. Quelque soit l'option choisie, on doit se préoccuper d'une phase de création d'interface graphique ou d'utilisation d'environnement graphique.

1.6. La programmation graphique

La programmation graphique ou visuelle fait référence à tout système permettant d'effectuer des actions de communication avec le monde en utilisant un ensemble d'agencements spatiales de symboles textuels ou graphiques ayant une interprétation sémantique. De par ce caractère sémantique elle est un outil de la systémique [Durand 98]. Un langage visuel est un langage qui permet de manipuler des informations visuelles, qui permet de programmer à l'aide d'expressions visuelles. Les langages de programmation visuelle peuvent être classés en fonction du type et du degré d'expression visuelles utilisés en :

- langages à base d'icônes,
- langages à base de formulaires,
- langages à base de diagrammes (graphes orientés, schéma-blocs …).

Quand on travaille dans un environnement graphique, on essaye de respecter les quelques points suivants qui démarquent de la programmation codée ou textuelle afin que le graphisme apporte un plus par rapport au texte :

- *Ressemblance entre la visualisation à l'écran et le monde réel*

  Pour cela, il faut définir ce qu'on appelle le monde réel. Qu'est ce qu'on est entrain de modéliser ? Pour modéliser, il faut connaître les buts du modèle et les frontières qui le délimitent de l'environnement. La réponse doit être claire avant qu'on ne puisse procéder à une implémentation graphique.

- *Vue d'ensemble de la structure du modèle* :

  Cette caractéristique s'applique dans un premier temps au modélisateur. Les langages visuels tout comme les langages textuels sont des structures d'informations et c'est au modélisateur (programmeur) de décider quelles sont les informations à transmettre pour que la description soit pertinente.

- *Facilité de lecture localement :*

  Dans un langage graphique, la partie strictement syntaxique est réduite. On transmet la majorité des informations de manière sémantique. La représentation par icônes peut être plus facile à distinguer que des noms textuels ou des expressions symboliques.

Dans le cas du chauffage, par exemple, le bloc robinet thermostatique se situera au niveau de la zone au voisinage du bloc représentant le radiateur. Néanmoins, la programmation graphique présente des faiblesses dont voici quelques unes :

- Le superlativisme (le superlativisme est une idée qui stipule que la programmation visuelle est mieux adaptée que la programmation textuelle pour les novices ou débutants) de la programmation graphique se vérifie quand les structures sont simples et courtes [Green 91].

- On ne peut maintenir la totalité des superlativismes dès que le système (donc le programme) devient plus complexe, ou encore pour des taches de déprogrammation ou l'utilisateur doit faire ressortir des objectifs de niveaux d'abstraction élevée et des plans du programme pour le comprendre et le modifier.

1.7. La modélisation UML

La description de la programmation par objets (ou programmation orientée-objets) a fait ressortir l'étendue du travail conceptuel nécessaire : définition des classes, de leurs relations, des attributs et méthodes, des interfaces etc. Pour programmer une application, il ne convient pas de se lancer tête baissée dans l'écriture du code : il faut d'abord organiser ses idées, les documenter, puis organiser la réalisation en définissant les modules et étapes de la réalisation. C'est cette démarche antérieure à l'écriture que l'on appelle modélisation ; son produit est un modèle.

Les spécifications fournies par la maîtrise d'ouvrage en programmation impérative étaient souvent floues : les articulations conceptuelles (structures de données, algorithmes de traitement) s'exprimant dans le vocabulaire de l'informatique, le modèle devait souvent être élaboré par celle-ci.

L'approche objet permet en principe à la maîtrise d'ouvrage de s'exprimer de façon précise selon un vocabulaire qui, tout en transcrivant les besoins du métier, pourra être immédiatement compris par les développeurs et parfois même les utilisateurs.

UML n'est pas une méthode (i.e. une description normative des étapes de la modélisation) : ses auteurs ont en effet estimé qu'il n'était pas opportun de définir une méthode en raison de la diversité des cas particuliers. Ils ont préféré se borner à définir un langage graphique qui permet de représenter, de communiquer les divers aspects d'un système d'information

(aux graphiques sont bien sûr associés des textes qui expliquent leur contenu) [Rumbaugh et al. 99].
UML est donc un métalangage car il fournit les éléments permettant de construire le modèle qui lui, sera le langage du projet. Il est impossible de donner une représentation graphique complète d'un logiciel, ou de tout autre système complexe, de même qu'il est impossible de représenter entièrement une statue (à trois dimensions) par des photographies (à deux dimensions). Mais il est possible de donner sur un tel système des vues partielles, analogues chacune à une photographie d'une statue, et dont la juxtaposition donnera une idée utilisable en pratique sans risque d'erreur grave.

Plus d'amples détails des spécificités et des mécanismes de cette modélisation graphique seront détaillés dans le chapitre III.

1.8. **CONCLUSION**

Des concepts qui sont nécessaires pour s'atteler à la tache de la modélisation et de la simulation ont été présentés dans ce chapitre. Les points les plus importants qui seront mis en œuvre dans la suite de ce travail sont :

- La description de l'approche de la programmation orientée-objets et la discussion des fondements de la modélisation dans le contexte du bâtiment et des installations d'air conditionné à la base de la notation UML, pourrait améliorer la communication et l'échange de l'information entre développeurs indépendamment des langages de programmation. Les besoins de l'utilisateur seront mieux cernés, le temps de l'élaboration de l'interface usager est réduit considérablement, et la maintenance du code, l'amélioration du programme intervient au niveau du modèle sans que le code soit largement modifié.

- La génération de prototype d'interface usager (IU) pour le fichier IDD (Input Data Definition) du programme EnergyPlus, est mise en évidence par l'emploi de l'approche scenario, la modélisation UML et la description de l'aspect dynamique et statique de l'interface usager par les réseaux colorés de Pétri(RdPs).

- Comme cadre méthodologique utilisant les techniques formelles et de prototypage, nous avons proposé un processus itératif (Figure 1.3) pour le développement du prototype de l'IU. Ce processus est le résultat de la fusion de deux processus itératifs de développement : un processus utilisant les techniques formelles (cycle supérieur de la figure 1.3) et un processus utilisant les techniques de prototypage (cycle inférieur de la figure 1.3).

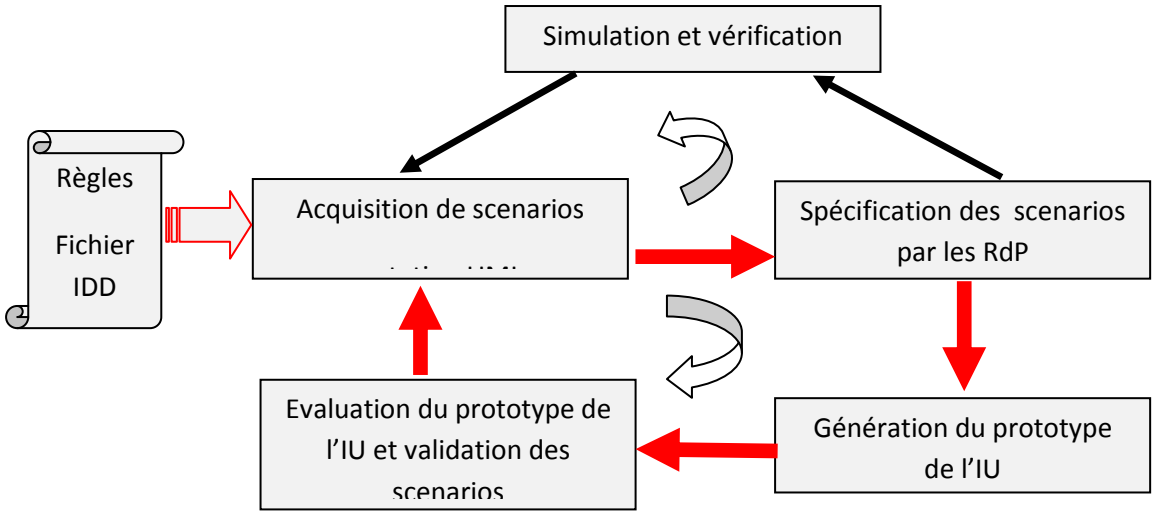


Figure 1.3. Processus combinant les techniques formelles et les techniques de prototypage

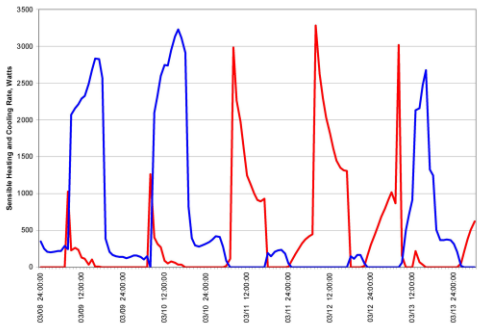
Jusqu'à date, les résultats de nos travaux ont été publiés dans :

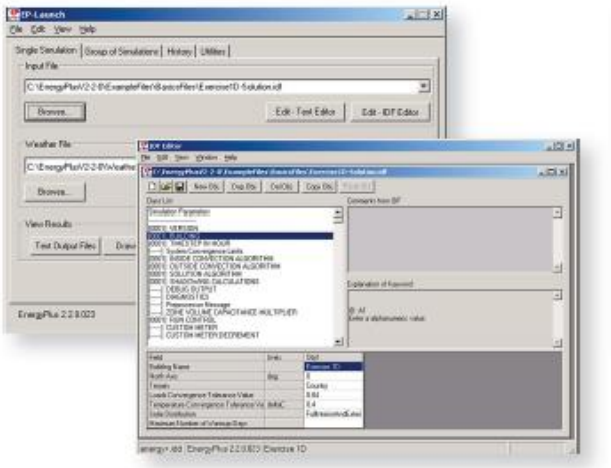
- [Bachkhaznadji et al. 2006] : 3rd International Congress of Climamed, Lyon France,
- [Bachkhaznadji et al. 2012]: Publication dans la Revue Energy Procedia, éditeur : Elsevier.
- [Bachkhaznadji et al. 2014] : Publication dans la Revue Nationale Sciences & Technologie B, éditeur : Université Constantine 1.

Chapitre 2

EnergyPlus

Présentation du Programme





Un nouveau programme de simulation des performances énergétiques du bâtiment est développé en vertu de l'appui du gouvernement des États-Unis et publié en avril 2001. EnergyPlus est basée sur les meilleures caractéristiques et capacités que possèdent ses prédécesseurs programmes : BLAST et DOE-2. EnergyPlus est un nouveau programme complètement écrit en Fortran 90, et comprend de nombreuses fonctionnalités innovantes de simulation : pas de temps variables configurables par l'utilisateur, systèmes modulaires, bilan thermique à base de flux d'air multizone, des polluants atmosphériques, le transfert d'humidité dans les composants de la construction, systèmes photovoltaïque etc. Les structures de données d'entrée et de sortie de la simulation sont adaptées pour faciliter l'intégration de nouveaux modules. Le développement d'interfaces usagers est laissée pour une tierce partie. EnergyPlus est avant tout un moteur de simulation, sans interfaces utilisateur - bien que celles-ci sont en cours de développement par le secteur privé. Le présent chapitre se concentre sur la description générale des méthodes et des capacités d'EnergyPlus.

2.1. **Introduction**

Depuis plus de vingt ans, le gouvernement des États-Unis a soutenu le développement de deux programmes de simulation des performances énergétiques des bâtiments, DOE-2 et BLAST. BLAST (Building System Laboratory 1999), parrainé par le Département américain de la Défense (DOD), a ses origines dans le NBSLD programme développé par le bureau national des standards des États-Unis (désormais NIST) dans les années 1970. DOE-2 [Winkelmann et al. 1993], parrainé par le Département américain de l'Énergie (DOE), a ses origines dans le programme Post Office écrit dans les années 1960 pour le bureau de poste des États-Unis. La principale différence entre le deux programmes est la méthode de calcul des charges, DOE-2 utilise l'approche *room weighting factor (Fonction de Transfert)*, par contre BLAST utilise l'approche par *bilan de chaleur*.

Les deux programmes sont largement utilisés partout dans le monde. Chaque programme comprend des centaines de routines qui travaillent ensemble pour simuler les transferts d'énergie thermique et de masse à travers un bâtiment. Dans certains cas, les sous-programmes dans DOE-2 étaient plus précis. En d'autres cas, les sous-programmes dans BLAST étaient plus précis.

Dans les deux cas, les méthodes de simulation sont souvent difficiles à retracer en raison de décennies de développement par plusieurs auteurs. BLAST et DOE-2 ont été rédigés dans des versions anciennes de Fortran et utilisent des fonctionnalités qui sont devenus obsolète dans les nouveaux compilateurs. Les deux programmes se composait d'une quantité importante de 'code en spaghetti' et leur structure est dépassées, ce qui les rendaient difficile à maintenir, à soutenir et améliorer.

Beaucoup de gens se sont demandé pourquoi le gouvernement des États-Unis supportait les deux programmes séparément avec des capacités comparables. Les discussions sur la fusion des deux programmes ont véritablement débuté en 1994. Le département Américain de l'énergie (DOE) a finalement eu l'initiative de commencer à élaborer un nouveau programme nommé EnergyPlus en 1996. L'équipe EnergyPlus comprenait : Laboratoires de recherche du génie de l'armée américaine(U. S. Army Construction Engineering Research Laboratories, CERL) , Université de l'Illinois (UI) , Laboratoire National Laurent Berkeley (LBNL) , Université de l'état d'Oklahoma (OSU) , GARD Analytics , et le DOE .

Dans ce qui suit, nous présenterons un aperçu de l'organisation et des capacités du programme EnergyPlus et expliquerons la structure derrière l'ensemble du programme.

2.1. Qu'est-ce que c'est EnergyPlus ?

EnergyPlus est un tout nouveau programme mais basé sur les caractéristiques et les capacités les plus populaires de BLAST et DOE-2. L'objectif majeur qui a été fixé pour le développement d'EnergyPlus était de créer un système bien organisé, avec une structure modulaire pour faciliter l'ajout de fonctionnalités et de liens à d'autres programmes. Malgré les avantages de la structure et l'orientation objet du langage C / C + +, l'équipe a décidé d'utiliser le Fortran 90 comme langage de programmation pour EnergyPlus. Ainsi, EnergyPlus comprend un nouveau code, modulaire, structuré et écrit en Fortran 90.

L'équipe de développement a concentré son effort dans l'élaboration d'un moteur de simulation sans interface utilisateur particulière. Les données d'entrée et de sortie (I/O) sont simples et séparées par des virgules, sous formes de fichiers texte ASCII, les données sont beaucoup plus simples que ceux dans DOE-2 ou BLAST.

Les deux programmes BLAST et DOE-2 ont réussi à attirer un grand nombre de tiers développeurs pour créer des interfaces utilisateur et de nouveaux modules. Au cours des tests bêta du programme EnergyPlus, l'équipe avait commencé à travailler avec ces mêmes développeurs pour s'assurer que des interfaces utilisateurs seront disponibles peu de temps après EnergyPlus ait été publié.

2.2. Structure modulaire du code

Un des principaux objectifs pour EnergyPlus est de créer un système bien organisé, la structure modulaire du code facilite l'ajout de fonctionnalités et de liens vers d'autres programmes. L'équipe de développement a décidé de sélectionner Fortran 90 comme langage de programmation pour programme EnergyPlus pour les raisons suivantes:

- Est un langage moderne et modulaire avec de bons compilateurs sur plusieurs plates-formes.
- Permet des structures de données semblables à celles qu'on trouve dans le langage C/C++.
- Fournit une structure qui commence à être orientée objets.
- Permet de longs noms de variables (jusqu'à 100 caractères).
- Offre une compatibilité descendante au cours du processus de développement.

L'équipe de développement a commencé à travailler sur EnergyPlus en modularisant (restructuration) le code du moteur qui calcule le bilan thermique dans IBlast, une version de recherche de BLAST avec des charges intégrées et calcul des systèmes de chauffage, ventilation et conditionnement d'air(CVC) [Taylor et al .1990, 1991]. Normalement une telle restructuration aurait entraîné une réécriture très lourde du code et impliquant un développement de longue période, et des tests approfondis pour s'assurer que le nouveau code s'exécute comme prévu.

Pour éviter ce problème l'équipe a conçu une démarche de développement qui l'avait appelée la Réingénierie Evolutive (RE) qui progressivement déplace le programme à

partir de l'ancien code non structurées vers un nouveau code à structure modulaire et en intégrant le nouveau et l'ancien code. Tout au long de ce processus, le code existant fonctionne toujours avec les données d'entrée, et est étendu pour générer les paramètres nécessaires par le nouveau code sous formes de modules. Ainsi, les nouveaux modules peuvent être vérifiés sans avoir à remplacer complètement la capacité fonctionnelle de l'ancien programme avec le nouveau code avant de pouvoir être testé.

Tout au long du processus du renouvellement du code, les vieilles routines sont remplacées par de nouvelles routines avec une restructuration des données. Cette action a permis une transition évolutive et douce tout en augmentant la capacité pour des essais de vérification. Au moment où la version Beta du programme a été atteinte, tous l'ancien code avait été remplacé par un nouveau code totalement écrit en Fortran 90, avec une structure modulaire.

2.3. *Structure du programme EnergyPlus*

Dans les ateliers du DOE/DOD [Crawley et al. 1997] il y avait un fort consensus pour un outil souple et robuste avec des fonctionnalités supplémentaires qui s'avère nécessaires. Les thèmes récurrents pour les besoins de simulation dans ces ateliers étaient la conception, l'environnement, l'économie, le confort des occupants et la sécurité.

Généralement, Les concepteurs ont besoin d'outils qui apportent des réponses à des questions très spécifiques au cours de la conception. Ils veulent des outils qui offrent le plus haut niveau de précision dans la simulation. Une des priorités était donc une approche pour une simulation intégrée (simultanée) pour la détermination précise de la température et une bonne prédiction du confort.

En réponse à ces observations, l'équipe de développement a décidé que la simulation intégrée devrait être le concept sous-jacent pour EnergyPlus – les charges calculées à la base du pas de temps (15 minute par défaut) spécifiée par l'utilisateur sont passés au module des systèmes de bâtiment pour une simulation au même pas de temps pour dimensionner les systèmes de chauffage, de refroidissement, électrique, etc.

En adoptant la technique de la solution intégrée pour le programme EnergyPlus, les simulations séquentielles ont été résolues. Les prévisions inexactes de la température des espaces occupés sont dues essentiellement à une mauvaise interaction entre les modules de simulation des systèmes CVC et le calcul des charges thermiques.

La prédiction exacte des températures des espaces est cruciale pour une conception énergétique efficace. Ainsi le dimensionnement des systèmes et des équipements, le confort thermique des occupants, tous dépendent de la prédiction exacte de la température de l'espace occupé.

La Simulation intégrée permet également aux utilisateurs de simuler et d'évaluer un certain nombre de processus dont on cite certains des plus importants :

- Une régulation réaliste du système.
- Adsorption et désorption de l'humidité des éléments d'une construction.
- Le chauffage par rayonnement et les systèmes de refroidissement.
- Débits d'air interzones.

Comme le montre la figure 2.1, EnergyPlus se compose essentiellement de trois types de composants : - un gestionnaire de simulation, - un Module de simulation du bilan thermique

et de masse (sur la base du programme IBlast), - et un nouveau Module de simulation des systèmes du bâtiment. Le Gestionnaire de simulation contrôle l'ensemble du processus de simulation.

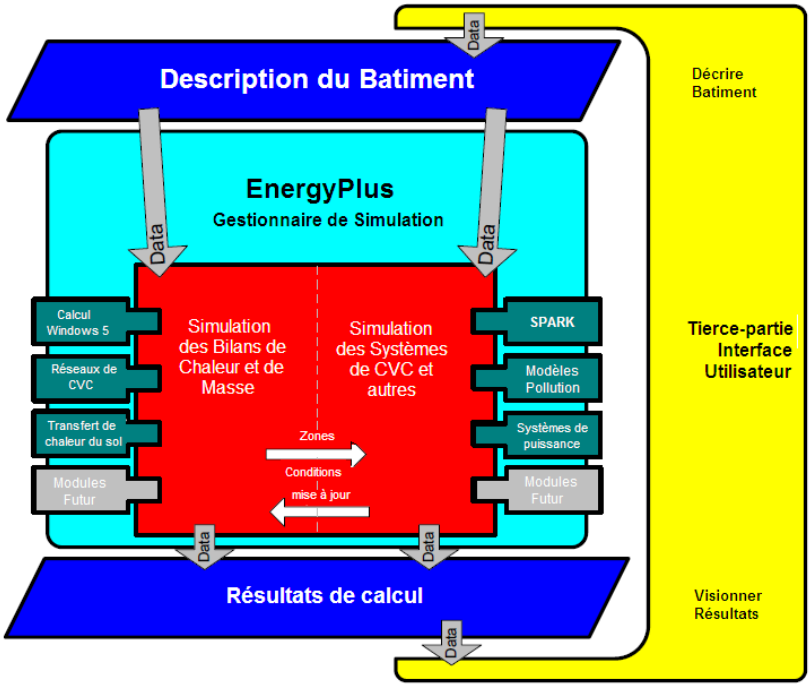


Figure 2.1 La structure globale du programme EnergyPlus

Le gestionnaire de simulation des systèmes du bâtiment gère la communication entre le moteur de calcul du bilan thermique et de masse, l'eau des systèmes CVC et les systèmes d'air bouclés et leurs composants (les batteries, les chaudières, les refroidisseurs, les pompes, ventilateurs, etc.). Le gestionnaire de simulation a était crée spécifiquement pour résoudre certains problèmes hérités du code en spaghetti et le manque de structure dans DOE-2 et BLAST. Le gestionnaire de simulation offre plusieurs avantages dont on citera l'essentiel :

- La simulation des principaux systèmes CVC en boucles sont contenus dans un seul module.
- Les modules sont autonomes et ont une orientation objets.
- L'accès aux données est commandé.
- De nouveaux modules peuvent être facilement ajoutés.

2.4. Bilan de masse et de chaleur

La méthode de calcul des charges thermiques dans le contexte zonage dans un bâtiment adoptée par les développeurs du programme EnergyPlus est la méthode du Bilan de Masse et de Chaleur. L'hypothèse fondamentale des modèles des équilibres thermiques est que l'air dans chaque zone thermique peut être modélisé, avec un mélange continu permettant d'obtenir ainsi une température uniforme. Bien que cela ne reflète pas la réalité physique, la seule alternative actuelle est d'utiliser la méthode CFD (Computational Fluid Dynamics) - un procédé de calcul complexe et de simulation intensive du fluide (dans ce cas, l'air) en circulation. Actuellement, la méthode CFD est la plus utile dans les recherches appliquées. Plusieurs groupes travaillent dans l'élaboration de modèles quelque part entre le modèle d'agitations continues et un calcul CFD intensif.

La structure modulaire du programme EnergyPlus permet à ces nouveaux modèles d'être inclus à l'avenir dans des versions postérieures dès qu'ils seront disponibles. L'autre grande hypothèse dans les modèles à bilan de masse et de chaleur, c'est que les surfaces de la pièce (murs, fenêtres, plafonds et planchers) ont :

- des températures uniformes.
- une Irradiation uniforme à longue et à courte onde.
- des surfaces rayonnantes diffuses.
- conduction thermique à une dimension.

La figure 2.2 montre la structure du programme EnergyPlus avec un gestionnaire de solution intégrée qui gère des modules et qui calculent les bilans thermiques des surfaces et agit comme une interface entre le bilan thermique et le gestionnaire de simulation des systèmes CVC associés au bâtiment.

Le Module du bilan thermique de surface simule le bilan thermique de surface à l'intérieur et à l'extérieur, établi les interconnexions entre ces bilans thermiques et les conditions aux limites de la conduction, la convection, le rayonnement, et les effets du transfert de masse de la vapeur d'eau. Le module du bilan massique de l'air traite aussi des divers flux de masse telle que la ventilation, l'extraction, et l'infiltration. Il représente la masse thermique de la zone d'air et évalue les gains directs de chaleur par convection. A travers ce module EnergyPlus est lié à COMIS [Fuestel 1990] pour une meilleure évaluation du mouvement de l'air dans des espaces multizones, de l'infiltration, de contaminants intérieurs, et les calculs de ventilation.

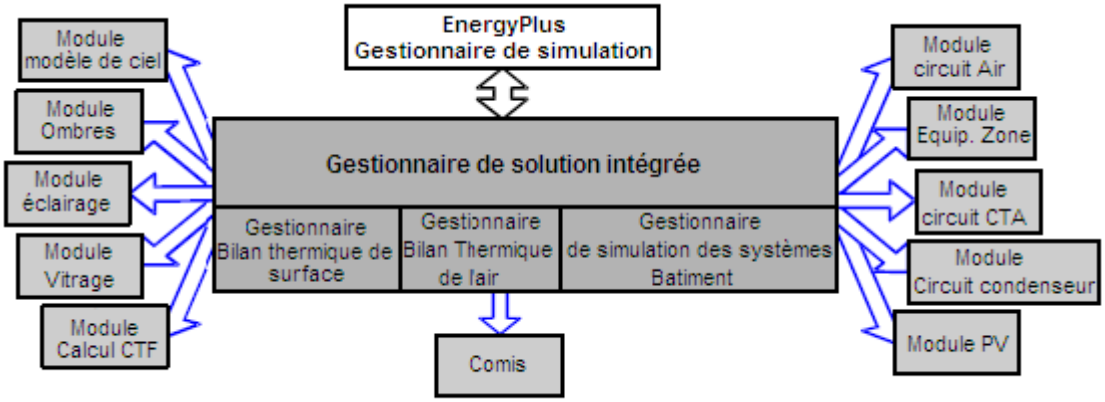


Figure 2.2 Gestionnaire de la simulation intégrée

Trois autres nouveaux modules à la base des capacités de DOE-2 ont vu le jour: le module de simulation de l'éclairage naturel [Winckelmann et Selkowitz 1985], le module Windows-4 à 5 pour les calculs basés sur les ouvertures et fenêtres [Arasteh et al. 1994], et le module du ciel anisotrope. Le module calcule l'éclairage naturel horaire intérieur, l'éblouissement causé par fenêtres, le contrôle de l'éblouissement, le contrôle de l'éclairage électrique, et calcule la réduction de l'éclairage électrique pour le module du bilan thermique et de masse. Les concepteurs peuvent entrer la description de la fenêtre couche par couche ou choisir des fenêtres depuis une bibliothèque (classique, de réflexion, de remplissage à gaz, etc.). L'ensoleillement peut être modélisé avec différents types de protections solaires : stores, rideaux à rouleaux, ou vitrage électro-chrome. Le module du modèle ciel inclut le calcul du rayonnement anisotrope et la distribution de la luminance du ciel basée sur le modèle empirique de Perez [1990, 1991] en fonction de la position du soleil et de la couverture nuageuse. Cette distribution du rayonnement non uniforme améliore le calcul du rayonnement solaire diffus sur les surfaces inclinées (murs et les toits en pente).

Plusieurs autres modules ont été remaniés dans le but de les inclure dans le programme EnergyPlus : La protection solaire de BLAST et les calculs de la fonction de transfert par conduction.

Les principales améliorations apportées au programme IBlast (et au programme EnergyPlus) sur le moteur de simulation du bilan thermique de Blast, comprennent le transfert de masse, le chauffage radiant et le refroidissement. La capacité de transfert de masse au sein du programme EnergyPlus permet la solution couche par couche pour le transfert de masse à travers des surfaces et un bilan de masse au niveau d'une zone d'air similaire au calcul du bilan thermique de l'air. Les modèles de chauffage radiant et de refroidissement sont une extension de la fonction de transfert thermique par conduction et incorporent les calculs de confort thermiques.

Ceci fournit un meilleur moyen pour l'amélioration de la modélisation et les capacités de contrôle pour le nouveau gestionnaire de simulation des systèmes CVC associés au bâtiment.

Un modèle d'humidité simplifiée connue sous le nom de Profondeur Efficace de pénétration d'humidité (Effective Moisture Penetration Depth (EMPD)) a était incorporé pour estimer les interactions entre l'humidité de l'air, les surfaces de l'espace intérieur et les meubles. EMPD est utile pour l'estimation des impacts associés à l'humidité lorsque les détails de la géométrie interne de l'espace zone et / ou la documentation détaillée des propriétés des matériaux utilisés ne sont pas facilement disponibles.

2.5. Gestionnaire de Simulation des systèmes CVC associés au Bâtiment

Après que le gestionnaire du bilan thermique complète la simulation en usant d'un pas de temps approprié (paramétrable par l'utilisateur), il fait appelle au gestionnaire de simulation des systèmes, qui contrôle la simulation des systèmes de CVC, des systèmes électriques, des équipements et des composants et fait les mises à jour des conditions de la zone d'air.

EnergyPlus n'utilise pas une méthode de simulation en ordre séquentiel (charges du bâtiment d'abord, puis les réseaux de distribution, ensuite les CTA) que l'on trouve généralement dans DOE-2 et BLAST, cela imposerait des limites rigides sur les structures du programme et impose aussi des limites à la flexibilité des données d'entrée. Au lieu de cela, EnergyPlus a était façonné pour que son gestionnaire de simulation des systèmes CVC répond aux objectifs suivants :

- Simulation entièrement intégré des charges thermiques, des systèmes CVC, et des CTA
- Modulaire.
- Extensible

La simulation intégrée permet de modéliser les limites des capacités de façon plus réaliste et fournit un couplage plus serré entre le coté air et le côté eau des systèmes CVC et CTA. La modularité est ainsi maintenue à la fois au niveau des composants et des systèmes. Cela facilite l'ajout de nouveaux composants et permet une configuration flexible du système à modéliser et au niveau du système, de l'équipement et les systèmes sont clairement liées aux modèles de zone dans le gestionnaire du bilan thermique. Pour mettre en œuvre ces concepts, la notion de boucles a était utilisée tout au long de la construction du gestionnaire de simulation

des systèmes - principalement les systèmes CVC(les boucles d'air) et les boucles d'eau. Les boucles imitent les réseaux de tuyaux et des conduits dans les bâtiments réels. EnergyPlus permettra plus tard de simuler les pertes de charges et les pertes de chaleur (tuyau / conduit) lors du déplacement du fluide dans chaque boucle.

La boucle d'air simule les réseaux de transport de l'air, ceux de la climatisation, et les mélanges et comprend aussi les ventilateurs des réseaux d'alimentation et de retour,

les réseaux des batteries de chauffage et de refroidissement, etc.

La boucle d'air est connectée à l'espace zone à travers des équipements. Ces équipements inclus les diffuseurs, les batteries de chauffage/refroidissement, la commande du réseau de soufflage (les volets de mélange, les unités d'induction, les volets des systèmes VAV, etc.). Nous citerons aussi d'autres équipements tels que les unités locales convectives (unités d'air conditionné, les ventilo-convecteurs, les pompes à chaleur). Les concepteurs peuvent spécifier plus d'un type d'équipement pour une zone. Cependant, ces équipements doivent être énumérés dans un ordre bien spécifié, pour qu'ils puissent répondre aux besoins en chaleur ou en refroidissement de la zone.

Pour la boucle d'air, le procédé de solution est itératif. Afin de spécifier les connexions des équipements à une boucle, les nœuds sont définis à des endroits clés autour d'une boucle, et chaque nœud possède un identifiant numérique unique, Figure 2.3. Les identificateurs de nœuds sauvegardent les variables de l'état de la boucle et les informations des points de consigne pour un emplacement dans la boucle. Pour résoudre les inconnus des variables d'état dans une boucle, une technique itérative est utilisée avec des représentations d'état de commande.

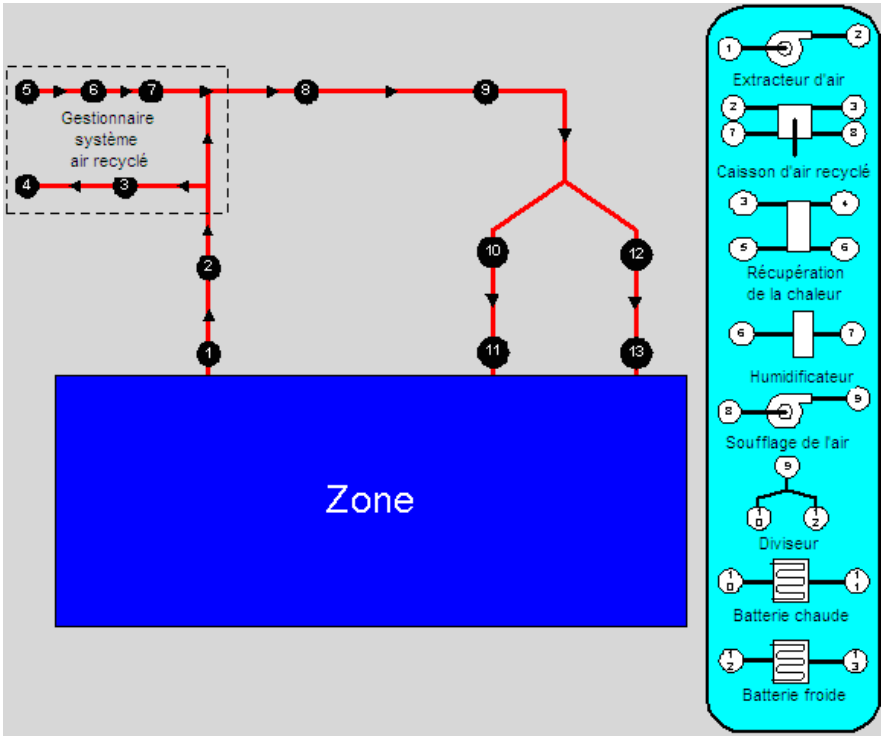


Figure 2.3. Nœuds dans une boucle d'air d'un diagramme simple

Ces représentations connectent les points de consigne au nœud avec la fonction de commande d'un composant, par exemple la position du volet de réglage de débit pour

ventilateur et de débit d'eau de refroidissement. Dans ce schéma, tous les composants de la boucle sont simulés en premier, puis les équations de contrôle sont mises à jour en utilisant des différences finies explicites. Cette procédure se poursuit jusqu'à ce que la simulation converge.

Il existe deux boucles pour des installations de CVC, une boucle primaire (pour les équipements d'alimentation tels que les chaudières, les refroidisseurs, stockage thermique, et les pompes à chaleur) et une boucle secondaire (pour les équipements de rejet de la chaleur comme les tours de refroidissement et les condenseurs)

La figure 2.4 présente une vue schématique des connexions d'équipements sur la boucle primaire des installations CVC. L'équipement est spécifié par type (chaudière à gaz, refroidisseur d'entraînement ouvert centrifuge) et son exploitation caractéristiques.

En raison de la structure modulaire du code avec lequel EnergyPlus a était écrit, il est plus facile pour les développeurs d'ajouter d'autres types de modèles. Comme dans la boucle d'air, les boucles primaires et secondaires utilisent explicitement les nœuds pour connecter les équipements à chaque boucle.

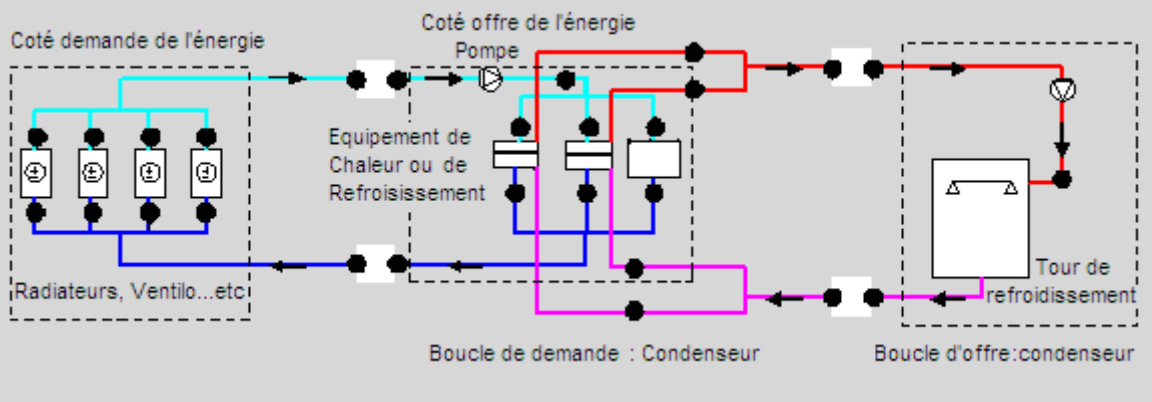


Figure 2.4 Exemple de boucle d'équipement

Les liens entre la boucle d'air et les équipements d'une zone, les boucles primaire et secondaire se font à travers une structure de données des nœuds et qui doit être explicitement défini dans le fichier d'entrée des données. Fisher et al. (1999) fournit des informations supplémentaires sur la structure modulaire, et l'approche à base de boucles dans la simulation des systèmes CVC des constructions du programme EnergyPlus.

Une approche similaire à base de boucles est proposée pour une nouvelle boucle électrique pour simuler les systèmes électriques d'alimentation (modules photovoltaïque, piles à combustible, etc.), la demande (charges de prises, l'éclairage, et d'autres charges électriques) et la mesure (mètres). À long terme, les utilisateurs du programme EnergyPlus auront plus d'options des systèmes et d'équipements à simuler grâce à un lien vers SPARK

[Buhl et al. 1993] - un outil de simulation basé sur les équations. SPARK est un meilleur solveur pour les problèmes complexes itératifs. SPARK a déjà une bibliothèque de composants de CVC basés sur les boîtes à outils primaires et secondaires d'ASHRAE [Lebrun et al .1999; Brandemuehl et al. 1993].

EnergyPlus continuera à se développer en recevant plus de modèles de système (dans les modèles de fichiers d'entrée), mais les développeurs et les utilisateurs avancés pourront créer facilement des nouveaux modèles de CVC complexe en utilisant le lien avec le logiciel SPARK. Un lien similaire est en cours de développement au modèle de simulation TRNSYS (Laboratoire d'énergie solaire 2000), ce qui permettra aux utilisateurs du photovoltaïque, et de la thermique solaire de simuler ces modèles ainsi que d'autres modèles de systèmes CVC et des modèles de CTA.

2.6. Données d'entrée, de sortie, et de météo

Les fichiers d'entrée du programme EnergyPlus sont conçus pour un entretien et une expansion facile. L'équipe de développement a choisie de conserver un fichier d'entrée simple pour accepter d'autres données d'entrée de simulation provenant d'autres sources, telles que les systèmes de CAO par exemple. Un fichier d'entrée du programme EnergyPlus, étant lisible, est cryptique et certainement pas convivial, il n'est pas destiné à être une interface principale pour les utilisateurs typiques. L'équipe de développement s'attende à ce que la plupart des utilisateurs utilisent le programme EnergyPlus à travers une interface développée à partir d'un développeur tiers. Pour rendre facile pour les utilisateurs réguliers des programmes DOE-2 et BLAST de se déplacer vers l'utilisation du programme EnergyPlus, l'équipe a écrit dix utilitaires qui convertissent les données d'entrée des charges thermiques des deux programmes vers la nouvelle structure des données d'entrée EnergyPlus.

Le format IFC (Industry Foundation Classes) de l'Alliance Internationale pour l'interopérabilité (IAI) est un format de fichier orienté objet utilisé par l'industrie du bâtiment pour échanger et partager des informations entre logiciels. Ce format a commencé à apparaître en 2001. L'IFC peut devenir un autre moyen de partage de l'information entre les logiciels dans l'industrie du bâtiment, y compris EnergyPlus (Bazjanac et Crawley 1999).

EnergyPlus utilise un fichier d'entrée de données de format libre qui contient une complète description du bâtiment et ses systèmes. La syntaxe de base est la suivante:

Objet, donnée, donnée, donnée …, donnée; (Object, data, data, data,…, data;)

L'objet est un mot prédéfini désignant un composant bâtiment, comme Surface, Matériau, Système d'éclairage, Ventilo-Convecteur, et Chaudière. Ce mot est suivi par une liste de valeurs de données et se termine par un point-virgule. Ces données décrivent les caractéristiques de performance et de l'utilisation prévue pour cet objet dans la simulation.

EnergyPlus lit un dictionnaire d'entrée des données lors de l'exécution pour déterminer la syntaxe de données d'entrée du fichier. La syntaxe générale du dictionnaire de données d'entrée est définie comme suit

Objet, A1 [un alpha], N1 [un certain nombre], ... ;

Par exemple, pour la commande du Lieu dans le programme EnergyPlus, la ligne dans le dictionnaire des données est la suivante:

Location, A1 [Location Name], N1 [Latitude], N2 [Longitude], N3 [Time Zone], N4 [Elevation];

Cela indique au processeur de gestion des données d'entrée qui , pour la commande *Location* ,de lire un champ de texte (A1) avec le nom de lieu , et quatre données d'entrées numériques (N1, N2 , N3 et N4) - latitude, longitude , fuseau horaire, et l'élévation, respectivement. Les mots entre crochets [ ] décrivent la variable et ses unités (mètres, litres / seconde, etc.).

Lors d'une simulation, EnergyPlus enregistre les résultats pour chaque pas de temps dans une structure de données de sortie et ces résultats peuvent être rapporté à chaque pas de temps ou réunies pour des temps d'intervalles plus longs. La structure des données de sortie utilise une approche semblable aux fichiers texte d'entrée des données - simple avec une syntaxe orientée-objets :

Time stamp, data, data, data; … (Horodateur, donnée, donnée, donnée; . . .)

Les données de sortie sont simples mais contiennent une grande partie des résultats de la simulation afin que les utilisateurs et les développeurs d'interfaces puissent facilement accéder aux résultats spécifiques sans modifier le moteur de calcul.

Parce que la structure de ces données est simple et séparées par des virgules, tout logiciel de post-traitement peut lire facilement ces données et peut créer des rapports plus élaborés. Un inconvénient avec ce type de format simple est que ces fichiers de sortie peuvent devenir très volumineux.

L'autre contribution majeure et l'introduction des données météorologiques. Plutôt que de représenter ces données sous forme de fichier binaire séparé et créé par un processeur, l'équipe de développement a préférée utiliser un format texte simple similaire aux fichiers de données d'entrée et de sortie. Le format des données météorologiques comprend dans les huit premières lignes les informations de base du site: lieu (nom, état /province/région, pays), la latitude, la longitude, l'altitude, le fuseau horaire, les conditions de conception de base pour le chauffage et le refroidissement, périodes de vacances, périodes d'économies journalière de la lumière, les périodes typique et extrême, etc. EnergyPlus ne demande pas une année pleine (8760 ou 8784 heures) de données pour ses fichiers de données météorologiques. En fait, EnergyPlus permet et lit des sous-ensembles sur un certain nombre d'années, voire même des sous-horaire (5 minutes, 15 minutes) de données. Le format des données météorologiques comprend aussi le champ "minutes".

EnergyPlus est livré avec un utilitaire qui lit des fichiers standards de types services de météorologie tels que TD1440 et DATSAV2 et les plus récent des fichiers météorologiques, telles que TMY2, IWEC, et WYEC2. Plus d'informations sur les formats des données météorologiques est contenue dans Crawley et al. (1999).

En résumé, tous les fichiers de données associés au programme EnergyPlus – les données d'entrée, de sortie, et les conditions météorologiques ont de simples et autonomes formats, mais ils peuvent devenir très volumineux. Les fichiers de données peuvent être facilement lus et interprétés par d'autres programmes tels que tableurs, bases de données ou des programmes d'interface personnalisée.

Le développement d'interfaces graphiques par des développeurs tierces, permet de garder la simplicité d'utilisation de ces fichiers par d'autres programmes que les concepteurs de bâtiments et leurs systèmes CVC associées utilisent.

2.7. Ajout d'un nouveau module

Un des principaux objectifs pour EnergyPlus est de rendre facile aux développeurs d'ajouter de nouvelles fonctionnalités et des modules. Le procédé est conceptuellement simple, avec les étapes générales suivantes :

- Trouver (ou développer votre propre) un modèle.

- Définir les paramètres du modèle, ces équations, ces coefficients spécialisés et les données nécessaires.

- Déterminer le point d'insertion "plug-in", où le module sera appelé dans le programme EnergyPlus.

- Ecrire les spécifications d'entrée.

- Déterminer les entrées qui seront nécessaires pour " exécuter" le modèle. Envisager comment cela peut être mis dans le Dictionnaire des données d'entrée / la structure du fichier d'entrée de données du programme EnergyPlus.

- Ecrire le nouveau module (s), ajouter l'appel au niveau supérieur du gestionnaire.

- Utilisation de la norme de programmation du programme EnergyPlus, écrire le code.

- Ecrire la routine ***GetInput***.

- Créer la syntaxe réelle des données d'entrée dans le Dictionnaire d'entrée des données pour produire le bon " GetInput " routine dans le nouveau module afin d'obtenir les données.

- Terminer la routine de simulation.

- utiliser l'option ' revérifier résultats', réparer le modèle.

2.8. Développement du programme EnergyPlus

Dans la documentation du programme EnergyPlus [Web 7], une rubrique est dédiée aux développeurs intéressés par le développement du programme EnergyPlus. Trois grands guides sont mis à la disposition des chercheurs que nous décrirons brièvement :

- *Guide pour les développeurs de Modules* : Toutes les informations que le développeur aura besoin pour développer des modules de simulation pour l'intégration avec le code source existant dans le programme EnergyPlus.

- *Guide pour les développeurs d'interfaces graphiques :* Toutes les informations que le développeur aura besoin pour développer des interfaces pour être utilisées avec le programme EnergyPlus, et plus précisément l'information sur les formats de fichiers des données d'entrée et de sortie.

- *La programmation standard* : Style de programmation utilisé par l'équipe de développement du programme EnergyPlus. Ce document aidera les utilisateurs ou les développeurs à comprendre le code source.

Dans notre travail, nous ne discuterons uniquement que le deuxième point, qui est le développement d'interfaces graphiques. Dans notre approche l'idée est de travailler sur le développement de prototype d'interface graphique pour la création d'objets susceptibles d'être intégrés dans le fichier de définition des données appelé IDD (Input Data Définition). Ce fichier sera utilisé par les développeurs de modules dans la création de nouveaux modules de simulation qui utilisent de nouveaux objets non encore intégrés dans ce fichier.

Comme il a été exposé précédemment, EnergyPlus a était conçu en tant que moteur de simulation sans interface graphique particulière. Toutes les versions d'EnergyPlus sont livrées avec un éditeur simple pour la préparation du fichier d'entrée de données appelé IDF (Input Data File). Cet éditeur lit le Dictionnaire de données IDD, et permet la création/révision des fichiers d'entrée de données IDF du logiciel EnergyPlus. Ce fichier est généralement produit par les développeurs du programme.

Les objets sont généralement définis dans le fichier IDD - fichier de définition des données d'entrée -. Le fichier est de type texte – ASCII - ou les objets sont définis comme suit :

Location, A1 [Location Name], N1 [Latitude], N2 [Longitude], N3 [Time Zone], N4 [Elevation];

La figure 2.5 illustre une portion du fichier IDD ou l'objet Location (c.-à-d. Lieu) et ses attributs sont définis. L'objet Location possède plusieurs attributs(ou champs) tels que les attributs Nom, latitude, longitude, etc. les attributs sont de type alpha (attribut nom) ou de type numérique (latitude, type réel).

```
Site: Location,
      \unique-object
       \min -fields 5
 A1, \field Name
      \required -field
```

Figure 2.5 Exemple de l'objet site(Location) décrit dans le fichier IDD.

En outre, le caractère spécial « \ » introduit un commentaire un par ligne et la plupart sont suivis par une valeur. Les commentaires peuvent s'appliquer à un champ ou à un objet ou à un groupe d'objets. Le fichier dictionnaire de données (IDD file), précise les exigences pour chaque élément.

Le processeur d'entrée des données du programme EnergyPlus utilise ces exigences pour traiter les entrées des données du fichier IDF pour la simulation et signaler toute anomalie trouvée.

Les deux fichiers d'entrée ont des structures similaires : 1) Les sections - des lignes simples ou des commandes, qui peuvent aider à regrouper les données d'entrée de la simulation pour une meilleure lisibilité et 2) Classes et Objets - les attributs de données pour la simulation. Les classes sont le terme utilisé dans le dictionnaire de données - chaque classe spécifie le type de données (de type alphanumérique- caractère- ou numérique) qui sera inclus dans les données d'entrée pour la simulation. Les objets sont des instances de ces classes et apparaissent dans le fichier IDF avec des valeurs appropriées, voir figure 2.6.

```
DENVER_STAPLETON_CO_USA_WMO_724690,  ! - Name
   39.77        !- Latitude {deg}
   -104.87      !- Longitude {deg}
```

Figure 2.6 Exemple de l'objet Location avec ces attributs décrit dans le fichier IDF

Les fichiers des données d'entrées du programme EnergyPlus (IDF), l'éditeur de ce fichier offre ce service.

Beaucoup de parties tierces travaillent sur le développement d'interface graphiques réelles qui prennent en charge la création et ou la modification du fichier IDF. Nous ne citerons que quelques interfaces à titre d'exemple :

➢ *Easy EnergyPlus* : est une interface en langue chinoise pour le programme EnergyPlus et qui est en cours d'élaboration par l'École des sciences et du génie de l'environnement, de l'Université de Tianjin en Chine. La version bêta de Easy EnergyPlus accepte les données d'entrée requises par EnergyPlus et exécute une simulation avec le moteur du programme EnergyPlus [Web 1].

➢ *EnergyPlugged* : est un AutoCAD plug-in publié par la firme Autodesk pour créer et éditer des fichiers d'entrée pour le programme EnergyPlus. EnergyPlugged a été conçu pour améliorer et accélérer la création de modèle pour le programme EnergyPlus sans perdre le contrôle de celui-ci tout en préservant souplesse d'une part et détecter les erreurs d'autre part [Web 1].

➢ *EP-Quick* : ce logiciel crée des fichiers d'entrée pour de nombreux types bâtiments à l'aide de modèles préconstruits pour la forme et la disposition des zones. En utilisant ces modèles, le temps nécessaire pour créer un fichier d'entrée pour le programme EnergyPlus est fortement réduit et cela fonctionne pour différente taille de bâtiment. La version 1.0 a été publiée en Avril 2005. EP-Quick est disponible pour téléchargement gratuit [Web 7].

➢ *EP GEO and EP SYS* : la compagnie NaturalWorks a développée deux feuilles de calculs basés sur des interfaces qui peuvent compléter les outils d'interface simples qui sont inclus dans l'installation standard du logiciel EnergyPlus [Web 7].

- *EP GEO* : Une feuille de calcul simple qui utilise un ensemble de macros simples pour créer la géométrie du bâtiment de forme rectangulaire, fenêtres, l'ombrage, l'infiltration, les gains internes et contrôle de la température. Les zones de formes rectangulaires peuvent être créées automatiquement dans un fichier IDF en entrant simplement la hauteur de la zone, sa largeur et sa longueur. L'utilisateur peut insérer plusieurs zones dans un fichier existant.

- *EP SYS* : Cette feuille de calcul permet la création de système de ventilo-convecteurs, et de systèmes à volume d'air variables dans un grand nombre de zones. Une liste de zones dans un fichier existant IDF peut être automatiquement importées et des zones individuelles peuvent être sélectionnées pour l'insertion de l'un des trois principaux types de systèmes disponibles dans l'outil.

2.9. Conclusion

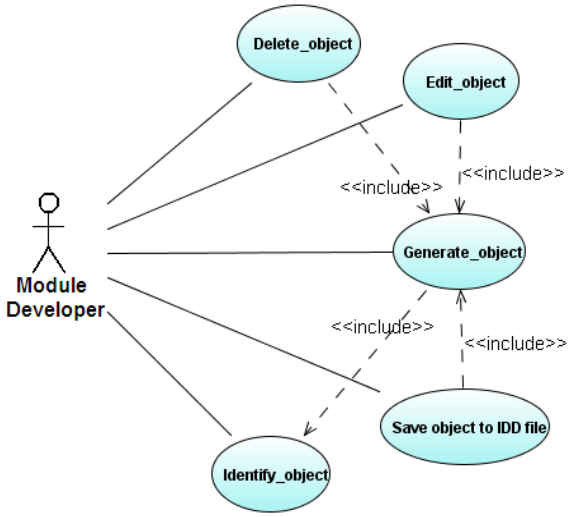
Dans ce chapitre nous avons passés en revue l'historique du programme EnergyPlus. Ce programme est basé sur un système bien organisé. Sa structure modulaire du code facilite l'ajout de fonctionnalités et de liens vers d'autres programmes. EnergyPlus est un programme développé en tant que moteur de simulation sans interfaces graphiques particulières. Le développement des interfaces graphiques est laissé à des parties tierces. Nous avons cités dans notre travail quelques interfaces graphiques développées par des personnes morales ou physiques. Ces interfaces permettent la création de fichiers IDF.

Dans notre travail nous proposons une méthodologie de création de prototype d'interface graphique dédiée à la création d'objets susceptibles d'être intégrés dans le fichier IDD lors de la création de nouveaux modules de simulation. Dans le chapitre suivant nous étalerons les deux concepts utilisés dans la réalisation du prototype d'IGC qui sont la notation UML et

l’approche scenarios. Nous aborderons aussi les techniques utilisées dans l'ingénierie des exigences de logiciels pour un système interactif (un programme offrant une interface exécuté par un utilisateur). On aura à produire une vision système et une formalisation des scénarios à l’aide des réseaux de Pétri (RdPs).

Chapitre 3

Notation UML & Scenarios



Les scénarios ont trouvés leur place dans la plupart des méthodes orientées objet pour identifier les objets du système et représenter ses exigences fonctionnelles. L'analyse orientée objets et les méthodes de conception - les méthodes de modélisation - deviennent ainsi un élément essentiel dans le développement de logiciels. La conception du logiciel qui serait difficile à décrire textuellement, peut facilement être transcrite par des diagrammes. La modélisation offre trois grands avantages : la visualisation, la gestion de la complexité, et une communication claire. La notation de modélisation unifiée (UML) qui fait partie de l'Object Management Group (OMG) est un langage visuel pour la spécification, la construction, et la documentation des artefacts d'un système à travers des modèles et des diagrammes, et il offre un bon cadre pour l'ingénierie des scénarios. Le comportement dynamique du système est étudié avant qu'il ne soit construit à l'aide de modèles des réseaux de Pétri, et qui peut être analysée par le biais d'une simulation.

3.1. Introduction

La construction des interfaces usager (UI) est devenue un rouage important dans l'ingénierie du logiciel. Au cours des dernières années, de nombreuses méthodes et outils ont vu le jour, permettant de réduire la charge de travail des développeurs, et de produire des interfaces de meilleure qualité. Au début des années quatre-vingt, le terme interface usager était pratiquement inconnu. Les développeurs s'intéressaient beaucoup plus à la validité des applications et qu'elles accomplissent les traitements attendus. Depuis leur apparition, les interfaces usager n'ont pas cessés d'évoluer en apportant plus de souplesse à l'usager, et en répondant à ses besoins de manière efficace.

Deux paramètres ont beaucoup contribués à promouvoir l'utilisation de ces interfaces, le progrès technologiques de l'informatique et les sciences cognitives, lesquelles ce sont intéressées à l'étude de l'interaction humaine avec les systèmes informatique. Par l'intégration des règles ergonomiques (facilité d'utilisation, la concision, la cohérence, la souplesse, etc.), ces interfaces méritait l'évaluation des usagers. Une autre raison principale de ce succès vient en raison du fait que l'usager est le directeur de la corrélation avec l'application pendant toute la séance de travail.

La mise en œuvre de ces types d'interfaces usager apporte toutefois une nouvelle complexité dans la tâche de développement de logiciels. La complexité réside dans la description de dialogue multitâche qui est exécuté par l'usager pour mener plusieurs tâches en même temps. Au niveau de la spécification de l'interface usager, les développeurs peuvent dessiner ces interfaces, mais il est souvent difficile de préciser leur comportement dynamique en raison de la complexité du dialogue. Les méthodes formelles (réseaux de Pétri de haut niveaux, les diagrammes d'états de Harel, etc.) seront d'une grande utilité dans la spécification et la vérification du comportement de ces interfaces usager avant leur mise en œuvre, ce qui permettra de réduire le temps des tests, et d'améliorer la qualité de l'interface usager.

En ce qui concerne la description des interactions des usagers avec un programme informatique, et pour l'étude de nouvelles solutions dans le re-engineering des approches, les scénarios ont reçu beaucoup d'attention, et ont été identifiés comme un moyen efficace pour comprendre les besoins des usagers, et pour analyser l'interaction homme-machine [Potts et al. 94]. Ils sont utilisés dans diverses activités des cycles de développement, depuis l'acquisition des besoins des usagers à des activités d'essai par voie de cahier des charges et des prototypes, et par l'évaluation des différentes alternatives de conception. Il est donc possible de dire que les scénarios sont utilisés à des fins de description et d'exploration.

Les scénarios trouvent aussi leur place dans la plupart des méthodes orientées objet telles qu'OMT, OOSE, etc., pour identifier les objets du système et représenter les exigences fonctionnelles.

L'analyse orientée objet et les méthodes de conception (méthodes de modélisation) deviennent ainsi un élément essentiel dans le développement du logiciel. La conception du logiciel qui serait difficile à décrire textuellement, peut facilement être

transportée à travers des diagrammes. La modélisation offre trois avantages clés : la visualisation, la gestion de la complexité, et une communication claire. Le langage de modélisation unifié (UML) qui fait partie de l'Object Management Group (OMG) est un fruit des efforts d'unification de plusieurs méthodes orientées objet. UML est un langage visuel pour la spécification, la construction, et de documentation des artefacts d'un système à travers des modèles et des diagrammes, et il offre un bon cadre pour l'ingénierie des scénarios.

Tout processus respectable de l'ingénierie des exigences du logiciel pour un système interactif (un programme offrant une interface exécuté par un usager) aura à produire une spécification du comportement du système à des fins de validation , de vérification, et d' évaluation par l'usager du prototype d'interface usager. Compte tenu de la pertinence dans l'acquisition des besoins des usagers, les scénarios sont utilisés comme un moyen pour une description du comportement du système.

Étant des descriptions partielles, les scénarios d'un système modélisé ont besoin d'une opération d'intégration, et la spécification résultante doit être formalisée par des méthodes formelles. Le comportement du système peut être étudié avant qu'il ne soit construit à l'aide, par exemple d'un modèle de réseaux de Pétri de haut niveaux ; modèle qui peut être analysée soit par le biais de la simulation (ce qui équivaut à l'exécution du programme et le débogage de celui-ci) ou par des méthodes plus formelles d'analyse [Jensen et Kristensen]. Le processus de la création de la description et la procédure d'analyse offre généralement au développeur une meilleure compréhension du système modélisé.

Dans ce chapitre on donnera un aperçu général du domaine des interfaces usagers (IUs) ou nous discuterons brièvement différents aspects de l'IU d'un système (nous étalerons quelques règles ergonomiques qui découlent des sciences cognitives, nous décrirons le modèle d'architecture MVC, et nous passerons en revue aussi quelque méthodes de conception orientées objet qui intègrent l'aspect IU comme activité principale dans leur cycle de développement). Nous présenterons aussi les différents aspects des scenarios, l'utilisation des scenarios dans la notation UML, et discuterons l'approche utilisée, dans l'intégration des scénarios.

3.2. Les systèmes interactifs

Un système interactif est une application proposant une interface dirigée par l'utilisateur; l'application ne prédéfinit aucune séquence d'opérations et se contente de répondre aux requêtes qu'elle reçoit de ses utilisateurs. Une interface peut être vue comme un dispositif qui sert de limite commune à plusieurs entités communicantes. Elle doit assurer à la fois la connexion physique entre les entités et effectuer des opérations de traduction entre les formalismes des parties communicantes. Dans le cas de l'IU la connexion a lieu entre l'image externe du système et les organes sensoriaux de l'utilisateur. La réalisation d'une telle interface suppose donc la connaissance précise du comportement de chacune des entités à relier, ce qui rend cette tâche complexe et souvent empirique. D'où l'intérêt des sciences cognitives qui se sont intéressées essentiellement à la modélisation du

processeur humain et à la définition de règles ergonomiques permettant de faciliter les interactions des utilisateurs.
Les sciences cognitives proposent des théories pour l'étude du comportement humain. L'intégration de ces sciences avec l'informatique est d'une grande utilité dans la conception des IUs. L'avancée technologique au niveau hardware et software a fait bondir les applications interactives au premier plan. Actuellement le développement de l'IU prend une grande place dans le développement des applications informatiques. Certaines évaluations [Myers 95] ont révélé que plus de 50% du temps de développement est consacré à la réalisation de l'IU, et que plus de 70% du coût d'un produit logiciel s'accumule dans sa maintenance où la maintenance de l'IU représente la grande part.

3.2.1. Sciences cognitives et l'interface usager

L'IU doit son évolution aux développements parallèles et partagés des sciences cognitive et informatique. Pour les ergonomes et psychologues, l'interaction homme-machine représente l'ensemble des phénomènes physiques et cognitifs qui interviennent dans la réalisation d'une tâche par l'humain. Les informaticiens sont plus concernés par l'aspect technologique de l'interaction et son adéquation avec la perception humaine. Les sciences cognitives s'intéressent essentiellement à la modélisation du processeur humain et à la définition de règles ergonomiques permettant de faciliter les interactions des utilisateurs.

3.2.2. modélisation du processus humain

On distingue deux grandes classes de modèles pour représenter le processeur humain: les modèles théoriques et les modèles appliqués. Les premiers tentent de formaliser les mécanismes qui régissent l'interaction homme-machine pour expliquer et prédire le comportement humain. Les modèles appliqués ont pour intérêt l'évaluation prédictive des performances dans l'interaction et servent de support de comparaison entre plusieurs conceptions.

Un des premiers modèles théoriques, proposé par Cardson [Cardson 83], représente un individu comme une machine intelligente munie de trois sous-systèmes : sensoriel, moteur et cognitif (figure 2).

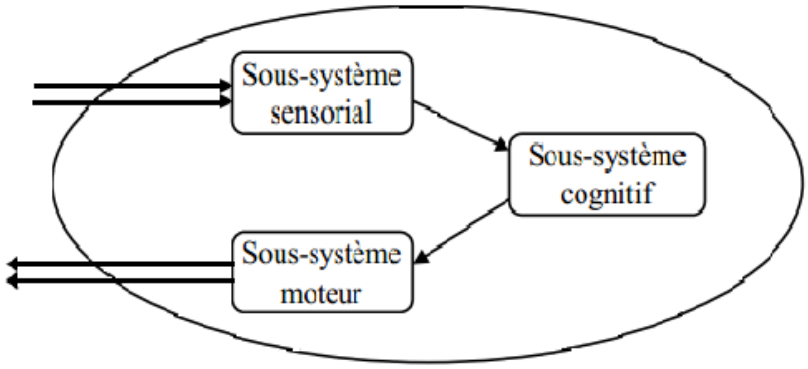


Figure 3.1 Modèle du processeur humain selon Cardson

Le sous-système sensoriel désigne le processeur des entrées. Il est responsable de la gestion de chaque organe sensoriel (device d'entrée). Le sous-système moteur est responsable du contrôle des mouvements de l'humain (device de sortie). Enfin, le sous-système cognitif dispose d'un ensemble de mémoires d'information à court et à long terme et contrôle le comportement de l'individu en fonction du contenu de ses mémoires. Ce dernier sous-système est semblable à un système de production (Reconnaissance → Action).

Les modèles appliqués les plus connus dans le monde des IUs sont les modèles GOMS et Keystroke. GOMS [Cardson 83] (Goal-Operator-Method-Selection) modélise l'activité cognitive d'une personne pour réaliser une tâche par des buts, des opérateurs, des méthodes et des règles de sélection. Un but désigne un état recherché du système. Un but peut être décomposé en sous-buts, les feuilles de cette hiérarchie étant des buts élémentaires appelés opérateurs. Il se peut que pour un but donné, il existe plusieurs alternatives possibles. Ces alternatives sont appelées des méthodes. Une méthode décrit donc le procédé pour atteindre un but. Des règles de sélection expriment alors le choix d'une méthode parmi celles possibles sous forme de conditions.

Le modèle Keystroke [Cardson 83] concerne les aspects syntaxiques et lexicaux de l'IU. Il suppose qu'il y a toujours une seule alternative pour réaliser un but donné et il s'intéresse au temps d'accomplissement d'une tâche en se basant sur six opérateurs de base :
K (Keystroking) : frappe de touche au clavier, P (Pointing) : déplacement de la souris vers une cible, H (Homing) : Rapatriement de la main, D (Drawing) : action de dessiner, M (Mental activity) : activité mentale pour décider l'action suivante et R (Response time) : temps de réponse du système. Pour chaque opérateur il existe une estimation moyenne de temps (T_K, T_P, T_H, T_D, T_M et T_R). Toute méthode peut être représentée avec ces six opérateurs (la commande "ls" par exemple : M K[l] K[s]K [retour chariot] = M3K), et on peut donc facilement estimer le temps qu'elle va prendre.

3.2.3. Règles ergonomiques

La définition de règles ergonomiques vise à améliorer l'utilisabilité des IUs. L'approche suivie pour construire une base de règles ergonomiques est généralement une approche expérimentale où on émet des hypothèses concernant l'environnement de travail et les tâches à réaliser, puis on essaie de les valider par des expérimentations pour décider de leur adéquation. Les règles ergonomiques doivent prendre en compte les aspects fondamentaux suivants [Coutaz 90] : la cohérence, la concision et la flexibilité.

3.2.3.1. Règles de cohérence

Les règles de cohérence visent l'organisation des étapes de réalisation d'une tâche. Elles concernent la spécification d'un plan de tâche, son exécution, sa perception et son interprétation. En général le choix d'une métaphore d'interaction (métaphore bureau par exemple) améliore énormément la cohérence de l'interaction et le temps d'apprentissage des utilisateurs. Voici, à titre d'exemple, trois règles sur la cohérence :

- Les sous-tâches partagées doivent être utilisées de la même façon dans les différents contextes d'appel.
- Il faut choisir une nomenclature précise et pertinente (lexique des termes) et une notation uniforme dans toutes les fonctionnalités de l'interface (notation préfixée par exemple : verbe objet). Le fait de donner plus de souplesse et d'accepter plusieurs notations n'est toujours en faveur de l'utilisateur.
- Les informations doivent être là où l'utilisateur les attend.

3.2.3.2. Règles sur la concision

Les règles de concision ont pour but d'éviter les surcharges d'information en sortie et de réduire le nombre d'actions en entrée. Pour la r éduction du nombre d'actions en entrée, plusieurs techniques ont été utilisées dans les IUs tels que : les abréviations, les macro-commandes, le couper-coller, les valeurs par défaut, les fonctions refaire et défaire, etc. Pour éviter les surcharges d'information en sortie, il faut organiser la complexité visuelle de l'interface et structurer la présentation des fonctions, des menus et des messages. Voici, à titre d'exemple, trois règles sur la concision :

- La présentation des fonctions doit prendre en compte l'évolution des connaissances de l'utilisateur et offrir plusieurs modes d'utilisation du système (débutant, assisté et expert).
- Les menus de l'application doivent être équilibrés, pas trop larges ni trop profonds. Des observations expérimentales d'applications usuelles ont montré qu'une profondeur de 3 est une limite à ne pas dépasser.
- Les messages doivent être exprimés sous la forme la plus adéquate (forme textuelle ou iconique) dans le but d'informer l'utilisateur pour le rassurer ou l'aider à corriger ses erreurs.

3.2.3.3. Règles sur la flexibilité

La flexibilité désigne la faculté d'adaptation de l'IU aux variations de l'environnement et aux différents types d'utilisateurs. Cette adaptation peut être soit manuelle soit automatique.

- Une adaptation automatique repose sur une modélisation dynamique de l'utilisateur (modèle de l'usager). Le système, lors de l'interaction avec l'usager, peut déterminer le niveau de ce dernier, et on parle alors d'interfaces intelligentes.
- Une adaptation manuelle correspond à une personnalisation de l'interface par l'utilisateur pour des besoins de présentation et d'interaction. L'IU doit donc permettre une représentation multiple du même concept.

Les règles ergonomiques contribuent à améliorer l'utilisation de l'IU et à limiter les erreurs qui peuvent se produire lors de l'interaction homme-machine. Ces règles ont été prises en considération dans notre approche de génération du prototype de l'IU à partir des spécifications qui sera décrite dans le chapitre 4.

3.2.4. *Modèles d'architecture pour l'interface usager*

C'est après l'invention de la souris en 1964 et l'apparition du graphisme et du multi-fenêtrage en 1969 que la recherche dans le domaine des IUs s'est lancée. En 1983, la ville de Seeheim a accueilli un workshop sur le rôle, le modèle, la structure et la construction des systèmes de gestion des interfaces usagers (User Interface Management Systems : UIMS). Lors de ce workshop les intervenants ont défini un modèle d'architecture pour les IUs qui porte d'ailleurs le nom de la ville : modèle de Seeheim (figure 3.2). Par après, ce modèle a servi de base pour la naissance de nouveaux modèles plus élaborés et influencés par le paradigme orienté objet tel que par exemple le modèle MVC, celui-ci est identifié comme patron de conception [Bass et al. 91], reste le modèle le plus utilisé au niveau des méthodes orientées objet et des environnements de développements des systèmes interactifs.

3.2.4.1. Le modèle de Seeheim

Ce modèle [Pfaf 85] est l'un des premiers à proposer un découpage fonctionnel de l'IU. Il décompose une application interactive en quatre modules : le noyau applicatif, le module de l'interface de l'application, le contrôle de dialogue et la présentation. Ces composants sont structurés en couches (figure 3.2) :

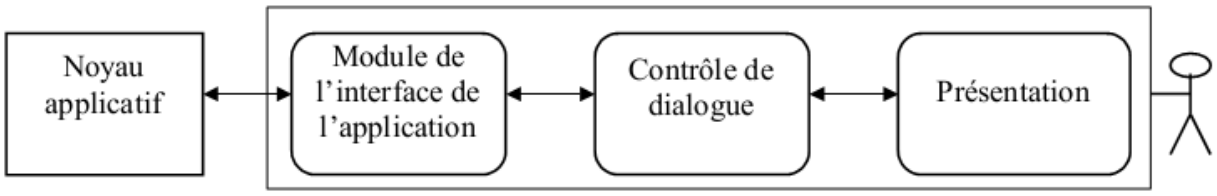


Figure 3.2 Modèle de Seeheim [Pfaf 85].

➢ Le module noyau applicatif regroupe l'ensemble des fonctions que réalise l'application (coté non interactif).

➢ Le module de l'interface de l'application relie le contrôle de dialogue au noyau applicatif par l'invocation des fonctionnalités offertes par l'application.

➢ Le module contrôle de dialogue gère le dialogue entre l'utilisateur et le système. C'est lui qui décide de l'accès ou du refus des interactions de l'utilisateur.

➢ Le module présentation représente la partie de l'application directement liée aux périphériques d'entrées/sorties. Il gère l'affichage à l'écran et les périphériques d'entrée (clavier, souris, etc.), et il est responsable de l'apparence de l'interface.

Cette architecture a été longuement critiquée du fait de sa structure en couches, de la centralisation du contrôle et de la passivité de son noyau applicatif. Le modèle de Seeheim définit clairement les rôles des quatre modules d'une application interactive au niveau conceptuel. Malheureusement, son implantation présente des difficultés puisque les rôles des modules de l'interface (le module de l'interface avec l'application, contrôle de dialogue et présentation) ne sont pas suffisamment explicités [Coutaz 90].

3.2.4.2. Le modèle MVC

Le modèle MVC (Model View Controlor) [Goldberg 84] est composé d'un triplé de composants autonomes qui communiquent entre eux (figure 3.3) :

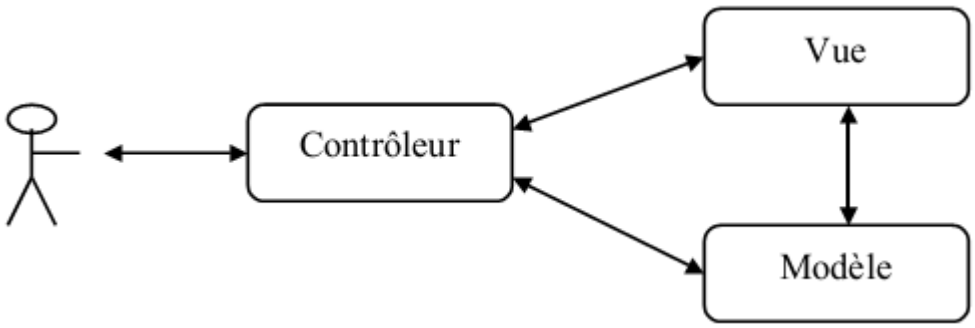


Figure 3.3 Modèle MVC

➢ Le modèle est la structure de données que l'on veut représenter à l'écran et qui est composée d'un certain nombre d'objets.

➢ La vue est la représentation externe du modèle. C'est par elle que l'utilisateur perçoit les objets du système et que le modèle reflète ses changements. Dans une application, à un même modèle peuvent correspondre plusieurs vues.

➢ Le contrôleur régule les interactions entre la vue et le modèle. Il est chargé de gérer les actions de l'utilisateur sur la vue, et il informe le modèle des changements faits sur celle-ci. Le modèle modifie son état et informe la vue du nouvel aspect qu'elle doit prendre.
Le modèle MVC se prête bien à des applications orientées objet.

3.2.5. L'interface usager dans les méthodes orientées objet

Pour obtenir des IUs de qualité, on doit tenir compte des capacités et limitations à la fois des humains et des systèmes informatiques existants, et des connaissances sociales et organisationnelles de l'environnement de travail de l'utilisateur. Ces facteurs pluridisciplinaires rendent difficile la tâche de trouver une méthode générale de conception des IUs. Les méthodes existantes favorisent généralement un des facteurs par rapport aux autres [Avision 90, Harton 90].

Au niveau des méthodes orientées objet (MOO), on parle plutôt d'interfaces utilisateurs orientées objet (Object-Oriented User Interface : OOUI) ou d'interfaces graphiques (Graphical User Interface : GUI). Des extensions ont été apportées aux MOOs pour tenir compte des IUs. Collins [Collins 95] a montré les principales activités qu'il faut ajouter aux MOOs pour supporter les OOUIs (figure 3.4).

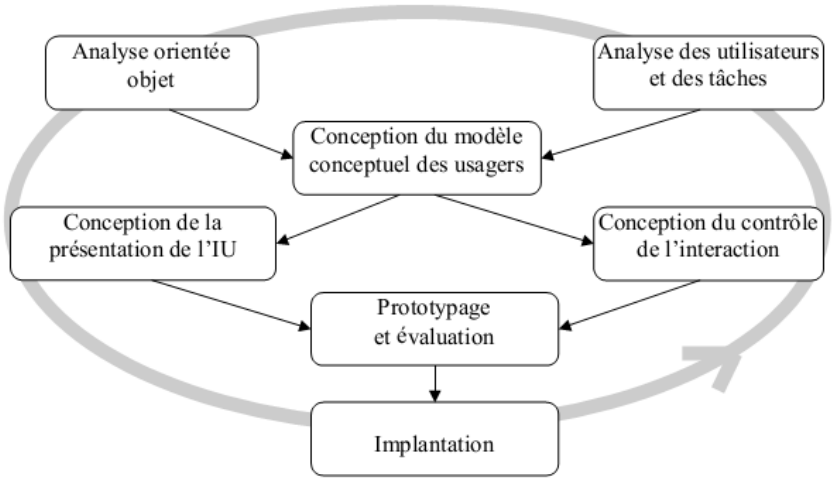


Figure 3.4 Activités de développement spécifique aux OOUIs [Collins 95]

L'analyse orientée objet concerne l'élaboration des modèles de données et de comportement du système. L'analyse des utilisateurs et des tâches permet de définir les buts attendus et le profil des utilisateurs futurs du système. La conception du modèle conceptuel des usagers concerne essentiellement le choix d'une métaphore pour améliorer l'utilisabilité de l'IU, qui sera par la suite raffinée par la conception de la présentation de l'IU et la conception du contrôle de l'interaction. L'étape prototypage et évaluation sert à la validation de l'IU avec l'utilisateur avant son implantation. Le cercle en gris de la figure 8 exprime le caractère itératif du processus de développement.

OMT++ [Jaa95] est un exemple de MOOs adaptée aux OOUIs. Cette méthode étend la méthode OMT (Object Modeling Technique) de Rumbaugh [Rumbaugh et al. 91] par l'ajout d'activités correspondantes à l'IU au niveau des étapes d'analyse, de conception et de programmation (figure 3.5) :

➢ Au niveau de l'étape d'analyse (AOO), la spécification de l'IU vient s'ajouter aux activités d'analyse des objets (par la détermination du diagramme des classes) et d'analyse du comportement (par les diagrammes d'états des classes pertinentes). La spécification des IUs consiste à établir la liste des tâches que l'utilisateur peut invoquer et à définir la structure de l'interface en organisant les tâches en une suite de dialogues. Un diagramme de dialogue (similaire au Statecharts) est utilisé pour exprimer le lien entre dialogues. En outre OMT++ offre une notation pour décrire les éléments d'un diagramme de dialogue (widgets).

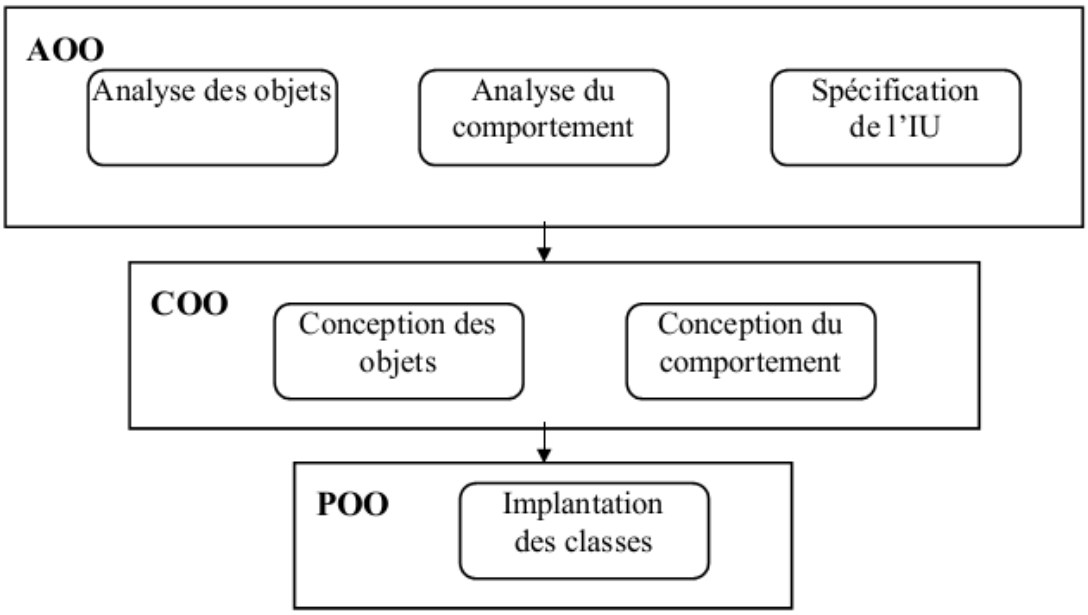


Figure 3.5 Étapes et activités de la méthode OMT++ [Jaa95]

➢ Dans la phase de conception (COO), la conception des objets consiste à concevoir les classes des vues et les classes des contrôleurs en se basant sur le modèle MVC. Pour un dialogue on associe un objet vue et un objet contrôleur.

La conception du comportement consiste à ajouter les opérations aux classes données, vues et contrôleurs.

- L'étape de la programmation objet (POO) a pour but l'implantation de l'ensemble des classes obtenues lors de l'étape de conception.

L'approche proposée dans notre travail qui utilise la notation UML est similaire au cadre offert par la méthode OMT++.

3.2.6. Les outils de prototypage

Dans le génie logiciel, le prototypage est considéré comme une approche ayant pour but l'amélioration de la planification et de l'exécution des activités de développement. Il supporte l'évaluation de la qualité du système dès les premières étapes du développement. Plusieurs expériences ont illustré l'impact de cette approche dans divers domaines. Gordon et al. [Gordon et al. 95] présentent un survol des expériences de prototypage dans trois domaines d'application : commercial, académique et militaire. Bäumer et al. [Bäumer et al. 96] comparent les approches de prototypage qui ont été utilisées dans 9 projets industriels selon le type de prototype construit (présentation, fonctionnel, etc.) et selon l'outil utilisé.

Dans les paragraphes suivants, on donnera une classification des approches de prototypage, des types de prototypes et des outils de prototypage qui peuvent être utilisés.

Une classification des approches de prototypage a été décrite par Bischofberger [Bischofberger et al. 92], selon le but attendu du système à développer :

- *Prototypage exploratoire :* permet aux développeurs de bien acquérir les besoins utilisateurs et de discuter la faisabilité de plusieurs solutions du même problème. La réalisation des prototypes est faite conjointement par les développeurs et les utilisateurs.

- *Prototypage expérimental :* a pour but la validation expérimentale d'une architecture de composantes du système ou des idées d'une solution. Ce type de prototype est réalisé uniquement par des développeurs pour permettre la conception des composantes d'un système où l'aspect qualité n'est pas primordial.

- *Prototypage évolutif :* dans cette approche le prototype ne sert pas uniquement comme support de validation avec l'utilisateur dans les premières phases, mais il est utilisé et amélioré pour aboutir au produit final.

Les prototypes obtenus à travers ces différentes approches peuvent être classifiés comme suit [Bischofberger et al. 92]:

- *Prototypes de présentation :* mettent surtout en évidence l'aspect visuel de l'IU.

➢ *Prototypes fonctionnels :* implantent les parties importantes de l'IU et les fonctionnalités du système.

➢ *Maquettes (breadboard):* permettent d'investiguer certains aspects techniques du système (architecture, fonctionnalités projetées, etc.) pour une évaluation des risques. Ils représentent un bon support de travail pour les développeurs.

➢ *Systèmes pilotes :* sont des prototypes plus mûrs et peuvent être directement utilisés dans la pratique.

Les outils utilisés dans les différentes approches de prototypage pour produire les différents types de prototypes précités peuvent être classés de la manière suivante [Szekely 95]:
➢ *Les outils de type hypertexte :* peuvent être utilisés pour le développement rapide de systèmes d'information simples avec des interfaces graphiques composées de fenêtres liées par des liens hypertextes contrôlés par des scripts associés aux composants d'une fenêtre (applications web).

➢ *Les constructeurs d'interface (interface builder) :* servent à définir des IUs à un haut niveau d'abstraction, soit sous forme textuelle en utilisant un langage, soit sous forme graphique en utilisant un éditeur. Ces derniers sont plus intéressants à utiliser dans le développement des IUs.

➢ *Les outils de 4ème génération :* offrent un environnement complet pour le développement des systèmes d'information. Ils offrent en plus des éditeurs graphiques pour les modèles de données et l'IU.

➢ *Les cadres d'application :* sont des librairies de classes qui comportent une conception abstraite et réutilisable pour les applications interactives. Ils offrent en plus du support de l'IU, un support pour l'architecture globale de l'application.

3.2.7. *Génération de l'interface usager*

La génération automatique des applications interactives touche à plusieurs domaines différents. Elle peut être appliquée soit lors de la conception, soit lors du prototypage, comme elle peut concerner l'IU ou toute l'application. Si l'on se reporte aux représentations d'une application interactive (figure 3.6), on constate qu'on peut avoir quatre types de générations possibles [Tarby 93] :

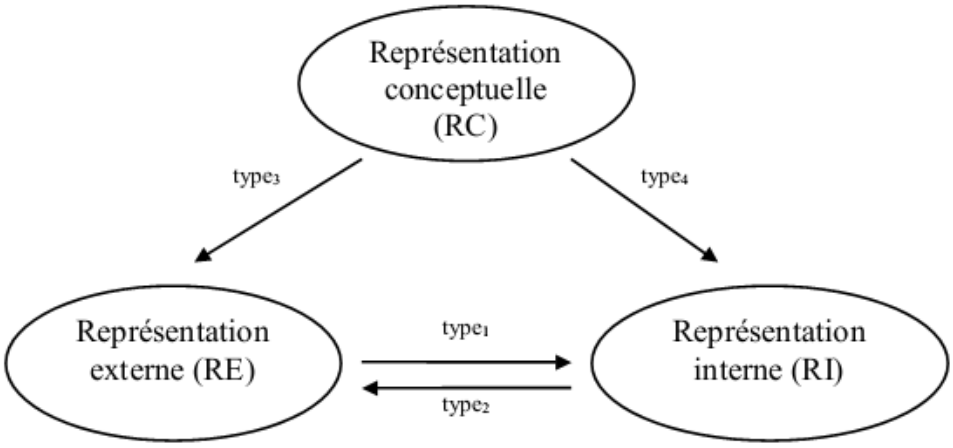


Figure 3.6 Types de générations possibles dans une application interactive [Tarby 93]

La génération de *type1* concerne la génération du code à partir d'une maquette d'écran. L'utilisateur dessine ses fenêtres, menus, etc. par l'intermédiaire d'un éditeur graphique. Les objets correspondants sont alors créés et traduits en un code particulier. Ce type de génération est supporté par les constructeurs d'interface.

- La génération de *type2* repose sur l'analyse du code (réingénierie). À partir de la structure de données manipulée dans le programme on peut dériver certains types de widgets (zone de saisie, boutons radios, etc.), et à partir de la structure de contrôle, on peut dériver la structure de menus.

- La génération de *type3* permet le passage d'une spécification conceptuelle (données, traitements et/ou comportements) à la représentation externe en se basant sur la syntaxe et la sémantique des spécifications.

- La génération de *type4* est plus générale que celle de *type3*, en effet elle concerne toute l'application et non seulement la partie présentation de l'IU. Le code généré n'est qu'un squelette qu'il faut compléter pour intégrer l'aspect sémantique de l'application.

Dans ce qui suit, nous présenterons aussi quelques travaux proposant la création des IUs à partir de modèles E/R. la génération de widgets en fonction de la sémantique des données manipulées, etc. :

- Johnson [Johson 92] a développé le système ACE pour générer des widgets en fonction de la sémantique des données manipulées. Pour chaque donnée, ACE propose un ou plusieurs widgets en fonction du type de la donnée et du critère de choix. Par exemple pour un critère de type 'un parmi n', les widgets proposés seront des boutons radios ou une liste d'éléments.

- Foley [Foley 93] propose un outil générant la présentation en se basant sur un ensemble de règles ergonomiques correspondantes aux données du système.

➢ Janssen [Janssen 93] a basé la génération de l'IU sur des vues du modèle Entité/Relation (E/R) du système. Une vue consiste en un sous-ensemble d'entités et de relations du modèle E/R global intervenant dans l'exécution d'une tâche. Une fenêtre composée d'objets graphiques liés aux entités et aux relations est associée à chaque vue.

➢ Bodart [Bodart et al. 94] combine l'analyse des données, l'analyse des tâches et l'analyse des fonctionnalités du système dans le développement de l'IU. Un graphe d'activité est construit pour lier les tâches interactives aux données et aux fonctions. Ce graphe sert par la suite à identifier les unités de présentations de l'IU.

➢ Balzert [Balzert 96] propose une approche pour dériver l'IU à partir du modèle objet en se basant sur un ensemble de règles. Par exemple : chaque classe du modèle objet donne lieu à une fenêtre où les attributs de la classe sont transformés en fonction de leur type en zones de saisie, boutons radios ou toute autre forme d'affichage ou de saisie. Les méthodes sont transformées en boutons.

3.3. Scenarios

Depuis plusieurs années, les scénarios ne cessent de prendre de plus en plus d'importance dans le génie logiciel. Ils ont été utilisés dans diverses activités du cycle de développement des systèmes informatiques depuis l'acquisition des besoins jusqu'aux activités de tests, en passant par la génération de spécifications et de prototypes [Hsia et al. 94] et par l'évaluation de diverses alternatives de conception [Roy95]. C'est dans le domaine de l'ingénierie des besoins (requirements engineering) que les scénarios ont eu le plus de succès jusqu'à date. Ils ont été utilisés pour identifier, décrire et raffiner les objectifs et les fonctionnalités du système [Potts 95].

Plusieurs définitions du terme scénario existent dans la littérature. Le dénominateur commun à ces définitions est qu'un scénario décrit une vue partielle du système. Cette notion de partialité offre la possibilité d'une description restreinte selon l'intérêt du descripteur et permet aussi d'avoir plusieurs vues différentes et complémentaires du même système. Dans le domaine des IUs, les scénarios sont vus comme une utilisation du système dans un contexte donné. Un scénario est une suite d'interactions entre l'utilisateur et l'ordinateur [Carroll 95]. Dans les systèmes d'information, les scénarios permettent de donner une image détaillée des aspects fonctionnels et organisationnels du système et de relier les besoins utilisateurs aux fonctionnalités du système [Kyng 95]. Enfin, dans le domaine du développement orienté objet, les scénarios sont utilisés pour accompagner toutes les phases du cycle de développement [Jacobson et al 92]. Un scénario est défini comme une suite d'interactions entre les objets du système.

3.3.1. Aspects des scénarios

Les scénarios ont évolué dans le temps selon plusieurs aspects, et leur interprétation semble dépendre du contexte d'utilisation et de la façon dont ils étaient acquis ou générés. Dans un travail de récapitulation, Rolland [Rolland et al. 98] a proposé un

cadre pour la classification des scénarios selon quatre aspects : la forme, le contenu, le but et le cycle de développement (figure 3.7).

3.3.2. Forme des scénarios

La forme des scénarios concerne essentiellement les opérations d'acquisition, de spécification et de représentation. Les scénarios sont acquis en utilisant différents médias, Scénario Contenu Cycle Forme But décrit sous possède a pour a un 30 ils sont décrits et définis d'une manière plus ou moins formelle, et ils sont reproduits ou représentés selon diverses formes dans des buts de validation ou d'évaluation.

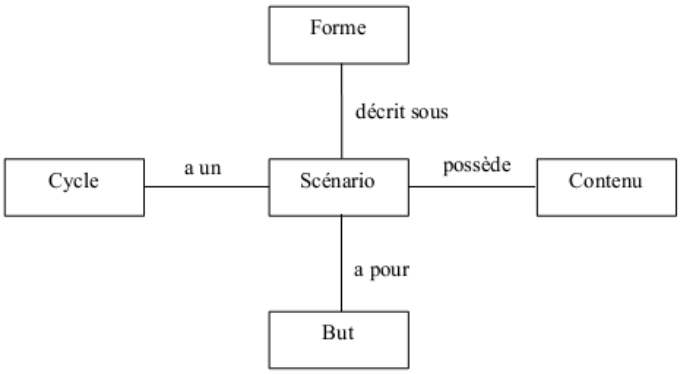


Figure 3.7 Aspects de scénarios [Rolland et al. 98]

Lors de la description des scénarios, des langages de modélisation (semi-formels) ou des techniques de spécification formelles peuvent être utilisés. Ces descriptions formelles ou semi-formelles sont très utiles pour la validation des besoins. Par exemple dans [Jacobson et al 92], les scénarios sont décrits en utilisant des diagrammes de séquence; dans UML les scenarios peuvent être décrits par des diagrammes de collaboration ou de séquence; dans [Hsia et al. 94] les scénarios sont décrits par des grammaires régulières; dans [Kawashita 97] ; les scénarios sont décrits par des automates; dans [Khriss et al. 99] les scénarios sont formalisés à l'aide des Statecharts; dans [Elkoutbi 98] les scénarios sont décrits en utilisant des formalismes dérivés des réseaux de Pétri; etc.

Lors de la validation et l'évaluation des besoins, les techniques de simulation, d'animation et de prototypage sont souvent utilisées [Heymans 98]. Ainsi les scénarios se trouvent représentés sous forme de prototypes destinés à l'animation pour la validation des spécifications ou destinés à la visualisation pour l'évaluation de l'aspect interface usager du système.

3.3.3. Contenu des scénarios

Le contenu d'un scénario fait référence au type d'informations et de connaissances qui y sont capturées. En effet, les scénarios peuvent adresser les niveaux

organisationnels ou stratégiques d'un système (scenario in the wide) [Kyng 95], comme ils peuvent concerner des descriptions détaillées de comportement (scenario in the narrow) [Jackson 96]. Selon Kuuti [Kuuti 95], le contenu d'un scénario peut décrire le fonctionnement interne du système en main, les interactions entre le système et son environnement, et possiblement les aspects organisationnels du système. Le contenu des scénarios dépend aussi de la manière dont les scénarios décrivent le système.

Dans notre travail, le contenu des scénarios dépend de la façon dont les scénarios décrivent le système. On distingue entre les scénarios abstraits et les scénarios concrets.

Les scénarios abstraits font référence à des objets du système (ex: Objet : Location, Longitude – champ numérique, Nom - alphanumérique, etc.), tandis que des scénarios concrets utilise des exemples particuliers d'objets d'entrée (ex. DENVER_STAPLETON_CO_USA_WMO_724690 ! Name ; -104.87 !- Longitude {deg}, etc., voir l'exemple du chapitre 2, § 2.6).

3.3.4. But des scénarios

Les scénarios sont utilisés pour la capture des besoins des utilisateurs, pour la description de l'interaction d'utilisateur avec un système, et pour l'investigation de nouvelles solutions dans l'approche de réingénierie [Potts 95]. Il est donc possible de dire que les scénarios sont utilisés à des fins de description et d'exploration. Les scénarios de description sont surtout utilisés pour comprendre et décrire les aspects comportementaux d'un système, où les scénarios représentent des points de vue des utilisateurs externes. Les scénarios d'exploration sont utilisés lorsque plusieurs solutions sont à explorer et à évaluer avec l'intention de choisir le meilleur.

3.3.5. Cycle de vie des scénarios

Si l'on considère des scénarios décrivant un système comme étant des objets d'un système, il est alors possible de parler de scénarios persistants par analogie aux objets persistants. Les scénarios persistants accompagnent le cycle de l'évolution depuis l'analyse des besoins jusqu'à la production de la documentation [Potts 95]. Par opposition. On définit les scénarios temporaires comme ayant servi seulement au niveau de certaines étapes du cycle de développement. C'est le cas, par exemple, ou un scénario peut être l'objet d'opérations suivantes durant son cycle de vie : capture (création), raffinement, intégration, extension et suppression. [Hsia et al. 94].

3.4. La notation UML et l'utilisation des scénarios

Les scénarios trouvent leur place dans la plupart des méthodes orientées objet (OMT, Fusion, OOSE, etc.) pour identifier les objets du système et représenter les exigences fonctionnelles. Dans la méthode OOSE [Jacobson et al 92], les scénarios (cas d'utilisation ou *use cases*) dirigent toutes les étapes de la conception orientée

objet. Ils sont utilisés comme point de départ pour la construction de tous les autres

modèles de la méthode à savoir le modèle objet, le modèle d'analyse, le modèle de conception, le modèle d'implantation et le modèle de tests.

UML [Rumbaugh et al. 99] est un langage de modélisation orienté objet qui fait partie des standards de l'OMG (Object Management Group). Il est le fruit d'efforts d'unification de plusieurs méthodes orientées objet avec des objectifs de simplification afin de profiter de tous les avantages de ces méthodes. Il a été conçu pour être utilisé comme un langage de modélisation, indépendamment des langages de POO classiques. UML a évolué avec de nombreuses versions depuis qu'elle a commencé officiellement en 1994. UML 2.0 est la version dans tous le processus de développement de notre travail. UML définit neuf types de diagrammes, chacun d'eux représente une vision spécifique du système :

- vue fonctionnelle ou interactive, décrit avec l'aide de diagrammes de cas d'utilisation, diagrammes de séquence et les diagrammes de collaboration.

- vue structurel ou statique, représentée avec l'aide de diagrammes de classes, diagrammes d'objets, de diagrammes de composants et des diagrammes de déploiement.

- vue dynamique, exprimée par des diagrammes d'états-transitions et des diagrammes d'activité.

Dans ce qui suit, nous discutons d'abord des diagrammes UML qui sont pertinents pour notre approche : le diagramme des cas d'utilisation, le diagramme de séquence, et le diagramme de classe.

3.4.1. Diagramme de classes(IDDClassD)

Un diagramme de classe correspond à la vue statique du système. Il permet d'identifier toutes les classes du système et de spécifier pour chaque classe ses attributs, ses opérations et ses liens avec les autres classes du système. La figure 4.1, donnée plus loin (chapitre 4), montre le diagramme *IDDClassD* du système adopté (diagramme proposé pour représenter les objets identifiant la structure du fichier IDD du programme EnergyPlus). Les relations entre classe permises par le diagramme de classes d'UML sont les associations, les agrégations (composition) et les généralisations (héritage). Le diagramme de classe est sans doute le diagramme central de toute la modélisation objet. Il est souvent enrichi et complété au fur et à mesure de l'élaboration des autres diagrammes.

3.4.2. Diagramme de cas d'utilisation (IDDUseCaseD)

Un diagramme de cas d'utilisation montre les différents cas d'utilisation du système et les différents acteurs qui interagissent avec le système. La figure 4.2 montre le diagramme d'utilisation proposé pour le fichier IDD. Un cas d'utilisation décrit une transaction complète faisant interagir les acteurs et les objets du système. Un acteur

est un objet externe du système qui interagit avec ce dernier dans le but d'obtenir un service offert par le système. Les cas d'utilisation sont représentés au sein du diagramme *de cas d'utilisation* par des ellipses, et les acteurs sont représentés par des icônes (voir figure 3.8). Un cas d'utilisation peut appeler les services d'un autre. UML définit deux relations permettant la structuration et la réutilisation des cas d'utilisation : utilise (uses) et étend (extends). Elles sont représentées par des flèches orientées de l'appelant vers l'appelé étiquetées par le type de relation (voir figure 3.8).

La figure 3.8 montre un exemple d'un cas d'utilisation comportant quatre cas d'utilisation (UC_1, UC_2, UC_3, et UC_4) et deux acteurs interagissant avec trois de ces cas d'utilisation (lignes continues). La relation utilise (uses) entre deux cas d'utilisation (UC_1 uses UC_3 dans la figure 3.8) signifie que le premier cas d'utilisation (UC_1) inclut toujours le comportement du deuxième cas d'utilisation (UC_3).

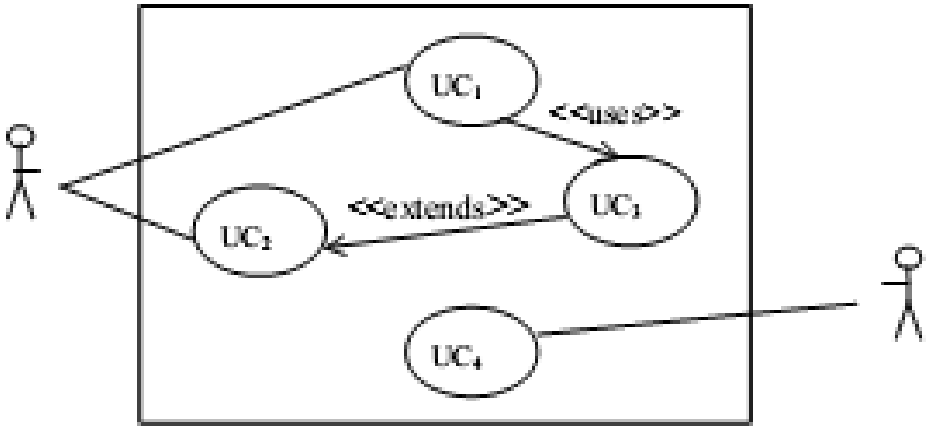


Figure 3.8 Exemple d'un cas d'utilisation

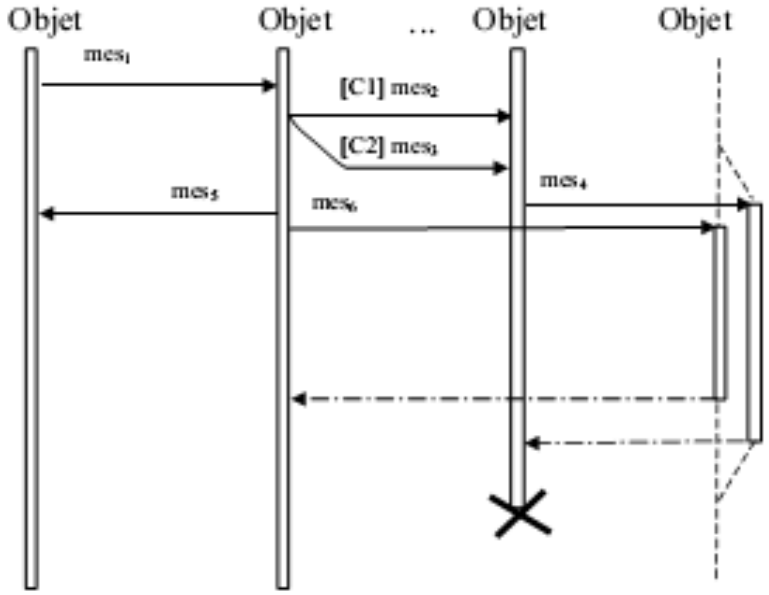
La relation étend (*extends*) entre deux cas d'utilisation (UC_3 extends UC_2 dans la figure 3.8) signifie que le deuxième cas d'utilisation (UC_2) peut se comporter sous certaines conditions (point d'extension) comme le premier cas d'utilisation (UC_3).

Dans notre travail, nous avons considéré uniquement la relation uses. La relation extends peut être vue comme une relation « *uses* » avec une condition sur l'appel.
Un cas d'utilisation est généralement décrit par un ensemble de scénarios. Un scénario est une exécution (instance) du cas d'utilisation et correspond à une série particulière d'interactions entre les objets du système. Dans UML, les scénarios peuvent être décrits en utilisant soit les diagrammes de séquence, soit les diagrammes de collaboration. Les deux types de diagrammes reposent sur la même sémantique, et il est toujours possible de passer d'un type de diagramme à un autre.

3.4.3. Diagramme de séquence (IDDSequenceD)

Un diagramme de séquence permet de montrer le séquencement des messages échangés entre les objets participant dans un scénario. La figure 4.3 illustre deux

IDDSequenceD décrivant deux scénarios pour le fichier IDD. Les lignes verticales (figure 3.9) représentent les objets et les flèches horizontales représentent les messages échangés. Dans ce type de diagramme, on s'intéresse à l'ordonnancement des messages en fonction du temps (l'axe du temps croît du haut vers le bas). Un diagramme peut capturer de manière visuelle des comportements conditionnels, itératifs et concurrents. La figure 3.9 illustre certains de ces comportements, par exemple, les messages mes_2 et mes_3 représentent un comportement conditionnel. Les messages mes_4 et mes_6 donnent lieu à un comportement concurrent de l'objet objet n.



.

Figure 3.9 Exemple d'un diagramme de séquence

Par rapport au cadre de classification des scénarios présentée dans la section 2.1 de ce chapitre, UML n'impose aucune contrainte sur les aspects but et contenu des scénarios. En effet, les scénarios d'UML peuvent être utilisés pour des buts de description, d'explication et d'exploration. Les connaissances et les informations qui y sont décrites dépendent uniquement du domaine d'application. Au niveau de l'aspect forme (section 3.3.2). UML propose une notation semi-formelle (*IDDUseCaseD*, *IDDSequenceD*), d'une sémantique riche, pour spécifier et structurer les scénarios du système. UML ne supporte pas les opérations de validation et d'évaluation des scénarios.

Dans notre travail de recherche, nous avons adopté l'approche qui supporte les opérations de validation et d'évaluation des scénarios. Les opérations définies dans l'aspect cycle de vie des scénarios (section 3.3.5) sont toutes supportées par UML à part l'opération d'intégration de scénarios. Pour cela, nous proposons une méthodologie semi manuelle pour l'intégration des scénarios d'UML.

3.5. Conclusion

Au vu des travaux d'intégration dans la section précédente, nous avons adopté, pour l'analyse du comportement système, une vision système où tous les objets d'un scenario sont analysés, pour en déduire un état scénario ou système. Ensuite on analyse l'impact des interactions sur les états du système.

Dans le contexte de notre travail, et pour la génération d'un prototype d'interface, l'état est définit par une liste des états d'objets manipulés par le système. L'approche adoptée supporte aussi la vision système, et permet de résoudre le problème d'entrelacement entre scenarios.

La figure 3.10 met en évidence une spécification résultat *SC* capturant non seulement les scenarios SC_1*(E1,E2, E3, E4, E5)* et SC_2 *(E1, E6, E3, E7, E5)* mais aussi deux autres scenarios *(E1,E2,E3,E7,E5)* et *(E1,E6,E3,E4,E5)*.

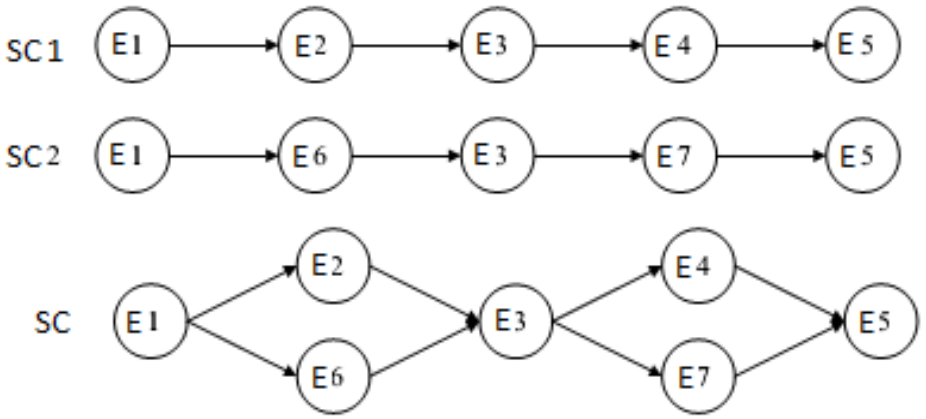
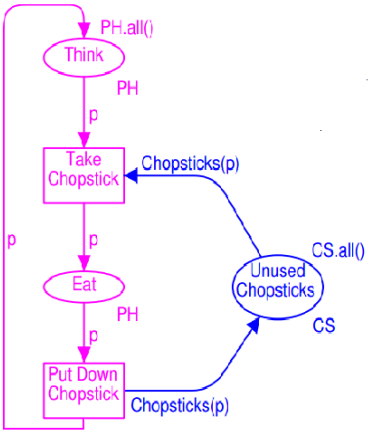


Figure 3.10 Problème d'entrelacement entre scénarios.

La solution au problème d'entrelacement entre scénarios est discuté avec plus de détails dans le chapitre 4 paragraphe 4.2.3.4.

Chapitre 4

Génération du prototype d'interface usager pour le fichier IDD du programme EnergyPlus en utilisant UML et les Réseaux colorés de Pétri



Basé sur les techniques des besoins des utilisateurs, cette étude suggère une nouvelle approche pour la génération d'un prototype d'interface usager pour les données d'entrée du fichier IDD du programme EnergyPlus. Cette approche nécessite le langage de modélisation unifié (UML) et les spécifications des scénarios pour analyser les interactions des utilisateurs avec le programme. Au niveau des spécifications de l'interface usager, le programmeur peut dessiner l'interface, mais il est difficile de préciser leur comportement dynamique. Les réseaux colorés de Pétri sont utilisés dans la spécification et la vérification du comportement du prototype de l'interface avant sa mise en œuvre. Le fichier de définition de données (IDD) est fourni à titre d'exemple pour illustrer le processus de génération d'un prototype d'interface exécutable qui peut être utilisé pour la validation des scénarios, et qui peut évoluer vers l'outil d'interface cible. Le résultat de l'approche est une spécification constituée d'un réseau coloré de Pétri global et qui peut être analysé au moyen d'une simulation avec le prototype d'interface usager généré et raffiné.

4.1. Introduction

EnergyPlus, parmi de nombreux programmes de simulation des performances énergétiques de bâtiment et les installations de CVC associées, a été délibérément conçu comme un moteur de simulation, sans interface utilisateur graphique conviviale particulière [Crawley et al. 99], [Crawley et al. 2001], [Fischer et al. 99]. Le programme lit les données d'entrée et de sortie, sous forme de fichiers texte. Une étude universitaire faite sur les programmes de simulation énergétique des bâtiments [Olsen 2002], stipule que les données d'entrée du programme EnergyPlus sont encore assez laborieuse, et le programme ne sera pas adoptée pour un usage général sans de véritables interfaces d'entrée de donnés.

Un certain nombre de travaux publiés sur la conception d'interface pour la construction d'outils de simulation des performances énergétiques[Edwin et al. 2002] se sont mis d'accord sur la nécessité de développer des interfaces usagers capables de gérer efficacement les besoins des utilisateurs , par exemple, la réduction du temps d'entrée des données qui définissent un bâtiment et ses systèmes[Barry et al. 2005] , ou par la création de nouvelles interfaces usagers avec des fonctions complètes pour des recherches rapides de systèmes CVC [Junjie et al. 2007], etc. En outre, des études comparatives [Shady et al. 2009] indiquent que la plupart des utilisateurs, du secteur privé ou publique, qui font usage des outils de simulation dans la pratique sont beaucoup plus préoccupés par (1) la facilité d'utilisation de l'information au niveau des interfaces et (2) l'intégration des connaissances sous forme de conception intelligente.

Les langages de programmation orientée objet (POO), tels que Visual Basic, Visual C / C + +, Python, Java etc., sont les outils de développement proposés par la plupart des études dans le développement des interfaces usagers pour le programme EnergyPlus. Deux interfaces *EPlusInterface, EasyEnergyPlus*, par exemple, sont développés en utilisant exclusivement deux langages de programmation orientée objet différentes : Python et Visual Basic 6.0. Tous ces langages ne permettent pas la génération d'un processus d'ingénierie des exigences utilisateur, cependant, ils peuvent être utilisés pour écrire le code de l'application.

Jusqu'à présent, aucun travail sur le développement d'un processus de l'ingénierie des besoins des utilisateurs, basée sur la notation UML et les spécifications sous forme de scénarios, n'a été suggéré dans le domaine de la conception d''interface usager pour le programme EnergyPlus. Ce processus, s'il est appliqué, permettra le développement de prototype d'interface usager avec des avantages supplémentaires qui sont comme suit:

- Maintenir un haut niveau de communication entre les éléments d'une équipe de développeurs en proposant une plate-forme de modélisation indépendante.
- Réduire le temps de développement de l'interface usager.
- Améliorer et accroître la participation des utilisateurs par la simulation et le prototypage d'interface.
- Décrire les comportements de l'interface comme une méthode formelle qui renforcera la confiance dans l'exactitude des interfaces par le raffinement, et les tests.

Dans ce chapitre, nous décrirons la vision système et la formalisation des scénarios à l'aide des réseaux colorés de Pétri. Ensuite nous aborderons la génération du prototype de l'interface usager à partir des spécifications comportementales. Ainsi, nous aurons décrit l'approche qui décrit une spécification globale sous forme de RdPs et un prototype d'interface dédier à la génération d'objets pour le fichier IDD.

4.2. vision système et formalisation des scénarios à l'aide des RdPs

Dans ce chapitre, nous allons présenter l'approche pour la modélisation d'un système interactif qui utilise les réseaux de Pétri à haut niveau comme formalisme de spécification. Cette approche propose un processus d'ingénierie des exigences permettant de produire un prototype de l'interface usager du système, ainsi que la spécification du comportement global du système à partir des scénarios.

Tout processus respectable d'ingénierie des besoins appliqué aux systèmes interactifs devra produire une spécification du comportement du système pour des besoins de validation et de vérification, et également un prototype de l'interface usager pour une évaluation avec les utilisateurs finaux. Notre objectif est de supporter toutes les activités de ce processus par l'utilisation de techniques, et d'outils disponibles.

Vu leur adéquation à l'acquisition des besoins, les scénarios sont utilisés comme moyen de description du comportement du système. Étant des descriptions partielles, les scénarios nécessitent une opération d'intégration pour produire le comportement global du système. Pour garder trace des différents scénarios de départ dans la spécification résultante, nous avons pensé à formaliser chaque scénario par un RdP coloré (couleur distincte attribuée à chaque scénario du système). Pour des raisons de modularité, nous nous sommes intéressés séparément aux niveaux cas d'utilisation (CU) et scénario. Nous produisons un premier RdP qui modélise le niveau CU, et plusieurs autres RdPs qui raffinent les CUs du premier RdP.

Pour la production du prototype de l'IU, nous avons étudié la possibilité de pouvoir le générer à partir de la spécification globale du système. L'opération de génération sera décrite en détail dans la section 4.3.

Pour la manipulation des RdPs (création, simulation et vérification) notre choix s'est porté sur un outil ayant les caractéristiques suivantes :

- Support des couleurs, vu qu'on attribue à chaque scénario une couleur qui le distingue des autres.
- Support de la hiérarchisation pour pouvoir lier les deux niveaux de modélisation que nous avons considérés séparément.
- Support d'un éditeur graphique pour l'édition, la simulation et la vérification de certaines propriétés des RdPs.
- Disponibilité gratuite.

Après avoir effectué une recherche des différents outils répondant aux critères

précités en se basant sur des classifications existantes [Pétri-Net 99, Wolf 99], nous avons retenu l'outil CPNTools [Kristensen et al. 2004] dont la distribution est gérée par l'Université d'Aarhus au Danemark. Cet outil supporte essentiellement la théorie des RdPs à haut niveau telle que définie par Jensen [Jensen 95]. CPNTools, qui est actuellement disponible sur des plates-formes Windows, Mac OS, Linux et Unix, offre les caractéristiques intéressantes suivantes outre sa gratuité et son support des aspects couleur et hiérarchie :

- Il inclut le langage de marquage standard (ML) pour la définition des couleurs, des conditions de garde au niveau des transitions et des fonctions de transformation de jetons au niveau des arcs. ML est fortement typé, et il comprend plusieurs types et structures de données (liste, ensemble, lot, etc.), ce qui offre plus d'expressivité lors de la description des RdPs.

- Il offre un simulateur intelligent qui permet de tester automatiquement ou interactivement des spécifications en RdPs.

- Il supporte un outil d'analyse de la structure des RdPs permettant de vérifier la majorité des propriétés comportementales d'un modèle. Dans la version actuelle, la vérification est basée sur la technique du graphe d'accessibilité. Une technique alternative, celle des invariants, est pour le moment parmi les travaux futurs.

- Il inclut un outil d'évaluation de performance des systèmes. En effet l'outil sert aussi à modéliser des RdPs temporisés où la composante temps peut être associée soit aux transitions soit aux arcs sous forme d'une valeur, d'un intervalle ou même d'une fonction aléatoire.

Dans le reste de cette section, nous donnerons la définition des RdPs à haut niveau que nous avons utilisé dans ce travail. Nous présenterons ensuite un aperçu du processus d'ingénierie des besoins basé sur les RdPs. ensuite nous détaillerons les différentes étapes de ce processus.

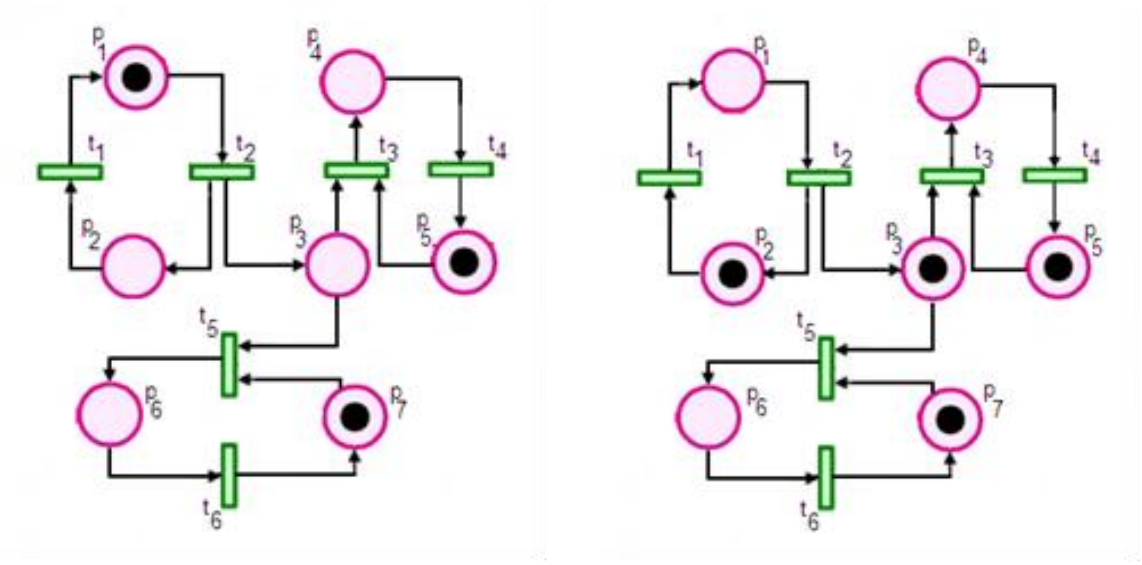
4.2.1. Réseaux de Pétri à haut niveau

La théorie des réseaux de Pétri (PN) a été développée par les travaux de Carl Adam Pétri et beaucoup d'autres dans les 60'ies et les 70'ies , et ils furent bientôt reconnus comme étant l'une des langues les plus adéquates de modélisation mathématique pour la description et l'analyse de synchronisation , de communication et des ressources entre des processus concurrents.

Les notions de base de la théorie des PNs pourraient être bien expliquées à travers les illustrations suivantes des figures 4.1, 4.2.

La représentation picturale des PNs comme un graphe contient deux types de nœuds, les cercles ou ellipses (dits *places*) et des bars ou des rectangles (appelés *transitions*). Ces nœuds, des places et des transitions, sont reliés par des *arcs* dirigés à partir des *places* vers les *transitions* et de *transitions* vers les *lieux*.

Si un arc est dirigé à partir du nœud i vers un nœud j (soit à partir d'une *place* à une *transition* ou d'une *transition* vers une *place*), i est une entrée à j, et j est une sortie de i. Dans la figure 4.1(a), par exemple, la *place* p_1 est une entrée à la *transition* t_2, tandis que P_2 et P_3 sont des *places* de sorties pour la transition t_2.



(a) (b)

Figure 4.1 Réseau de Pétri : (a) t_2 est activée, (b) t_2 n'est pas activée.

L'exécution des PNs est commandée par la position et le mouvement de marqueurs (appelés jetons) dans le réseau. Les Jetons indiqués par des points noirs, réside dans les *places* du réseau. Les jetons sont déplacés par le tir des *transitions* du réseau. Une *transition* doit être activé pour pouvoir tirer (une transition est activé lorsque l'ensemble de ses *places* d'entrée ont des jetons).

La *transition* se déclenche en supprimant les jetons résidents dans les *places* d'entrée et génère de nouveaux jetons qui sont déposés dans les *places* de sortie de la transition. Dans le PN marqué de la figure 4.1(a), la transition t_2 est activée car elle dispose d'un jeton dans sa place d'entrée p_1. La *transition* t_5, d'autre part, n'est pas activée car l'une de ses places d'entrée p_3 n'a pas un jeton. Si la transition t_2 se déclenche du PN marqué, la figure 4.1(b) est obtenue.

Le tir de la *transition* t_2 supprime le jeton de la *place* p_1 et met de nouveaux jetons dans les *places* de sorties p_2 et p_3, La répartition des jetons dans un PN marqué définit l'état du réseau et est appelée son marquage. Le marquage peut changer à la suite des tirs des transitions. Dans différents marquages, différentes transitions peuvent être activées. Par exemple, dans le PN marqué de la figure 4.1(b), trois *transitions* sont activés : t_1, t_3, t_5, et dont aucune n'a été activée dans le marquage de la figure 4.1(a).

En plus de propriétés statiques représentées par le graphique, les réseaux de Pétri ont des propriétés dynamiques qui découlent de son exécution. La figure 4.2, illustre trois propriétés utiles dans les systèmes de modélisation et est brièvement discutée dans ce qui suit :

- L'exécution Séquentielle : la transition t_2 du PN maqué peut se déclenchée que seulement après le tir de t_1. Cela impose la primauté de contraintes : t_2 après t_1, la figure 4.2 (a).
- La Synchronisation : la transition t_1 sera activée seulement s'il y a au moins un jeton à chacune de ses places d'entrée, la figure 4.2 (b).

- La Concurrence : les transitions t_1 et t_2 sont exécutées simultanément dans le temps, la figure 4.2 (c). Avec cette propriété, les réseaux de Pétri sont capables de modéliser des systèmes de contrôle distribués avec de multiples processus.

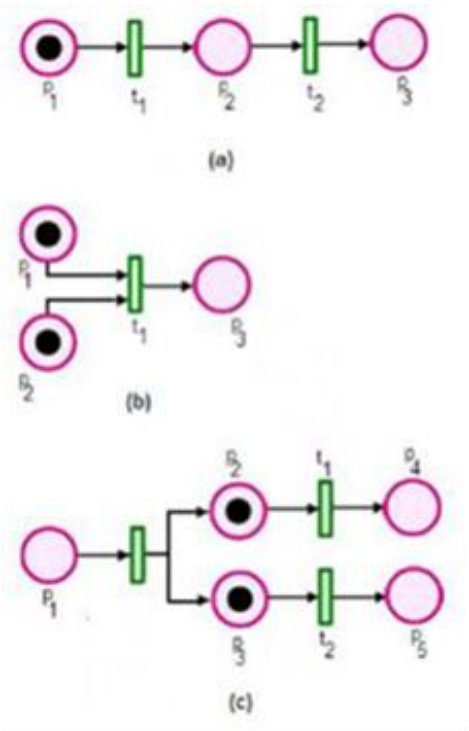


Figure 4.2 PNs marqués avec : (a) exécution séquentielle, (b) exécution en synchronisation, (c) exécution en concurrence

Toute extension aux Réseaux de Pétri de base (RdP) définit un RdP à haut niveau. Les RdPs colorés, à prédicats, temporels, hiérarchiques, stochastiques sont des exemples de telles extensions. Dans notre cas, cette extension concerne le support de la hiérarchisation et de l'aspect couleur (différents types de jetons).
L'aspect hiérarchique améliore la modularité des RdPs spécifiés. Il permet aux

analystes de construire des RdPCs de grande taille par la combinaison d'un nombre de RdPCs de petite taille. Il est toujours possible de passer d'un RdPC hiérarchique à un RdPC non-hiérarchique (mise à plat). Jensen [Jensen 95] définit un RdPC hiérarchique comme un ensemble de RdPCs (pages) liées par la substitution des transitions et/ou les places de fusion. La substitution d'une transition permet de raffiner une transition d'un RdPC par un autre RdPC. Les places de fusion permettent de spécifier que différentes places sur différentes pages sont identiques.

Dans ce travail, nous avons utilisé la substitution des transitions pour raffiner les comportements des cas d'utilisation (UCs) du système.

4.2.2. Langage Standard ML (SML)

Standard ML (SML) est un langage de programmation généraliste, modulaire, fonctionnel. Il est doté d'un système de typage statique fort par inférence de types.
SML descend directement du langage ML. Les inscriptions sur les RdPCs sont écrites avec le langage de programmation CPN ML qui est une extension du langage de programmation ML.

Les Types, les expressions sur les arcs et les gardes sur les transitions sont indiquées en utilisant le langage CPN ML qui est fortement typé. Les types de données peut être atomique (entier, chaîne de caractères, réel, booléen, énumération), et structuré (produits, records, unions, listes, sous-ensembles).

Les expressions sur les arcs, qui sont des inscriptions textuelles positionnés à côté des arcs individuels, sont construites à partir de variables typées, de constantes, d'opérateurs et de fonctions. Lorsque toutes les variables dans une expression sont liées à des valeurs (de type correct) l'expression peut être évaluée.

L'expression à l'arc évalue un ensemble multiple de jetons colorés (valeurs des données). L'exemple de la figure 4.3 qui représente le réseau de Pétri coloré (section *4.2*), qui est une adaptation du logiciel CPNTools [CPNTools 2006] du diagramme de cas d'utilisation proposée (figure 4.6). Considérons l'expression *ae* sur l'arc reliant à la fois la transition *GenerateObject* à la place *Sc_g*. Il contient la variable *ae* déclarée comme :

Var ae : l ;
où l, est déclaré comme un jeu de couleurs de type liste,
colset l = liste SC, où SC est un jeu de couleurs de type énumération,
colset SC = avec rc | ec ;

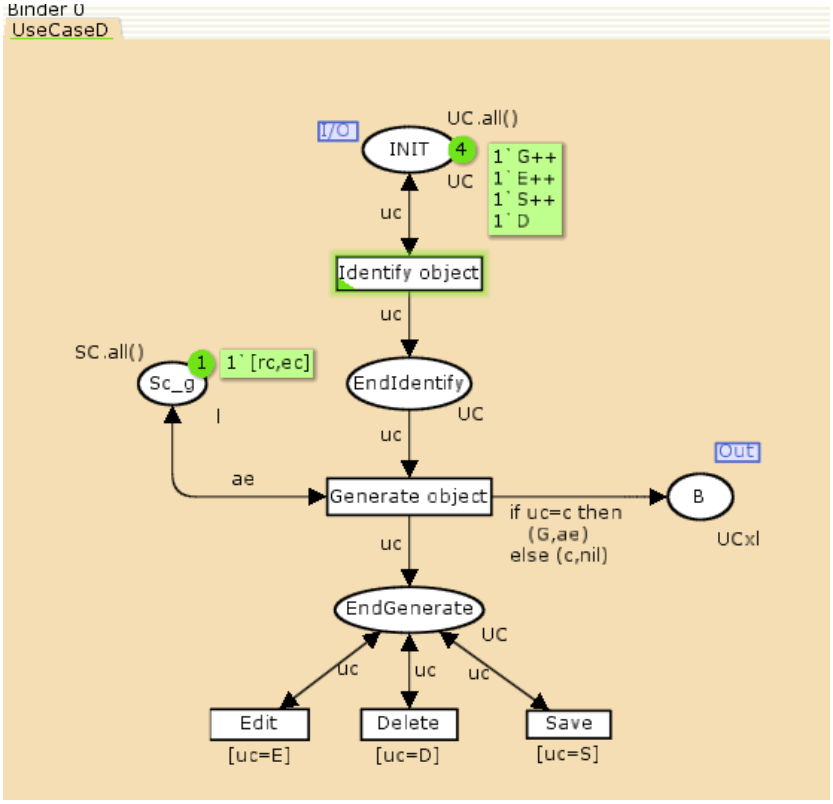


Figure 4.3 RdPC du cas d'utilisation adapté à l'outil CPN Tools.

À côté de chaque place, il y a une inscription qui détermine l'ensemble des couleurs des jetons que les places sont autorisées à avoir. L'ensemble des jetons colorés est spécifié à l'aide d'un type et il est appelé l'ensemble des jetons colorés de la *place.* Par convention, l'ensemble des jetons colorés est écrit en dessous de la *place*. Dans la figure 4.3, les places : *Init, EndIdentify, et EndGenerate* l'ensemble des jetons colorés *UC* est défini dans le langage de programmation CPN ML de type énumération :

Colset UC = avec G | E | S | D;

Cela signifie que les jetons résidant sur les trois *places* aura un type énumération comme leur jetons colorés. L'ensemble des jetons colorés *UC* est utilisée pour modéliser les états possibles du RdPC de la figure 4.3. Une place et une *transition* peuvent également être reliées des arcs à double-tête. Un arc à deux têtes est un raccourcie pour deux arcs dirigés dans des directions opposées entre une *place* et une *transition* qui ont toutes deux l'expression du même arc. Cela implique que la *place* est à la fois une place d'entrée et une place de sortie de la transition. Par exemple, la transition *IdentifyObject* et le place *Init* sont reliés par deux têtes du même arc (figure 4.3).

Le système actuel de marquage de chaque *place* est indiqué à côté de celle-ci. Le nombre de jetons sur une *place* dans le marquage courant est indiqué dans le petit cercle, tandis que les jetons colorés sont détaillés et indiqués dans la case placée à côté de petit cercle. Initialement, le marquage courant est égale au premier

marquage, notée M_0. Dans l'exemple de la figure 3.4, le marquage initial à quatre jetons sur la place *Init* et un jeton sur *place Sc_g*.

4.2.3. Processus d'ingénierie des besoins utilisant les RdPCs

La figure 4.4 présente l'ensemble des activités du processus d'ingénierie des besoins de l'approche proposé. Ce processus raffine et détail celui de la figure 1.3 (Chapitre 1, § 3) dont le but est de combiner dans le même processus les techniques de prototypage et les spécifications formelles.

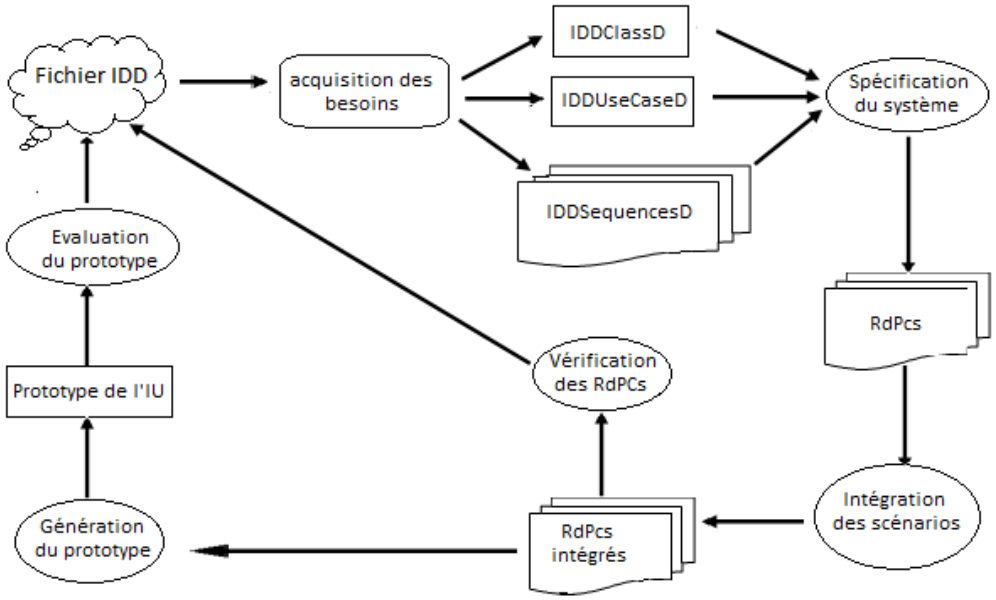


Figure 4.4 Activité du processus d'ingénierie des besoins.

Dans l'activité d'acquisition des besoins, l'analyse élabore en premier lieu le diagramme de classe *IDDClassD* pour modéliser l'aspect statique du système. L'aspect dynamique du système est modélisé par le diagramme des cas d'utilisation *IDDUseCaseD* et l'ensemble des diagrammes de séquences *IDDSequenceDs* décrivant les différents cas d'utilisation (CUs*)* du fichier IDD (notre système). L'activité de spécification du système consiste à dériver un ensemble de RdPCs à partir des diagrammes précédemment acquis (IDDUseCaseD et IDDSequenceDs). Les RdPCs qui correspondent au même CU sont fusionnés dans l'activité d'intégration des scénarios. Les RdPCs intégrés seront les entrées des activités de vérification des RdPCs et de génération du prototype de l'IU. Le prototype généré sera ensuite exécuté et évalué par les utilisateurs finaux - développeurs de modules pour le programme EnergyPlus - dans l'activité d'évaluation du prototype. En cas de détection d'incohérences ou d'incomplétudes lors des activités de vérification et/ou d'évaluation, l'analyse apporte les modifications nécessaires aux scénarios acquis.

Dans ce travail, nous nous concentrons sur l'activité de génération du prototype d'interface usager, qui est, la transformation représentée par les flèches en gras de la figure 1.3. Ce processus peut être décomposé en cinq activités :

- Acquisition des besoins (section 4.2.3.1) : élabore dans un premier temps le diagramme de classe pour modéliser l'aspect statique du système
- Acquisition de scénarios (section 4.2.3.2) : un diagramme de cas d'utilisation est précisé, et pour chaque cas d'utilisation, un certain nombre de diagrammes de séquence annotés avec les informations d'interface usager.

- Génération des spécifications (section 4.2.3.3) : cette activité consiste à dériver des RdPCs à partir du diagramme de cas d'utilisation acquis.

- intégration des scénarios (section 4.2.3.4) : les RdPCs correspondant au même cas d'utilisation sont combinés pour obtenir un système intégré de réseaux colorés de Pétri des cas d'utilisation.

- Génération du prototype d'interface du fichier IDD (section 4.2.3.5) : dans cette activité, le RdPC intégrée servira d'entrée pour le processus de génération du prototype d'IU.

4.2.3.1. Acquisition des besoins

Dans cette activité, une analyse de la structure du fichier IDD permet de recueillir les informations nécessaires sur le système à concevoir. Un diagramme de classe *IDDClasseD* a été élaboré dans un premier temps pour modéliser l'aspect statique du système.

Dans cette section, nous détaillerons la construction du diagramme de classes de la structure du fichier IDD [Bachkhaznadji et al. 2006], où les classes, les attributs, etc., sont identifiés. La figure 4.5 illustre le diagramme du modèle de classes proposé pour le fichier IDD. Pour la construction des objets (classes), nous avons adopté l'approche GOA (Goal Oriented Approach) [Liang 2003]. Le principe de cette approche consiste à identifier les classes par rapport aux objectifs assignés par les cas d'utilisation plutôt que d'utiliser des descriptions de cas en utilisant le procédé piloté par le cas d'utilisation [Graham 95].

Dans notre développement, les classes, dans le diagramme de classes, sont les entités qui participent à la réalisation des objectifs permettant à un utilisateur d'interagir avec les classes d'entrée décrites dans le fichier de structure IDD. Ils ont leurs propres caractéristiques et peuvent collaborer avec les cas d'utilisation. Le diagramme de classes montre les classes et les relations entre elles. Dans l'exemple de la figure 4.5, les types de relations sont des associations. Quatre classes ont été identifiées pour la structure du fichier IDD: une classe de groupe (*ObjectsGroup*), une classe d'entrée d'objet (*InputObject*), deux autres classes pour identifier un champ numérique (*NumField*) et un champ alphanumériques (*AlphaField*). Les deux

autres classes, dans la figure 4.5, identifient une classe (*Actor)* (développeur de module) et une classe interface (*ICS*).

La relation *association* montre les relations qui peuvent existées entre les instances des deux classes: *ObjectsGroup* et *InputObject*. Une multiplicité (1 à 1..*) est ajouté aux deux extrémités de la liaison pour pointer vers le rôle de chaque classe par rapport à l'autre. Dans le diagramme de classe (IDDClassD), une instance de la classe *ObjectsGroup* peut posséder au moins une instance de la classe *InputObject*.

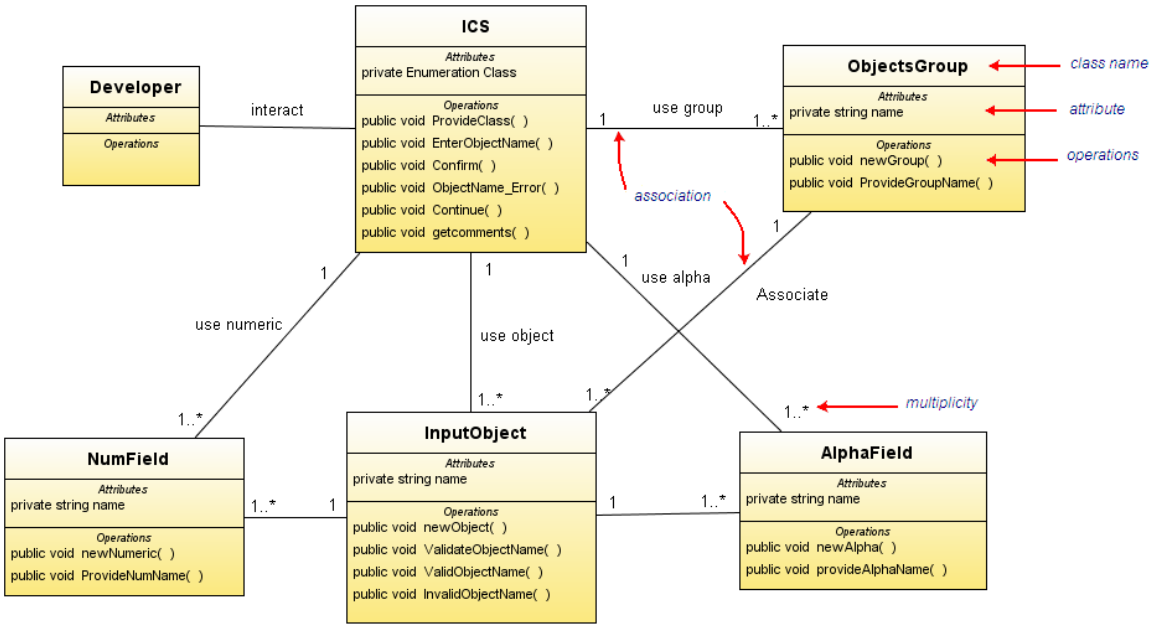


Figure 4.5 Diagramme de Classe IDDClassD du fichier IDD.

4.2.3.2. Acquisition des scénarios

Nous avons commencé par élaborer le diagramme des cas d'utilisation *IDDUseCaseD* (figure 4.6) qui capture les fonctionnalités du système. Et pour chaque cas d'utilisation du *IDDUseCaseD, on a* élaboré également les différents diagrammes de séquences *IDDSequenceD* correspondants aux scénarios qui le décrivent.

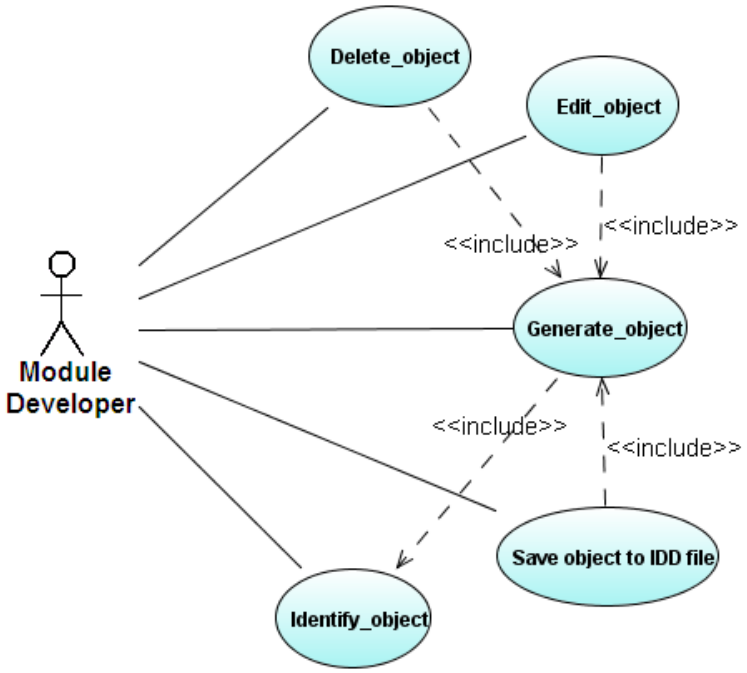
Le diagramme *IDDUseCaseD* décrit les interactions entre les acteurs externes et le système modélisé. Il décrit une séquence d'actions menées par un système dans le but d'offrir un service aux acteurs. Un cas d'utilisation est un résumé des scénarios pour une tâche ou un but simple. Un acteur (l'utilisateur) est celui qui déclenche les événements impliqués dans cette tâche. Les cas d'utilisation sont représentés par des ellipses, et les acteurs sont représentés sous forme d'icônes reliées par des lignes solides avec les cas d'utilisation avec lesquelles ils interagissent.

Un cas d'utilisation peut faire appel aux services d'un autre cas d'utilisation. Une telle relation est appelée une relation *include*, et sa direction n'implique pas l'ordre d'exécution.

Dans le diagramme IDDUseCaseD, Il y a un acteur (c.-à-d.- module développeur) interagissant avec cinq cas d'utilisation de base : *Identify_object* (identification du type d'objet), *Generate_object* (création de l'objet), *Edit_object* (éditer l'objet),

Delete_object (suppression de l'objet), et *Save_object* (sauvegarde de l'objet) dans le fichier IDD. Le cas d'utilisation *Generate_object*, fait appel au service du cas d'utilisation *Identify_object*, et l'inclus en tant que sous-tâche.

Une identification d'une classe d'objet d'entrée peut être un groupe d'objets d'entrée connexes, ou un objet d'entrée, ou un champ de donnée (de type numérique ou alpha). En outre, UML comprend la relation *Extend (*étendre), ce qui peut être considéré comme une variation de la relation *Include*, ainsi que les relations de généralisation qui indique qu'un cas d'utilisation est un type spécial d'un autre cas d'utilisation [Rumbaugh et al. 99]. Le diagramme de cas d'utilisation est utile pour visualiser le contexte de notre domaine d'application et les limites de l'ensemble du comportement de l'interface. Une exécution d'un cas d'utilisation sera généralement caractérisée par de multiples scénarios.



4.6 Diagramme des cas d'utilisation du fichier IDD (IDDUseCaseD).

Un scénario est une instance d'un cas d'utilisation. Il décrit un groupe d'interactions entre les acteurs et les objets des systèmes. En UML, le scénario peut être représenté sous la forme de diagrammes de collaboration et / ou des diagrammes de séquence. Les deux types de diagrammes reposent sur la même sémantique

sous-jacente, et la conversion de l'un à l'autre est possible [Glinz 2002].

Dans notre travail, nous avons choisi d'utiliser le diagramme de séquence pour leur simplicité. Un diagramme de séquence montre les interactions entre les objets participant à un scénario dans l'ordre temporel. Il représente les objets par leurs lignes de vie et montre les messages qu'ils échangent dans la séquence temporelle. Cependant, il ne tient pas compte des associations entre les objets. Un diagramme de séquence comporte deux dimensions : la dimension verticale représente le temps, et la dimension horizontale représente les objets. Les messages sont affichés en tant horizontales flèches solides de la ligne de vie de l'expéditeur objet à la ligne de vie du récepteur objet.

La figure 4.7 représente deux diagrammes de séquence (deux scénarios) du cas d'utilisation *Generate_object*. La figure 4.7(a) représente un scénario où le l'utilisateur à correctement saisie un nom d'entrée pour un objet de classe (*regularGeneration*), tandis que la figure 4.7(b) illustre le cas où un nom d'entrée incorrecte pour l'objet de classe (errorGeneration).

Les deux scénarios sont basés sur le guide de référence d'entrée/sortie du programme EnergyPlus (EnergyPlus, v2.0 ou plus). Ainsi, la définition de données d'entrée dans le fichier IDD, les règles suivantes s'imposent:

- Un nom de classe doit être unique.
- La longueur maximale d'un nom de classe est de 100 caractères.
- Ce n'est pas un nom vide.

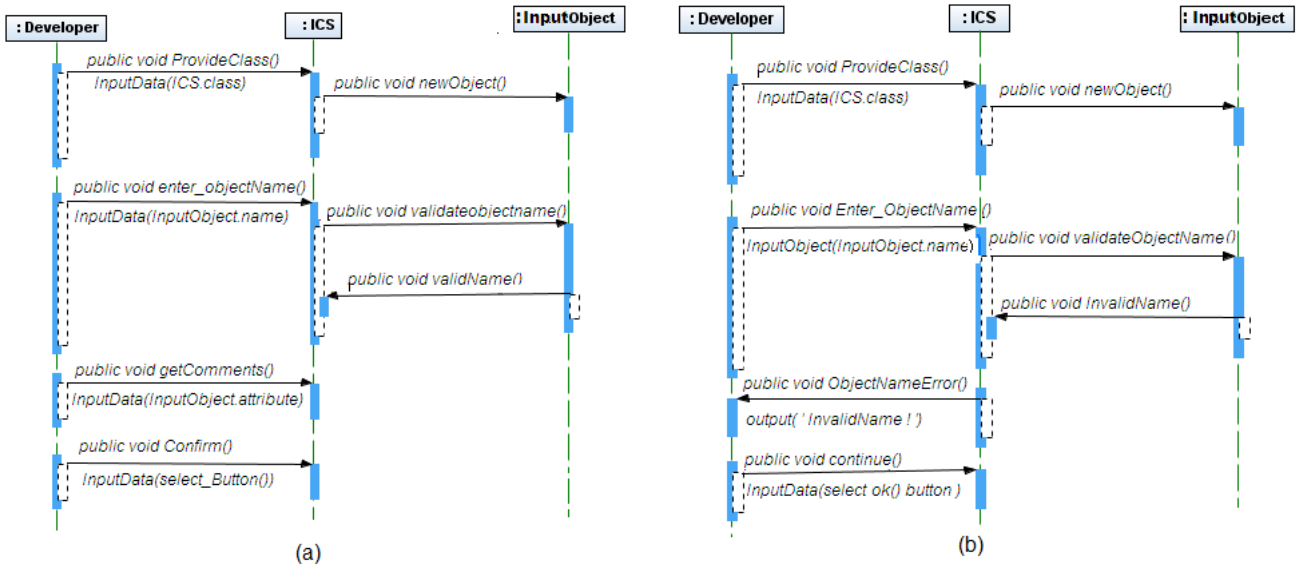


Figure 4.7 Diagramme de séquence (IDDSequenceD) : (a) génération correcte, (b) génération incorrecte.

En outre, un message dans notre travail est gardé par une contrainte conditionnelle et est enrichi avec les informations d'interface utilisateur. Au-delà des contraintes UML les messages standards trouvés dans le diagramme de séquence [Rumbaugh et al 99], deux contraintes supplémentaires *InputData (données d'entrée)* et *OutputData (données de sortie)* sont définis. La contrainte *InputData* indique que le message correspondant contient des informations d'entrée de l'utilisateur (*actor*). La contrainte *OutputData* précise que le message correspondant comporte des informations pour l'affichage. Les deux contraintes *InputData et OutputData* ont un paramètre qui indique le type de l'action de l'utilisateur. Il peut être soit un nom de méthode (une opération de classe), ou un ou plusieurs attributs de classe etc., voir figures 4.5 et 4.7.

Une fois ces contraintes InputData et OutputData sont spécifiés, cette information est utilisée pour déterminer les widgets correspondants qui apparaissent dans le prototype d'interface. La création des Widgets est basée sur la recommandation et la terminologie trouvée dans IBM 1991[IBM 91] et qui comprennent les huit points suivants :

- (WG_1) Un widget compatible TextField est généré en cas d'une contrainte *InputData* avec une dépendance à un attribut de type string (caractère), réel, entier, voir figure 4.5 et figure 4.7(a) :

EnterObjectName () {InputData(InputObject.name)}

- (WG_2) Un groupe de widgets de boutons radio dans le cas d'une contrainte *InputData* avec la dépendance à un attribut de type énumération ayant une taille inférieure ou égale à 6, voir figures 4.5 et 4.7(a) :

ProvideClass () {InputData(ICS.class)}

- (WG_3) Un widget de type *Liste* est activé et générée en cas de contrainte *InputData* ayant des attributs de type énumération dont le nombre est supérieure à 6 ou avec des attributs de type *Collection*.

- (WG_4) Un widget de type *Tableau* activé est générée avec une contrainte *InputData* possédant plusieurs attributs dépendants.

- (WG_5) Un widget de type *TextField* désactivé est généré pour une contrainte de sortie *OutputData* avec une dépendance à un attribut de type string, réel ou entier.

- (WG_6) Un widget de type *label* est généré pour une contrainte *OutputData* sans attribut dépendent, par exemple, dans les figures 4.5 et 4.7(b) :

ObjectNameError () {OutputData (« nom de classe incorrecte ! ")}

- (WG_7) Un widget de type Liste désactivé est généré en cas de contrainte *OutputData* avec des attributs dépendants de type *énumération* ayant une taille supérieure à 6 ou avec des attributs dépendants de type *collection*.

- (WG_8) Un widget de type *Tableau* désactivé est générée avec une contrainte *OutputData* avec plusieurs attributs dépendants.

4.2.3.3. Génération des spécifications du système

Cette activité consiste à dériver un Réseau de Pétri (RdP) directement du diagramme des cas d'utilisation *IDDUseCaseD* acquis (figure 4.6) et tous les diagrammes de séquence IDDSequenceD (figure 4.7). Ces dérivations seront expliquées dans les paragraphes suivants : spécifications des cas d'utilisation et la spécification des scénarios.

4.2.3.3.1 Spécification des cas d'utilisation

Le réseau de Pétri correspondant pour le diagramme de cas d'utilisation est obtenue en transformant les cas d'utilisation en *places*, voir figure 4.8. Les transitions menant à ces places sont gardées par les conditions d'initialisation des Cas d'utilisations (CUs) par l'utilisateur. La place *Init* est ajoutée pour modéliser l'état initial de notre système avant toute exécution des différents CUs. Nous supposons qu'après exécution d'un CU, le système revient à son état initial. La place *Init* peut contenir plusieurs jetons pour modéliser les exécutions parallèles de plusieurs CUs et de plusieurs copies du même CU. La figure 4.8 illustre le RdP dérivé du diagramme de cas d'utilisation IDDUseCaseD.

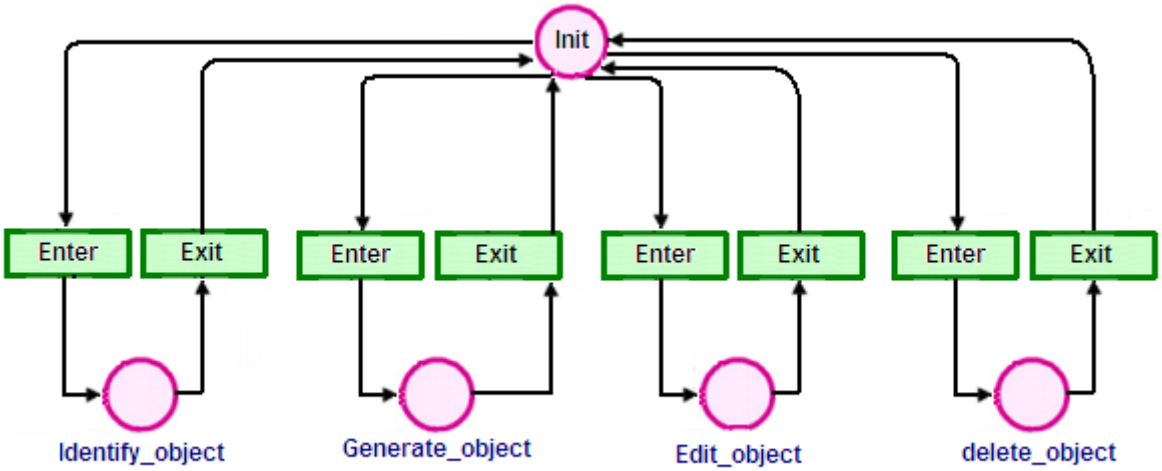


Figure 4.8 RdP généré à partir du diagramme IDDUseCaseD de la figure 4.6.

Dans un diagramme de cas d'utilisation IDDUseCaseD, un cas d'utilisation peut faire appel aux services d'un autre par le biais de la relation i*nclude*. Cette relation peut avoir plusieurs significations en fonction du système modélisé. Considérons deux cas d'utilisation: UC_1 et UC_2, figure 4.9: la relation *include* entre eux peut être interprétée de différentes manières [Elkoutbi 98]. La figure 4.9(a) donne la forme générale de cette relation, UC_1 peut être décomposé en trois sous cas d'utilisation : UC_{11} représente la partie de UC_1 exécuté avant l'appel de UC_2 ; UC_{12} est la partie exécutée en même temps que UC_2 et UC_{13} est la partie exécutée après la fin de UC_2 (synchronisation).

Il est possible que deux de ces trois cas d'utilisation sont vides , ce qui entraîne l'un des types de configuration illustré par la figure 4.9(b) , la figure 4.9(c) , la figure 4.9(d) , la figure 4.9(e) , la figure 4.9(f) , et la figure 4.9(g).

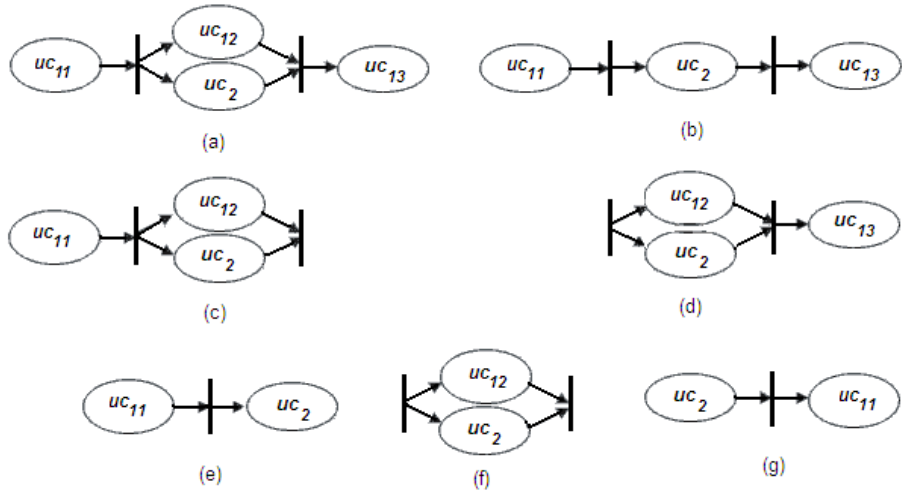


Figure 4.9 Raffinement de la relation *include.*

La relation de type (g) entre les UC_1 et UC_2 signifie que l'UC_2 précède UC_1, ce qui implique que l'UC_1 n'est accessible qu'à travers UC_2. Donc les transitions liant UC_1 à la place *Init* doivent être déplacées vers la place représentant UC_2.

C'est le cas des relations d'utilisation entre les trois couples de CUs de notre système : (UC_1: *Edit*; UC_2: *Generate_object*); (UC_1: *Save* ; UC_2: *Generate_object*); (UC_1: *Delete*; UC_2: *Generate_object*). Après la prise en considération du type de relation entre les CUs, le RdP de la figure 4.8 est alors mis à jour pour obtenir le RdP de la figure 4.10.

Le logiciel CPNTools que nous avons adopté dans notre travail, permet le raffinement des *transitions*, mais ne supporte pas le raffinement des *places*. Donc, pour substituer les UCs qui représentent les *places* pour le RdPC représentant les scenarios intégrés (section 4.2.3.4), le RdPC obtenue après raffinement de la relation *include* nécessite adaptation : chaque subnet représenté par (Enter $\rightarrow$ $place_i$ $\rightarrow$ Exit) est remplacé par une simple *transition* qui représente l'UC, cf. carré en pointillé (figure 4.10). Des *places* intermédiaires ont été introduites, comme *endIdentify* et *endGenerate (*figure 4.3*)*.

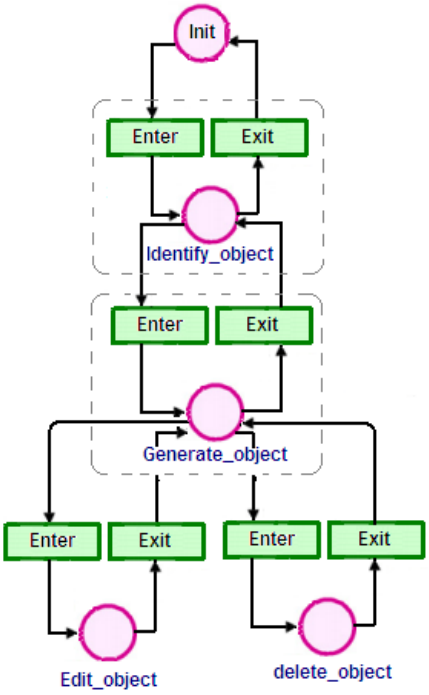


Figure 4.10 RdP mis à jour après raffinement de la relation *include.*

4.2.3.4. Génération des spécifications du système

Pour chaque scénario du cas d'utilisation *Generate_object*, nous construisons une table associée de l'état d'objet. Ce tableau est obtenu directement à partir du diagramme de séquence du scénario en suivant l'échange de messages de haut en bas, et d'identifier les changements dans l'état d'objet causées par les messages.
Dans notre exemple, les tableaux I et II illustrent les états des objets liés aux scénarios *regularGeneration*, et *errorGeneration*, voir figure 4.7.

Dans ces tableaux, un état scénario est représenté par un vecteur d'états des objets qui participent dans le scénario (la colonne état du *scénario* dans les tableaux I et II, respectivement). Dans notre exemple, les objets participants sont les classes *Developer, ICS et InputObject*. A partir de chaque tableau des états des objets, un CPN est généré par chaque état d'un scénario. Les états des scénarios sont remplacés par des places et les messages par des transitions, voir les figures 4.11 et 4.12. Chaque scénario est associé à une couleur distincte, par exemple, *rc* pour le scénario *regularGeneration*, et *ec* pour le scénario *errorGeneration*. Tous les CPNs (scénarios) du même cas d'utilisation recevront la place initiale (état) que nous appelons *B* (voir figure 4.11 et 4.12). La place *B* servira à établir un lien entre le CPN intégré (section 2.3) avec le CPN qui modélise le diagramme du cas d'utilisation *IDDUseCaseD* représenté par la figure 4.3.

TABLEAU I : Etat de l'objet associé au Scenario *RegularGeneration*

Messages d'objet	Developer	ICS	InputObject	Etat du Scenario
Fournir la classe	*Présent*	Sélectionnez l'objet	Vide	S1
Nouvel objet	Présent	Sélectionnez l'objet	objet généré	S2
Entrez le nom de l'objet	Présent	Invité de Nom	objet généré	S3
Valider Nom	Présent	Nom entré	Valider Nom	S4
Nom valide	Présent	Nom entré	Nom valide	S5
Obtenez Commentaires	Présent	Obtenez Commentaires	Nom valide	S6
Confirmer	Présent	Confirmer	Nom valide	S7

S1 = (présent, fournir la classe, vide).
S2= (Présent, fournir la classe, objet généré).
S3 = (Présent, invite de nom, objet généré).
S4 = (Présent, nom entré, valider le nom).
S5 = (Présent, nom entré, nom valide).
S6 = (Présent, obtenez commentaires, nom valide).
S7 = (Présent, confirmer, nom valide).

TABLEAU II : Etat de l'objet associé au Scenario *errorGeneration*

Messages d'objet	Developer	ICS	InputObject	Etat du Scenario
Fournir la classe	Présent	Sélectionnez l'objet	Vide	S1
Nouvel objet	Présent	Sélectionnez l'objet	objet généré	S2
Entrez le nom de l'objet	Présent	Invité de Nom	objet généré	S3
Valider Nom	Présent	Nom entré	Valider Nom	S4
Nom non valide	Présent	Nom entré	Nom invalide	S8
Erreur de nom	Présent	Obtenez Commentaires	Nom invalide	S9
Continuer	Présent	Confirmer	Nom invalide	S10

S1 = (présent, fournir la classe, vide).
S2= (Présent, fournir la classe, objet généré).
S3 = (Présent, invite de nom, objet généré).

S4 = (Présent, nom entré, valider le nom).
S8 = (Présent, nom entré, nom non valide).
S9 = (Présent, erreur de nom, nom non valide).
S10 = (Présent, continuer, nom non valide).

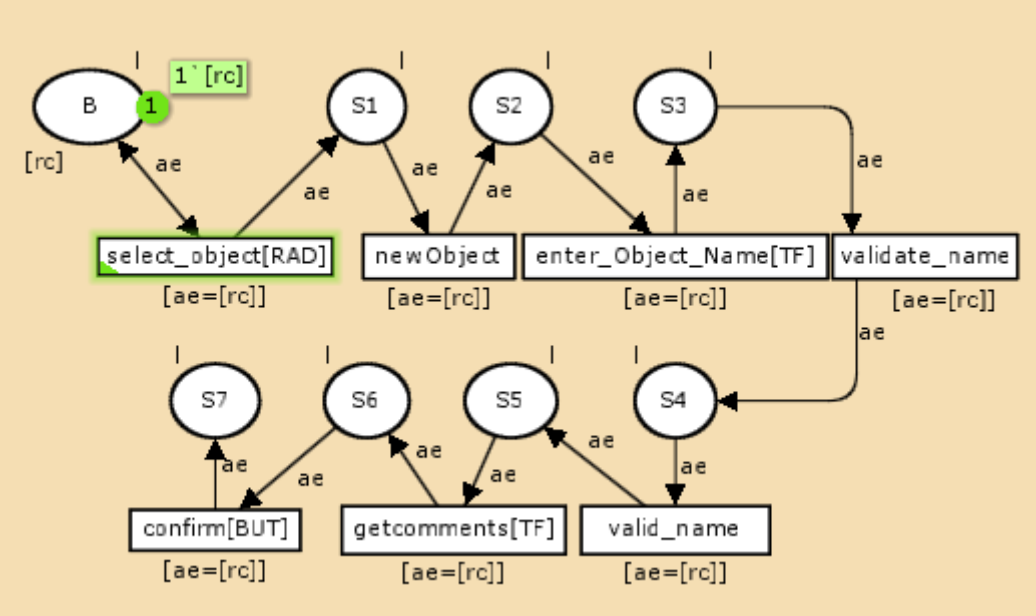


Figure 4.11 CPN du scenario regularGeneration.

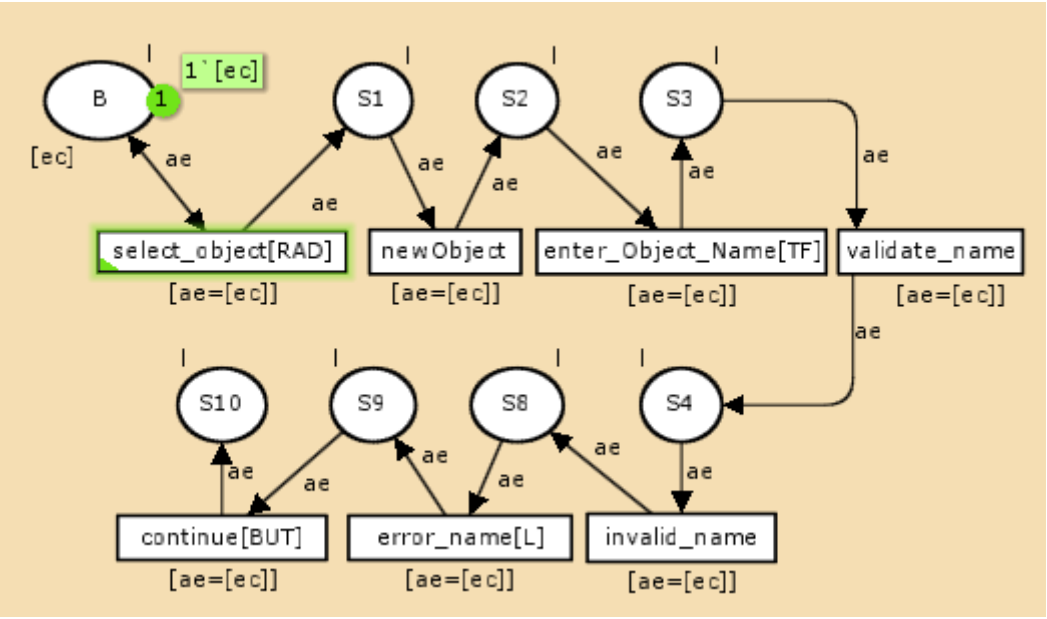


Figure 4.11 CPN du scenario errorGeneration.

4.2.3.4 Intégration des scenarios

Dans cette activité, nous cherchons à combiner toutes les CPNs correspondant au scénario du cas d'utilisation UCi, en vue de produire un CPN intégré qui modélise le comportement du cas d'utilisation. Après avoir intégré les deux scénarios, la place initiale *B* (voir figure 4.13) sera partagée. Jusqu'à ce présent nous ne savons pas quel scénario sera exécuté, et qu'elle couleur de jeton *rc* ou *ec* va être attribué à la place initiale *B*.

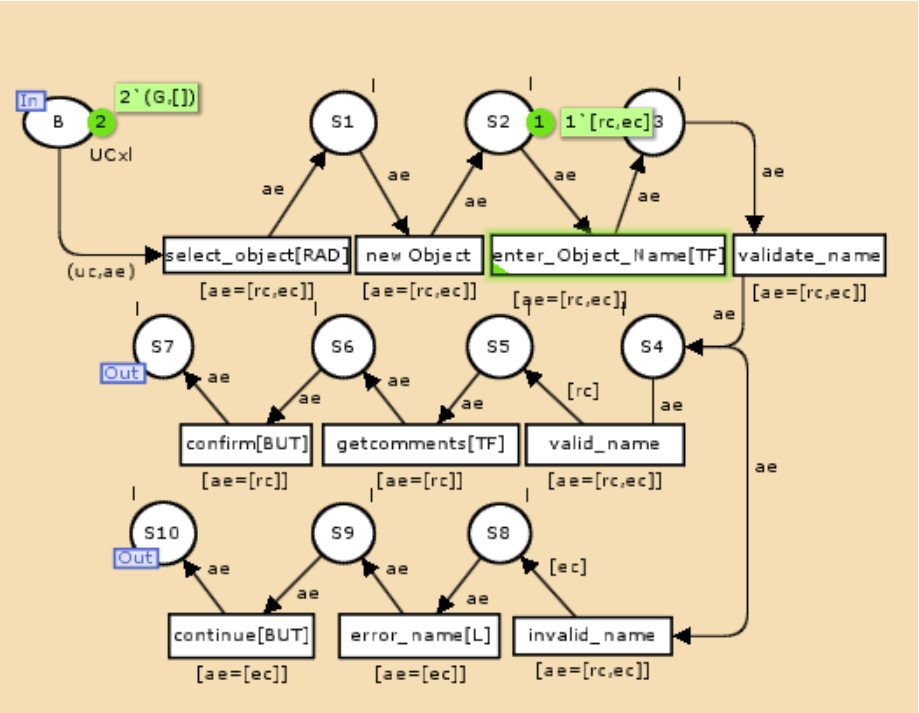


Figure 4.13 CPN des scénarios intégrés : *regularGeneration* et *errorGeneration*.

Pour résoudre le problème d'entrelacement [Elkoutbi 98, Elkoutbi 2000], nous proposons un ensemble de couleurs composite, c'est à dire, une couleur de jeton qui peut prendre plusieurs couleurs. Le logiciel CPN Tools [Jensen] permet d'établir ce type de jeu de couleurs qu'on peut modéliser à partir d'une liste de couleurs. Lors de son passage aux différentes places du CPN intégré, les places visitées seront marquées par l'intersection de ce jeu de couleurs composite.

Lorsque le jeton passe à la place S_4, il conserve le jeu de couleurs [rc, ec] pour la place S_5 sa couleur change pour [rc] et restera inchangée pour le reste de son parcours. Ou il passe de la place S_4 à la place S_8 sa couleur change vers [ec] et restera inchangée pour le reste de son parcours.

Les Transitions qui appartiennent à un seul scénario *regularGeneration* ou *errorGeneration* sera gardé par un jeton coloré des scénarios respectifs, voir les transitions de la figure 4.13 : [*getComments, confirmer*], et les transitions : [*errorName, continue*]. Ce qui n'est pas le cas pour les transitions [*validName et*

invalidName] qui sont nécessaires pour transformer les couleurs à partir du jeu composite de couleurs (liste des couleurs symboliques) à une seule couleur.
Par conséquent, ils doivent être gardés par un jeu de couleurs composite [*rc, ec*]. Pour les transitions qui sont partagées par les deux scénarios *regularGeneration* et *errorGeneration*, ils seront gardés par le jeu de couleurs composite.

Le CPN intégré correspondant au cas d'utilisation peut être reliée au réseau de Pétri coloré généré pour le diagramme de cas d'utilisation de la figure 4.3 par une transition de substitution UseCase (voir figure 4.14), qui est annexé à la place *B* du CPN intégré.

Ce CPN peut être organisé comme un ensemble de modules liés hiérarchiquement. La transition de substitution va transformer les jetons CPN du diagramme de cas d'utilisation de la couleur à un ensemble de couleurs composite du CPN intégré.

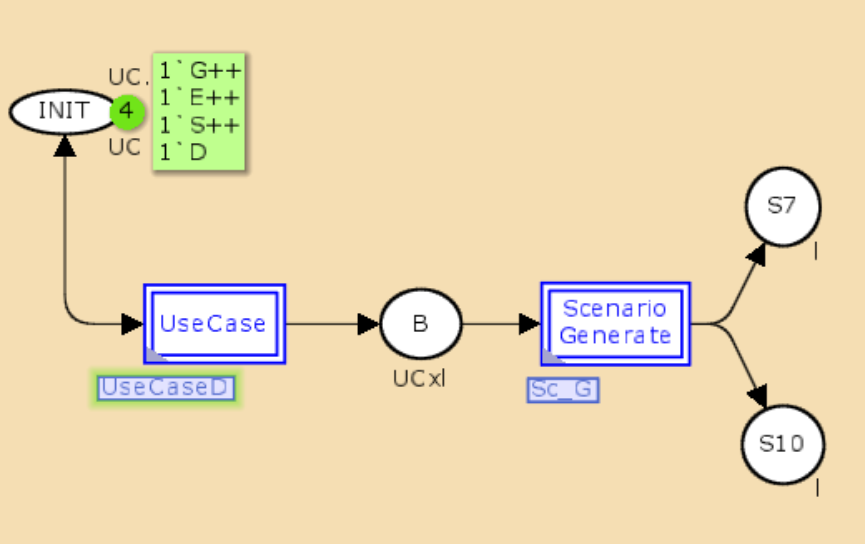


Figure 4.14 CPN de la transition de substitution UseCase.

4.3. Génération du prototype de l'interface usager

La plupart des travaux de génération d'IUs complètes ou de prototypes d'IU sont basés soit sur les données [Foley 93, Balzert 96 96] soit sur l'analyse des tâches [Bodart et al. 94]. Dans notre travail, nous somme orienté vers l'approche de la génération d'IUs guidée principalement par les aspects comportementaux du système. En effet l'IU doit refléter toutes les interactions entre le système et ses utilisateurs. Ces interactions se trouvent généralement décrites en détail lors de la description du comportement du système.

Nous proposons donc de générer le prototype de l'IU à partir des spécifications du comportement du système. Cependant, nous utiliserons également la spécification

des données pour déterminer le type de widget (aspect présentation de l'IU) approprié pour une interaction donnée tel que décrit précédemment dans la section

4.2.3.2. L'aspect modèle et contrôleur du patron MVC sont représentés respectivement par la spécification de données (*IDDClassD*) et la spécification du comportement (Réseaux de Pétri). L'aspect vue est automatiquement dérivé à partir du modèle et du contrôleur.

Dans cette activité, nous dérivons des spécifications des RdPCs un prototype d'interface usager. L'interface générée comprend un menu pour basculer entre les cas d'utilisation qui sont directement accessibles depuis l'état initial (la place *Init*) de l'interface IDD. Les différentes fenêtres (frames) du prototype d'interface représentent l'aspect statique, l'aspect dynamique du prototype d'interface, tels qu'ils figurent dans les spécifications des RdPCs sera représenté par des fenêtres de type dialogue (*dialog controls*) du prototype d'interface.

Dans notre implémentation actuelle, les prototypes sont générés à partir de l'environnement de développement NetBeans (version 6.9) [netbeans 2009]. L'application comprend un certain nombre de fenêtres (frames) et de fonctionnalités de navigation, voir Figure 4.15.

L'opération de génération prototype d'interface est composée des étapes suivantes :

- Génération du graphe des transitions (section 4.3.1).
- masquage des transitions non interactives (section 4.3.2).
- Identification des blocs d'interface (section 4.3.3).
- Composition des blocs d'interface (section 4.3.4).
- génération des fenêtres à partir des blocs composés d'interface (section 4.3.5).

4.3.1. Génération du graphe de transition

Cette opération consiste à dériver un graphe de transitions (GT) depuis le RdPC du cas d'utilisation *Generate_object*. Les transitions des réseaux de Pétri colorés seront représentés par des nœuds du GT, et les bords indique la priorité d'exécution entre les transitions. Si la transition t_1 précède la transition t_2, nous aurons un bord entre les nœuds représentant t_1 et t_2.
Un GT a une liste de nœuds (*nodeList)*, une liste de bords (edgeList), et une liste initiale de nœuds (*initialNodeList)* (nœuds d'entrée pour le graphe).

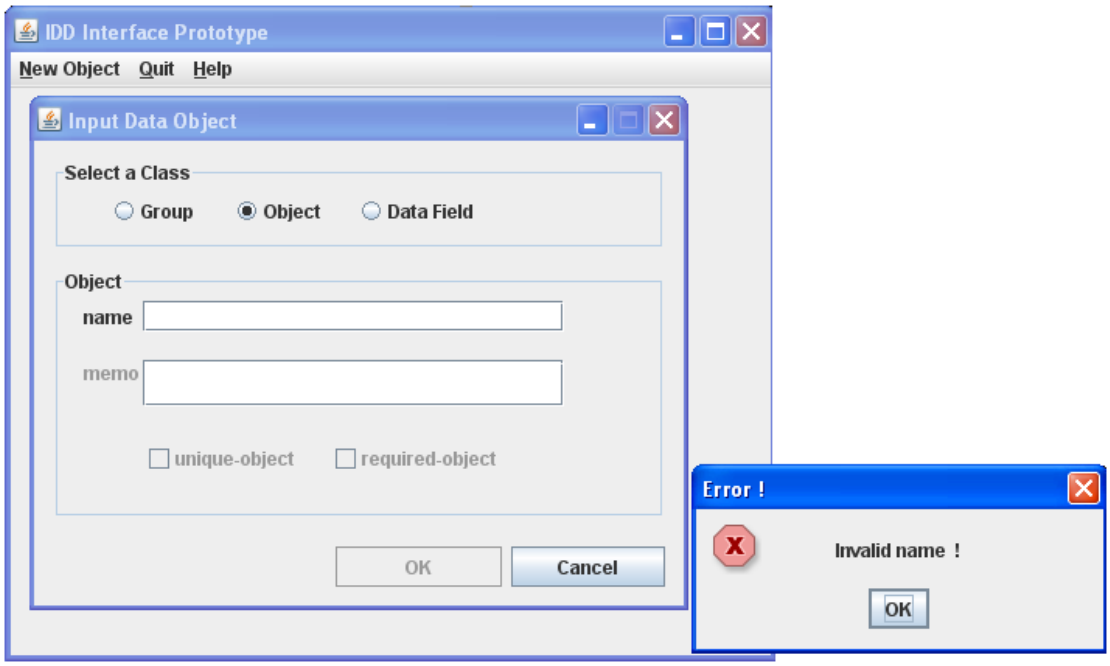


Figure 4.15. Mise en œuvre du prototype d'interface des données d'entrée du fichier IDD (Implémentation Java).

La liste des nœuds (*nodeList)* d'un GT est facilement obtenu, car il correspond à la liste des transitions du réseau de Pétri coloré, à portée de main. La liste des bords (edgeList) d'un GT est obtenue en reliant les transitions en amont d'une place (→ p) avec les transitions qui viennent à la suite de cette même place (p →).

Le caractère astérisque (*) est utilisé dans notre travail pour marquer les nœuds initiaux dans le graphe. Les différents nœuds du graphe de transition GT sont identifiés comme suit :

- t1 = Fournir la classe (*provide Class*).
- t2 = Nouvel objet (*new object*).
- t3 = Entrer le nom de l'objet (*Enter Object Name*).
- t4 = Valider le nom (*validate name*).
- t5 = Nom valide (*valid name*).
- t6 = Obtenez commentaires (get comments).
- t7 = confirmer (*confirm)*.
- t8 = Nom non valide (invalid Name).
- t9 = Erreur de nom (error name).
- t10 = Continuer (continue).

Le graphe de transition (GT) de RdPC intégré du cas d'utilisation *Generate_object* est représenté par la figure 4.16 (a).

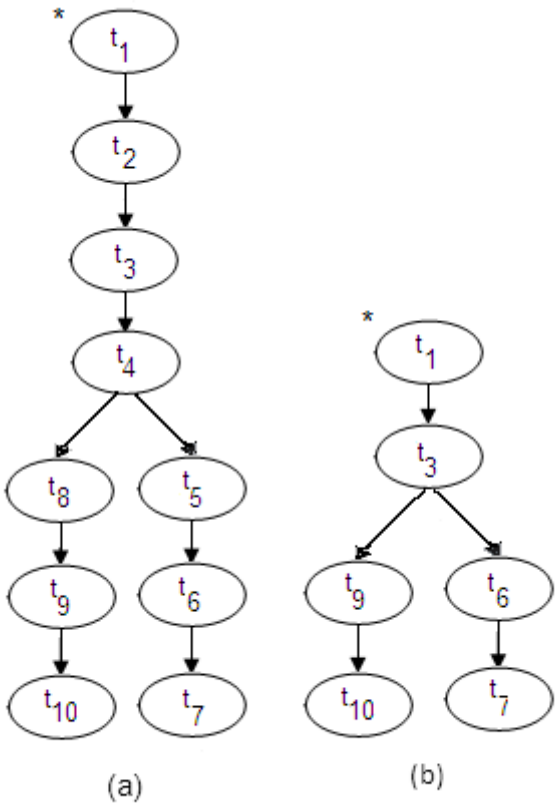


Figure 4.16. Le cas d'utilisation *Generate_object* : (a) graphe de transition GT; (b) Graphe de transition GT'

4.3.2. Masquage des transitions non interactives

Cette opération consiste à enlever toutes les transitions qui ne concernent pas directement l'interface (c'est-à-dire les transitions sans contraintes d'interface). Ces transitions sont appelés les transitions non interactives. Toutes ces transitions sont supprimées de la liste des nœuds (*nodeList*) et de la liste initiale des nœuds (*initialNodeList*), et tous les bords définis par les transitions sont aussi retirés de la liste des bords (*edgeList*).

Quand une transition t_i est supprimée de la liste des nœuds (nodeList), nous enlevons aussi tous les bords où la transition t_i prend part, et nous ajoutons un nouveau bord afin de combler les nœuds de transition enlevées. Si la liste initiale des nœuds (*initialNodeList*) des transitions initiales contient des transitions non interactives, ils sont remplacés par leurs successeurs nœuds. Le résultat de cette opération est illustré par le graphe de transitions de la figure 4.16 (a) est le graphe mis à jour des transitions GT ' de la figure 4.16 (b).

4.3.3. Identifier les blocs d'interface

Cette opération consiste à construire un graphe orienté où les nœuds représentent des blocs d'interface (*IBs*). Un *IB* est un sous-graphe du graphe GT qui consiste en

une séquence de nœuds de transition et qui est caractérisé par une seule entrée et un bord de sortie unique. Le début et la fin d'un *IB* sont identifiés à partir du graphe GT" sur la base des règles suivantes de génération des blocs (BGR_1 à BGR_6) [27] :

- (BGR_1) Un nœud initial du graphe GT 'est le début d'une *IB*.
- (BGR_2) Un nœud qui a plus d'un bord d'entrée est le début d'une *IB*.
- (BGR_3) Un successeur d'un nœud qui a plus d'un bord de sortie est le début d'une IB.
- (BGR_4) Un prédécesseur d'un nœud qui a plus d'un bord d'entrée termine un *IB*.
- (BGR_5) Un nœud qui a plus d'un bord de sortie termine un *IB*.
- (BGR_6) Un nœud qui a un bord de sortie à un nœud initial termine un *IB*.

En appliquant ces règles au graphe de transition GT' de la figure 4.16 (b), on obtient le graphe de blocs d'interface (*GIB*) montré par la figure 4.17 (a). Dans l'exemple, (BGR_1) détermine le début du IB_1 (t_1) et (BGR_5) est la fin du IB_1 (t_3). Les blocs d'interfaces IB_2 et IB_3 sont générées par l'application des deux règles (BGR_3) et (BGR_5).

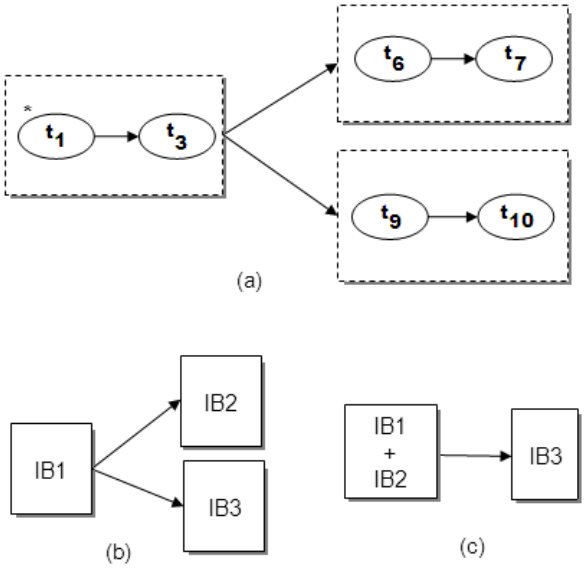


Figure 4.17. (a) bloc d'interface du graphe de transition GT; (b) bloc d'interface de la GT' ; (b) bloc d'interface de GIB '

4.3.4. Génération des fenêtres (frames) à partir des blocs d'interface composés

Dans cette opération, nous générons pour chaque *IB* d'un graphe de transitions (GT) une fenêtre graphique. Les fenêtres générées contiennent des widgets de toutes les transitions appartenant au bloc d'interface concernée (IB). Pour le prototype d'interface du fichier IDD, le menu généré est composé uniquement du cas d'utilisation *Identify_object*, voir la figure 4.18. C'est le seul cas d'utilisation qui a un accès direct à l'état initial (la place *Init*) de notre RdPC.

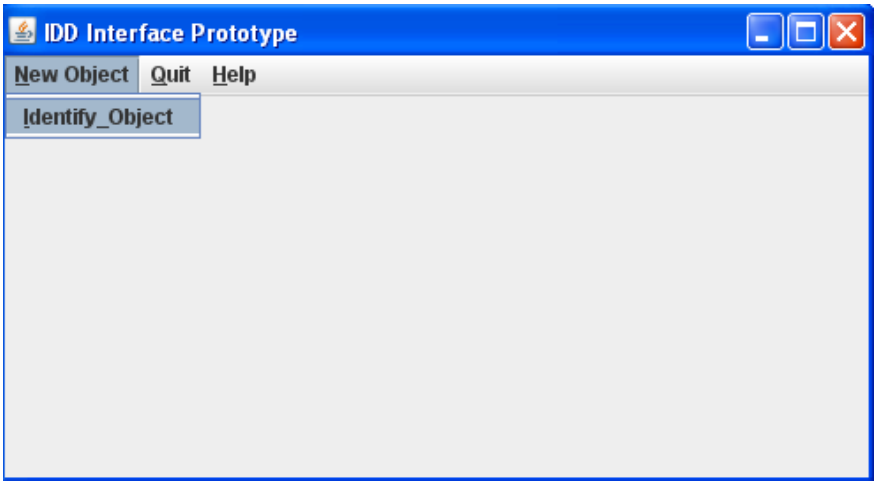


Figure 4.18. Prototype d'interface du fichier IDD : fenêtre principale.

Les deux autres fenêtres issues des blocs de construction composés, figure 4.17 (c), sont représentés par les figures 4.19 et 4.20. L'aspect dynamique du prototype d'interface est commandé par la spécification de comportement des deux graphes de transitions GT et GT'. L'exécution du prototype d'interface du fichier IDD s'appuie sur le parcours du graphe de transition GT'. Le prototype répond à tous les événements d'interaction avec l'utilisateur capturés par le graphe de transition GT' et ignore tous les autres événements.

Par exemple, après avoir sélectionné le cas d'utilisation *Identify_object* dans le menu principal de la figure 4.18 (fenêtre en haut), une autre fenêtre s'affiche permettant à l'utilisateur de choisir entre trois types de classes d'entrée : un *groupe d'objets* liés, ou un *objet d'entrée*, ou un champ de données (*alphanumérique* ou *numérique*).
Une fois un d'objet classe est sélectionné, l'affichage approprié de la saisie des données est activée dans la même fenêtre graphique, voir la figure 4.19. Lorsque l'exécution atteint un nœud du graphe de transition GT' à partir duquel plusieurs chemins de continuation sont possibles, le prototype d'interface IDD affiche la boîte de dialogue appropriée pour le bon scénario.

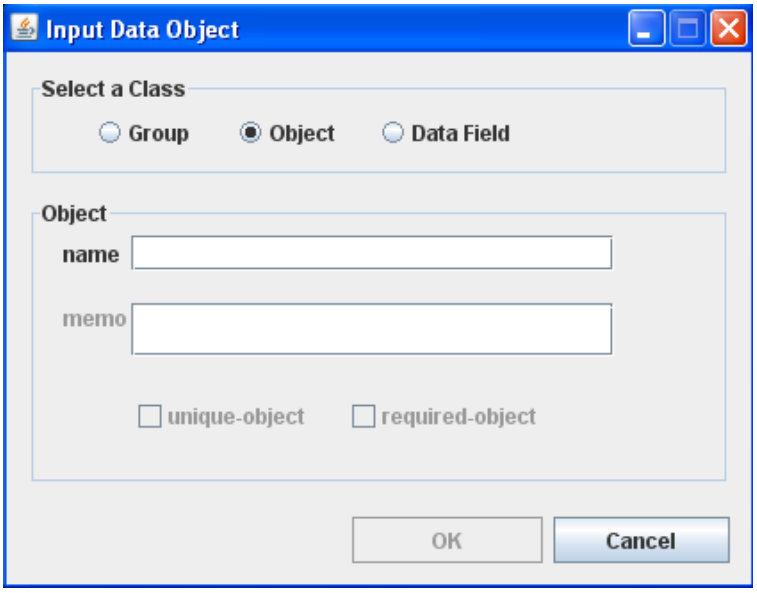


Figure 4.19. Prototype d'interface du fichier IDD : sélection d'une classe d'objet

Figure 4.20. Boite de dialogue pour le processus de génération d'erreur.

4.4. Conclusion

Cette approche de génération du prototype de l'IU du fichier IDD se distingue par une génération plus guidée par le comportement du système que par l'analyse fonctionnelle des tâches [Bodart et al. 94]. Les aspects statique et dynamique du prototype sont générés de manière semi manuelle, tandis que dans une approche précédente [Bachkhaznadji et al. 2006], seul l'aspect statique de l'IU est généré. L'aspect dynamique de l'IU est manuellement élaboré dans l'étape de la conception du dialogue.

Discussion de l'approche de génération d'interface usager et Conclusion

Ci-dessous, nous discuterons de l'approche en ce qui concerne les points suivants :
1. approche basée sur la notion scénario, 2. Les vues système et objets,
3. formalisme visuel pour la modélisation du prototype d'interface du fichier IDD.

5.1. Approche basée sur la notion scénario

L'approche de génération d'interface usager du fichier IDD du programme de simulation des performances énergétiques du bâtiments et des systèmes CVC associés, présente les avantages des techniques basées sur les scénarios. Nous mettons l'accent sur l'aspect de la modélisation (dynamique), et de générer le noyau dynamique de l'interface plutôt que de se concentrer sur la conception statique et visuelle à l'écran. Que les scénarios sont des descriptions partielles, il est nécessaire d'obtenir tous les scénarios possibles de l'interface du fichier IDD pour produire un descriptif complet.

Dans notre travail, les réseaux colorés de Pétri sont utilisés pour retenir les scénarios d'entrelacement [Elkoutbi et al. 2006]. Nous avons défini les variables de composition (jetons de couleur) comme décrit dans la section 4.2.3.4 du chapitre 4, afin d'interdire l'entrelacement des scenarios. Les spécifications qui en résultent permettront de saisir exactement le scénario d'entrée de données et pas plus. Il est bien connu [Elkoutbi et al. 2000] que l'évolutivité est un problème inhérent lorsqu'il s'agit de scénarios pour des applications de grande taille. La méthodologie développée facilite cette difficulté en intégrant les scénarios pour chaque cas d'utilisation, plutôt que de traiter une masse de scénarios non structurée.

5.2 Vues système et objets

Dans ce travail, nous nous concentrons sur l'utilisation des diagrammes de séquence sous forme de scénarios pour la restructuration du prototype de l'interface usager formelle du fichier IDD (vue système) plutôt que d'aborder la spécification d'objet individuel, tel que les classes ICS, Input Object, etc. ..., voir la figure 4.5, un système interactif (vue des objets).

Aux fins de générer le prototype d'interface usager du fichier IDD, la vue système est approprié lorsque dans tous les cas d'utilisation et les scénarios associés, un seul acteur interagit avec l'interface IDD. Dans le cas de tâches collaboratives (plus d'un utilisateur interagissant avec les cas d'utilisation), la vue objets, cependant, sera plus appropriée [Khriss et al. 99].

5.3 Formalisme visuel pour la modélisation du prototype d'interface du fichier IDD

Les réseaux colorés de Pétri sont parmi les plus puissants formalismes visuel utilisé pour spécifier un système complexe et interactif. Les réseaux colorés de Pétri sont connus pour leur soutien de la concurrence pure, toutes les transitions ayant un nombre suffisant de jetons dans leurs places d'entrée peuvent être activée simultanément.

Les réseaux colorés de Pétri dans leur forme de base ne prend pas en charge la hiérarchie, mais l'extension de ces réseaux utilisés dans des outils tels que le logiciel CPNTools permet de les hiérarchiser dans leur spécifications [Jensen 97].

Les choix non déterministes peuvent être plus facilement modélisés à l'aide de réseaux colorés de Pétri. Lorsque deux ou plusieurs transitions partagent les mêmes places d'entrée, le système (CPNTools) choisit au hasard pour activer une de ces transitions (section 4.2). Les Jetons qui sont spécifiques aux réseaux colorés de Pétri peuvent être utilisés à la fois dans le contrôle et la simulation du comportement de l'interface usager du fichier IDD et dans la modélisation des données et des ressources de ce système. Si par exemple, la place *Init* de la figure 4.3 (chapitre 4) contient un seul jeton, le système (interface IDD) ne peut simplement qu'exécuter un cas d'utilisation à la fois. Lorsque la place *Init* contient n jetons, n exécutions simultanées de différents cas d'utilisation sont possibles. Il est possible d'exécuter n scénarios du même cas d'utilisation (plusieurs instances).

5.4. Conclusion et travaux futur

Dans ce travail, nous avons défini un processus itératif pour l'ingénierie des exigences d'un système interactif pour le développement d'un prototype d'interface usager pour le fichier IDD du programme de simulation des performances énergétiques EnergyPlus. Ce processus a été raffiné par une approche de modélisation. L'approche vise l'étude du comportement de tout le système en utilisant les réseaux de Pétri à haut niveau comme formalisme de spécification.

L'approche de modélisation utilise les scénarios pour l'acquisition des besoins et génèrent un prototype de l'interface usager du système à partir des spécifications des comportements et des données du système. Les principaux apports de ce travail, outre le cadre méthodologique lié au processus itératif proposé et la proposition d'une approche pour la génération d'un prototype d'interface pour le fichier IDD à partir de scénarios. Les scénarios sont acquis sous forme de diagrammes de séquence selon la notation UML, et enrichi avec des informations d'interface utilisateur. Ces diagrammes de séquence sont transformés en spécifications sous forme de réseaux colorés de Pétri, à partir desquels un prototype d'interface du fichier IDD est généré. Les aspects statique et dynamiques du prototype d'interface sont dérivées depuis les spécifications des réseaux colorés de Pétri.

Les caractéristiques les plus intéressantes de cette approche résident dans l'utilisation de l'approche par scénarios qui aborde non seulement l'aspect séquentiel des scénarios, la spécification des scénarios dans le sens de l' UML (qui soutiennent aussi la concurrence dans les scénarios) , mais aussi dans la génération du prototype d'interface exécutable et qui peut être utilisé pour la validation des scénarios, et dégager l'interface usager cible du fichier d'entrée des données (IDD).
.

Dans le futur, nous envisageons de poursuivre le développement en particulier l'automatisation de l'activité d'intégration de scénarios en introduisant un algorithme qui prend une approche progressive d'intégration. Compte tenu deux scénarios avec deux réseaux colorés de Pétri correspondants (CPnet1 et $CPnet_2$), l'algorithme va fusionner toutes les places des réseaux $CPnet_1$ et $CPnet_2$ ayant les mêmes noms. Les places fusionnées auront comme couleurs symboliques l'union des couleurs symboliques des deux scénarios. Ensuite, l'algorithme cherche les transitions ayant la même places d'entrée et de sortie des deux réseaux colorés de Pétri et les fusionne pour obtenir un réseau coloré intégré.

Références Bibliographiques

[Annex21 98] ''Annexe21 – Calculation of Energy and Environment Performance of Buildings – Appropriate use of Programs''. IEA/ECBCS Publications, Vol. 1, May 1998, 146 pages. http:// www.ecbcs.org/ecbcss.html.

[Annex32 98] ''Annex30 – Bringing Simulation into Application''. Draft of final report. University of Liege(Operating Agent), Liège, Belgium, 1998.

[Bartholomew et al. 97]. Bartholomew D., Hand J., Irving S., Lomas K., Mc Elroy L., Parand F., Robinson D., Strachan P. '' An Application Manual for Building Energy and Environment Modeling''. Proceedings of the 5th International IBPSA Conference, Prague, Czech Republic, September 1997, 2: pp.387-393.

[Avision 90] Avision D. and Wood-Harper T.: '' Multiview Methodology''. Oxford. Blackwell Scientific Publishers, 1990.

[Bachkhaznadji et al. 2006] Bachkhaznadji A., Belhamri. A. ''A Model Driven Application for HVAC Energy Simulation Data: EnerMDA''. Paper presented at the 3rdCongress of HVAC Engineering, Climamed, Lyon, France, 2006; pp. 563- 572.

[Balzert 96] Balzert H.: From OOA to GUIs:'' The JANUS System. Journal of Object-Oriented Programming''. Vol.8, No.9, pp.43-47, 1996.

[Barry et al. 2005] Barry, O.S., and Marcus, K. ''Specification of an IFC Based Intelligent Graphical user Interface to support Building Energy Simulation''. Paper presented at the 9th IBPSA conference, Montreal, Canada. 2005

[Bäumer et al. 96] Bäumer D., Bischofberger W., Lichter H. and Zullighoven H.: ''User Interface Prototyping –Concepts, Tools And Experience''. IEEE Proceedings of ICSE-18, Berlin, Germany, pp.532-541 03/1996.

[Bischofberger et al. 92] Bischofberger W. and Pomberger G.: ''Prototyping-Oriented Software Development, Concepts and Tools''. Springer-verlag, 1992.

[Bodart et al. 94] Bodart F., Hennebert A.-M., Leheureux J.-M., Provot I. and Vanderdonckt J.: ''A Model-based Approach to Presentation: A Continuum from Task Analysis to Prototype''. Proceedings of the Euro graphics Workshop on Design, Specification, Verification of Interactive Systems, Carrara, Italy, Focus on Computer Graphics, Springer-Verlag, Berlin, pp.77-94 06/1994.

[Carroll 95] Carroll J. M.: ''The Scenario Perspective on System Development. In Scenario Based Design'': Envisioning Work and Technology in System Development. Ed Carroll J. M., pp.1-17, Wiley & Sons 1995.

[Collins 95] Collins, D. ''Designing Object-Oriented User Interfaces''. Reading, MA: Addison-Wesley 1995.

[Coutaz 90] Coutaz J. : ''Interface Homme-Ordinateur, Conception et Réalisation''. Dunod Informatique, 1990.

[CPNTools 2006] ''CPNTools version 2.2.0''. Department of Computer Science, University of Aarhus 2006, Available from http:// wiki.diami.au.dk/cpntools/.

[Crawley et al. 97] Crawley, D.B., Buhl W.F., Erden A.E., Fisher D.E., Lawrie LK., Liesen R.J., Pederson C.O., Winkelmann F.C. '' What Next for Building Energy Simulation – A glimpse of the. future''. Proceedings of the 5th International IBPSA conference, Prague, Czech Republic, September 1997, 2: 395-402.

[Crawley et al. 99] Crawley, D.B., Lawrie, L.K., Winkelmann, F.C., Buhl, W.F., Erdem, A.E., Huang, Y.J., Pedersen, C.O., Liesen, R.J., Fisher, D.E., Strand, R.K., & Taylor, R.D. ''EnergyPlus: a New Generation Building Energy Simulation Program''. Paper presented at the 6th International IBPSA Conference, Kyoto, Japan. 1999.

[Crawley et al. 2001] Crawley, D.B., Winkelmann, F.C., Laurie, L.K., and Pedersen, C.O. '' EnergyPlus: New Capabilities in a Whole Building Energy Simulation Program''. Paper presented at the 7th International IBPSA Conference, Rio de Janeiro, Brasil, 2001.

[Elkoutbi 98] Elkoutbi M. and Keller R. K.: ''Modeling Interactive Systems with Hierarchical Colored Petri Nets''. In Proc. of 1998 Adv. Simulation Technologies Conf., pp.432-437, Boston, MA (04/1998).

[Elkoutbi 2000] Elkoutbi, M., & Keller, R. K. '' User Interface Prototyping based on UML scenarios and High-level Petri-Nets''. Paper presented at the 21st International Conference on ATPN, Springer-Verlag LNCS 1825, Aarhur, Denmark, pp.166-186, 2000.

[Donn 97] Donn M.R. '' A Survey of Users of Thermal Simulation Programs''. Proceedings of the 5th International IBPSA conference, Prague, Czech Republic, September 1997, 3: 65-72.

[Durand 98] Durand D. ''La Systémique''. Série Que Sais-je ? Editions PUF, 1998, 128 pages.

[Dubois 91] Dubois A.M., Laret L. '' modélisation – Simulation Thermique : ALMETH''. Réf. CSTB MGL/91-1246/NB, septembre 1991.

[Ebert 93] Ebert R. '' Développement d'un environnement de Simulation de Systèmes Complexes – Application au Bâtiment''. Thèse présentée devant l'école Nationale des Ponts et Chaussées, Paris, 1993.

[Edwin et al. 2002] Edwin, O.S, Van Djik, L.R., Peter, G.L., and Prof, I.R. '' An architect Friendly Interface for a Dynamic Building Simulation Program''. Paper presented at the Sustainable Building . Conference. 2002.

[Fischer et al. 99] Fischer, D.E., Taylor, R.D., Buhl, F., Liesen, R.J., and Strand, R.K. '' A Modular Loop-Based Approach to HVAC Energy Simulation and its Implementation in EnergyPlus''. Paper presented at the 6th International IBPSA Conference, Kyoto, Japan. 1999.

[Foley 93] Foley J.D. and Sukaviriya P.: ''A Second Generation User Interface Design Environment: The Model and the Runtime Architecture''. INTERACT'93 and CHI'93, pp.375-382, Amsterdam (04/1993).

[Fuestel 90] Fuestel, H. E. 1990. "The COMIS Air-Flow Model –A Tool for Multizone Applications". Proceedings of the 5th International Conference on Indoor Air Quality and Climate, Vol. 4, pp. 121-126.

[Garnier 95] Garnier C. '' Conception et Développement d'une Bibliothèque de Modèles de Thermodynamiques appliquée''. Thèse présentée devant l'école des Mines de Paris, décembre de 1995.

|Gicquel 92] Gicquel R. '' Approche Systémique et Thermique Instationnaire''. Actes du colloque SFT92 de thermique ''Systèmes Thermiques Instationnaires '', Ecole des Mines de Paris, pp. 337-355, 1992.

[Glinz 2002] Glinz, M. ''Statecharts for Requirements Specifications-As simple as Possible, as Rich as Needed''. Position Paper in the ICSE 2002 Workshop: Scenarios and state machine models, Algorithms, and tools, Orlando, Florida, USA. 2002.

[Goldberg 84] Goldberg A.: ''Smalltalk-80, the Interactive Programming Environment''. Vol.1, Tome 2: The Language and its Implementation. Addison Wesley, 1984.

[Gordon et al. 95] Gordon V.S. and Bieman J.: ''Rapid Prototyping: Lessons Learned''. IEEE Software, Vol.12, No.1, pp.85-95 01/ 1995.

[Graham 95] Graham, I. '' Use Cases Combined with Booch/OMT/UML, Process and Product''. Journal of Object Programming, pp.76-78, 1995.

[Green et al. 91] Green T.R.G. Petre M., Bellamy R.K.E. ''Comprehensibility of Visual and Textual Programs: A test of Superlativism against the 'match-Mismatch' conjecture''. Proceedings of Empirical Studies of Programmers: 4th workshop, Ablex, New Jersey, pp. 121-141, 1991.

[Harton 90] Harton H.R. and Hix D.: ''Developping Human-Computer Interface Models and 166 Representation Techniques''. Software-Practice and Experience, Vol.20, No.5, pp.425- 457 (05/1990).

[Heymans 98] Heymans P. and Dubois E.: ''Scenario-Based Techniques for Supporting the Elaboration and the Validation of Formal Requirements. Requirements Engineering'', Vol.3, No.3/4, pp.202-218, 1998.

[Hsia et al. 94] Hsia P., Samuel J., Gao J., Kung D., Toyoshima Y. and Chen C: ''Formal Approach to Scenario Analysis''. IEEE Software, Vol.11, No.2, pp. 33-41, 1994.

[IBM 91] IBM ''Systems Application Architecture: Common User Access-Guide to User Interface Design-Advanced Interface Design Reference'', IBM, 1991.

[Janssen 93] Janssen C., Weisbecker A. and Ziegler J.: ''Generating User Interfaces from Data Models and Dialogue Net Specifications''. IFIP International Conference on Human-Computer Interaction INTERACT'93 and Conference on Human Factors in Computing Systems CHI'93, pp.418-423, Amsterdam 04/ 1993.

[Jacobson et al 92] Jacobson I., Christerson M., Jonsson P. and Oevergaard G. '' Object-Oriented Software Engineering'': A Use Case Driven Approach''. Addison Wesley, 1992.

[Jackson 96] Jackson M.: Software Requirements and Specifications. Addison Wesley, 1996.

[Jensen 95] Jensen K.: ''Coloured Petri Nets, Basic Concepts, Analysis Methods and Pratical Use'', Springer, 1995.

[Jensen] Jensen, K., & Kristensen, L.M. ''Coloured Petri-nets: Modelling and Validation of Concurrent Systems''. Springer-Verlag, Textbook in preparation. Available from: http://wiki.daimi.au.dk/cpntools/cpntoolsttt.wwik.

[Jensen 97] Jensen, K. ''Coloured Petri-nets, Basic Concepts, Analysis Methods and Practical use''. Volume3: Practical use, Springer-Verlag. 1997.

[Jensen et Kristensen] Jensen, K., & Kristensen, L.M. Coloured Petri-nets: Modelling and Validation of Concurrent Systems. Springer-Verlag, Textbook in preparation. Available from: http://wiki.daimi.au.dk/cpntools/cpntoolsttt.wwik.

[Johson 92] Johnson J.: ''Selectors: Going Beyond User-Interface Widgets''. Conference on Human Factors in Computing Systems CHI'92, pp. 273-279, Monterey, CA 05/ 1992.

[Judkoff 95] Judkoff R., Neymark J.:''Annex 21- Building Energy Simulation Test (Bestest) and Diagnostic Method''. IEA/ECBCS Publications, 1995. (http://www.ecbcs.org/ecbcs.html.

[Junjie et al. 2007] Junjie, L., Wenshen, L., and Xianjie, Z. ''The Study of the Simple Hvac Interface of EnergyPlus in Chinese''. Paper presented at the Building simulation Conference, Beijing, China. 2007.

[Kawashita 97] Kawashita I. : ''Spécification Formelle de Systèmes d'Information Interactifs Par La Technique de Scénarios''. Thèse de Maitrise, Université de Montréal, 1997.

[Keilholz 96] Keilholz W.P. '' Développement d'un outil de calcul intégré multi-domaine, basé sur L'approche orientée objets''. Thèse présentée à l'Université de Nice – Sophia-Antipolis, Janvier 1996.

[Khriss et al. 99] Khriss I., Elkoutbi M. and Keller R. K.: ''Automating the synthesis of UML Statechart Diagrams from Multiple Collaboration Diagrams''. In J. Bezivin and P. A. Muller, Ed. <<UML>>'98: Beyond the Notation, pp.132-147. Springer LNCS 1618, 1999.

[Kristensen et al. 2004] Kristensen, L.M., Jorgensen, J.B., & Jensen, K. Application of Coloured Petri-nets in System Development. In Lectures on Concurrency and Petri-nets – Advances in Petri-nets. Paper presented at the 4th Advanced Course on Petri-nets. Volume 3098,; p. 626-685, Springer-Verlag, 2004.

[Kuuti 95] Kuuti K.: ''Creating Contexts for Design''. In Caroll J.M. editor, Scenario-Based Design: Envisioning Work and Technology in System Development, pp.19-36. John Wiley and Sons, 1995.

[Kyng 95] Kyng M.: ''Creating Contexts for Design''. In Caroll J.M. editor, Scenario-Based 168 Design'': Envisioning Work and Technology in System Development, pp.85-107. John Wiley and Sons, 1995.

[LeBrun et al. 99] LeBrun, J., J.P. Bourdouxhe, and M. Grodent.. HVAC1 toolkit: A toolkit for primary HVAC System energy calculation, Atlanta: American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc., 1999.

[Laret 89] Laret L., ''Modélisation du Comportement Thermique de l'Immeuble Expérimental de la DETN''. Rapport final – tome II, Réf. CSTB : MGL-1075-B/NB, Juillet 1989.

[Le Gallou 94] Le Gallou F., Bouchon-Meunier (Coordonnateurs) : ''Systémique – Théorie et Applications''. Editions Tec & Doc, Paris, 1994, 341 pages.

[Liang 2003] Liang, Y. ''From Use Case to Classes: A Way of Building Object Model with UML''. Journal of Information and Software Technology, Volume 45,; p. 83-93, 2003.

[Myers 95] Myers B.A.:''User Interface Software Tools, ACM Transactions on Computer-Human Interaction''. Vol.2, No.1, pp. 64-103, 03/1995.

[Marshallsay et al. 97] Marshallsay P.G., et Luxton R.E. ''Design of the air Conditioning systems for efficient Life-cycle Operation using ZEBRA software-package''. Proceedings of the 5th International IBPSA conference, Prague, Czech Republic, September 1997

[Mc Elroy 97] Mc Elroy L.B., Hand J.W., Strachan P.A, ''Experience from a Design advice Service using Simulation''. Proceedings of the 5th International IBPSA conference, Prague, Czech Republic, 1:pp. 19-26, September 1997,

[netbeans 2009] http://www.netbeans.org

[Olsen 2002] Olsen, E.L. Performance comparison of UK Low Energy Cooling Systems by Energy Simulation (Master of Science, Massachusetts Institute of Technology). 2002.

[Pfaf 85] Pfaf G. E.: ''User Interface Management Systems''. Eurographics Seminars, Springer-Verlag , 1985.

[Petri-Net 99] Petri Net Tools Database Quick Overview: http://www.daimi.aau.dk/PetriNets/tools/quick.html, 1999.

[Potts et al. 94] Potts, C., Takahashi, K., & Anton, A. Inquiry-Based Scenario Analysis of System Requirements. Technical Report GIT-CC-94/14, Georgia Institute of Technology. 1994.

[Potts 95] Potts C.: ''Using Schematic Scenarios To Understand User Needs''. Proceedings of the Symposium on Designing Interactive Systems, Michigan USA, pp.247-256 08/1995.

[Rolland et al. 98] Rolland C., Ben Achour C., Cauvet C., RalytéJ., Sutcliffe A., Maiden N.A.M., Jarke M., Haumer P., Dubois P. K., and Heymans P.: '' A Proposal for a Scenario Classification Framework''. In Requirements Engineering Journal, Vol.3, No.1, pp.23-47 1998.

[Rumbaugh et al. 91] Rumbaugh J., Blaha M., Premerlani W., Eddy F. and Lorensen W.: ''Object-Oriented Modelling and Design''. Prentice-Hall Inc. 1991.

[Rumbaugh et al. 99] Rumbaugh, J., Jacobson, I., and Booch, G.'' The Unified Modelling Language Reference Manual''. Addison Wesley. 1999.

[Shady et al. 2009] Shady, A., Liliana, B., De Herde, A., and Jan, J. ''Architect Friendly: A Comparison of Ten Different Building Performance Simulation Tools''. Paper presented at the 11th International IBPSA Conference, Glasgow, Scotland. 2009.

[Soubra 92] Soubra S. '' Concepts et Méthodes pour le développement d'environnements de Simulation Intelligents – Application au Bâtiment''. Thèse présentée devant l'école Nationale des Ponts et Chaussées, Paris, 1992.

[Szekely 95] Szekely P.: ''User Interface Prototyping: Tools and Techniques''. Software Engineering and Human Computer Interaction: Joint Research Issues, pp.76-92. Springer LNCS 896, 1995.

[Tabary 97] Tabary L. ''CA-SIS – A design tool for thermal studies with gradual access''. Proceedings of the 5th International IBPSA conference, Prague, Czech Republic, pp: 41-48. September 1997,

[Tang et Clarke 93] Tang D., et Clarke J.A., ''Application of the Object Oriented Programming Paradigm to Building Plant System modeling''. Proceedings of the 3rd International IBPSA conference, Adelaide, August 16-18, 1993.

[Tarby 93] Tarby J-C. : ''Gestion automatique du dialogue homme-machine à partir de spécifications conceptuelles''. PhD thesis, Université de Toulouse, France ,09/1993.

[Taylor et al. 90] Taylor, R. D, C. E. Pedersen, and L. K. Lawrie. "Simultaneous Simulation of Buildings and Mechanical Systems in Heat Balance Based Energy Analysis Programs" Proceedings of the 3rd International Conference on System Simulation in Buildings, Liege, Belgium, December 3-5, 1990.

[Taylor et al. 91] Taylor R. D., C. E. Pedersen, D. E. Fisher, R. J. Liesen, and L. K. Lawrie. "Impact of Simultaneous Simulation of Building and Mechanical Systems in Heat Balance Based Energy Analysis Programs on System Response and Control''. Proceedings of Building Simulation '91, Nice, France , August 1991,.

[TRNSYS 96] TRNSYS - A transient Simulation Program. Solar Energy Laboratory of the University of Wisconsin; Madison, USA, July 1996.

[Winkelmann et al. 85] Winkelmann, F. C. and S. E. Selkowitz. "Daylighting Simulation in the DOE–2 Building Energy Analysis Program". Energy and Buildings, pp. 271-286. 8/1985.

[Winkelmann et al. 93] Winkelmann, F. C., B. E. Birdsall, W. F. Buhl, K. L. Ellington, A. E. Erdem, J. J. Hirsch, and S. Gates. DOE– 2 Supplement, Version 2.1E, LBL-34947, Lawrence Berkeley National Laboratory. Springfield, Virginia: National Technical Information Service, November 1993.

[Wolf 99] Wolf G.:'' Petri Net Tools''. http://home.arcor-online.de/wolf.garbe/petrisurv2.html 1999.

Références Internet

[WEB 1] http://www-epb.lbl.gov/ventilation/program.html

[WEB 2] http://erg.ucd.ie/combine.html

[WEB 3] http://iaiweb.lbl.gov/

[WEB 4] http://www.strath.ac.uk/Departments/ESRU/courseware/Class-mod+sim/

[WEB 5] http://www.eren.doe.gov/buildings/tools_directory/toolsdir.html

[WEB 6] http://www.granlund.fi

[WEB 7] http://www.eren.doe.gov/buildings/energy_tools/energyplus.htm.