# Prise en compte de la complexité de modélisation dans la gestion énergétique des bâtiments

Yanis Hadj Said

Communauté
UNIVERSITÉ Grenoble Alpes

# THÈSE

Pour obtenir le grade de

## DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : AUTOMATIQUE - PRODUCTIQUE

Arrêté ministériel : 25 mai 2016

Présentée par


## Yanis HADJ SAID


Thèse dirigée par **Stéphane PLOIX**,

préparée au sein du **Laboratoire de Sciences pour la Conception, l'Optimisation et la Production de Grenoble** dans **l'École Doctorale Electronique, Electrotechnique, Automatique, Traitement du Signal (EEATS)**

# Prise en compte de la complexité de modélisation dans la gestion énergétique des bâtiments

# complexity in modeling in building energy management

Thèse soutenue publiquement le **20 juillet 2016**,
devant le jury composé de :

**Monsieur Stéphane PLOIX**
Professeur , Institut polytechnique de Grenoble, Directeur de thèse
**Monsieur Hervé GUEGUEN**
Professeur, Supélec Rennes, Rapporteur
**Monsieur Jean-Jaques ROUX**
Professeur, INSA de LYON, Rapporteur
**Monsieur Frédéric WURTZ**
Directeur de recherche, CNRS, Examinateur
**Monsieur Pierre-Emmanuel CAVAREC**
Ingénieur, SOMFY, Examinateur
**Monsieur Patrick REIGNIER**
Professeur, INRIA, Examinateur
**Monsieur Cristian MURESAN**
Ingénieur, ENGIE, Examinateur
**Etienne WURTZ**
Directeur de recherche, INES CEA-Locie, Président

# Table des matières

Résumé : Du fait de son impact énergétique, la gestion énergétique dans le bâtiment est devenu un enjeu majeur ses dernières années, qu'il s'agisse d'encourager la sobriété énergétique ou de s'adapter aux besoins des réseaux énergétiques. Différents travaux de recherche ont conduit à des gestionnaires énergétiques souvent dotés de capacités d'anticipation. Les premiers résultats, bien qu'encourageants, n'envisagent pas la complexité tant du fait nombre d'éléments que de la diversité des applications de gestion énergétique.

Cette thèse propose d'apporter des éléments de solution à la problématique de complexité. Les travaux ont démarré par l'analyse du gestionnaire énergétique GHomeTech et son adaptation au prototype de bâtiment complexe CANOPEA. Les problématiques de composition sont explorées. Une solution favorisant la réutilisabilité d'éléments de modèles, l'agrégation et la transformation vers des modèles d'optimisation de type programmation linéaire en nombres entiers (PLNE), est proposée. L'outil résultant a été validé sur le projet CANOPEA.

La gestion énergétique ne se limite pas à l'optimisation PLNE. Différentes natures d'applications permettent d'offrir d'autres services : l'estimation paramétrique de modèles pour simplifier la configuration des gestionnaires énergétiques, la simulation pour la validation et la prédiction fonction de scénarios définis par exemple. Cette autre dimension de la complexité est abordée dans un second volet du manuscrit. Des solutions de réécriture automatique de modèles sont proposées grâce à des manipulations symboliques permettant différents types de transformation. Plusieurs exemples de génération automatique de modèles applicatifs sont présentés.

Abstract : Energy management for building has become a major issue this last decade because of its energy impact. Building energy management reduces energy wastes and enable a better matching between energy needs and grid capabilities. Different types of energy management systems are proposed in scientific literature, most of them with anticipation capacities. The first results do not really consider the complexity issue coming from the number of modeling elements and also coming from the diversity of energy management applications.

This thesis proposes elements of solution to the complexity problem. The work started by analyzing the energy management system 'GHomeTech' and its adaptation to the complex building prototype CANOPEA. The issue of composition from elementary models is explored. A solution is proposed ; it enables the reusability of elementary models. Aggregation and transformation into mixed integer linear programming optimization models is presented. The resulting tool has been validated on the CANOPEA project.

Energy management is not limited to MILP optimization. Different types of applications are also used to provide other services : parametric estimation models to simplify the configuration of energy management systems, simulation for validation and prediction depending on pre-defined scenarios for example. This other dimension of complexity is discussed in a second part of the manuscript. Solutions for automatic rewriting of models are detailed. It relies on symbolic manipulations in different types of processing. Several examples of applications illustrating the automatic generation of models are presented.

# Introduction générale

L'Homme a, de tout temps, eu besoin de s'abriter, de se nourrir et de se déplacer. Selon les époques, ces besoins ont été satisfaits de différentes manières. Le XVIIIème siècle a initié le début de l'aire industrielle qui a fait basculé les peuples vers le modèle de société de consommation selon lequel nous vivons encore actuellement. Néanmoins, ces trente dernières années ont été marquées par des constats alarmants sur les conséquences de notre mode de vie. Le modèle de consommation en constante croissance ne peut être soutenable à long terme d'une part en raison des ressources naturelles qui commencent à se tarir et d'autre part en raison des problèmes environnementaux liés aux déchets rejetés. Le travail nécessaire pour modifier les comportements en l'adaptant aux nouvelles contraintes de soutenabilité posées doit se faire sur les fondamentaux même de notre organisation de vie. Un de ces fondamentaux majeur est l'aspect énergétique de notre consommation. En effet, aujourd'hui, chaque élément qui nous entoure a nécessité de l'énergie pour sa production, son transport et peut aussi avoir besoin d'énergie pour son fonctionnement. Cela va de nos bâtiments à nos moyens de transport en passant par notre nourriture et nos loisirs. Les bâtiments ont nécessité de l'énergie pour leur construction, l'énergie grise, et nécessitent de l'énergie pour leur fonctionnement, l'énergie blanche.

Il existe plusieurs sources d'énergies et plusieurs modes d'utilisation. Les principales sources exploitées aujourd'hui en France sont le pétrole, le gaz, le charbon, la houille blanche (force de l'eau), l'uranium, les déchets organiques, les bio-combustibles, la géothermie, le vent et le soleil. Chaque source est propice à un usage particulier. Certaines le sont d'une manière directe, c'est-à-dire transformées sur le lieu de leur consommation finale. C'est le cas notamment du pétrole raffiné avec les moteurs à combustion ou le chauffage domestique au fioul. C'est le cas aussi du gaz naturel pour la cuisson et la production de chaleur dans l'habitat et dans l'industrie. D'autres sources sont exclusivement utilisées dans des usines de conversion en une forme mieux transportable. C'est le cas des boucles d'eau chaude et de l'électricité. Cette dernière représente plus de 22% (36,8 Mtep) (voir figure1) du total de l'énergie consommée en France dont 65% (23,97 Mtep) (voir figure 2) pour le seul bâtiment. L'électricité est principalement utilisée dans le bâtiment 65% et dans une moindre proportion dans l'industrie. Le transport et l'agriculture usent plus globalement du pétrole aujourd'hui. Cette configuration historique trouve sa justification par des hypothèses et des approches anciennes qui considéraient uniquement les coûts des infrastructures de production et de transport des énergies. Le coût de l'énergie dans sa forme primaire était négligé du fait de son abondance à l'époque. Le premier et le second choc pétroliers ont remis en cause les hypothèses de départ. Les hypothèses énoncées plus haut se sont heurtées à l'augmentation vertigineuse des couts de l'énergie primaire en raison de la raréfaction des ressources et de l'impact environnemental de l'usage de certaines d'entre elles. Ce dernier critère a poussé à revoir profondément notre manière de consommer l'énergie et surtout les sources que nous devons privilégier.

Le transport, par exemple, vit une véritable révolution en ce moment avec les poli-

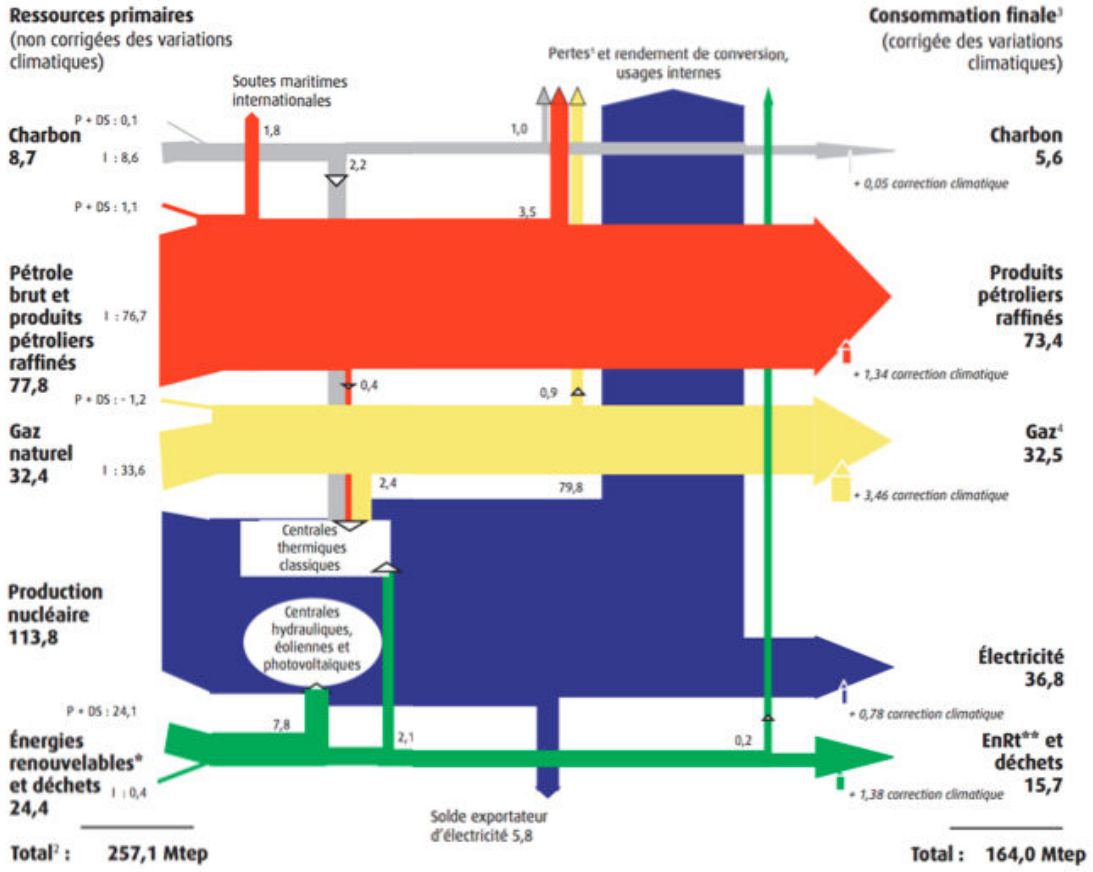



FIGURE *1 – sources et formes de consommation d'énergie en France.*
*Source : SOeS, bilan de l'énergie 2014*

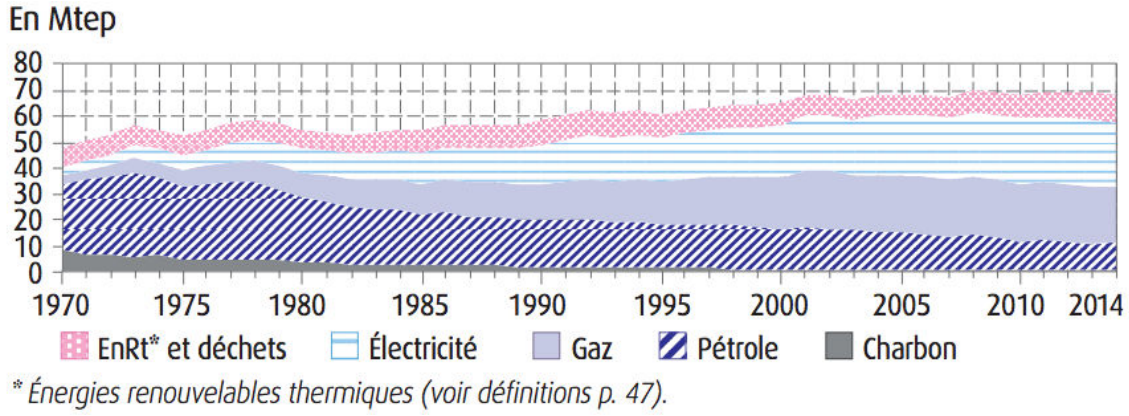



FIGURE *2 – consommation d'énergie dans le Bâtiment (résidentiel et tertiaire) par source.*
*Source : SOeS, bilan de l'énergie 2014*

tiques publiques qui réorientent les usagers vers les transports en communs, mutualisant ainsi l'usage. La révolution majeure à venir dans les transports concerne l'avénement du véhicule électrique. Celui-ci remplacera à terme, le véhicule thermique, qui a provoqué bien des scandales environnementaux et sanitaires. Le dernier en date concerne l'une des premières firmes automobiles au monde, le groupe Volkswagen, avec l'affaire du Diesel-Gate.

Les énergies renouvelables montent proportionnellement dans le mix énergétique. Ces deux changements, bien qu'utiles d'un point de vue environnemental, posent problème au niveau des infrastructures électriques. Pour l'instant, les énergies renouvelables se développement majoritairement sous forme de fermes de production à grande échelle qui peuvent s'insérer dans l'infrastructure comme le serait une centrale de production traditionnelle. Néanmoins, certaines sources d'énergies renouvelables, vent, marrées et soleil, par exemple sont intermittentes. Cela a pour conséquent d'apporter de la raideur et de l'incertitude, donc de l'instabilité sur le réseau électrique les supportant. Dans le même temps mais dans une proportion moindre, des moyens de génération électrique d'origine renouvelable, photovoltaïque ou mini-éolienne, beaucoup plus petits en capacité sont installés à même les bâtiments. Ceux-ci réinjectent leur production sur le réseau. Cette approche dite *décentralisée* changera radicalement la typologie des réseaux lorsque ce type d'installations montera en proportion dans l'équilibre global. Le bâtiment devient alors, plus qu'un simple consommateur final d'électricité. Il devient Hub énergétique.

La consommation dans le bâtiment est aussi sujette à beaucoup de recherches pour limiter le gaspillage notamment lié aux enveloppes et aux équipements HVAC inefficients. Des résultats assez probants tel que les labels BEPos ou BBC en ont résulté ainsi que dans les réglementations énergétiques (RT2012 notamment). Les maisons dite basse ou très basse consommation sont tellement performantes en terme d'isolation que le chauffage, qui est le premier poste de consommation dans un habitat traditionnel, ne consomme guère plus qu'un réfrigérateur à l'année. Les ratios de consommation par poste se retrouvent bouleversés et la prévisibilité de la consommation globale, indispensable au dispatcher du réseau électrique en pâti. Le véhicule électrique qui est aussi connecté au réseau via le bâtiment ou des bornes de recharge publiques induit des appels de puissance qui complique encore la gestion du réseau.

L'ensemble de ces problématiques devient difficilement gérable coté réseau et coté usager. Le réseau ne pourra plus continuer à assurer une fourniture optimale sans flexibilité qui induit une interaction avec les usagers. Les usagers eux mêmes se retrouveront alors débordés à cause de la multitude d'équipements et la complexité des interactions avec le réseau. Des solutions dites de gestion énergétique pour le bâtiment font leur chemin depuis la recherche scientifique vers le monde industriel pour répondre à cette nouvelle problématique.

Un outil de gestion énergétique dans le bâtiment est une solution matérielle et logicielle capable de piloter les flux énergétiques dans un espace de vie ou du moins assister leur pilotage. Il interagit avec un réseau de capteurs et d'actionneurs qui permettent la transmission d'informations entre l'environnement physique et le système logiciel. L'environnement physique est délimité par le périmètre de déploiement des capteurs et des actionneurs. Certains outils de gestion énergétique sont partiels dans le sens ou ils ne gèrent qu'un équipement ou un ensemble restreint d'équipements dans le bâtiment. L'interaction avec l'occupant du bâtiment n'en est qu'à ses balbutiements dans les approches de gestion énergétique dans le bâtiment.

La gestion énergétique dans le bâtiment au sens ou nous l'entendons aujourd'hui tire ses prémices de la gestion technique du bâtiment et de la domotique. A l'époque, la gestion technique concernait principalement les grands ensembles de bâtiments tertiaires dans lesquels les occupants n'étaient quasiment pas impliqués dans la gestion. Il fallut alors établir des moyens de commande des systèmes CVC (chauffage, ventilation, climatisation) et de l'éclairage couplés au PC sécurité. L'infrastructure était couteuse mais supportée grâce à l'économie d'échelle liée à la disposition des bâtiments tertiaires et au

fait que les lieux étaient des centres d'intérêts économiques majeurs.

La domotique concerne plus les bâtiments de type résidentiel individuel ou collectif. Elle regroupe un ensemble d'automatismes apportant du confort à l'occupant tel que l'ouverture commandée à distance des ouvrants, l'arrosage du jardin, la surveillance, etc...

Les outils de gestion énergétique avancés sont arrivé comme une surcouche logicielle qui permet d'intégrer des algorithmes d'anticipation, d'optimisation ou encore des interactions simplifiées entre les occupants et le bâtiment avec ses équipements. Cette surcouche peut s'avérer nécessaire pour une plus large gamme de bâtiments que ce que pouvait toucher la GTB. En effet, avec l'avénement du concept bâtiment hub énergétique, des bâtiments de type résidentiel individuel ou collectif aux bâtiments d'entreprises en passant par les galeries marchandes ou encore les bâtiments publiques tel que les gares ou les bibliothèques auront besoin d'assistance à la gestion de leurs flux énergétiques spécialement. L'approche sur laquelle nous avons basé nos travaux de recherche est baptisée GHomeTech. Elle a été développée dans un premier temps par (HA 2008) puis améliorée et complétée par (BADREDDINE 2012), (WARKOZEK 2011) et (LE MINH et al. 2011)

GHomeTech incarnent une approche de gestion multi-échelle. *Le terme multi-échelle signifie qu'un problème de gestion complexe est décomposé en plusieurs niveaux. Une gestion multi-échelle est un traitement hiérarchisé suivant différentes échelles de temps*. Telle est la définition que nous retrouvons dans (HA 2008). Par complexité, l'auteur adresse la complexité de calcul liée à la taille du problème de gestion. La hiérarchisation permet une décomposition des dynamiques des systèmes à travers un traitement séparé. Cette séparation permet la limitation de la dimension de l'espace de recherche pour les algorithmes de résolution.

La figure 3 montre les deux étages du traitement en plus de l'étage dit "terrain" qui représente l'asservissement et la commande directe des équipements. La couche anticipative est chargée de l'optimisation à grosses mailles. Celle-ci établie une stratégie de gestion énergétique sur un horizon de 24h avec un pas de temps d'une heure. Sur chaque pas de temps, la couche réactive intervient pour affiner la stratégie adoptée en signaux de commande à pas réduit pouvant être exécutés par les boitiers locaux de commande des équipements.

Nous traitons dans cette thèse le problème de l'appréhension de la complexité de modélisation pour la gestion énergétique.

Le chapitre I présente une revue de littérature des solutions de gestion énergétique. Nous synthétiserons l'analyse de l'étude par l'établissement de liens entre les différentes approches et notre problématique.

Deux aspects de la complexité vont être explorés dans deux parties : celui lié au nombre d'éléments dans un modèle et celui lié à la diversité des applications cibles.

La première partie du manuscrit réunit les chapitres II et III. Elle explore la gestion de la complexité liée au nombre.

Dans le chapitre II, nous présentons un ensemble de solutions existantes de gestion et de manipulation de modèles dans différents domaines afin d'établir un panorama. Il apparait nettement que la problématique de la complexité liée au nombre n'a pas encore été traitée. Ce chapitre s'intéresse à la problématique du nombre dans l'optimisation PLNE et propose une formulation de ce problème. Nous présentons dans le détail chaque partie de la solution ainsi que son implémentation informatique.

Le chapitre III décrit une première application de l'outil développé : le projet CANOPEA. Nous décrirons, dans un premier temps, chaque équipement intervenant dans l'habitat en rapport avec le bilan énergétique ou du confort. Nous montrerons par la suite

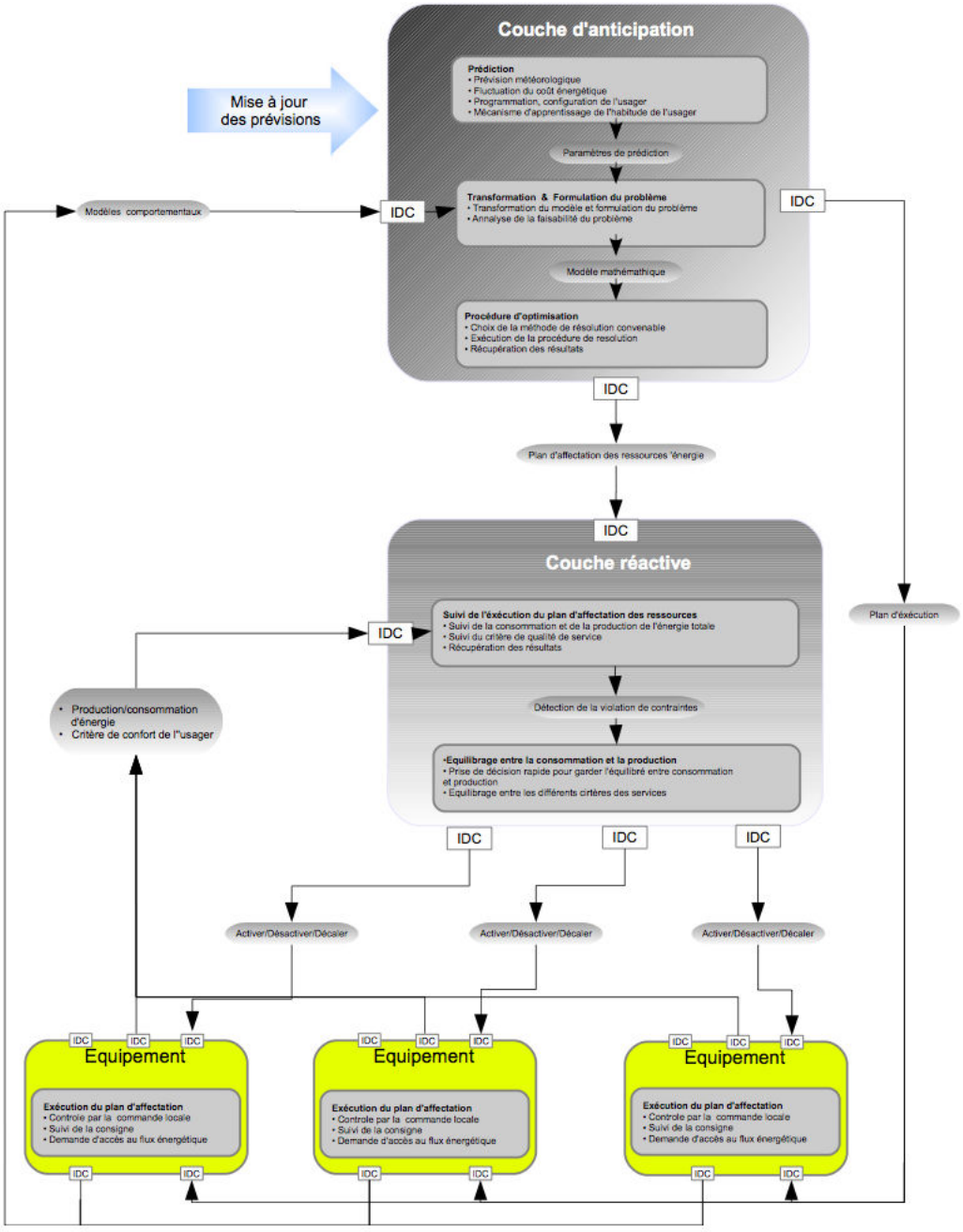



FIGURE *3* – *Architecture GHomeTech*

l'utilité des fonctionnalités apportées par l'outil de gestion de modèles pour l'optimisation PLNE.

La seconde partie du manuscrit réunit les chapitres IV, V et VI. Elle explore la gestion de la complexité liée à la diversité des applications de gestion énergétique.

Le chapitre IV explore les besoins de modèles pour les différentes applications de gestion énergétiques. Les applications simulation, estimation paramétrique et optimisation PLNE sont examinées à travers un exemple. Différentes natures de modèles apparaissent : acausal/causal, linéaire/ non-linéaire et orienté paramètres/orienté variables.

Après un état de l'art des approches multi-applicatives, le chapitre V propose de s'appuyer sur l'approche *model driven architecture* pour formaliser la problématique de trans-

formation de modèles vers différentes applications de gestion énergétique. Une méta-modélisation des différents types de modèles applicatifs est proposée et les principes de transformation sont énoncés. Ils s'appuient sur des manipulations formelles d'équations pour effectuer les transformations nécessaires : projection temporelle, linéarisation, ordonnancement causal,...

Le chapitre VI s'appuie sur la plateforme PREDIS MHI pour valider les processus de transformation. Il détaille les transformations d'un modèle descriptif vers un modèle d'optimisation PLNE puis vers un modèle de simulation pour le recuit simulé.

# Chapitre I

# Gestion énergétique optimisée des systèmes bâtiments : Revue de littérature

L'architecture obligatoirement centralisée choisi pour GHomeTech impose une densité importante de modèles liés par les interactions qu'il y a entre leurs variables. Les modèles touchent plusieurs aspects de l'habitat en raison du choix d'objectifs économique et confort combinés. Il est ainsi nécessaire d'avoir une description thermique de l'enveloppe, une description technico-physique des équipements, une description économique des réaction de ces équipements et une description du confort des occupants en plus de leur planning et d'autres grandeurs intervenant dans le système. L'espace de contrôle est aussi un élément pouvant ajouter de la complexité car l'ensemble des systèmes doit être modélisé afin que le BEMS puisse prendre en compte les différentes influences. Les contrôles type On/Off sont différents des contrôles continus en terme de modélisation et nécessitent des types de variables différents. Cette disparité de types de variables impose des méthodes de calcul et d'optimisation adaptées et qui nécessite des approches de modélisation spécifiques. La PLNE, choisi dans le cadre GHomeTech, impose une modélisation sous forme de contraintes. Ces contraintes ne peuvent contenir que des équations ou des inéquations algébriques linéaires. Enfin, l'horizon temporel et le pas d'échantillonnage sont très influent sur la complexité du modèle global résultant. Ils constituent un coefficient de démultiplication du nombre de variables qui constitueront le modèle final, lequel sera traité par le solveur. Il est donc nécessaire d'avoir la bon calibrage de cet aspect afin d'avoir un modèle résoluble dans un temps définit.

Nous proposons dans ce chapitre une revue d'autres solutions présentes dans la littérature scientifique selon un ensemble d'aspects entrant en jeu dans la complexification de la modélisation et des modèles. Nous conclurons par une synthèse de ces aspects et une mise en évidence d'autres éléments qui nous amèneront à la problématique traitée dans cette thèse.

## I.1 Aspects utiles d'analyse des systèmes de gestion énergétique

Nous avons choisi d'analyser la littérature traitant de la gestion dans le bâtiment sous différents aspects selon les informations disponibles et nos besoins de positionnement

scientifique. Le premier aspect de cette étude concerne l'objectif recherché par les concepteurs du système de gestion proposé. L'objectif définit le reste des autres critères. En effet, l'architecture, les méthodes d'optimisation ainsi que l'horizon de traitement dépendent de l'objectif. L'architecture est établie en fonction de cet objectif d'une part et de la configuration matérielle du système à gérer d'autre part. Nous nous attarderons, par la suite, sur les différentes typologies, leurs avantages et leurs inconvénients. L'espace de contrôle permet de décrire les types de contrôles applicables en fonction des équipements à contrôler. Cette description est primordiale dans le choix de la méthode de résolution. La méthode de résolution définit souvent le langage de modélisation lorsque celle-ci nécessite des modèles explicites. Dans d'autres cas, les modèles sont intrinsèques aux commandes. La résolution consiste alors en une vérification des conditions d'exécution des commandes en fonction des données mesurées. L'horizon temporel sur lequel est appliqué la solution a aussi son importance dans notre classification car il s'agit de distinguer entre vision à cours terme et vision à long terme. Les deux visions peuvent se superposer dans certaines solutions telles que (HA 2008) ou (LEFORT et al. 2013). Nous examinerons par la suite une série de travaux selon ces différents aspects avant de discuter des liens entre cette vision et le déploiement à large échelle.

## I.1.1 Objectif recherché

La gestion énergétique dans le bâtiment peut concerner un équipement en particulier ou un ensemble d'équipements. On parle alors de gestion dédiée ou globale. Certains objectifs de gestion nécessitent une vision globale afin d'atteindre de bonnes performances (HAGRAS et al. 2008) alors que d'autres objectifs sont dédiés à un équipement (ARI et al. 2005).

On peut classifier les objectifs suivant deux aspects :

— économique

— confort

Les objectifs évoluent avec la disponibilité technologique. Les capacités de calcul (UPTON et al. 2013) ainsi que les différents moyens de communication (JANG et al. 2010) permettent des recueils de données et des moyens de contrôle jusqu'ici impossibles ou très peu viables économiquement.

### I.1.1.1 Objectif *économique*

Certains systèmes de gestion dans le bâtiment ont vocation à réaliser des économies de coûts énergétiques (AGGARWAL et al. 2009). Ces systèmes prennent en compte le coût de l'énergie, le confort des occupants, la disponibilité des ressources énergétiques et le comportement thermique du bâtiment dans l'élaboration de plans de gestion (GEORGOPOULOU et al. 1998) qui sont les consignes des asservissements. Ce type de gestion peut s'appuyer sur de simples heuristiques (STORN 1997) ou sur des outils plus évolués tels que des algorithmes d'optimisation basés sur des modèles du système ou des prédictions (HA 2008).

L'objectif économique, dans le cas d'un tarif variable de l'énergie électrique, vise à limiter l'utilisation d'énergie pendant les périodes où le tarif est élevé au profit d'une utilisation durant les creux tarifaires. Si le tarif est fixe alors l'objectif peut être simplement la diminution de la consommation dans le respect des contraintes imposées. Dans le cas d'une gestion énergétique incluant des équipements de production d'énergie élec-

trique locaux tels que les panneaux PV, le solaire thermique ou tout autre type d'énergies renouvelables. Il existe trois scénarios optimaux selon les conditions :

**La revente de la production** Les prix variables peuvent amener à une situation dans laquelle il est plus intéressant de revendre l'énergie produite que de l'auto-consommer ou de la stocker. C'est principalement le cas dans un scénario de forte intégration de différents systèmes de production intermittents ou ces derniers doivent s'entre compenser (DAS et al. 2005). L'équilibre entre les différentes sources oriente alors le tarif et limite ainsi la consommation pour l'ensemble lorsqu'une des sources est limitée.

**Le stockage** Si ce moyen est disponible, les résultats seront encore différents : c'est un degré de liberté supplémentaire. Les systèmes de gestion énergétique préconisent cette solution lorsque la revente est impossible ou que la perspective de stockage promet un meilleur rendement. Un gestionnaire énergétique évolué peut prendre en compte le vieillissement des systèmes de stockage (SARJEANT et al. 1997)

**L'autoconsommation** S'il n'y a pas de moyens de stockage ou alors si le bâtiment ne peut être connecté à un réseau externe (isolement, choix délibéré,...) (BRAUN et al. 2009), le gestionnaire ne jouera que le rôle d'équilibrage réseau. Il pourra couper les moyens de production si l'énergie produite ne peut être consommée. Si le gestionnaire possède des capacités anticipatives alors le système pourra agir sur des dérives afin d'accumuler de l'énergie sous forme de surchauffe ou de surrefroidissement en prévision d'une température extérieure faible et d'une diminution des niveaux de production d'énergie par exemple.

#### I.1.1.2 Objectif *confort*

L'objectif confort pour la gestion dans l'habitat est lié à la notion de *smart-home* dans la littérature (SPIGEL 2005). Le *smart-home* est un concept qui fournit des outils pour capter le maximum d'informations dans l'habitat pour divers usages (BIEN et al. 2008 ; YERRAPRAGADA et al. 1993 ; MONCRIEFF et al. 2007). L'usage premier des smart-homes est le monitoring. Il s'agit de solutions de recueil d'informations à travers des réseaux de capteurs, couplées à des IHM ergonomiques tout en apportant un service à l'occupant. Ce type de solutions permet à des utilisateurs d'avoir une vision globale de leur habitat et donc un éclairage dans les prises de décision. Une couche supplémentaire peut permettre l'utilisation de ces informations pour commander des équipements d'une manière automatique afin de répondre aux exigences des occupants. Les technologies *smart-home* visent des catégories particulières telles que les personnes handicapées ou âgées. (ALLEN 1996 ; PARK et al. 2007 ; LEE et al. 2008) proposent un système qui récolte de l'information. En se basant sur ses propres modèles et cette information, il communique avec l'occupant et des centres d'aide distants pour apporter de l'aide quotidienne, prévenir les accidents et si nécessaire, prévenir les secours.

### I.1.2 Architecture

Nous entendons par architectures, l'organisation des solutions de gestion et la production de l'information dans les BEMS. L'information peut être liée aux capteurs ou générée par des algorithmes. Nous distinguons deux types d'architectures logicielles :

**architectures centralisées :** C'est une architecture qui concentre les informations dans un même composant logiciel (KASTNER et al. 2005 ; EHLERS et al. 1996 ; HUANG et al. 2004). Dans certaines approches, l'architecture centralisée est néanmoins compartimentée ou modularisée. Cette approche consiste en la dissociation des différentes fonctions du système logiciel de gestion pour une meilleure visibilité. Cette approche facilite l'évolution et la maintenance. L'atout principal de ce type d'architecture réside dans le compromis entre rapidité et globalité d'une solution optimale. L'architecture centralisée permet un usage plus simplifié de modèles de description sans la problématique d'interfaçage.

**architectures distribuées :** Une architecture distribuée est une topologie qui consiste en la gestion de composants logiciels autonomes par un composant orchestrateur. Ce type d'architecture fait appel à des algorithmes d'optimisation distribuée. Ceux-ci présentent l'avantage de décomposer le besoin de calcul et de réduire les capacités nécessaires à l'usage. Dans certains cas, la complexité augmente d'une manière exponentielle avec le nombre de variables à traiter du fait du besoin de conditionnement entre composants logiciels. La décentralisation permet alors de traiter le problème partie par partie et ainsi réduire le temps de calcul. La distribution offre ainsi une meilleure flexibilité en termes de changement et d'adaptation a posteriori. Elle est aussi plus facilement extensible mais ne concentre pas l'information. Ce qui peut être problématique lorsqu'il s'agit d'algorithmes d'optimisation qui ont besoin d'une information complète pour établir des résultats exhaustifs (GLOUDEMAN et al. 2000 ; CEBASEK et al. 2000).

### I.1.3 Espace de contrôle

Un système de gestion énergétique s'inscrit dans un espace de contrôle. C'est-à-dire des moyens et des types de contrôles possibles. Les équipements peuvent être commandés de manière discrète, en tout ou rien (On/Off) (SALSBURY 1998 ; THOMAS et al. 2005), en modes (un nombre fini de commandes possibles) ou de manière continue (ANG et al. 2005 ; DIRECTIONS 2005).

Le contrôle On/Off est assez répandu (SALSBURY 1998). Il s'appuie sur des technologies électriques, mécaniques ou thermiques tels que les relais. Cette technologie est simple à mettre en place, peu onéreuse et robuste mais n'offre que peu de possibilités de réglage.

Le contrôle continu, souvent qualifié de régulation, repose principalement sur des contrôleurs de type PID. Ce contrôle consiste en la mise en oeuvre d'une boucle fermée constituée du processus à contrôler (équipements et composants d'enveloppe dans le cas du bâtiment), de capteurs, d'actionneurs et d'une loi de régulation. Lorsqu'un équipement est commandé en continu, les contrôleurs de type PID (KAYA et al. 2007) peuvent être mis en oeuvre à travers des systèmes électriques, mécaniques, pneumatiques ou hydrauliques. Cette commande peut être aussi mise en oeuvre sur des cartes électroniques ou sur des calculateurs. Le contrôle continu est sensible aux bruits de mesure (KETATA 1992) ainsi qu'aux erreurs de modélisation et de paramétrisation (KELMAN et al. 2012). Des stratégies de contrôle de type PID en cascade ont été développées (KAYA et al. 2007) pour contrer ce problème.

L'espace de contrôle continu est l'asservissement, c'est-à-dire la poursuite de valeurs de consigne. Ces consignes sont pré-établies pour garantir le bon fonctionnement de l'équipement commandé et en tenant compte du confort et de la consommation éner-

gétique. C'est la gestion énergétique la plus élémentaire qui peut être mise en oeuvre. Cette commande est généralement dédiée à un seul équipement et constitue une brique essentielle dans la gestion énergétique.

### I.1.4 Méthodes de résolution

Nous trouvons un large panel d'outils et de méthodes d'optimisation utilisés dans la gestion énergétique du bâtiment. Il existe deux principales approches qui peuvent être complémentaires :

**la résolution directe :** Il s'agit d'approches à base de régulateurs et des règles heuristiques qui définissent directement les commandes à appliquer en fonction d'une situation donnée. L'objectif à atteindre n'est pas formalisé dans le but de rechercher les commandes à appliquer mais les règles vont tenter d'améliorer directement les performances du système. Les règles ainsi construites doivent être suffisamment génériques pour améliorer les performances de la plupart des situations rencontrées.

**la résolution inverse :** Ces approches s'appuient sur une formulation explicite du problème de gestion à résoudre incluant comportement du système et objectif à atteindre. L'inversion du problème est réalisée au moyen d'algorithmes d'optimisation. Les commandes à appliquer au système sont ainsi calculées. Les solutions peuvent ainsi être adaptées aux situations réelles et prédites. C'est la raison pour laquelle les *energy Smart-Homes* vont plutôt recourir à ce type de résolution.

Lorsqu'il s'agit de résolution inverse, le problème constitué par la description des éléments et de l'enveloppe doit être décrit selon des formulations correspondant aux méthodes de résolution envisagées. Les variables utilisées dans ces descriptions sont aussi conditionnées par le type de résolution choisi.

Nous classifions les méthodes de résolution selon deux grandes familles. Ces familles sont définies par l'approche de description du problème à résoudre. Un problème peut faire intervenir des variables déterministes exclusivement ou alors des variables stochastiques (HA et al. 2008 ; ALLISON 2004). L'existence ou non de ce dernier type de variables dans un problème définit sa nature :

**problème déterministe :** Un problème déterministe est un problème pour lequel les variables admettent une unique valeur possible dans un domaine de valeur.

**problème non-déterministe :** La résolution non déterministe ou stochastique est utilisées lorsqu'il y a plusieurs valeurs possibles pour une même variable.

Le traitement stochastique n'est pas l'objet de notre travail. Il ne sera donc pas développé dans le détail. Le reste des descriptions concerne, par conséquent, uniquement les approches déterministes.

Les performances des méthodes de résolution à travers les algorithmes d'optimisation dépendent de la nature des problèmes déterministes à résoudre (KOLDA et al. 2003 ; POSTOLACHE 2007) :

**problème linéaire :** Le traitement d'un problème passe par une description des éléments le composant au moyen de modèles. Une formulation linéaire, lorsqu'elle est possible, permet de résoudre de manière exacte des problèmes de grande, voire très grande dimension. Une formulation linéaire se fait parfois au prix d'approximations afin de rendre le problème résoluble par les outils de résolution linéaires.

**problème non-linéaire :** Une formulation non-linéaire conduit généralement à une solution optimale exacte si le problème est convexe mais, dans le cas général, il n'existe aucune garantie et les problèmes de grande dimension sont généralement très difficiles à résoudre. Il est très fréquent d'obtenir des optimums locaux. L'utilisation de solveurs non-linéaires intervient lorsque la linéarisation n'est pas possible ou que l'écart entre une formulation linéarisée et une formulation non-linéaire conduit à une mauvaise précision des résultats. Certaines approches de résolution non-linéaires se basent sur des linéarisations locales successives.

Que le problème soit linéaire ou non, la rapidité de résolution ainsi que l'assurance de convergence dépendent de la nature et de la largeur des domaines de valeurs possibles pour les variables.

Il existe trois familles de variables selon leur nature (LOVÁSZ 2010) :

**variables continues :** L'algorithme du simplex et l'algorithme du point intérieur sont les plus couramment utilisés pour le traitement des problèmes linéaires à variables continues. Ses approches sont rapides et peu onéreuses en temps de calcul. Elles permettent d'appréhender un nombre important de variables : la complexité n'est pas une réelle difficulté pour la résolution. Si le problème est non-linéaire, on aura plutôt recours à des algorithmes de descente de type Programmation Séquentielle Quadratique par exemple, ou à des méta-heuristiques.

**variables discrètes :** La procédure de séparation évaluation (*branch and bound*) est l'algorithme le plus souvent utilisé. C'est un algorithme de recherche combinatoire basé sur des successions d'estimations et des coupes (*cuts*) de certaines voies de recherche liées à des valeurs de variables. La méthode s'applique autant aux problèmes linéaires que non-linéaires.

**variables continues et discrètes :** Les solveurs mixtes ont recours à des approches de résolution hiérarchisées qui combinent alternativement des approches dédiées aux variables continues et d'autres aux variables discrètes.

Parmi les méthodes de résolution linéaires ou non linéaires, la présentation de l'objectif du problème d'optimisation est important selon que le problème est mono ou multi-objectifs(WANG et al. 2009 ; LOKEN 2007). Lorsqu'il y a plusieurs objectifs dans un problème d'optimisation, il existe plusieurs types d'approches pour résoudre ce problème (BELTON et al. 2002) en fonction de la forme du résultat escompté :

**Résolution multi-objectifs** (HALLIKAINEN et al. 2002 ; SRINIVAS et al. 1994 ; OOKA et al. 2009). L'optimisation multi-objectif consiste à résoudre des problèmes d'optimisation sous contraintes selon des critères différents. La solution globale se présente sous forme d'un espace de solutions à n-dimensions définies par les $n$ objectifs à minimiser. Le contour de cette solution, quand elle existe, est appelée *front de Pareto*. Tous les outils d'optimisation ne permettent pas de trouver la totalité de ce front mais essayent de trouver des points d'appartenance caractéristiques. Il existe deux grandes familles d'algorithmes de résolution, les résolutions automatiques et les résolutions assistées. Dans la résolution automatique, l'algorithme se charge d'apporter une solution au problème multi-objectif agrégés sans intervention humaine. La résolution assistée exploite des informations qui peuvent être apportées par un expert. Les informations peuvent être apportées avant le début de l'optimisation. Dans ce cas, l'algorithme de résolution est dit *a priori*. Ces informations peuvent alors correspondre à des points du front de Pareto à calculer. La méthode de calcul consistera alors à itérer l'une des méthodes que nous

détaillerons plus bas. L'expertise peut être aussi apportée après résolution en exploitant le front de Pareto pour choisir une solution unique parmi l'ensemble des solutions possibles, il s'agit d'algorithmes de résolution *a posteriori*. Des algorithmes interactifs proposent une solution de chaque itération. L'expert intervient au fur et à mesure pour orienter l'algorithme pour l'itération suivante.

**Résolution à objectifs pondérés :** (DOUKAS et al. 2007 ; AFGAN et al. 2002 ; AFGAN et al. 2004 ; AFGAN et al. 2008 ; GOUMAS et al. 2000) ont adopté une stratégie d'optimisation utilisant un objectif constitué d'une somme pondérée de sous-objectifs. Il s'agit donc in fine d'une résolution mono-objectif. Cette approche a l'avantage, d'une part, d'offrir plus d'opportunités de résolution car la plupart des optimiseurs ne résolvent que des problèmes mono-objectifs, et d'autre part, de requérir moins de temps de calcul car l'optimisation multi-objectif requière souvent plusieurs itérations d'optimisation. La solution obtenue coïncide en réalité avec une solution particulière du front de Pareto d'une optimisation multi-objectif considérant les sous-objectifs pondérés par des poids fixes. Cette optimisation présente l'avantage de conduire directement à des stratégies de gestion énergétique sans nécessiter de décision humaine a posteriori comme cela doit être le cas en multi-objectif. La difficulté de cette méthode est qu'il est difficile de donner un sens absolu aux pondérations des sous-objectifs. Celles-ci doivent être définies arbitrairement puis ajustées après coup en fonction des résultats obtenus.

**epsilon-contrainte :** C'est une méthode itérative de resserrement de bornes sur les objectifs et en cherchant à chaque itération une solution acceptable(LAUMANNS et al. 2006). Cette méthode permet ainsi de constituer le front de Pareto partiellement selon les besoins.

### I.1.5 Horizon temporel

Tout système de gestion énergétique se base sur de l'information en entrée pour émettre des décisions en sortie. Selon l'horizon de temps choisi pour la collecte des informations, on recense trois grandes familles d'outils de gestion énergétique.

**gestion réactive :** La gestion réactive considère le court terme pour générer des commandes rapides. L'asservissement est un type de gestion réactive. Les boucles PID (HOLMBERG 2001) sont largement utilisées dans le contrôle des équipements présents dans le bâtiment. Les boucles Tout Ou Rien TOR sont aussi un moyen de commande des systèmes à événements discrets (MARINAKIS et al. 2013). L'ensemble des moyens d'asservissement permet un contrôle autonome suivant des consignes prédéfinies par les utilisateurs ou par les gestionnaires d'espaces. Les contraintes liées à ce type gestion sont d'ordre technique ou les prérogatives d'usage de chaque équipement doivent être intégrés dans les modèles d'asservissement. Un autre type de gestion réactive dans le bâtiment développé est basé sur des règles heuristiques. C'est de niveau supérieur à l'asservissement. Les règles heuristiques sont établies par des experts pour satisfaire des objectifs propres de confort, de coût ou mixtes. Cette même approche réactive a été utilisée pour faire le lien entre une gestion anticipative et l'asservissement des équipements (HA 2008). C'est une manière indirecte de satisfaire les objectifs précédemment cités.

**gestion prédictive :** La gestion prédictive contrairement à la gestion réactive se base sur des prévisions de comportement du système pour calculer une stratégie de

gestion (KOLOKOTSA et al. 2009). Pour ce faire, des algorithmes de prévision ou des modèles physiques sont utilisés. Les algorithmes de prévision peuvent s'appuyer sur des procédures d'apprentissage utilisant des données d'historiques et des données présentes remontées par les capteurs. Les modèles sont basés sur des structures pré-établies et dont certains paramètres sont ajustés. L'approche anticipative offre l'avantage d'utiliser les connaissances acquises à travers les modèles physiques, météorologiques, comportementaux ainsi que les plannings organisationnels. L'approche anticipative n'est pas toujours déterministe et associe parfois une notion de confiance à chaque grandeur anticipée. Différents moyens ont été développés pour traiter cette problématique d'anticipation. (BASU et al. 2013) propose une solution à base d'un classificateur et de caractéristiques suggérées par un système nommé *oracle* pour réaliser des prédictions sur l'usage des équipements électroménagers. La prédiction se résume à prévoir si l'équipement étudié sera allumé ou éteint dans l'heure à venir.

(CETIN et al. 2014) propose une étude qualitative sur les tendances d'utilisation de certains équipements électroménagers. Cette étude a été réalisée sur la base d'un échantillon de 40 maisons à Austin. Les résultats sont proposés sous la forme d'une base normative pour la prédiction des usages.

(EDWARDS et al. 2012) compare différentes méthodes de *machine learning* pour la prédiction la consommation d'un habitat à H+1 avec des données capteurs reçues toutes les 15 minutes. L'application est réservée au bâtiment résidentiel car selon l'auteur la méthode efficace pour la prédiction dépend de l'usage du bâtiment.

(TSANAS et al. 2012) présente un modèle statistique utilisé pour prédire la charge de chauffage et la charge de refroidissement à partir des huit paramètres intrinsèques à l'enveloppe thermique parmi lesquels la compacité, la surface du sol, des murs, du vitrage, du toit, l'orientation et enfin la masse.

## I.2 Travaux existants

Dans cette partie nous analysons certains travaux de la littérature scientifique afin de mettre en évidence les liens entre les aspects évoqués et le déploiement à large échelle.

**Nan2013** a développé un algorithme de gestion d'un équipement de régulation d'humidité. Le travail présenté concerne une roue disséquante. Cet équipement est intégré dans un système d'échange entre l'air intérieur et extérieur afin de réguler l'humidité de l'air ambiant. L'objectif est clairement le confort même s'il y a une contrainte de rationnement énergétique. Cette dernière vise à limiter les pertes et non à gérer efficacement la consommation énergétique. L'architecture n'a pas été spécifiée dans la description mais la limitation du champ d'action du système au seul système d'hygrométrie montre une architecture centralisée. Aucune connexion n'est prévue dans le travail avec d'autres gestionnaires en vue de la constitution d'un système complet. L'algorithme de résolution choisi est le MPC (Model Prédictive Control). L'approche est prédictive et adaptative. L'adaptativité est là une forme de réactivité aux erreurs de prédiction. La séquence de commande anticipée est réinitialisée à chaque pas de temps. Le pas de temps choisi est d'une heure pour un horizon de 10h soit dix pas de temps. Le problème contient alors 80 variables et nécessite 4 minutes de calcul chaque heure sur une machine de calcul intel core2 duo 2.5 GHz.

**Zhu2013** a développé un algorithme de gestion du niveau du CO2 dans un système

de ventilation forcée. L'objectif de l'étude mixe entre confort et économie. Un modèle simple établi ainsi le lien entre le taux de ventilation, l'occupation et le niveau de CO2 en fonction des paramètres fixes de l'enveloppe. Le coût de ventilation est associé au modèle pour calculer l'impact économique du contrôle. Les déperditions de chaleur ont été négligées en considérant une récupération des calories par échange statique entre flux entrants et sortants. Le système se focalise sur la seule gestion du CO2 et d'une architecture complètement centralisée. Une approche d'optimisation appelée essaim de particules a été utilisée pour trouver les taux de CO2 optimaux. Un contrôleur basé sur la logique floue exécute par la suite les commandes pour avoir les taux décidés. L'horizon de commande choisi est de 24h pour un pas de temps de 15 minutes. Des paramètres spécifiques à la méthode ont été choisis afin de garantir la convergence de l'optimisation.

(LEFORT et al. 2013) a développé une approche plus globale mais décentralisée dans le but de gérer le confort et la rationalisation énergétique à la fois. Une architecture décentralisée est basée sur des modules d'optimisation s'imbriquant pour gérer d'une manière locale chaque système physique. Les pas de temps et les horizons dépendent du niveau. La résolution au niveau de chaque module est basée sur le MPC.

(HA 2008) et (LE MINH et al. 2011) ont développé le système GHomeTech, point de départ de cette thèse, Le système est basé sur une structure centralisée cloisonnant les différents systèmes sous la forme de services. L'approche PLNE pour la résolution du problème choisi a imposé l'architecture car il s'agit là de composer un problème complet avant de le soumettre à l'outil d'optimisation qui lui est externalisé : CPLEX ou GLPK. L'horizon de 24h et le pas d'échantillonnage de 1h ont été imposés par la nature du problème et les dynamiques traitées. Le système a pour objectif de minimiser la consommation tout en maximisant le confort. Le système a aussi pour objectif de maintenir l'équilibre entre offre et demande d'électricité à l'échelle d'un habitat doté d'équipements de production renouvelable.

(LE et al. 2014) propose un système de gestion des occultants basé sur une approche hybride de commande prédictive MPC et branch & bound. Nous estimons que cette approche a pour but le confort de l'occupant. Le modèle est basé sur la luminosité et la température extérieure ainsi qu'une pénalité sur le mouvement des stores. L'objectif est alors de minimiser l'écart entre une quantité de lumière jugée idéale et la lumière entrant. De même pour la température tout en minimisant le nombre de mouvements des stores. L'horizon de temps est de 24H, le standard en gestion énergétique. Le travail est, là encore, ponctuel et ne touche qu'un équipement particulier en l'occurrence les occultants.

(ZAMORA-MARTÍNEZ et al. 2014) propose un système de gestion dans le bâtiment qui se base essentiellement sur la fiabilité des données récoltées et prédites. Il propose l'application de méthodes d'apprentissage partant d'une structure de modèle avec un jeu de paramètres. Ces paramètres sont initialisés par de la connaissance experte ou pour un jeu de paramètres complètement aléatoire. Un réseau de capteurs à l'intérieur et à l'extérieur de l'habitat transmet de l'information à un algorithme d'apprentissage. Deux méthodes d'apprentissages ont été comparées sur plusieurs types de structures de modèles. Une méthode utilise un algorithme du gradient et l'autre une méthode bayésienne. La fiabilité des données recueillies permet alors d'établir des conseils aux occupants assez précis, ce qui conduit, selon l'étude, à 30% d'énergie économisée grâce à ce procédé. Globalement, le système nécessite 4 à 5 jours d'horizon de temps pour l'apprentissage. L'objectif de ces travaux est double : économie et confort à travers de conseils.

Les travaux (DOUNIS et al. 2009) correspondent à une architecture distribuée industrielle basée sur des automates programmables. La gestion est basée sur des règles heuris-

tiques, simples à programmer dans un automate programmables (PLC). Cette démarche a fait ses preuves dans le tertiaire où de nombreux espaces sont souvent similaires : la gestion peut ainsi être découpée en sous-problèmes de gestion comparables. Le déploiement de masse de ces solutions est bien maitrisé dans le secteur socio-économique. Nous retrouvons le matériel de base des systèmes de contrôle/commande tels que les capteurs, les concentrateurs (hub) et les automates programmables (KNIBBE 1996). Cet ensemble associé à une gestion à base de règles heuristiques constitue ce qui est communément nommé la Gestion technique de Bâtiment (GTB) telle qu'elle a été développée au début des années 90. L'architecture des GTB n'est guère flexible (RIGOS et al. 1993 ; VIRK et al. 1990), tant d'un point de vue matériel que logiciel et de bonnes performances sont difficiles à atteindre en ne s'appuyant que sur des règles heuristiques s'appliquant à la plupart des contextes d'occupation, de météorologie, de coûts énergétiques rencontrés. Ces approches sont peu ergonomiques et requièrent certaines compétences techniques pour (re-)configurer les automates programmables. Ces systèmes GTB sont principalement installés dans le tertiaire, là où l'usage est planifié et routinier et où l'intervention fréquente de techniciens est économiquement acceptable. L'habitat a contrario évolue plus rapidement à travers des besoins nouveaux liés à des changements de situation familiale, de mode de vie ou tout simplement d'humeur.

(JUN et al. 2011) présente une solution qui a été implémentée à partir d'un modèle de bâtiment connecté à un réseau électrique à tarif variable avec des panneaux photovoltaïques et une batterie ainsi qu'une mini-éolienne. La consommation du bâtiment est modélisée par un élément appelé charge. L'objectif est de minimiser le coût de la facture énergétique en respectant la contrainte d'équilibre des flux électriques en utilisant comme levier la seule production et le stockage de l'énergie ainsi que l'interaction avec le réseau mais sans moduler les charges.

(KING et al. 2005) allie l'assistance aux personnes âgées et à motricité réduite avec la gestion énergétique dans l'habitat. Le confort est mis au premier plan tout en exécutant des heuristiques pour économiser l'énergie. Cette approche est intéressante car elle allie des objectifs de sécurité, de confort et d'économie.

(DOUKAS et al. 2007) intègre dans sa fonction objectif une variable représentant le confort thermique. Cette variable est calculée à travers le modèle analytique PMV (FANGER et al. 1970). Le principe du PMV (Predicted Mean Vote) est d'associer une valeur moyenne comprise entre $0.5 \leq PMV \leq 0.5$. Celle-ci es tirée de l'équation de Fanger qui associe les paramètres : température ambiante, température moyenne radiante, humidité relative, vitesse de l'air, ratio métabolique et l'isolation vestimentaire. Cette méthode a été reprise dans l'ASHRAE qui est la norme principale utilisée aux USA pour quantifier le confort de l'occupant. Concernant la production de chaleur, des convecteurs électriques sont représentés par une égalité entre l'énergie électrique prélevée et la chaleur produite. La gestion des équipements est alors intégrée dans des règles heuristiques qui comparent les PMV calculés par rapport à des seuils et enclanchent des actions de façon à ramener le PMV vers $0$.

## I.3 Problématique générale et motivation de la thèse

Les aspects de traitement choisis permettent de développer une vision sur la possibilité de déploiement ainsi que la scalabilité des travaux.

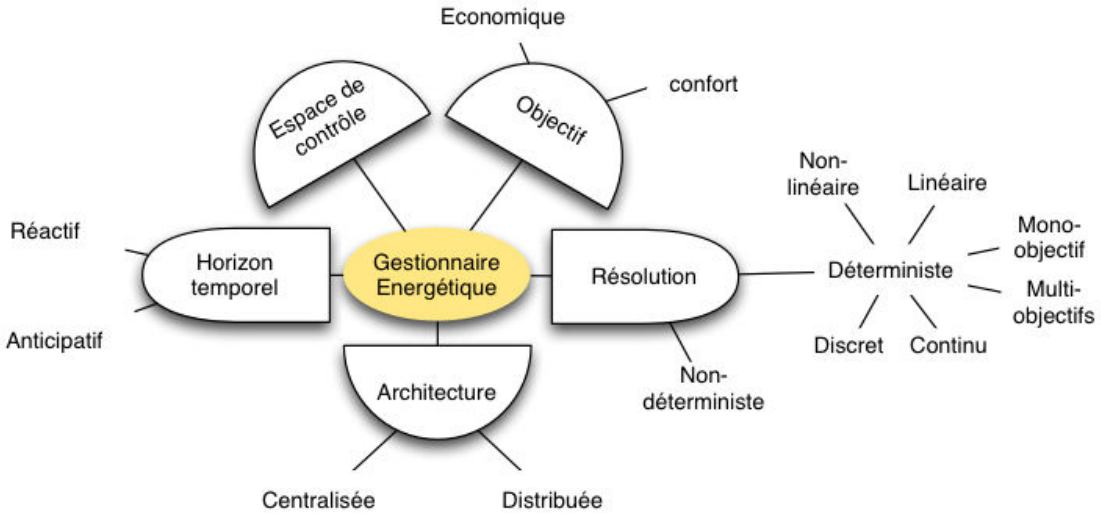



FIGURE *I.1 – Synthèse des points de vue*

Le schéma I.1 synthétise les différentes dimensions dans lesquelles s'inscrivent les gestionnaires dans le bâtiment. Certaines portent clairement la mention "gestion énergétique dans le bâtiment" ou son acronyme anglais BEMS (Building Energy Management System) tandis que d'autres sont qualifiées d'approches smart-homes ou encore smart-buildings. Dans le cas des BEMS, l'économie est clairement affichée comme objectif mais rarement dans le smart-home. Le confort reste dans le smart-home l'objectif principal. Les différents aspects (architecture, résolution et horizon temporel) donnent une vision globale sur les perspectives possibles à ces travaux de recherche.

Les travaux **Zhu2013** ; **Nan2013** concernent des équipements particuliers qui réagissent en autonomie par rapport au reste du bâtiment. L'anticipation dans ce cas présente un problème lié à l'incertitude d'occurrence d'événements environnants liés aux autres équipements, à l'enveloppe et aux usagers. L'anticipation devient alors caduque à chaque occurrence non attendue. Le MCP répond en partie à cette problématique à travers le calcul d'un nouveau vecteur de commande à chaque pas de temps mais nécessite de grandes capacités de calcul **Nan2013** qui ne sont, pour l'instant, envisageables que sur des implémentations ponctuelles sur des modèles maitrisés.

La communauté du smart-grid propose à travers des approches tel que le demand-response des solutions facilement déployables et dont la complexité est bien moins importante que pour les solutions BEMS à l'échelle de l'habitat. A cette échelle, ces solutions consistent en l'installation de délesteurs d'équipements (chauffage et ECS) lesquels sont actionnés par le gestionnaire de réseau à distance. Cette solution présente néanmoins un inconfort pour certains utilisateurs et présage ainsi d'un manque d'acceptation. Récemment, le compteur Linky a commencé a être déployé sur le territoire par ErDF et les premières plaintes et refus d'installation ont suivi aussitôt. Le refus est, le plus souvent, justifié par la peur de la violation de la vie privée d'une part et par la peur de la perte de contrôle d'autre part. Il nous parait donc nécessaire d'apporter des solutions plus locales afin que l'utilisateur final reste maître de ses équipements. Les BEMS peuvent encore offrir cet avantage à condition que ces derniers puissent démontrer la viabilité économique de leur déploiement à large échelle. La complémentarité entre les BEMS et le réseau peut être donc envisagée à condition que les BEMS démontrent une possibilité d'industrialisa-

tion.

La viabilité économique et l'industrialisation sont rarement étudiées dans les développements d'approche BEMS. Les études sont présentées sur un cas unique, physique ou simulé. Le résultat est apprécié à travers la comparaison des temps de calcul par rapport au réferentiel. Le passage du prototype qu'on peut qualifier de preuve de concept vers un système robuste, scalable, et déployable de manière économiquement raisonnable n'est pas clairement abordé. Le délai de développement du modèle, susceptible de constituer un frein à la généralisation de l'approche, n'est pas vraiment documenté. Les habitats ne se ressemblent pas par l'enveloppe, les équipements installés ainsi que les usages globalement mais au mieux par certaines parties. Les modèles sont donc établis en fonction des spécificités de chaque habitat puis testés avant d'être formatés selon l'approche de résolution choisie dans le BEMS, PLNE en l'occurrence pour GhomeTech. Le portage du BEMS d'un habitat à un autre d'un point de vue modèles, même structurellement identiques, nécessite des paramètres particuliers liés à l'environnement, à l'agencement et aux dimensions qui ne sont que très rarement identiques.

Le problème que nous proposons de traiter dans cette thèse concerne le développement et la manipulation des modèles dans les différentes phases de conception du BEMS en vue de la réduction du délai de développement global. Notre approche consiste à appréhender la complexité de modélisation en raison des différentes natures de variables utilisées, de la multitude de sous modèles à lier et des transformations nécessaires pour l'adaptation aux spécificités imposées par l'algorithme de résolution du problème décrit par ces modèles. Les deux chapitres suivants traiteront de la manipulation de modèles dans la phase de leur développement pour l'optimisation PLNE à travers la présentation d'un outil dédié. Il s'agira d'un outil limitant la complexité induite par le traitement brut des modèles en offrant une interface de simplification permettant de faire le lien entre une vision de développement de modèles naturelle qui peut être celle d'un physicien et le langage informatique qui peut lui être étranger s'il n'a pas de double compétence. Les trois derniers chapitres étendront la vision de simplification de développement en proposant une plate-forme de transformation de modèles entre les applications simulation, estimation paramétrique et optimisation. Ces trois applications entrent en jeu dans le développement des modèles pour le BEMS. L'approche a été imaginée pour éviter la répétition de développement de modèles pour chacune des applications séparément,. Ainsi un gain de temps et une meilleure fiabilité des modèles sont espérés.

# Première partie

# Gestion de la complexité liée au nombre

# Chapitre II

# Outil de développement de modèles

Dans le précédent chapitre, nous avons passé en revue la littérature scientifique et mis en évidence que les solutions de BEMS proposées étaient mal adaptées à la résolution de problèmes de gestion énergétique complexes alors que la tendance laisse présumer une complexité croissante des systèmes bâtiments. Dans ce chapitre, nous analyserons l'approche GhomeTech par rapport aux exigences relevées dans le chapitre précédent. Nous formulerons par la suite la problématique scientifique telle que constatée dans l'analyse. Nous proposerons une solution puis nous conclurons ce travail avant de passer au chapitre d'application.

# II.1 Approche de modélisation par service

La notion de *service* est l'unité de base de modélisation dans GHomeTech. Cette notion a été initiée dans la thèse de (HA 2008) puis reprise dans les thèses de (WARKOZEK 2011). Cette entité reflète une tâche accomplie par un équipement ou un ensemble d'équipements au service de l'occupant.

Un *service* est lié d'un coté à l'énergie consommée par celui-ci pour accomplir la tâche et de l'autre coté à une métrique de satisfaction de l'usager. Cette dernière est facultative mais l'ensemble trouve une justification cohérente d'un point de vue optimisation. Lorsqu'un algorithme d'optimisation est appliqué sur un problème de gestion énergétique dont le seul objectif est de minimiser la consommation d'énergie, la solution évidente est l'extinction de l'ensemble des équipements susceptibles d'accroitre cette consommation. Les métriques de satisfaction permettent de modéliser l'appréciation sur le service rendu. cette appréciation est pondérée par des modèles spécifiques tel que PMV (FANGER et al. 1970). En associant ces métriques aux consommations énergétiques dans un objectif commun, on aboutit alors à un modèle d'optimisation complet. Dans certains cas, les modèles n'ont pas d'impact négatif sur la consommation énergétique, c'est le cas notamment des moyens de production ou encore le modèle d'équilibre du réseau électrique domestique. Ces modèles sont alors dépourvus de métrique de satisfaction mais restent néanmoins liés à des services.

Un service est aussi lié à une série d'attributs permettant de le classifier dans l'une des catégories génériques de modélisation pour l'optimisation :

**Selon son comportement dans le temps :** permanent ou temporisé.

**Selon sa flexibilité :** décalable, interruptible, interruptible avec contraintes ou modifiable.

**Selon la contrôlabilité :** contrôlable ou non contrôlable.

L'application de GhomeTech sur un cas concret a porté sur un habitat spécifié dans le projet Multisol (BIBLIOGRAPHIE BACHA et al. 2006). Ce projet a apporté un cadre de déploiement pour la solution logicielle.

Cet habitat (figure II.2) est constitué de 5 espaces. Il est doté d'équipements standards (figure II.1) que sont : lave-vaisselle, lave-linge, chauffage électrique et un moyen de production d'eau chaude sanitaire. Pour la fourniture d'énergie électrique, l'habitat est raccordé au réseau publique et dispose d'une capacité de production PV de 147m2.

La modélisation a été scindée en 5 services :

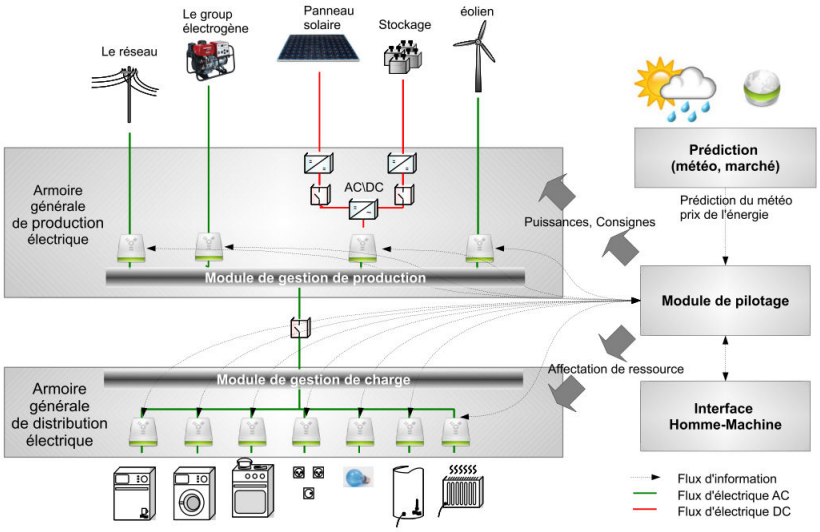



FIGURE *II.1 – Equipements du périmètre théorique de déploiement de GHomeTech*


FIGURE *II.2 – Plan architectural de l'enveloppe thermique décrite pour GHomeTech*

### II.1.1 Service chauffage

$$\begin{bmatrix} \frac{dT_m}{dt} \\ \frac{dT_a}{dt} \end{bmatrix} = \begin{bmatrix} \frac{-1}{r_i C_e} & \frac{1}{r_i C_e} \\ \frac{1}{r_i C_i} & -\frac{1}{r_a C_i} - \frac{1}{r_i C_i} \end{bmatrix} \begin{bmatrix} T_m \\ T_a \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{r_a C_i} & \frac{1}{C_i} & \frac{W}{C_i} \end{bmatrix} \begin{bmatrix} T_{ext} \\ \phi_r \\ \phi_s \end{bmatrix}$$

Avec

- $T_m$ la température de l'enveloppe de la pièce
- $T_a$ la température ambiante de la pièce
- $T_{ext}$ la température extérieure
- $\phi_r$ le flux thermique généré par le radiateur
- $\phi_s$ le flux énergétique apporté par le rayonnement solaire
- $C_e$ la capacité thermique de l'enveloppe de la pièce
- $C_i$ la capacité thermique du volume d'air dans la pièce
- $r_i$,$r_a$ les résistances thermiques
- $W$ la superficie de la fenêtre

$$\begin{aligned} |PMV(T_a(i,k))| &= \delta_a(i,k) \times 0,5(T_a(i,k) - T_{opt}) + (1 - \delta_a(i,k)) \times 0,5(T_{opt} - T_a(i,k)) \\ &\quad \forall i \in \{1,...,5\},\ k \in \{0,...,18,17,...,24\} \\ &= F_1\delta_a(i,k) + F_2T_a(i,k) + F_3T_a(i,k) \times \delta_a(i,k) + F_4 \\ &\quad \forall i \in \{1,...,5\},\ k \in \{0,...,18,17,...,24\} \\ &= F_1\delta_a(i,k) + F_2T_a(i,k) + F_3z_a(i,k) + F_4 \\ &\quad \forall i \in \{1,...,5\},\ k \in \{0,...,18,17,...,24\} \end{aligned}$$

$$U(i) = \sum_{k=1}^{K} \frac{|PMV(T_a(i,k))|}{24} \ \forall i \in [1,...,5]$$

Ce service est assez complexe à gérer une fois le modèle établi car il contient au moins 1085 symboles. Ce chiffre a été calculé en incluant les variables pour chaque pas de temps et les paramètres. A défaut d'un outil d'aide, le modèle a été introduit manuellement dans le système de gestion lequel, rappelons le, intègre les modèles dans le corps du code.

### II.1.2 Service eau chaude sanitaire

$$E_e = 2,0 + 1,3 \times N = 2,0 + 1,3 \times 5 = 8,5kWh$$

$$\sum_{k=1}^{K} E(6,k) = E_e$$

$$E(6,k) \leq 1500 \forall k \in [1,...24]$$

Le service ECS est un service relativement petit en terme de nombre de symboles il n'en contient que 26 au moment de l'intégration du modèle.

### II.1.3 Service Lave-linge

$$\begin{aligned} s(7,2) &= f(7,1) \\ s(7,3) &= f(7,2) \end{aligned}$$

$$U(i) = \delta_u(7) \times (f_{opt}(7) - f(7,3)) + (1 - \delta_u(7)) \frac{f(7,3) - f_{opt}(i)}{6}$$

$$s(7,1) \in [f_{min}(7) - d(7,3) - d(7,1),\ f_{max}(7) - d(7,3) - d(7,1)] = [7,5h, 14,5h]$$

$$f(7,1) \in [8,5h,\ 15,5h]$$

$$E(7,j,k) = \begin{cases} E'(7,j,k) = P_{max}(7)\,(\mathrm{Min}[f(7,j),(k+1)\Delta] - \mathrm{Max}[s(7,j),k\Delta)] & \text{Si} \quad E'(i,j,k) > 0 \\ 0 & \text{Si} \quad E'(7,j,k) \leq 0 \end{cases}$$

$$E(7,j,k) = \begin{cases} E'(7,j,k) = P_{max}(7)\,(\mathrm{Min}[f(7,j),(k+1)\Delta] - \mathrm{Max}[s(7,j),k\Delta)] & \text{Si} \quad E'(i,j,k) > 0 \\ 0 & \text{Si} \quad E'(7,j,k) \leq 0 \end{cases}$$

$$E(7,j,k) = (1 - \delta_{t3}(i,j,k))E'(7,j,k)$$

$$E(7,k) = \sum_{j=1}^{3} E(7,j,k) \forall k \in [1,...,K]$$

Ce service intègre deux aspects. Un aspect temporel pour la gestion du timing de démarrage et du séquencement des étapes d'exécution de la tâche et un autre aspect pour l'estimation de la consommation à travers le calcul des énergies sur chaque pas de temps. L'ensemble totalise 252 symboles à manipuler pour le seul modèle du lave-linge.

### II.1.4 Service lave-vaisselle

Un modèle similaire au modèle du lave-linge est associé à ce service. Néanmoins, au moment de l'introduction du modèle, il faut arriver à nommer différemment les variables et les paramètres du modèle de lave-linge afin de déclarer clairement que les modèles sont distincts par l'instanciation. Cette procédure rajoute donc 252 symboles à manipuler.

### II.1.5 Service ressources électriques

$$E(9,k) \leq 6 \forall k \in [1,...,K]$$

$$E(10,k) + E(9,k) = E(8,k) + E(7,k) + E(6,k) + E(5,k) + E(4,k) + E(3,k) + E(2,k) + E(1,k)$$
$$\forall k \in [1,...,K]$$

Cette partie reprend plusieurs autres variables des autres modèles, il est donc important de respecter la nomenclature. Les symboles propres à cette partie concernent la production et l'énergie fournie par le réseau. L'énergie produite est considérée comme donnée liée à la météo fournie avant résolution tandis que l"énergie prélevée du réseau est considérée variable et est définie par l'optimisation. Le nombre de symboles nécessaire est de 48 sur cette partie.

### II.1.6 Les objectifs

$$J_2 = \sum_{k=1}^{K} (C(9,k)E(9,k) - 0.3E_{ex}(k))$$

$$J_1 = U(1) + U(2) + U(3) + U(4) + U(5) + U(6) + U(7)$$

L'objectif final regroupe les deux objectifs : économique et confort. Chacun des deux est une agrégation des appréciations remontées par chaque service. Le confort est généré par les fonctions d'estimation pour chaque service lorsque celui-ci en est doté. L'objectif économique est lié à l'énergie consommée par un prix unitaire qui peut être variable au cours du temps. Divers scénarios peuvent être envisagés dans les pondérations pour le couplage des objectifs et sont d'une importance capitale (WARKOZEK 2011). 75 symboles interviennent dans la modélisation et le calcul de cet objectif.

## Discussion

Les services décrits ci-haut présentent des modèles relativement simples. L'ensemble de ces modèles mobilise pas moins de 1738 symboles au total entre paramètres et variables d'optimisation. Le modèle a été entièrement intégré au code à la main. Les paramètres ne peuvent être que manuellement modifiés pour tester différents scénarios.

(WARKOZEK 2011) a proposé d'adjoindre une interface de développement au gestionnaire énergétique. Cette interface basée sur le formalisme XML permet aux déployeurs de la solution d'apporter ou de modifier le jeu de paramètres sans avoir à accéder aux modèles physiques car ces paramètres n'existent pas textuellement mais sont directement insérés dans le code informatique de l'application. Même si le modèle utilisé parait représentatif d'un habitat standard. Il est principalement question dans un travail de recherche d'anticiper les habitats à venir. Des équipements tels que les échangeurs de chaleur air/air, des chaudières à bois ou biocarburants ainsi que les systèmes de diffusion latente. Ces équipements offrent plus de moyens de contrôle et une meilleure efficacité énergétique. Prenons exemple du chauffage, lequel est de loin le plus gros consommateur d'énergie dans un habitat standard. Le modèle de chauffage utilisé dans GhomeTech est un modèle de résistance électrique or aujourd'hui, l'inefficacité de ce mode de chauffage n'est plus à prouver tant l'ensemble des pertes subies dans la chaine de transformation et de transport de l'énergie nécessaire jusque dans l'habitat. L'énergie est finalement extraite de chaleur pour à la fin être transformée en chaleur. A contrario, des systèmes de type cogénération ou tout simplement chaudière gaz ou à bois reliés à des systèmes de diffusion à fluide caloporteur présentent un meilleur bilan d'efficacité énergétique. D'autre part, les bâtiments dit basse consommation s'appuient sur une isolation thermique. De fait, le chauffage occupe une proportion de consommation énergétique bien moindre que dans un habitat traditionnel. L'ensemble des équipements présents rejette de la chaleur par pertes. Ces pertes sont si importantes proportionnellement qu'elles influent sur la température dans l'habitat. Ces pertes doivent être prises en compte. Enfin les apports métaboliques des occupants seront aussi non négligeables dans le bilan énergétique de l'habitat.

La ventilation n'a pas été intégrée dans ce test, ce qui aurait d'avantage compliqué la tâche en augmentant le nombre de symboles même dans sa simple formulation théorique qui considère que la ventilation n'influe que sur le taux de CO2. Cette approche néglige les pertes énergétiques liées au remplacement de l'air intérieur par l'air extérieur surtout par temps extrême dans des maisons hautement isolées. Ces pertes sont tellement importantes qu'il est tacitement intégré dans le standard des concepteurs de maisons basse consommation l'intégration d'un échangeur statique Air/air. L'approche consiste à faire passer les flux d'air sortants (tempérés) par des échangeurs qui récupèrent une partie de l'énergie vers les flux d'air entrants. Cette approche réduit l'impact de la ventilation sur la température ambiante dans sa version statique. Certains systèmes offrent la possibilité de pilotage pour une meilleure efficacité. Ce pilotage nécessite un modèle comportemental

supplémentaire.

Une grande partie de l'efficacité de GHomeTech repose sur les leviers utilisés dans l'habitat contrôlé. Chaque levier est lié à un modèle comportemental du système physique. Ce modèle relie l'action sur le levier au résultat attendu. D'autre part, l'habitat est équipé de systèmes répondant à ses besoins selon le climat, la position géographique, les moyens financiers mis à disposition et l'usage du bâtiment en question. Il est nécessaire, comme exprimé dans le chapitre précédent, d'avoir une solution de gestion énergétique déployable facilement et à moindre coût pour espérer une diffusion large. Cette solution doit être adaptable aux spécificités de chacun des habitats à équiper. Pour répondre à cette contrainte,la solution GHomeTech a dû embarquer le minimum de leviers comme montré dans son application afin d'être déployable sur un large spectre de types d'habitats. Or, cette approche est spécialement non-viable sur les habitats basses consommation du fait de leurs caractéristiques singulières d'adaptation à leur environnement et de la sensibilité de ces habitats à des paramètres bien plus complexes que les habitats traditionnels.

Dans la section suivante, nous proposons d'étudier les solutions existantes avec un point de vue gestion de la complexité de modélisation. Nous proposerons une revue de littérature des outils faisant intervenir des moyens d'appréhension telle que l'approche orientée objet ou encore les approches bibliothèques utilisées dans les outils de simulation avant de proposer un moyen adéquat pour la gestion des modèles pour GhomeTech.

## II.2 D'autres approches d'association de modèles

A l'image d'un puzzle, un modèle est composé d'une multitude d'éléments. Un modèle est un ensemble de contraintes (équations et inéquations) et de variables décrivant les relations existant entre des phénomènes physiques. Un phénomène physique est associé à un ou plusieurs symboles représentant les grandeurs variables ou fixes.

Le choix de découpage d'un modèle est stratégique car il conditionne la possibilité de réutilisation. Prenons l'exemple d'une chaudière à gaz de la figure II.3. Celle-ci est considérée dans certains logiciels de simulation comme une unité de modélisation. Le modèle mathématique correspondant est directement utilisé pour fournir des informations tels que la température de sortie de l'eau en fonction de la température d'entrée, du débit d'eau et de l'énergie utilisée pour chauffer. Si un développeur de modèles était appelé à modéliser un tel équipement, il s'apercevrait qu'il est composé de trois sous-systèmes : une chambre de combustion, un échangeur de chaleur air/eau et une éventuelle pompe intégrée. Ces trois éléments peuvent être retrouvés dans d'autres modèles d'équipements à certaines variantes près. Les convecteurs électriques ont un système de résistances semblables à celui d'une chaudière électrique d'un point de vue modélisation :

L'échangeur de chaleur type air/eau peut être aussi rencontré dans des systèmes de ventilation équipés de cassettes d'eau chaude dans la figure II.4. La pompe à eau électrique est un élément souvent retrouvé dans les systèmes de chaleur à fluide caloporteur.

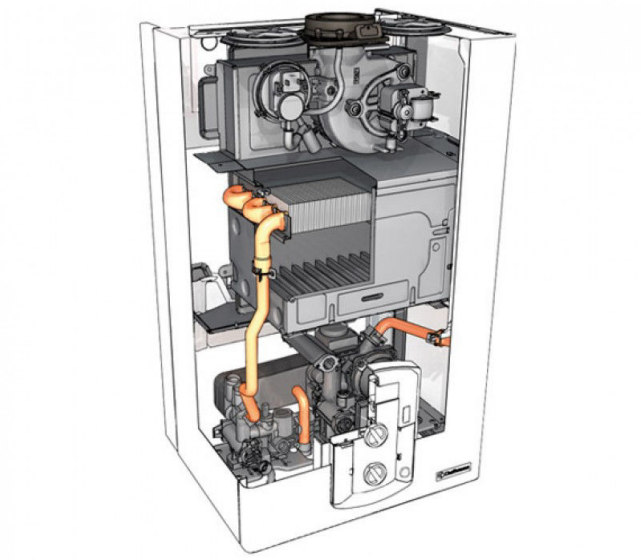


FIGURE *II.3 – chaudiere à gaz*

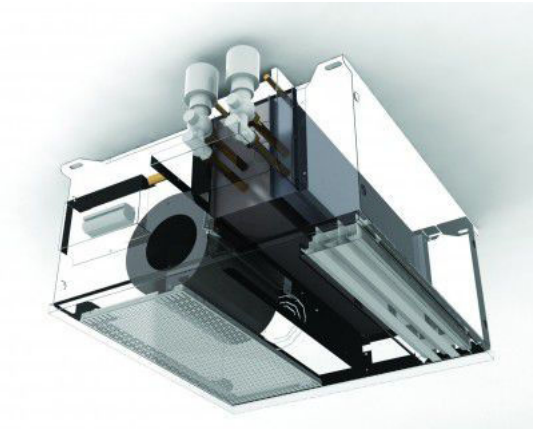


FIGURE *II.4 – cassetteEchangeur*

A travers cet exemple, nous pouvons déduire que si une bibliothèque d'éléments de modèle était constituée alors nous pourrions construire le modèle de la chaudière juste en composant et en adaptant les paramètres. L'idée de la modularité vient donc de ce besoin de capitalisation de l'information et de réutilisation pour éviter les doublons.

## II.2.1 Gestion de la complexité par le paradigme objet

Avant d'explorer la modélisation orientée objet, explorons la programmation orientée objet qui en est l'inspiration. Le problème de la complexité s'est posé très tôt en programmation informatique. Au début des années 60, Ole-Johan Dahl et Kristen Nygaard ont élaboré un paradigme pouvant répondre à cette problématique et ainsi mieux organiser la programmation qui n'était qu'à ses débuts.

Le paradigme objet consiste à encapsuler dans des mêmes unités les données et les algorithmes qui s'y appliquent : c'est la notion d'objet. A l'instar des échangeurs de chaleur, il peut exister de nombreux objets ayant les mêmes caractéristiques, mais dont les valeurs diffèrent. La notion de type d'objet a été introduite : on parle souvent de classe d'objets qui s'instancie en des objets en affectant des valeurs aux caractéristiques (ou attributs). De

plus, les classes d'objets peuvent être spécialisées. Par exemple, la classe *pompe à chaleur* peut se spécialiser en la classe *pompe à chaleur air/eau* qui hérite des caractéristiques de *pompe à chaleur* tout en y ajoutant des caractéristiques propres. Depuis les années 60, ce paradigme a été largement repris et implémenté. Aujourd'hui, il est à la base de la majeure partie des architectures logicielles développées. Il présente l'avantage d'être intuitif car il permet une vision structurée comme l'est le monde réel, hiérarchisée et globale.

Apparue au début des années 1990, la modélisation orientée objet (MOO) (ELMQVIST et al. 1993) est le pendant de la programmation orientée objet. La MOO permet de structurer la description des modèles physiques de la même façon que la POO a permis d'assurer une structuration au développement logiciel. Par la structuration en "objets" permettant de créer des unités de modélisation, l'information est capitalisée sous une forme intelligible. De plus, la prise en compte des notions d'héritage et d'agrégation permet d'enrichir la base de connaissance par itérations successives.

#### II.2.1.1 Héritage

Comme pour la POO, il est possible de créer un modèle nouveau à partir d'un modèle existant via le mécanisme de spécialisation ou d'héritage. Outre la structuration naturelle de la connaissance rendue possible par ce paradigme, cela permet au concepteur de construire rapidement un nouveau modèle par modification d'un modèle existant en favorisant la réutilisation des éléments de modèles.

#### II.2.1.2 Agrégation

L'agrégation est un moyen de construire un modèle complexe à partir de modèles élémentaires existant dans une bibliothèque de modèles. Le concepteur dispose d'un ensemble de "briques" de modélisation élémentaires dont l'utilisation lui permet de construire un modèle plus complexe sans se préoccuper de la modélisation des briques élémentaires.

#### II.2.1.3 Domaine d'utilisation

L'utilisation actuelle de ces techniques de modélisation se développe dans le domaine de la simulation temporelle (THEVENON 2000 ; FÁBIÁN et al. 1998 ; ELMQVIST et al. 1993). Cette technique n'est néanmoins pas très utilisée dans le domaine de l'optimisation. En particulier dans le bâtiment car jusqu'à présent les approches se voulaient standards ou particulières à des prototypes spécifiques mais pas dans un esprit de réutilisation pour appréhender la complexité, objectif poursuivi ici.

### II.2.2 Urbanisation informatique

L'urbanisation d'un système d'information ou d'une organisation est une discipline de l'ingénierie informatique consistant à faire évoluer son système d'information (SI) pour qu'il soutienne et accompagne de manière efficace et efficiente les missions de cette organisation et ses transformations (SASSOON 1998).

En urbanisation, les dépendances doivent respecter les notions de cohérence forte et de couplage faible entre les unités :

— au sein d'une application : entre les différents modules
— au sein d'un module : entre les différents composants

La notion de cohérence forte / couplage faible indique que deux applications doivent communiquer entre elles de façon simple et efficace, mais que la dépendance entre ces deux unités est minimale (idéalement inexistante). Cela permet donc de retirer un bloc pour le remplacer sans perturber le reste du système d'information.

Le système d'information peut donc être comparé au quartier d'une ville : si ce dernier est bien bâti et bien urbanisé, il est possible de détruire un bâtiment au cœur du quartier sans mettre en péril tout le secteur, et de le remplacer par un autre édifice, en le raccordant aux différents réseaux d'échanges : voirie d'accès, électricité, évacuation des eaux usées, etc. L'urbanisation consiste donc à créer un SI agile, modulable et évolutif.

## II.2.3 Composition dans les environnements de simulation

### II.2.3.1 Matlab / Simulink (Mathworks)

Matlab (*mathworks*) est un environnement de développement dédié à des calculs numériques couvrant des applications variées dont essentiellement la conception des systèmes de commande ainsi que le traitement du signal et d'images. Matlab s'est enrichi avec le temps et propose aujourd'hui trois approches de modélisation-simulation :

**Le script Matlab :** C'est la méthode originelle qui permet de poser des modèles mathématiques de phénomènes physiques dans différents domaines.

**La méthode graphique Simulink :** Une approche graphique du script Matlab qui permet en théorie de mieux maitriser la complexité au développeur de modèles. Cette approche tout comme l'approche script sont des méthodes dites causales. C'est à dire que les modèles doivent obligatoirement obéir à un ordre temporel de résolution. A défaut de quoi, il peut y avoir des formations de boucles algébriques qui empêchent la résolution numérique basée sur des équations différentielles.

**Approche Simscape :** Se définit comme une alternative aux deux approches causales à travers un paradigme de modélisation orienté physique et A-causal. Cette approche permet à Matlab une meilleure communication avec des plate-formes externes.

Nous avons choisi d'étudier plus profondément Simulink car nous estimons que ce dernier est le plus pertinent pour nous apporter des éclaircissements sur le traitement de la complexité dans la modélisation. Simulink offre une interface graphique pour la modélisation. L'interface met à disposition une bibliothèque de blocs triés par métiers en plus de la section principale *Simulink* contenant les blocs élémentaires II.5.

Le bloc est l'unité de modélisation dans Simulink. Chaque bloc est associé à une boite de dialogue pour le parametrage et dans certains cas pour le développement de script Matlab.

A la jonction entre blocs de modélisation, il y a la notion de branchement. Cette notion est la clé de la transmission d'informations à travers un signal entre deux blocs distincts. Le branchement se fait à travers des ports. Le signal qui passe à travers ces ports d'entrée et de sortie est orienté avec un sens possible unique de transmission et typé sur la nature de l'information transmise. Le signal est propagé dans le modèle global depuis l'entrée des données externes jusqu'au résultat de simulation. Cette méthode permet une décorélation de calcul et facilite ainsi la montée en puissance dans la complexité des modèles. C'est une approche dite causale. Néanmoins, elle pose le problème de la boucle algébrique. C'est une erreur que retourne Simulink pour signifier que l'ordre de transmission des informations n'est pas cohérent avec l'ordre des calculs. L'exemple typique qui provoque

cette erreur est le branchement d'une sortie d'un bloc à son entrée sans élément intégrateur (retardateur de signal).

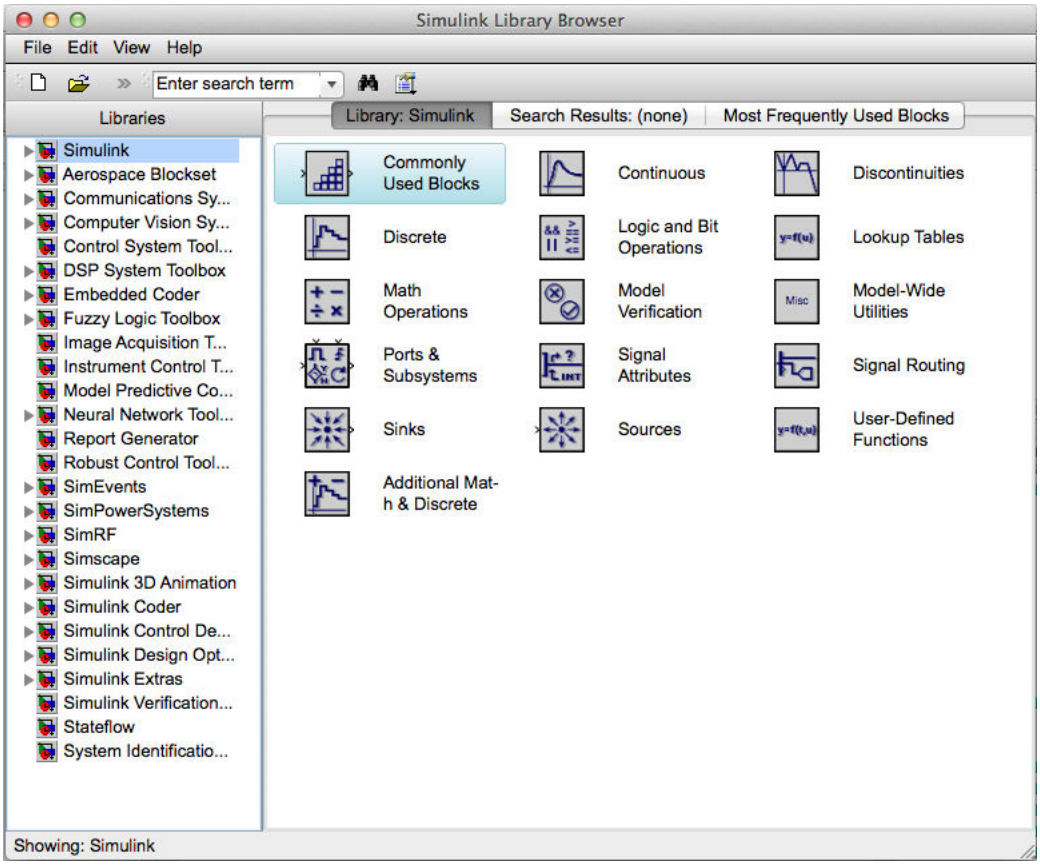



FIGURE *II.5 – Boite de dialogue simulink*

Enfin, il existe un bloc de modélisation *S-function* qui permet d'embarquer différents types de scripts (C++, VHDL, Matlab,etc.) Ce bloc est traité de la même manière que les autres blocs dans la composition et au moment de la résolution.

En ce qui concerne la gestion et la composition de modèles, Simulink offre trois possibilités : l'encapsulation, la composition et la gestion de versions.

**L'encapsulation** permet le stockage d'un modèle dans une capsule équivalente à un bloc. Cette capsule est pourvue d'entrées, de sorties et de boites de dialogue pour la paramétrisation si nécessaire. Celle-ci peut s'utiliser dans tout autre modèle au même titre qu'un bloc prédéfini.

**La composition** est le mécanisme qui consiste à gérer la profondeur d'encapsulation, c'est à dire qu'un modèle peut être composé de sous modèles lesquels sont eux mêmes composés de sous modèles, etc.

Enfin, **la gestion de versions** à travers le bloc *Variant subsystem* permet de faire porter dans une même composition différentes variantes. Au moment de simuler, il faut néanmoins fixer la variante à utiliser. Cette méthode permet de gérer les versions, les variantes ou les différents jeux de paramètres utilisables dans un même modèle de base.

### II.2.3.2 Modelica

Le langage Modelica est un langage dédié à la simulation de systèmes physiques complexes et multidisciplinaires. C'est un langage "open source" qui est régi par l'organisa-

tion "Modelica assocation". En outre, il s'agit d'un langage de modélisation orienté objet offrant la possibilité d'exploiter les concepts d'héritage facilitant ainsi la réutilisation d'éléments de modèle.

Modelica (FRITZSON 2010) est un langage multidisciplinaire pouvant couvrir plusieurs domaines physiques. La bibliothèque standard de Modelica couvre plusieurs domaines physiques : électrique (analogique et numérique), magnétique, mécanique, fluide et thermique.

Par son aspect orienté objet, Modelica choisit le paradigme de représentation de modèles. Néanmoins cette vue est limitée aux seules représentations causales car les implémentations de Modelica embarquent des modes de résolution différentiels. Les implémentations respectent la norme Modelica afin que les développeurs d'une plate-forme puissent passer à une autre facilement. De plus, les librairies deviennent portables et constituent une valeur ajoutée commerciale tout comme celle-ci peut être rétro-inspectée par leur utilisateur final qui peut y apporter ses changements. Cette approche séduit tant dans les milieux de la recherche que dans les milieux industriels car elle offre justement les deux avantages de la protection de la propriété intellectuelle et la possibilité d'introspection.

La structuration des modèles dans Modelica est plus orientée physique en comparaison avec Matlab dans lequel l'aspect mathématique prime. Dans Modelica, l'unité de modélisation est l'élément physique représenté. Les librairies sont regroupées en domaines et sous domaines. Contrairement à Matlab qui est dans une logique complètement verrouillée où il reste le seul à pouvoir fournir les librairies : Modelica est "open source". Il existe plusieurs éditeurs d'environnement de simulation Modelica mais le principal reste Dymola. L'architecture est accessible aux développeurs autres que ceux de l'éditeur. Les modèles ne sont certes pas validés mais peuvent être directement introspectés par les utilisateurs. Pour éditer un modèle, le développeur dispose des éléments suivants :

**Paramètres :** Les paramètres sont des éléments fixes pour la durée d'une simulation mais modifiables d'une simulation à une autre. Ils sont habituellement utilisés pour emmagasiner les constantes de calibrage des modèles et autres entrées fixes du système. Au niveau de l'initialisation des paramètres, deux possibilités sont disponibles :

- Initialisation directe : La valeur du paramètre est fournie directement à l'initialisation.
- Initialisation indirecte : La valeur du paramètre provient d'un autre ou d'une combinaison d'autres paramètres.

**Variables :** Les variables sont habituellement des entités dont la valeur n'est pas connue d'emblée pour l'ensemble de la simulation. On peut catégoriser les différents types de variables de la manière suivante :

- Variables à résoudre : Ce sont habituellement les variables différenciées. Ce sont les seules variables pouvant être initialisées. Les variables non-différenciées ne peuvent pas, quant à elles, être initialisées.
- Variables de sortie : Les variables dont on veut communiquer la valeur à l'extérieur de l'objet et qui sont associées à une sortie de l'objet.
- Variables d'entrée : Les variables d'entrées doivent être associées à une entrée de l'objet. Puisque la valeur initiale proviendra de l'extérieur, ces variables ne peuvent donc pas être initialisées.
- Variables calculées : Données calculées à partir des autres variables et paramètres de l'objet.

**Equations :** La section *equation* du code est l'endroit où apparaissent les lois décrivant le fonctionnement du système à représenter. Ces équations sont habituellement issues des lois de conservation de la matière et de l'énergie. C'est aussi l'endroit où l'on spécifie les règles de calcul des variables autres qu'à résoudre. Les énoncés d'interconnexion des composants du modèle font aussi partie de cette section. Afin d'avoir un système déterminé, il est important de spécifier le même nombre de variables à résoudre que d'équations. Autrement, la traduction du modèle à la compilation entrainera un message d'erreur de système singulier surspécifié ou sous-spécifié. Cette simple règle peut paraitre triviale en soit mais peut devenir un véritable casse-tête lorsque le nombre de variables augmente. L'implantation de la mise en équation dans un contexte matriciel est simplifiée car en plus des primitives adaptées de Modelica à cet égard, le formalisme de l'environnement de commande de Matlab peut être utilisé, à quelques exceptions près.

**Connecteurs :** La classe *connector* est utilisée pour définir les points d'interconnexion des composants d'un modèle. Dans Modelica, les deux principales catégories de connecteurs disponibles sont les connecteurs physiques (aussi appelés ports physiques) qui sont basés sur la circulation d'un flux d'énergie et les connecteurs d'entrées/sorties qui sont basés quant à eux simplement sur l'échange d'information.

— Connecteurs physiques : Ces connecteurs sont définis par un doublet de variables de flux et de potentiel. Par exemple, pour un connecteur dans le domaine de l'électricité, la variable de flux est le courant et la variable de potentiel est la tension. Selon les lois de la physique, une variable de type flux a une sommation à zéro à chaque point de connexion (en électricité cela correspond à la loi des nœuds de Kirchhoff) et les variables de type potentiel ont toutes la même valeur pour un point de connexion donné. Ce type de connecteur est celui qui permet une connexion de type acausal entre deux composants ou plus.

— Connecteurs d'entrée/sortie : Ces connecteurs sont définis simplement comme des entrées ou des sorties et sont donc, par le fait même, la destination ou la source de flux orientés. Ce type de connecteur est celui qui permet une connexion de type causale entre deux composants ou plus.

**Connexion :** La primitive *connect* est utilisée pour effectuer la connexion entre deux connecteurs ou plus et doit être spécifiée dans la section *equation* du code Modelica. Les connexions peuvent être soit causales ou acausales dépendamment du type de connecteur impliqué dans la connexion. Lorsqu'on utilise un environnement de développement avec interface graphique, les énoncés de connexion sont générés automatiquement au fur et à mesure que l'on procède à l'interconnexion de composants dans un modèle.

#### II.2.3.3 Composants logiciels : exemple Core-lab

Nous traitons maintenant une catégorie d'outils moins populaires que la MOO, Matlab ou Modelica mais qui présente aussi une façon de gérer les modèles particulière. Le concept de composant logiciel a vu son apparition dans l'ingénierie informatique au début des années 90 (MAURER 2000 ; SZYPERSKI 2002 ; BARBIER et al. 2002). Ce paradigme a été introduit après la programmation orientée objet pour dépasser ses limites en terme de décomposition et d'autonomie. Un composant logiciel est défini par Szyperski (SZY-

PERSKI 2002) en tant qu'«unité de composition qui est spécifiée, contractuellement, par ses interfaces et ses dépendances contextuelle explicites. Un composant logiciel peut être déployé indépendamment et est sujet de composition par des tierces entités.». Il s'agit donc :

— d'une entité autonome de déploiement, capable de s'exécuter indépendamment de l'outil de génération.
— d'une unité informatique encapsulant un code mais masquant son contenu en exposant uniquement des interfaces avec des ports d'entrée/sortie.
— d'une entité de composition capable de se connecter à d'autres composants via des ports de connexions (MEIJLER et al. 1997).

Dans les travaux (FISCHER et al. 2009), la norme ICAr, pour Interfaces for Component Architecture, a été proposée commme implémentation du concept de composants logiciels pour répondre aux besoins de modélisation et d'action sur des entités hétérogènes. Ces entités peuvent être des modèles mathématiques, des algorithmes d'optimisation ou autre. L'idée de Icar est d'encapsuler chaque entité dans un composant logiciel capable d'être utilisé par différents outils de simulation. Un standard de construction du composant a été défini pour que celui-ci soit identifié et que les services fournis par le composant soient identifiés. L'approche, intéressante d'un point de vue organisationnelle, a été reprise en partie par les travaux (CHOUIKH 2012) mais un problème ergonomique est notoire du fait de la proximité avec le langage informatique JAVA, plus difficile à prendre en main par un physicien habitué aux environnements de modélisation physique plus qu'informatique. Néanmoins, cette proximité traduit une motivation plus technique et moins orientée besoin du développeur de modèles. Core-lab est un outil développé au laboratoire G2Elab à Grenoble entièrement basé sur la norme Icar.

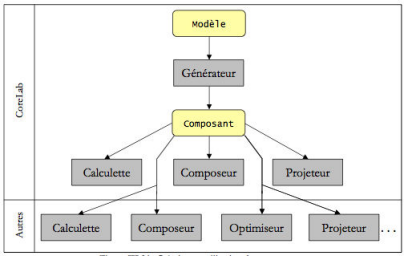



FIGURE *II.6 – architecture CoreLab*

L'architecture de Core-lab II.6 illustre la transformation du modèle formel vers un fichier binaire où le modèle source n'est plus accessible. Cette transformation induit un problème de perte de visibilité et donc d'informations sur le modèle de départ dans le cas de la modélisation. Les composants logiciels générés sont alors composables comme il pourraient l'être dans Simulink ou Dymola-Modelica mais malheureusement plus éditables.

### II.2.4 Gestion de modèles dans les outils d'optimisation

Il existe plusieurs approches de modélisation. Celle qui nous intéresse à tout point de vue dans cette thèse est l'approche PLNE. La modélisation en PLNE n'offre pas de moyens de décomposer un modèle. A vrai dire nous parlons de problème à optimiser. Ce problème est associé à un modèle indivisible car la résolution est A-causale, c'est à dire que l'algorithme de résolution est appliqué sur le modèle complet mis à plat.

Un modèle d'optimisation PLNE doit avoir trois sections d'information :

**la partie déclarative :** dans laquelle les variables, les paramètres et les ensembles de variation utilisés dans le modèle y sont déclarés.

**le modèle :** un ensemble d'équations arithmétiques a-causales définissant les rapports entre variables et paramètres sous forme de contraintes

**les données :** ce sont les paramètres définis en partie déclarative et utilisés dans le modèle qui sont instanciés dans cette partie.

### II.2.5 Conclusion

Le parcours de ces solutions hétérogènes en soi nous donne un aperçu des méthodes et moyens déjà en place pour la gestion de modèles. Cette problématique est récurrente dans bon nombre d'applications en commençant par le codage à l'optimisation en passant par la simulation. La modélisation orientée objet offre un paradigme de modélisation assez intéressant et fort pour modéliser plusieurs phénomènes. Néanmoins, cette approche trouve son intérêt principalement dans la communauté des experts en codage logiciel ou du moins qui ont un lien avec le développement informatique car le paradigme orienté objet est un paradigme fort qui nécessite une réflexion et une vision particulière.

La solution Simulink est une solution très intuitive car principalement graphique. Elle permet aux développeurs de modèles de se concentrer pleinement à leur tâche de modélisation physique en leur simplifiant la phase d'introduction du modèle. Néanmoins, cette approche est limitée au traitement des problèmes causaux dans sa pleine puissance. Les problèmes A-causaux ne peuvent être qu'encapsulés dans des blocs *C-functions* et de ce fait ne peuvent correspondre à notre problématique.

L'approche Modelica est plus textuelle (tout en gardant l'aspect ergonomique de l'interface graphique) mais là encore, cette approche est principalement développée pour la simulation.

Notre problématique concerne la gestion de modèles dans le cas de problèmes d'optimisation PLNE. Les outils de développement PLNE aujourd'hui présents sur le marché ne permettent pas autant d'agilité de manipulation de modèles que dans les précédents cas étudiés. Les plus développés permettent une séparation entre la structure du problème qui regroupe la partie déclarative et le modèle (contraintes) de la partie donnée qui peut être sujet à des changements et des modifications.

## II.3 Formulation du problème

Le problème auquel nous voulons apporter des réponses dans ce chapitre concerne la gestion des modèles dans GhomeTech. Ce besoin reflète la montée en complexité dans la modélisation dans une application : l'optimisation PLNE appliquée à la gestion anticipative dans le bâtiment. Cette complexité, bien que maitrisée jusqu'ici manuellement dans le projet Multisol, mobilise l'attention du développeur de modèles dans une proportion bien trop importante à nos yeux. Il sera de plus en plus difficile de prétendre proposer des modèles réalistes avec cette approche. Nous avons éprouvé un besoin d'aide à la tâche comme ce qui peut être déjà le cas dans les outils de simulation. La manipulation de scénarios, bien que améliorée par (WARKOZEK 2011) ne permet pas une vision ergonomique. Cette vision est très importante notamment dans le debogage des modèles qui nécessite une grande attention et une grande concentration sur le problème lui même.

Le problème consiste à combiner les avantages des solutions apportées par les outils de simulation puis les implémenter pour le développement des modèles d'optimisation, lesquels répondent à un paradigme particulier. Concrètement il s'agit de proposer un outil de traitement de modèles qui alimentera automatiquement le solveur PLNE de la couche anticipative de GHomeTech.

L'outil prendra en charge la gestion des compositions et certaines tâches répétitives dans le développement de problèmes PLNE jusque là réalisées manuellement.

Pour la composition, nous nous sommes inspirés de ce qui est mis en oeuvre dans Simulink à travers la composition et de la modélisation orientée objet. Cette approche consiste à considérer un modèle comme une classe et son utilisation comme une instance dans une autre classe. Néanmoins d'un point de vue simulink, il revient à considérer un modèle comme un composant utilisé dans d'autres. Nous devons offrir la pleine liberté d'utiliser les noms de symboles et de contraintes qui conviennent au développeur sans soucis de duplications de variables qui mènera à des connexions non-désirées. Lorsque le développeur veut établir des connexions entre variables, il pourra se servir d'un mécanisme de connexion.

La linéairisation de certaines formes présentes dans les modèles est une tâche très laborieuse et répétitive comme on le présentera dans la section "Patterns". En effet, il n'est pas simple d'avoir un modèle complètement linéaire. En modélisation pour la gestion énergétique dans le bâtiment, nous avons des modèles de représentation de phénomènes physiques comme le comportement thermique de l'enveloppe et des modèles d'équipements présents dans l'habitat. L'objectif des modèles de ces derniers est de lier l'effet de la commande sur les grandeurs physiques et la satisfaction des occupants par le service rendu. Nous utilisons souvent le produit entre la variable représentant la commande et la variable représentant la grandeur physique. Ce produit est pondéré par un poids qui permet de donner un sens physique au résultat du produit. Néanmoins, Ces produits génèrent souvent des combinaisons de variables d'optimisation non linéaires qu'il faut reprendre manuellement (figure II.1). Un traitement automatique évitera ce traitement source de confusion car le résultat devient moins lisible et permet aussi un gain de temps au développeur.

L'autre tâche récursive est la projection temporelle. Cette tâche consiste à exprimer toutes les contraintes des modèles sous une forme purement algébrique. La notion de temps n'existe pas en PLNE. La projection temporelle consiste alors à transformer les équations exprimées dans le temps grâce à des variables représentant chaque pas de temps, c'est une discrétisation temporelle.

Le débogage des modèles est aussi une tâche non négligeable. Il n'existe pas de méthode automatique pour ce faire. Néanmoins, un moyen intéressant consiste à modifier les domaines de variation des variables afin de deviner les problèmes à partir du comportement de la solution. Il s'agit ensuite de resserrage, libération voire de fixation de domaine de telle ou telle variable jusqu'à parvenir à un modèle au comportement cohérent.

Cette plateforme se doit d'offrir un langage d'interface entre le développeur de modèles et le solveur de problèmes PLNE. Comme nous l'avons vu jusqu'ici, la modularité dans le développement de modèles est la solution de gestion de la complexité. Les modèles doivent être complètement indépendants de GhomeTech, ce qui permettra à des non-initiés au développement informatique et précisément aux non-initiés au code de GhomeTech de manipuler des modélisations et leurs variantes rapidement. Ce langage impliquera une sémantique propre qui transformera les modèles développés en des problèmes d'optimisation PLNE, selon ce qui est voulu par l'utilisateur. C'est cette sémantique que nous

développerons dans ce qui suit.

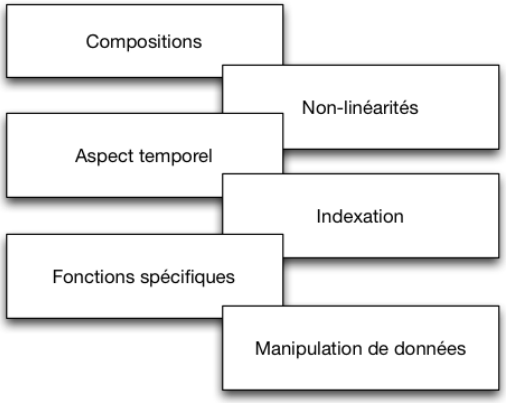



FIGURE *II.7 – Problèmes relevés dans l'implémentation GHomeTech*

## II.4 Formalisation de l'approche MILP

La gestion énergétique, telle que nous l'envisageons, est au carrefour entre deux disciplines : la modélisation et l'optimisation. La solution a donc pour ambition de rapprocher au maximum la programmation de modèles à ce qui se ferait dans un outil de simulation et d'adapter automatiquement ceux-ci aux exigences de l'optimisation.

Nous proposons à l'ensemble des problèmes soulevés une interface entre le gestionnaire énergétique GhomeTech et le développeur de modèles. Cette interface se matérialisera par un environnement de développement qui comprend un langage de développement et l'outil informatique qui traite les modèles ainsi développés. Cet environnement est baptisé "MILP-workshop". Il incombera au développeur de modèles de mettre en place les modèles de façon indépendante pour faciliter la réutilisabilité. La composition de ces modèles donne lieu à d'autres modèles. Les compositions pourront s'imbriquer jusqu'à ce qu'un modèle complet pour la gestion anticipative soit constitué. Cet agrégat de modèles est alors traité par MILP-workshop pour en faire un problème d'optimisation PLNE. Le langage n'est pas la priorité dans cette thèse. Par conséquent, nous avons choisi d'utiliser des mots clé à minima inspirés d'autres outils d'optimisation et de simulation. L'environnement sera accompagné d'une notice d'usage lorsque celui-ci sera diffusé.

Notre contribution principale dans cette partie de la thèse réside dans la mise en place des mécanismes de traitement des modèles ainsi que la mise en place des sémantiques liées aux mots-clés de développement afin de permettre au développeur de modèles de se concentrer sur sa tâche de modélisation. Nous discuterons dans cette section uniquement des principes de mise en oeuvre. La mise en oeuvre logicielle sera décrite dans le chapitre suivant.

Nous avons regroupé le traitement nécessaire en plusieurs groupes de tâches II.8 :

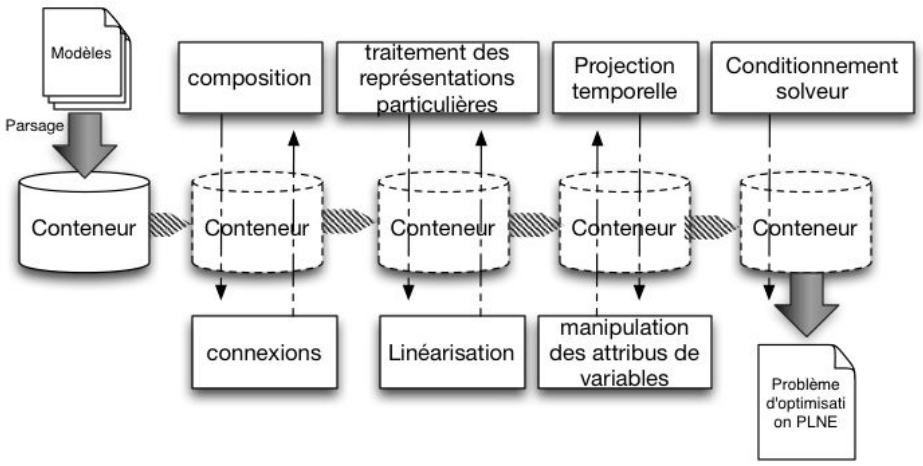



FIGURE *II.8 – Traitements de MILP-workshop depuis les modèles de librairie à un problème d'optimisation PLNE*

**constitution du problème d'optimisation** : C'est l'ensemble des étapes qui touchent à la mise en place du problème global.
- — parsage des modèles
- — composition
- — projection temporelle
- — conditionnement solveur

**traitement des symboles** : Regroupe l'ensemble des étapes de manipulation des attribues de variables et des paramètres.
- — connexions
- — typage
- — manipulation des domaines de variation

**traitement des contraintes** : Les contraintes sont traitées de façon à pouvoir déceler les formes non admises par les solveurs PLNE. En effet, seuls les formes polynomiales de premier degré sont acceptées.
- — linéarisation
- — traitement des représentations particulières

## II.5 Constitution du problème d'optimisation

Les tâches explicitées dans cette section touchent l'ensemble des parties des modèles. L'ordre chronologique d'exécution de chaque tâche dépend en partie de la formulation de modèles par le développeur et d'autre part, de l'ordonnancement des étapes pré-établi dans la figure II.8.

### II.5.1 Conteneur de modèles

Le conteneur est un principe emprunté au génie logiciel. C'est un objet informatique qui est chargé de contenir le modèle et de supporter les changements apportés par les différentes tâches. Le résultat final peut être prélevé afin d'être transmis à un outil tiers. Dans notre cas, cet outil est le solveur PLNE embarqué dans GhomeTech.

Le conteneur II.9 est chargé par le parseur à partir des modèles et des compositions du développeur de modèles. Le conteneur reprend la structure d'un problème d'optimisation PLNE . Il contient donc la partie symboles, la partie contraintes et la partie données. La partie variables regroupe toutes les informations nécessaires à chaque grandeur intervenant dans le problème : le nom du symbole, sa nature, son type, ses bornes inférieure et supérieure ainsi que ses préfixes. Cette notion de préfixes est relative à la composition et sera expliquée dans le paragraphe traitant de cette tâche. La signification de chaque attribut sera renseignée dans la sous section traitement de symboles.

La partie contraintes contient le nom, le préfixe, les expressions (gauche et droite) qui forment la contrainte ainsi que le lien entre les expressions (égalité ou inégalité). Cette partie associe aussi des tags de traitement à chaque contrainte tel que le tag de linéarité ou de temporalité.

La partie données charge toutes les données du problème liées aux paramètres.

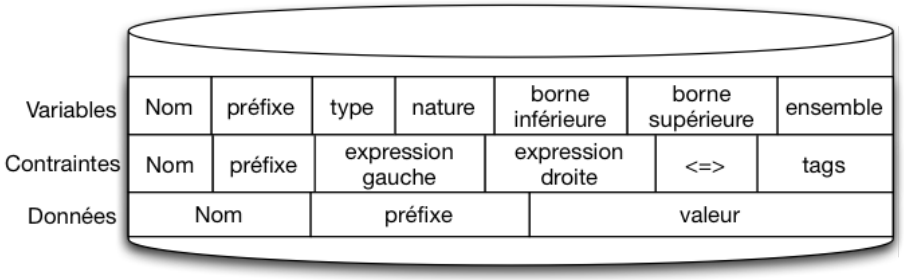



FIGURE *II.9 – Représentation d'un conteneur*

## II.5.2 Composition de modèles

La construction d'un modèle composé se fait brique par brique en y ajoutant le liant à chaque étape de composition. Ce liant peut être des connexions entre variables ou des ajouts de contraintes. La composition est le mécanisme clef pour concrétiser la notion de modularité. La modularité adoptée dans ce travail est issue du paradigme de modélisation orienté objet et s'inspire du langage Modelica. Un problème inhérent à la composition est le chevauchement des noms, que ce soit des noms de variables, des contraintes ou des paramètres. L'exemple de la figure II.10 montre ce problème par les deux modèles *model1* et *model6* qui sont des instances d'un même modèle à la base (mêmes variables et même fonction). Dans un modèle complet le même nom donné à deux variables peut porter à confusion car en réalité ce sont deux variables distinctes. Modelica s'inspire de la notion de classe et d'objet. Une description de modèle est considérée comme une classe et son utilisation dans une composition ou dans une instance d'exécution est une instance de cette classe. L'appel à un modèle dans un autre modèle (composition) se fait à travers une instanciation du modèle dans la composition. Cette instanciation prend la forme du choix du modèle à travers son nom de fichier accompagné du nom d'instance choisi. La nomenclature des variables, contraintes et paramètres de l'instance prend comme préfixe le nom de l'instance dans la composition. Ainsi le système peut permettre deux instances d'un même modèle dans une même composition sans créer d'interférence entre les deux. Dans notre exemple, le nom effectif des variables intervenant dans les modèles *model1* et *model6* devient alors *model1* et *model6 LevelC.LevelB.LevelA.model1.X* et *Level6.LevelB.model6.X*. Cette technique permet une plus grande liberté de nomenclature

durant la modélisation. Ce point permet une réelle liberté de travail tout en ayant un point de vue global sur le système.

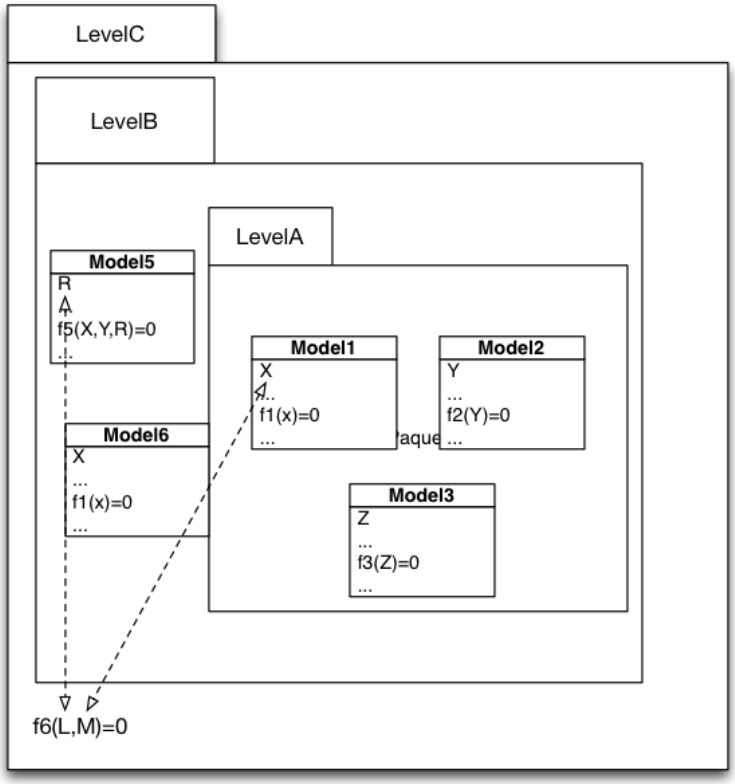



FIGURE *II.10 – Exemple de composition*

## II.5.3 Projection temporelle

L'autre difficulté de la modélisation pour la gestion énergétique est la prise en compte de l'aspect temporel qui induit des démultiplications de variables en fonction du nombre de pas temporels considérés. Cette démultiplication interfère sur la visibilité des modèles par leur développeur. Ce problème complique alors la tâche de débogage. Nous parlons aussi de la transformation de représentations différentielles, où l'on peut trouver des intégrales et des dérivées, en des représentations discrétisées au format algébrique dans lesquelles il ne s'agit alors que d'opérations entre grandeurs indépendantes.

Ce problème de démultiplication a été résolu via une extension de la notion d'indexation que nous traiterons en détail plus bas. Il est nécessaire de déclarer chaque variable évoluant dans le temps afin de pouvoir agréger l'évolution d'une variable temporelle dans un graphique comme c'est le cas avec un outil de simulation. C'est la case *type* citée dans la représentation du conteneur. Cette déclaration permet de manipuler aisément une variable non projetée temporellement au préalable et déclencher une projection sur l'horizon de temps choisi au moment du traitement par MILP-workshop. En pratique, le modèle est décrit pour un pas de temps avec ces dépendances passées. Il est ensuite projeté avec la période d'échantillonnage et l'horizon choisi. Cette projection pose néanmoins le problème des fonctions temporelles dérivées et intégrales.

Pour illuster nos propos, voici un modèle d'une résistance tel qu'il peut être présenté

par un simulateur :

$$U = R \times I \tag{II.1}$$

Dans un problème de gestion où il est question de connecter ce modèle avec des modèles temporels telle que l'action d'un condensateur ou d'une bobine, une projection temporelle s'impose.

La projection temporelle se chargera alors de transformer le modèle II.1 a priori A-temporel en modèle échantillonné temporel II.2. Cette démarche est réalisée en économisant le nombre de variables car seules les variables $U$ et $I$ sont projetées dans le temps. La projection crée deux vecteurs $U$ et $I$ où chaque élément du vecteur représente la valeur de $U$ à l'instant $t$. La variable $R$ correspond ici à un paramètre et ne requiert donc qu'une seule variable quel que soit l'horizon de temps. Cette projection est possible si le développeur de modèles le spécifie.

$$U[t_0] = R \times I[t_0] U[t_1] = R \times I[t_1] U[t_i] = R \times I[t_i] U[t_n] = R \times I[t_n] \tag{II.2}$$

#### II.5.3.1 Intégrale

Pour une fonction discrétisée, l'intégrale peut être interprétée comme un calcul d'aire. Ce mécanisme a été adopté dans le langage MILP-workshop :

$$\int_{n_0 T_{sample}}^{n_1 T_{sample}} f(t)dt = T_{sample} \times \sum_{n_0}^{n_1} f(t) \tag{II.3}$$

avec $T_{sample}$, la période d'échantillonnage.

#### II.5.3.2 Dérivée

La dérivée d'une fonction en un point peut être interprétée comme l'inclinaison de la droite tangente à cette fonction en ce point. La définition formelle de la dérivée est :

$$\frac{df(t)}{dt} = \lim_{dt \to 0} \frac{f(t+dt) - f(t)}{dt} \tag{II.4}$$

Une première approche avec une erreur que nous estimons acceptable dans les applications étudiées dans le cadre de cette thèse, est d'utiliser le pas d'échantillonnage des fonctions discrétisées.

Le champ d'application de la dérivée selon cette approche reste restreint à des problématiques pratiques (mesures, modèles identifiés) mais ne peut aucunement faire preuve d'une justification mathématique sur l'équivalence entre une solution continue et sa discrétisation.

### II.5.4 Conditionnement solveur

L'approche a été développée dans le but d'être utilisée avec des solveurs PLNE. La PLNE nécessite une approche de modélisation spécifique. La principale caractéristique des problèmes PLNE est l'a-causalité.

En PLNE, un problème est résolu en une seule fois et aucun séquencement ou subdivision n'est possible sans altérer la preuve de globalité de la solution obtenue. Les problèmes de causalité rencontrés lors de nos travaux de modélisation des équipements et de l'enveloppe du bâtiment tels que la dérivée, l'intégrale ou encore les fonctions condition et implications sont traitées avant la transmission du problème à résoudre au solveur. La linéarité du problème est aussi impérative pour toute résolution en PLNE. Cette contrainte est forte d'un point de vue modélisation et l'intérêt principal de cet outil est de limiter l'impact de la contrainte linéarité sur la modélisation. Un certain nombre de non-linéarités identifiées peuvent être ainsi levées automatiquement.

Le conditionnement d'un problème PLNE consiste essentiellement à formuler un problème d'optimisation sous une forme canonique :

$$\begin{cases} \text{Minimize(or maximize)} : & Obj(vars) \\ \text{subject to} & A \times vars \leq a \\ \text{and} & B \times vars \geq b \\ \text{and} & C \times vars = c \\ \text{with } vars = & \{v \in \mathbb{R} \vee \mathbb{Z}\ with\ \ Min \leq v \leq Max\} \end{cases} \tag{II.5}$$

La composition de sous-modèles conduit, après transformation, à un ensemble d'équations égalités et inégalités. Dans la composition, on trouve la paramétrisation des variables nécessaires. Le problème de gestion est ensuite composé sur la base de ce modèle avec la définition de sa fonction objectif, du pas d'échantillonnage et de l'horizon de temps. Le mécanisme de linéarisation selon les patterns est défini ci-dessus. Il n'y a pas de garantie de linéarité complète. Une interaction entre le système et le développeur de modèles est établie pour l'orienter dans la formulation des modèles.

Finalement, le problème est traduit en langage d'optimisation et envoyé au solveur.

### II.5.5 Indexation

L'indexation consiste à rassembler des entités de même nature dans un vecteur. Cette démarche a pour but de faciliter certains automatismes dans la gestion des données et des variables a priori et a posteriori. L'indexation existe dans de nombreux langages de simulation, à l'instar de Modelica. Il permet d'appréhender des variables sous forme de vecteurs ou de matrices et d'être ensuite manipulé par des boucles d'itération comme "For" ou "While". Dans notre langage *Milp-workshop*, cette démarche offre une économie de code et une meilleure visibilité figure II.11.

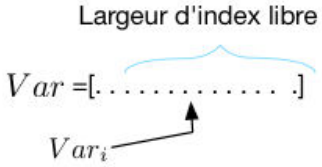



FIGURE *II.11 – index*

Il y a deux types d'indexation nécessaire dans le langage :

**indexation des variables** Cela permet de travailler sur des variables de manière récursive. C'est le cas dans la modélisation de l'enveloppe avec les températures

entourant les enveloppes. Il est ainsi possible de poser une seule contrainte avec un jeu de températures indexés et d'appliquer des boucles dessus.

**indexation des contraintes** Chaque équation porte un nom de référence choisi par le développeur de modèles afin que celui-ci puisse communiquer avec le système en cas d'erreur. Dans la continuité de l'exemple précédent, les contraintes sont aussi indexées de façon à les distinguer une fois la boucle appliquée.

## II.6 Traitement des variables

La variable en optimisation signifie degré de liberté pour l'algorithme de résolution. L'ensemble des degrés de liberté est lié par les contraintes. L'outil d'optimisation cherche, en suivant les contraintes, des combinaisons susceptibles d'optimiser l'objectif qui lui même est un degré de liberté. Cette approche est propre à l'optimisation : on ne la retrouve pas dans la simulation par exemple. D'autre part, notre approche de passage depuis un modèle de représentation vers un problème d'optimisation nécessite certains renseignements particuliers sur les variables. Ainsi, la définition d'une variable prend un sens particulier et nécessite une série d'informations complémentaires.

**Nom :** la dénomination de la variable dans les contraintes

**Préfixe :** la situation de la variable dans la hiérarchie des compositions. C'est un complément du nom qui le rend unique dans le problème d'optimisation.

**Type :** concerne la temporalité ou non d'une variable. C'est-à-dire si celle-ci est sensée être variable dans le temps sur l'horizon choisi.

**Nature :** la nature d'une variable peut être binaire, entière ou continue. Les deux natures binaire et entière impliquent des ensembles de valeurs possibles dans lesquels l'algorithme peut choisir la valeur à affecter. La nature continue permet un choix dans un domaine défini par des bornes

**Bornes inférieure et supérieure :** elle est principalement utilisée dans le cas d'une variable continue mais peut être utilisée pour définir les limites d'un ensemble d'entiers. Si la borne inférieure est $0$ et la borne supérieure est $1$ alors la variable devient tacitement binaire.

**Ensemble fini de valeurs :** c'est un attribut facultatif qui sert à énumérer les valeurs possibles à une variable discrète mais pas entière.

### II.6.1 Manipulation des attributs de variables

Le développeur de modèles doit spécifier la nature et les bornes de chaque variable en fonction de la nature du phénomène physique qu'il souhaite représenter.

Ce choix peut être aussi motivé par des considérations liées à la linéarisation du problème résultant ou encore en vue de la simplification du calcul.

La linéarité est une condition primordiale pour le problème d'optimisation résultant. Celle-ci doit être parfois assurée par des techniques de modélisation. Nous faisons en sorte que toute non-linéarité non évitable dans le modèle corresponde à un des *patterns* que nous verrons plus bas. Parfois, le développeur ou MILP-workshop sont amenés à l'appliquer sur des variables suite à des connexions. Il est donc nécessaire de permettre un accès rapide vers les attributs des variables. D'un point de vue optimisation, le choix de la nature d'une variable oriente vers un type d'algorithme d'optimisation du problème

global. C'est aussi un élément déterminant pour la rapidité de l'optimisation. Les variables entières sont traitées par des algorithme de type Branch and bound tandis que les variables continues sont traitées par des algorithmes de type Simplex ou point intérieur. L'optimisation d'un problème à fort taux de variables discrètes prendra plus de temps de résolution qu'un problème de même taille à variables strictement continues.

Toujours par rapport à la complexité de traitement, les bornes jouent un rôle important car cela délimite l'espace de recherche. Il est vrai que l'aspect physique prime sur la détermination de ces bornes mais il arrive souvent que les développeurs de modèles élargissent les bornes dans le doute ou au contraire les resserrent par négligence. Si les bornes sont trop élargies, le problème prendra plus de temps pour être résolu, voire même ne pas être résolu du tout si le problème est complexe et que l'espace mémoire de l'optimiseur arrive à saturation. A contrario, si les bornes sont trop resserrées alors le problème risque de ne pas trouver de solution. Un travail de débogage est alors nécessaire pour détecter la source d'une absence de solution. Celle-ci pouvant aussi se mêler à d'autres problèmes telles que les erreurs de signe dans les équations.

Enfin, dans certains cas, le développeur de modèle cherche à imposer des valeurs ponctuelles possibles à une variable. Ces valeurs ne sont pas entières et ne peuvent donc pas être déclarées comme telles. Une technique de modélisation que nous avons transcrit en un mécanisme automatique consiste alors à introduire un nombre de variables binaires égal au nombre de valeurs que peut prendre la variable discrète comme dans l'équation II.6

$$Var_{discrete} \in \{val_1, val_2, val_3\} \Leftrightarrow \begin{cases} Var_{discrete} = \delta_1 * val_1 + \delta_2 * val_2 + \delta_3 * val_3 \\ \delta_1 + \delta_2 + \delta_3 = 1 \\ avec : \\ \delta_1, \delta_2, \delta_3 \; binary \end{cases}$$

Avec ce mécanisme, il devient alors simple de déclarer une variable discrète. Cette approche est chargée dans l'attribut facultatif *ensemble* d'une variable.

### II.6.2 Connexions entre variables

La connexion entre variables induit l'uniformisation des attributs. Dans Modelica, il existe une restriction qui oblige la connexion d'une grandeur physique avec une grandeur de même nature physique (température, énergie, puissance, etc.). Cette restriction est utilisée en simulation dans Modelica car les compositions pour la simulation traitent le plus souvent de problématiques purement physiques. Il arrive aussi parfois que des grandeurs électriques tel que le courant ou la tension interviennent dans ces mêmes schémas pour représenter des systèmes plus complexes. Mais il n'y a pas d'aspect économique proprement dit dans les simulateurs Modelica.

En gestion énergétique, la diversité des grandeurs est plus importante car, en plus d'être multi-physiques, un programme de gestion énergétique peut comprendre des grandeurs économiques, des estimations environnementales, du confort ou encore des variables d'état ou tout simplement des grandeurs d'information sans unité. En ce qui concerne les variables d'optimisation, il existe aussi une diversité de types au sens mathématique. Une variable peut être continue, discrète ou binaire. Il est donc peu opportun d'utiliser la restriction comme c'est le cas en simulation. Cela impliquera trop de cas à proposer ou alors trop de limitations dans la modélisation. Notre objectif est au contraire d'offrir un maximum de liberté au développeur tout en lui apportant les facilités nécessaires.

Pour ce faire, nous avons choisi une approche de synchronisation entre variables. Cela se passe à travers des mécanismes pour chacun des attributs de variable.

La connexion entre deux ou plusieurs variables est explicitement précisée avec une commande $connect$. Cette commande crée une nouvelle variable générique pour l'ensemble des variables connectées en gardant un lien avec celles-ci. Nous appellerons cette variable dans le reste du manuscrit $connexion$. La commande $connect$ donne la main au développeur de modèle pour imposer les attributs de $connexion$. Les nouveaux attributs ainsi posés écrasent l'ensemble des attributs des variables d'origine. Dans le cas d'une non imposition d'attributs par le développeur, des mécanismes de définitions se mettent en place.

#### II.6.2.1 La nature

Sachant que $\mathbb{B} \in \mathbb{Z} \in \mathbb{Q}$, le type de $connexion$ sera le type le plus englobant des ensembles des variables à connecter. Ce choix a été fait afin de garder l'inclusivité de la variable résultante. Cette approche est la plus intéressante car elle n'élimine pas d'emblée des solutions possibles et du coté utilisateur, il est toujours possible d'imposer la nature si telle est son intention.

#### II.6.2.2 Le type

L'aspect temporel doit être géré rationnellement car les grandeurs atemporelles doivent être considérées comme telles pour des économies d'espace et de temps de calcul en optimisation combinatoire principalement.

Nous définissons la temporalité de la variable $var$ à travers la fonction $temp(var)$ dans l'illustration qui suit avec $temp(var) = 1$ pour dire que la variable est temporelle et $temp(var) = 0$ pour une variable atemporelle.

La proposition suivante définit la relation entre la temporalité de $connexion$ et les variables connectées :

$$temp(connexion) = temp(var_1) \vee temp(var_i) \vee temp(var_n) \tag{II.6}$$

#### II.6.2.3 Les bornes

Soient $borne_{sup}(var_i)$, $borne_{inf}(var_i)$ ainsi que $borne_{sup}(connexion)$ et $borne_{inf}(connexion)$, les bornes de l'ensemble des variables.

Le domaine de variation de la variable $connexion$ est l'intersection de l'ensemble des domaines de variation des variables connectées comme défini par les équations II.7.

$$\begin{cases} borne_{inf}(connexion) = \max_i(borne_{inf}(var_i)) \\ borne_{sup}(connexion) = \min_i(borne_{sup}(var_i)) \end{cases} \tag{II.7}$$

Le choix de l'intersection est dans la logique d'optimisation de l'algorithme du simplexe. Néanmoins, dans certains cas, il est possible d'avoir un ensemble vide qui se caractérise par une borne inférieure supérieure à une borne supérieure. Ce cas doit retourner une alerte au développeur qui doit alors gérer manuellement le problème.

#### II.6.2.4 Nom

En optimisation, chaque variable représente un degré de liberté supplémentaire pour le solveur. Une économie de variable peut donc mener à une économie de temps si la structure du problème reste inchangée. Extraire des variables inutiles du problème peut donc alléger sensiblement la charge de calcul. Le mécanisme de connexion peut y contribuer. L'ensemble des variables connectées est remplacé dans le modèle résultant par la variable synthétique $connexion$. Cette approche a aussi pour but d'uniformiser les attributs des variables et ainsi donner les indications nécessaires aux prochaines étapes.

### II.6.3 Instantiation et restriction

Cette étape consiste à faire le choix entre les variables de modélisation qui seront considérées comme des degrés de liberté dans le problème et les variables du modèle qui seront paramètres dans le sens PLNE. Plusieurs mécanismes peuvent conduire à une instanciation.

**instantiation initiale** : Ce type d'instanciation sert principalement à décrire des ratios fixes quelque soit l'adaptation que l'on peut faire du modèle en question. Il peut aussi représenter des paramètres physiques constants telles que la densité de l'air ou encore l'intensité lumineuse de référence du soleil.

**instantiation automatique par connexion** : Ce type d'instanciation survient lors de la composition. Plusieurs variables de modélisation sont alors associées. Parmi lesquelles certaines qui sont déjà instanciées. D'un point de vue problème, il est donc inutile d'instancier le reste des variables : il suffit de poser l'égalité. Dans certains cas, cette instantiation est nécessaire car elle peut lever certaines non-linéarités.

Un exemple de ce type de connexion et de son utilité est illustré. Un premier modèle comporte une seule variable $Var_1$

$$Model_1.Var_1 = 1 \tag{II.8}$$

Le second modèle comporte trois variables :

$$Model_2.Var_1 = Model_2.Var_2 \times Model_2.Var_3 \tag{II.9}$$

Le modèle $Model_2$ comporte une non-linéarité qui ne peut être levée si $Var_2$ et $Var_3$ sont continues. La connexion entre les deux variables $Var_1$ de $Model_1$ et $Var_2$ de $Model_2$ revient à poser

$$Model_1.Var_1 = Model_2.Var_2 \tag{II.10}$$

Le problème global devient :

$$\begin{cases} Model_1.Var_1 = 1 \\ Model_2.Var_1 = Model_2.Var_2 * Model_2.Var_3 \\ Model_1.Var_1 = Model_2.Var_2 \end{cases} \tag{II.11}$$

Après procédure de connexion, il vient :

$$\begin{cases} connexion.Var_1 = 1 \\ Model_2.Var_1 = connexion.Var1 \times Model_2.Var_3 \\ connexion.Var1 = Model_2.Var_2 \\ connexion.Var1 = Model_1.Var_1 \end{cases} \tag{II.12}$$

Le problème tel que posé ici va être lu par un solveur PLNE comme non linéaire à cause du produit. Il est clair que ce problème ne l'est pas en réalité. Il suffit alors de remplacer $connexion.Var_1$ par sa valeur dans l'ensemble des équations et celui-ci redeviendra linéaire

$$\begin{cases} connexion.Var_1 = 1 \\ Model_2.Var_1 = 1 * Model_2.Var_3 \\ connexion.Var1 = Model_2.Var_2 \\ connexion.Var1 = Model_1.Var_1 \end{cases} \tag{II.13}$$

Un algorithme peut ainsi balayer le problème à la recherche de variables de modélisation fixées pour qu'elles soient considérées comme paramètre ou remplacées dans les équations afin de lever les non-linéarité. Une condition d'arrêt est néanmoins nécessaire à l'algorithme de recherche sous peine, dans certains cas, de résoudre l'ensemble du problème avant même sa résolution par le solveur PLNE. La résolution a priori ne peut être acceptée car bien plus lente qu'une résolution PLNE, elle pénaliserait donc l'ensemble du processus. La condition d'arrêt de l'algorithme est la vérification de la forme du problème. A partir de l'itération où le problème devient linéaire, l'instanciation des variables n'est plus nécessaire.

**Instantiation volontaire par le modélisateur** : Dans le fichier de composition d'un problème, est déclaré l'ensemble des variables de modélisation qui doivent être considérées comme paramètres. Les valeurs de ces paramètres sont ainsi déclarées et remplacées dans toutes les équations du problème.

**instantiation par l'utilisateur final de l'application** : Ce dernier cas d'instanciation est lié à l'interaction avec l'occupant ou avec l'utilisateur final du BEMS.

# II.7 Traitement des contraintes

La contrainte est le liant entre les variables et les paramètres. Sa représentation dans le conteneur comprend la dénomination à travers le nom donné par le développeur et le préfixe définit par la composition. Vient ensuite la représentation mathématique avec ses expressions et le lien =, < ou >. Nous avons aussi laissé place à des tags de marquage. Les contraintes sont triées selon leur temporalité et selon leur linéarité pour être traitées à posteriori.

## II.7.1 Traitement des propositions logiques

Ce traitement répond à certains besoins récurrents en modélisation pour la gestion énergétique notamment lorsqu'il s'agit de décrire les équipements. Les implications ainsi que les descriptions conditionnelles sont les deux principales fonctions que nous utilisons le plus souvent.

### II.7.1.1 Implications

Dans cette section, il est question de représenter sous forme algébrique des connecteurs logiques de type implication et équivalence. Ces connecteurs retournent des informations sur la liaison ou non entre deux propositions logiques. C'est aussi la brique de base de propositions logiques à venir.

D'un point de vue mathématique, les connecteurs sont des fonctions qui retournent une variable binaire. Par convention, le résultat est égal à 1 si la connexion est juste, et 0 sinon. En modélisation, l'utilisation qui en est souvent faite, concerne uniquement le résultat juste, c'est-à-dire que la connexion est supposée juste. A l'intérieur des propositions, les valeurs sont alors affectées aux variables de façon à ce que la connexion soit effectivement juste.

Le connecteur implication est caractérisé par deux parties : la proposition de départ et la proposition d'arrivée. Pour développer le mécanisme de transformation, il est question de chercher la représentation la plus universelle possible. Nous constituerons sur la base de cette dernière un patron de transformation *pattern* qui pourra être implémenté pour transformer automatiquement la proposition en contrainte sous forme algébrique.

Nous classifions l'ensemble des éventualités pour la proposition de départ ou celle d'arrivée sous quatre formes :

— $f(x) \leq K$
— $f(x) \geq K$
— $f(x) = K$
— $\delta = 1$

La forme $\delta = 1$ est une spécification de la forme $f(x) = K$. Néanmoins, elle nous est utile dans la méthode de transformation car elle intègre la solution d'une manière directe. Cette solution sera déroulée pour des combinaisons dans lesquelles au moins la proposition de départ ou d'arrivée sera de la forme $\delta = 1$ comme retracé dans le tableau II.1.

| | $f(x) \leq K$ | $f(x) \geq K$ | $f(x) = K$ |
|---|---|---|---|
| from binary | $\delta = 1 \rightarrow f(x) \leq K$ | $\delta = 1 \rightarrow f(x) \geq K$ | $\delta = 1 \rightarrow f(x) = K$ |
| to binary | $f(x) \leq K \rightarrow \delta = 1$ | $f(x) \geq K \rightarrow \delta = 1$ | $f(x) = K \rightarrow \delta = 1$ |

TABLE *II.1 – Inventaire des formes d'implication*

La solution consiste en une combinaison sous forme algébrique qui vérifie le tableau de vérité de l'implication de chacune des six propositions comme énuméré dans le tableau II.2 :

| | |
|---|---|
| $\delta = 1 \rightarrow f(x) \leq K$ | $f(x) \leq (1-\delta)\overline{f(x)} + \delta K$ |
| $\delta = 1 \rightarrow f(x) \geq K$ | $f(x) \geq (1-\delta)\underline{f(x)} + \delta K$ |
| $\delta = 1 \rightarrow f(x) = K$ | $f(x) \leq (1-\delta)\overline{f(x)} + \delta K$ $\wedge$ $f(x) \geq (1-\delta)\underline{f(x)} + \delta K$ |
| $f(x) \leq K \rightarrow \delta = 1$ | $f(x) \geq \delta\underline{f(x)} + (1-\delta)K$ |
| $f(x) \geq K \rightarrow \delta = 1$ | $f(x) \leq \delta\overline{f(x)} + (1-\delta)K$ |
| $f(x) = K \rightarrow \delta = 1$ | $f(x) \leq \delta\overline{f(x)} + (1-\delta)K$ $\wedge$ $f(x) \geq \delta\underline{f(x)} + (1-\delta)K$ |

TABLE *II.2 – Inventaire des formes possibles pour les représentations*

Nous avons utilisé les deux paramètres $\underline{f(x)}$ et $\overline{f(x)}$ représentants respectivement le minimum et le maximum de la fonction $f(x)$ en les associant à la variable binaire $\delta$.

Afin de prouver l'exactitude de la solution, nous proposons de vérifier l'une des six propositions en dressant le tableau de vérité de l'équivalence entre les deux formes (implication et forme algébrique) face à l'instanciation de la variable $\delta$ :

| $f(x) \geq K$ | $\delta = 1$ | $f(x) \geq K \rightarrow \delta = 1$ | $f(x) \leq \delta\overline{f(x)} + (1-\delta)K$ |
|---|---|---|---|
| 0 | 0 | ✓ | ✓ |
| 0 | 1 | ✓ | ✓ |
| 1 | 0 | X | X |
| 1 | 1 | ✓ | ✓ |

TABLE *II.3 – Table de vérité Vs instanciation forme algébrique*

Le résultat montre que les deux représentations sont équivalentes. Ce constat permet donc de remplacer les implications par leur équivalent dans le format algébrique.

Il reste le cas d'une implication de type :

$$f_1(x_1)(\leq = \geq)K_1 \rightarrow f_2(x_2)(\leq = \geq)K_2 \tag{II.14}$$

Cette forme est traitée comme une suite de deux formes précédemment traitées avec $\delta$ comme lien entre les deux :

$$\begin{cases} f(x_1)_1(\leq = \geq)K_1 \rightarrow \delta = 1 \\ \delta = 1 \rightarrow f_2(x_2)(\leq = \geq)K_2 \end{cases} \tag{II.15}$$

Il suffit alors d'appliquer les deux patrons précédemment présentés relatifs aux propositions en jeu pour avoir la forme algébrique de cette implication.

### II.7.1.2 Equivalences

Une démarche similaire est entreprise pour l'équivalence. L'équivalence n'étant qu'une implication à double sens, nous retrouverons donc les contraintes des implications dans les deux sens qui se superposent dans le tableau II.4

| $f(x) \leq K$ | $f(x) \geq K$ | $f(x) = K$ |
|---|---|---|
| $\delta = 1 \leftrightarrow f(x) \leq K$ | $\delta = 1 \leftrightarrow f(x) \geq K$ | $\delta = 1 \leftrightarrow f(x) = K$ |

TABLE *II.4 – Inventaire des formes d'équivalence*

La formalisation des patrons de transformation pour l'équivalence s'inspire des patrons d'implication. Nous appliquons à chaque équivalence les deux patrons correspondants aux implications dans les deux sens. Le résultat est le suivant :

$$\delta = 1 \leftrightarrow f(x) \leq K \Leftrightarrow \left\{ \begin{array}{lcl} f(x) & \leq & (1-\delta)\overline{f(x)} + \delta K \\ f(x) & \geq & \delta\underline{f(x)} + (1-\delta)K \end{array} \right.$$

$$\delta = 1 \leftrightarrow f(x) \geq K \Leftrightarrow \left\{ \begin{array}{lcl} f(x) & \leq & (1-\delta)\underline{f(x)} + \delta K \\ f(x) & \geq & \delta\overline{f(x)} + (1-\delta)K \end{array} \right.$$

$$\delta = 1 \leftrightarrow f(x) = K \pm \epsilon \Leftrightarrow \left\{ \begin{array}{lcl} f(x) & \leq & (1-\delta)\overline{f(x)} + \delta(K+\epsilon) \\ f(x) & \geq & (1-\delta)\underline{f(x)} + \delta(K-\epsilon) \end{array} \right.$$

### II.7.1.3 Condition

La structure de condition/exécution, *ifThenElse* ou *switch*, est très utile pour décrire les équipements et systèmes physiques à états. Cette approche est utilisée en modélisation d'heuristiques de contrôle notamment. Elle se présente sous la forme :

$$\begin{array}{l} \text{If} \quad (f(x) \leq K) \\ \qquad \text{Then} \\ \qquad\qquad g(x,y) = E_1 \\ \text{Else} \\ \qquad g(x,y) = E_2 \end{array} \tag{II.16}$$

Cette structure peut être vue sous forme d'implications c'est-à-dire que la proposition de départ de l'implication devient la condition du $if$ et la proposition d'arrivée devient l'instruction à exécuter II.12.

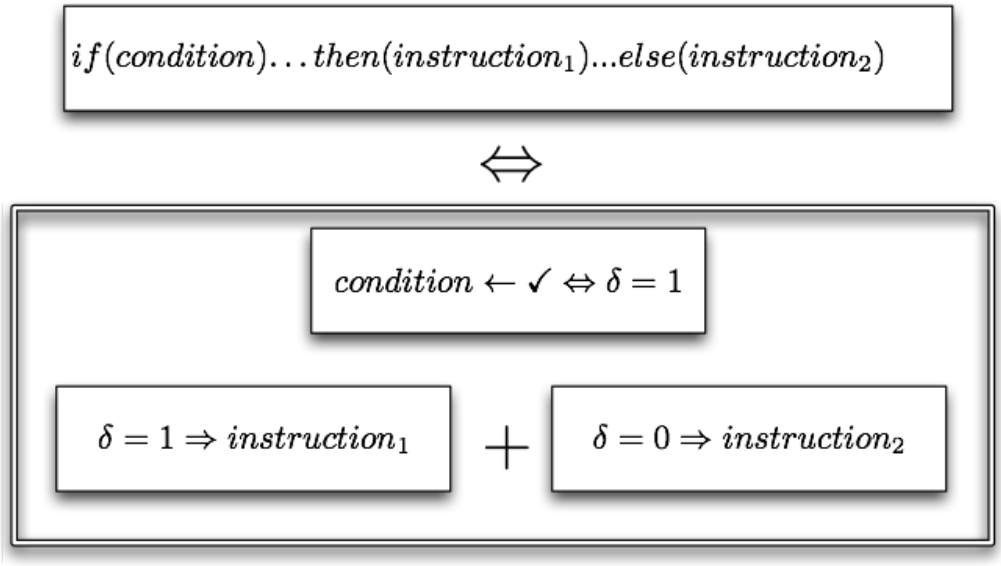



FIGURE *II.12 – transformation If ... Then ... Else ...*

A travers cette conversion, le patron de transformation se déduit alors aisément de ce qui a été vu précédemment.

En raison du nombre important de combinaisons selon les types de propositions ( $\leq$ , $\geq$ , =) pour la condition et les instructions, seul le cas II.16 des 27 cas possibles est présenté dans ce paragraphe.

$$\begin{array}{l} \text{If} \quad (f(x) \leq K) \\ \qquad \text{Then} \\ \qquad\qquad g(x,y) = E_1 \end{array} \iff \begin{cases} f(x) \geq (1-\delta)\underline{f(x)} + \delta K \\ \\ g(x,y) \leq (1-\delta)\overline{g(x,y)} + \delta E_1 \\ g(x,y) \geq (1-\delta)\underline{g(x,y)} + \delta E_1 \\ \\ g(x,y) \leq \delta\overline{g(x,y)} + (1-\delta)E_2 \\ g(x,y) \geq \delta\underline{g(x,y)} + (1-\delta)E_2 \end{cases} \quad \begin{array}{l} \\ \\ \\ \\ \\ \\ \text{Else} \\ \quad g(x,y) = E_2 \end{array} \tag{II.17}$$

Sur le même principe, le reste des patterns peut être déduit.

## II.7.2 Gestion des non-linéarités

La description mathématique d'un problème physique passe par des modélisations non-linéaires mais qui peuvent très souvent être formulées sous forme linéaire mixte moyennant parfois quelques approximations.

Pour la gestion énergétique dans le bâtiment, le problème de non linéarité est souvent lié à certaines variables de commande agissant directement sur les grandeurs physiques. De ce fait, la modélisation de cette interaction se fait essentiellement à travers des produits de variables (variables de contrôle et variables physiques sur lesquelles elles agissent) pour décrire l'effet de la commande sur le système. L'exemple II.18 en est l'illustration.

$$Q_{air} = Q_{nominal} * On \tag{II.18}$$

avec : $Q_{air}$ : débit d'air à la sortie de la machine $Q_{nominal}$ : débit d'air nominal de la pompe à air $On$ : état de la pompe, $1$ pour dire pompe en état de fonctionnement et $0$ pour dire qu'elle est à l'arrêt.

L'approche de résolution PLNE adoptée ne s'applique pas en présence de non-linéarités. Les questions qui se posent alors sont :

— Doit-on utiliser un solveur non linéaire ?
— Doit-on simplifier le problème pour l'exprimer sous une forme linéaire en acceptant éventuellement une distorsion de l'information ou de la perte de celle-ci ?
— Doit-on garder le problème et essayer de l'exprimer autrement ?

La possibilité d'utiliser des méthodes de résolution non linéaires peut être envisagée mais le bénéfice temps/efficacité de résolution n'est pas avéré. En effet, la thèse (MISSAOUI et al. 2014) a montré que la résolution linéaire conduisait à des temps de calcul très faibles et garantissait un optimum global, ce qui n'est pas le cas des méthodes d'optimisation non-linéaires.

La linéarisation d'emblée d'un problème de gestion énergétique dans le bâtiment peut être envisagée. Cette linéarisation ne peut pas être envisagée au détriment de la qualité et de l'exactitude des commandes de gestion des systèmes par le BEMS.

Certaines non-linéarités peuvent être levées d'une manière systématique à travers le mécanisme des patrons précédemment évoqué. Toutes les non-linéarités ne peuvent pas être lévées de cette manière car cela peut être techniquement impossible ou alors elles sont suffisament rares dans la problématique que nous traitons pour que celles-ci méritent l'implémentation de leur patron. Elles peuvent être enlevées par le développeur de modèles lui-même.

Les produits non-linéaires peuvent, dans certains cas, être exprimés sous une forme linéaire en optimisation. L'expression de produits non-linéaires sous forme linéaire trouve sa solution dans l'utilisation de contraintes inégalités, de variables binaires et de discrétisation de variables continues.

Le produit de variables exprime souvent une discontinuité. Cette discontinuité est générée par la variable de décision sur une variable. Si les cas sur lesquels sont appliquées ces variables de décision sont linéaires alors il y a moyen d'exprimer les discontinuités par des transformations linéaires ingénieuses explicitées ci-dessous : voir II.7.2 page 31.

La technique de transformation a été initiée dans (WILLIAMS 1999). Celle-ci émane de la logique propositionnelle. En effet, une variable de contrôle est souvent binaire ou du moins entière. C'est ce qui a permis d'orienter les recherches vers ce genre de transformations. A la base, le produit de deux variables binaires est un produit Booléen. On peut déduire la table de vérité de ce produit puis, en exploitant la possibilité d'utiliser les contraintes inégalités, les résultats du produit sont bornés par les variables sans créer de non-linéarités.

Cette approche n'est valable que pour l'optimisation PLNE car la solution est recherchée dans un espace. Pour la simulation, par exemple, cette approche n'est pas adaptée car la linéarisation impose l'usage de variables binaires dans des systèmes intégrant des équations et des inéquations. En simulation, seules les équations sont intégrées dans les algorithmes de résolution. Les inéquations sont appliquées a posteriori pour valider les solutions. Il y a donc perte d'une partie de l'information pour la simulation.

### II.7.2.1 Patrons de linéarisation d'un produit semi-continu

Le produit semi continu est le produit d'une variable continue et d'une variable binaire ne pouvant prendre que deux valeurs 0 ou 1.

$C$ **:** variable continue variant entre $[\underline{C}, \overline{C}]$

$\beta$ **:** variable binaire $\in \{0, 1\}$

$Z$ **:** résultat du produit

$$Z = C \times \beta \tag{II.19}$$

Le produit semi continu peut être exprimé sous la forme :

$$\begin{cases} Z = C & Si \quad \beta = 1 \\ Z = 0 & Si \quad \beta = 0 \end{cases} \tag{II.20}$$

La transformation en écriture linéaire passe par une délimitation de bornes supérieure et inférieure pour exprimer chacune des deux situations :

$$\begin{cases} Z = C & Si \quad \beta = 1 \\ Z = 0 & Si \quad \beta = 0 \end{cases} \Leftrightarrow \begin{cases} Z & \leq & C - \underline{C} * (1 - \beta) \\ Z & \geq & C - \overline{C} * (1 - \beta) \\ Z & \leq & \overline{C} * \beta \\ Z & \geq & \underline{C} * \beta \end{cases} \tag{II.21}$$

La démonstration de la justesse de cette linéarisation se fait par l'exécution de la table de vérité de la proposition logique II.21.

— Cas 1 : $\beta = 1$

$$(Z = 0 \quad Si \quad \beta = 1) \tag{II.22}$$

car :

$$\begin{cases} Z & \leq & C - \underline{C} * 0 \\ Z & \geq & C - \overline{C} * 0 \\ Z & \leq & \overline{C} \\ Z & \geq & \underline{C} \end{cases} \Leftrightarrow \begin{cases} Z & \leq & C \\ Z & \geq & C \\ Z & \leq & \overline{C} \\ Z & \geq & \underline{C} \end{cases} \tag{II.23}$$

Dans ce cas, seules les deux premières équations de II.23 interviennent dans le jeu d'équations influentes. Celles-ci limitent le produit semi-continu à sa juste valeur $Z = C$. Les deux équations restantes ne font que répéter l'influence de $C$ qui est aussi celui de $Z$.

— Cas 2 : $\beta = 0$

$$(Z = 0 \quad Si \quad \beta = 0) \tag{II.24}$$

car :

$$\begin{cases} Z & \leq & C - \underline{C} \\ Z & \geq & C - \overline{C} \\ Z & \leq & \overline{C} \times 0 \\ Z & \geq & \underline{C} \times 0 \end{cases} \tag{II.25}$$

et

$$\begin{cases} C & \geq & \underline{C} \\ C & \leq & \overline{C} \\ Z & \leq & C - \underline{C} \\ Z & \geq & C - \overline{C} \end{cases} \Leftrightarrow \begin{cases} C - \underline{C} & \geq & 0 \\ C - \overline{C} & \leq & 0 \\ Z \in ]C - \overline{C}, 0] & \cup & [0, C - \underline{C}[ \end{cases} \tag{II.26}$$

$$\Leftrightarrow \begin{cases} Z \in ]C - \overline{C}, 0] & \cup & [0, C - \underline{C}[ \\ Z & \leq & 0 \\ Z & \geq & 0 \end{cases} \tag{II.27}$$

Dans ce cas, les deux premières équations deviennent inutiles. Le système II.27 donne une seule solution $Z = 0$.

### II.7.2.2 Patrons de linéarisation d'un produit binaire-binaire

Le produit de deux variables binaires est un cas purement logique et est à l'origine de l'ensemble des transformations (WILLIAMS 1999).

Soit :

$$Z = \beta_1 \times \beta_2 \tag{II.28}$$

avec

$Z$ : le résultat du produit lequel est aussi de type binaire

$\beta_1$ : le premier binaire du produit

$\beta_1$ : le second binaire du produit

$$Z = \beta_1 * \beta_2 \Leftrightarrow \begin{cases} Z & \leq & \beta_1 \\ Z & \leq & \beta_2 \\ Z & \geq & \beta_1 + \beta_2 - 1 \\ Z & is & Binary \end{cases} \tag{II.29}$$

Pour preuve :

La table de vérité II.7.2.2 va être démontrée cas par cas en exécutant le système.

| $cas$ | $Z$ | $\beta_1$ | $\beta_2$ |
|---|---|---|---|
| cas1 | 1 | 1 | 1 |
| cas2 | 0 | 0 | 1 |
| cas3 | 0 | 1 | 0 |
| cas4 | 0 | 0 | 0 |

cas 1 :

$$\begin{cases} z & \leq & 1 \\ z & \leq & 1 \\ z & \geq & 1 \\ Z & is & Binary \end{cases} \Rightarrow \{Z = 1 \tag{II.30}$$

cas 2 :

$$\begin{cases} z & \leq & 0 \\ z & \leq & 1 \\ z & \geq & 0 \\ Z & is & Binary \end{cases} \Rightarrow \{Z = 0 \tag{II.31}$$

cas 3 :

$$\begin{cases} z & \leq & 1 \\ Z & \leq & 0 \\ Z & \geq & 0 \\ Z & is & Binary \end{cases} \Rightarrow \{Z = 0 \tag{II.32}$$

cas 4 :

$$\begin{cases} z & \leq & 0 \\ Z & \leq & 0 \\ Z & \geq & -1 \\ Z & is & Binary \end{cases} \Rightarrow \{Z = 0 \tag{II.33}$$

#### II.7.2.3 Patrons de linéarisation d'un produit de $n$ variables binaires

Ce pattern est une généralisation du produit de deux variables binaires. Il n'est pas à confondre avec la ré-application du même pattern en traitant les variables deux à deux. Cette solution peut être envisagée dans des cas plus rares et dans des problèmes d'une complexité modérée. Ce pattern a été développé afin de réduire le nombre de variables générées ainsi que le nombre de contraintes et donc la complexité du problème résultant.

$$Z = \beta_1 \times ... \times \beta_i \times ... \times \beta_n \tag{II.34}$$

$Z$ : le résultat du produit lequel est aussi de type binaire

$\beta_i$ : le binaire $i$ du produit $i \in \{1, n\}$

$$Z = \prod_{i=1}^{i=n} \beta_i \Leftrightarrow \begin{cases} Z & \leq & \beta_1 \\ . & ... & . \\ Z & \leq & \beta_n \\ Z & \geq & \sum_{i=1}^{i=n} \beta_i - (n-1) \\ Z & is & Binary \end{cases} \tag{II.35}$$

#### II.7.2.4 Patrons de linéarisation d'un produit discret-continu

Le produit entier-continu est assez présent dans la modélisation bâtiment. Il nous sert principalement dans la commande par modes d'une variable continue, c'est-à-dire que l'équipement peut avoir plusieurs états de fonctionnement et chaque état influence singulièrement une variable continue comme illustré par l'exemple II.36.

$$Z = mode \times Var \tag{II.36}$$

$Z$ **:** résultat du produit

$mode$ **:** variable discrète $mode \in \{value_1, ..., value_n\}$

$Var$ **:** variable continue $Var \in [Min, Max]$

La technique utilisée dans ce patron est la composition de la variable discrète en une somme pondérée de variables binaires. Cette somme est ensuite reformulée pour composer une série de produits semi-continus.

$$mode = (value_1 \times Bin_1 + value_i \times Bin_i + value_n \times Bin_n) \tag{II.37}$$

ou :

$$mode = \sum_{i=1}^{i=n} value_i \times Bin_i \tag{II.38}$$

En remplaçant II.38 dans II.36, on obtient :

$$Z = (\sum_{i=1}^{i=n} value_i \times Bin_i) \times Var \tag{II.39}$$

En distribuant $Var$ on a :

$$Z = \sum_{i=1}^{i=n} value_i \times (Bin_i * Var) \tag{II.40}$$

A partir de cette étape, le produit semi-continu est caractérisé. Il suffit de créer une variable de synthèse pour chaque produit et appliquer le pattern adéquat.

$$Z = \sum_{i=1}^{i=n} value_i \times PSC_i \tag{II.41}$$

avec :

$$PSC_i = Bin_i \times Var \tag{II.42}$$

Cette dernière forme est bien linéarisable.

#### II.7.2.5 Patrons de linéarisation d'un produit de $n$ entiers

$$Z = \prod_{i=1}^{i=n} Int_i \tag{II.43}$$

$Z$ **:** résultat du produit

$Int_i$ **:** variable discrète $Int_i \in \{value_1, ..., value_{n_j}\}_i$

Par analogie au produit entier-continu, le produit de $n$ variables entières est ramené à un produit de $n$ variables binaires :

$$Z = \prod_{i=1}^{i=n} (\sum_{j=1}^{j=n_i} Bin_{i_j} \times Value_{i_j}) \tag{II.44}$$

Cette équation peut être réécrite sous cette forme :

$$Z = \sum_{j=1_1}^{j=n_1} ... \sum_{i=n_1}^{i=n_j} (\prod_{i=1}^{i=n} Bin_{i_j} \times Value_{i_j}) \tag{II.45}$$

finalement :

$$Z = \sum_{j=1_1}^{j=n_1} ... \sum_{i=n_1}^{i=n_j} (\prod_{i=1}^{i=n} Bin_{i_j} \times \prod_{i=1} Value_{i_j}) \tag{II.46}$$

Le produit binaire est clairement isolé dans l'expression II.46. Celui-ci est ensuite linéarisé avec le pattern adéquat.

Ce pattern est très gourmand en ressource car il utilise de nombreuses variables binaires. Il est à utiliser avec modération et en limitant le nombre de valeurs possibles pour chaque variable discrète. Celui-ci peut très rapidement saturer le problème global et le rendre impossible à résoudre.

### II.7.3 Implémentation informatique

Pour une implémentation rapide et à usage académique, le choix de Beanshell a été retenu : il s'agit d'un interpréteur de code Java que nous avons enrichi de nos propres instructions. Cette approche offre une grande flexibilité car, en plus des instructions définies, on bénéficie de toute la puissance de Java. Notre but est de démontrer la faisabilité de l'approche. Les fichiers de modèles ne sont pas intégrés dans le système de gestion directement comme c'est le cas dans d'autres travaux mais à travers une bibliothèque de modèles propre au développement de modèles.

Nous avons proposé l'architecture (figure II.13) pour mettre en place les concepts développés dans les chapitres précédents. Cette architecture a été construite avec une considération particulière pour la possibilité d'étendre le traitement des non-linéarités et des formes particulières dans la projection temporelle. Un package spécifique regroupe l'ensemble des patrons pour la linéarisation et un autre pour les patrons de projection temporelle.

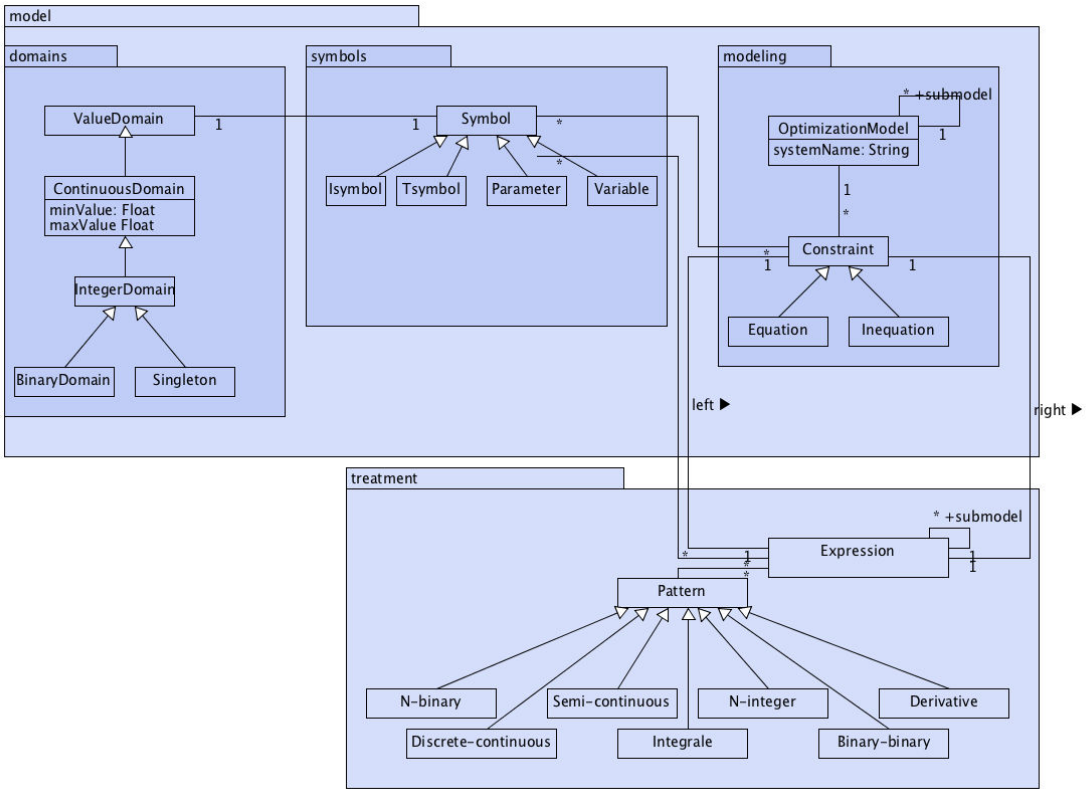



FIGURE *II.13 – modèle UML de la solution logicielle Milp-workshop*

Une partie du parsage de modèles se fait à travers les mots clés énumérés dans le package *commands*. Les boucles sont aussi parsées grâce à BeanShell directement au même titre que l'organisation du document (passage à la ligne, séparation des commandes, etc. ). Le contenu des commandes est parsé en considérant le contenu comme chaine de caractère. Nous avons choisi l'implémentation sous cette forme car nous estimons que c'est le moyen le plus simple pour le développeur de modèles de concentrer l'effort sur l'aspect modèles. La capture d'écran II.14 montre le résultat de la solution logicielle baptisée Milp-workshop.

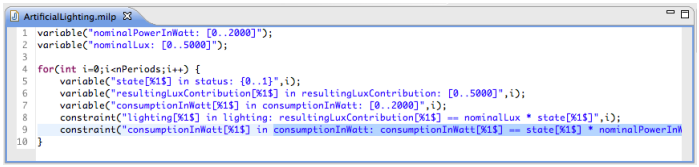



FIGURE *II.14 – Exemple de modèle sous Milpworkshop*

## II.8 Conclusion

Dans ce chapitre, nous avons commencé par mettre en évidence les lacunes constatées dans le travail de (HA 2008), lequel sert de base de travail à cette thèse. Ces lacunes ont été constaté au niveau de l'introduction de modèles. Lorsque le concept du gestionnaire énergétique GHomeTech a été introduit, la partie développement de modèles n'a pas été évoqué. La solution s'est posée alors comme l'implémentation d'un outil résolvant un

problème d'optimisation fixe. Or, nous avons constaté que la mise en pratique d'un gestionnaire énergétique requiert des bibliothèques de modèles au vu de l'hétérogénéité des bâtiments dans lesquels il peut être déployé. A partir de ce constat, moyennant une revue de littérature d'autres outils faisant intervenir des modèles d'une manière massive, nous avons décidé de mettre en place un outil indépendant qui se chargera de la gestion de la modélisation. Cet outil que nous avons baptisé MILP-workshop se doit de faire l'interface entre le développeur de modèles qui peut être autre que le développeur de GHomeTech et GHometech lui même. Nous avons proposé une série de mécanismes que nous avons jugés utiles afin de simplifier la modélisation. Ces mécanismes tournent autour de l'automatisation de certaines tâches afin d'économiser l'effort au développeur de modèles et garder ce dernier lisible pour déceler d'éventuels bugs a posteriori. Parmi les tâches pour lesquelles nous avons proposé des mécanismes d'automatisation nous citons la composition de modèles, la projection temporelle, le traitement de propositions spécifiques, la manipulation d'attributs de variables et enfin la linéarisation. Certaines de ces tâches sont automatisables grâce à une notion de patrons.

# Chapitre III

# Application au bâtiment CANOPEA

Les outils présentés dans les chapitres précédents permettent d'appréhender la gestion énergétique des systèmes bâtiments complexes. Ils ont été appliqués au bâtiment CANOPEA conçu à l'occasion de la compétition *Solar Decathlon Europe 2012* par l'équipe Rhône-Alpes.

La compétition *Solar Decathlon* est une compétition internationale initiée par l'US Department of Energy et ouverte aux universités du monde entier. La compétition consiste en la présentation d'un prototype échelle 1 d'un bâtiment par une équipe d'étudiants. Le prototype est soumis à un cahier des charges préétabli par le pays organisateur. Celui-ci fixe la superficie, le budget ainsi que les règles à respecter.

L'objectif fondateur de cette compétition est de concevoir des bâtiments de grande qualité architecturale avec une intégration des moyens de production Photovoltaïques optimale et très efficace d'un point de vue énergétique, en valorisant notamment l'énergie solaire collectée sur site.

L'édition *Solar Decathlon 2012* a été accueillie par Madrid en Espagne pour la seconde fois consécutive dans sa déclinaison Européenne. L'objectif de cette édition a été de tendre vers l'autosuffisance énergétique en travaillant, d'un côté, sur la sobriété énergétique et de l'autre, sur des moyens de production justement dimensionnés. Le cahier des charges a été proposé courant 2010. Vingt équipes, parmi lesquelles figure l'équipe Rhône-Alpes, ont été présélectionnées dont deux qui ont abandonné.

L'équipe Rhône-Alpes a eu deux ans pour préparer son projet et construire son prototype. Le prototype présenté pour l'édition 2012 a été baptisé Canopea ; il a fini vainqueur de la compétition. Le prototype Canopea n'est qu'une partie du projet global de l'équipe Rhône-Alpes. En effet, Canopea représente les deux derniers étages d'une nanotour. L'avant dernier étage est un habitat type tandis que le dernier étage est un espace de vie commun non-tempéré abritant, entre autres, une partie des équipements techniques.

La compétition *Solar Decathlon 2012* n'impose pas de contraintes sur le type d'urbanisation des projets. La tradition est néanmoins de proposer des prototypes de type habitat individuel extra urbain. Cette tradition est aussi implicitement comprise à travers le cahier des charges qui limite l'emprise au sol ainsi que la hauteur de chaque prototype. Les participants mettent l'accent sur l'autosuffisance et l'intégration architecturale des moyens de production d'énergie, des matériaux et équipement.

## III.1 Règles de la compétition *Solar Décathlon 2012*

Les règles de la compétition sont détaillées dans ce paragraphe car elles ont été déterminantes pour la conception des stratégies de gestion énergétique. Des règles et des contraintes ont été imposées dans le cahier des charges initial afin de simuler l'occupation effective dans chaque prototype car celui-ci n'avait pas un usage conventionnel durant l'exposition sur le site de la compétition.

| No. | Contest/Sub-contest Name | Contests Points | SubContests Points | Assigned by |
|---|---|---|---|---|
| 1 | Architecture | 120 | | Jury |
| 2 | Engineering & Construction | 80 | | Jury |
| 3 | Energy Efficiency | 100 | | Jury |
| 4 | Electrical Energy Balance | 120 | | |
| | 4.1 Electricity Autonomy | | 60 | Monitored performance |
| | 4.2 Temporary correlation | | 40 | Monitored performance |
| | 4.3 Electricity use per measurable area | | 20 | Monitored performance |
| 5 | Comfort Conditions | 120 | | |
| | 5.1 Temperature | | 70 | Monitored performance |
| | 5.2 Humidity | | 10 | Monitored performance |
| | 5.3 Indoor Air Quality | | 5 | Monitored performance |
| | 5.4 Workstation Lighting | | 20 | Task / Monitored |
| | 5.5 Acoustic | | 15 | Monitored performance |
| 6 | House Functioning | 120 | | |
| | 6.1 Refrigerator | | 5 | Monitored performance |
| | 6.2 Freezer | | 5 | Monitored performance |
| | 6.3 Clothes Washer | | 20 | Task + Monitored |
| | 6.4 Clothes Dryer | | 10 | Task Completion |
| | 6.5 Dish Washer | | 15 | Task + Monitored |
| | 6.6 Home Electronics | | 5 | Task + Monitored |
| | 6.7 Oven | | 15 | Task + Monitored |
| | 6.8 Cooking | | 15 | Task Completion |
| | 6.9 Hot Water Draws | | 20 | Task Completion |
| | 6.10 Dinner | | 10 | Guests |
| 7 | Communication and Social Awareness | 80 | | Jury |
| 8 | Industrialization & Market Viability | 80 | | Jury |
| 9 | Innovation | 80 | | Jury |
| 10 | Sustainability | 100 | | Jury |

FIGURE *III.1 – Répartition des points selon les aspects*

Le classement final est basé sur la notation de la figure III.1 qui représente la pondération choisie par l'organisation pour chaque aspect abordé en compétition. Il existe deux types de notations : les notes attribuées sur la base de mesures et les notes attribuées par des jurys d'experts sur la base d'examens de dossiers et d'inspections.

Le gestionnaire énergétique du bâtiment mis en place impacte les épreuves 4, 5 et 6 c'est-à-dire 36% du résultat final. Nous avons aussi voulu contribuer à l'aspect industrialisation et viabilité économique du projet, épreuve 8, à travers un BEMS exploitant un modèle énergétique global du système Canopea. Le choix de l'équipe a été de privilégier la maîtrise énergétique parfois au détriment de la consommation totale, pénalisée par l'ajout de nombreux capteurs et d'actionneurs pouvant atteindre 450W [1] sur un total de 4000W maximum figure III.2, soit plus de 11% de la consommation globale.

1. Il faut noter que cette consommation n'est pas inéluctable et que des technologies comme ENOCEAN ou xBee permettent de réduire réduire quasiment à 0 ces consommations additionnelles

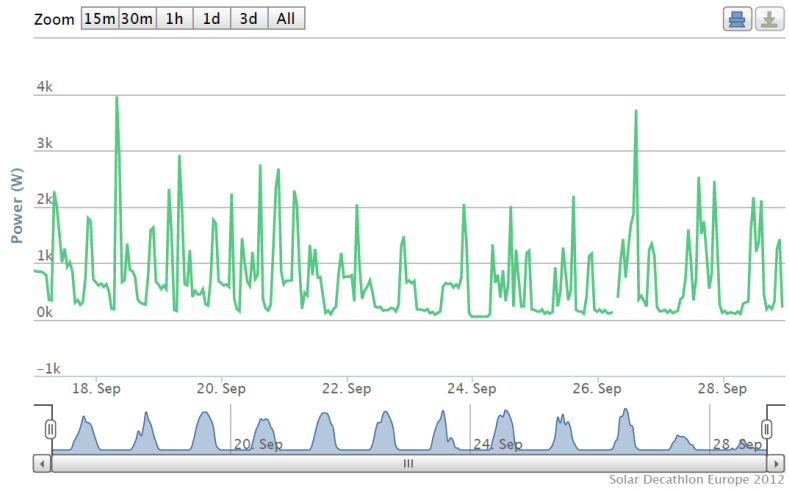



FIGURE *III.2 – Consommation électrique totale de CANOPEA*

Un autre volet important de la compétition *Solar Decathlon* est le confort d'utilisation des habitats présentés :

— confort thermique : 70/120pts, plus de la moitié du poids total du confort
— confort lumineux : 20/120pts
— confort acoustique : 15/120pts
— humidité : 10/120pts
— qualité de l'air à travers la mesure de la quantité de CO2 : 5/120pts.

La température est ainsi le principal critère de confort. Selon (PÉREZ-LOMBARD et al. 2008), le conditionnement thermique (Heating, Ventilation and Air-Conditioning ou acronyme anglais HVAC) peut représenter plus de 50% de la consommation énergétique annuelle juste avant l'ECS et le réfrigérateur dans un habitat. L'importance donnée à l'aspect thermique du confort était un moyen de forcer à travailler sur l'efficacité consommation énergétique / confort thermique résultant.

Dans la partie fonctionnement des équipements (épreuve 6), les points sont attribués en fonction de la performance de chaque équipement électroménager. L'objectif de cette épreuve est d'imposer des équipements à l'ensemble des prototypes en compétition. Répondre à ces contraintes se fait par une sélection rigoureuse d'équipements. Il est à noter que la machine à laver ainsi que le séchoir et le lave vaisselle ont une plage de fonctionnement imposée sans que les instants de démarrage ou de fin soient précisés. Le choix du moment de déclenchement de ces équipements est laissé libre au gestionnaire de la maison, humain ou artificiel. Ce degré de liberté peut ainsi être exploité comme un levier d'équilibrage électrique instantanée de l'habitat en choisissant les moments de déclenchement les plus opportuns.

L'aspect architectural reste important dans le classement de la compétition *Solar Decathlon*. La communication, la faisabilité industrielle, l'innovation et l'ingénierie comptent pour 8% chacun dans la grille de notation.

La grille de notation a constitué une représentation d'exigence d'habitants types. Le BEMS développé pour cette compétition devait donc apporter performance et interactivité avec les occupants. Ce critère éliminait d'emblée les solutions heuristiques, non optimales et surtout difficiles à élaborer pour des systèmes aussi complexes que Canopea. La capacité à adapter facilement la gestion énergétique entrait dans l'épreuve de viabilité économique et industrielle du projet présenté.

## III.2 Le projet Rhône-alpins

FIGURE *III.3 – Maquette du projet nanotours*

"Nano-tours" est le concept architectural présenté par l'équipe Rhône-Alpes pour la compétition SDE2012 [2]. Les nano-tours sont des blocs d'habitations collectifs insérables en milieu urbain. Le concept lie les avantages de l'habitat collectif à ceux de l'habitat individuel. L'habitat collectif permet une forte densification. Il permet une meilleure proximité aux transports, aux lieux de travail ainsi qu'aux réseaux de distribution. L'habitat individuel a pour principal caractéristique le confort à travers l'intimité procurée, des espaces privatifs et de l'exposition au soleil. Le concept "nano-tours" rassemble les avantages des deux types d'habitats : il s'agit d'empiler 5 à 7 étages avec un appartement par étage tout en offrant un espace privé autour de chaque habitat avec jardin à chaque étage. Le bâtiment est prévu pour être connecté à un réseau de distribution électrique "smart-grid", à une boucle d'eau chaude, à un système de collecte de déchets triés et à divers types de transports mutualisés tels que le co-voiturage, le bus, le tramway, et individuels tels que les systèmes de partages de voitures, bicyclettes ou motocycles.

A l'intérieur du bâtiment, des fonctionnalités sont mutualisées à l'image de la production d'énergie solaire avec des panneaux photovoltaïques et hybrides (solaire thermique et photovoltaïque) installés sur le toit ou encore l'eau chaude sanitaire qui est chauffée à l'aide de récupérateurs de calories d'eaux usées. Le chauffage est aussi mutualisé : deux moyens parallèles de chauffage ont été installés pour équilibrer selon les sources disponibles et le confort souhaité.

Certains des équipements mutualisés ont été calibrés pour les 2 étages du prototype construit et non selon les estimations des besoins d'une nano-tour entière. Ainsi, certains réseaux urbains pensés dans le concept nano-tour ne seront pas pratiquement intégrés au

2. Solar Decathlon 2012

fonctionnement du prototype car ils ne sont pas disponibles sur le site qui accueille le prototype dans le cadre de la compétition.

Le prototype Canopea proposé à Madrid à l'échelle 1/1 constitue les deux derniers étages d'une nano-tour. L'avant dernier étage représente un habitat type 2 ou 3 selon la configuration et le dernier étage est un espace de détente commun appelé "Canopée". Il abrite les équipements techniques mutualisés.

Chaque habitat est doté d'un tableau de bord pour le pilotage et la supervision des équipements. Etant donné que ce dernier est construit autour de l'efficacité énergétique, nous utiliserons l'acronyme anglais BEMS pour le qualifier. Le BEMS intègre l'interface vers un réseau local d'aide à la gestion de la vie commune et permet le partage d'équipements tel que le lave-linge et le sèche-linge ou encore l'espace de vie commun "Canopée".

Une règle de la compétition impose que les équipements techniques utilisés soient normalisés, ce qui implique l'utilisation d'équipements grand public ou du moins, ayant passés des tests de vérification de mise aux normes. Des équipements électroménagers grand public ont été utilisées : une plaque vitro-céramique, un réfrigérateur, une télévision ou encore des éclairages basse consommation. Ces équipements ne sont pas communicant mais des mesures de consommation et de puissance sont relevées en temps réel via des wattmètres installés sur le départ des lignes électriques d'alimentation. Les prises d'alimentation sont commandables à distance et peuvent ainsi jouer le rôle d'interrupteur si l'équipement branché supporte une coupure d'alimentation brusque.

## III.3 La gestion énergétique

La gestion énergétique du prototype Canopea vise à réduire la consommation d'énergie tout en maintenant un bon niveau de confort pour les occupants et en cumulant un maximum de points dans les différentes épreuves concernées. Certaines fonctionnalités notamment la gestion inter-habitations ont été développées sur la base de la nano-tours mais qui n'ont aucune utilité à l'échelle du prototype. Les règles de la compétition ne justifiant pas à elles seules l'objectif du BEMS, une étude IFOP a été menée entre décembre 2007 et février 2008 pour le compte de l'association PROMOTELEC sur un panel de 1000 personnes. Le résultat a été que pour plus de 70% des personnes interrogées, les automatismes doivent aider à la protection de l'environnement, la gestion thermique et la réalisation d'économies d'énergie.

Dans le détail, voici un ensemble de données recueillies par cette étude :

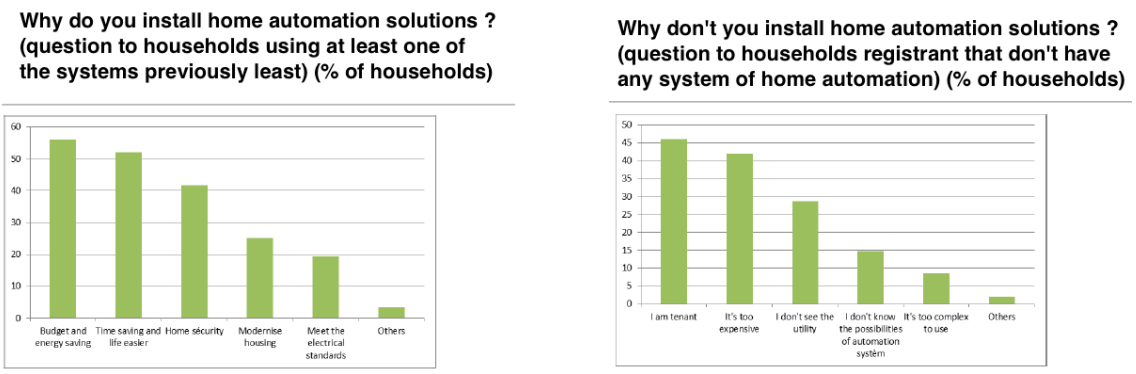



FIGURE *III.4 – Considérations générales sur l'usage d'un BEMS (Source : IFOP enquête réalisée entre décembre 2007 et février 2008 pour PROMOTELEC)*

**Trammels to purchase home automation systèms:**

| | Homeowner | Tenant |
|---|---|---|
| Expensive | 78% | 85% |
| Dependence of a third person in case of failure | 57% | 67% |
| Fear of breakdown | 45% | 51% |
| Complexity of the installation | 37% | 30% |
| Easiness to use | 19% | 12% |

**Motivation for the purchase home automation solutions:**

| | Homeowner | Tenant |
|---|---|---|
| Comfort | 86% | 84% |
| Budget saving | 82% | 87% |
| Life easier | 80% | 79% |
| Environmental protection | 69% | 77% |
| Time saving | 67% | 74% |
| security | 65% | 72% |

FIGURE *III.5 – Motivations et entraves concernant un BEMS (Source : IFOP enquête réalisée entre décembre 2007 et février 2008 pour PROMOTELEC)*

**Interest by type of application : for which fonctions, this home automation system interested you ?**

| | Homeowner | Tenant |
|---|---|---|
| Programming of the heating/cooling systèm | 82% | 86% |
| Leaks detection systèm | 74% | 80% |
| Scenarisation solutions | 71% | 71% |
| Switch | 71% | 70% |
| Alarm | 68% | 44% |
| Rolling shutters | 67% | 65% |
| Appliances programming | 65% | 63% |

FIGURE *III.6 – Fonctionnalités attendues d'un BEMS (Source : IFOP enquête réalisée entre décembre 2007 et février 2008 pour PROMOTELEC)*

L'étude met en évidence trois principales pistes d'amélioration pour les systèmes de gestion actuels. Les tableaux III.5 montrent que les limites, autant pour les propriétaires que pour les locataires, sont le prix de l'installation, la peur d'un système opaque ainsi que la complexité d'installation et d'utilisation. Le BEMS proposé se doit donc de répondre, même en partie, à ces préoccupations.

Le tableau III.6 montre les équipements dont les utilisateurs souhaitent déléguer la gestion. Ces données ont été corrélées aux capacités techniques des équipements, la possibilité de les contrôler automatiquement ainsi qu'aux capteurs nécessaires à l'ajustement de toute commande automatique. Le rapprochement entre les attentes et les possibilités a débouché sur une personnalisation du BEMS GhomeTech spécifique à l'application Canopea.

Le BEMS développé pour le prototype CANOPEA reprend l'architecture de base de GhomeTech en trois couches : anticipative, réactive et terrain. La couche terrain est constituée des systèmes de contrôle d'équipements. Ces systèmes sont chargés d'assurer le suivi de consigne. La couche réactive est assurée par un ensemble de règles heuristiques émises par le logiciel de supervision (Vesta-Energy [3]) d'une manière automatique à pas de temps inférieur au pas de temps anticipatif (différent selon l'équipement commandé).

3. Nom commercial de la solution GHomeTech

L'occupant peut aussi intervenir à travers une interface homme-machine (IHM) qui permet l'interaction entre le système de gestion et son utilisateur. Il peut alors ajuster les Plans d'Utilisation Anticipés générés par la couche anticipative avant leur mise en exécution par la couche réactive. La commande directe depuis la tablette est aussi possible : la commande des volets roulants, de la lumière, certaines prises réglables, la température de consigne...

La couche anticipative (logiciel Vesta-Energy) génère les plans d'usage anticipés à partir de données de prédictions météorologiques, le planning de présence ainsi qu'une représentation virtuelle représentant l'ensemble des paramètres influant sur le comportement énergétique. Cette représentation virtuelle est basée sur le modèle physique de l'enveloppe et les modèles comportementaux des équipements présents. La représentation virtuelle de l'habitat associée aux données d'entrée est formalisée dans milp-workshop, que nous avons développé pour appréhender la complexité de Canopea, et présenté dans les premiers chapitres de cette thèse. Cette formalisation associée à un critère à minimiser permet la conversion du modèle physique en un problème d'optimisation PLNE [4] à travers le module Milp-workshop. Le problème ainsi formalisé est ensuite transmis à un solveur PLNE (CPLEX ou GLPK) communicant avec Vesta-Energy. L'ensemble des étapes est réitéré à des fréquences différentes et nécessite une grande capacité de calcul. Cette capacité de calcul a été externalisée sur un serveur distant mutualisable car la mobilisation de ses capacités de calcul était relativement faible, quelques minutes au plus, par jour pour un problème de 12000 variables sur une machine à 30 coeurs de 2Ghz chacun). Cette externalisation permet un investissement initial moindre ainsi qu'une économie d'échelle dans le cas d'installation du BEMS sur plusieurs habitats.

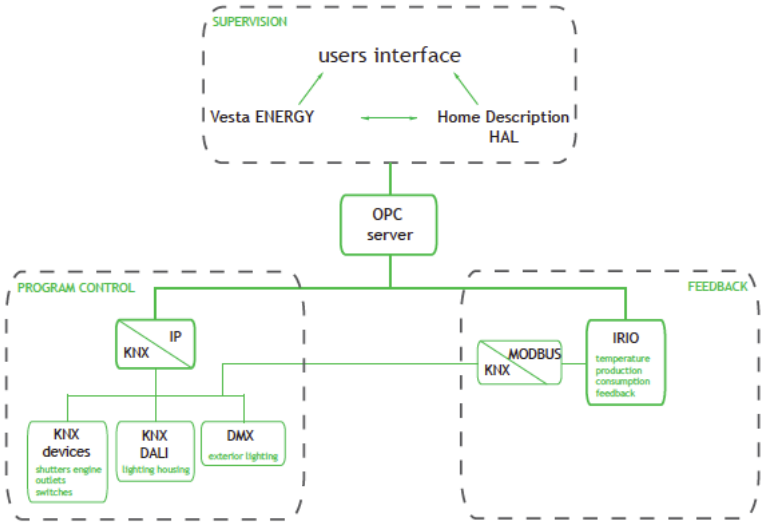



FIGURE *III.7 – Topologie du gestionnaire énergétique*

Le superviseur, comprenant Vesta-Energy (figure III.7) est pensé comme une surcouche à un système déjà installé de type Gestion Technique Bâtiment (GTB). La GTB est composée de deux modules "program control" et "feedback". Ces deux entités sont des modules de communication avec les équipements. On y retrouve une multitude de protocoles : KNX, IP, MODBUS. L'IRIO permet de rassembler les informations remon-

4. programmation linéaire avec nombres entiers

tées en temps réel et émet aussi des ordres de commande aux équipements. Il comporte certaines fonctionnalités qui lui permettent d'implémenter des règles heuristiques mais celles-ci n'ont pas été utilisées.

Le module de supervision que nous avons développé, intègre trois composants : HAL, une IHM et Vesta-Energy.

Le système HAL (Home Abstraction Layer) est une interface de communication homogène de type service Web RESTful (RICHARDSON et al. 2008) vers les capteurs et les actionneurs de Canopea. Il permet le recueil d'informations depuis les systèmes terrains avec des protocoles très diverses tel que modbus, X10, différents protocoles de Web services. Le système HAL a été développé pour contenir tous les pilotes (drivers) nécessaires à l'interfaçage avec les différents protocoles. HAL transmet l'ensemble des mesures, en continu ou sur requêtes, sous format hiérarchisé à VESTA-Energy ainsi qu'à l'IHM. HAL permet aussi l'insertion d'algorithmes pour émuler des capteurs virtuels, autrement dit, des estimateurs. Ainsi, à partir d'autres mesures, HAL peut générer des estimations en continu.

L'IHM développée pour CANOPEA tente de répondre à la demande de transparence formulée par l'échantillon d'utilisateurs potentiels de l'enquête PROMOTELEC. L'IHM a été conçue en collaboration avec des étudiants architectes ainsi que des designers, non techniciens, afin de s'assurer de sa simplicité d'utilisation. L'accent a été mis sur l'interaction entre le système et ses utilisateurs en permettant des modifications des plans d'utilisation anticipés. L'interface permet aussi le déclenchement partiel ou total de l'outil de gestion énergétique à travers des modes : automatique, semi-manuel ou manuel. Elle permet enfin une personnalisation des plans d'utilisation calculés avant leur exécution. Des captures d'écrans ainsi que des explications détaillées sont fournies dans la partie *résultats*.

Vesta-energy est le composant de gestion. La partie anticipative est déportée sur un calculateur distant. En local, un mini-PC industriel opale de la société UXP est inséré dans la boucle. Celui-ci embarque la couche réactive ainsi qu'un module de communication avec le serveur distant.

## III.4 Représentation énergétique du système Canopea

Le travail réalisé sur la représentation virtuelle a été exclusivement réalisé pour la couche prédictive du BEMS. La couche réactive a été reprise d'autres travaux dont (KABANZA et al. 1997). L'objectif de MILP-workshop, que nous avons développé, est de permettre de générer un problème PLNE à partir d'une description de modèles physiques, comportementaux et de données de prédiction. La transformation de cet ensemble d'informations en un problème PLNE requiert des transformations de formulation et de linéarisation ainsi qu'un objectif à optimiser. L'ensemble des mécanismes de transformation et de linéarisation sont décrits dans le chapitre précédent. L'objectif à optimiser dans le cas de Canopea est multi-dimensionnel : coût, confort, respect de la corrélation consommation/production. Cet objectif est en accord avec les règles de la compétition exprimées à travers les points attribués à chaque critère. La pondération entre les différents critères en est directement inspirée. Le résultat, plan d'usage anticipé, ainsi obtenu reflète, dans l'absolu, l'équilibre optimal selon l'objectif choisi. Néanmoins, la confiance donnée à ce résultat est pondérée par la justesse des modèles utilisés et des données externes fournies.

# III.5 Modèles développés pour la gestion du prototype CANOPEA

La représentation virtuelle de l'habitat passe par les modèles de chaque composant physique ou comportemental de l'habitat. Une composition finale lie les variables communes des différents modèles utilisés. La décomposition en unités de modélisation obéit au principe de réutilisabilité : chaque modèle correspond à un élément physique dans l'habitat lorsque celui-ci est clairement défini. Le comportement ainsi que le ressenti (satisfaction) correspondent aussi à des unités de modélisation par type. Par exemple, la courbe générale de satisfaction thermique est l'unité de modélisation et reste la constante (d'un problème de gestion à un autre). La spécification de la température de confort optimale et les bornes de plage de variation autorisée feront du modèle une instance particulière au cas d'utilisation. Le prototype Canopea est d'une complexité peu commune aux habitats actuels. Il est doté d'une enveloppe thermique à haute performance énergétique, un réseau électrique et un réseau thermique complexes, lesquels seront décrits ci-dessous.

Un premier découpage du système complexe a été le découpage selon les interactions. Trois grands groupes, dont les interactions internes sont très fortes, se sont dégagées :

- le réseau électrique figureIII.22 comprenant les panneaux PV, la batterie, la connexion au smart-grid et charges intérieurs.
- le réseau thermique figure III.8 comprenant le déphaseur de température, deux pompes à chaleur : la compact P et la JVP, les stocks thermiques : MCP et ECS et aussi les dissipateurs.
- l'enveloppe figure III.26 comprenant les parois, les planchers, les ouvrants et les occultants

Des modèles de satisfaction qui ne sont compris dans aucun des groupes sont aussi intégrés dans la description finale. Les modèles doivent être suffisamment simplifiés pour pouvoir être intégrés dans MILP-workshop. La comparaison quantitative de variables internes avec les mesures n'était pas tout le temps fondée. Seuls des plans d'usage anticipés acceptables et cohérents donnaient une appréciation des résultats.

La simulation de l'ensemble a été réalisée par parties dans des simulateurs dédiées à chaque discipline. Les réalisations thermiques ont été simulées sur TRNSYS. La production PV et l'ensemble du réseau électrique ont été simulés par le simulateur PVSYST. Contrairement à la simulation, la gestion de l'ensemble des équipements du prototype CANOPEA impose un outil unique et surtout un modèle reprenant l'ensemble des interactions entre les différents systèmes thermiques, électriques, climatiques ainsi que l'occupant et son usage. Les modèles de gestion inspirés des modèles de simulation sont orientés action/réaction car le principal objectif d'un modèle de gestion reste l'anticipation par déduction d'une réaction à chaque action possible. L'aspect qualitatif prime sur l'aspect quantitatif car il n'est pas aisé de faire sortir des anticipations plus précises que les données d'entrée elles mêmes tels que le planning d'usage, le comportement humain ou encore la météorologie. Il était donc important de trouver un niveau de précision adéquat pour les modèles sans surcharger les calculateurs afin d'avoir des résultats acceptables rapidement. Nous avons utilisé les modèles de simulation, les schémas ainsi que les prospectus techniques pour déterminer les degrés de liberté de chaque équipement et ses réactions aux commandes. Dans la partie électrique, nous avons limité les actions possibles aux seules actions sûres et produisant un courant de qualité constante. L'aspect électron n'a pas été abordé dans la gestion. La gestion de puissances/énergies était plus pertinente pour la tâche imposée. La balance énergétique impérativement équilibrée est

le point de liaison des différents flux. La forte complexité du réseau thermique nous a amené à considérer plus de détails que la partie électrique dans la gestion. La modélisation thermique est analogue à la modélisation électrique en prenant pour équivalent de la température du fluide caloporteur la tension (U) et comme équivalent du flux du fluide caloporteur le courant (I). Certains comportements physiques sont non-linéaires et non-linéarisables via les mécanismes présentés dans le chapitre précédent. Ces modèles ont été préalablement simplifiés pour qu'ils puissent être intégrés. Cette simplification peut conduire à certaines approximations ou erreurs lors de l'édition des plans d'usage anticipés mais sont compensés par l'interaction avec l'utilisateur qui peut corriger un plan avant sa mise en exécution ou alors via les mécanismes réactifs et du contrôle local des équipements.

La modélisation ainsi que la composition dans la bibliothèque, constituée par l'ensemble des modèles, suit la hiérarchie des groupes d'interactions précédemment énoncés.

### III.5.1 Modèle du réseau thermique

Le réseau thermique de Canopea utilise comme source externe la chaleur du soleil et la fraicheur de la voûte céleste grâce à des panneaux hybrides (thermique/photovoltaïque). D'autres sources, dont l'énergie qui est véhiculée par une boucle locale d'eau chaude, ont été prévues dans le système mais n'ont pas été implémentées sur le prototype car la boucle n'est pas réalisable sur le site de la compétition. Un déphasage de la température de l'air intérieur de 12h a été rendu possible grâce à un système à base de matériau à changement de phase. L'électricité est utilisée pour faire tourner les pompes à chaleur et les pompes de circulation des différents fluides. Une résistance électrique est installée dans le ballon d'eau chaude sanitaire (ECS) et est utilisée comme source d'appoint. Le système thermique, d'un point de vue utilisateur, régule la température par deux moyens : le rayonnement (à travers les diffuseurs thermiques) et le conditionnement de l'air.

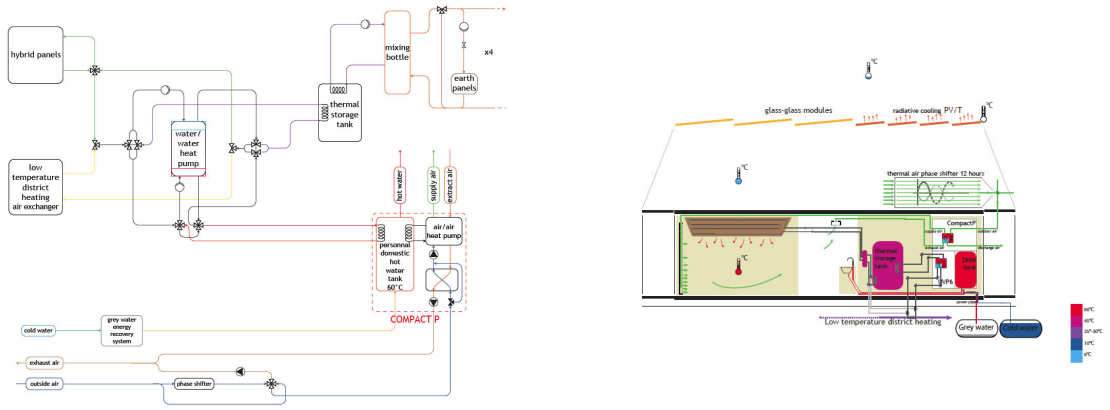



FIGURE *III.8 – Schéma du système thermique*

Le schéma III.8 illustre le réseau de circulation des fluides caloporteurs. Un premier circuit transmet l'énergie entre les panneaux hybrides et la pompe à chaleur JVP. Un second réseau relie la JVP au ballon de stockage thermique et un troisième vers le ballon d'ECS. Un dernier circuit de fluide caloporteur transmet l'énergie depuis le ballon de stockage thermique jusqu'aux panneaux rayonnants à travers une "bouteille d'eau" qui sert à appliquer une loi d'eau.

L'ensemble des réseaux communique avec la pompe à chaleur JVP, sauf le dernier. Une installation intégrée dans la JVP permet la re-configuration des flux autour de la pompe à chaleur à travers des jeux de vannes trois ou quatre voies.

Le second réseau traité dans la modélisation achemine l'air dans l'habitat pour son renouvellement (hormis les fenêtres qui ne sont pas commandables). L'air acheminé est thermiquement conditionné à travers une pompe à chaleur air/air. Cette pompe à chaleur est intégrée en série avec un échangeur statique dans un équipement appelé compact P. La compact P intègre aussi le ballon d'ECS lequel est relié à la pompe à chaleur air-air d'un côté et au réseau à fluide caloporteur de l'autre. Enfin, la compact P puise l'air extérieur à travers le déphaseur ou directement depuis l'extérieur selon la commande reçue.

### III.5.1.1 Panneau Hybride

Les panneaux hybrides ont la double fonction de produire de l'électricité et de la chaleur. Un réseau de fluide caloporteur serpente sous les cellules photovoltaïques des panneaux hybrides pour les refroidir en récupérant la chaleur avec une pompe à chaleur. Le rendement des cellules photovoltaïques est ainsi augmenté (ZONDAG et al. 2002).

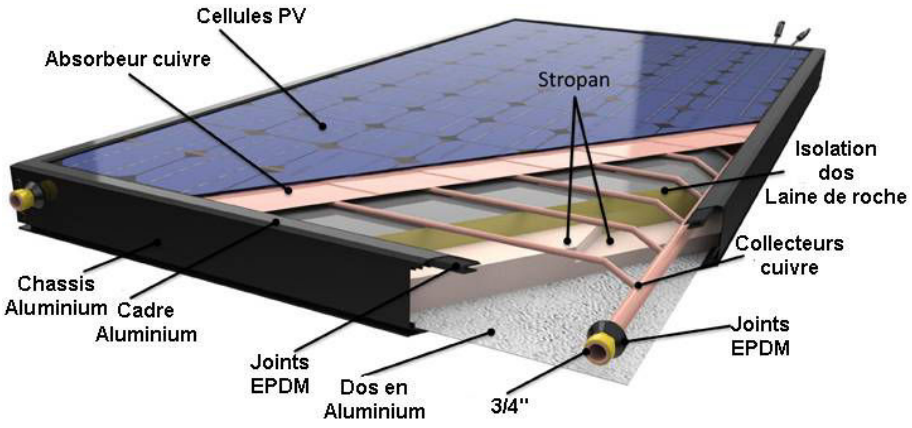



FIGURE *III.9 – Panneau hybride*

Le calcul permettant de trouver la puissance thermique captée est tiré d'un modèle 1D représenté par les équations de Hottel–Whillier (DUFFIE et al. 1980).

$$F_R = \frac{\phi * C_P}{(S_{PV} * U_{loss})} * (1 - e^{-S_{PV} * U_{loss} * F' / (\phi * C_P)}) \tag{III.1}$$

$$P_{Ther} = S_{PV} * F_R * \sigma_{abs} * G + U_{loss} * (T_{outdoor} - T_{Input}) \tag{III.2}$$

La température de sortie du panneau hybride est déduite de l'équation suivante :

$$P_{Ther} = \phi * C_P * (T_{Output} - T_{Input}) \tag{III.3}$$

avec :

$F_R$ : facteur de dissipation de chaleur.

$\phi$ : flux du fluide caloporteur dans le panneau.

$C_P$ : capacité thermique.

$S_{PV}$ : surface d'exposition de panneau hybrid.

$U_{loss}$ : coefficient de transfert thermique.

$F'$ : résistance thermique entre cellules et absorbeur thermique.

$G$ : irradiation solaire surfacique.

$P_{Ther}$ : puissance thermique récupérée.

$T_{Output}$ : température de sortie du fluide caloporteur.

$T_{Input}$ : température d'entrée du fluide caloporteur.

$T_{outdoor}$ : température ambiante autour du panneau.

$\sigma_{abs}$ : rendement solaire thermique du panneau.

Seul un fonctionnement aux conditions nominales est envisagé pour le rendement des cellules photovoltaïques. Le gain obtenu par le refroidissement de celles-ci avec le système de récupération de chaleur est considéré constant.

Le modèle électrique structurel du panneau hybride ne diffère par du modèle d'un panneau photovoltaïque simple. Nous utiliserons donc le modèle PV en composition

L'intégration du modèle dans milp-workshop a été réalisée en deux étapes. La première étape a consisté à intégrer le modèle structurel tel que présenté précédemment avec la définition du type, des bornes inférieures et supérieures pour chacune de ces variables. Les constantes physiques connues y sont aussi définies. La seconde étape a consisté à fixer des valeurs pour certaines variables. Les valeurs de variables estimées théoriquement ou à partir de données terrains fournies sont ainsi renseignées dans une sur-couche au modèle de base. Cette méthode en deux couches permet de garder un modèle structurel largement réutilisable et de le spécialiser au besoin à des situations particulières.

Seules les variables $P_{Ther}$, $T_{Output}$, $T_{Input}$ et $T_{outdoor}$ restent variables au sens de optimisation pour la gestion. Les autres variables deviennent des paramètres définis au cours des différents stades de la modélisation. Dans le modèle du panneau hybride, l'ensemble des variables servant à calculer $F_R$ sont définies. Néanmoins la valeur de $F_R$ doit aussi être exprimée en tant que paramètre car à défaut, des non-linéarités apparaissent dans d'autres équations. Dans le modèle présenté, le cas ne s'est pas présenté car les variables multipliés par $F_R$ sont aussi des paramètres. Dans la dernière équation, il est impératif de fixer le débit entrant dans le panneau hybride $\phi$ afin d'avoir une équation linéaire. Un débit continument variable aurait imposé une non-linéarité avec la température, continument variable. Cette non-linéarité ne peut être levée par les schémas de linéarisation développés dans milp-workshop. Il n'était pas nécessaire de mettre une commande de fermeture de vanne au niveau du panneau hybride car chaque vanne est rattachée à l'équipement exploitant le fluide sortant, la pompe à chaleur JVP dans le cas de Canopea. En cas de fermeture de la vanne au niveau JVP, le modèle du panneau hybride donnera alors des températures d'entrée et de sortie erronées. Cette erreur n'influe en rien sur le plan d'usage anticipé car elle n'est pas prise en compte en cas d'inutilisation du panneau hybride et, de surcroit, l'ajout de variables ralenti le calcul.

Une dernière partie des variables dans le modèle représente des connexions avec d'autres modules ou encore des données d'entrée tel que $G$. Ces variables sont connectées aux bases de données et fixées avant chaque nouveau lancement de calcul.

#### III.5.1.2 Pompe à chaleur

La pompe à chaleur est une machine basée sur le premier principe de la thermodynamique. Un gaz circule entre deux serpentins traversant une source froide et une source chaude grâce à une pompe, appelée compresseur, et un détendeur. Sous l'effet de la compression, l'enthalpie augmente et donc aussi la température du gaz. Le gaz passe ensuite à travers la source chaude pour se refroidir au contact, indirect via un échangeur, d'un fluide externe puis revient vers la source froide à travers une vanne de détente III.10.

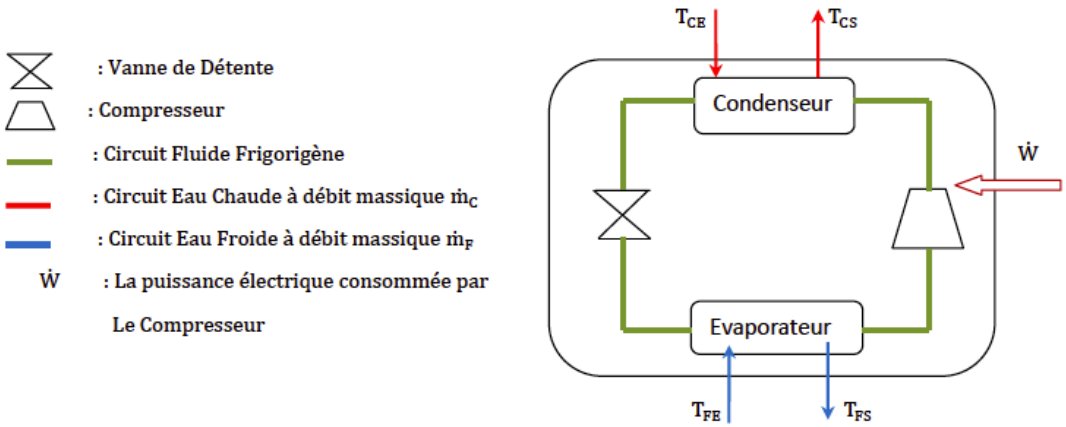



FIGURE *III.10 – Principe de fonctionnement d'une pompe à chaleur*

Le modèle générique le plus simplifié d'une pompe à chaleur est basé sur son bilan énergétique. Il repose sur le gain de la machine appelé $COP$ (coefficient de performance) ainsi que les températures des différents fluides externes.

$$\phi_h = \dot{m}_h C_p (T_{out}^{cond} - T_{in}^{hot}) \tag{III.4}$$

$$\phi_c = \dot{m}_c C_p (T_{out}^{cold} - T_{in}^{cold}) \tag{III.5}$$

$$\phi_{elec} = \phi_h + \phi_c \tag{III.6}$$

$$COP * \phi_{elec} = \phi_h \tag{III.7}$$

$$\phi_{elec} = \delta_{on}^{off} * P_{elec}^{compressor} \tag{III.8}$$

variables et paramètres du modèle :

$T_{in}^{hot}$ : temperature du fluide froid injecté

$T_{out}^{hot}$ : temperature du fluide chauffé (rejeté)

$T_{in}^{cold}$ : temperature du fluide chaud injecté

$T_{out}^{cold}$ : Temperature du fluide refroidi (rejeté)

$COP$ : efficacité de la pompe à chaleur.

$\delta_{on}^{off}$ : commande du compresseur (le système)

$P_{elec}^{compressor}$ : consommation nominale du compresseur

Le modèle de la pompe à chaleur est un modèle élémentaire réutilisé dans la modélisation d'équipements plus complexes intégrant une pompe à chaleur telle que la pompe à chaleur JVP ou encore la pompe à chaleur compact P. Seule la variable représentant la chaleur spécifique $C_p$ est fixée à ce stade de la modélisation.

La vue globale extérieure du modèle est schématisée par la représentation III.21.


FIGURE *III.11 – Modèle du composant pompe à chaleur*

### III.5.1.3 Modèle de l'échangeur statique air/air :

Un échangeur statique est un élément d'échange de calories direct, par opposition au principe d'une PAC, à travers une surface de contact entre deux fluides. Les équations physiques d'un échangeur statique sont basées sur le principe de conservation de l'énergie et l'expression du gain de l'échangeur, c'est-à-dire sur l'énergie échangée entre les fluides traversant l'échangeur. Le modèle proposé est prévu pour l'air ainsi que pour les fluides gazeux sous pression atmosphérique.

L'équation de conservation d'énergie énonce que l'énergie extraite du premier fluide traversant l'échangeur est intégralement transmise au second fluide le traversant.

$$\rho_{air} c_{air} Q_{pul}(T_{pul} - T_{out}) = \rho_{air} c_{air} Q_{exp}(T_{in} - T_{exp}) \tag{III.9}$$

Le rendement d'un échangeur statique est fixe contrairement au rendement d'un échangeur rotatif, lequel peut évoluer selon la vitesse de rotation de la roue d'échange thermique. Le gain est le rapport entre l'énergie récupérée et l'énergie maximale récupérable :

$$\zeta = \frac{\rho_{air} c_{air} Q_{pul}(T_{pul} - T_{out})}{\rho_{air} c_{air} min(Q_{pul}, Q_{exp})(T_{in} - T_{out})} \tag{III.10}$$

$$\zeta = \frac{\rho_{air} c_{air} Q_{exp}(T_{in} - T_{exp})}{\rho_{air} c_{air} min(Q_{pul}, Q_{exp})(T_{in} - T_{out})} \tag{III.11}$$

avec :

$\rho_{air}$ : masse volumique de l'air

$c_{air}$ : capacité calorifique de l'air

$Q_{pul}$ : débit de l'air pulsé (premier fluide traversant)

$T_{pul}$ : température d'entrée de l'air pulsé

$T_{out}$ : température de sortie de l'air pulsé

$Q_{exp}$ : débit de l'air expulsé (second fluide traversant)

$T_{in}$ : température d'entrée de l'air expulsé

$T_{exp}$ : température de sortie de l'air expulsé

$\zeta$ : le rendement de l'échangeur qui est fourni par le constructeur dans la fiche technique. Ce paramètre peut être ajusté par une identification s'il y a un jeu de données car il dépend aussi des conduits d'acheminement de l'air depuis et vers l'échangeur.

Le volume d'air est constant dans un volume donné à pression atmosphérique. Par conséquent, le volume d'air prélevé par un système de renouvellement d'air doit être impérativement remplacé par le même volume d'air renouvelé. Dans le cas contraire, des fuites d'air existent nécessairement pour équilibrer le volume à pression atmosphérique constante. Dans cette hypothèse, nous pouvons donc supposer que $Q_{exp} = Q_{pul}$. Dans ce cas, les équations (III.10) et (III.11) deviennent :

$$T_{pul} = \zeta T_{in} + (1 - \zeta) T_{out} \tag{III.12}$$

$$T_{exp} = \zeta T_{out} + (1 - \zeta) T_{in} \tag{III.13}$$

Le modèle résultant est mis sous la forme de la représentation globale externe figure III.12.

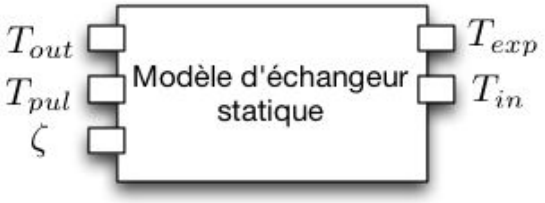



FIGURE *III.12 – Modèle composant de l'échangeur statique*

La modélisation milp-workshop impose qu'il n'y ait pas de non-linéarités. Les équations (III.10) et (III.11) ne peuvent pas être intégrées directement car elles font apparaître

un rapport de variables. Ce rapport pourrait être supprimé en multipliant par $\zeta$ lequel est ensuite considéré constant. Cette manipulation ne peut pas être assurée par l'optimiseur CPLEX directement : c'est au générateur de problème d'optimisation milp-workshop de gérer cette transformation qui constitue un autre type de manipulation algébrique qui sera développé en seconde partie du manuscrit.

Les équations (III.12) et (III.13) sont complètes et représentatives. Elles sont utilisées dans la modélisation milp-workshop. $\zeta$ est la seule variable à fixer dans ce modèle.

#### III.5.1.4 Stockage thermique

Le stockage thermique est basé sur le principe de conservation de l'énergie. Un fluide est mis dans un conteneur adiabatique. Le fluide contient l'énergie sous forme de chaleur sensible dans le cas d'un fluide caloporteur ou de chaleur latente et de chaleur sensible dans le cas d'un matériau à changement de phase telle que la paraffine.

L'énergie contenue est fournie par des sources externes à travers des échangeurs de chaleur ou par une source interne dans le cas d'un système autonome de production de chaleur. Dans le cas de l'ECS, le fluide stocké est renouvelé à chaque évacuation de façon à maintenir le conteneur constamment rempli. La base de modélisation du stockage thermique est le premier principe de la thermodynamique.

$$E_{stock}(t) = E_{stock}(t-1) + \sum E_{in} - \sum E_{out} + E_{produced} \tag{III.14}$$

Il reste à déterminer la somme des échanges thermiques avec le milieu extérieur ainsi que la production interne.

1. Dans le cas d'un stockage thermique conventionnel, l'ensemble des échanges thermique est sous forme de chaleur sensible et induit donc l'élévation de la température du fluide et, en contrepartie, un refroidissent des fluides entrants dans le cas de chauffage et inversement dans le cas de refroidissement.
   La projection de l'équation (III.14) donne :

$$E_{stock}(t) = m * C_p * T_{stock}(t-1) + Te * C_p \sum_i (\dot{m}_i(t) * (T_i^{out}(t) - T_i^{in}(t))) + E_{produced}(t) \tag{III.15}$$

   L'équation suivante est toujours vérifiée dans ce cas :

$$E_{stock}(t) = m * C_p * T_{stock}(t) \tag{III.16}$$

2. Dans le cas d'un matériau à changement de phase une partie de l'énergie est stockée durant la période de changement de phase sous forme de chaleur latente. La température du matériau n'augmente pas mais sa structure moléculaire change. Il est donc nécessaire de distinguer cette phase (figure III.13) :

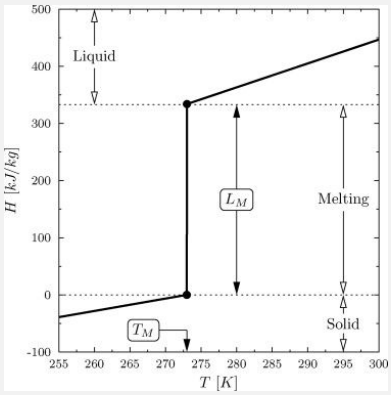



FIGURE *III.13* – *Evolution de l'énergie stockée en fonction de la température d'un MCP (*STÉPHANE *et al. 2012)*

La projection de l'équation (III.14) ne peut directement fournir la température du stock.

$$E_{stock}(t) = E_{stock}(t-1) + T_e C_p \sum_i (\dot{m}_i(t) * (T_i^{out}(t) - T_i^{in}(t))) + E_{produced(t)} \quad \text{(III.17)}$$

Pour déterminer la température du stock et ainsi avoir la température de retour fluide est déduite par la fonction linéaire par partie :

$$\begin{cases} E_{stock}(t) - E_{stock}(0) = mC_p(T_{stock}(t) - T_{stock}(0)) & \text{if} E_{stock}(t) \leq E_{stock}^{H0} \\ T_{stock}(t) = T_M & \text{if} E_{stock}^{H0} \leq E_{stock}(t) \leq E_{stock}^{Hmax} \\ E_{stock}(t) - E_{stock}^{Hmax} = m * Cp * (T_{stock}(t) - T_M) & \text{if} E_{stock}(t) \geq E_{stock}^{Hmax} \end{cases} \quad \text{(III.18)}$$

Le modèle de stockage thermique ainsi proposé doit être spécialisé. Le nombre de circuits d'eau intervenant dans le stockage $i$ est à définir. Dans le cas de l'ECS, le remplacement d'eau chaude consommée par de l'eau à température ambiante (source externe) est considéré comme un circuit d'eau supplémentaire. Cette représentation a néanmoins une limite : la non-linéarité induite par la multiplication du débit $\dot{m}$ par les températures du fluide dans le stockage. Pour pallier à cette non-linéarité, la variable $\dot{m}$ a été discrétisée suivant 3 modes :

— aucune utilisation dans l'heure $\dot{m} = 0$
— une douche dans l'heure $\dot{m} = \frac{V_1}{t_{flow}}$
— usage cuisine $\dot{m} = \frac{V_2}{t_{flow}}$

Cette discrétisation permet l'application du schéma de linéarisation discret-continu dans MILP-workshop.

Le modèle global résultant est schématisé dans la figure III.14.

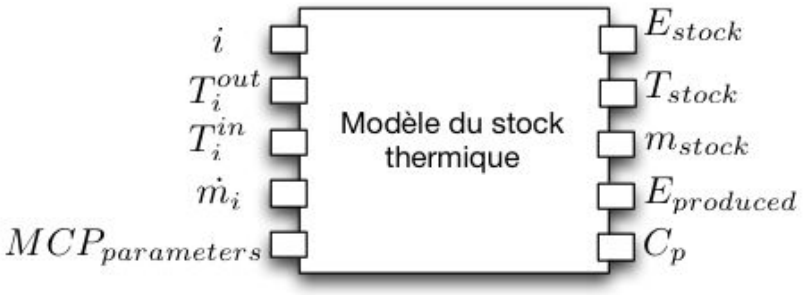



FIGURE *III.14 – Modèle composant du stockage thermique*

### III.5.1.5 Bouteille de brassage

La bouteille de brassage est un composant du réseau fluide thermique installé en amont de la boucle d'eau vers les panneaux rayonnants. La bouteille de brassage a pour but de réguler la température du fluide caloporteur à l'entrée des panneaux rayonnants autour d'une température de consigne. Le réseau étant réversible, c'est-à-dire qu'il peut chauffer ou refroidir, les températures de consignes possibles sont multiples.

Cette température de consigne est déterminée par le système de gestion. Selon le type de commande du matériel, la consigne peut être choisie parmi une gamme de températures pré-établis ou bien déterminée par le gestionnaire selon le besoin exact dans les limites imposées par le matériel. La modélisation du régulateur de température est basée sur le bilan énergétique des flux. La régulation se fait sur la base du mélange entre fluide en entrée de la bouteille et le retour fluide depuis les panneaux rayonnants. Le débit total à

l'entrée de la bouteille est le même que le débit en sortie de celle-ci vers les panneaux rayonnants.

L'équilibre énergétique n'est porté que par le brassage des fluides.

$$\dot{m}_{in} C_p (T_{in}^{supp} - T_{out}^{supp}) = \dot{m}_{out} C_p (T_{out}^{pannels} - T_{in}^{pannels}) \tag{III.19}$$

Sachant que $\dot{m}_{in} = \dot{m}_{out}$, le bilan énergétique peut s'exprimer alors uniquement avec les températures

$$T_{in}^{supp} - T_{out}^{supp} = T_{out}^{pannels} - T_{in}^{pannels} \tag{III.20}$$

La température de consigne n'est pas atteignable dans le cas d'un stock insuffisant. Ce point a aussi été modélisé :

$$\begin{cases} T_{out}^{pannels} = T_{setpoin} & \text{; if } (T_{in}^{supp} \geq T_{out}^{supp} \wedge T_{setpoint} \leq T_{in}^{supp}) \vee (T_{in}^{supp} \leq T_{out}^{supp} \wedge T_{setpoint} \geq T_{in}^{supp}) \\ T_{out}^{pannels} = T_{in}^{supp} & \text{; otherwise.} \end{cases} \tag{III.21}$$

avec :

$T_{in}^{supp}$ : température en entrée du coté fournisseur d'énergie

$T_{out}^{supp}$ : température en sortie du coté fournisseur d'énergie.

$T_{out}^{pannels}$ : température en sortie du coté panneaux de rayonnement.

$T_{in}^{pannels}$ : température en entrée du coté panneaux de rayonnement.

$T_{setpoint}$ : température de consigne pour l'alimentation des rayonnants.

La structure "if-then-else" implique une causalité qui ne peut être exprimée en PLNE. Cette causalité est considérée comme un verrou à l'exécution du modèle. Le verrou est détourné via une expression conditionnelle. Cette expression associe une variable binaire à chacune des deux propositions. La variable binaire est, elle, associée à la condition de la proposition adéquate par une inégalité. Le résultat est que la variable binaire est mise à $1$ si la condition est vérifiée, et à $0$ sinon. L'instruction attachée à la condition n'est exécutée que dans le cas où la variable binaire est mise à $1$.

#### III.5.1.6 Panneau rayonnant

Le panneau rayonnant est un échangeur de chaleur statique de type eau/air. Les paramètres de l'équipement ne peuvent être directement récupérés des fiches techniques car le panneau est noyé dans la masse (sol ou plafond). Les paramètres ont donc été estimé depuis des simulations dynamiques. Les grandeurs prises en compte dans le modèle du panneau rayonnant sont la température d'entrée et de sortie du fluide caloporteur ainsi que la température ambiante. Les paramètres du modèle de gestion sont la surface d'échange, le flux du fluide caloporteur ainsi que le coefficient d'échange de chaleur.

La première équation permet d'exprimer le lien entre la différence de températures du fluide caloporteur et la puissance dégagée.

$$P_{exchanged} = \delta C_{calo} \phi (T_{input}^{calo} - T_{output}^{calo}) \tag{III.22}$$

Ensuite, la seconde équation exprime le lien entre la puissance dégagée par le panneau rayonnant et le différentiel entre température moyenne du panneau rayonnant et la température ambiante.

$$P_{exchanged} = S_{echange} Coeff_{thermique} (\frac{T_{input}^{calo} + T_{output}^{calo}}{2} - T_{ambiante}) \tag{III.23}$$

La dernière équation permet de modéliser la commande tout ou rien de l'ouverture / fermeture de la vanne de circulation à l'entrée du panneau rayonnant.

$$T_{input}^{calo} - \delta(t) * T_{input}^{calo} = T_{output}^{calo} - \delta(t) * T_{output}^{calo} \tag{III.24}$$

avec :

$T_{input}^{calo}$ : température à l'entrée du panneau rayonnant.

$T^{calo}_{output}$ : température à la sortie du panneau rayonnant.
$T_{ambiante}$ : température ambiante dans l'espace tempéré.
$P_{exchanged}$ : puissance thermique échangée
$\phi$ : débit de circulation du fluide caloporteur dans le panneau.
$\delta$ : variable binaire de commande ouverture fermeture de vanne d'arrivée du fluide caloporteur.
$S_{echange}$ : surface d'échange
$Coeff_{thermique}$ : coefficient d'échange thermique.
$C_{calo}$ : capacité thermique du fluide caloporteur.

D'un point de vue modélisation dans milp-workshop, il y a une non linéarité de type produit binaire $\delta x T^{calo}_{output}$ qui est levée par le schéma de linéarisation correspondant. Il y a un produit non-linéaire et non-linéarisable dans l'état $\phi x T_{output}$. La variable $\phi$ peut être discrétisée pour les besoins de la modélisation. Il s'agira alors d'une bridage dans le système car l'énergie injectée ne sera plus continument variable. La seconde solution proposée, pour corriger le modèle à insérer dans le BEMS, est la suppression de l'équation (III.23) du modèle. Cette équation sert à déterminer la quantité d'énergie extractible du panneau rayonnant d'une manière dynamique. En supprimant cette équation et en la remplaçant par un maximum extractible fixe, nous perdons de l'information mais nous pouvons dans ce cas utiliser une commande en puissance directement sur les panneaux rayonnants en laissant la traduction de la commande en énergie à la boucle d'asservissement locale. Cette boucle reçoit une consigne issue du plan de gestion énergétique émis de la couche anticipative et le traduit en commande de la vanne de débit à l'intérieur du pas de temps anticipatif.

Les variables $S_{echange}$, $Coeff_{thermique}$, $C_{calo}$ sont fixées comme des paramètres du système.

#### III.5.1.7 Déphaseur de température

Le déphaseur de température est un équipement mis en oeuvre à l'EPFL de Lausanne (HOLLMULLER 2003), (HOLLMULLER et al. 2014), (CAMPANIÇO et al. 2014), (BRUN et al. 2013). Le déphaseur utilise l'inertie thermique d'une masse de matériau à changement de phase pour décaler la température d'ambiance de 12h approximativement afin de profiter des décalages de température jour/nuit durant certaines périodes de l'année selon les régions.

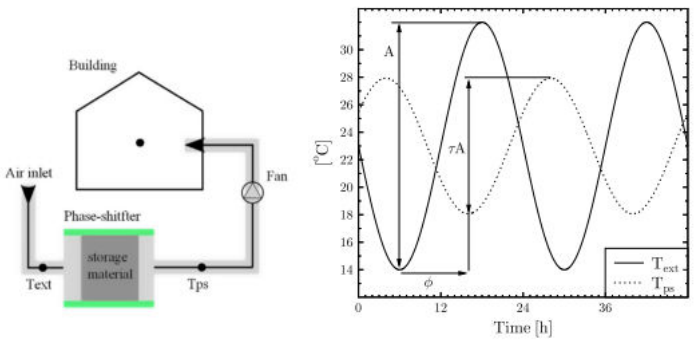



FIGURE *III.15 – Principe de fonctionnement du phase shifter (*BRUN *et al. 2013)*

Un courant d'air est constamment soufflé sur une masse de matériau à changement de phase figure III.15. L'air entre à la température extérieure, échange de l'énergie avec le

matériau, disposé en lames parallèles pour maximiser l'échange entre l'air et le matériau. Ainsi, en sortie, l'air est à la température du matériau corrigée par un coefficient de rendement $\rho$. Lorsque l'équipement est correctement calibré (masse du matériau justement dosée), la température de sortie correspond à la température de l'air extérieur propulsé $\Delta t$ avant, ce qui correspond à la durée de déphasage.

Le modèle de gestion reprend la philosophie du modèle physique en utilisant les deux caractéristiques fondamentales du déphaseur : $\rho$ et $\Delta t$. Néanmoins, son fonctionnement au point nominal n'est garanti que lorsque la pompe à air a été fonctionnelle depuis au moins un cycle entier $2x\Delta t$. Le déphaseur est donc continuellement fonctionnel mais un système de vannes permet de le court-circuiter si la température de sortie du déphaseur est moins intéressante que la température extérieure directe.

Le modèle du déphaseur est représenté par l'équation :

$$T_{out}(t) = \rho T_{in}(t - \Delta t) \tag{III.25}$$

Le système avec court-circuit est représenté par l'équation III.26 dans laquelle une variable $bypass$ a été insérée.

$$T_{out}(t) = \rho T_{in}(t - \Delta t)(1 - bypass(t)) + T_{in}(t) bypass(t) \tag{III.26}$$

$\rho$ et $\Delta t$ sont les paramètres à définir pour le modèle.

#### III.5.1.8 Pompe à chaleur Compact P

La pompe à chaleur Compact P est le nom commercial d'un équipement de conditionnement d'air produit par la société NILAN.

La compact P est constituée d'une pompe à chaleur couplée à un échangeur statique utilisé en pré-traitement de l'air. La compact P standard est prévue pour embarquer des options supplémentaires. La version utilisée dans Canopea est équipée d'un ballon d'eau chaude sanitaire (ECS) (figure III.16). Le ballon d'ECS est relié d'un côté à un apport externe à la compact P, la JVP, et de l'autre à la pompe à chaleur de la compact P comme montré dans le schéma III.16.

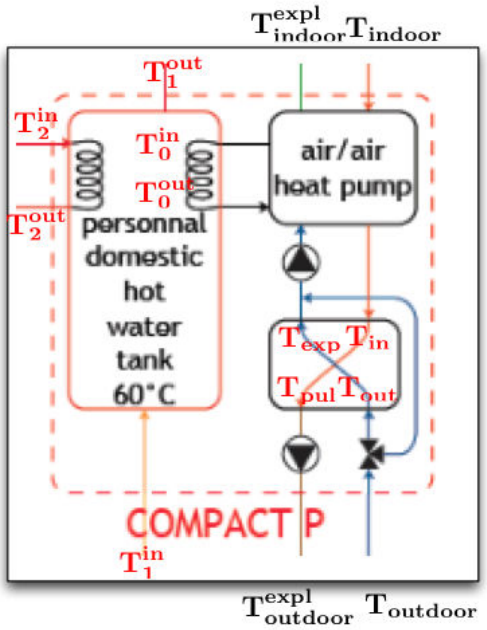



FIGURE *III.16 – Principe de fonctionnement de la compact P*

La modélisation de la compact P fait appel aux modèles d'échangeur statique, de pompe à chaleur et de stockage thermique. Le modèle global de la Compact P lie les variables de ces trois modèles élémentaires à travers une connexion directe "connect" entre variables ou avec des équations mathématiques propres.

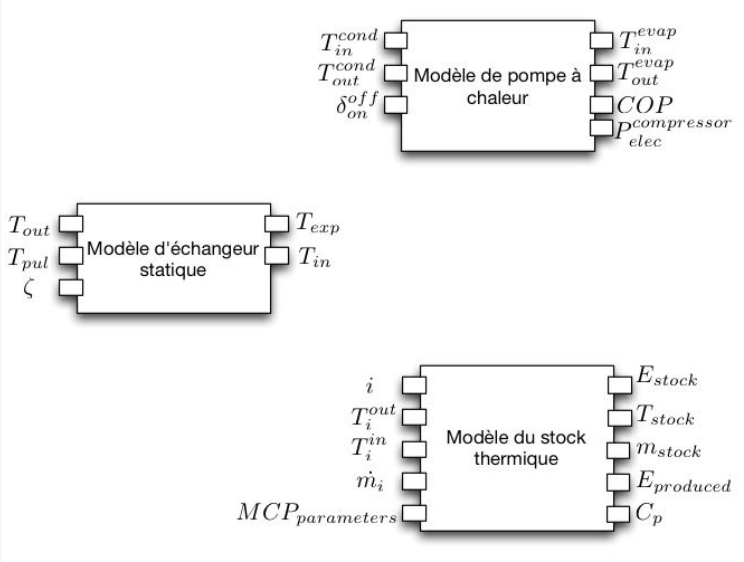



FIGURE *III.17 – Composition de modèles externes pour la Compact P*

La compact P est soumise à cinq différents modes de fonctionnement en plus du mode arrêt. Ces modes sont modélisés avec des variables binaires 1 si le mode correspondant est adopté et, 0 sinon.

$mode_{BP}$ : mode ByPass (free ventilation).

$mode_{AAH}$ : mode Active Air Heating.

$mode_{PAH}$ : mode Passive Air Heating.

$mode_{AAC}$ : mode Active Air Cooling.

$mode_{HWP}$ : mode Hot Water Production.

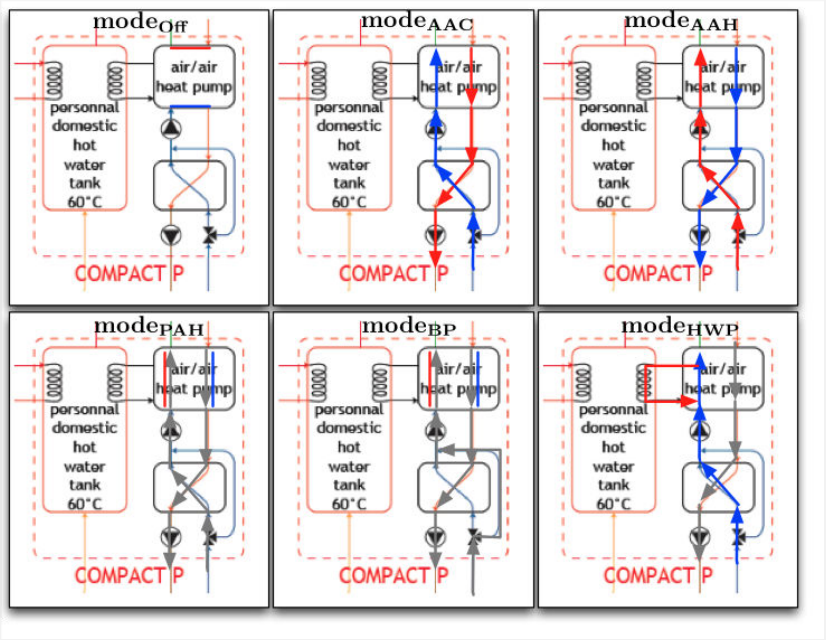



FIGURE *III.18 – Direction des fluides selon le mode dans la Compact P*

Le choix exclusif est exprimée par l'équation :

$$mode_{Off}(k) + mode_{BP}(k) + mode_{AAH}(k) + mode_{PAH}(k) + mode_{AAC}(k) + mode_{HWP}(k) = 1 \quad \text{(III.27)}$$

La connexion entre la variable marche / arrêt $\delta_{on}^{off}$ de la PAC et les actions des modes est exprimé par l'équation :

$$\delta_{on}^{off} = 1 - (mode_{PAH} + mode_{Off} + mode_{BP}) \quad \text{(III.28)}$$

Chaque mode exige une connexion particulière de la pompe à chaleur au reste des équipements. La synthèse est apportée par la série d'équations suivantes :

$$\begin{aligned} T_{in}^{hot} =& T_{indoor} mode_{Off}(k) + T_{outdoor} mode_{BP}(k) + T_{exp}(mode_{AAH}(k) + ... \\ & ... + mode_{PAH}(k)) + T_{indoor} mode_{AAC}(k) + T_0^{out} * ode_{HWP}(k) \end{aligned}$$

$$\begin{aligned} T_{out}^{hot} =& T_{indoor}^{expl} mode_{Off}(k) + T_{indoor}^{expl} mode_{BP}(k) + ... \\ & ... + T_{indoor}^{expl}(mode_{AAH}(k) + mode_{PAH}(k)) + T_{in} * mode_{AAC}(k) + T_0^{in} mode_{HWP}(k) \end{aligned}$$

$$\begin{aligned} T_{in}^{cold} =& T_{in} mode_{Off}(k) + T_{indoor} mode_{BP}(k) + ... \\ & ... + T_{indoor}(mode_{AAH}(k) + mode_{PAH}(k)) + T_{exp}(mode_{AAC}(k) + mode_{HWP}(k)) \end{aligned}$$

$$\begin{aligned} T_{out}^{cold} =& T_{exp} mode_{Off}(k) + T_{outdoor}^{expl} mode_{BP}(k) + ... \\ & ... + T_{in} * (mode_{AAH}(k) + mode_{PAH}(k)) + T_{indoor}^{expl}(mode_{AAC}(k) + mode_{HWP}(k)) \end{aligned}$$

$$
\begin{aligned}
T_{out} mode_{AAH}(k) &= T_{outdoor} mode_{AAH}(k) \\
T_{pul} mode_{AAH}(k) &= T^{expl}_{outdoor} * mode_{AAH}(k) \\
T_{out} mode_{PAH}(k) &= T_{outdoor} mode_{PAH}(k) \\
T_{pul} mode_{PAH}(k) &= T^{expl}_{outdoor} * mode_{PAH}(k) \\
T_{out} mode_{AAC}(k) &= T_{outdoor} mode_{AAC}(k) \\
T_{pul} mode_{AAC}(k) &= T^{expl}_{outdoor} * mode_{AAC}(k) \\
T_{out} mode_{HWP}(k) &= T_{indoor} mode_{HWP}(k) \\
T_{pul} mode_{HWP}(k) &= T^{expl}_{outdoor} * mode_{HWP}(k) \\
T^{out}_0 mode_{Off} &= mode_{Off} T^{in}_0 \\
T^{out}_0 mode_{AAC} &= mode_{AAC} T^{in}_0 \\
T^{out}_0 mode_{AAH} &= mode_{AAH} T^{in}_0 \\
T^{out}_0 mode_{PAH} &= mode_{PAH} T^{in}_0 \\
T^{out}_0 mode_{BP} &= mode_{BP} T^{in}_0
\end{aligned}
$$

avec :

$T^{hot}_{in}$ : temperature du fluide froid injecté

$T^{hot}_{out}$ : temperature du fluide chauffé (rejeté)

$T^{cold}_{in}$ : temperature du fluide chaud injecté

$T^{cold}_{out}$ : Temperature du fluide refroidi (rejeté)

La connexion des variables de débits est apportée par l'équation :

$$
\begin{aligned}
\dot{m}_h(k) &= 0 mode_{Off}(k) + \phi_{air}(k)(mode_{BP}(k) + mode_{AAH}(k) + mode_{PAH}(k) + mode_{AAC}(k)) + \dot{m}_0(k) mode_{HWP}(k) \\
\dot{m}_c(k) &= 0 mode_{Off}(k) + \phi_{air}(k)(mode_{BP}(k) + mode_{AAH}(k) + mode_{PAH}(k) + mode_{AAC}(k)) + \dot{m}_0(k) mode_{HWP}(k)
\end{aligned}
$$

Enfin, la résistance de fuite générée par le flux d'air est calculé entre les températures intérieure $T_{indoor}$ et la température de l'air pulsé à l'intérieur $T^{expl}_{indoor}$

$$
R_v(k) = \frac{1}{\phi_{air}(k) * C^{air}_p} \tag{III.29}
$$

Le modèle est composé des variables suivantes :

— 16 variables de température continues-temporelles

— 7 variables de décision binaires- temporelles

— 6 variables diverses dont 5 discrètes-temporelles et un paramètre.

A cela, s'ajoutent les variables et paramètres n'intervenant pas directement dans le modèle de la compact P mais encapsulés dans les sous-modèles de pompe à chaleur, stockage thermique et échangeur statique.

**Pour le modèle de la pompe à chaleur :** 1 variable discrète-temporelle et un paramètre.

**Pour le modèle de stock thermique :** 2 variables discrètes-temporelles et 7 variables continues-temporelles.

**Pour le modèle de l'échangeur statique :** un paramètre supplémentaire.

Une fois projeté sur l'horizon de temps de la couche prédictive de 24h avec un pas de temps de 1h, l'ensemble des variables intervenant dans ce seul modèle est de 552 variables continues, 168 variables binaires, 168 variables discrètes et 3 paramètres.

L'utilité de MILP-workshop tool à travers MILP-workshop language est de faire abstraction de cette complexité en offrant une vue hiérarchisée sur le problème. Ainsi, la

modélisation de cet équipement sans l'outil, directement dans un langage de programmation linéaire, aurait été très difficile avec une complexité telle que toute erreur serait difficile à localiser.

#### III.5.1.9 Pompe à chaleur JVP

"JVP" est le nom commercial d'une pompe à chaleur réversible fabriquée par NILAN. Elle est constituée d'une pompe à chaleur eau/eau à large amplitude de température entourée d'un réseau de conduits (figure III.19), qui permet de configurer les orientations des circuits de fluides caloporteurs. La tuyauterie offre un second niveau de configuration qui augmente la capacité de connexion à quatre circuits. La pompe à chaleur réversible JVP permet des reconfigurations dynamiques du réseau. Pour ce faire, une première couche de commande intervient à l'intérieur de l'équipement. La couche commande interdit certaines combinaisons de positions de vannes qui peuvent mettre en danger ou détériorer l'équipement. Cette même couche offre une série de "modes" de contrôles pour l'ensemble de fonctionnalités possibles par la machine (voir figure III.20).

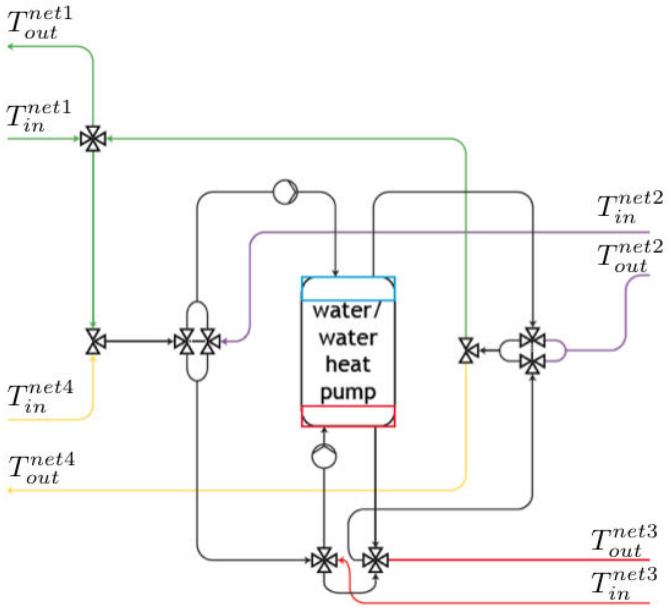



FIGURE *III.19 – Principe de fonctionnement de la pompe à chaleur JVP*

Le modèle de gestion développé pour la pompe à chaleur JVP associe les modes aux comportements correspondants du système de chauffage / refroidissement. La gestion par mode est une gestion discontinue qui associe des variables de contrôle binaires mais aussi discrètes et continues. La gestion par mode justifie en outre l'utilisation de la programmation linéaire en nombres entiers car elle permet d'appréhender simultanément des variables de contrôle continues et discrètes. La JVP offre six modes de fonctionnement (figure III.20).

Le modèle de la JVP est décomposé en deux parties : la pompe à chaleur elle-même et les circuits reconfigurables. Le modèle de pompe à chaleur est développé dans un composant à part entière (décrit ci-dessus). Le modèle de pompe à chaleur élémentaire est directement repris dans la description MILP-workshop grâce au mot clés : "load" et l'appel à des variables se fait depuis le modèle de la JVP directement.

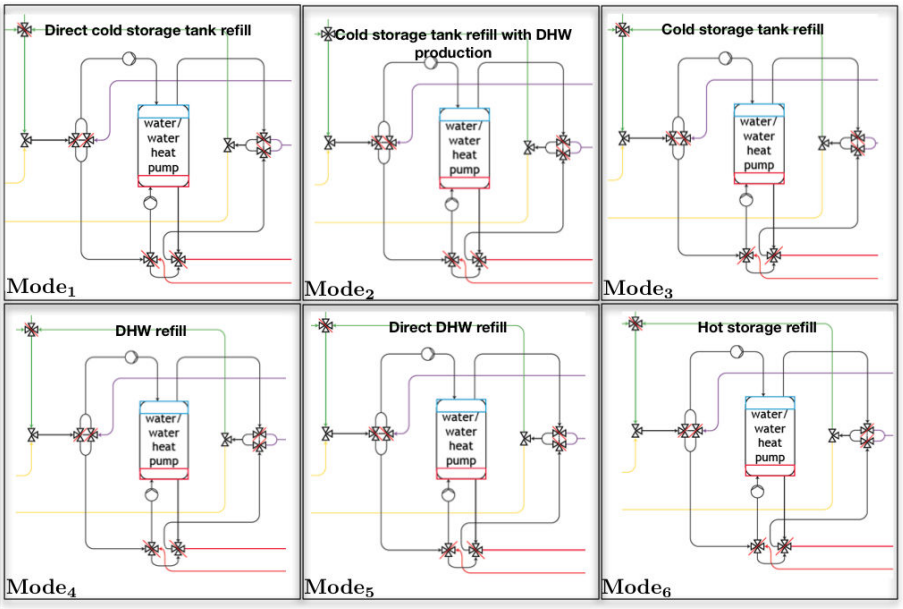



FIGURE *III.20 – Les modes de gestion de la JVP*

Le modèle de la pompe à chaleur est schématisé par la figure III.21.

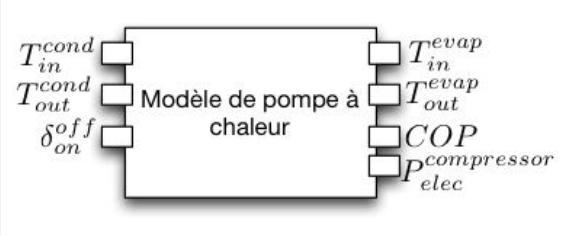



FIGURE *III.21 – Modèle du composant pompe à chaleur JVC*

$T_{in}^{hot}$ : température du fluide froid injecté
$T_{out}^{hot}$ : température du fluide chauffé rejeté
$T_{in}^{cold}$ : température du fluide chaud injecté
$T_{out}^{cold}$ : température du fluide refroidi rejeté
$COP$ : efficacité de la pompe à chaleur.
$\delta$ : commande du compresseur (le système)

Le modèle de la JVP reprend ces variables, lesquelles sont associées au reste des variables du modèle de la JVP. La modélisation du circuit de vannes consiste à reconfigurer les variables (débits et températures) aux différents conduits du circuit.

En raison des débits qui peuvent être différents en entrées dans certaines configurations, les températures décrites ci-dessous sont pondérées par un coefficient $\alpha_i$ qui représente le ratio entre le débit réel du réseau et le débit nominal utilisé dans le modèle de la pompe à chaleur $\dot{m}$

$$
\begin{aligned}
T_{in}^{HotSource} =& mode_1\alpha_3 T_{in}^{net3} + mode_2\alpha_3 T_{in}^{net3} + mode_3\alpha_1 T_{in}^{net1} + \dots \\
& \cdots + mode_4\alpha_3 T_{in}^{net3} + mode_5\alpha_1 T_{in}^{net1} + mode_6\alpha_2 T_{in}^{net2} \\
T_{out}^{HotSource} =& mode_1\alpha_3 T_{out}^{net3} + mode_2\alpha_3 T_{out}^{net3} + mode_3\alpha_1 T_{out}^{net1} \dots \\
& \cdots + mode_4\alpha_3 T_{out}^{net3} + mode_5\alpha_1 T_{out}^{net1} + mode_6\alpha_2 T_{out}^{net2} \\
T_{in}^{ColdSource} =& mode_1\alpha_1 T_{in}^{net1} + mode_2\alpha_2 T_{in}^{net2} + mode_3\alpha_2 T_{in}^{net2} \dots \\
& \cdots + mode_4\alpha_1 T_{in}^{net1} + mode_5\alpha_2 T_{in}^{net2} + mode_6\alpha_1 T_{in}^{net1} \\
T_{out}^{ColdSource} =& mode_1\alpha_1 T_{out}^{net2} + mode_2\alpha_2 T_{out}^{net2} + mode_3\alpha_2 T_{out}^{net2} \dots \\
& \cdots + mode_4\alpha_1 T_{out}^{net1} + mode_5\alpha_2 T_{out}^{net2} + mode_6\alpha_1 T_{out}^{net1}
\end{aligned}
$$

Le coefficient de performance COP dépend du régime d'utilisation de la pompe à chaleur. Le COP suit une abaque caractéristique non-linéaire très complexe à modéliser. Nous avons choisi une série de valeurs de COP en fonction de l'utilisation de la machine. Chaque utilisation est réalisée autour d'un point de fonctionnement, ce qui permet l'estimation d'un COP fixe avec une erreur limitée.

$$COP = mode_1 * COP_1 + mode_2 * COP_2 + mode_3 * COP_3 + mode_4 * COP_4 + mode_5 * COP_5 + mode_6 * COP_6 \quad \text{(III.30)}$$

L'équation ci-dessous permet le choix d'un seul mode lorsque la variable correspondante est mise à 1 et les autres à 0.

$$mode_1 + mode_2 + mode_3 + mode_4 + mode_5 + mode_6 = 1 \quad \text{(III.31)}$$

Une série d'équations est nécessaire pour modéliser les fermetures de circuits qui ne traversent pas la pompe à chaleur :

$$mode_1 T_{in}^{net2} = mode_1 T_{out}^{net1} \quad \text{(III.32)}$$
$$mode_2 T_{in}^{net1} = mode_2 T_{out}^{net1} \quad \text{(III.33)}$$
$$mode_3 T_{in}^{net3} = mode_3 T_{out}^{net3} \quad \text{(III.34)}$$
$$mode_4 T_{in}^{net2} = mode_4 T_{out}^{net2} \quad \text{(III.35)}$$
$$mode_5 T_{in}^{net3} = mode_5 T_{out}^{net3} \quad \text{(III.36)}$$
$$mode_6 T_{in}^{net3} = mode_6 T_{out}^{net3} \quad \text{(III.37)}$$

$$[mode_1 + mode_2 + mode_3 + mode_4 + mode_5 + mode_6] T_{in}^{net4} = [mode_1 + mode_2 + mode_3 + mode_4 + mode_5 + mode_6] T_{out}^{net4} \quad \text{(III.38)}$$

Le $mode_1$ est un mode qui fait intervenir les trois sources utilisées par la JVP. Il fait traverser le liquide froid venant du panneau hybride à travers le coté froid de la PAC puis l'envoie vers le stockage thermique où il décharge le froid avant de ressortir vers le panneau hybride. Le coté chaud de la PAC est branché au ballon d'ECS. Pour que ce branchement soit possible physiquement, il faut préalablement s'assurer que les débits sur les deux réseaux mis en série soient les mêmes.

En modélisation PLNE, une partie de la modélisation est consacrée à l'expression des liens physiques entre variables et une autre partie est dédiée à la limitation des variables dans un domaine acceptable. Cette seconde partie est évidente dans un langage de calcul séquentiel ou une seule variable est manipulée à la fois alors que l'ensemble des variables restantes est figé. En PLNE, l'ensemble des variables évolue dynamiquement et parallèlement. Il est donc nécessaire de préciser l'état de chaque variable en toutes circonstances à travers les équations III.32,...,III.37 pour le modèle de la pompe à chaleur JVP. Chaque équation est activée dans le mode correspondant grâce à la variable binaire associée. Cet ensemble d'équations permet le calcul des températures dans les circuits de fluides caloporteurs non directement connectés à la pompe à chaleur.

## III.5.2 Modèle du système électrique

Le système électrique mis en place dans Canopea est composé de deux sources d'alimentation : les panneaux photovoltaïques et un raccordement à un réseau externe "smart-grid". Le réseau est qualifié de smart-grid car un câble data (RG-45) accompagne le câble acheminant l'énergie. La consommation en temps réel de l'habitat est remontée à une centrale de gestion. De même que l'injection d'énergie dans le réseau est contrôlée par cette centrale. Enfin, des informations sur la qualité de l'électricité sont transmises à l'habitat en temps réel. Une batterie de stockage électrique est aussi présente. Le système de distribution local est constitué d'un ensemble de prises commandables remontant des mesures, d'un système d'éclairage ainsi que de l'alimentation des équipements techniques. Un convertisseur réversible DC/AC commandé par un BMS (Batterie Management System) est installé entre le réseau et la batterie afin de gérer les prélèvements sur le réseau

local. Deux Onduleurs (SunEzy [5]) munis d'un tracker MPPT (Maximum power point tracker) ont été installés pour connecter l'installation PV au système électrique.

Le système étant relativement simple, le découpage en éléments de modélisation a été moins précis que pour le système thermique. Par exemple, l'onduleur et les panneaux PV sont intégrés dans le modèle PV. La batterie est associée au BMS et au convertisseur.

Les modèles des composants électriques utilisés pour la gestion n'ont pas pour vocation de représenter le fonctionnement du système dans le détail comme chercherait à le faire un concepteur en phase de dimensionnement. Ils ont pour objectif de déduire des stratégies de gestion énergétique pertinentes. Un bilan énergétique et une cartographie de l'ensemble des degrés de libertés sur l'ensemble du système suffit.

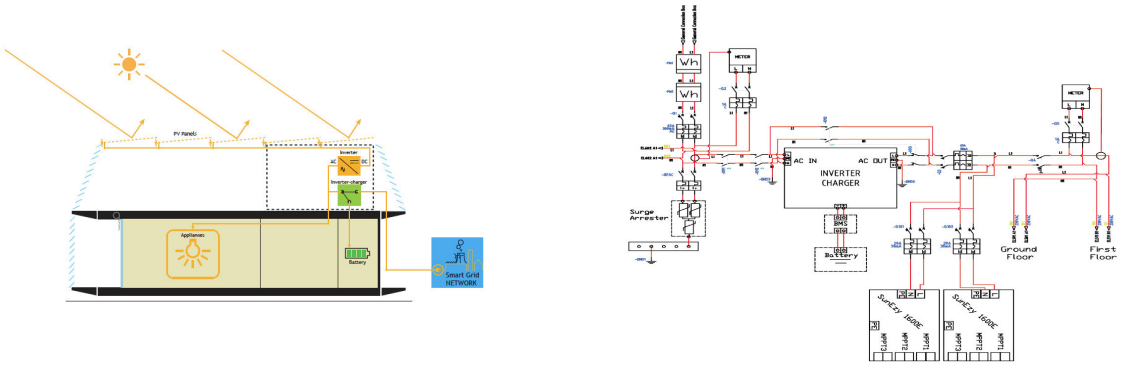



FIGURE *III.22 – Schéma du système électrique*

### III.5.2.1 Modèle de batterie

Le modèle de la batterie est repris de la littérature (DARGAHI 2014) Il s'agit de décrire les mécanismes de commande et le bilan énergétique impliquant la batterie afin que celui-ci soit intégré dans le bilan électrique global. La puissance et l'énergie sont utilisées dans les bilans du fait de la relation liant la puissance à l'énergie consommée durant le temps de consommation.

Deux variables de type binaire sont intégrées au modèle pour représenter l'état de la batterie : $\delta_{charge}(k)$ , $\delta_{discharge}(k)$ selon l'équation :

$$\delta_{charge}(k) + \delta_{discharge}(k) \leq 1 \tag{III.39}$$

L'état de charge de la batterie est exprimé par les trois équations suivantes :

$$\forall k \in \{0, \ldots, n-2\}\,,\ SOC(k+1) = SOC(k) + P_{absorbed}(k)T_e \tag{III.40}$$

$$SOC(0) = SOC(n-1) + P_{absorbed}(n-1)T_e \tag{III.41}$$

$$SOC(0) = SOC_{init} \tag{III.42}$$

Le bilan des flux transitants dans la batterie est exprimée par :

$$\forall k \in \{0, \ldots, n-1\}\,,\ P_{absorbed}(k) = \rho P_{in}(k) - P_{out}(k) \tag{III.43}$$

La commande de la batterie consiste à configurer le sens de circulation de l'énergie : charge, décharge ou arrêt. Une fois la commande établie, la puissance transitant peut varier entre 0 et $P_{max}$.

$$P_{in} \leq \delta_{charge}(k) P_{max} \tag{III.44}$$

$$P_{in}(k) \geq 0 \tag{III.45}$$

$$P_{out} \leq \delta_{discharge}(k) P_{max} \tag{III.46}$$

$$P_{out}(k) \geq 0 \tag{III.47}$$

avec :

5. Onduleur commercialisé par Schneider Electric

— $n$ : nombre de périodes.
— $SOC$ : taux de charge de la batterie.
— $\delta_{charge}(k)$ : commande de charge de la batterie.
— $\delta_{discharge}(k)$ : commande de décharge de la batterie.
— $\rho$ : efficacité de la batterie.
— $P_{absorbed}(k)$ : puissance transmise à l'habitat.
— $P_{in}(k)$ : puissance entrante dans la batterie.
— $P_{out}(k)$ : puissance sortante de la batterie.

### III.5.2.2 Modèle de l'installation photovoltaïque

Un panneau photovoltaïque est composé d'une matrice de cellules photovoltaïques produisant un courant électrique continu. Ce courant passe ensuite vers la batterie à travers le Battery Management System (BMS) ou encore à travers un convertisseur DC/AC comme montré dans le schéma III.22.

La modélisation d'un panneau PV pour la gestion doit représenter la capacité de production électrique avec des paramètres liés aux panneaux.

$$P_{PV} = S_{PV} * \delta * (\rho_{direct} * G_{direct} + \rho_{diffus} * G_{diffus}) \tag{III.48}$$

avec :
— $P_{PV}$ : puissance fournie par le panneau photovoltaïque.
— $S_{PV}$ : surface du panneau photovoltaïque.
— $\rho$ : rendement du panneau PV.
— $\delta$ : commande de déclenchement des panneaux PV (à travers le SunEzy dans le prototype). Cette commande joue le rôle de disjoncteur commandable qui permet l'isolation des panneaux en cas de besoin.
— $G$ : apport solaire direct et diffus sur le panneau PV.

### III.5.2.3 Modèle du réseau électrique "smart-grid" :

Le modèle de réseau développé pour la gestion énergétique CANOPEA correspond à la politique imposée par la compétition. Celle-ci est bâtie sur trois objectifs :

**Autonomie électrique :** La production annuelle doit être supérieure ou égale à la consommation annuelle de la maison. Une métrique a été mise en place afin de quantifier le degré d'autonomie de chaque prototype de la compétition. Elle consiste en l'attribution de points en fonction du différentiel entre énergie produite et énergie consommée (III.49). A la base, ce différentiel doit être annuel mais pour la compétition, ce différentiel a été ramené à une semaine. Ainsi, nous ramenons la règle à un cycle d'optimisation en adaptant l'amplitude du glissement autorisé.

$$min \leq E_{prod} - E_{consum} \leq max \tag{III.49}$$

avec :
— $min$ : prélèvement du réseau maximum autorisé.
— $E_{prod}$ : énergie produite ou déstockée.
— $E_{consum}$ : énergie consommée ou stockée.
— $max$ : injection sur le réseau maximum autorisée.
Les points sont attribués en fonction de la courbe III.23

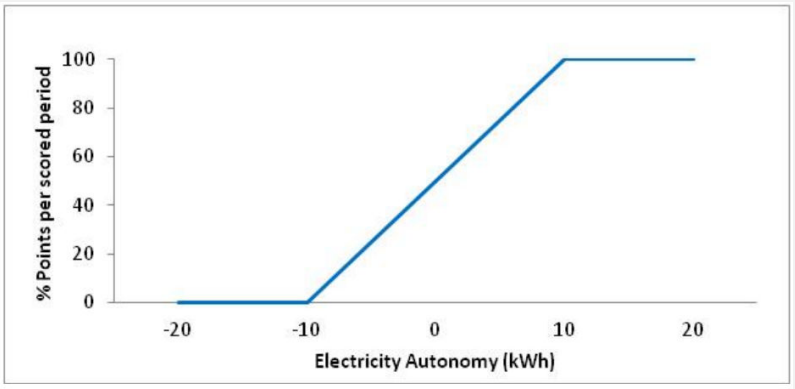



FIGURE *III.23 – Attribution des points selon l'autonomie*

**Correlation temporelle :** Cette règle a été instaurée afin de différencier les équipes par rapport à leur contrainte instantanée sur le réseau. Le modèle est basé sur l'équation III.50.

$$\xi = \frac{E_{G-L} + E_{bat-L}}{E_L} \tag{III.50}$$

$$SC = SC_{max} * \xi \tag{III.51}$$

avec :
- $E_{G-L}$ : Production instantanément consommée
- $E_{bat-L}$ : Energie provenant de la batterie consommée
- $E_L$ : Energie totale consommée
- $SC$ : Score obtenu
- $SC_{max}$ : Score maximum

Cette notation montre qu'il n'y a aucun bénéfice à avoir une production instantanée supérieure à la consommation instantanée dans l'estimation de la corrélation.

**Sobriété énergétique :** Les points sur la sobriété énergétique sont répartis entre l'ensemble des prototypes de bâtiments en compétition d'une manière relative n'ayant pas de limite inférieure. Le prototype le plus sobre doit ainsi obtenir le plus de points. L'ensemble des autres prototypes obtiendra une note au prorata de la meilleure notation. La sobriété énergétique est la consommation ramenée à la surface de chaque habitat.

$$E_{LS} = \frac{E_{L-average}}{S} \tag{III.52}$$

avec :
- $E_{LS}$ : énergie surfacique consommée.
- $E_{L-average}$ : énergie totale consommée.
- $S$ : surface totale habitable.

Le score obtenu sur cette épreuve est alors calculé en fonction de la courbe III.24.

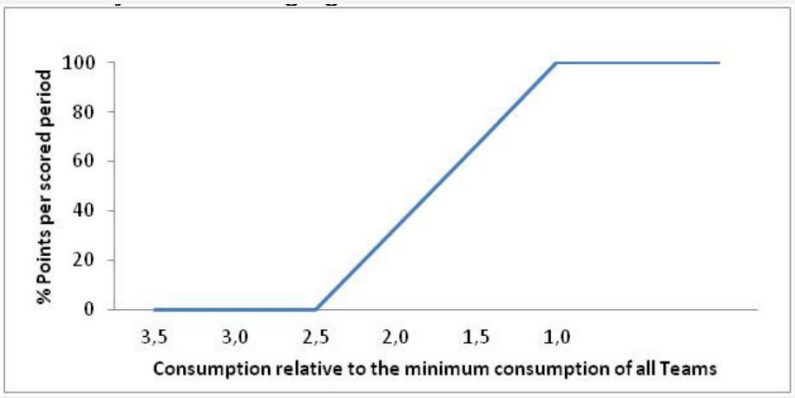



FIGURE *III.24 – Attribution des notes selon la sobriété énergétique*

**document-reglement**

### III.5.2.4 Charge électrique modulable :

Le modèle de la charge électrique modulable traduit le besoin de déplacer dans le temps des charges pouvant l'être. Le modèle est une expression de la relation entre le temps et l'énergie consommée. Le modèle fait appel à plusieurs schémas de linéarisation. Nous avons besoin d'une équation qui mette en lien l'énergie consommée à chaque pas de temps et les données ajustables du moment de lancement de la charge et donc la fin de celle-ci sachant que la durée de la charge ne peut être contractée.

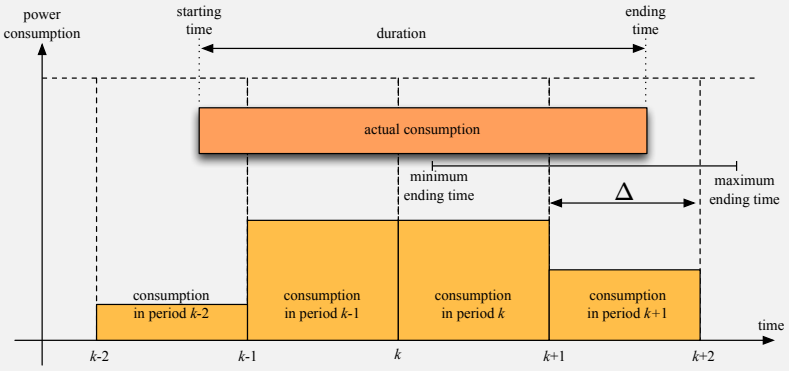



FIGURE *III.25 – Modèle temporel d'une charge électrique décalable*

L'équation bilan sur le slot de temps $k$ pose le lien entre la puissance nominale, la variable d'état de consommation $\delta_3$ ainsi que la durée de fonctionnement sur le slot de temps $k$.

$$E(k) = P\delta_3(k)d(k) \tag{III.53}$$

avec :
- $E(k)$ : Energie utilisée par l'équipement à l'instant $k$
- $P$ : Puissance nominale de l'équipement géré (donnée de l'équipement)
- $\delta_3(k)$ : Variable binaire précisant l'état de fonctionnement de l'équipement à l'instant $k$.
- $d(k)$ : Durée de fonctionnement de l'équipement dans l'intervalle $k$.

Le calcul de la durée de fonctionnement sur le pas de temps $k$ est donné par l'équation :

$$d(k) = min(f, (k+1)\Delta) - max(f - D, k\Delta) \tag{III.54}$$
$$\delta_1(k) = 1 \leftrightarrow f \leq k\Delta \tag{III.55}$$
$$\delta_2(k) = 1 \leftrightarrow f - D \leq k\Delta \tag{III.56}$$

avec :
- $\Delta$ : largeur du pas de temps d'échantillonage.
- $D$ : La durée totale de la tâche (en unité continue).

Le calcul du $min$ et du $max$ fait intervenir respectivement $\delta_1$ et $\delta_2$ :

$$min(f, (k+1)\Delta) = (k+1)\Delta - (k+1)\Delta\delta_1(k+1) + \delta_1(k+1)f \tag{III.57}$$
$$max(f - D, k\Delta) = -D + D\delta_2(k) + k\Delta\delta_2(k) + f - \delta_2(k)f \tag{III.58}$$

ce qui donne une expression finale pour $d(k)$ :

$$d(k) = D + (k+1)\Delta - (k+1)\Delta\delta_1(k+1) - D\delta_2(k) - k\Delta\delta_2(k) - f + \delta_1(k+1)f + \delta_2(k)f \tag{III.59}$$

$\delta_3(k)$ dépend de la durée de fonctionnement de l'équipement sur la période $d(k)$. Si celle-ci est strictement négative (possible par le calcul), cela se traduit par l'arrêt de l'équipement durant l'intervalle $k$ alors $\delta_3(k) = 0$.

L'équation qui décrit cette logique est :

$$\delta_3(k) = 1 \leftrightarrow d(k) \geq 0 \tag{III.60}$$

#### III.5.2.5 Modèle de charge électrique subie

Le modèle de la charge électrique subie est un modèle qui représente l'ensemble des charges électriques qui ne peuvent pas être modulées. Le modèle servira uniquement à intégrer ces charges dans le bilan global et ainsi avoir une balance énergétique plus réaliste.

$$E(k) = \sum E_i(k) \tag{III.61}$$

avec :
- $E(k)$ : la charge totale subie.
- $E_i(k)$ : les charges partielles tel que la plaque chauffante, le micro-onde, le réfrigérateur ainsi que les charges résiduelles des éléments en veille et autres équipement en fonctionnement permanent.

### III.5.3 Modèle d'enveloppe (parois)

Le modèle de l'enveloppe représente l'ensemble des échanges thermiques entre l'intérieur de l'habitat et l'extérieur. Il représente aussi l'effet des apports énergétiques volontaires (chauffage ou climatisation) ainsi que les apports tel que le soleil. La structure de l'habitat est complexe à modéliser (figure III.26). Un modèle fin tel que ceux utilisés en simulation est impossible à intégrer dans le BEMS Vesta-Energy à cause du nombre de variables engendré dans le problème d'optimisation et des non-linéarités présentes. Un modèle linéaire de premier ordre a été proposé en agrégeant l'ensemble des complexités liées au modèle thermique de l'enveloppe, pour un pas horaire. Ce modèle présente l'avantage de rendre possible le recalage paramétrique ("Determination of relevant model structures for self-learning energy management system"), ce que ne permettrait pas un modèle très précis comportant un nombre considérable de paramètres. Le pas d'échantillonnage d'une heure permet d'effacer les dynamiques inférieures à l'heure. Des modèles prenant en compte les dynamiques rapides seront utilisés dans la couche réactive (MAHENDRA et al. 2015). Une méthodologie d'estimation des paramètres du modèle en deux étapes a été utilisée :

**calcul physique,** effectué à partir des données physiques des matériaux utilisés. Un premier jeu de valeurs est ainsi établi.

**recalage paramétrique,** Cette étape est mise en oeuvre une fois le bâtiment mis en place. Cela consiste en un algorithme d'optimisation utilisant une approche inverse mis en oeuvre afin de recaler les valeurs des paramètres à partir de données capteurs sur certaines températures ("Determination of relevant model structures for self-learning energy management system"). Les valeurs ainsi obtenues sont utilisées comme paramètres pour la gestion.

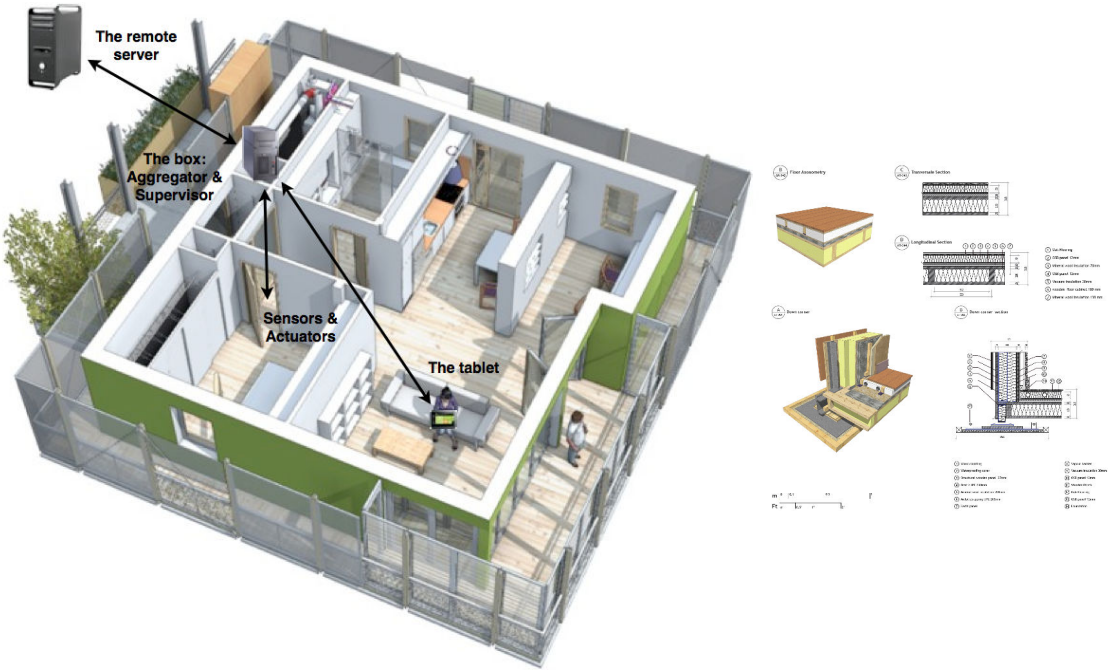



FIGURE *III.26 – Structure et composition de l'enveloppe*

La modélisation de l'enveloppe est basée sur un modèle équivalent électrique dans lequel l'inertie due aux matériaux denses non isolés est représentée par une capacité. Les apports énergétiques solaires et calorifiques sont représentés par les deux générateurs de courant (le courant étant l'équivalent du flux d'énergie), différentes résistances thermiques pour les parois extérieurs. La résistance $R_v$ représente les pertes thermiques liées au renouvellement de l'air : elle dépend donc du débit d'air.

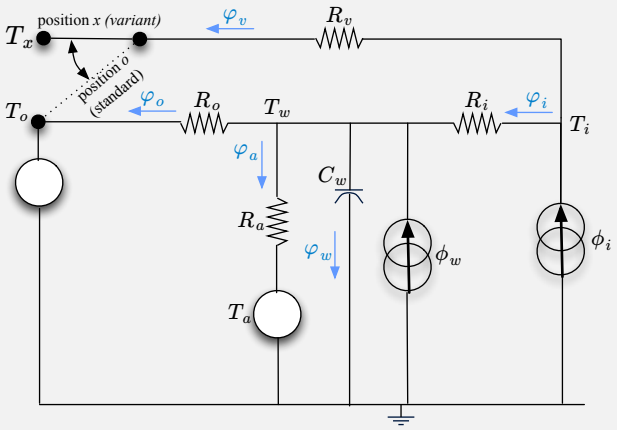



FIGURE *III.27 – Schéma électrique équivalent du modèle d'enveloppe*

Un bilan en lois des noeuds, lois des mailles est établi pour le schéma III.27 comme base de départ pour déduire la représentation d'état du système.

$$T_w - T_o = R_o \varphi_o \tag{III.62}$$
$$T_w - T_a = R_a \varphi_a \tag{III.63}$$
$$\varphi_w = C_w \frac{dT_w}{dt} \tag{III.64}$$
$$\varphi_i + \phi_w = \varphi_o + \varphi_w + \varphi_a \tag{III.65}$$
$$T_i - T_w = R_i \varphi_i \tag{III.66}$$
$$T_i - T_x = R_v \varphi_v \tag{III.67}$$
$$\varphi_i + \varphi_v = \phi_i \tag{III.68}$$
$$\tag{III.69}$$

- $T_o$ : température extérieure
- $T_i$ : température intérieure de la zone thermique
- $T_w$ : température des murs entourant la zone thermique
- $T_a$ : température des zones adjacentes
- $T_x$ : température alternative de l'air admis par ventilation.
- $\phi_i$ : énergie admise dans la zone thermique par chauffage direct
- $\phi_w$ : énergie injectée directement dans les murs
- $\varphi_i$ : énergie absorbée par les murs de la zone thermique
- $\varphi_o$ : énergie évacuée par les mur vers l'extérieur.
- $\varphi_w$ : énergie stockée puis restaurée par les murs
- $\varphi_a$ : énergie échangée avec les zones adjacentes à la zone thermique traitée.
- $\varphi_v$ : énergie échangée directement avec le milieu extérieur.
- $R_o$ , $R_i$ : résistance thermique équivalente
- $R_v$ : résistance thermique (déduite de l'échangeur de chaleur)
- $C_w$ : capacité thermique équivalente des murs
- $R_a$ : résistance thermique équivalente définissant la puissance échangée avec les zones thermiques adjacentes.

Finalement, le système d'équations devient :

$$\frac{d}{dt}[T_w] = \left[-\left(\frac{1}{R_a} + \frac{1}{R_o} + \frac{1}{R_v + R_i}\right)\frac{1}{C_w}\right][T_w] + \begin{bmatrix} \frac{1}{R_o C_w} & \frac{1}{R_a C_w} & \frac{1}{(R_v+R_i)C_w} & \frac{1}{C_w} & \frac{R_v}{(R_v+R_i)C_w} \end{bmatrix} \begin{bmatrix} T_o \\ T_a \\ T_x \\ \phi_w \\ \phi_i \end{bmatrix} \tag{III.70}$$

$$[T_i] = \left[\frac{R_v}{R_v + R_i}\right][T_w] + \begin{bmatrix} 0 & 0 & \frac{R_i}{R_v+R_i} & 0 & \frac{R_i R_v}{R_v+R_i} \end{bmatrix} \begin{bmatrix} T_o \\ T_a \\ T_x \\ \phi_w \\ \phi_i \end{bmatrix} \tag{III.71}$$

Le modèle de l'enveloppe n'est pas complètement linéaire à cause de $R_v$. Cette résistance représente l'ensemble des échanges thermiques par transport de l'air. Elle prend en compte l'air injecté par la compact P ainsi que les fuites résiduelles. La valeur $R_v$ est déduite du débit d'air échangé avec l'intérieur. Cette relation induit un caractère variable pour $R_v$. $R_v$ est, par ailleurs, dans la matrice $C$ de la représentation d'état, laquelle est multipliée par le vecteur des variables d'état. Cette représentation est donc non-linéaire et n'est pas intégrable en l'état dans un problème PLNE. La solution proposée à cette non-linéarité est la discrétisation de la variable $R_v$ via une discrétisation du débit d'air. Cette solution est ensuite implémentée à travers plusieurs représentations d'état commutées, fonction du nombre de discrétisations du débit d'air. Chaque valeur de débit est accompagnée d'une variable binaire d'activation $\zeta_i$. Chaque instance est associée à une représentation d'état et fait donc de $R_{v_i}$ un paramètre dans cette représentation. Le choix de la représentation d'état à considérer dans l'optimisation est fait selon $\zeta_i$. Si cette variable binaire est égale à 1, les autres $\zeta_j; j \neq i$ étant obligatoirement à zéro.

La discrétisation doit être impactée sur l'ensemble des autres modèles auxquels est associée la variable $R_v$. Cette démarche peut être source d'erreur si elle est accomplie

manuellement, d'où le process de branchement mis en place dans Milp-workshop.

### III.5.4 Modèle d'occultants

Le modèle d'occultants intègre la commande mécanisée des volets roulants de l'enveloppe interne de Canopea. La variable $G_L$ est calculée dans un modèle distinct en fonction de l'exposition, de l'orientation, de la surface ainsi que de la position du soleil et du masque présent (nuages, chapeaux de toit).

$$sleepingTime(k) <= closed(k) \tag{III.72}$$

$$light(k) = opened(k) * G_L(k) + 0.5 * half(k) * G_L(k) \tag{III.73}$$

$$heat(k) = opened(k) * G_T(k) + 0.5 * half(k) * G_T(k) \tag{III.74}$$

$$closed(k) + half(k) + opened(k) = 1 \tag{III.75}$$

### III.5.5 Modèle d'irradiation solaire

Un modèle d'irradiation solaire a été proposé pour le modèle global et présenté ci-dessous. Le modèle d'irradiation intervient de manière déterministe et non probabiliste de l'anticipation. Les fournisseurs de données météorologiques fournissent des informations approximatives concernant l'emplacement géographique. Le modèle a pour objectif de fournir des informations plus précises à travers à une reconstruction partielle du modèle d'irradiation solaire grâce aux coordonnées géographiques à une date et une heure précises moyennant une estimation de la nébulosité récupérable dans les données de prévisions météorologiques.

$$R_{direct} = \text{Coefficientdetransmissivité} * F_{atm} * e^{-\text{MassedAiroptiquerelative} * \text{Rayleigh}_{ep} * \text{FacteurtroubledeLinke}} \tag{III.76}$$

$$\text{Coefficientdetransmissivité} = (0.6)^{(\sqrt{1229+(614*sin(\text{Altitude}))^2}-614*sin(\text{Altitude}))*((288-0.0065*h)/288)^{5.256}} \tag{III.77}$$

$$F_{atm} = 1367 * (1 + 0.033 * cos(\frac{360 * \text{JourDeLannée}}{365})) \tag{III.78}$$

$$\text{MassedAiroptiquerelative} = \frac{P_{atm}}{101325 * sin(\text{Altitude}) + 15198.75 * (3.885 + \text{Altitude})^{-1.253}} \tag{III.79}$$

$$P_{atm} = 101325 * (1 - 2.26 * 10^{-5} * h)^{5.26} \tag{III.80}$$

$$\text{Rayleigh}_{ep} = \frac{1}{0.9 * \text{MassedAiroptiquerelative} + 9.4} \tag{III.81}$$

$$\text{FacteurdetroubledeLinke} = 2.4 + 14.6 * B + 0.4 * (1 + 2 * B) * ln(P_v) \tag{III.82}$$

$$P_v = 2.165 * (1.098 + T/100)^{8.02} * H_r \tag{III.83}$$

$$B = \begin{cases} 0.002 & \text{pour un lieu situé en montagne} \\ 0.05 & \text{pour un lieu rural} \\ 0.10 & \text{pour un lieu urbain} \\ 0.20 & \text{pour un lieu industriel (atmosphère polluée)} \end{cases} \tag{III.84}$$

$$AngleIncidence = cos^{-1}(cos(Altitude)*sin(Inclinaison)*cos(Azimut-Exposition)+sin(Altitude)*cos(Inclinaison) \quad \text{(III.85)}$$

$$R_{diffus} = Constantesolaire * (0.271 - 0.294 * \tau^{M}) * sin(Altitude) \quad \text{(III.86)}$$

$$R_{reflechi} = 0.2 * 1367 * (0.271 + 0.706 * \tau^{M}) * sin(Altitude) * sin(\frac{Inclinaison}{2})^2 \quad \text{(III.87)}$$

Le modèle est constitué d'un ensemble de paramètres utiles à l'optimisation. Les résultats de ce modèle, $R_{direct}$, $R_{difus}$ et $R_{reflechi}$, sont des valeurs indépendantes du reste du problème et peuvent être calculées a posteriori. Le calcul se fait ainsi avant tout lancement de l'optimisation à travers le processus de pré-conditionnement de MILP-workshop. Le résultat du calcul est pris en compte comme paramètre dans le processus d'optimisation pour la génération des plans d'usage anticipés.

### III.5.6 Modèle d'évolution du taux de CO2 dans l'air

Le taux de CO2 dans l'air est une donnée mesurable en temps réel. Néanmoins, un modèle est nécessaire afin d'anticiper le taux et préparer le plan d'usage anticipé en fonction de l'ensemble des paramètres.

Le calcul du taux de CO2 fait intervenir les éléments contribuant à sa modification, à savoir les occupants et la ventilation (équation VI.26).

$$\frac{dC(t)}{dt} = \left(\frac{Q_i(t)C_i}{V} + \frac{S(t)}{V}\right) - \frac{Q_o(t)}{V}C(t) \quad \text{(III.88)}$$

avec :

- $Q_i$ débit volumétrique d'air frais entrant
- $Q_o$ débit volumétrique d'air rejeté
- $V$ volume de l'espace
- $C(t)$ concentration intérieure du $CO_2$ à l'instant $t$
- $C_i$ concentration extérieure de $CO_2$
- $S$ taux de génération de $CO_2$, masse par unité de temps à partir de l'occupation ou d'autres sources.

### III.5.7 Modèle de luminosité

Le modèle de luminosité fournit des informations à propos de l'apport lumineux dans l'habitat. Cet apport est naturel, le rayonnement solaire à travers les fenêtres, ou artificiel via l'éclairage.

$$G_L(k) = \frac{\text{éclairage naturel en lumen(k)}}{\text{Surface habitat}} = 93 * 0.4 * R_{tot}(k) \quad \text{(III.89)}$$

$$R_{tot}(k) = (R_{direct}(k) * A_{cos} + R_{difus}(k) + R_{\text{réfléchi}}(k)) * S_{\text{windows}} * K_{dim} \quad \text{(III.90)}$$

avec :

- $G_L$ : apport lumineux
- $R_{tot}$ : puissance solaire rayonne transmise
- $S_{windows}$ : surface des fenêtres éclairées
- 93 : quantité de lumière émise par le soleil en [lumen/W]
- $0,4$ : taux de lumière dans le spectre du visible (le spectre convertible en [lux])
- $K_{dim}$ : coefficient d'atténuation
- $A_{cos}$ : cosinus de l'angle d'incidence des rayons solaires sur les fenêtres.

$$S_{windows} = S^{max}_{windows} * K_{expo} \tag{III.91}$$

avec : $K_{expo}$ : ratio du vitrage exposé au rayonnement direct.

$$G_{artificiel}(k) = nominalLux * state(k) \tag{III.92}$$
$$P_L(k) = state(k) * nominalPowerInWatt \tag{III.93}$$

avec :
- $G_{artificiel}$ : contribution artificielle à l'éclairage
- $P_L(k)$ : puissance utilisée pour l'éclairage.

## III.5.8 Modèle de confort

### III.5.8.1 Confort thermique

Le confort thermique a été calculé sur la base du cahier des charges imposé par la réglementation de la compétition. Le cahier des charges distingue deux situations : présence et absence, l'exigence thermique n'est alors plus la même. En cas d'absence seule une température minimale est requise afin d'éviter le phénomène de condensation et ainsi avoir un milieu propice au développement de champignons et bactéries sur les parois et dans les couches d'isolants. En cas de présence, une température idéale est alors préconisée. Celle-ci est fixée à 21 degrés celsius dans le cadre de la compétition. Le confort est ensuite considéré comme se dégradant graduellement jusqu'à des températures min et max au dessous desquelles il est jugé inacceptable.

**Inoccupation** : $\pi(k) = 0$ La formulation de l'inconfort est la suivante :

$$\begin{cases} if & (T^{felt}(k) \geq T_{uno}) : \\ & \varsigma(k) = 0 \\ else & : \\ & \varsigma(k) = 1 \end{cases} \tag{III.94}$$

**Occupation** : $\pi(k) = 1$

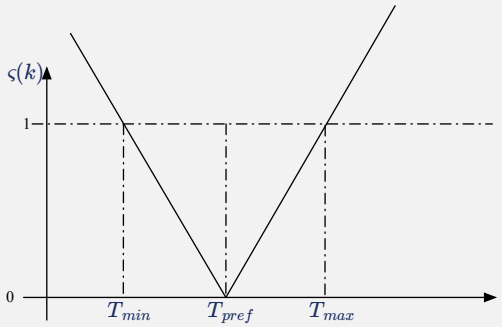



FIGURE *III.28 – courbe d'insatisfaction thermique*

La formule de l'inconfort thermique suit alors la courbe III.28 avec la variable $\varsigma(T^{felt}(k))$

$$switch(cases) \begin{cases} case : & T_{min} \leq T^{felt}(k) \leq T_{pref} \\ & \varsigma(k) = \frac{T_{pref} - T^{felt}(k)}{T_{pref} - T_{min}} \\ case : & T_{pref} \leq T^{felt}(k) \leq T_{max} \\ & \varsigma(k) = \frac{T^{felt}(k) - T_{pref}}{T_{max} - T_{pref}} \\ else : & \\ & \varsigma = 10000 \end{cases} \tag{III.95}$$

### III.5.8.2 Qualité de l'air

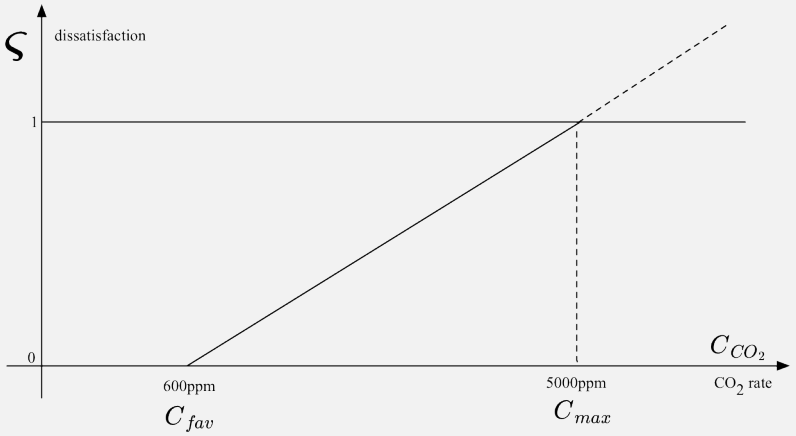



FIGURE *III.29 – courbe de confort CO2*

$$\varsigma(k) = \frac{C_{CO_2}(k) - C_{fav}}{C_{max} - C_{fav}} \tag{III.96}$$

$$\varsigma(k) \leq 1 \tag{III.97}$$

$$C_{CO_2} \leq C_{max} \tag{III.98}$$

$$d(k) = 1 - \delta_{presence}(k) + \delta_{presence}(k)\varsigma(k) \tag{III.99}$$

avec $d(k)$ : insatisfaction liée au taux de CO2 en prenant en compte la présence.

### III.5.8.3 Eclairage

Le modèle de satisfaction lumineuse est basé sur deux sous modèles :
— éclairage présence active (éveillée)
— éclairage présence passive (sommeil)
L'insatisfaction est considérée comme nulle lorsqu'il y a absence d'occupation.

Le modèle de satisfaction en cas de présence active est un modèle de type insatisfaction thermique avec des seuils adaptés (valeurs et unités).

Le modèle d'insatisfaction en cas de présence passive correspond à une plage de luminosité autorisée (entre un min et un max) dans laquelle l'insatisfaction est nulle. En dehors des bornes, l'insatisfaction est maximale.

L'équation liant les trois situations est donnée par :

$$\delta(k) = Presence * (sleepingTime(k) * \delta^{-}Sleeping(k) + \delta_{Awake}(k) - sleepingTime(k) * \delta_{Awake}(k)) \tag{III.100}$$

## III.6 Composition

La composition consiste en l'association de plusieurs modèles élémentaires décrivant un système bâtiment, pour nous, en vue de sa gestion énergétique. Un modèle global per-

met de déduire des stratégies globales portant sur l'ensemble des équipements, enveloppe et composants d'enveloppes. Le modèle global ainsi que les exigences de linéarité imposée par le solveur utilisé induisent une complexité considérable : la composition aboutit à un problème PLNE de plus de 12000 variables. Le schéma VI.12 présente la composition du bâtiment CANOPEA.

Les interconnexions entre les modèles élémentaires sont des phénomènes physiques induits par l'association de certains équipements. La principale interconnexion entre équipements sont les flux de chaleur et les flux électriques. Dans un habitat à forte efficacité énergétique, l'apport calorifique des équipements est non négligeable et souvent source de chaleur. Ces sources sont négligées dans les travaux **??**à cause de la complexité induite par leur prise en compte. Les travaux **??**prennent en compte une approximation résiduelle des apports qui n'est pas tout le temps représentative. Selon l'état de l'équipement, l'émission de chaleur n'est pas la même. Il est donc approximatif de considérer une émission résiduelle constante. A l'échelle de l'habitat, le fournisseur met en place la mesure de la puissance active appelée. C'est sur la base de la mesure de cette puissance que le comptage énergétique est effectué et donc facturé. De même qu'à l'échelle des mesures de consommation des équipements c'est le même type d'énergie qui est mis en jeu. L'énergie active correspond à l'énergie dissipée en chaleur volontairement ou à travers les pertes. Il est donc intéressant d'ajouter l'énergie consommée par les équipements dans le bilan des apports en chaleur si ceux-ci se trouvent dans la zone thermique

ajouter ref(s)

ajouter ref(s)

$$P_{elec} = P_{heating} \tag{III.101}$$

Vu la complexité du modèle global, il est inconcevable pour un développeur de modèles de construire un tel modèle directement. MILP-workshop, que nous avons développé à l'origine pour résoudre le problème CANOPEA, a permis de mettre au point son modèle global progressivement. Les compositions, car les modèles sont associés pas à pas pour mieux localiser les problèmes, sont des étapes clefs qui permettent de valider ou non une étape de modélisation : elles se traduisent par un fichier de composition au même format que celui d'un modèle élémentaire. D'un point de vue pratique, rien ne distingue une composition d'un modèle élémentaire : une composition, comme le modèle élémentaire, est vue et utilisée comme un modèle dans lequel des variables sont connectées via des contraintes égalités, des équations peuvent être ajoutées (bilans,..) et des paramètres instanciés. Cette vision permet ainsi de composer à l'infini et monter en complexité autant que nécessaire.

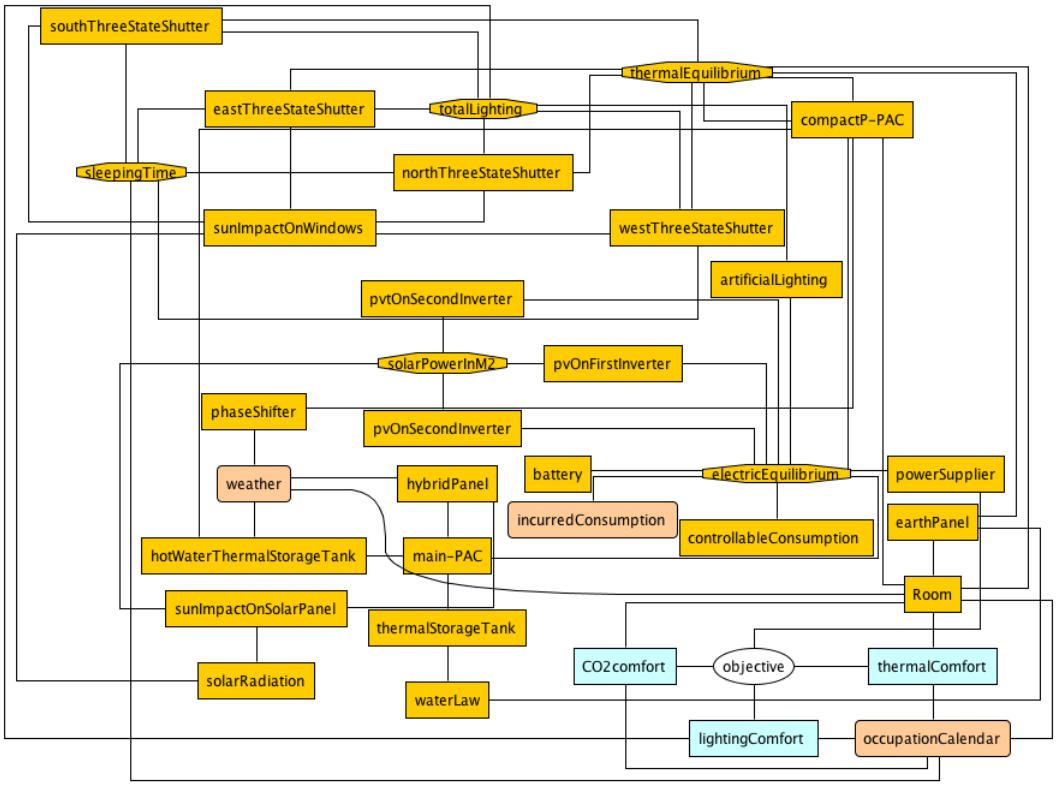



FIGURE *III.30 – Schéma de composition global*

## III.7 Résultats

Nous présentons les résultats des travaux selon plusieurs objectifs intermédiaires en relation avec l'objectif final de la thèse. Rappelons à ce titre que l'objectif final de la thèse est la conception d'une solution de gestion énergétique largement déployable et économiquement viable. Parmi les objectifs intermédiaires que nous nous sommes fixés à ce stade de la thèse : la fourniture d'un outil de gestion de modèles et d'une implémentation de l'ensemble sur un cas concret. Nous avons choisi pour ce cas CANOPEA qui a été présenté tout au long de ce chapitre.

L'objectif dans le cadre de la participation au solar-décathlon 2012 était double. D'un côté, l'objectif était le respect du cahier des charges imposé par l'organisation et la mise en place d'une solution qui puisse répondre aux exigences notées de la compétition de la meilleure manière possible. De l'autre côté, l'objectif était de démontrer la faisabilité de la solution sur un cas concret pour prouver la pertinence de cette solution et relever les problèmes associés, lesquels seront traités en seconde partie.

Le travail de modélisation s'est concentré sur la partie anticipative de la solution GHomeTech. C'est à ce niveau que nous avons estimé l'apport scientifique assez intéressant pour qu'il soit présenté dans cette thèse.

Nous présentons les résultats sous trois aspects :

— L'apport pour la modélisation à travers une synthèse des modèles utilisés et le traitement de leur complexité.
— Les résultats d'optimisation du gestionnaire énergétique obtenus grâce aux modèles développés
— Les résultats finaux de la solution de gestion énergétique pour CANOPEA

### III.7.1 Résultats concernant l'aspect modélisation

**Inoccupation** : $\pi(k) = 0$

$$\begin{cases} if & (T^{felt}(k) \geq T_{uno}) : \\ & \varsigma(k) = 0 \\ Else & : \\ & \varsigma(k) = 1 \end{cases} \quad (1)$$

**Occupation** : $\pi(k) = 1$

$$switch(cases) \begin{cases} case : & T_{min} \leq T^{felt}(k) \leq T_{pref} \\ & \varsigma(k) = \frac{T_{pref} - T^{felt}(k)}{T_{pref} - T_{min}} \\ case : & T_{pref} \leq T^{felt}(k) \leq T_{max} \\ & \varsigma(k) = \frac{T^{felt}(k) - T_{pref}}{T_{max} - T_{pref}} \\ else : & \\ & \varsigma = 10000 \end{cases} \quad (2)$$

FIGURE *III.31 – Description physique du modèle de confort thermique*

**if..then..else**

$$\begin{cases} if & T^{felt}(k) \geq T_{uno} : \\ & \varsigma_1(k) = (1 - \pi(k)) * 0 \\ else & \\ & \varsigma_1(k) = (1 - \pi(k)) * 1 \end{cases} \quad (1)$$

**switch (cases)**

$$\begin{cases} case : & T_{min} \leq T^{felt}(k) \leq T_{pref} \\ & \varsigma_2(k) = \frac{T_{pref}}{T_{pref} - T_{min}} * \pi(k) - \frac{1}{T_{pref} - T_{min}} * (\pi(k) * T^{felt}(k)) \\ case : & T_{pref} \leq T^{felt}(k) \leq T_{max} \\ & \varsigma_2(k) = \frac{1}{T_{max} - T_{pref}} * (\pi(k) * T^{felt}(k)) - \frac{T_{pref}}{T_{max} - T_{pref}} * \pi(k) \\ Else & : \varsigma_2 = 10000 \end{cases} \quad (2)$$

$$\varsigma(k) = \varsigma_1(k) + \varsigma_2(k) \quad (3)$$

FIGURE *III.32 – Description algébrique du modèle de confort thermique*

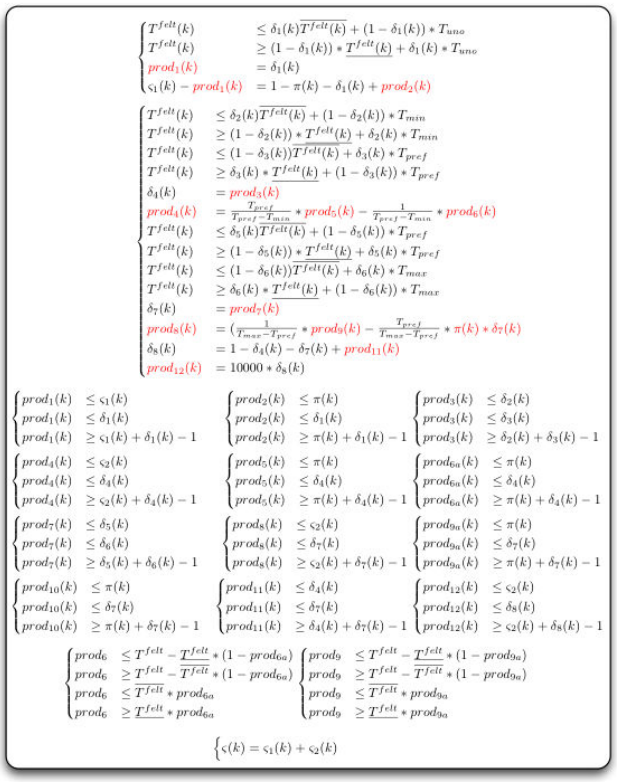


FIGURE *III.33 – Description PLNE du modèle de confort thermique*

Le résultat de l'application Milp-workshop est un espace de développement de modèles dans une bibliothèque dédiée. Ces modèles sont développés grâce à une syntaxe propre à Milp-workshop inspirée d'autres solutions étudiées dans la bibliographie et des besoins propres à l'application optimisation énergétique dans le bâtiment.

Les représentations du modèle de confort thermique (figure III.31), (figures III.32 et III.33) sont un exemple de l'évolution de la représentation depuis un modèle algébrique (représentation mathématique) à une représentation au format PLNE.

Nous avons au départ le modèle (figure III.31), qui est une représentation physique telle que nous la retrouvons dans des documents scientifiques. Le modèle est posé ainsi par le physicien dans un esprit répondant précisément au besoin de la gestion énergétique. Ce modèle est ensuite traduit dans le langage milp-workshop (figure III.32) selon les contraintes liées au parseur. Le modèle physique contient une imbrication de conditions (structure "if..then..else" à l'intérieur d'une autre). Cette imbrication ne peut être prise en compte par le parseur de milp-workshop car ce dernier n'a pas été enrichi de cette représentation que nous avons croisée rarement. L'écriture du modèle est finalement quelque peu modifiée afin d'intégrer cette contrainte d'ordre pratique liée à MILP-workshop sans modifier la teneur physico-algébrique du modèle. Le développement des équations est aussi imposé afin de mettre en évidence les non-linéarités et ainsi les rendre détectables. Le nouveau modèle ainsi obtenu (figure III.32), est considéré comme contraint par l'implémentation du langage de modélisation car le développeur de modèles doit adapter sa description à l'environnement dans lequel il travaille mais ne remet pas en cause l'approche théorique développée.

Le résultat du modèle dans le formalisme admis par MILP-workshop comporte deux structures conditionnelles dans lesquelles les instructions contiennent des non-linéarités

de type binaire-binaire et binaire-continu. Le résultat, après traitement de ces non-linéarités est ensuite projeté temporellement sur l'horizon choisi de 24h, avec un pas de temps de 1h. La figure III.33 représente le problème PLNE correspondant à avant la projection temporelle. Cette représentation est illustrée uniquement afin de simplifier la lecture.

Le problème PLNE projeté sur un horizon de 24h à pas de temps d'une heure avec toutes les déclarations de variables contient 2112 lignes dont 1512 (63x24) contraintes qui représentent l'ensemble des équations et inéquations du modèle. Les variables au nombre de 600 (25x24) sont aussi toutes déclarées. Il est utile de distinguer variables binaires de variables continues. Il est aussi utile de distinguer les variables binaires créées par le développeur de modèle des variables binaires crées automatiquement lors de la linéarisation. Cette dernière distinction fournit une appréciation de la complexité calculatoire ajoutée par la linéarisation ainsi que les autres transformations apportées par Milp-wokshop. Nous synthétisons un bilan en terme de variables pour le modèle de confort thermique dans le tableau III.1.

| Type de modèle | variables binaires avant projection temporelle | variables binaires après projection temporelle | variables continues avant projection temporelle | variables continues après projection temporelle | nombre de paramètres |
|---|---|---|---|---|---|
| Modèle algébro-physique | 1 | 24 | 4 | 96 | 4 |
| Modèle PLNE | (1+8) | 216 | (4+12) | 384 | 4 |

TABLE *III.1 – Comparaison entre forme algébrique et forme PLNE d'un même modèle*

Dans le modèle algébro-physique du confort thermique, il y a une seule variable binaire $\pi$ laquelle est projetée sur les 24 pas de temps d'échantillonnage. Le modèle PLNE, après projection temporelle, contient lui 216 variables binaires. Le modèle algébro-physique contient 4 variables continues tandis que son équivalent PLNE en contient 384.

Un même calcul dans le même ordre de grandeurs peut être effectué sur les contraintes.

Au final, le modèle algébro-physique offre au développeur la possibilité de manipuler 5 variables et 4 paramètres tandis que le même modèle développé en PLNE en aurait nécessité 600. Comme montré tout au long des chapitres de cette thèse, la différence entre les deux formulations réside dans la présentation de mêmes informations. La formulation PLNE est une formulation horizontale dans laquelle toutes les informations sont contenues dans le modèle. La représentation Milp-workshop consiste en la hiérarchisation de l'information. Elle permet ainsi de faire abstraction des développements dans un premier temps pour diminuer la complexité apparente au développeur puis, dans un second temps, développer le modèle PLNE d'une manière automatique selon les principes énoncés dans les patrons.

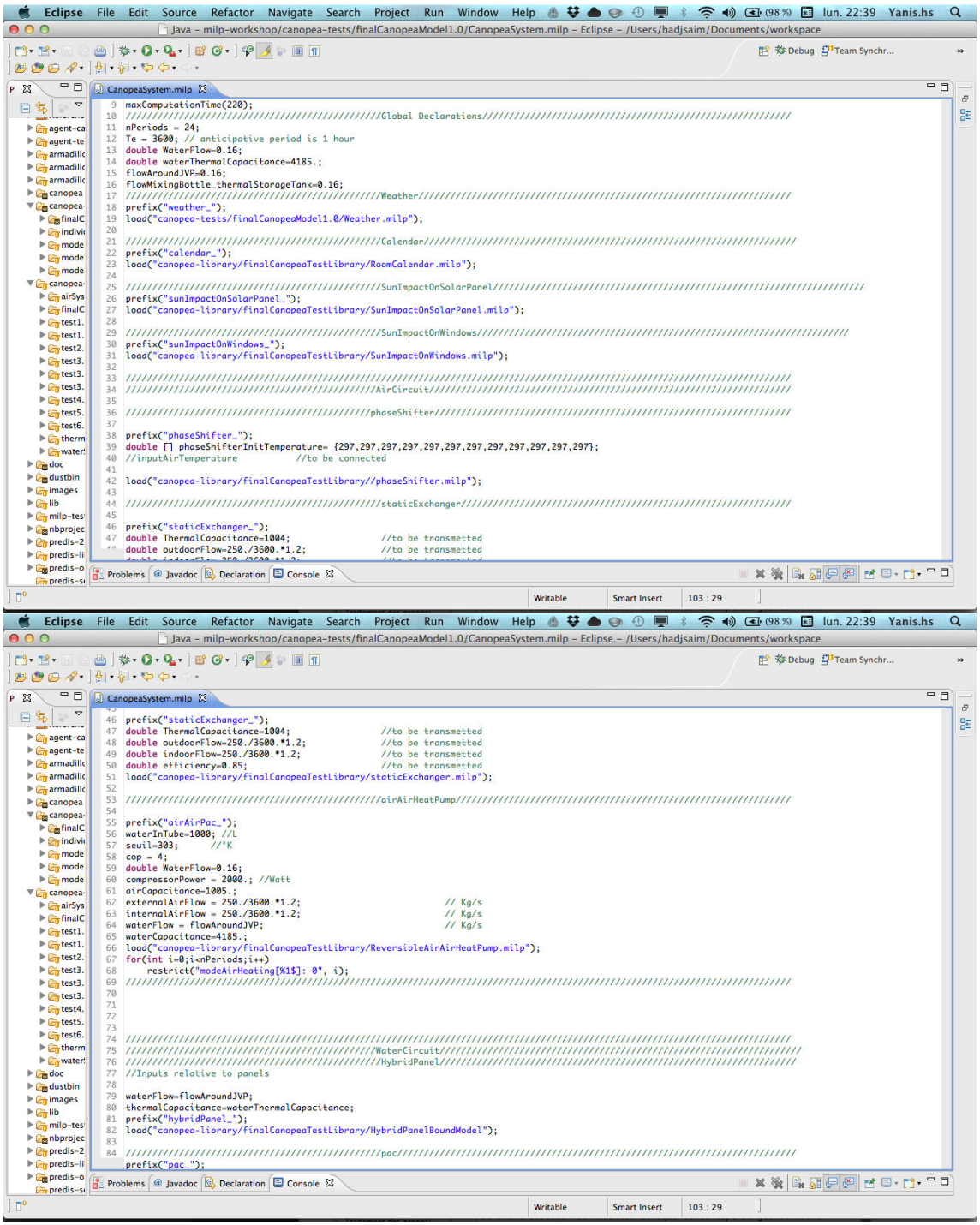
A ce stade, le modèle n'est pas encore un problème d'optimisation car il lui manque l'objectif à optimiser. Le problème PLNE de CANOPEA contient plus de 46.000 lignes dans sa formulation finale PLNE. Le but de la présentation de ce résultat est la mise en évidence de la simplification apportée par MILP-workshop.

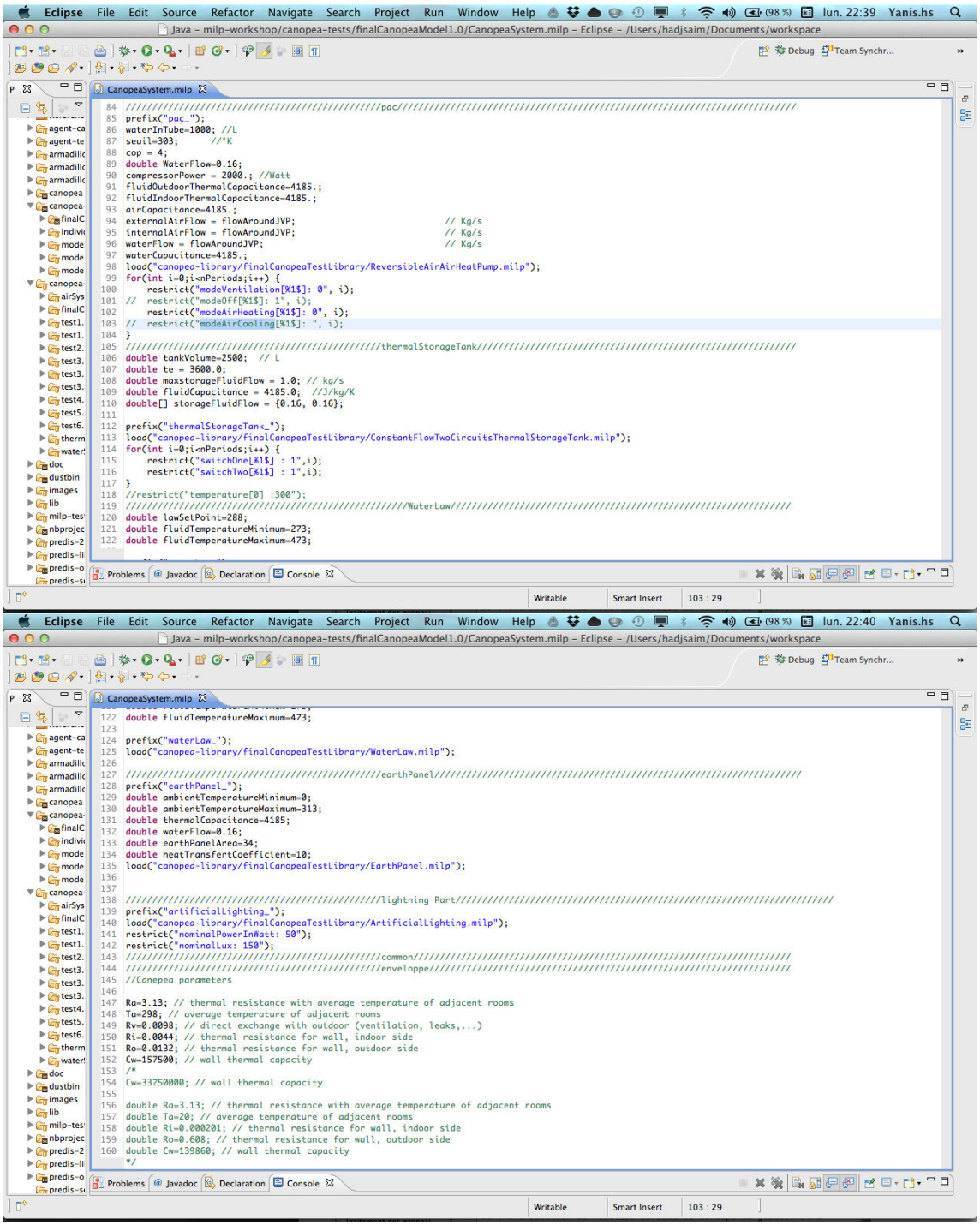
### III.7.2 Résultats concernant l'aspect traitement et résolution de modèles

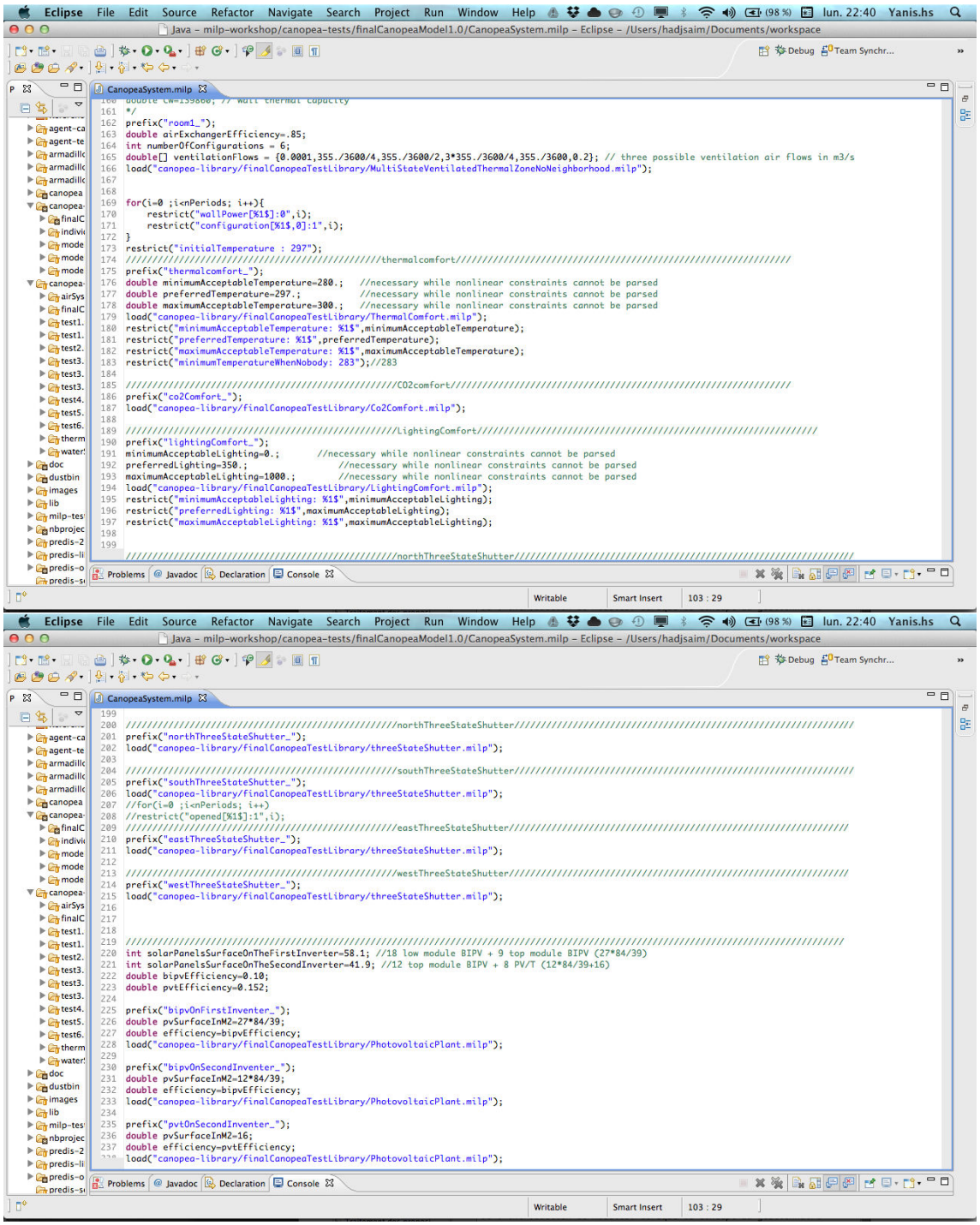
Nous avons choisi, pour l'illustration de nos résultats sur la modélisation de CANOPEA, quelques captures d'écran de l'environnement de développement III.34. Ces captures representent la composition générale constituant ainsi le problème d'optimisation

global. Cette composition intègre aussi les instanciations de paramètres et les valeurs ainsi choisies pour le cas étudié. Nous enchainerons par la suite avec un plan d'usage anticipé. Ce plan correspond à une situation météorologique d'inter-saison qui met en évidence certaines coordinations entre équipements que sont les volets roulants, le système de production d'énergie et les moyens de production de chaleur. C'est ainsi le moyen de démontrer l'intérêt du traitement global du problème.

La composition intègre l'ensemble des éléments et des sous éléments traités dans ce chapitre. Chaque instanciation d'un élément est accompagnée d'un nom propre afin de le distinguer d'autres éléments similaires. C'est le cas des occultants appelés *shutter* dans la composition.




Eclipse File Edit Source Refactor Navigate Search Project Run Window Help lun. 22:39 Yanis.hs
Java – milp-workshop/canopea-tests/finalCanopeaModel1.0/CanopeaSystem.milp – Eclipse – /Users/hadjsaim/Documents/workspace
CanopeaSystem.milp
84 ////////////////////////////////////////////////////pac/////////////////////////////////////////////////////////////////////
85 prefix("pac_");
86 waterInTube=1000; //L
87 seuil=303;  //°K
88 cop = 4;
89 double WaterFlow=0.16;
90 compressorPower = 2000.; //Watt
91 fluidOutdoorThermalCapacitance=4185.;
92 fluidIndoorThermalCapacitance=4185.;
93 airCapacitance=4185.;
94 externalAirFlow = flowAroundJVP;  // Kg/s
95 internalAirFlow = flowAroundJVP;  // Kg/s
96 waterFlow = flowAroundJVP;  // Kg/s
97 waterCapacitance=4185.;
98 load("canopea-library/finalCanopeaTestLibrary/ReversibleAirAirHeatPump.milp");
99 for(int i=0;i<nPeriods;i++) {
100  restrict("modeVentilation[%1$]: 0", i);
101 //  restrict("modeOff[%1$]: 1", i);
102  restrict("modeAirHeating[%1$]: 0", i);
103 //  restrict("modeAirCooling[%1$]: ", i);
104 }
105 ////////////////////////////////////////////////////thermalStorageTank//////////////////////////////////////////////////////////
106 double tankVolume=2500;  // L
107 double te = 3600.0;
108 double maxstorageFluidFlow = 1.0; // kg/s
109 double fluidCapacitance = 4185.0;  //J/kg/K
110 double[] storageFluidFlow = {0.16, 0.16};
111
112 prefix("thermalStorageTank_");
113 load("canopea-library/finalCanopeaTestLibrary/ConstantFlowTwoCircuitsThermalStorageTank.milp");
114 for(int i=0;i<nPeriods;i++) {
115  restrict("switchOne[%1$] : 1",i);
116  restrict("switchTwo[%1$] : 1",i);
117 }
118 //restrict("temperature[0] :300");
119 ////////////////////////////////////////////////////WaterLaw//////////////////////////////////////////////////////////
120 double lawSetPoint=288;
121 double fluidTemperatureMinimum=273;
122 double fluidTemperatureMaximum=473;
Problems Javadoc Declaration Console
Writable Smart Insert 103 : 29
Eclipse File Edit Source Refactor Navigate Search Project Run Window Help lun. 22:40 Yanis.hs
Java – milp-workshop/canopea-tests/finalCanopeaModel1.0/CanopeaSystem.milp – Eclipse – /Users/hadjsaim/Documents/workspace
CanopeaSystem.milp
122 double fluidTemperatureMaximum=473;
123
124 prefix("waterLaw_");
125 load("canopea-library/finalCanopeaTestLibrary/WaterLaw.milp");
126
127 ////////////////////////////////////////////////////earthPanel//////////////////////////////////////////////////////////
128 prefix("earthPanel_");
129 double ambientTemperatureMinimum=0;
130 double ambientTemperatureMaximum=313;
131 double thermalCapacitance=4185;
132 double waterFlow=0.16;
133 double earthPanelArea=34;
134 double heatTransfertCoefficient=10;
135 load("canopea-library/finalCanopeaTestLibrary/EarthPanel.milp");
136
137
138 ////////////////////////////////////////////////////lightning Part//////////////////////////////////////////////////////////
139 prefix("artificialLighting_");
140 load("canopea-library/finalCanopeaTestLibrary/ArtificialLighting.milp");
141 restrict("nominalPowerInWatt: 50");
142 restrict("nominalLux: 150");
143 ////////////////////////////////////////////////////common//////////////////////////////////////////////////////////
144 ////////////////////////////////////////////////////enveloppe//////////////////////////////////////////////////////////
145 //Canepea parameters
146
147 Ra=3.13; // thermal resistance with average temperature of adjacent rooms
148 Ta=298; // average temperature of adjacent rooms
149 Rv=0.0098; // direct exchange with outdoor (ventilation, leaks,...)
150 Ri=0.0044; // thermal resistance for wall, indoor side
151 Ro=0.0132; // thermal resistance for wall, outdoor side
152 Cw=157500; // wall thermal capacity
153 /*
154 Cw=33750000; // wall thermal capacity
155
156 double Ra=3.13; // thermal resistance with average temperature of adjacent rooms
157 double Ta=20; // average temperature of adjacent rooms
158 double Ri=0.000201; // thermal resistance for wall, indoor side
159 double Ro=0.608; // thermal resistance for wall, outdoor side
160 double Cw=139860; // wall thermal capacity
*/
Problems Javadoc Declaration Console
Writable Smart Insert 103 : 29


Eclipse File Edit Source Refactor Navigate Search Project Run Window Help lun. 22:40 Yanis.hs
Java – milp-workshop/canopea-tests/finalCanopeaModel1.0/CanopeaSystem.milp – Eclipse – /Users/hadjsaim/Documents/workspace
CanopeaSystem.milp
*/
prefix("room1_");
double airExchangerEfficiency=.85;
int numberOfConfigurations = 6;
double[] ventilationFlows = {0.0001,355./3600/4,355./3600/2,3*355./3600/4,355./3600,0.2}; // three possible ventilation air flows in m3/s
load("canopea-library/finalCanopeaTestLibrary/MultiStateVentilatedThermalZoneNoNeighborhood.milp");
for(i=0 ;i<nPeriods; i++){
restrict("wallPower[%1$]:0",i);
restrict("configuration[%1$,0]:1",i);
}
restrict("initialTemperature : 297");
//////////thermalcomfort//////////
prefix("thermalcomfort_");
double minimumAcceptableTemperature=280.; //necessary while nonlinear constraints cannot be parsed
double preferredTemperature=297.; //necessary while nonlinear constraints cannot be parsed
double maximumAcceptableTemperature=300.; //necessary while nonlinear constraints cannot be parsed
load("canopea-library/finalCanopeaTestLibrary/ThermalComfort.milp");
restrict("minimumAcceptableTemperature: %1$",minimumAcceptableTemperature);
restrict("preferredTemperature: %1$",preferredTemperature);
restrict("maximumAcceptableTemperature: %1$",maximumAcceptableTemperature);
restrict("minimumTemperatureWhenNobody: 283");//283
//////////CO2comfort//////////
prefix("co2Comfort_");
load("canopea-library/finalCanopeaTestLibrary/Co2Comfort.milp");
//////////LightingComfort//////////
prefix("lightingComfort_");
minimumAcceptableLighting=0.; //necessary while nonlinear constraints cannot be parsed
preferredLighting=350.; //necessary while nonlinear constraints cannot be parsed
maximumAcceptableLighting=1000.; //necessary while nonlinear constraints cannot be parsed
load("canopea-library/finalCanopeaTestLibrary/LightingComfort.milp");
restrict("minimumAcceptableLighting: %1$",minimumAcceptableLighting);
restrict("preferredLighting: %1$",maximumAcceptableLighting);
restrict("maximumAcceptableLighting: %1$",maximumAcceptableLighting);
//////////northThreeStateShutter//////////
Problems Javadoc Declaration Console
Writable Smart Insert 103 : 29
Eclipse File Edit Source Refactor Navigate Search Project Run Window Help lun. 22:40 Yanis.hs
Java – milp-workshop/canopea-tests/finalCanopeaModel1.0/CanopeaSystem.milp – Eclipse – /Users/hadjsaim/Documents/workspace
CanopeaSystem.milp
//////////northThreeStateShutter//////////
prefix("northThreeStateShutter_");
load("canopea-library/finalCanopeaTestLibrary/threeStateShutter.milp");
//////////southThreeStateShutter//////////
prefix("southThreeStateShutter_");
load("canopea-library/finalCanopeaTestLibrary/threeStateShutter.milp");
//for(i=0 ;i<nPeriods; i++)
//restrict("opened[%1$]:1",i);
//////////eastThreeStateShutter//////////
prefix("eastThreeStateShutter_");
load("canopea-library/finalCanopeaTestLibrary/threeStateShutter.milp");
//////////westThreeStateShutter//////////
prefix("westThreeStateShutter_");
load("canopea-library/finalCanopeaTestLibrary/threeStateShutter.milp");
int solarPanelsSurfaceOnTheFirstInverter=58.1; //18 low module BIPV + 9 top module BIPV (27*84/39)
int solarPanelsSurfaceOnTheSecondInverter=41.9; //12 top module BIPV + 8 PV/T (12*84/39+16)
double bipvEfficiency=0.10;
double pvtEfficiency=0.152;
prefix("bipvOnFirstInventer_");
double pvSurfaceInM2=27*84/39;
double efficiency=bipvEfficiency;
load("canopea-library/finalCanopeaTestLibrary/PhotovoltaicPlant.milp");
prefix("bipvOnSecondInventer_");
double pvSurfaceInM2=12*84/39;
double efficiency=bipvEfficiency;
load("canopea-library/finalCanopeaTestLibrary/PhotovoltaicPlant.milp");
prefix("pvtOnSecondInventer_");
double pvSurfaceInM2=16;
double efficiency=pvtEfficiency;
load("canopea-library/finalCanopeaTestLibrary/PhotovoltaicPlant.milp");
Problems Javadoc Declaration Console
Writable Smart Insert 103 : 29

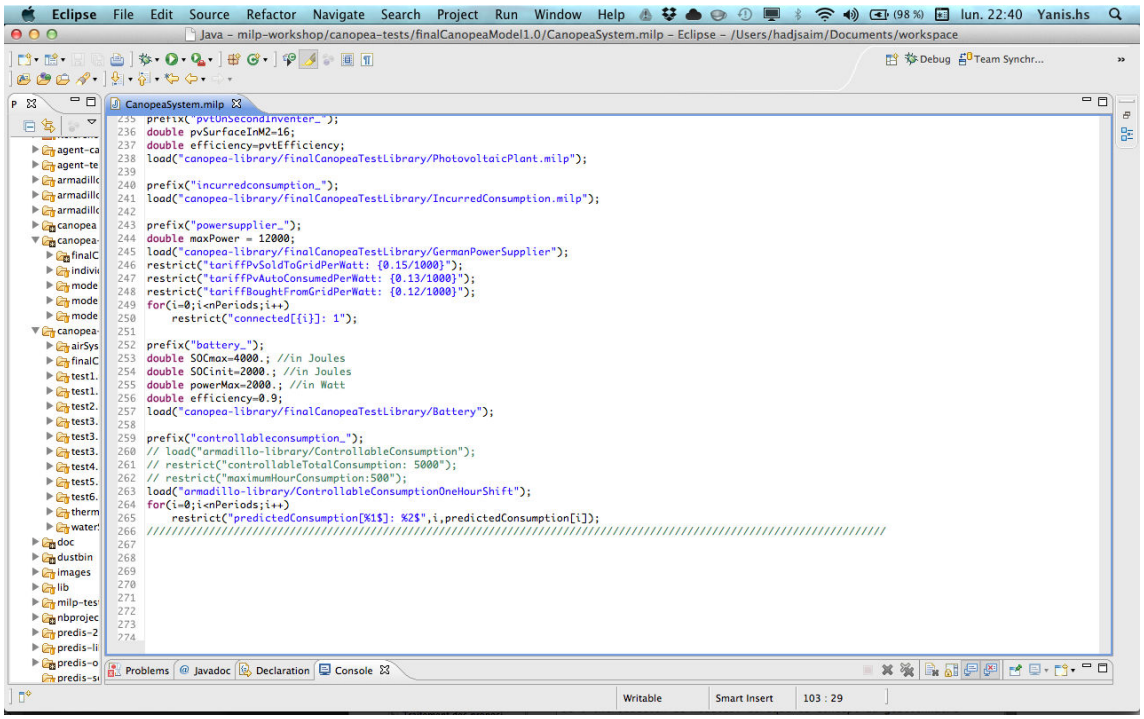



FIGURE *III.34 – Plan d'usage anticipé*

La structure de la bibliothèque des modèles est libre. Nous avons choisi de créer des dossiers pour chaque étape de développement. Chaque modèle est crée dans un fichier *Milp*. Il s'agit alors de tester chaque modèle séparément afin de relever ses bugs et ainsi les corriger tout en gardant un historique des modèles précédents. Cette tâche est nécessaire dans notre méthode de modélisation plus que dans d'autres applications car le paradigme de modélisation pour la gestion impose une démarche particulière. Il faut, en effet, commencer à partir des degrés de liberté envisagés la modélisation. Il s'agit ensuite de poser des contraintes autour de ces degrés de liberté jusqu'à avoir un comportement satisfaisant dans les conditions de tests réels. Cette méthode n'est pas infaillible et a tendance à embarquer des bugs qui ne sont visibles que dans des conditions particulières d'utilisation du modèle. Il est donc important d'éprouver chaque modèle avec des conditions d'usage normal mais aussi limites afin de poser l'ensemble des contraintes nécessaires. Nous avons développé notre bibliothèque autour de ce paradigme qui permet ainsi de retracer les évolutions de chaque modèle pour, à la fin, avoir le modèle le plus juste. Ce dernier est le seul utilisé dans la composition finale ainsi présentée, si cette dernière peut révéler des bugs lors de la composition. Ceux-ci sont d'autant plus complexes à traiter que les sources sont multiples à ce niveau. D'un point de vue langage, nous utilisons encore certaines écritures liées à JAVA dans les modèles. Cette caractéristique est liée à l'utilisation de BeanShell pour le parsage des modèles.

D'un point de vue résolution, il est utile de distinguer deux phases : La phase de traitement des modèles et la phase de calcul des plans d'usage anticipés. La première phase est due exclusivement à la surcouche Milp-workshop. C'est la composition des différents modèles et le traitement pour la mise en place du problème PLNE qui occasionne ce temps de calcul supplémentaire, lequel n'existe pas dans GhomeTech à la base. Le traitement dépend de la taille de la composition de modèles et des linéarisations et autres traitements à établir. Ce temps est aussi fonction de la machine et des ressources allouées à la JVM pour le traitement de la tâche. Le goulot d'étranglement dans notre traitement

est la mémoire vive disponible pour la JVM qui est de 500Mo par défaut. Celui-ci n'a pas été atteint durant la première phase. Cette phase dure entre 10 micro secondes pour le modèle du confort thermique traité seul et 5 secondes pour le modèle complet de Canopea sur une machine d'usage mixte (MAC OS, core I7 2,7KHz, 8Go RAM). La seconde phase de traitement est déléguée au solveur CPLEX comme dans GhomeTech. Nous avons eu, par ailleurs, recours aux mêmes connecteurs utilisés dans GHomeTech pour transférer le problème d'optimisation au solveur CPLEX. Le problème d'optimisation relatif à CANOPEA contient 12000 variables parmi lesquelles nous dénombrons 5480 variables binaires. La distinction entre les deux est importante car le nombre de variables binaires influe directement sur le temps de traitement du problème. CPLEX comme les autres solutions commerciales d'optimisation PLNE s'appuie sur deux types d'algorithmes : le simplex et le branch & bound. Ce dernier traite des variables discrètes dont les variables binaires. Celui-ci est gourmand en espace mémoire et en resources de calcul. Il nous est arrivé de saturer la mémoire vive accordée à la JVM de 500Mo. Nous l'avons étendu à 1 Go mais le temps de traitement explose. Le temps de traitement est la caractéristique déterminante dans notre outil car il s'agit de fournir l'information le plus rapidement possible afin qu'elle soit envoyée aux autres couches ou alors à l'utilisateur directement comme nous allons le présenter plus bas. Le temps de calcul est fonction du positionnement du problème, des simplifications numériques opérées et du domaine de définition des variables. Il nous est arrivé d'avoir des délais de traitement de 30 secondes à 2 heures pour une même structure du problème. La différence se joue à certaines modifications de degrés de liberté. Par exemple, lorsque nous choisissons une température de consigne fixe, c'est à dire que nous réduisons le domaine de variation à un singleton, le temps de calcul est réduit à 1min 30s (dans l'itération exécutée). Cette réduction qui parait spectaculaire dans un premier temps ne l'est pas en réalité car le fait de réduire la température de consigne à une valeur induit une diminution du nombre de variables assez importante pour le confort. En effet, le confort thermique est ainsi figé et toutes les variables binaires attachées deviennent fixes (singleton). Pour schématiser, cela reviendrait à fixer plus de 600 variables (le décompte relatif au modèle de confort) dans le problème. Le pré-traitement CPLEX élimine ainsi ces variables en les instanciant par des valeurs fixes (par l'utilisateur ou après calcul).

L'importance de la durée de calcul nous a mené à chercher à comprendre quelles sont les différentes étapes de calcul dans CPLEX et quelles étaient celles qui pouvaient être contractées. Nous avons alors trouvé des moyens pour limiter le temps de calcul du solveur. Le plus simple est l'ajustement du paramètre *timelimit* lequel est par défaut 1e75 secondes. Il existe aussi *ProbeDetTime* comme paramètre à régler. Ce dernier joue sur la preuve d'optimalité. C'est l'étape qui prend généralement le plus de temps dans le calcul total car il s'agit dans un premier temps de trouver une solution acceptable et puis, dans un second temps, trouver la meilleure solution ou alors prouver que c'est la meilleure solution. Cette étape est importante lorsqu'il s'agit de calculs précis alors que dans d'autres cas, il s'agit uniquement de donner des orientations grâce à des données toutes aussi grossières. En ce sens, un troisième paramètre concerne la tolérance sur l'optimalité de la solution *mipgap*. Ce paramètre est utilisé pour indiquer à CPLEX d'arrêter dès qu'il a trouvé un solution en nombres entiers réalisable dont il est prouvé qu'elle se trouve dans la marge déterminée. Pour définir une marge de cinq pour cent de la solution optimale par exemple, le *mipgap* doit être de 0,05.

Pour résumer nos approches de réduction du temps de calcul, nous utilisons CPLEX ainsi : la durée de recherche de preuve d'optimalité globale est aléatoire et peut prendre

plusieurs heures alors que des solutions approchées sont trouvées plus rapidement. Ces solutions approchées sont obtenues en modifiant certains paramètres du solveur CPLEX. Selon les objectifs et la résilience autorisée, la solution approchée peut être obtenue en limitant le temps de calcul total, le temps de preuve d'optimalité ou encore en élargissant le domaine de tolérance de l'erreur sur la solution optimale. D'un autre côté notamment, le temps de calcul peut être réduit selon certains scénarios d'usage qui consistent alors en des modifications des domaines de variation de certaines variables.

Entre les deux solutions, les critères de choix sont la contrainte technique et l'environnement dans lequel CPLEX est utilisé. Techniquement, il n'est pas toléré d'accorder une incertitude sur le délai de délivrance d'un résultat, au risque de déstabiliser le système dans certains cas. En cas de relance depuis la couche réactive pour cause d'écarts trop importants dus par exemple à des erreurs de prévisions météorologiques ou à des modifications de calendriers, le système doit être en mesure de fournir en un temps maximal fixé une solution car la couche réactive continue à appliquer le plan d'usage anticipé devenu inapproprié entre temps.

### III.7.3 Exemple d'usage des modèles CANOPEA

Après avoir abordé les différents moyens de développement puis de traitement des modèles, nous avons abordé les aspects calculatoires. Ces derniers ne sont pas dans le coeur du travail de cette thèse mais se sont imposés comme contraintes de travail qu'il fallait traiter. Revenons maintenant à l'aspect physique du problème traité dans ce chapitre. Notons qu'il est impossible, voire inutile, d'exposer l'ensemble des tests de validation sur chacun des modèles. Comme expliqué avant, chaque modèle est développé par itération et par tests. Nous avons néanmoins choisi d'exposer une expérimentation synthétique qui comprend l'ensemble des modèles d'équipements et du bâti de CANOPEA (figure III.36). Ceci afin de mettre en avant l'intérêt d'une telle composition par rapport à des optimisations partielles. La complexité de gestion des modèles d'une part et la complexité de calcul d'autre part poussent à segmenter le traitement du problème de gestion énergétique. Néanmoins, le résultat (figureIII.36) montre un certain intérêt du traitement global. Le résultat d'optimisation (figureIII.36) a été calculé sur la base d'un scénario dit *hiver* par rapport à la température extérieure et au niveau d'ensoleillement (figureIII.35). La première courbe en haut à gauche de (figureIII.35) décrit l'évolution du rayonnement solaire sur la maison durant 24h. Celui-ci, rappelons-le est en lien avec la production PV d'un coté et avec l'apport direct à travers les fenêtres d'autre part. La seconde courbe de (figureIII.35) intègre la température extérieure ainsi que la température intérieure choisie (température de consigne variable) et la température du stock thermique. La troisième courbe de (figureIII.35) représente l'occupation. Si nous corrélons les deux précédentes courbes, nous observons un stockage de chaleur dans l'espace de vie. Ceci a du sens car la période d'inoccupation correspond aussi à la période de forte production PV où le confort thermique importe peu. Le choix du stockage dans l'environnement au lieu du stock thermique peut être lié à une question de dynamiques (la dynamique de l'air est plus rapide que la dynamique du matériau à changement de phases) tout comme elle peut être liée à des approximations de modélisation qui orientent l'optimisation vers ce sens.

Les deux premières courbes de (figureIII.36) synthétisent le comportement de la CompactP. Nous remarquons que celle-ci se met en marche (à travers sa consommation) durant la présence des occupants principalement pour être en mode chauffage. Nous remarquons des instants ou le mode refroidissement est enclenché mais la consommation de la com-

pactP est minime à ces moments là. C'est en réalité des moments où la compactP est en veille mais comme ce mode n'est pas exprimé dans le modèle, le mode refroidissement est alors activé. C'est là une des limites de la modélisation PLNE car en cherchant des modèles justes en toutes circonstances celui-ci devient complexe. Nous acceptons donc ces erreurs à condition qu'elles ne soient pas mal interprétées dans la couche réactive de GHomeTech.

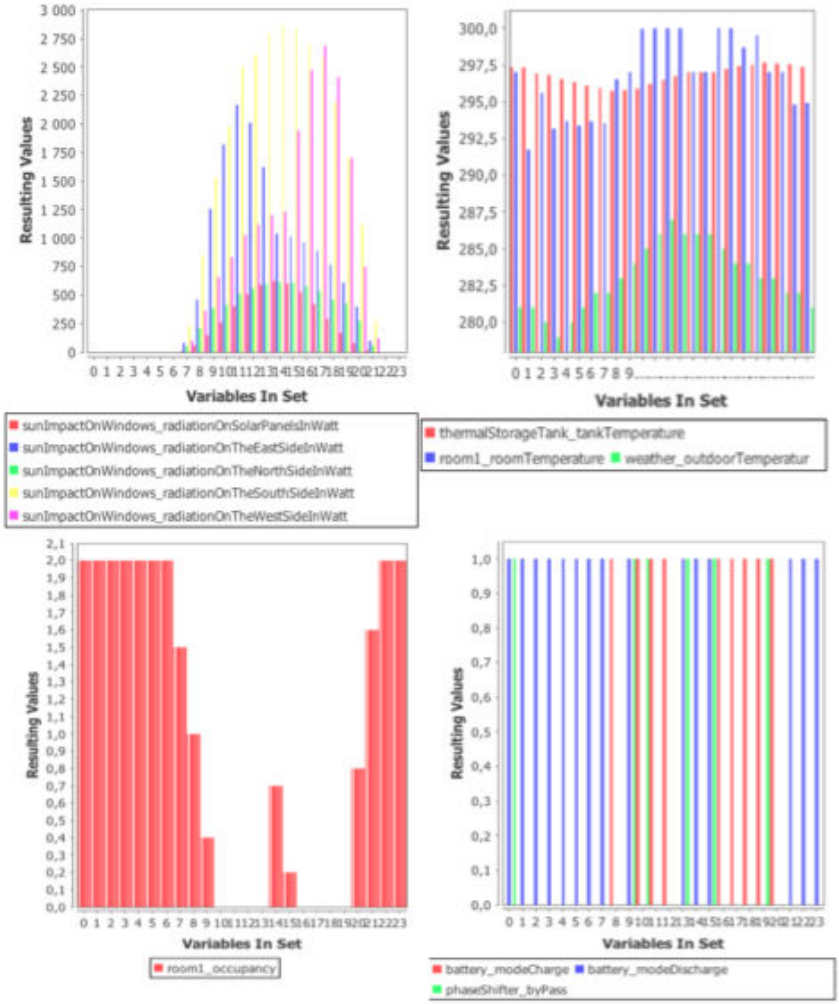



FIGURE *III.35 – Résultats d'optimisation(1/3)*

Enfin, la dernière (figure III.37) représente l'état des ouvrants qui modifient l'éclairage et l'isolation de l'enveloppe. Celle-ci devient permissive en cas d'ouverture et isolante dans le cas contraire. La nuit lorsqu'il fait froid et qu'il n'y a pas de soleil, le système préconise la fermeture. A contrario, en journée et selon l'orientation, les occultants sont en position ouverte pour la maximisation de l'apport énergétique.

### III.7.4 IHM du gestionnaire énergétique de CANOPEA

Avec l'aide d'une entreprise partenaire et d'une équipe d'ergonomes de l'école d'architecture de Grenoble, nous avons planché sur la question de la présentation des résultats à l'occupant en vue d'une meilleure acceptation d'une part et pour tenter de récolter de

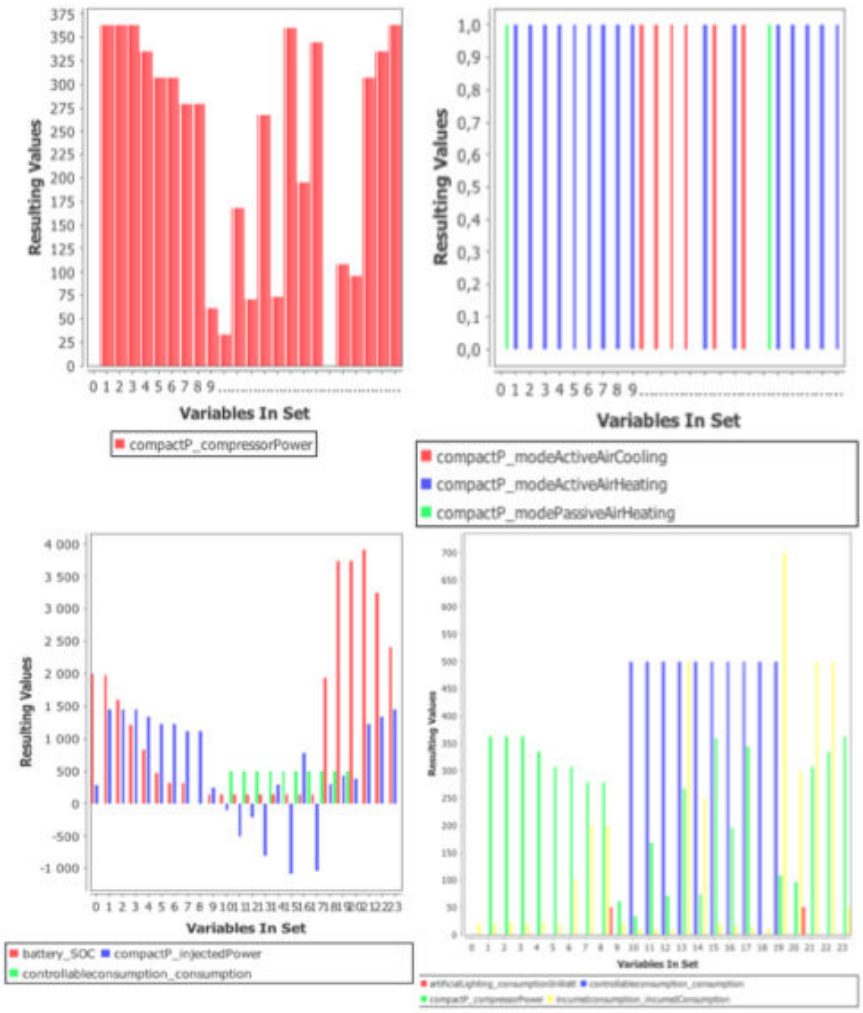



FIGURE *III.36 – Résultats d'optimisation(2/3)*

l'information dans un même temps d'autre part. Nous avons posé un cahier des charges sur la réalisation de cette interface :

- Présenter l'ensemble des données remontées par les capteurs.
- Proposer la commande à distance de l'ensemble des équipements qui peuvent l'être
- Intégrer la gestion du planning des occupants et de la vie sociale dans le bâtiment devant héberger CANOPEA.
- Soumettre à validation toute commande succeptible d'être émise par le système de gestion
- Permettre à l'utilisateur de personnaliser certains paramètres de gestion.
- Offrir à l'utilisateur plusieurs paliers dans l'automatisation de la gestion

Ces exigences ont été imposées à l'application Android développée comme IHM. Cette application traite de l'interaction avec l'occupant d'un côté et avec un serveur distant de l'autre. Le serveur distant se charge du traitement déporté de l'ensemble des tâches gourmandes en capacité de calcul et en capacités de stockage dynamique. GHomeTech et Milp-workshop y sont exécutés. L'interface Android présente dans sa fenêtre d'accueil une vue d'ensemble sur les systèmes technologiques dans l'habitat (figureIII.38).

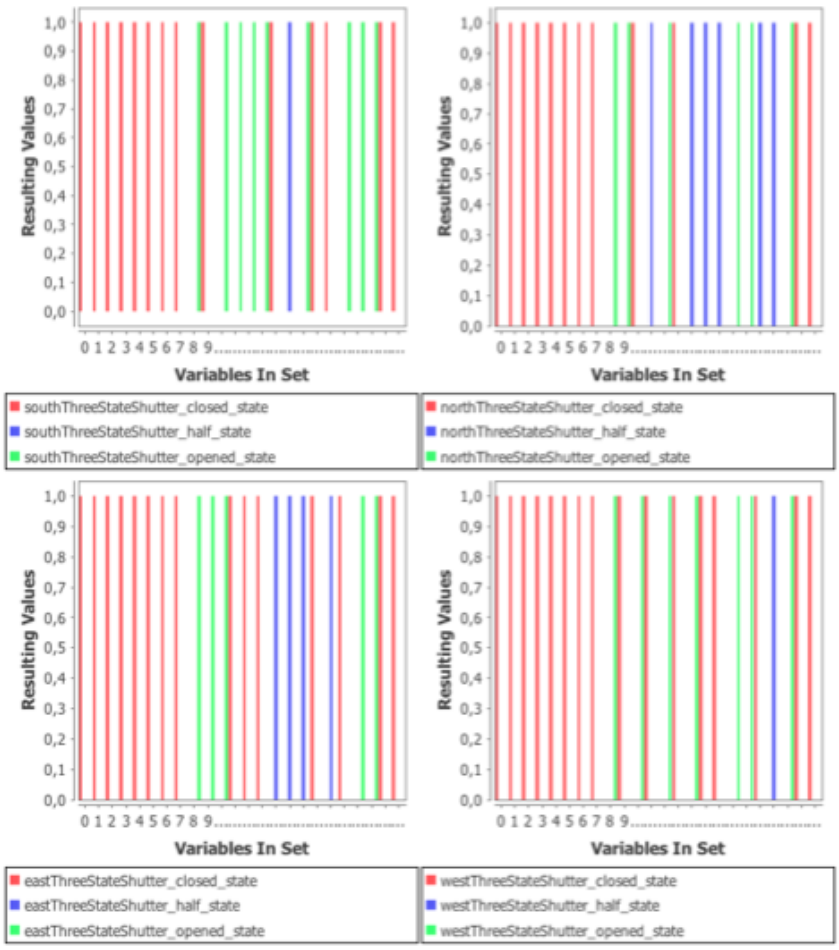



FIGURE *III.37 – Résultats d'optimisation(3/3)*


FIGURE *III.38 – Page d'accueil de l'IHM*

Celle-ci offre un onglet vue globale sur l'habitat, un onglet météorologie, un onglet environnement intérieur, un mur de publications de conseils énergétiques, un tableau de bord électrique avec production et consommation en temps réel, des liens pour accéder à l'historique et à des détails plus techniques.

En entrant dans la vue d'ensemble de l'habitat (figureIII.39), l'ensemble des capteurs/-contrôleurs est représenté par zone de vie. Il suffit alors à l'utilisateur de cliquer sur l'item voulu pour visualiser les dernières informations renvoyées par les capteurs. L'utilisateur peut aussi exécuter un ordre sur l'allumage ou l'extinction des lumières ou d'équipements

raccordés aux prises commandables. L'utilisateur peut aussi modifier l'état des occultants. Enfin, la température de consigne peut y être modifiée lorsque l'utilisateur choisit le mode de contrôle manuel.

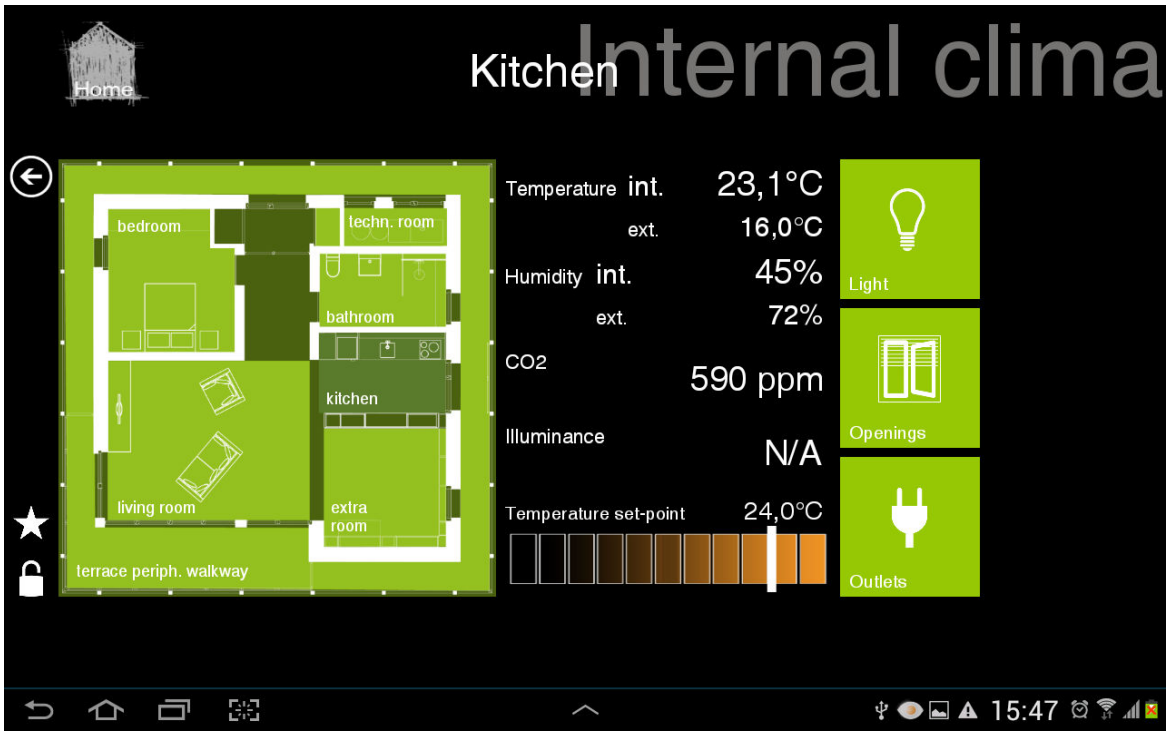



FIGURE *III.39 – Tableau de commande de l'habitat*

Dans l'onglet 'préférences' (figureIII.40), il est possible de choisir entre les différents modes de gestion proposés. Ceux-ci sont graduels entre la gestion complètement manuelle et la gestion complètement automatique à travers GhomeTech. Entre les deux, nous avons proposé des modes intermédiaires qui sont le conseil énergétique et la gestion semi-automatique (manuelle en cas de présence et automatique durant l'absence d'occupants).

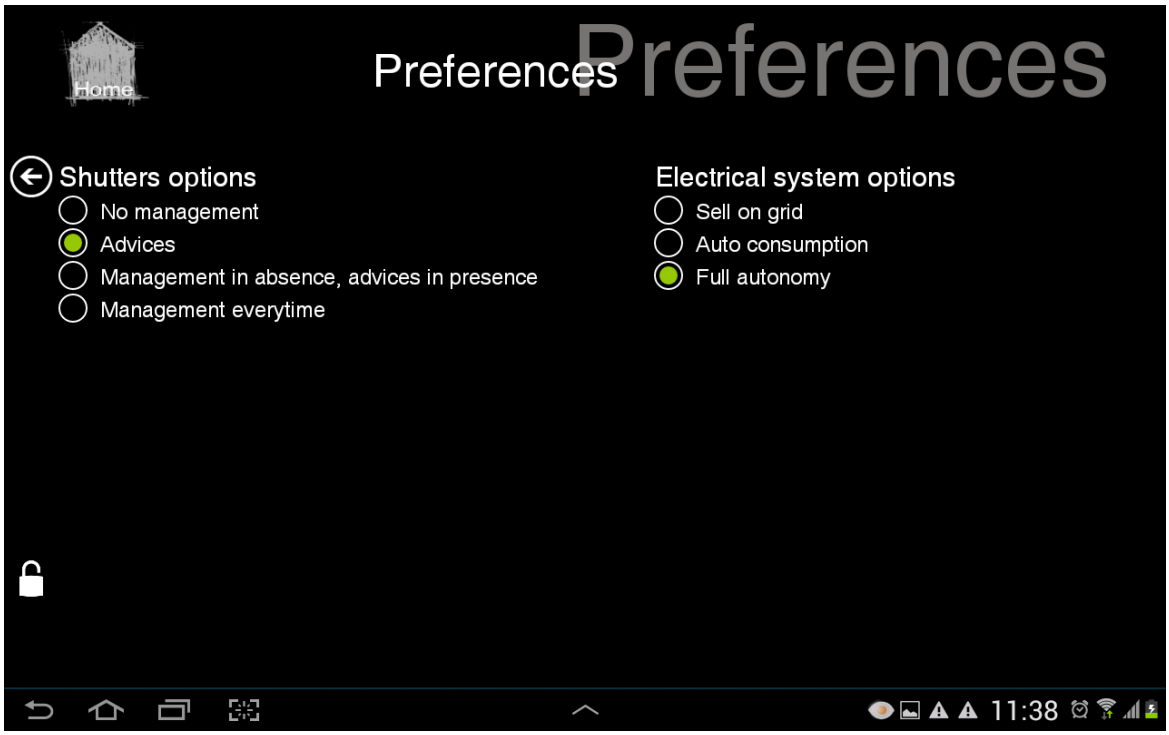



FIGURE *III.40 – Interface de choix du mode de gestion*

A côté de cela, il y a aussi un choix possible sur les objectifs de la gestion de certains équipements qui sont obligatoirement gérés par le système telle que la production et le stockage électriques. Dans l'onglet "énergies" de l'accueil (figureIII.41), nous retrouvons l'ensemble des historiques de consommation et de production d'électricité. Cet onglet a été mis en place afin de favoriser la prise de conscience et le suivi pour l'utilisateur.

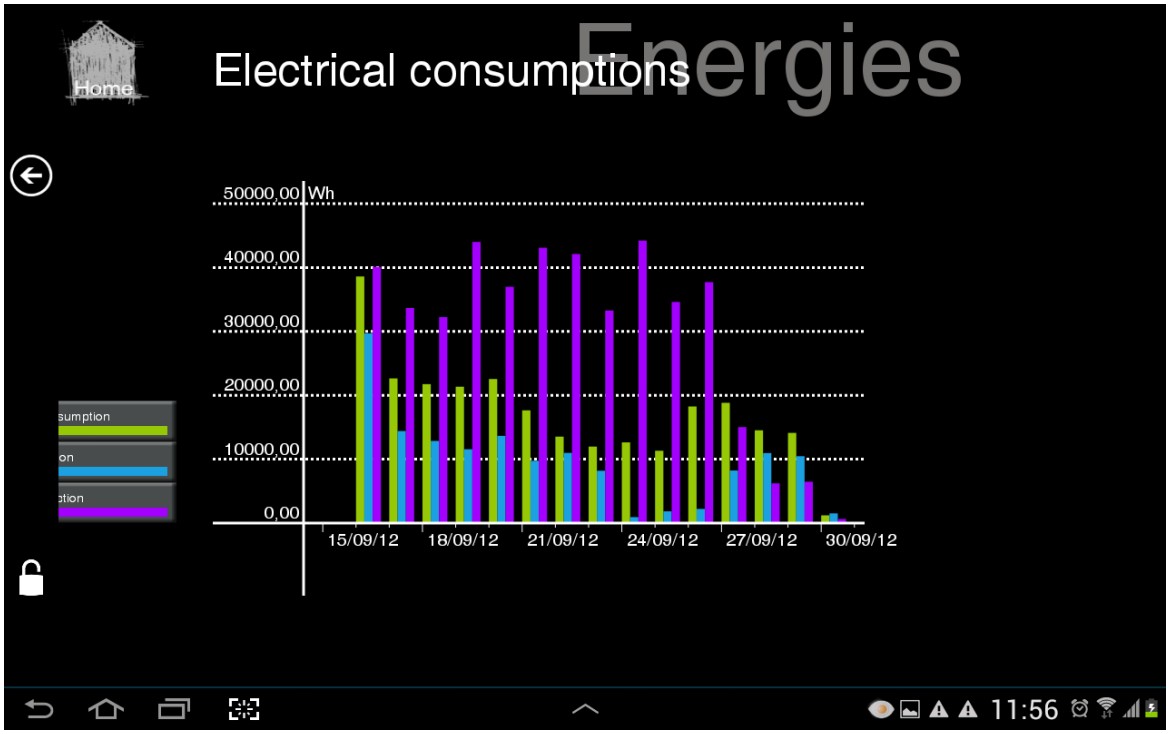



FIGURE *III.41 – Historique des consommations*

Enfin, dans l'onglet "maintenance" (figure III.42) il y a l'historique des conseils et alertes, il y a aussi une vue interactive du système thermique. Cette vue a été placée dans un objectif pédagogique.

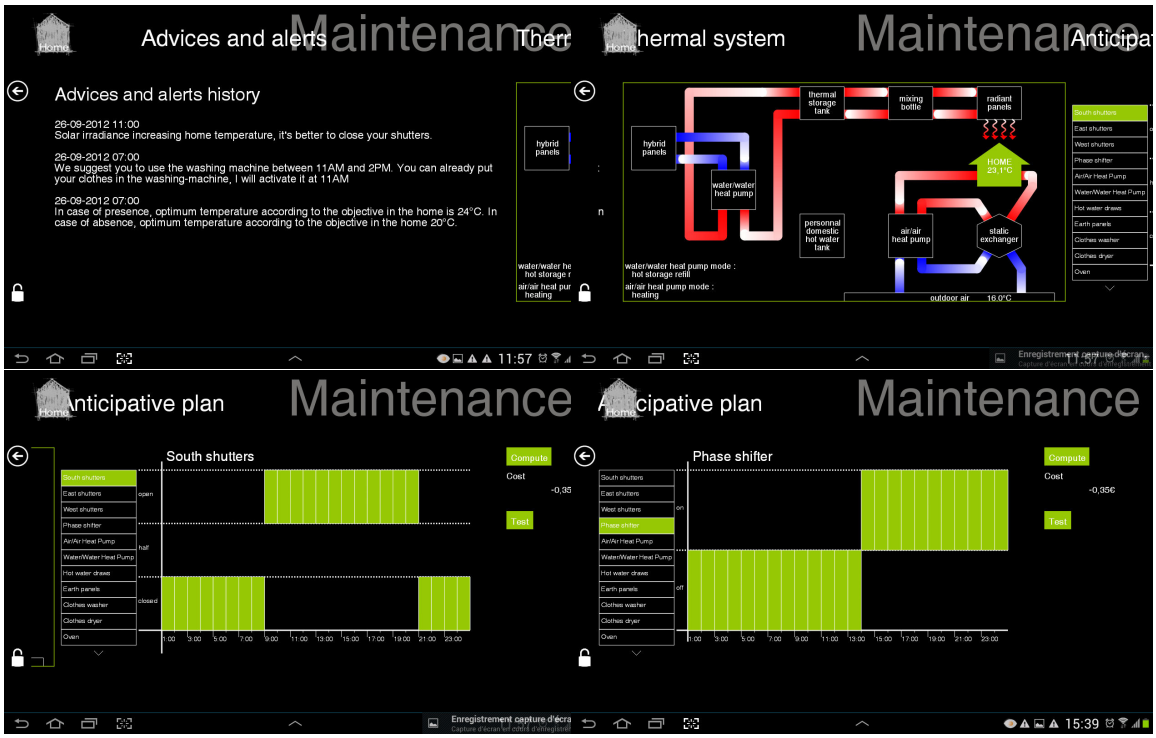



FIGURE *III.42 – Interaction pour la modification des plans d'usage anticipés*

La dernière partie concerne l'interaction avec l'occupant. La gestion automatique à travers GhomeTech peut être appuyée par l'occupant en validant l'ensemble des plans de gestion anticipés dans cette fenêtre. Il est alors permis à l'utilisateur de modifier les plans soumis tout en visualisant l'impact de la modification par rapport au plan initial.

Ce qui précède concerne la présentation de l'application IHM préparée pour CANOPEA. La commande des équipements était réellement implémentée au même titre que la communication avec le serveur distant dans lequel le processus de modélisation a été réalisé à travers Milp-workshop ainsi que les étapes de transformation vers un modèle PLNE. Seules les données d'entrée changent par la suite dans chaque nouvelle optimisation en vue du calcul d'un plan d'usage anticipé. Ce dernier résultat de calcul, est présenté à l'occupant qui peut effectivement le personnaliser à travers la manipulation des courbes de commande avant d'accepter son application effective sur l'habitat. Nous avons aussi insisté sur la possibilité offerte à l'occupant de reprendre la main sur la gestion des équipements à n'importe quel moment, hormis les équipements techniques qui eux ne sont pas de nature à être manipulés par l'occupant non expert. Nous avons voulu cette caractéristique du système afin de garantir à l'occupant la sensation de pleine maitrise de son habitat. Cette garantit permettra à terme une meilleure acceptation de la solution.

## III.8 Conclusion

Dans ce chapitre, nous avons présenté la complexité de l'application CANOPEA et comment cette complexité nous a conduit à developper de nouveaux concepts et outils

pour la gestion énergétique anticipée à 24h, avec un pas de résolution horaire. Comparée aux applications traitées jusqu'alors, la complexité du problème à traiter est très significativement supérieure.

Nous avons présenté comment les transformations proposées dans le chapitre précédent permettent d'appréhender la complexité d'un tel problème. L'implémentation des concepts que nous avons introduits s'est concrétisée dans l'outil Milp-Workshop développé en Java. Le langage parsé par l'outil permet d'appliquer un ensemble de mécanismes de transformations automatiques. Ces transformations conduisent aux modèles globaux qui peuvent être construits et validés étape par étape en étant transformés automatiquement en problèmes d'optimisation PLNE.

Les éléments principaux de CANOPEA ont été pris en compte à travers leur modèles dans l'élaboration du gestionnaire énergétique. Les deux systèmes de conditionnement de l'ambiance que sont le réseau à fluide caloporteur et le réseau d'air pulsé ont été modélisés sous forme d'éléments de modèle afin de garantir la flexibilité et l'interchangeabilité d'éléments. L'enveloppe constituée de parois ainsi que d'ouvrants et d'occultants, a aussi été prise en compte. Un ensemble d'équipements électroménagers consommateurs d'énergie commandables ou pas sont aussi intégrés au bilan énergétique global.

La mise en commun de l'ensemble des modules a été réalisé dans un fichier de composition. Ce fichier, de même format qu'un modèle, permet de connecter les variables des différents modèles qui sont en réalité les expression de liens d'interconnexion entre modèles. La définition de certains paramètres y est aussi présente. L'objectif d'optimisation ainsi que certaines contraintes et relations n'appartenant à aucun modèle y sont définis.

Le modèle ainsi complété est ensuite transformé par MILP-workshop. Les paramètres sont remplacés, les non-linéarités sont détectées, les schémas de linéarisation sont appliqués puis la projection temporelle intervient pour finaliser le problème et ainsi le poser au format PLNE.

Les résultats présentés dans ce chapitre concernent les trois parties que sont la modélisation puis le traitement des modèles et la présentation des résultats à travers l'IHM. Concernant la modélisation, il a été mis en évidence l'amélioration de la lisibilité des modèles en séparant l'apport du développeur de modèles entre la partie purement descriptive de l'élément modélisé des considérations calculatoires que sont la linéarisation et la projection temporelle. Concernant le traitement des modèles, nous avons soulevé le problème du temps de calcul et des différentes méthodes utilisables dans les algorithmes d'optimisation PLNE pour limiter ce temps. Enfin, la dernière partie concerne la présentation de l'interface pour la gestion énergétique de CANOPEA

A l'image des films de sciences fiction qui se retrouvent parfois moteurs voir inspirateurs d'innovations, cette application nous a permis à travers son test grandeur nature de comprendre ce que le public demande et ce qui peut être bien perçu de ce qui est inutile. De la présentation de cette dernière, nous avons constaté l'attrait des testeurs pour l'interaction. Nous avons implémenté la partie modification de plans d'usage anticipés uniquement à travers la mise en avant de l'impact de cette dernière mais d'autres leviers peuvent être exploités parmi lesquels le calcul de nouveaux plans d'usages anticipés à partir du plan initial et en prenant en compte les nouvelles contraintes de l'utilisateur. Cette approche nécessiterait alors des méthodes de calcul plus rapides que la PLNE et les algorithmes de CPLEX.

# Deuxième partie

# Gestion de la complexité liée à la diversité

# Chapitre IV

# Vers un environnement multi-application

Les applications qui concourent à la gestion énergétique dans le bâtiment au sens large sont envisagées ici. Chaque application est interconnectée avec les autres via des échanges de données. Chaque application comporte une volet 'résolution' et un volet 'modèle'. La partie résolution est spécifique à chaque application et requiert un formalisme de modélisation spécifique. Le volet 'modèle' doit correspondre au format requis pour décrire le système que constitue l'habitat, ses équipements et ses usagers. Chaque modèle dépend autant de l'élément physique à modéliser que de l'application pour lequel il est développé.

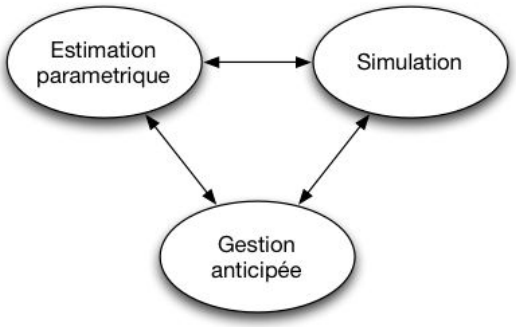



FIGURE *IV.1 – Interconnexions entre applications*

Il est d'usage de développer ou de réécrire un modèle spécifique pour chaque application mais les interconnexions requièrent une cohérence de modélisation pour les différentes applications. Par exemple, l'application d'estimation paramétrique permet d'estimer les valeurs des paramètres du modèle exploité pour la génération de stratégies de gestions : les 2 modèles doivent donc être cohérents pour que les données échangées puissent être interprétées. Les échanges apparaissent au niveau des flèches de la figure IV.1 .

Pour illustrer la problématique, analysons l'usage du modèle d'une enveloppe thermique par différentes applications. La représentation physique adoptée pour la modélisation de l'enveloppe thermique est conforme aux modélisations précédemment citées dans le manuscrit.

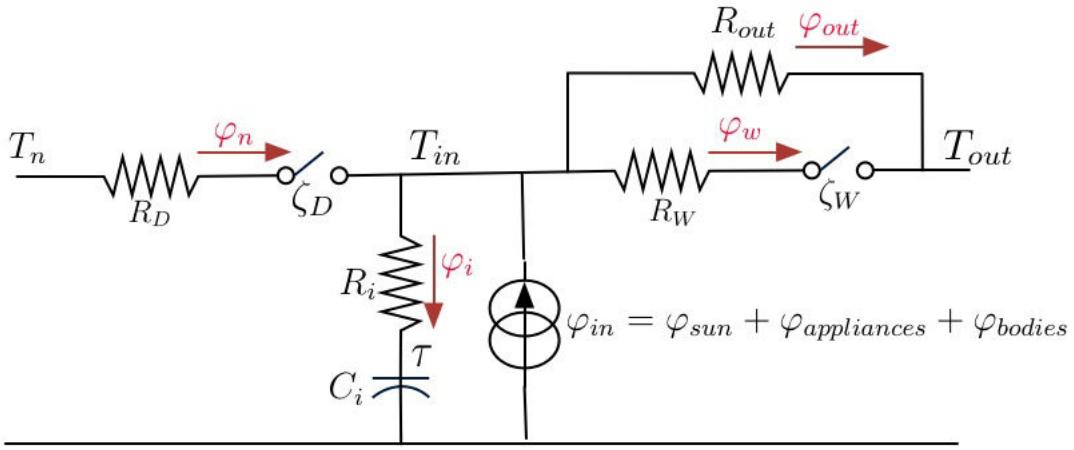



FIGURE *IV.2 – Modèle équivalent électrique de l'enveloppe*

Cette modélisation se base sur un schéma équivalent électrique figure IV.2 établie pour un bureau dans lequel sont représentées les résistances des parois et les inerties thermiques des masses. Les flux de chaleur ainsi que les températures en chaque point du modèle sont déduits par la loi des nœuds à la température $T_{in}$ et la loi des mailles sur les différentes boucles comme développé dans le système d'équations suivant :

$$\begin{aligned}
\zeta_D(T_n - T_{in}) &= R_D\varphi_n \\
T_{in} - \tau &= R_i\varphi_i \\
C_i\frac{d\tau}{dt} &= \varphi_i \\
T_{in} - T_{out} &= R_{out}\varphi_{out} \\
\zeta_W(T_{in} - T_{out}) &= R_W\varphi_w \\
\varphi_n + \varphi_{in} &= \varphi_i + \varphi_{out} + \varphi_w
\end{aligned} \tag{IV.1}$$

avec :

$T_n$ **:** température de l'espace environnant (couloir)

$T_{in}$ **:** température ambiante de l'espace modélisé

$T_{out}$ **:** température extérieure

$R_D$ **:** résistance thermique de la porte d'accès

$R_{out}$ **:** résistance thermique du mur séparant l'espace traité de l'extérieur

$R_W$ **:** résistance thermique de la fenêtre

$\tau$ **:** variable de calcul

$R_i$ **:** résistance de calcul

$C_i$ **:** inertie thermique des masses (murs et mobilier)

$\varphi_i$ **:** flux de chaleur de calcul

$\varphi_n$ **:** flux de chaleur traversant la porte

$\varphi_w$ **:** flux de chaleur traversant la fenêtre

$\varphi_{in}$ **:** flux de chaleur générée à l'intérieur de l'espace

$\varphi_{out}$ **:** flux de chaleur échangée avec l'extérieur

$\zeta_D$ **:** symbole binaire représentant le lien entre $T_n$ et $T_{in}$. C'est la représentation thermique de l'état de la porte entre le bureau et l'environnement externe l'entourant. Lorsque $\zeta_D$ est à 1 cela veut dire que la porte est ouverte et qu'il y a contact à travers la résistance $R_D$ entre $T_n$ et $T_{in}$.

$\zeta_W$ **:** symbole binaire représentant le lien entre $T_{in}$ et $T_{out}$ pour schématiser le comportement thermique lié à une fenêtre. Seul $R_{out}$ représente la résistance thermique entre l'intérieur $T_{in}$ et l'extérieur $T_{out}$. Lorsque $\zeta_W$ est fermé, $R_{out}$ et $R_w$ sont mis en parallèle pour représenter le nouvel état thermique du système.

D'un point de vue modélisation, l'écriture adoptée dans le système d'équations est une mise en relation de symboles et d'opérateurs. La représentation n'embarque pas d'informations concernant chaque symbole. Le système d'équations est posé en fonction de l'énoncé de la loi des nœuds et de la loi des mailles. Cette écriture ne prend pas en compte les spécificités des algorithmes de résolution de chaque application. Cette représentation est nommée dans ce qui suit *modèle descriptif* par opposition aux modèles utilisés par les applications qui sont nommés *modèles d'applications*.

En ce qui concerne les symboles utilisés, à ce stade, nous pouvons être affirmatifs sur certaines informations qui relèvent de la nature physique. Il y a d'abord des symboles qui représentent des grandeurs invariantes dans le temps. Ce sont les constantes liées au système physique que nous représentons par un symbole faisant référence à une grandeur invariante dans le temps, nommée $Isymbol$. Chaque environnement est caractérisé par sa propre température et délimité par les résistances thermiques l'entourant. Ces résistances font office de barrière thermique plus ou moins pénétrables. Certaines résistances thermiques ont un sens physique, en représentant des murs par exemple $R_i$, tandis que d'autres sont synthétiques et regroupent les différents types de fuites thermiques et des ponts. Les résistances $R_W$, $R_D$, $R_{out}$ associées à des interrupteurs représentent l'ensemble des combinaisons possibles. Les discontinuités de représentation sont alors intégrées de manière dynamique dans le modèle. $C_i$ représente la capacité de l'inertie thermique apportée par l'ensemble des masses présentes à l'intérieur et autour de l'espace (les murs).

Par défaut, le reste des symboles du modèle est considéré variant dans le temps, on parle de $Tsymbol$. Cette affectation par défaut peut être modifiée a posteriori mais uniquement en réduction, c'est à dire que seul le passage de $Tsymbol$ à $Isymbol$ est possible.

Enfin, un autre type d'information est aussi possible. Il concerne le domaine de variation. Les températures par exemple ne peuvent varier que dans des plages dites "envisageables". Nous pouvons choisir un domaine large regroupant tout type de température ou alors distinguer entre les températures intérieures et extérieures. C'est cette solution que nous choisissons :

$$
\begin{array}{lll}
T_n[k] & \text{in} & [10, 25] \\
T_{in}[k] & \text{in} & [10, 25] \\
T_{out}[k] & \text{in} & [-10, 45] \\
\tau[k] & \text{in} & [10, 25]
\end{array}
$$

### IV.0.1 Modèles d'application

Il peut y avoir autant de modèles d'application associés que d'applications liées à une même description physique. Chaque modèle d'application embarque, en plus de l'infor-

mation descriptive, de l'information liée au calcul et à la résolution du problème associé. Nous présentons tout d'abord le modèle correspondant à l'exemple IV.2 utilisés dans l'application gestion énergétique anticipée du BEMS GhomeTech. Nous présenterons ensuite des modèles pour l'application simulation et un modèle pour l'identification des paramètres.

### IV.0.1.1 Modèle pour la gestion énergétique anticipée

Le BEMS GhomeTech utilise la PLNE comme approche de résolution. Cette approche est applicable sur des modèles acausaux linéaires uniquement. Outre les opérateurs, deux types de symboles peuvent être utilisés : ceux correspondant à des variables et ceux à des paramètres. La variable est représentée par un symbole qui dépend du temps. Le domaine de variation associé est délimité par des bornes supérieure et inférieure, qui parfois se confondent lorsque la valeur est connue. Le paramètre est quant à lui un symbole invariant dans le temps et dont la valeur doit être connue dans cette application. La linéarité est imposée par la résolution PLNE. Une illustration du modèle déduit de l'équation IV.1, est présenté ci-dessous où $\zeta_{W,k}T_{in,k}$ et $\zeta_{D,k}T_{in,k}$ ont été linéarisés.

| | | | |
|---|---|---|---|
| $k$ | parameter | integer | $1..1..t_n$ |
| $\zeta_D[k]$ | variable | binary | $[0,1]$ |
| $\zeta_W[k]$ | variable | binary | $[0,1]$ |
| $T_n[k]$ | parameter | real | $[10,25]$ |
| $T_{in}[k]$ | variable | real | $[10,25]$ |
| $T_{out}[k]$ | parameter | real | $[-10,45]$ |
| $\tau[k]$ | variable | real | $[10,25]$ |
| $\varphi_n[k]$ | variable | real | $[-2000,2000]$ |
| $\varphi_i[k]$ | variable | real | $[-2000,2000]$ |
| $\varphi_w[k]$ | variable | real | $[-2000,2000]$ |
| $\varphi_{in}[k]$ | variable | real | $[-2000,2000]$ |
| $\varphi_{out}[k]$ | variable | real | $[-2000,2000]$ |
| $R_i$ | parameter | real | . |
| $R_D$ | parameter | real | . |
| $R_{out}$ | parameter | real | . |
| $R_W$ | parameter | real | . |
| $C_i$ | parameter | real | . |

$$
\begin{array}{lcl}
\tau(0) - init & = & 0 \\
Z_1(k) - Z_2(k) - R_D * \varphi_n(k) & = & 0 \\
T_{in}(k) - \tau(k) - R_i\varphi_i(k) & = & 0 \\
C_i * \tau(k) - C_i * \tau(k-1) - \Delta * \varphi_i(k) & = & 0 \\
T_{in}(k) - T_{out}(k) - R_{out}\varphi_{out}(k) & = & 0 \\
Z_3(k) - Z_4(k) - R_W\varphi_w(k) & = & 0 \\
\varphi_n(k) + \varphi_{in}(k) - \varphi_i(k) - \varphi_{out}(k) - \varphi_w(k) & = & 0 \\
Z_1(k) - T_n(k) + Min_{T_n} - Min_{T_n}\zeta_D(k) & \leq & 0 \\
Z_1(k) - T_n(k) + Max_{T_n} - Max_{T_n} * \zeta_D(k) & \geq & 0 \\
Z_1(k) - Max_{T_n} * \zeta_D(k) & \leq & 0 \\
Z_1(k) - Min_{T_n} * \zeta_D(k) & \geq & 0 \\
Z_2(k) - T_{in}(k) + Min_{T_{in}} - Min_{T_{in}} * \zeta_D(k)) & \leq & 0 \\
Z_2(k) - T_{in}(k) + Max_{T_{in}} - Max_{T_{in}} * \zeta_D(k)) & \geq & 0 \\
Z_2(k) - Max_{T_{in}} * \zeta_D(k) & \leq & 0 \\
Z_2(k) - Min_{T_{in}} * \zeta_D(k) & \geq & 0 \\
Z_3(k) - T_{in}(k) + Min_{T_{in}} - Min_{T_{in}} * \zeta_W(k) & \leq & 0 \\
Z_3(k) - T_{in}(k) + Max_{T_{in}} - Max_{T_{in}} * \zeta_W(k)) & \geq & 0 \\
Z_3(k) - Max_{T_{in}} * \zeta_W(k) & \leq & 0 \\
Z_3(k) - Min_{T_{in}} * \zeta_W(k) & \geq & 0 \\
Z_4(k) - T_{out}(k) + Min_{T_{out}} - Min_{T_{out}} * \zeta_W(k) & \leq & 0 \\
Z_4(k) - T_{out}(k) + Max_{T_{out}} - Max_{T_{out}} * \zeta_W(k) & \geq & 0 \\
Z_4(k) - Max_{T_{out}} * \zeta_W(k) & \leq & 0 \\
Z_4(k) - Min_{T_{out}} * \zeta_W(k) & \geq & 0
\end{array}
$$

### IV.0.1.2 Modèles de simulation

Nous présentons dans cette partie trois cas de simulations selon différents scénarios. Nous appelons scénario la combinaison entrées/sorties/paramètres du jeu de symboles utilisés dans le modèle. Une entrée se distingue d'un paramètre par la variation dans le temps pour l'entrée et l'invariance du paramètre.

**IV.0.1.2.1 Modèle de simulation pour l'estimation des flux** L'objectif de ce modèle est de permettre l'estimation des flux de chaleur circulant dans le circuit électrique représentant l'enveloppe : $\varphi_n$, $\varphi_i$,$\varphi_{out}$, $\varphi_w$ et $Q_{stock}$.

| | | | | | |
|---|---|---|---|---|---|
| $\zeta_D$ | input | $C_i$ | parameter | $Q_{stock}$ | output |
| $\zeta_W$ | input | $R_i$ | parameter | $\varphi_n$ | output |
| $T_n$ | input | $R_W$ | parameter | $\varphi_i$ | output |
| $T_{in}$ | input | $R_D$ | parameter | $\varphi_w$ | output |
| $T_{out}$ | input | $R_{out}$ | parameter | $\varphi_{in}$ | output |
| $\tau$ | input | | | $\varphi_{out}$ | output |

$$\begin{aligned}
\varphi_n &= \zeta_D \frac{T_n - T_{in}}{R_D} \\
\varphi_i &= \frac{T_{in} - \tau}{R_i} \\
\varphi_{out} &= \frac{T_{in} - T_{out}}{R_{out}} \\
\varphi_w &= \zeta_W \frac{T_{in} - T_{out}}{R_W} \\
Q_{stock} &= C_i(\tau - T_{in})
\end{aligned} \tag{IV.2}$$

Dans le cas présenté, l'ordre de résolution n'est pas important puisque chaque équation est indépendante des autres. Pour atteindre l'objectif défini, l'ensemble des symboles à droite des équations doit être remplacé par une unique valeur à chaque instant.

**IV.0.1.2.2 Modèle pour la simulation de la température intérieure** $T_{in}$ Un autre exemple d'usages du modèle descriptif initial est la déduction de la température $T_{in}$ par simulation suivant un scénario dans lequel nous choisissons les combinaisons de façon à avoir un modèle simulable comme suit :

| | | | | | |
|---|---|---|---|---|---|
| $\zeta_D$ | entrée | $R_i$ | parametre | $Q_{stock}$ | sortie |
| $\zeta_W$ | entrée | $R_D$ | parametre | $\varphi_n$ | sortie |
| $T_n$ | entrée | $R_{out}$ | parametre | $\varphi_i$ | sortie |
| $T_{out}$ | entrée | $R_W$ | parametre | $\varphi_w$ | sortie |
| | | $C_i$ | parametre | $\varphi_{in}$ | sortie |
| | | | | $\varphi_{out}$ | sortie |
| | | | | $T_{in}$ | sortie |
| | | | | $\tau$ | sortie |

$$
\begin{aligned}
\tau_{k+1} &= \left(1 + \frac{T_s(\mathcal{R}_k - R_i)}{R_i^2 C_i}\right)\tau_k + \frac{T_s \mathcal{R}_k}{R_i C_i}\varphi_{in,k} + \frac{T_s \mathcal{R}_k}{R_i C_i}\left(\frac{1}{R_{out,k}} + \frac{\zeta_{W,k}}{R_W}\right)T_{out,k} + \dots \\
&\quad \dots \frac{T_s \mathcal{R}_k \zeta_{D,k}}{R_D R_i C_i} T_{n,k} \\
T_{in,k} &= \frac{\mathcal{R}_k}{R_i}\tau_k + \mathcal{R}_k \varphi_{in,k} + \mathcal{R}_k\left(\frac{1}{R_{out,k}} + \frac{\zeta_{W,k}}{R_W}\right)T_{out,k} + \frac{\mathcal{R}_k \zeta_{D,k}}{R_D} T_{n,k} \qquad \text{(IV.3)} \\
&\quad ivar : R_i = ., R_W = ., R_D = ., C_i = . \\
&\quad \text{with } \frac{1}{\mathcal{R}_k} = \frac{1}{R_i} + \frac{1}{R_{out}} + \frac{\zeta_{W,k}}{R_W} + \frac{\zeta_{D,k}}{R_D}
\end{aligned}
$$

$\mathcal{R}_k$ représente une variable de synthèse qui permet de fluidifier la lecture du modèle. Dans un système automatisé, cette démarche n'est pas nécessaire.

La résolution de ce système d'équations donne comme résultat une série de valeurs temporelles correspondant à chaque instant.

**IV.0.1.2.3 Modèle de simulation pour l'estimation des besoins de chauffage** Parmi les applications courantes dans le secteur bâtiment, on trouve le calcul de la puissance consommée. Pour ce faire, nous avons besoin d'orienter le modèle de façon à avoir comme grandeur de sortie $\phi_{in}$.

La notion de temps est aussi prise en compte afin de rendre le modèle compatible avec les données en entrée :

| | | | | | |
|---|---|---|---|---|---|
| $\zeta_D$ | input | $R_i$ | parameter | $Q_{stock}$ | output |
| $\zeta_W$ | input | $R_D$ | parameter | $\varphi_n$ | output |
| $T_n$ | input | $R_{out}$ | parameter | $\varphi_i$ | output |
| $T_{out}$ | input | $R_W$ | parameter | $\varphi_w$ | output |
| $T_{in}$ | input | $C_i$ | parameter | $\varphi_{out}$ | output |
| | | | | $\tau$ | output |
| | | | | $\varphi_{in}$ | output |

$$
\begin{aligned}
\tau_{k+1} &= (1 - \frac{T_s}{R_i C_i})\tau_k + \frac{T_s}{R_i C_i} T_{in,k} \\
\varphi_{in,k} &= \frac{T_{in,k}}{\mathcal{R}} - (\frac{\tau_k}{R_i} + \frac{T_{n,k}\zeta_{D,k}}{R_D} + \frac{T_{out,k}}{R_{out}} + \frac{\zeta_{W,k} T_{out,k}}{R_W}) \qquad \text{(IV.4)} \\
&\quad ivar : R_i = ., R_W = ., R_D = ., C_i = . \\
&\quad R_{out,k} = .\forall, \varphi_{in,k} = .\forall, T_{out,k} = .\forall, T_{n,k} = .\forall, \zeta_{D,k} = .\forall, \zeta_{W,k} = .\forall \\
&\quad \text{with } \frac{1}{\mathcal{R}_k} = \frac{1}{R_i} + \frac{1}{R_{out}} + \frac{\zeta_{W,k}}{R_W} + \frac{\zeta_{D,k}}{R_D}
\end{aligned}
$$

#### IV.0.1.3 Modèle utilisé pour l'estimation paramétrique

Nous présentons ici un exemple de modèle pour l'estimation paramétrique. Ces paramètres sont utilisés dans les autres modèles d'application.

$$
\begin{aligned}
\tau_{k+1} &= \left(1 + \frac{T_s(\mathcal{R}_k - R_i)}{R_i^2 C_i}\right)\tau_k + \frac{T_s\mathcal{R}_k}{R_i C_i}\varphi_{in,k} + \frac{T_s\mathcal{R}_k}{R_i C_i}\left(\frac{1}{R_{out,k}} + \frac{\zeta_{W,k}}{R_W}\right)T_{out,k} + \dots \\
&\quad \dots \frac{T_s\mathcal{R}_k\zeta_{D,k}}{R_D R_i C_i}T_{n,k} \\
T_{in,k} &= \frac{\mathcal{R}_k}{R_i}\tau_k + \mathcal{R}_k\varphi_{in,k} + \mathcal{R}_k\left(\frac{1}{R_{out,k}} + \frac{\zeta_{W,k}}{R_W}\right)T_{out,k} + \frac{\mathcal{R}_k\zeta_{D,k}}{R_D}T_{n,k} \qquad \text{(IV.5)} \\
\Delta_k^T &= T_{in,k} - T_{in,k}^{sensor} \\
& ivar : R_i = ., R_W = ., R_D = ., C_i = . \\
& R_{out,k} = .\forall, \varphi_{in,k} = .\forall, T_{out,k} = .\forall, T_{n,k} = .\forall, \zeta_{D,k} = .\forall, \zeta_{W,k} = .\forall \\
& \text{with } \frac{1}{\mathcal{R}_k} = \frac{1}{R_i} + \frac{1}{R_{out}} + \frac{\zeta_{W,k}}{R_W} + \frac{\zeta_{D,k}}{R_D}
\end{aligned}
$$

## IV.0.2 Problématique

Les applications citées en exemple sont utilisées à différents instants

La simulation est souvent une étape préalable à la gestion pour la validation de la structure de modèle utilisée. Les paramètres sont, dans un premier temps, déduits des données physiques et la première simulation permet de vérifier que l'ensemble des sorties est physiquement cohérent. Mais la simulation tout comme différentes estimations peuvent être utilisées indépendamment de la gestion, pour prédire une situation présente ou future.

Les applications citées sont certes de natures différentes mais elles reposent sur une partie commune : le modèle descriptif.

Le processus impose une cohérence des modèles d'une application à une autre car au final, l'ensemble des manipulations a pour objectif la gestion énergétique au sens large d'un même système bâtiment.

Dans l'exemple traité, les représentations du modèle de l'enveloppe thermique sont issues de transformations et de compléments au modèle descriptif contenus dans le scénario. Les transformations peuvent être menées d'une manière manuelle ou en utilisant des outils de calcul symbolique, ou Computer Algebra System (CAS) en anglais.

Les parties suivantes détaillent chacune des transformations nécessaires pour la formulation du modèle applicatif à partir du modèle descriptif.

Le format ainsi que les éléments de chaque modèle applicatif est imposé par l'application et son solveur. L'objectif des transformations est donc d'aboutir à un modèle répondant au cahier de charge appelé modèle applicatif à partir du modèle descriptif.

L'ensemble des étapes énumérées pour les différentes applications ont été manuellement réalisées. La procédure est la même pour chaque modèle composant la description d'un habitat. Certaines étapes sont nécessaires chaque fois que nous souhaitons installer le BEMS dans un nouvel habitat même si les modèles sont structurellement identiques.

Nous nous sommes posés la question de la possibilité d'une automatisation, même partielle, des processus de transformation pour le passage d'un modèle à un autre. Pour cela nous décrivons les étapes nécessaires à l'obtention des modèles pour chacune des trois applications citées. Chaque description sera conclue par une discussion sur la possibilité et la pertinence de l'automatisation de certaines étapes.

## IV.1 Du modèle descriptif au modèle pour la gestion anticipative

Nous reprenons dans cette section l'exemple du modèle pour la gestion anticipative de l'enveloppe thermique afin de détailler étape par étape le travail effectué par le développeur de modèle pour passer du modèle descriptif au modèle applicatif pour la PLNE de GhomeTech. Pour ce faire, nous avons défini les deux besoins principaux pour un modèle PLNE :

— Le modèle doit contenir la description des variables à optimiser en termes de labels (symboles) et de domaine de valeurs.
— La structure du modèle est contenue dans des équations et des inéquations linéaires avec des variables réelles ou entières. Les équations doivent être de forme polynomiale de degrés 1.

Le problème est constitué du modèle respectant le cahier de charge associé. L'objectif est une variable choisie parmi les variables d'optimisation ou alors défini dans une nouvelle équation.

Dans un problème PLNE, il existe deux fonctions possibles pour les symboles utilisés dans les contraintes.

**Variable.** Une variable est un symbole (ou label) et un domaine de valeurs non réduit à une unique valeur. Une variable peut être temps variant avec des valeurs potentiellement différentes à chaque instant ou sa valeur peut-être invariante (notion d'ivar).

**Paramètre.** Un paramètre est un symbole (ou label) et un domaine de valeurs réduit à une unique valeur. Un paramètre peut être un invariant temporel, à l'instar de $R_i$, $R_W$ ou $C_i$ dans l'exemple que nous traitons. Un paramètre peut aussi dépendre du temps, à l'instar de $T_{out}$ qui est fournie par la météorologie : il représente dans ce cas une donnée temporelle.

La première étape consiste à recenser l'ensemble des symboles du modèle descriptif pour distinguer les grandeurs à déclarer comme variables d'optimisation des grandeurs à renseigner comme paramètres du problème d'optimisation. Le choix de déclaration de chaque symbole dépend des considérations physiques et de l'objectif attendu de l'usage du modèle applicatif (cible). Certaines grandeurs peuvent être classifiées intuitivement tandis que d'autres doivent satisfaire la contrainte de forme des équations.

Le modèle associe la température intérieure $T_{in}$ à la chaleur apportée dans l'espace $\phi_{in}$ en fonction de grandeurs subies telles que les températures des espaces voisins $T_n$, la température extérieure $T_{out}$, les apports solaires $\phi_{sun}$, les caractéristiques de l'enveloppe $R_D$, $R_W$, $R_i$, $C_i$ et l'impact des occupants à travers leur chaleur émise $\phi_{bodies}$, leur activité $\phi_{appliances}$, et l'état des ouvertures $\zeta_D$ et $\zeta_W$.

Reste les symboles $\tau$, $\phi_{heating}$ qui peuvent paraître non intuitifs à classifier mais qui doivent répondre à une exigence de liberté de résolution. Si $\tau$ ou $\phi_{heating}$ sont définis,

cela induit une définition indirecte des variables à optimiser et donc cela interdit toute possibilité d'optimisation. $\tau$ et $\phi_{heating}$ doivent être par conséquent des variables.

Une fois le travail de classification des symboles effectué, nous devons renseigner l'ensemble des variables d'optimisation sur la nature de leur domaine de valeurs (discrète, continue ou binaire), sur leur nature temporelle (variable dans le temps ou pas) ainsi que leur domaine de définition (borne min et borne max). Les paramètres doivent être déclarés comme tels et renseignés par leur valeur.

| | | | |
|---|---|---|---|
| $k$ | parameter | integer | $1..t_n$ |
| $\zeta_D[k]$ | variable | binary | $0..1$ |
| $\zeta_W[k]$ | variable | binary | $0..1$ |
| $T_n[k]$ | parameter | real | . |
| $T_{in}[k]$ | variable | real | $[10, 25]$ |
| $T_{out}[k]$ | parameter | real | . |
| $\tau[k]$ | variable | real | $[10, 25]$ |
| $\varphi_n[k]$ | variable | real | $[-2000, 2000]$ |
| $\varphi_i[k]$ | variable | real | $[-2000, 2000]$ |
| $\varphi_w[k]$ | variable | real | $[-2000, 2000]$ |
| $\varphi_{in}[k]$ | variable | real | $[-2000, 2000]$ |
| $\varphi_{out}[k]$ | variable | real | $[-2000, 2000]$ |
| $R_i$ | parameter | real | . |
| $R_D$ | parameter | real | . |
| $R_{out}$ | parameter | real | . |
| $R_W$ | parameter | real | . |
| $C_i$ | parameter | real | . |

La forme de contrainte acceptée par les solveurs PLNE exige un développement complet en contraintes linéaires où chaque variable n'apparaît que dans un monôme. Pour ce faire, toutes les contraintes sont passées à la fonction "expand" pour le développement des termes. Ce développement associé à la spécification des symboles permet de détecter les non-linéarités dans le système d'équations d'un côté et de mettre le modèle en conformité avec la forme admise par le solveur d'un autre côté. La linéarisation consiste en l'application des patterns développés dans le second chapitre de cette thèse pour chacune des formes non-linéaires recensées. Le résultat est un modèle dans lequel les produits et autres formes non-linéaires sont remplacés par des variables additionnelles contraintes de façon à reproduire le lien non linéaire entre les variables initiales sans la forme non linéaire. Cette étape ne garantit pas la linéarité du modèle résultant. Quand c'est le cas, nous avons proposé une possibilité d'interaction avec le développeur de modèle en lui exposant la forme non linéaire non traitée pour l'aider à trouver une solution de contournement. C'est le cas lorsqu'il y a un produit de deux variables continues : le contournement manuel peut être alors de reconsidérer la nature de l'une des deux variables.

$$\begin{aligned}
\zeta_D * T_n - \zeta_D * T_{in} &= R_D \varphi_n \\
T_{in} - \tau &= R_i \varphi_i \\
C_i \frac{d\tau}{dt} &= \varphi_i \\
T_{in} - T_{out} &= R_{out} \varphi_{out} \\
\zeta_W * T_{in} - \zeta_W * T_{out} &= R_W \varphi_w \\
\varphi_n + \varphi_{in} &= \varphi_i + \varphi_{out} + \varphi_w
\end{aligned} \tag{IV.6}$$

Quatre non-linéarités apparaissent dans le modèle. Les 4 non-linéarités sont de type semi-continu ou produit binaire-continu. Le pattern de linéarisation a été développé dans le chapitre 2 de cette thèse. Le résultat après application du pattern est le suivant :

$$\begin{aligned}
Z_1 - Z_2 &= R_D \varphi_n \\
T_{in} - \tau &= R_i \varphi_i \\
C_i \frac{d\tau}{dt} &= \varphi_i \\
T_{in} - T_{out} &= R_{out} \varphi_{out} \\
Z_3 - Z_4 &= R_W \varphi_w \\
\varphi_n + \varphi_{in} &= \varphi_i + \varphi_{out} + \varphi_w \\
Z_1 &\leq T_n - Min_{T_n} * (1 - \zeta_D) \\
Z_1 &\geq T_n - Max_{T_n} * (1 - \zeta_D) \\
Z_1 &\leq Max_{T_n} * \zeta_D \\
Z_1 &\geq Min_{T_n} * \zeta_D \\
Z_2 &\leq T_{in} - Min_{T_{in}} * (1 - \zeta_D) \\
Z_2 &\geq T_{in} - Max_{T_{in}} * (1 - \zeta_D) \\
Z_2 &\leq Max_{T_{in}} * \zeta_D \\
Z_2 &\geq Min_{T_{in}} * \zeta_D \\
Z_3 &\leq T_{in} - Min_{T_{in}} * (1 - \zeta_W) \\
Z_3 &\geq T_{in} - Max_{T_{in}} * (1 - \zeta_W) \\
Z_3 &\leq Max_{T_{in}} * \zeta_W \\
Z_3 &\geq Min_{T_{in}} * \zeta_W \\
Z_4 &\leq T_{out} - Min_{T_{out}} * (1 - \zeta_W) \\
Z_4 &\geq T_{out} - Max_{T_{out}} * (1 - \zeta_W) \\
Z_4 &\leq Max_{T_{out}} * \zeta_W \\
Z_4 &\geq Min_{T_{out}} * \zeta_W
\end{aligned} \tag{IV.7}$$

Après la linéarisation, le modèle doit être discrétisé en temps car la résolution se fait sur des périodes de 1 heure et sur un horizon de 24h. La résolution PLNE est basée sur un modèle complètement discrétisé pour lequel elle retourne le résultat pour chaque échantillon. Elle considère chaque échantillon temporel d'une variable du modèle comme une variable d'optimisation.

La projection temporelle nécessite une information supplémentaire dans le modèle : la nature temporelle de chaque symbole. L'ajout de cette information permet de projeter (c'est-à-dire démultiplier) les symboles liés au temps en fonction de la période et de l'horizon temporel visés.

Une fois les symboles décrits par des labels, les équations sont alors implicitement projetées car chaque équation impliquant une variable temporelle au minimum devient elle-même temporelle. Ceci est illustré dans le système ci-dessous. Les dérivées doivent elles-aussi être discrétisées. Le détail concernant le choix du type de discrétisation a été développé dans le chapitre 2. Néanmoins, nous rappelons que le choix s'est porté sur la méthode des différences finies qui a pour formule simplifiée :

$$\frac{d\tau}{dt} = \frac{\tau(t) - \tau(t - \Delta)}{\Delta} \tag{IV.8}$$

avec $\Delta$, le pas d'échantillonnage choisi et $\tau(0)$, l'initialisation nécessaire à la résolution. Le modèle devient ainsi :

$$\begin{aligned}
k\ in\ 1..24 \\
\tau(0) &= init \\
Z_1(k) - Z_2(k) &= R_D \varphi_n(k) \\
T_{in}(k) - \tau(k) &= R_i \varphi_i(k) \\
\varphi_i(k) &= C_i \frac{\tau(k) - \tau(k-1)}{\Delta} \\
T_{in}(k) - T_{out}(k) &= R_{out} \varphi_{out}(k) \\
Z_3(k) - Z_4(k) &= R_W \varphi_w(k) \\
\varphi_n(k) + \varphi_{in}(k) &= \varphi_i(k) + \varphi_{out}(k) + \varphi_w(k) \\
Z_1(k) &\leq T_n(k) - Min_{T_n} * (1 - \zeta_D(k)) \\
Z_1(k) &\geq T_n(k) - Max_{T_n} * (1 - \zeta_D(k)) \\
Z_1(k) &\leq Max_{T_n} * \zeta_D(k) \\
Z_1(k) &\geq Min_{T_n} * \zeta_D(k) \\
Z_2(k) &\leq T_{in}(k) - Min_{T_{in}} * (1 - \zeta_D(k)) \\
Z_2(k) &\geq T_{in}(k) - Max_{T_{in}} * (1 - \zeta_D(k)) \\
Z_2(k) &\leq Max_{T_{in}} * \zeta_D(k) \\
Z_2(k) &\geq Min_{T_{in}} * \zeta_D(k) \\
Z_3(k) &\leq T_{in}(k) - Min_{T_{in}} * (1 - \zeta_W(k)) \\
Z_3(k) &\geq T_{in}(k) - Max_{T_{in}} * (1 - \zeta_W(k)) \\
Z_3(k) &\leq Max_{T_{in}} * \zeta_W(k) \\
Z_3(k) &\geq Min_{T_{in}} * \zeta_W(k) \\
Z_4(k) &\leq T_{out}(k) - Min_{T_{out}} * (1 - \zeta_W(k)) \\
Z_4(k) &\geq T_{out}(k) - Max_{T_{out}} * (1 - \zeta_W(k)) \\
Z_4(k) &\leq Max_{T_{out}} * \zeta_W(k) \\
Z_4(k) &\geq Min_{T_{out}} * \zeta_W(k)
\end{aligned} \tag{IV.9}$$

Chaque indice *k* représente un pas d'échantillonnage.

La dernière étape pour mettre le modèle en conformité avec le formalisme PLNE consiste à réorganiser les équations au format PLNE. Pour ce faire, nous jouons sur une série de commandes dans un langage de calcul formel. La contrainte est chargée dans Maple par exemple sous la forme de deux expressions et la relation qui les lie "<" , ">", "=". Pour aboutir à la forme souhaitée, il suffit alors de multiplier l'expression de droite par $-1$ et la sommer avec l'expression de gauche. Il est intéressant après coup d'appliquer un "Simplify" sur la contrainte pour avoir la forme développée et contractée. Il suffit ensuite de récupérer le dernier terme, la constante, pour la remettre comme terme de gauche.

| | | | |
|---|---|---|---|
| $k$ | parameter | integer | $[1..t_n]$ |
| $\zeta_D[k]$ | variable | binary | $[0,1]$ |
| $\zeta_W[k]$ | variable | binary | $[0,1]$ |
| $T_n[k]$ | parameter | real | $[10,25]$ |
| $T_{in}[k]$ | variable | real | $[10,25]$ |
| $T_{out}[k]$ | parameter | real | $[-10,45]$ |
| $\tau[k]$ | variable | real | $[10,25]$ |
| $\varphi_n[k]$ | variable | real | $[-2000,2000]$ |
| $\varphi_i[k]$ | variable | real | $[-2000,2000]$ |
| $\varphi_w[k]$ | variable | real | $[-2000,2000]$ |
| $\varphi_{in}[k]$ | variable | real | $[-2000,2000]$ |
| $\varphi_{out}[k]$ | variable | real | $[-2000,2000]$ |
| $R_i$ | Parameter | | |
| $R_D$ | Parameter | | |
| $R_{out}$ | Parameter | | |
| $R_W$ | Parameter | | |
| $C_i$ | Parameter | | |

$$
\begin{array}{lcl}
\tau(0) - init & = & 0 \\
Z_1(k) - Z_2(k) - R_D * \varphi_n(k) & = & 0 \\
T_{in}(k) - \tau(k) - R_i\varphi_i(k) & = & 0 \\
C_i * \tau(k) - C_i * \tau(k-1) - \Delta * \varphi_i(k) & = & 0 \\
T_{in}(k) - T_{out}(k) - R_{out}\varphi_{out}(k) & = & 0 \\
Z_3(k) - Z_4(k) - R_W\varphi_w(k) & = & 0 \\
\varphi_n(k) + \varphi_{in}(k) - \varphi_i(k) - \varphi_{out}(k) - \varphi_w(k) & = & 0 \\
Z_1(k) - T_n(k) + Min_{T_n} - Min_{T_n}\zeta_D(k) & \leq & 0 \\
Z_1(k) - T_n(k) + Max_{T_n} - Max_{T_n} * \zeta_D(k) & \geq & 0 \\
Z_1(k) - Max_{T_n} * \zeta_D(k) & \leq & 0 \\
Z_1(k) - Min_{T_n} * \zeta_D(k) & \geq & 0 \\
Z_2(k) - T_{in}(k) + Min_{T_{in}} - Min_{T_{in}} * \zeta_D(k)) & \leq & 0 \\
Z_2(k) - T_{in}(k) + Max_{T_{in}} - Max_{T_{in}} * \zeta_D(k)) & \geq & 0 \\
Z_2(k) - Max_{T_{in}} * \zeta_D(k) & \leq & 0 \\
Z_2(k) - Min_{T_{in}} * \zeta_D(k) & \geq & 0 \\
Z_3(k) - T_{in}(k) + Min_{T_{in}} - Min_{T_{in}} * \zeta_W(k) & \leq & 0 \\
Z_3(k) - T_{in}(k) + Max_{T_{in}} - Max_{T_{in}} * \zeta_W(k)) & \geq & 0 \\
Z_3(k) - Max_{T_{in}} * \zeta_W(k) & \leq & 0 \\
Z_3(k) - Min_{T_{in}} * \zeta_W(k) & \geq & 0 \\
Z_4(k) - T_{out}(k) + Min_{T_{out}} - Min_{T_{out}} * \zeta_W(k) & \leq & 0 \\
Z_4(k) - T_{out}(k) + Max_{T_{out}} - Max_{T_{out}} * \zeta_W(k) & \geq & 0 \\
Z_4(k) - Max_{T_{out}} * \zeta_W(k) & \leq & 0 \\
Z_4(k) - Min_{T_{out}} * \zeta_W(k) & \geq & 0
\end{array}
$$

## IV.2 Du modèle physique vers des modèles de simulation

Nous avons pris comme exemple trois modèles de simulations dans l'introduction avec pour chacun un objectif distinct.

Pour avoir un modèle simulable, il faut tout d'abord orienter le modèle descriptif selon les données disponibles et les variables à calculer. La causalité n'est pas systématiquement assurée mais celle-ci est une condition nécessaire pour que le modèle soit simulable. Un modèle est simulable si les équations le composant peuvent être résolues et l'ensemble des variables réduites à une unique valeur après calcul.

Pour l'objectif d'estimation des flux de chaleur, l'orientation des entrées sorties de façon à assurer la causalité est suffisante. Le modèle résultant peut être résolu soit de manière continue par intégration, soit par récurrence en cas de pré-intégration sur une période.

L'orientation choisie pour l'objectif est résumée par le listing ci dessous :

| | | | | | |
|---|---|---|---|---|---|
| $\zeta_D$ | input | $C_i$ | parameter | $Q_{stock}$ | output |
| $\zeta_W$ | input | $R_i$ | parameter | $\varphi_n$ | output |
| $T_n$ | input | $R_W$ | parameter | $\varphi_i$ | output |
| $T_{in}$ | input | $R_D$ | parameter | $\varphi_w$ | output |
| $T_{out}$ | input | $R_{out}$ | parameter | $\varphi_{in}$ | output |
| $\tau$ | input | | | $\varphi_{out}$ | output |

Ce scénario offre la causalité nécessaire pour que le système d'équations puisse être résolu. L'étape suivante consiste à orienter chaque équation de façon à n'avoir que des symboles aux valeurs connues au moment de la résolution de l'équation à travers la fonction "solve" du logiciel de calcul formel. Cette fonction n'est pas posée pour résoudre les équations directement mais pour trouver une variable en fonction des autres. La démarche est ici très simple car chaque équation devient indépendante des autres :

| | | | | | |
|---|---|---|---|---|---|
| $\zeta_D$ | input | $C_i$ | parameter | $Q_{stock}$ | output |
| $\zeta_W$ | input | $R_i$ | parameter | $\varphi_n$ | output |
| $T_n$ | input | $R_W$ | parameter | $\varphi_i$ | output |
| $T_{in}$ | input | $R_D$ | parameter | $\varphi_w$ | output |
| $T_{out}$ | input | $R_{out}$ | parameter | $\varphi_{in}$ | output |
| $\tau$ | input | | | $\varphi_{out}$ | output |

$$\begin{aligned}
\varphi_n &= \zeta_D \frac{T_n - T_{in}}{R_D} \\
\varphi_i &= \frac{T_{in} - \tau}{R_i} \\
\varphi_{out} &= \frac{T_{in} - T_{out}}{R_{out}} \\
\varphi_w &= \zeta_W \frac{T_{in} - T_{out}}{R_W} \\
Q_{stock} &= C_i(\tau - T_{in})
\end{aligned} \tag{IV.10}$$

Le système (IV.10) admet une solution car chaque équation est entièrement renseignée par les entrées. L'ordre de résolution n'est pas important.

Un travail similaire à ce qui a été fait dans le précédent modèle est nécessaire concernant l'orientation entrées/sorties

| | | | | | |
|---|---|---|---|---|---|
| $\zeta_D$ | input | $R_i$ | parameter | $Q_{stock}$ | output |
| $\zeta_W$ | input | $R_D$ | parameter | $\varphi_n$ | output |
| $T_n$ | input | $R_{out}$ | parameter | $\varphi_i$ | output |
| $T_{out}$ | input | $R_W$ | parameter | $\varphi_w$ | output |
| | | $C_i$ | parameter | $\varphi_{in}$ | output |
| | | | | $\varphi_{out}$ | output |
| | | | | $T_{in}$ | output |
| | | | | $\tau$ | output |

La principale variable recherchée dans le modèle est $T_{in}$ les autres sorties peuvent être isolées du système d'équation comme suit en utilisant la fonction "solve" comme suit :

`solve( eq,` $T_{in}$`)`

$$\begin{aligned}
T_{in} - \tau &= R_i * C_i \frac{d\tau}{dt} \\
\frac{\zeta_D(T_n - T_{in})}{R_D} + \varphi_{in} &= \frac{T_{in} - \tau}{R_i} + \frac{T_{in} - T_{out}}{R_{out}} + \frac{\zeta_W(T_{in} - T_{out})}{R_W}
\end{aligned} \tag{IV.11}$$

$$\begin{aligned}
\varphi_n &= \frac{\zeta_D(T_n - T_{in})}{R_D} \\
\varphi_i &= \frac{T_{in} - \tau}{R_i} \\
\varphi_{out} &= \frac{T_{in} - T_{out}}{R_{out}} \\
\varphi_w &= \frac{\zeta_W(T_{in} - T_{out})}{R_W}
\end{aligned} \tag{IV.12}$$

Les deux premières équations ne contiennent plus que deux sorties. En terme de causalité, cela indique qu'il y a une possibilité pour que le système des deux équations à deux inconnues soient résolvables. Il faut trouver une forme dans laquelle chacune des

deux variables soit déduite d'une équation. Le système de deux équations représente une forme particulière : une représentation d'état donc la résolution est connue.

$$\frac{d\tau}{dt} = -\frac{1}{R_i C_i}\tau + \frac{1}{R_i C_i}T_{in}$$
$$(\frac{\zeta_D}{R_D} + \frac{1}{R_i} + \frac{1}{R_{out}} + \frac{\zeta_W}{R_W})T_{in} = \frac{1}{R_i}\tau + (\varphi_{in} + \frac{T_n\zeta_D}{R_D} + \frac{T_{out}}{R_{out}} + \frac{\zeta_W T_{out}}{R_W}) \quad \text{(IV.13)}$$

Pour une meilleure lisibilité des équations, nous considérons la variable synthétique $\mathcal{R}$ selon la formule suivante :

$$\frac{1}{\mathcal{R}} = \frac{1}{R_i} + \frac{1}{R_{out}} + \frac{\zeta_W}{R_W} + \frac{\zeta_D}{R_D}$$

En intégrant la variable synthétique, nous aboutissons au système suivant :

$$\frac{d\tau}{dt} = \frac{\mathcal{R} - R_i}{R_i^2 C}\tau + \frac{\mathcal{R}}{R_i C_i}\varphi_{in} + \frac{\mathcal{R}}{R_i C_i}\left(\frac{1}{R_{out}} + \frac{\zeta_W}{R_W}\right)T_{out} + \frac{\mathcal{R}\asymp_{\mathcal{D}}}{R_D R_i C_i}T_n$$
$$T_{in} = \frac{\mathcal{R}}{R_i}\tau + \mathcal{R}\varphi_{in} + \mathcal{R}\left(\frac{1}{R_{out}} + \frac{\zeta_W}{R_W}\right)T_{out} + \frac{\mathcal{R}\asymp_{\mathcal{D}}}{R_D}T_n \quad \text{(IV.14)}$$

Etant donné que le modèle attendu est un modèle discret. Nous choisissons la solution discrète de la représentation d'état. Pour ce faire, il faut discrétiser le système d'équations :

| | | | | | |
|---|---|---|---|---|---|
| $\zeta_D$ | input | $R_i$ | parameter | $Q_{stock}$ | output |
| $\zeta_W$ | input | $R_D$ | parameter | $\varphi_n$ | output |
| $T_n$ | input | $R_{out}$ | parameter | $\varphi_i$ | output |
| $T_{out}$ | input | $R_W$ | parameter | $\varphi_w$ | output |
| | | $C_i$ | parameter | $\varphi_{in}$ | output |
| | | | | $\varphi_{out}$ | output |
| | | | | $T_{in}$ | output |
| | | | | $\tau$ | output |

$$\tau_{k+1} = \left(1 + \frac{T_s(\mathcal{R}_k - R_i)}{R_i^2 C_i}\right)\tau_k + \frac{T_s\mathcal{R}_k}{R_i C_i}\varphi_{in,k} + \frac{T_s\mathcal{R}_k}{R_i C_i}\left(\frac{1}{R_{out,k}} + \frac{\zeta_{W,k}}{R_W}\right)T_{out,k} + \dots$$
$$\dots \frac{T_s\mathcal{R}_k\zeta_{D,k}}{R_D R_i C_i}T_{n,k}$$
$$T_{in,k} = \frac{\mathcal{R}_k}{R_i}\tau_k + \mathcal{R}_k\varphi_{in,k} + \mathcal{R}_k\left(\frac{1}{R_{out,k}} + \frac{\zeta_{W,k}}{R_W}\right)T_{out,k} + \frac{\mathcal{R}_k\zeta_{D,k}}{R_D}T_{n,k} \quad \text{(IV.15)}$$
$$ivar : R_i = ., R_W = ., R_D = ., C_i = .$$
$$R_{out,k} = .\forall, \varphi_{in,k} = .\forall, T_{out,k} = .\forall, T_{n,k} = .\forall, \zeta_{D,k} = .\forall, \zeta_{W,k} = .\forall$$

La résolution de ce système d'équations est possible si l'ordre est respecté. La première équation donne la valeur de $\tau$ puis la seconde équation donne la valeur de $T_{in}$ à chaque pas de temps $k$.

Le troisième cas de figure relève de la même procédure que la précédente avec des conditions de départ différentes. La quantité d'apport de chaleur ' $\phi_{in}$ est la sortie attendue pour un profil de température intérieure $T_{in}$ imposé.

L'orientation devient :

| | | | | | |
|---|---|---|---|---|---|
| $\zeta_D$ | input | $R_i$ | parameter | $Q_{stock}$ | output |
| $\zeta_W$ | input | $R_D$ | parameter | $\varphi_n$ | output |
| $T_n$ | input | $R_{out}$ | parameter | $\varphi_i$ | output |
| $T_{out}$ | input | $R_W$ | parameter | $\varphi_w$ | output |
| $T_{in}$ | input | $C_i$ | parameter | $\varphi_{out}$ | output |
| | | | | $\tau$ | output |
| | | | | $\varphi_{in}$ | output |

La démarche pour aboutir au modèle nécessaire est similaire à celle adoptée pour le modèle précédent. Les équations finales sont réorientées encore une fois afin de mettre en évidence la nouvelle grandeur recherchée $\varphi_{in}$. Cela commence par la mise en évidence de $\varphi_{in,k}$ dans la seconde équation à travers la commande :

`solve( eq,` $\varphi_{in,k}$`)`

Le résultat est :

$$\frac{d\tau}{dt} = -\frac{1}{R_i C_i}\tau + \frac{1}{R_i C_i}T_{in}$$
$$(\frac{\zeta_D}{R_D} + \frac{1}{R_i} + \frac{1}{R_{out}} + \frac{\zeta_W}{R_W})T_{in} = \frac{1}{R_i}\tau + (\varphi_{in} + \frac{T_n\zeta_D}{R_D} + \frac{T_{out}}{R_{out}} + \frac{\zeta_W T_{out}}{R_W}) \quad \text{(IV.16)}$$

avec

$$\frac{1}{\mathcal{R}} = \frac{1}{R_i} + \frac{1}{R_{out}} + \frac{\zeta_W}{R_W} + \frac{\zeta_D}{R_D}$$

L'équation devient :

$$\frac{d\tau}{dt} = -\frac{1}{R_i C_i}\tau + \frac{1}{R_i C_i}T_{in}$$
$$T_{in} = \frac{\mathcal{R}}{R_i}\tau + \mathcal{R}(\varphi_{in} + \frac{T_n\zeta_D}{R_D} + \frac{T_{out}}{R_{out}} + \frac{\zeta_W T_{out}}{R_W}) \quad \text{(IV.17)}$$

De l'équation (IV.17) permet de déduire $\tau$. Celui-ci servira à déduire par la suite $\varphi_{in}$ de la seconde équation avec le reste des entrées comme suit :

$$\frac{d\tau}{dt} = -\frac{1}{R_i C_i}\tau + \frac{1}{R_i C_i}T_{in}$$
$$\varphi_{in} = \frac{T_{in}}{\mathcal{R}} - (\frac{\tau}{R_i} + \frac{T_n\zeta_D}{R_D} + \frac{T_{out}}{R_{out}} + \frac{\zeta_W T_{out}}{R_W}) \quad \text{(IV.18)}$$

Après discrétisation, le modèle devient :

| | | | | | |
|---|---|---|---|---|---|
| $\zeta_D$ | input | $R_i$ | parameter | $Q_{stock}$ | output |
| $\zeta_W$ | input | $R_D$ | parameter | $\varphi_n$ | output |
| $T_n$ | input | $R_{out}$ | parameter | $\varphi_i$ | output |
| $T_{out}$ | input | $R_W$ | parameter | $\varphi_w$ | output |
| $T_{in}$ | input | $C_i$ | parameter | $\varphi_{out}$ | output |
| | | | | $\tau$ | output |
| | | | | $\varphi_{in}$ | output |

$$
\begin{aligned}
\tau_{k+1} &= (1 - \frac{T_s}{R_i C_i})\tau_k + \frac{T_s}{R_i C_i} T_{in,k} \\
\varphi_{in,k} &= \frac{T_{in,k}}{\mathcal{R}} - (\frac{\tau_k}{R_i} + \frac{T_{n,k}\zeta_{D,k}}{R_D} + \frac{T_{out,k}}{R_{out}} + \frac{\zeta_{W,k} T_{out,k}}{R_W}) \\
& ivar : R_i = ., R_W = ., R_D = ., C_i = . \\
& R_{out,k} = .\forall, \varphi_{in,k} = .\forall, T_{out,k} = .\forall, T_{n,k} = .\forall, \zeta_{D,k} = .\forall, \zeta_{W,k} = .\forall
\end{aligned}
\tag{IV.19}
$$

Les étapes de transformations du modèle descriptif en différents modèles d'application de simulation ont été illustrées dans ce que nous venons de présenter. Il y a une procédure similaire dans les trois transformations pour les différents usages. Des informations supplémentaires ont été nécessaires telles que les orientations entrées/sortie. C'est le scénario.

Le travail de transformation a consisté à positionner les équations de façon à retourner un modèle résolvable d'une part et à adapter ce modèle au type de solveur selon qu'il est continu ou discret d'autre part. L'utilité de chaque modèle est avérée dans le processus global de gestion énergétique. Ce travail revient souvent en modélisation car le choix d'une structure particulière de modélisation thermique n'est pas fixe et les applications multiples. Le problème est que, jusqu'à présent, il n'y a pas de procédure pouvant automatiser le processus de transformation et produire le modèle sans intervention importante d'un expert.

## IV.3 Du modèle descriptif vers un modèle d'estimation paramétrique

L'estimation paramétrique est fondamentale à la gestion énergétique car elle permet de faire correspondre un modèle à un contexte réel mesuré.

Le modèle étant non-linéaire en les paramètres à estimer (IV.19), contrairement à la gestion anticipative, il faut avoir recours à un algorithme d'optimisation non-linéaire.

Le modèle pour l'estimation paramétrique est un modèle de simulation où certains paramètres, du modèle d'optimisation ou de simulation correspondant, deviennent des variables invariantes dans le temps, et certaines variables, dans le même sens, des paramètres variant dans le temps où les valeurs coïncident avec des valeurs mesurées. Le modèle d'estimation paramétrique est donc déduit du modèle descriptif à l'instar du modèle de simulation ou d'optimisation. Néanmoins, il est aussi nécessaire de rajouter le calcul de l'écart entre les données mesurées et les données calculées par le simulateur. L'écart est la variable à minimiser par l'algorithme d'optimisation non-linéaire.

Appliquons la transformation du modèle descriptif vers un modèle d'estimation paramétrique sur l'exemple de l'enveloppe thermique.

Le modèle devient :

| | | | | | |
|---|---|---|---|---|---|
| $\zeta_D$ | input | $R_i$ | parameter | $Q_{stock}$ | output |
| $\zeta_W$ | input | $R_D$ | parameter | $\varphi_n$ | output |
| $T_n$ | input | $R_{out}$ | parameter | $\varphi_i$ | output |
| $T_{out}$ | input | $R_W$ | parameter | $\varphi_w$ | output |
| $T_{in,k}^{sensor}$ | benchmark input | $C_i$ | parameter | $\varphi_{in}$ | output |
| | | | | $\varphi_{out}$ | output |
| | | | | $T_{in}$ | output |
| | | | | $\tau$ | output |

$$
\begin{aligned}
\tau_{k+1} &= \left(1 + \frac{T_s(\mathcal{R}_k - R_i)}{R_i^2 C_i}\right)\tau_k + \frac{T_s\mathcal{R}_k}{R_i C_i}\varphi_{in,k} + \frac{T_s\mathcal{R}_k}{R_i C_i}\left(\frac{1}{R_{out,k}} + \frac{\zeta_{W,k}}{R_W}\right)T_{out,k} + \dots \\
&\quad \dots \frac{T_s\mathcal{R}_k\zeta_{D,k}}{R_D R_i C_i}T_{n,k} \\
T_{in,k} &= \frac{\mathcal{R}_k}{R_i}\tau_k + \mathcal{R}_k\varphi_{in,k} + \mathcal{R}_k\left(\frac{1}{R_{out,k}} + \frac{\zeta_{W,k}}{R_W}\right)T_{out,k} + \frac{\mathcal{R}_k\zeta_{D,k}}{R_D}T_{n,k} \qquad \text{(IV.20)} \\
\Delta_k^T &= T_{in,k} - T_{in,k}^{sensor} \\
& ivar : R_i = ., R_W = ., R_D = ., C_i = . \\
& R_{out,k} = .\forall, \varphi_{in,k} = .\forall, T_{out,k} = .\forall, T_{n,k} = .\forall, \zeta_{D,k} = .\forall, \zeta_{W,k} = .\forall
\end{aligned}
$$

Le modèle d'estimation paramétrique est dérivé du modèle de simulation auquel il a été rajouté de l'information concernant l'objectif à minimiser.

# IV.4 Conclusion

Dans ce chapitre, nous avons évoqué les différentes applications intervenant dans la gestion énergétique de bâtiments au sens large : la gestion anticipative, la simulation, l'estimation de grandeurs physiques et l'estimation paramétrique. La gestion anticipative désigne uniquement la gestion à travers l'optimisation PLNE. Il est évident que d'autres démarches et d'autre méthodes d'optimisation existent. Nous ne discutons pas de l'efficacité de celle choisie dans ce document mais nous discutons d'un outil d'aide à la transformation de modèles.

L'exemple étudié a mis en évidence une partie commune à tous les modèles qui se confond avec le modèle descriptif. Nous avons vu qu'il était possible de transformer, moyennant des compléments d'informations, ce modèle descriptif vers différents modèles applicatifs. Le modèle descriptif peut donc servir de pivot pour les différentes transformations à effectuer.

La transformation pour une application est composée d'étapes d'enrichissement en information et d'étapes de manipulations. Nous avons retracé ces différentes étapes. L'exemple étudié a pu être résolu manuellement car c'est un modèle très simple or, une description

d'un habitat peut comporter une multitude d'éléments à modéliser. Elle change d'un habitat à l'autre.

Il existe des outils pour réaliser certaines manipulations et transformations tels que les CAS que nous avons utilisés pour le calcul formel mais l'utilisation de ces seuls outils n'est pas suffisante et il n'est pas aisé d'enchaîner l'ensemble des transformations qui passent par des manipulations symboliques d'équations, de la linéarisation, de la discrétisation... La question que nous nous sommes posés après avoir eu à développer un modèle très complexe dans le cadre du projet CANOPEA est : *est-il possible de faciliter la procédure ? De la systématiser ? même en partie*. En d'autres termes, nous posons la question de l'automatisation du processus de production de modèles d'application. Un tel système rendrait d'autant plus viable une industrialisation d'une solution de gestion énergétique complète jusque-là gourmande en temps de travail d'experts.

# Chapitre V

# Vers un outil de modélisation multi-application

Le travail de fond mené dans cette thèse est la recherche de solutions et d'outils pour rendre viable le développement d'un gestionnaire énergétique à large échelle. Pour ce faire, la première partie de cette thèse a été consacrée à l'évolution de la solution GhomeTech qui n'était qu'à l'étape de preuve scientifique en un outil capable de résoudre des problèmes réels très complexes. La complexité de développement et de gestion des modèles nous ont mené à proposer une solution pour accompagner les développeurs de modèles dans cette tâche : l'outil Milp-workshop. Cette solution a permis de développer les modèles nécessaires pour le projet CANOPEA mais il nous est apparu un nouveau problème lié à la portabilité des modèles vers d'autres applications que la gestion énergétique par anticipation. Le prototype du gestionnaire énergétique de CANOPEA devait en effet inclure trois couches de gestion qu'étaient la couche anticipative, la couche réactive et la couche terrain. Les deux premières couches nécessitent des modèles différents. Le déploiement du BEMS nécessite une étape d'identification et de calibrage des paramètres utilisés en aval. De plus, le BEMS incluait un générateur de conseils sur la base des modèles d'optimisation ainsi que du calcul énergétique interactif sur la base de modèles de simulation. L'ensemble est regroupé dans une interface interactive. Chacune des applications citées nécessite une forme particulière de modèles issue d'une base commune afin d'avoir des résultats cohérents entre applications.

Le chapitre précédent a illustré par un exemple simple les différents besoins de modélisation de différentes applications liées à la gestion énergétique au sens large ainsi que les processus à systématiser pour y parvenir. Ce chapitre explore la possibilité d'automatisation des processus de transformation afin de faciliter le passage d'un modèle descriptif à un modèle applicatif.

Dans ce chapitre, nous présentons une formalisation du problème à partir de l'exemple traité dans le chapitre précédent. Nous effectuons une revue de la littérature à travers des outils se rapprochant du problème traité dans d'autres domaines. Nous proposons une solution au problème posé en nous appuyant sur les principes de l'ingénierie dirigée par les modèles (MDE pour Model Driven Engineering) empruntés au génie logiciel. Nous accompagnerons cette solution par l'approfondissement de chaque concept, des mécanismes et des procédés qui permettront un traitement automatique de certaines tâches du processus de transformation. Dans la même partie, nous proposerons des procédés permettant le déroulement des transformations depuis le modèle descriptif vers des modèles d'optimisation, de simulation et d'identification. Nous conclurons ce chapitre par une discussion

sur les limites constatées de cette approche avant de présenter un cas d'étude concret dans le chapitre suivant.

## V.1 Problématique scientifique

La problématique traitée dans cette seconde partie se rapporte aux applications d'estimations paramétriques et de simulation qui, toutes deux, se rapportent à des modèles liés à ceux de la gestion énergétique. Nous allons analyser ce qui est commun à tous ces modèles et étudier les mécanismes de transformation automatique de modèles applicatifs.

Les questions que nous nous sommes posées sont :

— Quel est le lien entre ces différentes applications ?
— Comment minimiser l'intervention d'un expert pour la configuration de ces différentes applications ?
— Qu'est-ce qui est systématisable (ou automatisable) et qu'est-ce qui ne l'est pas ?

Ce questionnement est animé par deux principales motivations :

— la recherche d'un éco-système de modélisation plus cohérent
— la recherche de concept de transformation de modèles conduisant à des économies en temps de développement.

Les travaux liés à l'application Canopea répondent en partie à la première question concernant les liens entre applications de manipulation et de conception. Les modèles développés dans Milp-workshop constituent le socle de travail car ces modèles servent à composer le problème d'optimisation. Ce problème d'optimisation est résolu dans le cadre de la couche prédictive. Avant la résolution du problème, les paramètres de celui-ci doivent être trouvés à partir de données remontées des capteurs de la couche terrain. D'un autre coté, les résultats de l'optimisation de la couche anticipative sont soumis à l'approbation de l'utilisateur. Pour ce faire, une application de calcul énergétique interactive a été proposée. Elle s'appuie sur des modèles de simulation afin de visualiser l'effet d'actions sur le système bâtiment. Une optimisation rapide mais sans garantie d'optimalité est aussi proposée pour ajuster les stratégies anticipatives lors d'ajouts de contraintes supplémentaires par un utilisateur. Dans ce cas, nous avons proposé d'utiliser une optimisation dont le temps de calcul est maîtrisable a priori : l'algorithme d'optimisation de recuit simulé que nous expliquerons dans le chapitre suivant.

A travers l'exemple du chapitre 5, nous avons retranscrit les principales étapes que nous avons parcourues pour proposer les modèles nécessaires à l'ensemble des applications citées dans la figureV.1.

Le problème posé dans ce chapitre concerne la possibilité de mise en place de procédures pouvant automatiser tout ou une partie des processus de transformation. Pour ce faire, nous avons trié les modèles en deux catégories principales : les modèles applicatifs et les modèles descriptifs, ces derniers étant indépendants de toute application spécifique. Chaque application nécessite un format spécifique et des données particulières pour constituer un modèle valide. L'ensemble de ces exigences est décrit dans un *objectif de transformation*. L'objectif est une méta-description du modèle applicatif. Le modèle descriptif est considéré comme un modèle source pour l'ensemble des modèles applicatifs. Le processus de transformation nécessite parfois des informations supplémentaires. Ces informations peuvent être générées par des règles automatisées. Dans certains cas, l'information ne peut être générée automatiquement et nécessite une intervention humaine.

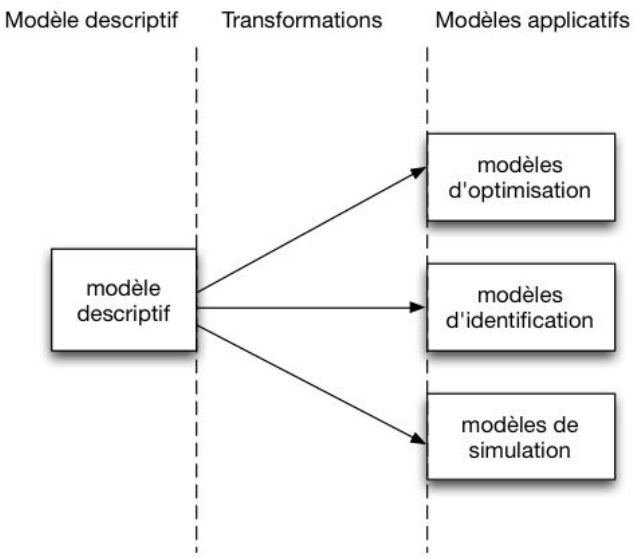



FIGURE *V.1 – schéma de transformations*

## V.2 Eléments de réponse

La littérature scientifique fait référence à des problèmes de modélisation multi-facette et des solutions sont proposées. Néanmoins, il faut distinguer les solutions de portage entre environnements qui consistent à modifier la syntaxe et la structure de modèles, des solutions multi-applicatives où la nature des variables d'un modèle change, d'un modèle acausal pour la gestion optimisée à un modèle causal pour la simulation par exemple. Un même symbole peut être tantôt associé à une série temporelle (entrée), à une valeur (paramètre) ou déduit (sortie).

Distinguons plus en détails les différentes solutions et ce que nous avons l'ambition d'apporter. Pour cela, nous clarifions certains éléments de langage que nous emploierons dans ce qui suivra. Une norme est une ou plusieurs règles à suivre dans la mise en place d'un système normé. Un protocole est un ensemble de règles à suivre dans la transmission d'informations. Un logiciel que nous pourrions aussi évoquer sous l'appellation 'outil' est une implémentation informatique d'un concept fournissant un service. Un langage est un moyen de communication entre émetteur et récepteur. Chacun des deux cotés peut être humain ou machine. Une plateforme ou environnement de développement est une suite logicielle optimisée qui est sensée fournir l'ensemble des outils nécessaires au développement d'une application.

Dans la simulation, des plateformes intégrées tel que Matlab a son langage propre ou Dymola qui utilise le langage Modelica offrent des passerelles entre leur format "propriétaire" respectif afin de rendre certaines applications d'inter-opérabilité possibles. Des normes indépendantes (des éditeurs d'outils) d'inter-opérabilité se sont développées afin de permettre plus de portabilité pour les outils de simulation tels que les FMU/FMI (Functional Mockup Interface) qui seront détaillés plus loin. Or, le choix d'outils intégrés dans les plateformes se limite souvent à un seul métier comme c'est le cas dans l'automobile avec le principe de développement collaboratif entre acteurs : les FMU/FMI.

### V.2.1 Plateformes de modélisation haut niveau pour l'optimisation avancée

Les AML, pour Algebric Modeling Language, sont des langages de modélisation pour les problèmes d'optimisation associés. Chaque AML est associé à un outil de gestion de modèles qui permet l'inter-opérabilité entre outils d'optimisation. Nous parlons de plateforme pour l'ensemble associant chaque langage à son outil dédié. Certaines plateformes disposent d'un éditeur de modèles et d'interfaces graphiques dédiées, tel que GAMS, tandis que d'autres s'appuient sur des éditeurs externes tout en imposant le langage de modélisation, lequel est associé à un format de fichiers modèles. Les plateformes les plus populaires sont AIMMS, AMPL IDE et GAMS. Ces plateformes sont utilisées pour formuler des problèmes d'optimisation dans un langage haut niveau proche des écritures mathématiques de modélisation. Elles ont été conçues pour répondre à deux exigences principales :

— disposer d'un langage ergonomique pour la modélisation de problèmes.
— permettre d'avoir recours à plusieurs algorithmes de résolution pour les problèmes d'optimisation sans avoir à re-développer les modèles pour chacune des applications.

GAMS est parmi les premières plateformes à avoir été développées. Celle-ci a été imaginée par l'institut de recherche et de prospectives de la banque mondiale. Ils ont cherché à répondre au besoin d'une plateforme avec un langage de modélisation proche des formulations mathématiques théoriques pour permettre le développement de modèles complexes associant modèles agricoles, modèle d'accroissement des populations, modèles d'évolution d'industries, etc . . .La réponse à la problématique fût la mise en place d'un paradigme dissociant la structure du modèle des données **bussieck2004general brooke1998general** afin que chaque partie puisse être manipulée séparément. Ceci à travers le langage notamment.

Les autres plateformes que sont AIMMS et AMPL IDE (FOURER et al. 2004 ; GAY 2014) ont été initiées plus tard avec les mêmes objectifs mais avec des approches ergonomiques différentes, chacune adaptée à un domaine d'application particulier. Les plateformes AML n'embarquent pas d'algorithmes de résolution propres. Celles-ci s'appuient sur les algorithmes existants embarqués dans les outils de résolution dédiés tel que CPLEX, GLPK, CHOCO, etc...

Les plateformes AML sont chargées de la traduction de modèles d'optimisation depuis leur langage haut niveau, celui qu'utilise le développeur de modèles, vers un langage machine pour chacun des outils d'optimisation supporté. Les outils d'optimisation disposent de leurs propres langages plus bas niveau que les langages AML. Chacune de ces plateformes est associée à un format de fichier de modèles. Les outils d'optimisation disposent, en plus, d'une API qui sert de prise de communication avec d'autres applications. Les plateformes AML utilisent ces API pour communiquer les modèles aux outils d'optimisation inter-opérables.

Les plateformes AML ont apporté les deux principaux avantages :

— L'ergonomie dans la modélisation à travers un langage inspiré des écritures mathématiques théoriques.
— L'universalité des modèles développés par rapport aux outils d'optimisation.

Néanmoins, les évolutions continues des outils d'optimisation et le support des principaux langages de modélisation bas niveau remettent en cause de plus en plus l'universalité revendiquée par les AML (voir tableau V.2). Des formats standardisés de modélisation tel

que "LP" ou encore "QCP" s'imposent car les développeurs d'outils d'optimisation s'efforcent de les intégrer dans leur gamme de formats supportés comme indiqué dans le tableau V.2. Le même tableau donne une visibilité globale des algorithmes supportés dans GAMS (jusqu'en septembre 2015) associée aux formats supportés par chacun des outils d'optimisation.


1/10/2015 GAMS Documentation Center

| Solver/Model type availability - 24.5 | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LP | MIP | NLP | MCP | MPEC | CNS | DNLP | MINLP | QCP | MIQCP | *Stoch.* | *Global* |
| ALPHAECP | | | | | | | | ✔ | | ✔ | | |
| ANTIGONE 1.1 | | | ✔ | | | ✔ | ✔ | ✔ | ✔ | ✔ | | ✔* |
| BARON 15.9 | ✔ | ✔ | ✔ | | | ✔ | ✔ | ✔ | ✔ | ✔ | | ✔* |
| BDMLP | ✔ | ✔ | | | | | | | | | | |
| BONMIN 1.8 | | | | | | | | ✔ | | ✔ | | |
| CBC 2.9 | ✔ | ✔ | | | | | | | | | | |
| CONOPT 3 | ✔ | | ✔ | | | ✔ | ✔ | | ✔ | | | |
| COUENNE 0.5 | | | ✔ | | | ✔ | ✔ | ✔ | ✔ | ✔ | | ✔* |
| CPLEX 12.6 | ✔ | ✔ | | | | | | | ✔ | ✔ | | |
| DECIS | ✔ | | | | | | | | | | ✔ | |
| DICOPT | | | | | | | | ✔ | | ✔ | | |
| GLOMIQO 2.3 | | | | | | | | | ✔ | ✔ | | ✔* |
| GUROBI 6.0 | ✔ | ✔ | | | | | | | ✔ | ✔ | | |
| GUSS | ✔ | ✔ | ✔ | ✔ | | ✔ | ✔ | ✔ | ✔ | ✔ | | |
| IPOPT 3.12 | ✔ | | ✔ | | | ✔ | ✔ | | ✔ | | | |
| KESTREL | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | | |
| KNITRO 9.1 | ✔ | | ✔ | | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | | |
| LGO | ✔ | | ✔ | | | | ✔ | | ✔ | | | ✔ |
| LINDO 9.0 | ✔ | ✔ | ✔ | | | | ✔ | ✔ | ✔ | ✔ | ✔ | ✔* |
| LINDOGLOBAL 9.0 | ✔ | ✔ | ✔ | | | | ✔ | ✔ | ✔ | ✔ | | ✔* |
| LOCALSOLVER 5.5 | | ✔ | ✔ | | | ✔ | ✔ | ✔ | ✔ | ✔ | | |
| MILES | | | | ✔ | | | | | | | | |
| MINOS | ✔ | | ✔ | | | ✔ | ✔ | | ✔ | | | |
| MOSEK 7 | ✔ | ✔ | ✔ | | | | ✔ | | ✔ | ✔ | | |
| MSNLP | | | ✔ | | | | ✔ | | ✔ | | | ✔ |
| NLPEC | | | | ✔ | ✔ | | | | | | | |
| OQNLP | | | ✔ | | | | ✔ | ✔ | ✔ | ✔ | | ✔ |
| PATH | | | | ✔ | | ✔ | | | | | | |
| SBB | | | | | | | | ✔ | | ✔ | | |
| SCIP 3.2 | | ✔ | ✔ | | | ✔ | ✔ | ✔ | ✔ | ✔ | | ✔* |
| SNOPT | ✔ | | ✔ | | | ✔ | ✔ | | ✔ | | | |
| SOPLEX 2.2 | ✔ | | | | | | | | | | | |
| SULUM 4.3 | ✔ | ✔ | | | | | | | | | | |
| XA | ✔ | ✔ | | | | | | | | | | |
| XPRESS 28.01 | ✔ | ✔ | | | | | | | ✔ | ✔ | | |

data:text/html;charset=utf-8,%3Ctable%20bgcolor%3D%22%23f5f5f5%22%20border%3D%220%22%20cellpadding%3D%220%22%20cellspacing%3D%220… 1/1

FIGURE *V.2 – interopérabilité GAMS (*No Title*)*

A travers la revue des plateformes AML, nous avons voulu évaluer l'approche qui consiste à concevoir un modèle le plus exhaustif possible, le modèle principal, puis de permettre à ce modèle d'être utilisé par différents algorithmes d'optimisation. Les solutions s'appuyant sur des AML ont été développées pour la seule application 'optimisation'. Cette approche a séduit de par son langage haut niveau de modélisation et pour

l'universalité des modèles développés. L'approche nous a intéressé dans son principe de transformation. Les modèles principaux sont transformés automatiquement de façon à être exécutés par les différents algorithmes d'optimisation. Cette automatisation du procédé est facilitée par l'unique champ d'application : l'optimisation. L'ensemble des informations nécessaires aux modèles exécutables est contenu dans le modèle principal. Ainsi, les transformations ont pu être automatisées à partir du modèle principal vers le modèle exécutable pour chaque outil de résolution. L'optimisation obéit à un schéma de modélisation commun que nous retrouvons dans le modèle principal et les modèles exécutables :

— déclaration des variables et des paramètres
— déclaration des équations et des contraintes
— introduction de l'objectif à minimiser ou à maximiser
— instanciation des paramètres du problème à résoudre

La plateforme se charge alors de mettre au format, pour chaque algorithme, la syntaxe et l'organisation du problème à partir du modèle principal sans intervention manuelle.

Par rapport à notre problématique scientifique, le problème traité par les AML est partiel car il n'adresse que les transformations pour l'optimisation, c'est-à-dire un seul type d'application. La nature d'un symbole ne change pas, le problème de causalité n'est jamais abordé. Nous qualifions alors les plateformes à base d'AML de mono-applicatives alors que notre problématique est multi-applicative.

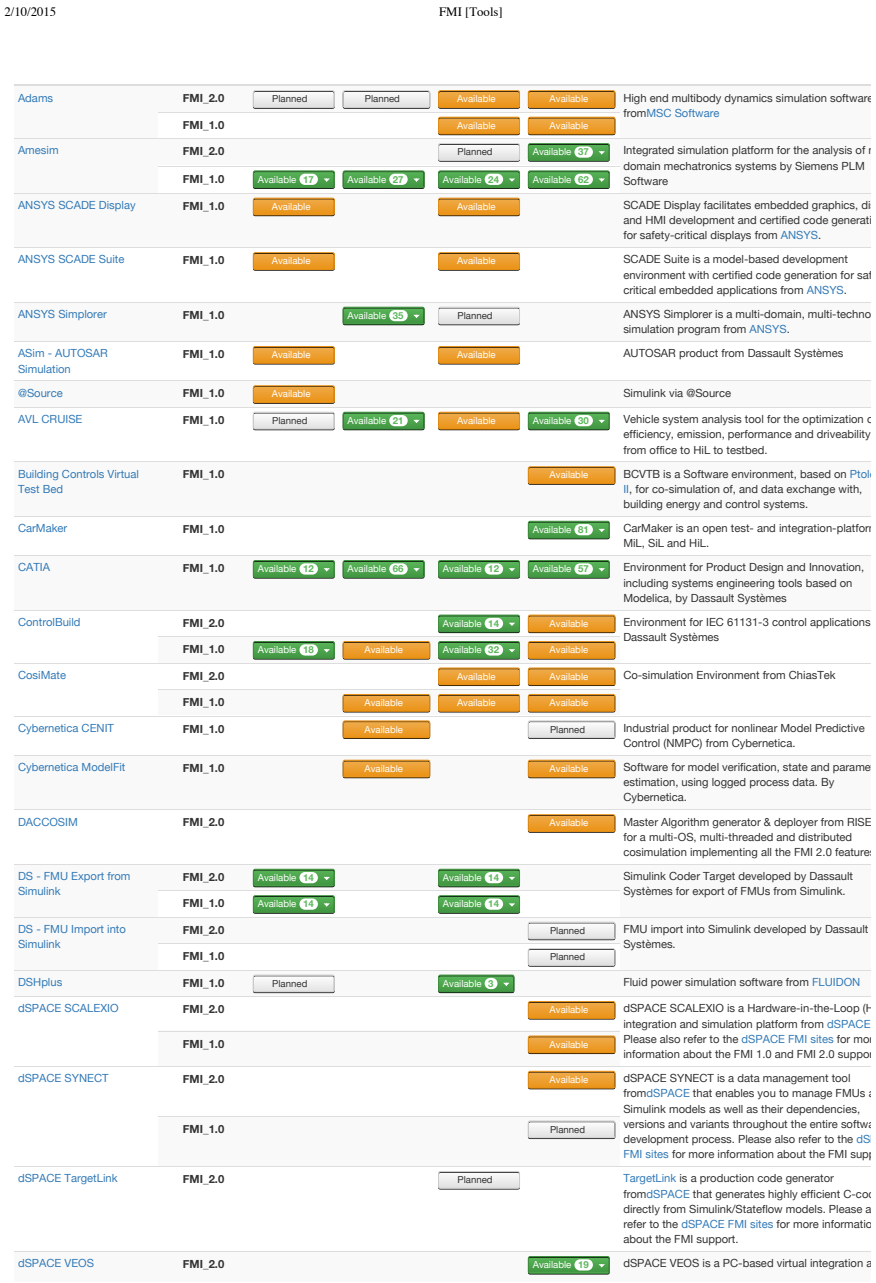
### V.2.2 L'implémentation de la norme FMI/FMU

La norme FMI, pour (Functional Mockup Interface), est un ensemble de règles pour l'implémentation d'interfaces d'interaction avec des capsules de modèles appelées composants logiciels FMU pour (Functional Mockup unit). Cette norme a été initiée par l'industrie automobile. Daimler a formulé le besoin d'une solution pour l'échange de modèles de simulation entre les différents acteurs de l'industrie automobile que sont les constructeurs, les équipementiers et les instituts de R & D. Pour établir le standard, les exigences suivantes ont été posées :

— besoin d'un moyen d'échanges de modèles entre simulateurs
— besoin d'un moyen d'échange sécurisé avec graduation de l'accès au savoir faire nécessaire à l'élaboration des modèles échangés
— besoin d'un moyen de co-simulation.
— besoin d'un système évoluant sur les principaux outils de simulation du standard Modelica.

La réponse formalisée par la norme FMI (BLOCHWITZ et al. 2011) est une interface pour une capsule de modèle appelé FMU (Function Mockup Unit). Le FMU est un fichier ZIP qui contient un fichier descriptif XML et un fichier source en langage C. Le fichier XML décrit l'ensemble des variables intervenant dans le modèle embarqué. Le type, l'unité, la variabilité, la causalité ainsi que des attributs du modèle et son exécution, tels que les headers C du FMU, y sont renseignés ("Functional Mockup Interface 2 . 0 : The Standard for Tool independent Exchange of Simulation Models"). Le fichier source C constitue l'exécutable du FMU. Celui-ci embarque le modèle sous forme de fonctions C qui s'exécutent en même temps que d'autres modèles natifs dans l'application cible moyennant un plug-in. Le plug-in chargé de faire ce traitement est développé pour chaque outil de simulation afin qu'il puisse procéder à l'import et à l'export des modèles. Une série d'outils de simulation ont vu leur plug-in développé par les fournisseurs de ces mêmes outils pour les rendre compatibles FMI et qui y voient ainsi une valeur

ajoutée à leur produit, ou alors par des tiers dans le cas de simulateurs ouverts comme énumérés dans le tableau V.3.



2/10/2015 FMI [Tools]

| | | | | | | |
|---|---|---|---|---|---|---|
| Adams | FMI_2.0 | Planned | Planned | Available | Available | High end multibody dynamics simulation software fromMSC Software |
| | FMI_1.0 | | | Available | Available | |
| Amesim | FMI_2.0 | | | Planned | Available 37 | Integrated simulation platform for the analysis of r domain mechatronics systems by Siemens PLM Software |
| | FMI_1.0 | Available 17 | Available 27 | Available 24 | Available 62 | |
| ANSYS SCADE Display | FMI_1.0 | Available | | Available | | SCADE Display facilitates embedded graphics, di and HMI development and certified code generati for safety-critical displays from ANSYS. |
| ANSYS SCADE Suite | FMI_1.0 | Available | | Available | | SCADE Suite is a model-based development environment with certified code generation for saf critical embedded applications from ANSYS. |
| ANSYS Simplorer | FMI_1.0 | | Available 35 | Planned | | ANSYS Simplorer is a multi-domain, multi-techno simulation program from ANSYS. |
| ASim - AUTOSAR Simulation | FMI_1.0 | Available | | Available | | AUTOSAR product from Dassault Systèmes |
| @Source | FMI_1.0 | Available | | | | Simulink via @Source |
| AVL CRUISE | FMI_1.0 | Planned | Available 21 | Available | Available 30 | Vehicle system analysis tool for the optimization c efficiency, emission, performance and driveability from office to HiL to testbed. |
| Building Controls Virtual Test Bed | FMI_1.0 | | | | Available | BCVTB is a Software environment, based on Ptol II, for co-simulation of, and data exchange with, building energy and control systems. |
| CarMaker | FMI_1.0 | | | | Available 81 | CarMaker is an open test- and integration-platfor MiL, SiL and HiL. |
| CATIA | FMI_1.0 | Available 12 | Available 66 | Available 12 | Available 57 | Environment for Product Design and Innovation, including systems engineering tools based on Modelica, by Dassault Systèmes |
| ControlBuild | FMI_2.0 | | | Available 14 | Available | Environment for IEC 61131-3 control applications Dassault Systèmes |
| | FMI_1.0 | Available 18 | Available | Available 32 | Available | |
| CosiMate | FMI_2.0 | | | Available | Available | Co-simulation Environment from ChiasTek |
| | FMI_1.0 | | Available | Available | Available | |
| Cybernetica CENIT | FMI_1.0 | | Available | | Planned | Industrial product for nonlinear Model Predictive Control (NMPC) from Cybernetica. |
| Cybernetica ModelFit | FMI_1.0 | | Available | | Available | Software for model verification, state and parame estimation, using logged process data. By Cybernetica. |
| DACCOSIM | FMI_2.0 | | | | Available | Master Algorithm generator & deployer from RISE for a multi-OS, multi-threaded and distributed cosimulation implementing all the FMI 2.0 feature |
| DS - FMU Export from Simulink | FMI_2.0 | Available 14 | | Available 14 | | Simulink Coder Target developed by Dassault Systèmes for export of FMUs from Simulink. |
| | FMI_1.0 | Available 14 | | Available 14 | | |
| DS - FMU Import into Simulink | FMI_2.0 | | | | Planned | FMU import into Simulink developed by Dassault Systèmes. |
| | FMI_1.0 | | | | Planned | |
| DSHplus | FMI_1.0 | Planned | | Available 3 | | Fluid power simulation software from FLUIDON |
| dSPACE SCALEXIO | FMI_2.0 | | | | Available | dSPACE SCALEXIO is a Hardware-in-the-Loop (H integration and simulation platform from dSPACE Please also refer to the dSPACE FMI sites for mo information about the FMI 1.0 and FMI 2.0 suppo |
| | FMI_1.0 | | | | Available | |
| dSPACE SYNECT | FMI_2.0 | | | | Available | dSPACE SYNECT is a data management tool fromdSPACE that enables you to manage FMUs a Simulink models as well as their dependencies, versions and variants throughout the entire softwa development process. Please also refer to the dS FMI sites for more information about the FMI sup |
| | FMI_1.0 | | | | Planned | |
| dSPACE TargetLink | FMI_2.0 | | | Planned | | TargetLink is a production code generator fromdSPACE that generates highly efficient C-co directly from Simulink/Stateflow models. Please a refer to the dSPACE FMI sites for more informatio about the FMI support. |
| dSPACE VEOS | FMI_2.0 | | | | Available 19 | dSPACE VEOS is a PC-based virtual integration a |

data:text/html;charset=utf-8,%3Ctable%20class%3D%22table%20table-striped%20table-condensed%22%20style%3D%22box-sizing%3A%20border-box%3B… 1/4

FIGURE *V.3 – Disponibilité du plug-in FMI dans les simulateurs*

L'approche FMI offre aussi un moyen de couplage faible entre applications : la co-simulation. Il s'agit d'exécuter deux simulations sur des simulateurs différents en parallèle suivant un schéma maître/esclave ou alors suivant un schéma distribué.

Le portage de modèles sous le standard FMI a été intégré dans cette revue de littérature car il met en évidence une approche de portage intéressante : les composants logiciels FMU. Néanmoins, ce portage est limité à une seule application : la simulation. Cette solution ne répond clairement pas à l'exigence de multi-application de notre problématique. Reste que l'idée des composants logiciels pour porter des modèles a inspiré les autres

démarches qui vont être abordées.

### V.2.3 L'implémentation de la norme Icar

La norme Icar a été développée au sein du laboratoire G2Elab pour des problématiques de simulations électrotechniques dans un premier temps (FISCHER et al. 2004). Cette norme est basée sur l'approche plug-in / plug-out de composants logiciels. (CHOUIKH 2012) a adapté le standard Icar à la problématique bâtiment. L'approche développée consiste à dissocier l'étape de modélisation de l'étape d'exécution du modèle pour un ensemble de simulateurs. Voici un exemple emprunté à (CHOUIKH 2012) pour illustrer le problème traité.

Il s'agit d'importer un modèle d'enveloppe thermique issu de COMFIE dans Dymola et de le coupler au modèle de la VMC développé en Modelica. L'ensemble sera ensuite importé dans Matlab / Simulink afin de tester différentes stratégies de contrôle-commande. Le tout pourra être de surcroît couplé à un modèle d'occupants développé en Brahms.

Le composant Icar permet d'encapsuler le modèle de l'enveloppe thermique à travers le plug-out figure V.4 installé sur COMFIE qui traduit le modèle dans un langage machine. Le composant logiciel est ensuite combiné au modèle de la VMC sous Dymola via le plug-in dédié figure V.4 pour constituer une entité autonome, laquelle sera à son tour encapsulée dans un autre composant Icar via le plug-out de Dymola. Ce nouveau composant sera associé à Brahms.

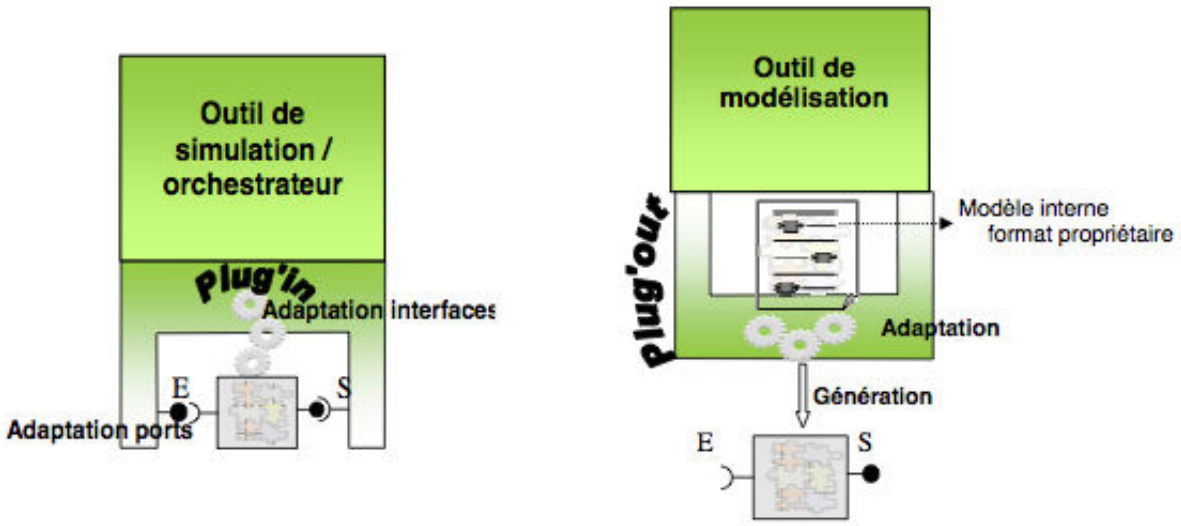



FIGURE *V.4 – plug-in et plug-out Icar*

Chaque composant Icar devient une entité indépendante qui interagit avec son environnement à travers la notion de service. Ainsi, l'environnement dans lequel le composant Icar est inséré envoie des requêtes de calcul au composant Icar et prend les résultats du calcul en retour. C'est une approche dite boite noire car le modèle n'est pas visible dans l'environnement exécutant.

Plus généralement, un bus à composants a été mis en place pour un corpus d'outils afin qu'ils puissent interagir entre eux figure V.5.

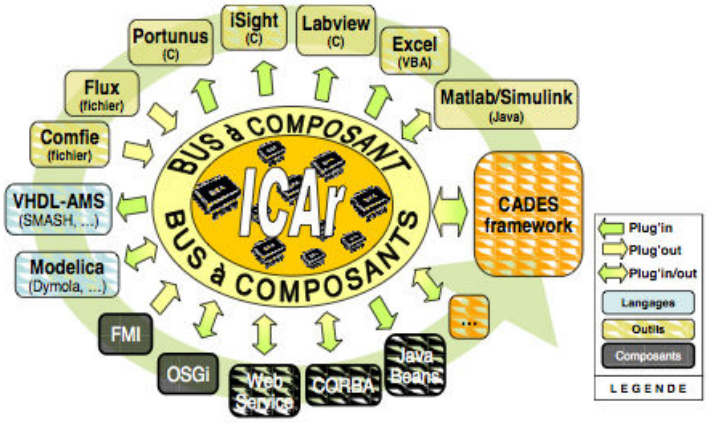



FIGURE *V.5 – Disponibilité des plug-in/plug-out Icar*

Cette approche répond bien à la problématique du portage d'un modèle écrit dans un langage dédié à une application vers une autre application. Néanmoins, la difficulté est dans sa mise en pratique car nous remarquons que les plug-ins et plug-outs developpés jusqu'ici le sont principalement pour des outils de simulation/commande mais le principal outil faisant appel à des algorithmes d'optimisation, CADES **delinchant2012cades** est plus complexe à mettre en oeuvre. La transformation du modèle de simulation vers un modèle d'optimisation et vice versa pose un problème car la nature des symboles utilisés en modélisation est différente. Notre analyse par rapport à cette limitation dans l'approche réside dans le principe même d'usage de langages de modélisation d'applications établies pour l'élaboration de modèles supposés universels. Selon le théorème d'incomplétude de Godël **godel2009formally** il est impossible de construire un langage formel, consistant et complet pour tous les types de simulations physiques. Ce théorème se base sur le contenu des modèles et peut être vu sous l'angle des différentes applications physiques. D'autre part, ce système a des limites car les modèles ne sont plus visibles une fois encapsulés dans l'Icar, ils ne sont plus modifiables à cause du caractère irréversible de l'encapsulation : la causalité est ainsi figée ce qui limite le nombre d'applications pouvant exploiter un tel modèle. D'un point de vue ergonomique, cette encapsulation est problématique dans le travail du développeur de modèles car s'il y a un bug, celui ci est complètement caché.

### V.2.4 La plateforme MAEL-HAGEL-RAMA

MAEL pour Module for Analysis of Equations and Logic, HAGEL pour Hybrid Automaton for Generation of Equation and Logic et RAMA pour (Rule Applicator for Mathematic Analysis) sont trois outils complémentaires développés par Loig Allain (ALLAIN 2003) et son équipe.

Ce travail a porté sur la problématique du dimensionnement et de simulation associée en électromagnétique et plus largement en électrotechnique. MAEL a été la solution proposée au problème en implémentant le principe de boules et de boites. Ce principe consiste à considérer un modèle a-contextualisé comme étant une boule, le modèle est à ce stade non orienté au sens (entrées, sorties, paramètres, variables internes). Selon que le modèle sera utilisé pour le dimensionnement ou pour la simulation dans un contexte imposé. Ce contexte oriente le modèle auquel il faut ensuite trouver un ordonnancement causal pour la résolution des équations composant le problème : c'est la détermination de la séquence de calcul. Une partie des étapes menant du modèle boule au modèle boite a

été automatisée, comme montré dans le schéma V.6, tandis que la partie apport en information reste à la charge du concepteur de modèle.

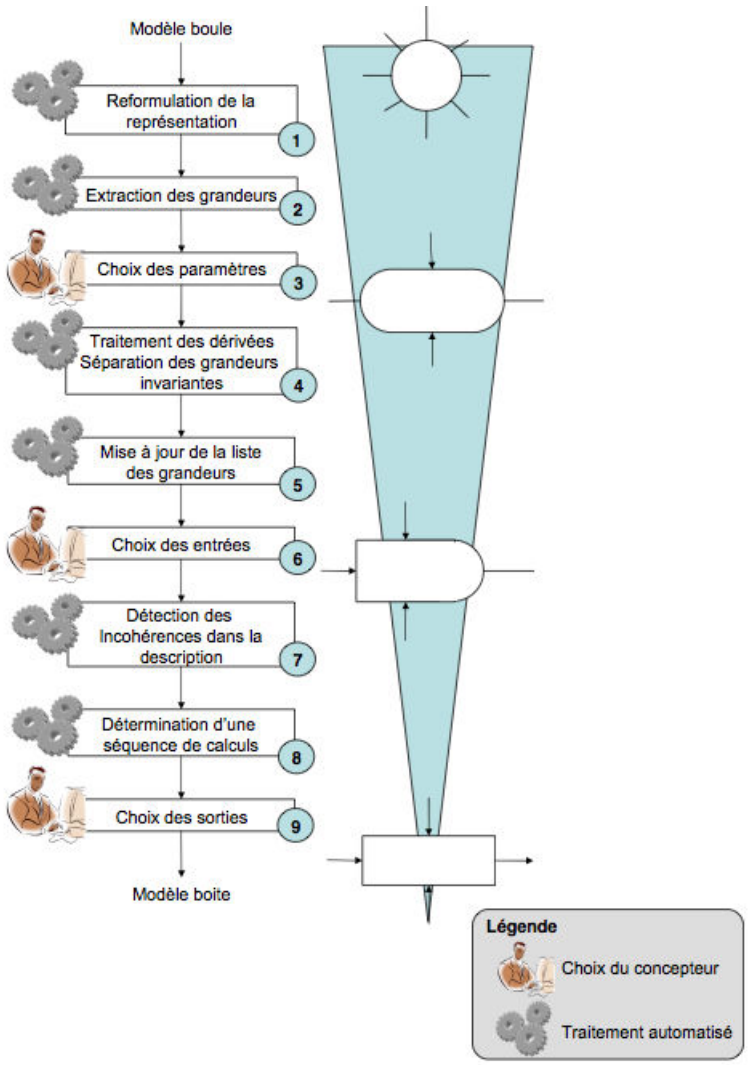



FIGURE *V.6 – Processus de transformation d'une boule en boite*

Les manipulations algébriques nécessaires dans la transformation du modèle boule en modèle boite ont été implémentées dans l'outil RAMA allégé par rapport aux outils commerciaux Mathematica ou Maple. HAGEL est la suite de MAEL qui produit le code informatique correspondant au modèle en vue de sa simulation.

L'approche est très intéressante dans le cadre de nos besoins sur plusieurs points :

**transparence :** L'approche est dite aussi boite blanche car le modèle n'est embarqué sur l'application de son exécution que dans un second temps après avoir été orienté et transformé.

**Semi-automatisée :** La transformation est semi-automatisée et peut donc être supervisée par l'expert. Cette approche permet de plus larges possibilités de prise en compte de singularités et d'aspects particuliers non automatisables. C'est un compromis entre le besoin de rapidité et d'exhaustivité qui est intéressant à souligner. De plus, le scénario est intégré en alternance avec des processus automatiques. Cette approche permet à l'expert de vérifier, d'affiner et de modifier si nécessaire

le scénario choisi à des étapes de transformation plus ou moins avancées. Cette interaction est très intéressante d'un point de vue ergonomique et donc d'efficacité dans le travail de l'expert.

**Prise en compte de différents cas de simulation :** C'est une approche multi-scénarios, aussi appelée modélisation non orientée car la boule n'intègre pas d'orientation pour les variables du modèle. Cette information complémentaire est fournie par le scénario qui regroupe le choix des paramètres, des entrées et des sorties.

Ce dernier point est aussi la limite par rapport à nos besoins. L'approche n'est pas multi-application car le rayon d'action des modèles développés se limite à la seule simulation. Les variables ne peuvent être définies soit comme entrée, paramètre ou sortie.

### V.2.5 Synthèse

Les trois approches étudiées présentent les travaux que nous avons sélectionnés pour leur lien avec notre problème. Le recours à un AML est une approche dite "boite blanche" car la transformation reste au niveau de la modélisation pour passer du modèle principal aux modèles exécutables. Cette approche est intéressante en théorie car elle permet une prise en main après la transformation et permet donc un contrôle voire des modifications a posteriori même si en pratique, dans GAMS et AMPL IDE cette fonctionnalité n'a pas été implémentée. Les modèles restent a-causaux et doivent être utilisés ainsi par les solveurs cibles.

Les standards FMI et Icar sont des approches composant logiciel dit "boites noires". Ce sont des approches basées sur la notion de service. Le modèle est encapsulé dans le composant logiciel et celui-ci répond à des requêtes émises par l'application l'utilisant sans être intégré comme un modèle natif. Cette approche nécessite des modèles complètement causaux à défaut de quoi, le solveur doit être lui même embarqué dans le composant Icar. D'autre part, certains plug-out ne sont tout simplement pas possibles à réaliser car l'information contenue dans le modèle de départ est insuffisante pour confectionner le modèle d'arrivée. L'automatisation entière du processus trouve donc ses limites.

MAEL répond à ces problèmes en offrant une plateforme transparente pour le passage d'un modèle boule à un modèle boite. C'est une projection d'un modèle a-contextualisé vers un modèle de simulation destiné à un usage particulier : dimensionnement ou simulation. Ce passage se fait à travers des étapes automatisées et des étapes manuelles car il y a apport d'information dans le processus. De cette manière, il est possible à l'utilisateur de modifier son modèle à n'importe quelle étape.

Notre proposition s'inspire de l'approche MAEL dans sa mise en oeuvre tout en répondant à la problématique multi-applications. La notion de boule résume une méthode de constitution du corps de l'information nécessaire à l'utilisation d'un modèle dans différents cas. MAEL traite les différents cas en simulation à travers la variation de scénarios qui impliquent des spécialisations de symboles en entrée, sortie ou paramètre, lesquelles sont de même nature dans la boule. Dans notre problème, les symboles peuvent avoir des natures complètement différentes selon les applications. Outre la nature entrée, sortie ou paramètres pour la simulation, un symbole peut être paramètre ou variable de décision en optimisation PLNE. Un symbole peut être aussi entrée ou sortie ou sortie étalon en estimation paramétrique. Pour arriver à ces formes à partir de symboles, des informations supplémentaires sont nécessaires. Des traitement conjoints entre symboles et contraintes les comportant sont aussi nécessaires.

## V.3 Outil de formulation de la solution proposée : MDA

Nous avons choisi de nous appuyer sur les concepts de l'architecture dirigée par les modèles ou en anglais Model Driven Architecture (MDA) pour structurer notre proposition de solution. MDA est une architecture de modélisation dédiée au logiciel en le structurant en différents niveaux d'abstraction. Cette approche a été proposée pour intégrer différents métiers **miller2003mda** Cette architecture a été proposée par l'OMG (Object Management Group) (*No Title*) afin d'améliorer le portage des systèmes d'information d'une génération de technologie à une autre, d'un métier à l'autre moyennant des mécanismes de transformations qui évoquent notre problématique **bezivin2005unification atkinson2005generalized** La modélisation traité par le MDE (Model Driven Engineering), discipline usant du formalisme MDA, est quelque peu différente de la modélisation traitée dans le manuscrit. Néanmoins, MDE a des points en commun avec notre travail à travers l'approche de transformation que nous souhaitons développer et offre à ce titre un support intéressant à explorer. La modélisation en conception logicielle n'a pas toujours été pratiquée en raison du travail supplémentaire nécessaire augmentant les coûts à court terme. Le MDE propose une démarche qui permet d'économiser du volume de travail à moyen et long terme à travers la transformation automatique de modèles pour passer d'une technologie logicielle à une autre.

Le MDE repose sur des normes, des paradigmes, des langages et les outils associés tel que UML et ses éditeurs dédiés, la POO et ses langages de programmation tel que java **harmonmda** XML et ses schémas XSD **shen2005performance** La MDA est dite multi-niveaux d'abstraction. Le code informatique est le niveau le plus bas dans cette échelle mais en est exclu dans certaines descriptions (BLANC et al. 2011). Les niveaux au-dessus offrent une structuration de code. L'intêret de la MDA par rapport à d'autres approches d'abstraction est dans le passage d'un niveau à un autre appelé transformation. Les transformations sont automatisées afin de permettre des gains de temps lors des changements de technologies de développement de code.

Un niveau d'abstraction dans MDA représente un modèle d'une partie de l'information ou un point de vue nécessaire à l'élaboration du logiciel final. En terme d'information, le niveau le plus haut est le plus abstrait. Il contient donc moins d'information et, en contre partie, offre la plus large visibilité sur le modèle du logiciel. A contrario, le niveau d'abstraction le plus bas est le plus riche en terme d'information et est proche du code d'exécution final.

Trois niveaux d'abstraction sont définis en MDA **zhang2005transformation** (BLANC et al. 2011) :

**le modèle d'exigences CIM (Computation Independant Model) :** La première étape de tout développement logiciel est la définition de la spécification client. L'objectif est de créer un modèle d'exigence qui doit représenter la future application dans son environnement afin de définir quels sont les services offerts par l'application et quels sont les autres entités avec lesquelles elle interagirait. Ce modèle ne doit pas contenir d'éléments de réalisation ni de traitement. En langage UML, cette représentation peut se résumer à un diagramme de cas d'utilisation avec des acteurs. Plus largement, un modèle d'exigences est considéré comme une entité complexe constituée entre autres d'un glossaire, de définitions des processus métier, d'exigences et des cas d'utilisation ainsi que d'une vue systémique de l'application.

**le modèle d'analyse et de conception abstraite PIM (platform independant model) :** Par conception, il est entendu l'étape qui consiste à structurer l'application en mo-

dules et sous modules. L'application de patrons de conception (design patterns) fait partie de ce modèle. Seule la conception abstraite indépendante des techniques d'implémentation est posée dans ce niveau d'abstraction. Le rôle des modèles d'analyse et de conception est d'être pérenne et de faire le lien entre le modèle d'exigences et le code d'application. Ces modèles doivent être par ailleurs productifs puisqu'ils constituent le socle de tout le processus de génération de code défini par MDA. Un PIM doit être assez précis et contenir suffisamment d'information pour qu'une génération automatique de code soit envisageable.

**le modèle de code ou de conception concrète PSM (platform specific model) :** Une fois les modèles d'analyse et de conception finalisés, le travail de génération de code peut commencer. Cette phase, la plus délicate du MDA, doit elle aussi utiliser des modèles. Elle inclut l'application des patrons de conception technique. MDA considère que le code d'une application peut être facilement obtenu à partir de modèles de code. La différence principale entre un modèle de code et un modèles d'analyse ou de conception réside dans le fait que le modèles de code est lié à une plateforme d'exécution. Il existe une différence entre le code d'une application et un modèle de code. Un code d'application se résume à une suite de lignes textuelles tandis qu'un modèle de code est plutôt une représentation structurée incluant, par exemple, les concepts de boucle, condition, instruction, composant, événement, etc...

Concrètement, les transformations ne sont possibles qu'entre modèles structurés. L'OMG a proposé un standard de structuration, MOF (Meta-Object Facility) **bezivin2001towards caron2007ingenierie** qui consiste à associer chaque modèle à une cascade de métamodèles. Ces derniers dissocient les paramètres d'un modèle de sa structure. Le méta-modèle synthétise la seule structure du modèle. De même qu'un modèle peut être à son tour métamodélisé. Le standard MOF offre une structuration en quatre couches comme énnoncé dans le graphique V.7.

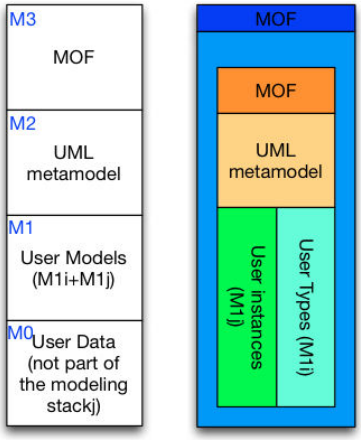



FIGURE *V.7 – représentation MOF* **atkinson2005concepts**

La couche $M_0$ concerne les données à intégrer dans le système modélisé.

La couche $M_1$ est le modèle à proprement dit. En MDE et particulièrement en POO la couche $M_1$ est la couche où sont définis les codes de l'application à travers les classes et les instances de classes.

La couche $M_2$ est la méta-représentation (dans UML généralement) de la solution logicielle.

La couche $M_3$ représente la définition du langage logique. Cette couche est caractérisée comme telle mais néanmoins, elle englobe l'ensemble des trois couches comme le montre le schéma V.7.

**wimmer2013testing** distingue deux types de transformations :

**M2M :** Pour model to model lequel touche les trois types d'abstractions précédemment énumérées. Il est possible d'imaginer des transformations entre niveaux d'abstraction hiérarchisés ou alors des transformations entre différents point de vue autour d'un système logiciel. Ce type de transformation est laborieux à mettre en place mais permet une interconnexion automatisée à terme entre les différents niveaux d'abstraction (PIM -> PSM) et les différents points de vue (PIM -> PIM). Le changement dans un modèle peut être alors répercuté sur l'ensemble des autres modèles d'une manière automatique.

**M2T :** Pour model to text. Ce type de transformation a pour but de générer de la documentation ou encore du code à partir de modèles PSM. La mise en oeuvre de ce type de transformation n'est pas encore standardisée même si certaines normes apparaissent pour la génération de documentation ou de code.

Nous nous intéressons dans cette thèse aux transformations M2M, plus précisément aux transformations PIM, qui est à rapprocher du modèle descriptif, vers PSM, qui est à rapprocher du modèle applicatif, car c'est ce que nous souhaitons utiliser dans la formalisation. En MDE, la mise en place de transformations entre modèles PIM et PSM consiste d'abord en la construction de liens entre les méta-modèles de la couche $M_2$ figure V.7 puis de la projection de cette représentation dans la couche $M_1$. Ces liens sont appelés règles. Il existe trois principales approches de transformation de modèles selon (BLANC et al. 2011).

**Approche par programmation :** Cette approche consiste à programmer les règles de transformation dans des langages orientés objet. L'idée est de programmer une transformation de modèles de la même manière qu'est programmée n'importe quelle application informatique. Cette application utilise les interfaces de manipulation de modèles Taylored et réflectives. Les interfaces Taylored sont adaptées à un unique type de modèles. Elles offrent des opérations spécifiques permettant une navigation dans les modèles instances d'un unique métamodèle. Les interfaces reflectives, contrairement aux interface Taylored, sont utilisables sur n'importe quels modèles quelque soit leur méta-modèle. Tous les modèles sont considérés comme des ensembles d'éléments (instances de méta-classes) reliés entre eux. L'intérêt de ce type d'interfaces est qu'elles offrent des moyens d'accéder aux méta-classes des méta-modèles. A partir d'un élément d'un modèle, il est possible d'accéder à sa méta-classe et ainsi obtenir les informations qui le structurent selon la figure V.8 (attributs, références, etc...)

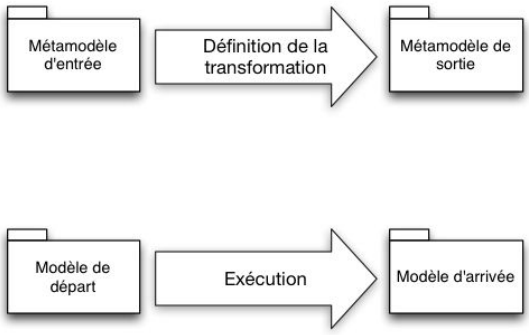



FIGURE *V.8 – Principe de transformation*

**Approche par template :** Consiste à définir les canevas des modèles cibles souhaités en y déclarant des paramètres. Ces paramètres seront substitués par les informations contenues dans les modèles sources. Les canevas sont appelés modèles cibles paramétrés ou modèles templates. L'exécution d'une transformation consiste à prendre un modèle template et à remplacer ses paramètres par les valeurs d'un modèle source. Cette approche nécessite un langage particulier permettant la définition des modèles templates. Sa puissance réside principalement dans le langage de définition des modèles templates.

**Approche par modélisation :** Cette approche consiste à appliquer les concepts de la MDE sur les transformations elles-mêmes. L'objectif est de modéliser les transformations de modèles et de rendre les modèles et les transformations pérennes et productifs grâce à leur indépendance vis-à-vis des plateformes d'exécution. Le standard MOF2.0 (Meta-Object Facility) et le standard QVT pour (Query/View/-Transformation) ont été développés dans ce sens V.9.

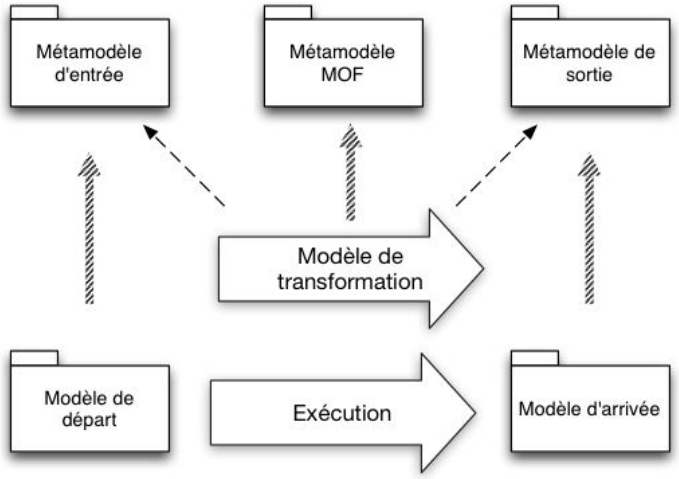



FIGURE *V.9 – Schéma de transformation méta-modélisée*

La nature du problème traité dans cette thèse est quelque peu différent de ce pourquoi la MDA a été imaginé. Néanmoins, L'approche par modélisation est celle qui se rapproche le plus des éléments du problème traité. Celle-ci a été retenue pour la suite. La descrip-

tion de la solution proposée expliquera en détail les analogies établies qui permettront d'énoncer la solution.

## V.4 Description de la solution proposée

La MDE est une discipline développée pour le génie logiciel. Celle-ci s'adresse aussi par extension à la communauté de l'intelligence artificielle. Néanmoins, des approches "métier" ont été structurées grâce à la MDA. Les travaux de **warkozek2011generation** ont constitué une approche de modélisation de problèmes liés à la gestion énergétique dans le bâtiment par optimisation PLNE. Ces problèmes enveloppent des modèles physiques ou économiques pour définir les stratégies de gestion et de consommation optimales selon un critère choisi. La résolution du problème est exécutée à travers plusieurs solveurs. En terme de modélisation, l'approche consiste à mettre en oeuvre des modèles dans le format XML suivant une structure prédéfinie dans un schéma XSD. Le modèle constitué (arbre d'information) est ensuite projeté vers d'autres structures XML, chacune relative à un solveur. Ces structures envoient le modèle au solveur. Les modèles introduits en XML sont les modèles PIM. Les modèles utilisés dans les solveurs sont les PSM au sens MDA. Les schémas xsd représentent les méta-modèles des PIM et des PSM. Les transformations consistent en l'extraction des informations du modèle XML de départ pour alimenter le modèle XML à l'arrivée. La structure de celui-ci n'est pas préalablement définie mais s'alimente au fur et à mesure des données reçues. L'objectif de ce travail est proche de l'objectif affiché par les AML. Néanmoins, celui-ci permet d'être inséré dans les structures G-homeTech développées à cette époque dans notre laboratoire.

Toujours dans une approche "métier" de la MDE, nous proposons de structurer notre vision de la solution de portage des modèles. Nous rappelons que notre objectif est de mettre en place un outil de transformation de modèles qui permettrait de passer d'un modèle dit descriptif à un modèle spécifique pour une application. Pour ce faire, nous avons choisi de poser le modèle descriptif comme PIM (Plateform Independant Model) et les modèles descriptifs comme des modèles PSM (Plateform Dependant Model) comme décrit dans le schéma V.10 ou le PIM est à gauche et les PSM à droite. Dans le sens vertical, nous avons choisi d'illustrer notre approche selon MOF. La couche M1 constitue la couche de modèles tandis que la couche M2 correspond aux méta-modèles des modèles PIM et PSM. Les modèles PSM sont mis en place à partir des exigences et de l'objectif de modélisation à ce niveau. Le modèle PSM ainsi composé doit être directement exécutable dans son application cible. Ainsi les transformations sont mises en place selon les méta-modèles de départ et d'arrivée sous forme d'un ensemble de règles. Ces règles sont ensuite utilisées dans des processus de transformation spécifiques à chacune des transformations. Ces processus de transformation entre chaque modèle sont posés pour définir l'enchaînement des étapes, entre règles automatiques et interventions manuelles. Une fois l'ensemble des éléments posés, nous proposons une mise en oeuvre associée à son application sur la salle expérimentale PREDIS-MHI.

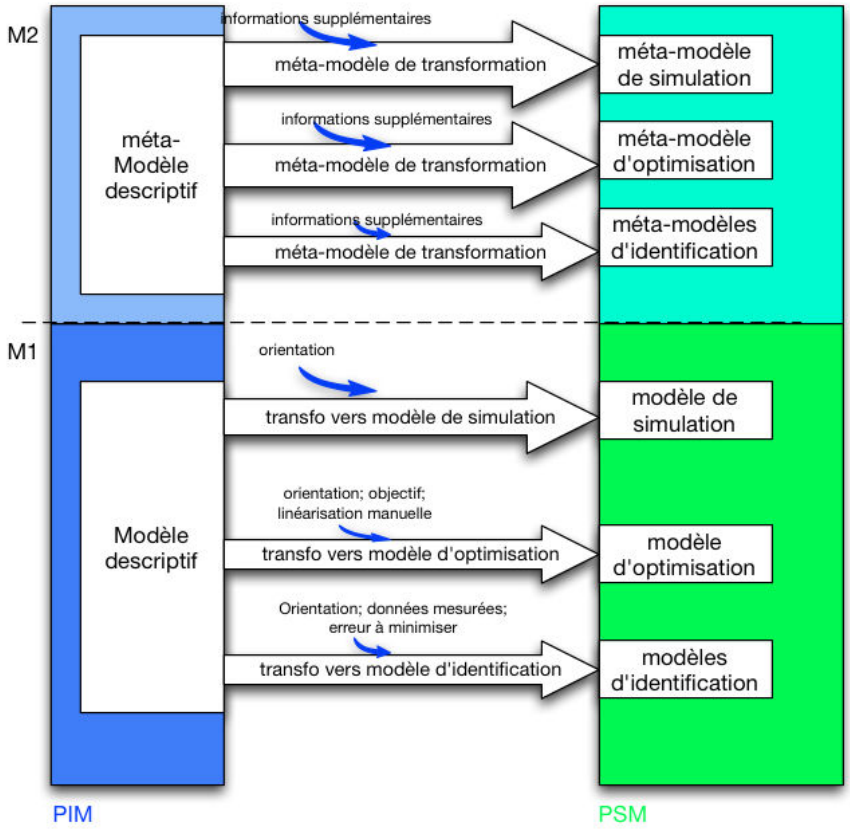



FIGURE *V.10 – Formalisation MDE du problème*

### V.4.1 Méta-modèle descriptif

Le modèle descriptif a été imaginé comme pivot pour les modèles applicatifs. Examinons quel doit être le format du modèle descriptif. Quelles informations doit-il contenir ?

Commençons par les informations. L'objectif du modèle descriptif est d'offrir une base commune aux modèles applicatifs afin que ceux-ci soient cohérents dans leurs interactions. Les approches à base d'AML et, dans une moindre mesure, à base de composants logiciels Icar ou FMU, privilégient des transformations complètement automatiques pour lesquelles l'information complète doit être fournie dans le modèle de départ. Cette condition impose au mieux le maintien de l'information dans le modèle d'arrivée sinon sa diminution. Cela implique un sens unique pour la transformation.

L'approche "boite blanche" développée dans MAEL (ALLAIN 2003) pallie cette limitation en offrant la possibilité de compléter un modèle de départ manuellement pour aboutir à un modèle d'arrivée avec pour unique condition le format et les besoins de ce dernier. La notion de symbole non orienté est aussi très intéressante. Cette solution parait plus appropriée pour notre problématique.

L'a-causalité du modèle descriptif est une condition qui permet de définir a posteriori les paramètres et le choix des entrées et des sorties. Cette observation trouve ses limites néanmoins car elle ne permet pas un champ d'action illimité en terme de choix de la représentation du temps : continue ou selon une discrétisation donnée. Les phénomènes observés et donc modélisés diffèrent selon le pas de temps choisi, problème parfois délicat. Par exemple, l'évolution de la température est une superposition de phénomènes lents et de phénomènes rapides. Le choix d'une plage de pas de discrétisation est primordial

pour la définition du type de dynamiques à modéliser.

La seconde discussion concerne le format du modèle descriptif. Le format de celui-ci répond à la volonté de garder la notion de pivot qui facilite au mieux les transformations vers chacun des modèles applicatifs attachés. Il est aussi important de souligner que le modèle descriptif peut être issu de l'un des modèles applicatifs. Les transformations doivent être réversibles. Nous avons observé que, le plus souvent, le modèle de départ pour l'optimisation constitue un modèle descriptif intéressant car répondant à notre exigence d'a-causalité. Cette exigence est issue du fait que le modèle de gestion anticipé est a-causal. De plus, la causalité d'un modèle est défini par le scénario de son usage selon les grandeurs qui doivent être considérées variables et celles qui doivent être considérées paramètres (domaine de valeurs possible réduit à un singleton).

La conséquence est que, pour ne pas nuire à la généralité, chaque grandeur intervenant dans le modèle descriptif doit être considéré comme un *symbole*. Cette exigence permet d'avoir un modèle non influencé par l'application à laquelle il est destiné car il ne doit pas y avoir de limitations pour passer d'un modèle à un autre. Ces limitations sont à l'origine d'une perte d'information liée à l'instanciation de symboles ou leur simplification.

Nous avons choisi un modèle qui inclut un maximum d'informations à partir duquel nous pourrons avoir un modèle applicatif sans changer d'information, en ne spécialisant que ce qui est nécessaire. Par exemple, nous proposons d'associer aux symboles des domaines de valeurs, ou domaines de définition, même larges, et de les réduire par la suite. En effet, les domaines de valeurs des symboles seront, selon les applications, amenés à être considérés comme des entrées, des sorties, des variables d'optimisation ou comme de simples paramètres. Chaque domaine de définition doit englober tous les cas de figures possibles dans les différents usages du modèle descriptif.

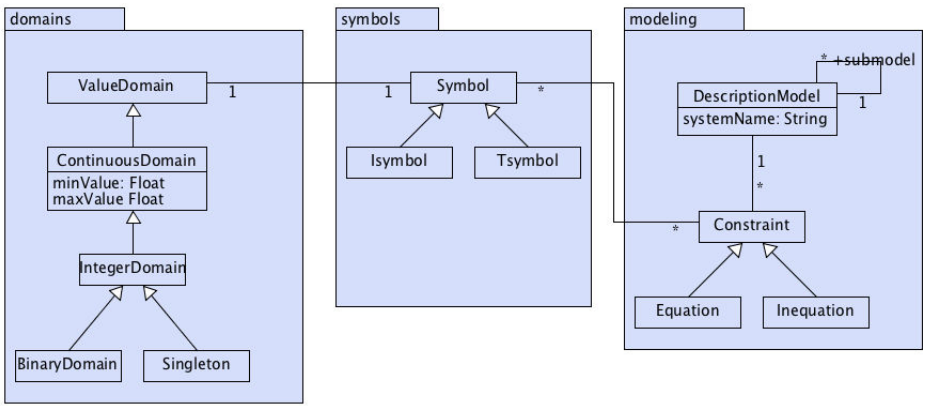



FIGURE *V.11 – méta-modèle descriptif*

Le méta-modèle résultant de l'ensemble de ces exigences a été résumé dans V.11. Il comporte trois ensembles distincts d'éléments :

— modeling : C'est la partie structurelle du modèle. Elle embarque la partie nominative et la gestion de cascade de modèles. Elle comporte aussi les contraintes que sont les équations et les inéquations.

— Symboles : La notion de symbole est la description de toute grandeur dans une contrainte, quelle soit de type équation ou inéquation. D'autre part la notion de symbole est liée à la nature temporelle de la grandeur représentée. Dès les premières étapes, certaines grandeurs révèlent leur temporalité. Certaines grandeurs

telles que les caractéristiques physiques de l'enveloppe ou des équipements installés sont inhérentes au matériel lui-même et ne sont donc pas variables dans le temps ce sont des *Isymbole*. D'autres grandeurs telles que les températures ou les puissances sont, elles, variables dans le temps en fonction d'autres éléments externes ce sont les *Tsymbole*

— Domaine : A chaque symbole est associé un domaine de définition. Ce domaine est l'énumération des valeurs et le type que peut prendre le symbole dans les différents modèles d'application.

Cette structure minimaliste en terme d'informations est organisée à la façon d'un problème d'optimisation car celui-ci constitue le coeur de notre problème mais aussi la forme la plus complète des applications mises en jeu. D'un point de vue MDE, le méta-modèle V.11 représente le méta-modèle *M2* du PIM dans le schéma V.10.

## V.4.2 Méta-modèle applicatif

### V.4.2.1 Méta-modèles d'optimisation PLNE

L'optimisation au sens où nous l'entendons dans cette thèse est l'optimisation PLNE en vue de développer des stratégies de gestion anticipative dans GHomeTech. Le problème d'optimisation est établi selon un scénario d'usage délimitant les degrés de liberté permis sur les équipements. Ce scénario se décline dans le problème par la définition des symboles.

Un problème d'optimisation PLNE doit contenir quatre parties :

**la définition des symboles :** Dans un problème d'optimisation, chaque symbole utilisé dans les contraintes peut être fixe (paramètre) ou libre (variable d'optimisation). La spécialisation d'un symbole en paramètre doit être déclaré comme telle. La définition d'un symbole en variable implique un renseignement détaillé sur :

— la nature de cette variable selon qu'elle est réelle, discrète ou binaire.

— le domaine de valeurs : c'est l'espace que l'algorithme d'optimisation doit explorer. Étant donné qu'une variable en PLNE ne peut être que scalaire, le domaine en question est de dimension 1, limité par une borne inférieure et une borne supérieure.

Les paramètres peuvent être scalaires, vectoriels ou matriciels. Les variables d'optimisation ne peuvent être que scalaires. De ce fait, la temporalité doit être exprimée en amont, à travers la procédure de transformation. Dans le problème d'optimisation PLNE pour la gestion, chaque variable correspond à un pas de temps discrétisé.

**la définition des contraintes :** Une contrainte d'optimisation est une équation égalité ou inégalité. Elle permet d'exprimer les liens entre symboles. Dans l'application d'optimisation, nous avons choisi une résolution linéaire, ce qui implique la linéarité des contraintes. Les fonctions mathématiques ne peuvent être directement intégrées dans une contrainte PLNE. Seules les fonctions arithmétiques "+", "-","*" sont permises. Enfin, Une contrainte doit être a-causale. En ce sens et par exigence des interfaces, une contrainte doit avoir une forme polynomiale :

$$AX + BY + CZ \leq = \geq 0$$

**la définition de l'objectif :** L'objectif est une variable associée aux autres variables et paramètres du problème dans le jeu de contraintes. Par ailleurs, c'est la métrique

de choix de solution durant le processus d'optimisation. L'objectif peut être une variable déjà existante dans le modèle que le développeur de modèle choisi de spécifier en temps que tel ou bien une nouvelle variable définie manuellement.

**le renseignement des paramètres (données) :** L'ensemble des données du problème doit être renseigné avant la résolution.

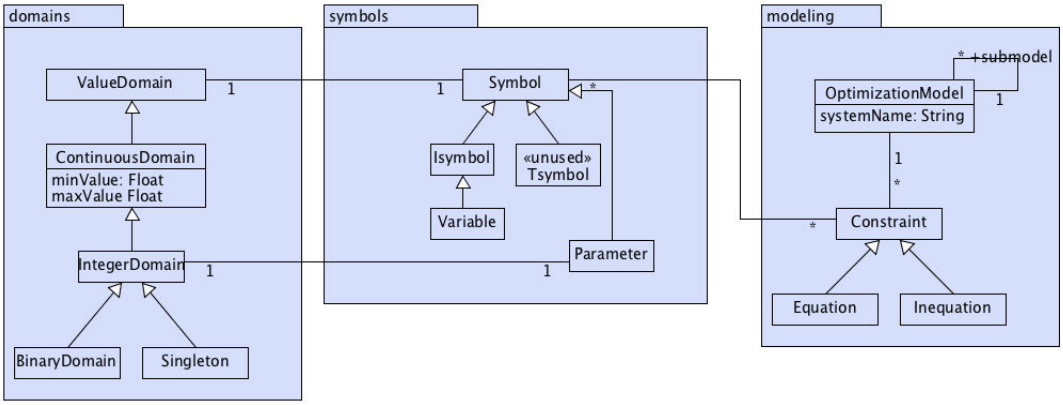



FIGURE *V.12 – méta-modèle d'optimisation*

Le méta-modèle reprend le même schéma que le modèle descriptif, ce, afin qu'il soit cohérent avec le méta-modèle descriptif. Ce qui facilite la mise en place des transformations.

Les spécificités liées à l'optimisation sont aménagées dans le méta-modèle à travers des liens spéciaux entre entités, l'apparition de nouvelles entités ou encore la désactivation de certaines *«unused»*. Dans le schéma V.12, il y a les notions de variable et de paramètre qui sont associées à un symbole. Dans notre schéma *parameter* et *variable* héritent de symbol d'une manière directe ou indirecte. Cela veut dire que ces deux grandeurs héritent des attributs de *Symbol*. Ce sont principalement le nom complet (intégrant sa position dans la hiérarchie des modèles) et la connexion à un domaine de valeur. Ce dernier définit l'espace de liberté sur chaque variable pour l'outil d'optimisation. Il définit aussi le type de valeurs que peut prendre la variable : continu, discret ou binaire. Ce type est défini dans les entités *ContinuousDomain*, *IntegerDomain* et *BinaryDomain*. L'héritage d'une entité à partir d'une autre montre l'inclusivité. Le domaine binaire est un ensemble de deux valeurs 0 et 1. Le domaine des entiers est aussi un ensemble dans lequel des valeurs sont énumérées. La variable d'optimisation liée à cet ensemble ne peut alors prendre que l'une des valeurs énumérées dans l'ensemble. Enfin Le domaine continu est un fragment de l'ensemble des réels limité par des bornes inférieures et supérieures.

Les paramètres sont ici des singletons c'est à dire des valeurs fixées. Les symboles, qu'ils soient paramètres ou variables, sont utilisés dans des contraintes telles que définies plus haut.

### V.4.2.2 Méta-modèles de simulation

Un modèle de simulation est un système d'équations dans lequel les *entrées* sont associées aux *paramètres* du modèle pour générer des *sorties* V.13. Les possibilités d'usage sont multiples en simulation. La transformation depuis le modèle de description vers le modèle de simulation nécessite un scénario décrivant le cas de figure choisi. Le scénario

est un complément d'information au modèle descriptif. Le scénario apporte l'information concernant le choix des orientations des symboles.

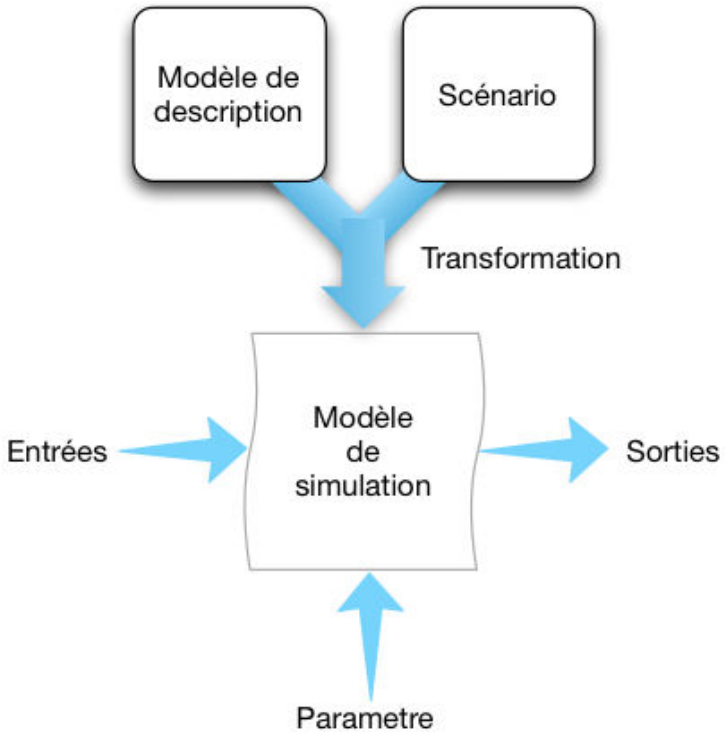



FIGURE *V.13 – processus de simulation*

Le calcul des sorties se fait d'une manière causale sur la base d'un système d'équations égalités clairement ordonnées et suffisamment renseignées pour fournir le résultat sur l'ensemble des sorties. L'ordonnancement des équations est une étape importante mais qui n'est pas assurée d'aboutir. Cette étape n'est possible que si le modèle est juste déterminé. Un modèle sous déterminé ne peut avoir une solution unique alors qu'un système d'équations sur-déterminé ne peut être résolu car trop contraint . La condition de juste détermination est liée à l'orientation des symboles en entrées/sorties. En effet, seul un espace limité de combinaisons est possible. Il est donc nécessaire de vérifier que le scénario choisi par le développeur de modèle rentre dans l'espace des combinaisons juste déterminées.

ajouter ref(s)

Un modèle peut être décrit sous forme continue ou discrète mais un modèle de simulation est un modèle temporel. Dans le cas d'un modèle continu, sa temporalité est assurée par l'organisation des entrées sous forme de courbes continues. Dans le cas d'un modèle discret, un pas d'échantillonnage et un horizon doivent être spécifiés pour assurer la dimension temporelle du modèle. Les entrées sont contenues dans des vecteurs et chaque entrée est identifiée par un symbole et un indice de temporalité lorsque celui-ci est variant dans le temps. Si elles ne sont pas variantes dans le temps, dans ce cas, leur représentation n'intègre pas d'indice et l'entrée correspondante est un scalaire.

Un modèle de simulation est donc décrit par le scénario de son usage qui comprend le tri des symboles en entrée/sorties, la dimension temporelle de chaque symbole, la causalité permettant d'ordonnancer ses équations et finalement ses contraintes égalité. Un module supplémentaire peut être embarqué contenant l'ensemble des inégalités. Ce mo-

dule n'est pas traité par le simulateur en premier lieu, il sert à une étape de validation de sorties du simulateur schéma V.13.

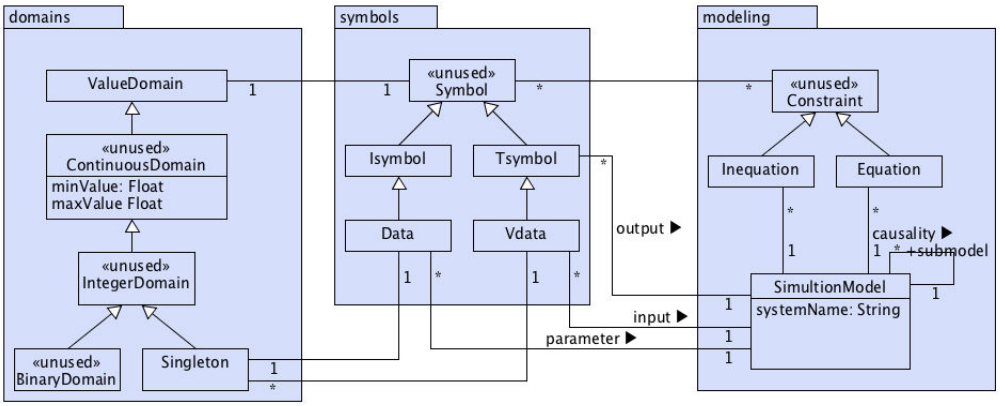



FIGURE *V.14 – méta-modèle de simulation*

Le méta-modèle proposé suivant l'ensemble des exigences énumérées est schématisé dans la figure V.14, toujours sur la base du modèle descriptif. Le modèle proposé intègre la structure du modèle de simulation.

La notion d'équation et d'inéquation représente le corps du modèle. La notion de contrainte n'est pas utilisée dans un modèle de simulation, elle est donc désactivée. La causalité est le lien qui définit l'ordre de résolution des équations. De chaque équation, ou système d'équations indissociable, doit être retournée une sortie, c'est-à-dire un symbole déduit.

D'un point de vue symboles, la temporalité est intégrée à travers la distinction entre *Tsymbole* et *Isymbole*. Les entrées sont des *Vdata* dans le système tandis que les sorties sont des *Tsymbole* car ce sont des inconnus variant dans le temps pour le modèle. Les paramètres sont des invariants temporels et ils sont renseignés dans le modèle par un singleton.

### V.4.2.3 Méta-modèles d'identification paramétrique

L'identification paramétrique est utilisée pour quantifier la valeur de certaines variables invariantes (Isymbol) nommées 'paramètres'. Cette étape est nécessaire à l'étape de déploiement. Le principe consiste à utiliser un modèle dual au modèle cible (modèle d'optimisation ou de simulation) afin d'en déduire ses paramètres à partir de valeurs mesurées sur le terrain. Ce n'est pas considéré comme étant de l'apprentissage car l'algorithme est appliqué sur un jeu de données archivé.

Il est nécessaire de fournir à cette application une causalité qui s'inspire du modèle de simulation. Le mécanisme de résolution choisi pour le problème d'estimation paramétrique est une approche récursive qui fait appel à la simulation d'un modèle à chaque itération pour estimer l'erreur entre des sorties simulées et des sorties mesurées. Le but des itérations est de minimiser cette erreur en choisissant le jeu de données optimal.

Un modèle d'identification paramétrique doit comprendre figure V.15 :

— un modèle de simulation
— un scénario d'estimation
— des données terrain correspondant à certaines entrées et à toutes les sorties du modèle de simulation

— une variable de comparaison (l'erreur) entre les valeurs étalon (données terrain) et les sorties du modèle de simulation (données estimées), utilisé dans l'objectif à minimiser.

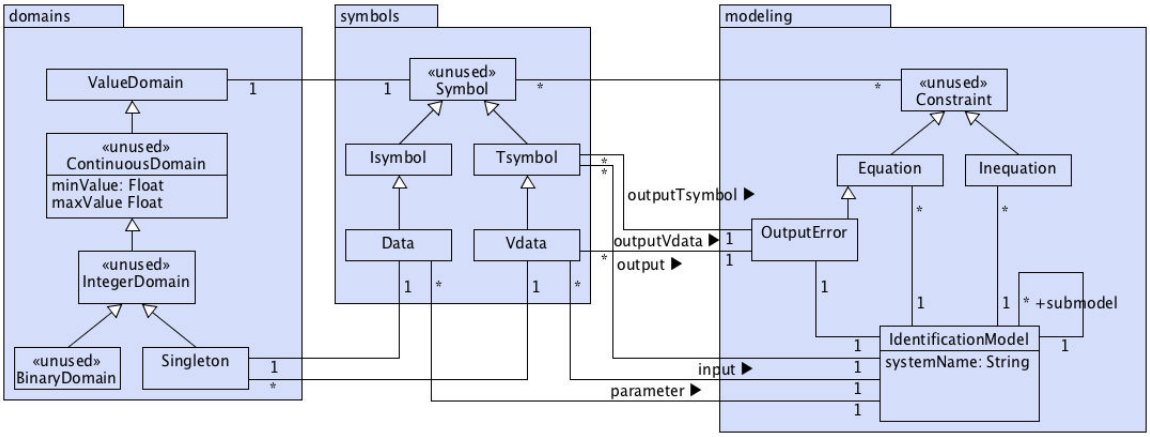



FIGURE *V.15 – méta-modèle d'identification*

Le scénario de simulation consiste à définir les entrées, les sorties et les paramètres du modèle. Le scénario d'estimation est le même que pour la simulation avec, en plus, le choix des variables étalons. Ces variables étalons appelées *outputVdata* sont des données tandis que les sorties au sens simulation sont des inconnues *outputTsymbol*. D'autre part, les paramètres n'ont plus la même définition que dans la simulation qu'est le *singleton*. Ce sont là des *Isymbol*, c'est à dire des inconnus à trouver pour l'algorithme d'optimisation. Le *Isymbol* est associé à un domaine de valeur continue ou discret dans lequel l'outil d'estimation recherche une solution.

## V.4.3 Eléments de transformation

Ce que nous avons communément appelé jusqu'ici transformation est la procédure qui consiste à automatiser des tâches de manipulation en vue du passage d'un type de modèles vers un autre.

Le chapitre précédent a mis en évidence les tâches nécessaires pour la transformation manuelle de l'exemple de l'enveloppe thermique depuis un modèle descriptif vers différents types de modèles applicatifs. Dans cette partie, nous tenterons de généraliser l'approche et ainsi d'établir des ensembles de règles utilisables dans chacune des transformations. Nous proposerons ensuite des procédures de transformation qui consistent à aligner ces règles dans une démarche automatisée ou semi-automatisée afin de passer d'un modèle à un autre. Nous conclurons ensuite sur les possibilités offertes par l'approche et les limites de l'automatisation.

La démarche se démarque des travaux recensés dans la littérature scientifique par sa flexibilité entre intervention humaine et étapes automatisées par rapport aux composants logiciels FMI et Icar et par sa généralité par rapport à la solution MAEL qui se limite à la seule simulation dans différents scénarios.

L'exemple d'enveloppe thermique choisi pour le chapitre précédent est un élément de modèle pour la gestion énergétique dans le bâtiment. Un ensemble d'autres modèles d'équipements se connecte à celui-ci pour former la partie thermique du problème. Nous retrouvons en parallèle à cette partie thermique la partie qualité de l'air qui est liée au

niveau de CO2 dans l'espace. Cette partie ressemble en terme de structure de modèle à la partie thermique. Ces deux éléments de modèles représentent la partie la plus complexe en terme de modélisation car ils comportent des dérivées, de ces non-linéarités. A côté de ces deux sous-systèmes, nous retrouvons les modèles représentant les bilans des différents vecteurs énergétiques que sont l'électricité, le gaz, le bois, etc...

Dans la première partie de cette thèse le modèle complet de gestion de CANOPEA a été décrit. Une vision globale concernant la complexité peut être nourrie à partir de cette modélisation. Nous avons choisi de classifier et énumérer les difficultés rencontrées dans le tableau V.1 reprenant chaque modèle en fonction de tâches à réaliser dessus.

| modèle | produit non linéaire | dérivée | fonction mathématique | complexité particulière |
|---|---|---|---|---|
| modèle d'enveloppe | ✓ | ✓ | ✓ | Certains modèles d'enveloppe contiennent des choix conditionnels |
| modèle de JVP | ✓ | | | |
| modèle compact P | ✓ | | | |
| modèle CO2 | ✓ | ✓ | | |
| modèle confort thermique | ✓ | | | |
| modèle confort CO2 | ✓ | | | |
| modèle Panneau hybride | ✓ | | | |
| modèle PV | | | | |
| modèle réseau électrique | | | | |
| modèle balance électrique | | | | |

TABLE *V.1 – Classification des difficultés de modélisation*

La synthèse du tableau V.1 concerne l'application d'optimisation uniquement car les modèles décrits dans le travail sur CANOPEA s'y est limité. De ce fait, nous remarquons que le modèle d'enveloppe thermique concentre toutes les difficultés dans cette application. La présence de fonction mathématique n'a pas été intégrée dans le modèle mais devra faire l'objet d'une attention particulière dans le développement des transformations. En ce qui concerne les autres applications que sont l'identification et la simulation, le travail pour chaque tâche de transformation supplémentaire est le même et plus simple sur le traitement des fonctions. Il est donc pertinent de considérer la démarche manuelle appliquée sur le modèle d'enveloppe comme exhaustive d'un point de vue traitement hors fonctions. Nous nous basons sur cette approche en considérant la solution évolutive.

Nous discutons les transformations dans le sens du modèle descriptif vers les autres modèles applicatifs.

Le modèle de départ, le modèle descriptif, est entièrement établi par le développeur de modèles dans notre hypothèse. Nous avons fourni à travers la première partie de la thèse l'outil de gestion, de hiérarchisation et de composition de modèles Milp-workshop. Les mécanismes de l'outil sont entièrement repris dans le développement et la gestion des modèles descriptifs.

Le début de l'orientation vers un modèle d'application commence pour toutes les transformations par une définition du cas d'usage. Cette définition est contenue dans le *scénario*. Le scénario consiste à adapter le modèle à un cas précis selon les données disponibles et selon l'attente qu'il y a de l'application l'utilisant. Un scénario comporte la spécification des symboles ainsi que des renseignements sur les variables dans le cas de l'optimisation.

La mise en place du modèle descriptif ainsi que le scénario sont un apport d'information. Nous considérons donc ces étapes comme manuelles et ne pouvant être fournies que par le développeur de modèles ou peut être l'utilisateur de l'application dans certains cas.

Afin d'être en accord avec la représentation MDA de notre solution, il est nécessaire de définir les mécanismes de transformation. La transformation, proprement dite, sera le processus de transformation complet depuis un modèle type vers un autre.

Rappelons du chapitre 5, les étapes parcourues manuellement pour chacun des trois types de transformations de l'exemple de l'enveloppe thermique :

du modèle descriptif au modèle d'optimisation

- application des réductions de domaines
- classification et spécification des symboles en variables ou paramètres
- simplification des équations
- linéarisation des équations
- discrétisation temporelle (dérivées, variables et contraintes)
- intégration de l'objectif d'optimisation
- formatage

du modèle descriptif aux modèles de simulation

- classification et spécification des symboles en entrées/sorties
- discrétisation
- ordonnancement du système d'équations
- formatage

du modèle descriptif au modèle d'identification

- mise en place d'un modèle de simulation
- intégration du calcul de $\Delta$
- formatage

D'un autre côté, suivant l'architecture MDA et au travers des méta-modèles, nous mettons en relief ces transformations entre méta-modèles comme dans les trois schémas V.16, V.17, V.18.

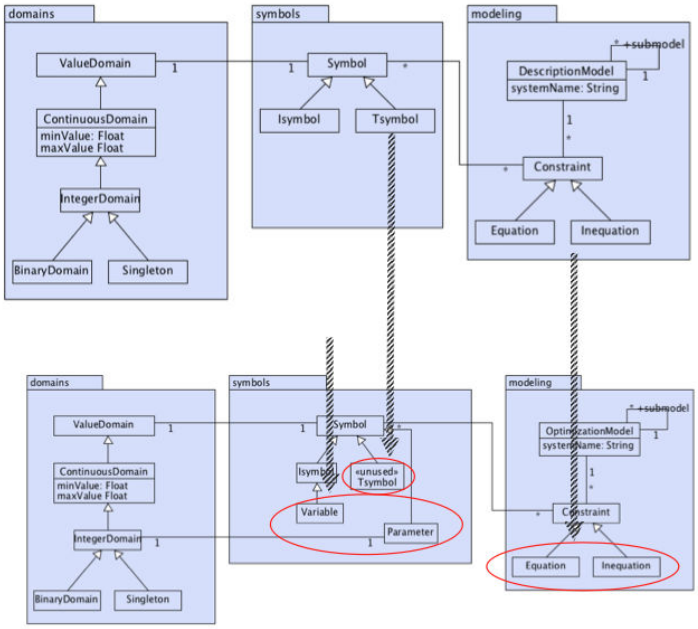



FIGURE *V.16 – transformation depuis un modèle descriptif vers un modèle de gestion anticipée*

Le passage d'un modèle descriptif vers un modèle de gestion anticipée nécessite tout d'abord la spécification du scénario puis son implémentation. Son implémentation consiste à spécialiser en variable d'optimisation ou en paramètre chaque symbole. Les domaines de valeurs peuvent être aussi resserrés.Vient ensuite la discrétisation. Le modèle de gestion anticipée n'intègre pas la notion de temps. Celui-ci est alors exprimé dans une série de variables indépendantes $Isymbol$.

Du côté *modeling* les équations et inéquations doivent être linéaires et dans un format spécifique décrit précédemment.

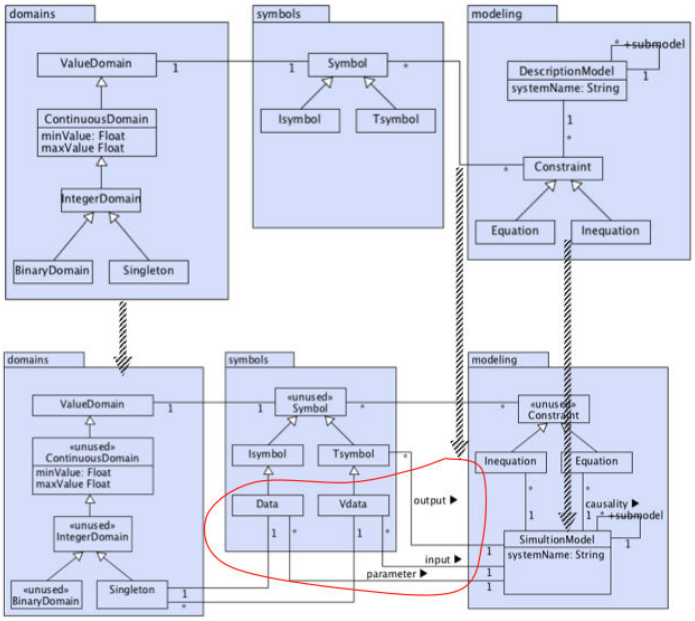



FIGURE *V.17 – transformation depuis un modèle descriptif vers un modèle de simulation*

La transformation vers le modèle de simulation nécessite lui aussi le scénario. Celui-ci contient l'orientation des symboles en Entrée, Sortie ou Paramètre. D'autre part, la structure du modèle doit contenir des formats spécifiques pour les données : les $Data$ et les $Vdata$ en plus des $Tsymbol$ qui embarquent les sorties du modèle. Du côté *modeling* la transformation principale est l'ordonnancement causal des équations. C'est l'étape nécessaire pour la résolution avec les outils choisis. Les domaines et les inéquations ne sont pas automatiquement utilisés dans cette étape. Ils sont facultatifs et uniquement pour la validation (pour les inéquations).

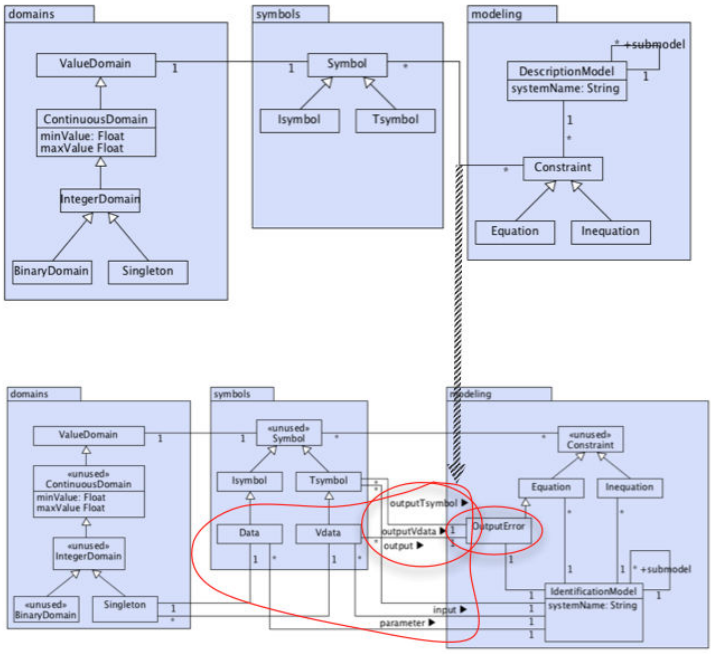



FIGURE *V.18 – transformation depuis un modèle descriptif vers un modèle d'estimation paramétrique*

La transformation pour l'estimation paramétrique nécessite les mêmes étapes que la transformation pour la simulation. Néanmoins, les paramètres ne sont plus des $Data$ mais plutôt des $Isymbol$. Cette différence signifie que ces derniers sont des variables d'optimisation dans le processus de résolution. Ils constituent les résultats de l'estimation paramétrique, avec des informations additionnelles nécessaires que sont la spécification des variables à utiliser comme étalon ainsi que de l'erreur à minimiser durant le processus itératif d'optimisation. Cette dernière valeur constitue une appréciation de la qualité des résultats estimés que sont les paramètres.

Nous regroupons certaines des étapes de différentes transformations en tâches dans le méta-modèle de transformation car elles utilisent les mêmes mécanismes. Nous classifions les tâches en deux groupes : les tâches automatisables et les tâches manuelles qui ne peuvent être automatisées. Les tâches manuelles correspondent à des tâches d'alimentation en information ou des tâches de manipulation irrégulières d'où l'impossibilité de leur automatisation. Elles seront intégrées dans le processus final de transformation pour chaque application mais ne seront pas spécialement décrite.

### V.4.4 Tâches automatisées

La classification et la spécification des symboles est une tâche qui relève de mêmes mécanismes pour l'ensemble des transformations qui sera appelé introduction du scénario. Il en va de même pour l'étape de formatage et de simplification des équations qui relève d'une tâche générale que nous appellerons manipulation d'équations. L'introduction d'objectif est une étape partagée entre la transformation vers le modèle d'optimisation et d'identification. Enfin, ce qui relève du traitement de l'aspect temps et de la discrétisation seront regroupés dans une tâche que nous appellerons projection temporelle.

### V.4.4.1 Manipulation d'équations

La manipulation algébrique d'équations intervient à plusieurs reprises dans un processus de transformations. La manipulation algébrique consiste à passer d'une forme d'équation à d'autres selon les exigences des étapes suivantes.

La linéarisation nécessite le développement de toutes les équations a priori. Cela permet d'appliquer l'algorithme de recherche des formes non-linéaires d'une manière exhaustive. L'algorithme de recherche ne reconnaît que les non-linéarités (NL) de forme produit entre symboles dont les domaines de valeurs ne sont pas réduits à un singleton. Par exemple, la forme $X \times Y + X$ est identifiée comme produit NL si $X$ et $Y$ ont des domaines de valeurs non-singleton. A contrario, la forme $X \times (Y + 1)$ ne peut être identifiée comme une non-linéarité car $(Y + 1)$ n'est pas considéré comme symbole.

Un autre cas d'utilisation de la manipulation algébrique est la simplification des équations après une phase d'instanciation de paramètres. La phase d'instanciation intervient en amont de la phase de linéarisation. La simplification augmente les chances de résolution naturelle de non-linéarités à travers des valeurs de paramètres nulles, des expressions qui s'annulent ou des fractions qui se simplifient après calcul par exemple.

La manipulation algébrique d'équations permet aussi le formatage des équations en fonction des exigences des applications. Dans le cas des modèles pour l'optimisation, seul un format de type V.1 est accepté :

$$A_1 \times X + A_2 \times Y + A_3 \times Z \leq = \geq A_4 \tag{V.1}$$

avec l'usage d'une même variable au maximum une fois dans une même équation et le tri des inconnues d'un côté et de l'autre la valeur telle que dans la forme V.1. L'équation ne peut être plus contractée ou simplifiée.

Le modèle de simulation nécessite un ordonnancement des équations. Un outil de manipulation algébrique permet de mettre en œuvre l'ordonnancement choisi. Cette mise en œuvre consiste à extraire une variable en fonction du reste dans chacune des contraintes-égalités du modèle.

Par exemple, si l'ordonnancement nécessite la déduction de $X$ dans

$$A_1 \times X + A_2 \times Y + A_3 \times Z = A_4 \tag{V.2}$$

il suffira alors d'appliquer une commande du logiciel de calcul formel ("solve" dans Xcas ou python Sympy) pour obtenir :

$$X = -\frac{A_2}{A_1} \times Y - \frac{A_3}{A_1} \times Z + \frac{A_4}{A_1} \tag{V.3}$$

L'automatisation des transformations algébriques est assurée par des outils externes dédiées à la tâche : les CAS. Les CAS ou Computer Algebra System sont des outils de calcul formel. Leur utilisation constitue une étape principale dans les transformations de modèles car c'est une automatisation pertinente. Un CAS est un outil qui embarque les principaux principes de l'algèbre. Il existe une multitude de systèmes GAP, Mathematica, Maple, Maxima, Sympy, Meditor, MuPAD, Magma, SAGE, Xcas / Giac, Yacas, PARI/GP, Derive, Mathcad. Chaque outil présente ses spécificités et sa palette de fonctionnalités. Certains sont payants alors que d'autres gratuits. Certains outils présentent des IHM et permettent ainsi l'interactivité entre utilisateur et l'algorithme. L'usage du CAS dans nos exemples de manipulation du chapitre précédent s'est basé sur des CAS aux interfaces

logicielles disponibles. Nous avons utilisé Sympy et XCAS/Giac pour les transformations manuelles. Ce sont deux outils sous licence BSD et GNU-GPL respectivement.

L'usage des CAS par d'autres applications passe à travers les API logicielles. Celles-ci offrent une passerelle pour transmettre les commandes et les expressions. Pour que cette communication puisse avoir lieu, il est primordial de connaître la structuration des données à fournir aux API des CAS. Chaque API a sa spécificité mais un principe commun réside dans l'arbre de représentation des équations. L'exemple sur l'ordonnancement V.2 est représenté par son arbre V.19. La commande Solve applique des règles prédéfinies **Livre**

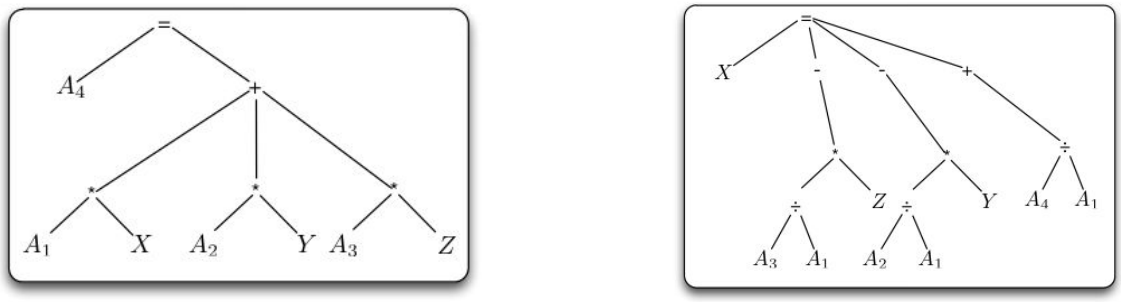



FIGURE *V.19 – Arbre de représentation*

### V.4.4.2 Linéarisation

La linéarisation ne concerne que l'optimisation. De ce fait, c'est une procédure que nous avons déjà traitée en première partie de ce manuscrit pour l'application d'optimisation. Le travail à réaliser pour automatiser cette procédure est l'intégration du module, déjà développé pour Milp-workshop dans le schéma V.20. Le traitement des non-linéarités dans un modèle d'optimisation se fait en deux temps. La première étape consiste à détecter puis identifier le type de non-linéarités. Pour ce faire, un développement des équations s'impose afin de mettre à plat toutes les formes mathématiques comme expliqué en détails dans la tâche de manipulation. La seconde étape de linéarisation consiste à appliquer la transformation adéquate.

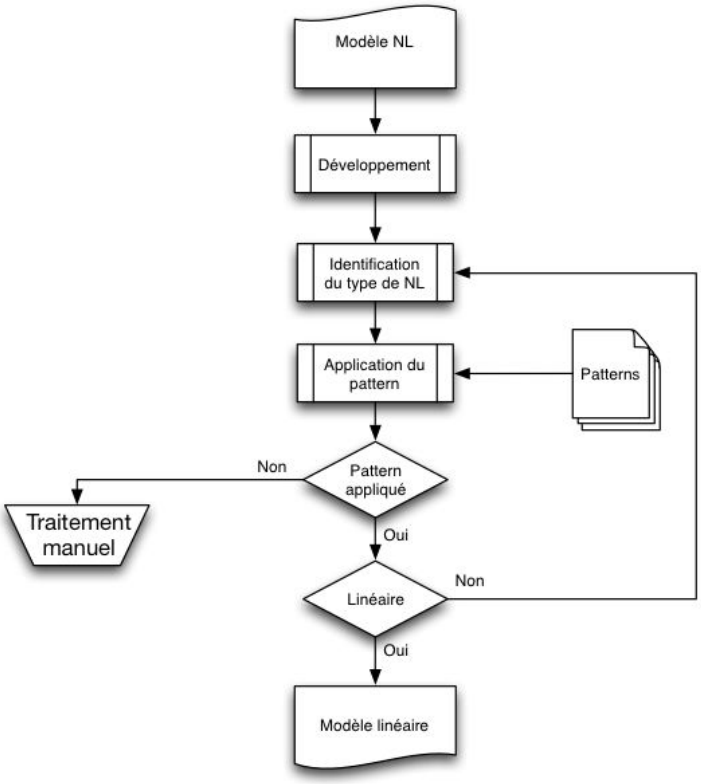



FIGURE *V.20 – Procedure de Linéarisation*

### V.4.4.3 Projection temporelle

Cette étape est commune aux trois applications : optimisation, simulation et identification. La projection temporelle consiste à ajouter la dimension temps à un modèle. Cette partie comporte un ajout d'information mais aussi une re-formulation des variables et des équations. L'information nécessaire est le pas d'échantillonnage, l'horizon de travail et, par déduction, le nombre d'intervalles. Chaque intervalle est associé à un indice que nous appellerons indice temporel.

La projection temporelle intervient sur des modèles déjà orientés par leurs scénarios. Il est donc question de variables-paramètres (V/P) dans le cas de l'optimisation et d'entrées-sorties (E/S) dans le cas d'une simulation et de paramètres-données (P/D) dans le cas d'une identification. Le traitement des entités E/S, P/D, V/P est effectué sous la même procédure. Nous continuerons donc à parler de symboles pour généraliser la description du traitement.

L'aspect temporel intervient en premier lieu au niveau des symboles. Chaque symbole Tvar est ainsi projeté en fonction du nombre d'intervalles. Le processus projette le symbole en le démultipliant en symboles temporalisés comme dans le schéma V.21. Une trace est intrinsèquement gardée de la transformation en encapsulant chaque projection de variable dans un vecteur.

La seconde étape concerne les équations. Là encore, un mécanisme qui parcourt les équations à la recherche d'éventuels symboles temporels Tvar est utilisé. Si la situation se présente, l'équation est alors considérée comme temporelle. Cette équation est projetée sur chaque pas de temps. La projection d'une équation dans la dimension temporelle consiste à mettre en commun, à l'identique de l'équation source, les $Tsymbol$ de même

indice que l'indice temporel de l'équation comme dans le schéma V.21.

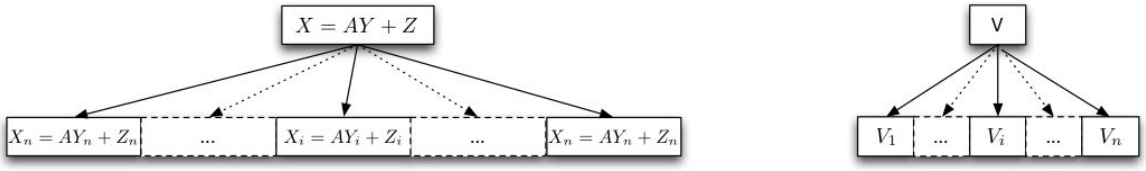



FIGURE *V.21 – Projection temporelle*

Les équations peuvent aussi contenir des fonctions ou des opérateurs sur des variables. Ces fonctions doivent être traitées séparément du reste des opérations arithmétiques. Il peut y avoir une infinité de cas à traiter mais dans notre pratique de modélisation pour la gestion, nous nous sommes limités au traitement de la dérivée et de l'intégrale. Le traitement de ces deux fonctions a été réalisé avec la même approche que le traitement des non-linéarités, c'est à dire que dans un premier temps les fonctions sont détectées et dans un second temps, un pattern leur est appliqué. Pour rappel, les patterns utilisés pour la dérivée et l'intégrale sont les suivants :

$$\dot{X(t)} \rightarrow \frac{X(k+1) - X(k)}{\Delta_t} \tag{V.4}$$

$$\int_{t_1}^{t_2} X(t)\mathrm{d}t \rightarrow \sum_{k_1}^{k_2} X(k) \tag{V.5}$$

### V.4.4.4 Vérification de la causalité

Cette étape est préliminaire à l'ordonnancement. Elle consiste à vérifier la possibilité de résolution du système d'équations d'une manière séquentielle. La vérification est basée sur un inventaire de l'ensemble des symboles considérés comme sorties du modèle pour la simulation associés aux contraintes de ce dernier. Cet inventaire est contenu dans une matrice d'incidence. Le traitement de cette matrice consiste alors à vérifier la juste-détermination de l'ensemble à défaut de quoi, le système doit être en mesure de communiquer au développeur de modèle les parties sous ou sur déterminées comme il sera expliqué en détail dans la partie V.5.

### V.4.4.5 L'ordonnancement des équations

L'ordonnancement des équations est l'étape qui consiste à préparer un système d'équations à être résolu séquentiellement d'une manière automatisée. La plupart des solveurs embarquent cette fonctionnalité. Néanmoins, aucun ne garantit une solution. Un travail en amont doit être effectué pour vérifier l'existence d'une solution et, éventuellement, fournir des précisions sur les conflits à résoudre ou les éléments nécessaires pour résoudre le système d'équations. Cette étape est donc automatique sous condition d'existence de solution.

L'ordonnancement rentre dans une règle de transformation plus générale qui est la détermination de la causalité d'un modèle. Cette règle permet dans un premier temps de vérifier la causalité du modèle descriptif associé à un scénario d'usage. Il peut y avoir

trois cas de figures : la sous détermination, la juste-détermination et la sur-détermination (YASSINE 2008). Seul un modèle juste déterminé est causal. En cas de sur détermination, il faut alors identifier les conflits entre variables principalement pour pouvoir y remédier. En cas de sur-détermination, il y a alors conflit potentiel entre contraintes. La résolution de ces conflits n'obéit à aucune règle susceptible d'être automatisée. Seul le développeur peut modifier le scénario d'usage pour rendre l'ensemble causal. Une fois la causalité vérifiée, il est ensuite nécessaire de définir l'ordre de résolution des équations afin de pouvoir l'exécuter d'une manière séquentielle. Cette règle a été entièrement définie et développée dans notre approche car elle constitue une étape importante dans la mise en place d'un modèle simulable. Nous développerons l'ensemble de la démarche dans la partie suivante.

### V.4.5 Processus de transformations

Les processus de transformation représentent l'enchaînement des tâches précédemment citées afin de transformer le modèle descriptif en un modèle applicatif. La figure V.22 représente les légendes des formes utilisées pour décrire chaque processus dans la figure V.23.


FIGURE *V.22 – Entité du processus de transformation*

### V.4.5.1 Processus de transformation pour l'obtention du modèle d'optimisation PLNE

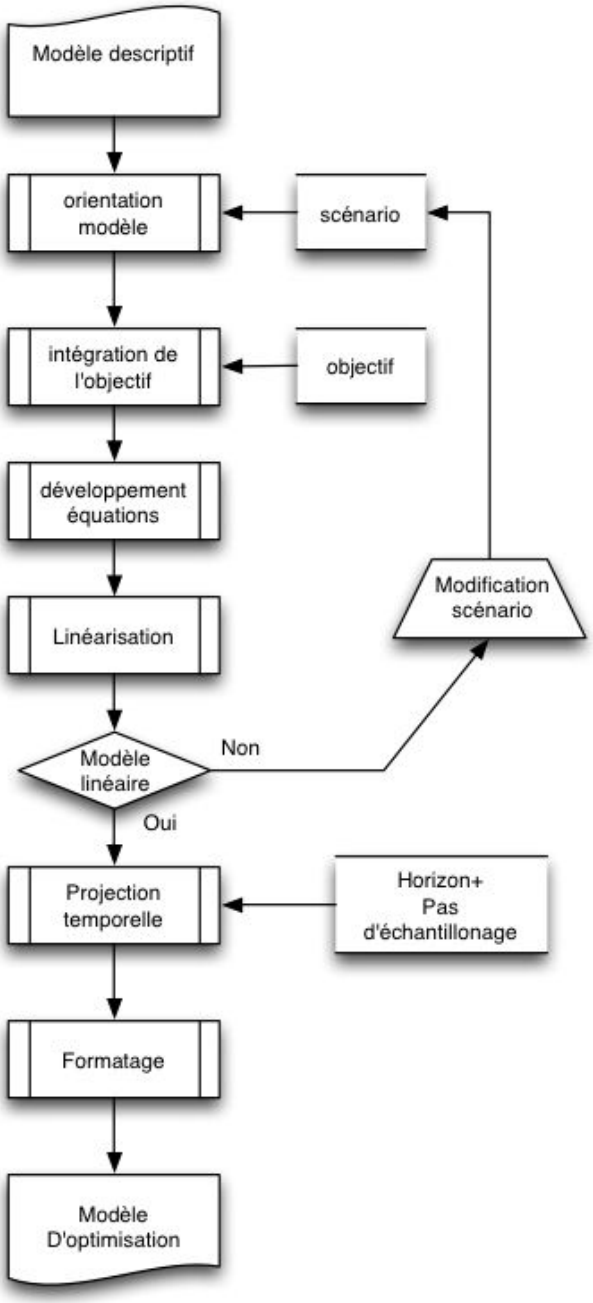



FIGURE *V.23 – Processus de transformation de modèle pour l'optimisation PLNE*

Le processus associe des tâches automatiques aux tâches manuelles. Le modèle descriptif et le scénario choisi par le développeur sont associés. L'objectif d'optimisation complète le modèle. Le modèle résultant est ensuite soumis à l'étape de linéarisation. Cette étape peut passer automatiquement dans le cas où le modèle initial ne comporte pas de NL non répertoriées dans les patterns. Si c'est le cas, alors le développeur reprend la main pour modifier le scénario et ainsi éviter ces non-linéarités. Une fois le modèle linéaire obtenu, l'étape de projection temporelle est exécuté automatiquement. Le modèle

résultant est formaté et ainsi un modèle d'optimisation PLNE est établi.

### V.4.5.2 Processus de transformation pour l'obtention d'un modèle de simulation discrétisé

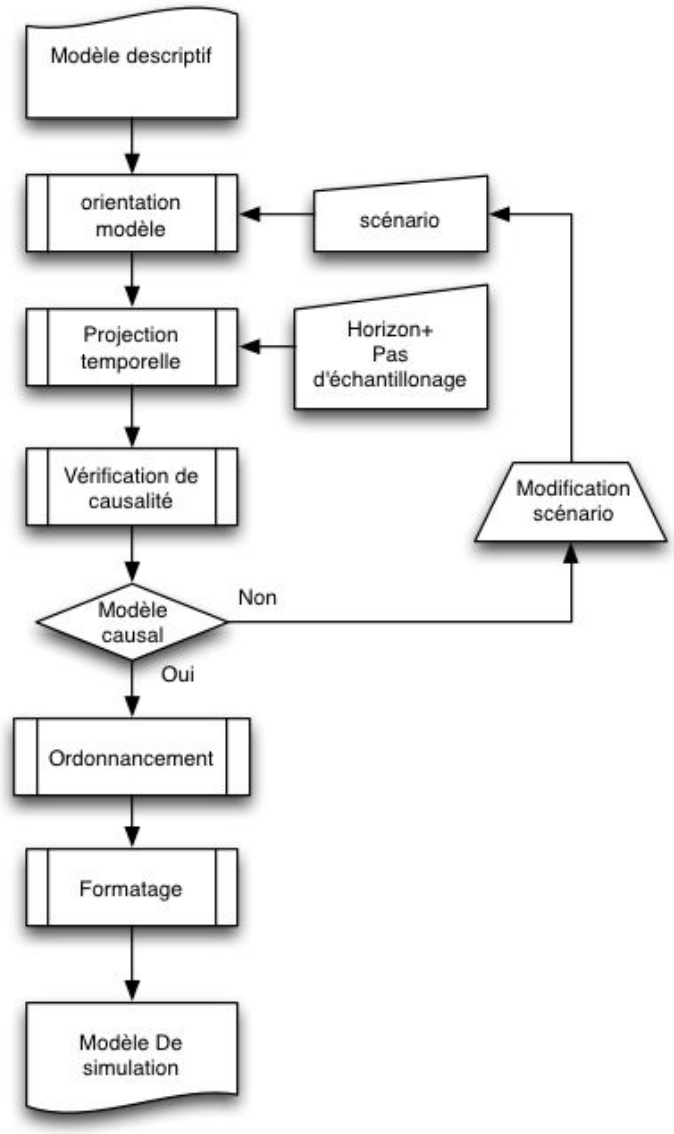



FIGURE *V.24 – Processus de transformation de modèle pour la simulation discrète*

Un modèle est dit simulable si celui-ci est juste-déterminé. Cette condition est liée au scénario associé. Le modèle est d'abord temporellement projeté. Ce choix n'est pas optimal en temps de calcul de causalité car le séquencement temporel est une causalité de fait. Néanmoins, nous avons préféré ignorer cette propriété et considérer l'ensemble des équations comme indépendantes afin de simplifier en réduisant les étapes nécessaires au traitement. Le développeur de modèle est associé à la vérification de la causalité en interagissant pour obtenir des informations en cas de problème. Une fois la causalité fixée, les équations sont ordonnancées avant d'être résolues séquentiellement par un simulateur ou un simple CAS si nécessaire.

### V.4.5.3 Processus de transformation pour l'obtention d'un modèle d'identification paramétrique

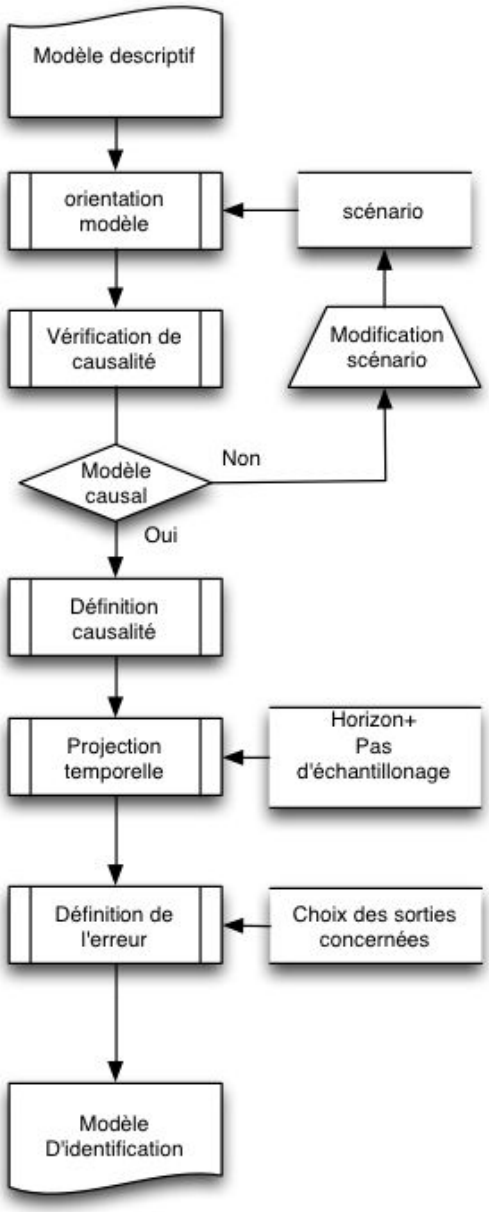



FIGURE *V.25 – Processus de transformation de modèle pour l'identification paramétrique*

L'estimation paramétrique est un moyen d'apprendre le comportement d'un système bâtiment en exploitant des mesures terrain pour calibrer des modèles (MOUNIER et al. 2014). Le système d'identification que nous avons choisi de mettre en place est simple. Son automatisation requiert une partie manuelle qui n'a pas pu être automatisée. Cette partie concerne les choix stratégiques pour les sorties mesurées étalons. Celles-ci orienteront la robustesse des paramètres du modèle par rapport aux erreurs d'approximation.

L'intervention manuelle pour le développement d'un modèle d'identification est sensiblement le même que pour un modèle de simulation à la différence du choix des grandeurs

à intégrer dans l'erreur à calculer.

## V.5 Détermination de la causalité d'un modèle : développement de la règle implémentée

La résolution d'un problème d'optimisation diffère de la résolution d'un problème de simulation par plusieurs aspects. Citons parmi ces aspects l'algorithme de résolution et la faisabilité du problème reçu. L'optimisation PLNE nécessite une a-causalité de modélisation alors que la simulation utilise des modèles causaux. Ces différences sont liées à la nature même des solveurs. Nous avons jugé qu'il était plus simple de transformer un modèle a-causal en modèle causal plutôt que l'inverse. Cette observation nous a mené à considérer les modèles descriptifs comme des modèles a-causaux. Ceux ci peuvent ensuite être spécialisés pour les applications nécessitant des modèles causaux à travers le processus d'établissement de causalité que nous énoncerons dans cette partie.

Ce processus répond à deux enjeux :

**existence de solutions pour la simulation** Un modèle est simulable si un jeu de sorties unique peut être associé à un jeu de données en entrée. Cette condition se traduit par la notion de juste détermination entre équations et variables. A contrario, un modèle est dit non simulable si celui-ci ne peut générer de solution unique. Le modèle est alors sur ou sous-déterminé. Lorsque le modèle est sur-déterminé, il ne peut pas y avoir de solution à cause de contraintes contradictoires dans le modèle. Un modèle sous-déterminé a plusieurs, voire une infinité, de solutions possibles à un même jeu de données. Cette situation rend le problème concrètement insolvable car il manque des données ou des contraintes pour choisir une solution parmi celles possibles. Dans ce cas de figure, le problème peut être contraint par des contraintes additionnelles ou alors par la conversion d'une ou plusieurs variables en paramètres à renseigner avant simulation.

**organisation des équations pour la résolution** La première étape dans la résolution d'un problème de simulation est le tri entre les contraintes égalité et inégalité. La méthode de résolution n'est pas la même pour la simulation. La première partie, celle des égalités, relève d'une résolution stricte alors que la seconde partie, celle des inégalités, relève d'une validation de résultat mais ne peut produire de résultat unique sauf par combinaison. Le tri des équations permet d'appliquer les méthodes de résolution adéquates pour chaque type. La seconde étape concerne la partie des équations égalité. La résolution séquentielle est progressive. Chaque étape de résolution ne peut utiliser que les informations déduite au moment de la résolution de l'étape. Il est nécessaire de mettre en place une hiérarchie de résolution afin de répondre à cette condition à chaque pas de résolution. Cet ordre existe si et seulement s'il existe une solution pour la simulation.

Les deux enjeux cités ci-dessus sont traités par une approche commune. Cette approche consiste à vérifier, et le cas échéant à déterminer, l'existence de cet ordre de résolution séquentiel.

Un modèle descriptif (composition de modèles descriptifs ou modèle élémentaire) embarque l'information commune à plusieurs applications. La spécialisation arrive en fonction de l'application dans laquelle le modèle sera utilisé. Cette spécialisation n'est pas une redéfinition du modèle mais un complément d'information permettant son usage

sur la base commune du modèle descriptif. Afin de passer du modèle descriptif à un modèle de simulation, les entrées, les paramètres et les sorties de celui-ci doivent être définis. Les entrées sont toutes les informations qui alimentent le modèle en temps réel. Les paramètres sont les informations constantes du modèle. Pour la détermination de la causalité, les deux types singleton (paramètres et entrées) sont gérés d'une manière similaire, ce sont des données. Les sorties sont les symboles considérés variables pour le solveur d'équations.

La classification des symboles fait partie de la procédure de spécialisation d'un modèle descriptif en modèle d'application. Cette spécialisation est réalisée sur la base des connaissances expertes du développeur de modèle. La causalité est une conséquence de cette spécialisation mais celle-ci n'est pas garantie. La causalité d'un système d'équations est liée à la juste-détermination du système d'équations égalités contenues dans le modèle.

### V.5.1 Définition des éléments de travail pour la décomposition Dulmage-Mendelsohn

Afin d'illustrer le travail de mise en évidence de la causalité, nous avons choisi de prendre le modèle V.6 comme exemple.

$$
\begin{aligned}
&contrainte_1 : \alpha_1 z + \beta_1 u = 0 \\
&contrainte_2 : \frac{\alpha_2}{x} + \frac{\beta_2}{y} + \gamma_2 u = 0 \\
&contrainte_3 : \alpha_3 z + \beta_3 u = 0 \\
&contrainte_4 : \alpha_4 y + \beta_4 z + \gamma_4 (v \times w) = 0 \\
&contrainte_5 : \alpha_5 w^2 + \beta_5 u = 0 \\
&contrainte_6 : \alpha_6 u^2 + \beta_6 v + \gamma_6 w = 0
\end{aligned}
\tag{V.6}
$$

La décomposition de Dulmage-Mendelsohn est un algorithme qui s'applique, à la base, sur les graphes bipartites **hopcroft1973n**

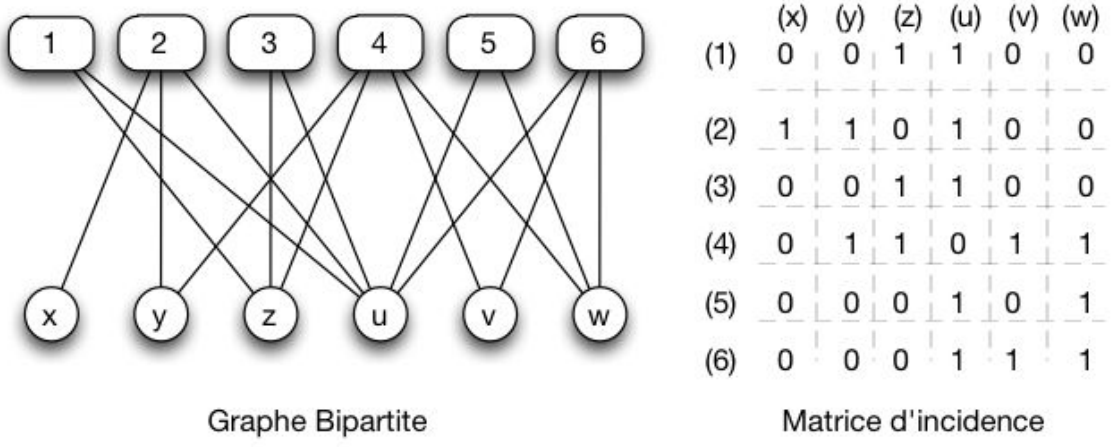



FIGURE *V.26 – représentation de l'exemple*

Un graphe bipartie est un graphe dont les sommets peuvent être décomposés en deux ensembles, où les arrêtes connectées aux éléments d'un ensemble ne sont jamais connectées à d'autres éléments du même ensemble. Un jeu d'équations égalité peut être modélisé sous la forme d'un graphe bipartite en considérant chaque équation comme un sommet de l'ensemble des équations et chaque variable comme un sommet de l'ensemble des variables. Les arrêtes entre les ensembles représentent alors l'implication des variables dans chacune des équations V.26.

Pour alléger l'algorithme, la représentation bipartie n'est pas utilisée. On lui préfère une représentation matricielle à travers la matrice d'incidence ou matrice creuse (sparse matrix). Cette matrice est composée de $0$ et de $1$ d'où son nom de matrice "creuse". Elle schématise l'implication de chaque variable (colonne) dans chacune des équations (lignes) par un $1$.

L'algorithme original de la décomposition Dulmage-Mendelsohn est basé sur cette modélisation dans laquelle il y a une hypothèse forte : celle de la déductibilité supposée des variables de toutes les équations. En réalité, pour la simulation, les équations ne sont pas forcément linéaires et peuvent utiliser des fonctions non inversibles, par exemple, ou induire des dérivations temporelles indésirables. Cette condition de déductibilité ne peut donc pas être garantie au risque de limiter l'utilisation de l'environnement de transformation à travers son application simulation. Nous avons proposé à travers cette thèse d'adapter l'algorithme à notre cas de travail. Cette adaptation passe par l'ajout de la possibilité de prise en compte de la non-déductibilité à travers une spécification dans la matrice d'incidence sous la forme d'un $-1$ qui correspond à une orientation partielle du graphe bipartite spécifiant que la variable concernée ne peut être déduite de la contrainte concernée.

### V.5.2 Les étapes de décomposition Dulmage-Mendelsohn

Un algorithme basé sur la décomposition de Dulmage-Mendelsohn permet de déterminer la causalité d'un système d'équations. Cette décomposition est basée sur la matrice d'incidence découlant de l'étape précédente. Il y a deux étapes proprement liées à la décomposition : la recherche d'un couplage maximal et la décomposition en blocs. L'ordonnancement est une étape supplémentaire qui découle de la réorganisation de la matrice d'incidence.

La première étape est de trouver un couplage maximal. L'algorithme Hopcroft–Karp est utilisé pour trouver ce couplage. Une fois ce couplage maximal déterminé, la classification entre éléments faisant partie du couplage et ceux ne faisant pas partie donne lieu à une réorganisation des équations et des variables en plusieurs groupes : VR, SR, HR, VC, SC, HC V.27.

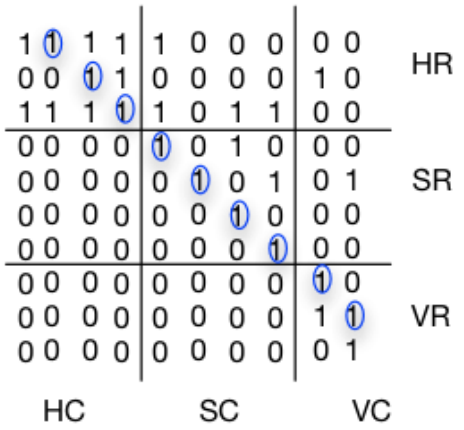



FIGURE *V.27 – Classification sur la base du couplage maximal*

Une fois le tri effectué, la causalité est alors vérifiée si et uniquement si les ensembles VR, HR, VC, HC sont vides. Autrement dit, toutes les contraintes et toutes les variables sont dans SR et SV respectivement. Dans le cas contraire, les ensembles non-vides renseignent sur les variables sous-déterminées et les contraintes sur-déterminantes. L'information peut être présentée au développeur de modèle qui reviendra sur les jeux de paramètres en décidant de chercher plus d'informations sur certaines variables ou contraintes, les fixer ou alors les relaxer.

### V.5.2.1 Recherche du couplage maximal

Pour expliquer la fonction et le fonctionnement de l'algorithme, il faut d'abord définir quelques notions. Pour commencer, un couplage est un sous-ensemble d'arêtes 1 à 1 entre nœuds variables et nœuds contraintes du graphe biparties qui n'ont pas de sommets en commun. Dans la matrice, ceci correspond directement à des entrées non-nulles qui ne sont ni sur la même ligne, ni sur la même colonne. Un sommet incident à une arête dans le couplage est dit couplé, sinon non-couplé.

Un chemin est une séquence de sommets (répétitions possibles) tel qu'un sommet est connecté au prochain par une arête. Dans la matrice, cela correspond à une ligne 'en escalier' allant d'une entrée non-nulle à une autre. Un chemin alternant est un chemin sans répétitions et dont les arêtes alternent entre ne pas être dans un couplage et l'être. Un chemin augmentant est un chemin alternant commençant et finissant sur des sommets non-couplés. Un couplage est maximum s'il a le plus grand nombre d'éléments possible. Dans la matrice, c'est la diagonale du plus grand nombre d'entrées non-nulles. Un couplage est complet sur les colonnes si tous les sommets correspondants aux colonnes sont couplés. Il est complet sur les lignes si ceci est vrai pour les lignes. Un couplage est complet s'il est parfait sur les lignes et les colonnes.

L'algorithme de recherche du couplage maximum consiste en la recherche d'un couplage quelconque $M$, d'abord, de manière mécanique, c'est-à-dire en vérifiant chaque colonne pour une entrée non-nulle et la couplant si elle n'occupe pas la même ligne qu'une entrée déjà couplée.

L'adaptation de l'algorithme à notre hypothèse de travail impose une contrainte supplémentaire dans la détermination d'un couplage. Les arêtes unidirectionnelles, marquées par $-1$ dans la matrice d'incidence ne peuvent être utilisées dans la recherche de couplage mais restent des liens entre deux sommets à prendre en compte.

L'algorithme cherche un chemin augmentant associé au couplage initial $M$ en y associant deux sommets ne faisant pas partie du couplage. Si un tel chemin existe, il suffit d'inverser l'inclusion dans le couplage des arêtes du chemin, obtenant ainsi un couplage plus grand d'une arrête $M'$. L'itération d'après le couplage $M'$ est utilisée à la place du chemin $M$ et la même opération est exécutée jusqu'à ce qu'il ne reste plus de sommets hors couplage ou alors ceux restants ne peuvent être associés à un couplage augmentant. Dans ce cas, le couplage est dit maximal.

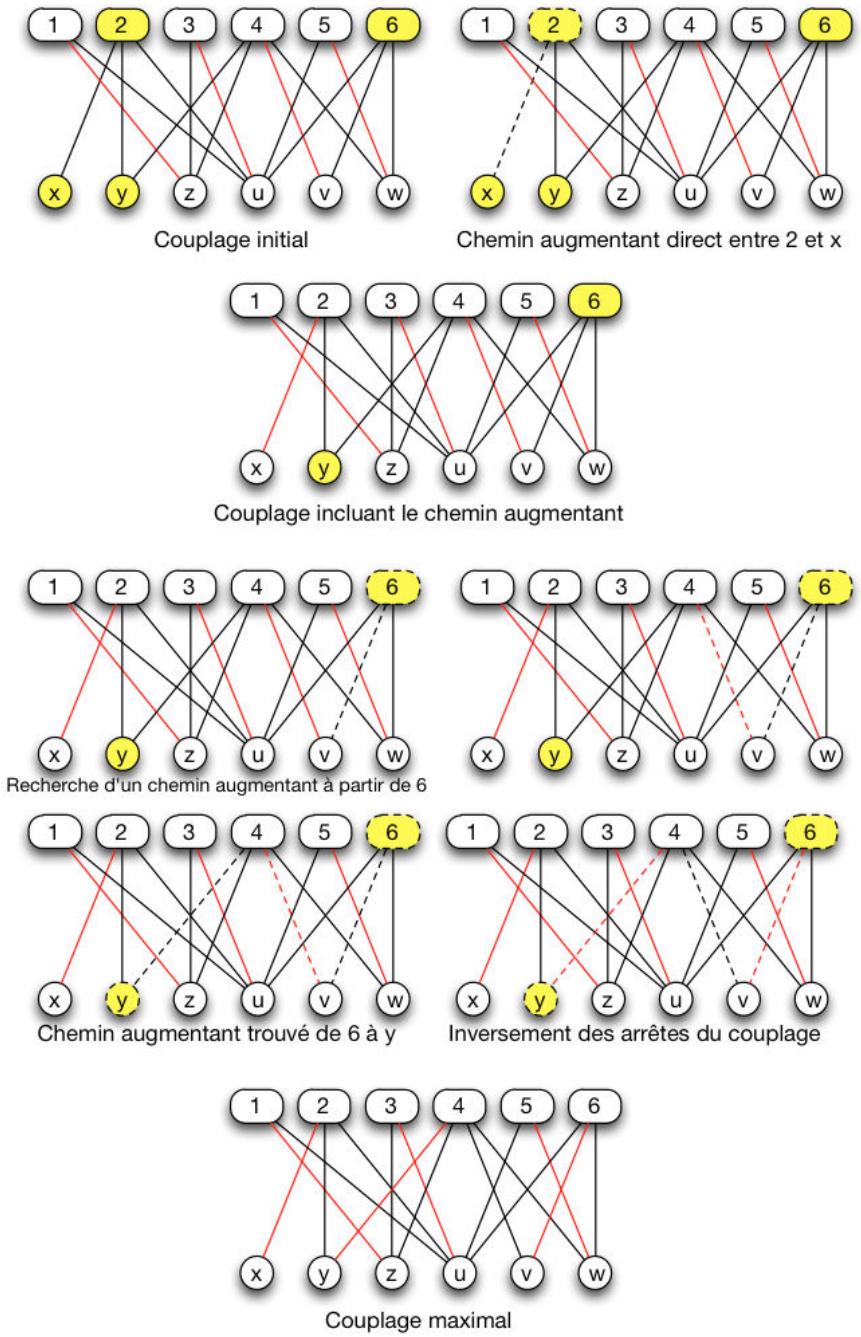



FIGURE *V.28 – algorithme de recherche du couplage maximal*

### V.5.2.2 Décomposition en blocs

La recherche d'un couplage maximal permet par la suite la décomposition proprement dite en blocs sous, juste et sur-déterminés. En effet, les trois parties du graphe obéissent à certains lemmes : leurs lignes et colonnes doivent être disjointes entre elles. Aucune colonne de la partie sous-déterminée n'est jointe à une ligne des autres parties, tandis qu'aucune ligne de la partie sur-déterminée n'est jointe à une colonne des autres parties. Dans la matrice, cela veut dire que moyennant une réorganisation, toute la partie sous le bloc sous-déterminé et à gauche du bloc sur- déterminé doit être remplie de zéros. Le choix du couplage maximum (s'il y en a plusieurs) n'a pas d'influence sur la décomposition résultante car elle est unique. La définition des blocs indique que les colonnes non-couplées doivent faire partie du bloc sous-déterminé et les lignes non-couplées du bloc sur-déterminé.

Ainsi, grâce aux lemmes, nous savons que toute ligne ou colonne atteignable à partir d'une colonne non-couplée est dans la partie sous-déterminée, et que toute ligne ou colonne atteignable à partir d'une ligne non-couplée est dans la partie sur-déterminée. Le restant est la partie juste-déterminée.

Cette dernière conclusion permet la mise en place de l'algorithme de tri. Il suffit d'effectuer un parcours alterné du graphe, à partir des éléments non couplés, par un algorithme d'exploration en largeur et marquer l'ensemble des sommets atteignables (sous-déterminé si à partir d'une colonne ne faisant pas partie du couplage maximal et sur-déterminé si le parcours commence par une ligne hors couplage maximal) comme dans l'exemple suivant (qui est une modification du premier exemple afin de ne pas le rendre juste déterminé)

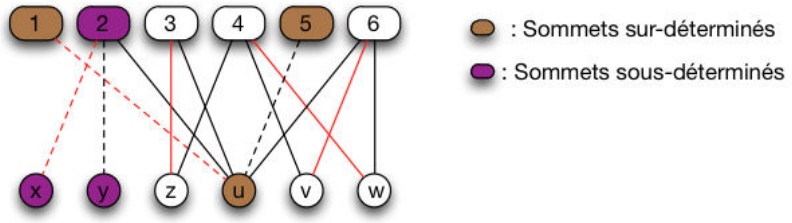



FIGURE *V.29 – Exécution de l'algorithme de décomposition en blocs*

### V.5.2.3 Ordonnancement des équations

Cette partie n'est utile dans le processus de détermination de la causalité que si le modèle est juste-déterminé. Pour ce faire, il nous est donc inutile de traiter les cas où les ensembles sur-déterminés et sous-déterminés ne sont pas vides.

Nous utilisons le couplage maximal pour trouver la matrice triangulaire supérieure. Le principe de l'algorithme est la réorganisation de la matrice en partant du coin inférieur droit qui représente la première étape de résolution à la première ligne en haut qui représente la dernière étape de résolution. A chaque itération, l'objectif est de déterminer l'arête (élément de la matrice) faisant partie du couplage maximal à mettre sur la diagonale. Cet élément doit faire partie d'une ligne dans laquelle tous les autres éléments non-nuls font partie de lignes dont l'élément de couplage a été déjà placé.

Dans certains cas, cette condition n'est pas vérifiable à cause de l'existence de couplage fort entre deux ou plusieurs éléments du couplage maximal. Ce cas se traduit par un bloc dans la diagonale de la matrice résultante. Il y a un classement ex æquo dans la

résolution qui est traduit par un bloc de résolution indissociable.

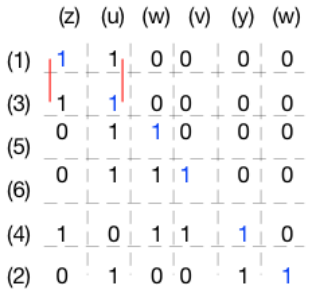



FIGURE *V.30 – Ordonnancement des équations*

Le processus mis en place pour la détermination d'une causalité dans un modèle permet la transition d'un modèle initialement non simulable - le modèle descriptif - vers un modèle de simulation proprement dit. La mise en place d'un modèle causal passe par la spécialisation d'un jeu de symboles en entrées (paramètres ou données dynamiques). Une fois la spécialisation mise en place, l'algorithme de Dulmage-Mendelsohn intervient dans un premier temps pour vérifier la simulabilité du modèle à travers la recherche d'un couplage maximal du système d'équations égalité. Le couplage maximal est complet lorsqu'il n'y a pas d'éléments sur-déterminés ou sous-déterminés. Si la condition n'est pas vérifiée, les ensembles sous-déterminés et sur-déterminés fournissent des informations pour orienter le développeur de modèle dans ses corrections et modifications. En cas de sur-détermination, l'algorithme met en évidence les équations en conflit. Dans le cas d'une sous-détermination, les variables ne pouvant être déduites par le processus de résolution sont mises dans l'ensemble des nœuds sous-déterminés. Le développeur de modèles peut alors prendre la décision de fixer certaines de ces variables ou alors reconsidérer le modèle dans son ensemble en ayant connaissance de la source du problème.

Une fois l'étape de validation de la causalité passée, l'étape de réorganisation des équations en vue de leur résolution est effectuée sur la même base du couplage maximal. Cette organisation est intrinsèque pour certains solveurs, néanmoins le travail de recherche du couplage maximal étant déjà réalisé, la solution la moins gourmande en ressource est d'utiliser l'information quitte à recréer une fonction existante.

## V.6 Conclusion

Nous avons développé dans ce chapitre une nouvelle approche pour répondre au problème des modèles dans un contexte multi-application. Nous proposons une solution qui permet de capitaliser l'information commune dans un modèle dit descriptif. Ce modèle est ensuite projeté vers des modèles spécialisés pour des applications précises. Les applications autour desquelles nous avons validé notre solution sont l'optimisation PLNE, la simulation et l'identification. Ces trois applications sont les principales applications de gestion énergétique de GHomeTech.

Le passage du modèle descriptif vers les modèles d'application est une tâche fastidieuse lorsqu'elle doit être réalisée manuellement et dont il existe des étapes répétitives. Nous avons retracé le processus manuel de chacune des transformations dans le chapitre précédent et nous avons proposé une solution de transformation semi-automatique.

Cette solution permet un gain de temps par rapport à une transformation complètement manuelle mais pas autant qu'un système complètement automatisé. L'automatisation entière des processus de transformation revient à ignorer les cas où il y a des problèmes de scénario d'usage ou d'orientation. Certains de ces cas interdisent l'obtention d'un modèle applicatif à partir du modèle descriptif. Un système entièrement automatisé conclura sur une impossibilité de délivrance de solution alors que la solution proposée permet des interactions pour trouver des solutions manuellement.

Néanmoins, il existe encore des limites. La linéarisation utilise des patterns. Ces patterns ne couvrent pas tous les types de non-linéarité. Les fonctions non-linéaires n'ont pas encore été implémentées. Dans les modèles traités actuellement, le besoin ne s'est pas encore manifesté ou alors a été contourné durant la phase de modélisation. Le développement du module de linéarisation a été motivé par la volonté de fournir une aisance et une liberté qui permettent au développeur de fournir les modèles les plus représentatifs possibles. Le traitement du produit continu-continu a lui aussi pas encore été traité. Pour l'instant, lors d'un cas de ce type, le développeur de modèle peut choisir de discrétiser une des deux variables afin de transformer le produit continu-continu en produit continu-discret.

Certaines procédures automatisées ne peuvent pas encore être aussi adaptées que les solutions manuelles. C'est le cas de certains patterns de linéarisation qui peuvent saturer les modèles applicatifs en nombre de variables binaires alors qu'il y a moyen, au cas par cas, d'en économiser certaines. Cela est une limite actuelle difficile à lever.

Enfin, notre travail a été développé pour des modèles purement déterministes. Il conviendrait, dans des travaux futurs, d'explorer les améliorations à mettre en œuvre pour intégrer des modèles statistiques dans le système.

# Chapitre VI

# Validation du process

Ce chapitre met en pratique des concepts énumérés sur la transformation automatique de modèles. Nous avons mis en oeuvre deux solutions logicielles concurrentes avec pour chacune ses propres spécificités. La première solution a été développée sous JAVA en utilisant la librairie de manipulation algébrique Xcas. La seconde solution a été mise en oeuvre sous python en utilisant la librairie Sympy. Le choix du modèle d'illustration s'est rapporté à Predis MHI pour lequel nous avons installé un système de récolte de données baptisé HAL (Home Abstraction Layer).

## VI.1 Implémentation informatique de la solution

L'implémentation des concepts développés dans le chapitre suivant a été réalisée dans l'environnement Python. Le choix de l'environnement s'est imposé d'abord par rapport à la richesse des librairies intégrées, de la facilité de sa prise en main et aussi par rapport à son faible typage. Cette dernière caractéristique nous a permis de proposer un langage de modélisation intuitif et ergonomique. Pour cela nous avons bâti une approche logicielle décrite dans le schéma VI.1

Le conteneur est l'élément central de l'implémentation. Celui-ci charge les informations contenues dans chaque modèle élémentaire et garde l'ensemble des transformations que le modèle y subit. La première étape consiste en la lecture et la composition. La lecture ou le terme anglais utilisé "parsing" consiste à transformer le contenu texte du modèle en un objet informatique pouvant être traité ultérieurement. Dans notre cas, le parsing est partiel car nous avons adopté pour stratégie d'utiliser des instanciations de classes dans les modèles. D'ailleurs, d'un point de vue informatique, un modèle ou une composition de modèles est une classe Python (figure VI.2). Dans cette classe, nous retrouvons des instanciations d'autres classes préalablement définies pour contenir chaque bribe du modèle. La déclaration d'un symbole (figure VI.3) est une classe *var* pour laquelle nous renseignons des paramètres de classe. Ces paramètres sont obligatoires pour certains comme la dénomination, et facultatifs pour d'autres tel que les bornes qui sont par défaut infinies. Vient ensuite la déclaration des contraintes à travers les mots clé *eqcon*, *lecon*, *gecon* comme présenté dans l'exemple (figure VI.4). Cette classe nécessite un parsage car l'équation est entrée sous forme de chaine de caractère afin de permettre un maximum de flexibilité.

La composition est le mécanisme de mise en commun de deux ou plusieurs modèles, qui peuvent être eux-mêmes des compositions. Une composition est une instantiation de modèles interdépendants. Les symboles des différents modèles sont connectés pour signifier que les grandeurs associées dans chaque modèle sont les mêmes. Il est aussi

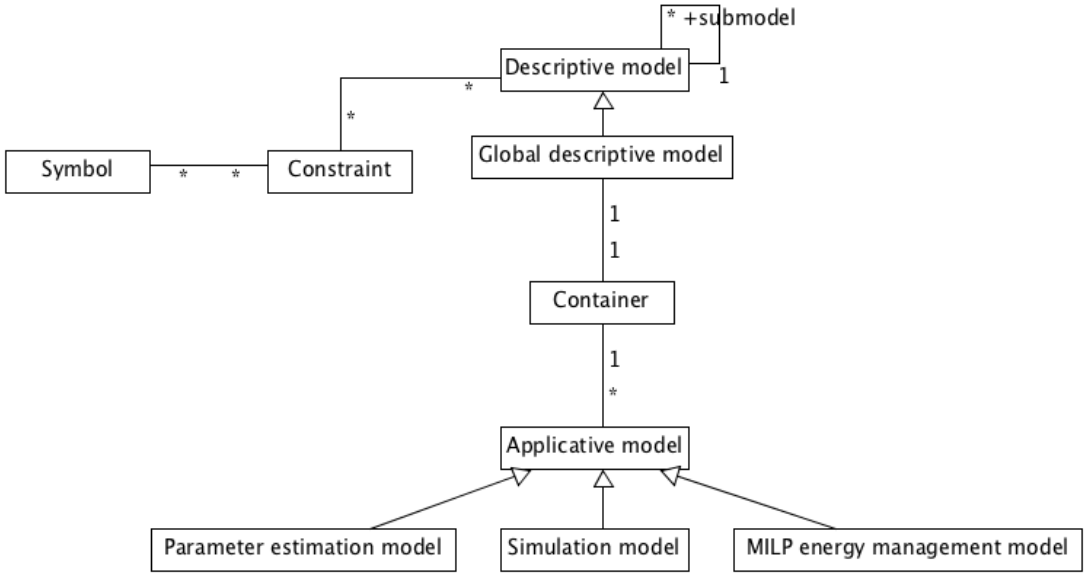



FIGURE *VI.1 – Schéma UML de la solution proposée*

```python
class Envelop(ElementModel):
    def __init__(self,elementName:str):
        "len variable: 18"
        "len constraint: 3"
        ElementModel.__init__(self, elementName)
        Tspace = self.var('Tspace',None,False,5,40)
        Toffice = self.var('Toffice',None,False,5,40)
        Tcorridor = self.var('Tcorridor',None,False,5,40)
        Tw = self.var('Tw',None,False,5,40)
        Tdown = self.var('Tdown',None,False,5,40)
        Tout = self.var('Tout',None,False,5,40)
        Tin = self.var('Tin',None,False,-400,400)
        phi_in = self.var('phi_in',None,False,0,2000)
        der_Tw = self.var('der(Tw)',None,False,4,40)
        HL = self.var('HL',None,False,1,1000)
        Hspace = self.var('Hspace',None,False,1,1000)
        Hoffice = self.var('Hoffice',None,False,1,1000)
        Hcorridor = self.var('Hcorridor',None,False,1,1000)
        Hventilation =self.var('Hventilation',None,True,0,280000,[0,60240,120480,180720,230920,271000])
        H0 = self.var('H0',None,False,1,1000)
        H1 = self.var('H1',None,False,1,1000)
        thermalCapacitance = self.var('thermalCapacitance',None,False,0,10^8)

        self.eqcon('SSRDynamicEquation',der_Tw*thermalCapacitance + H0*(Tw-Tdown),H1*(Tin-Tw))
        self.eqcon('SSROutputEquation',Hventilation*(Tin-Tout)+Hspace*(Tin-Tspace)+Hoffice*(Tin-Toffice)+Hco

        self.loadObjects(elementName)
```

FIGURE *VI.2 – Exemple de modèle descriptif sous python*

```python
Tcorridor = self.var('Tcorridor',None,False,5,40)
```

FIGURE *VI.3 – commande de déclaration d'un symbole*

```python
self.eqcon('SSRDynamicEquation',der_Tw*thermalCapacitance + H0*(Tw-Tdown),H1*(Tin-Tw))
```

FIGURE *VI.4 – commande de déclaration d'une contrainte*

possible d'intégrer d'autres symboles, contraintes ou objectif. Le scénario est dans la pratique associé aux compositions par soucis de clarté et d'organisation. Le scénario consiste

d'abord en le réajustement des bornes des symboles. Cet ajustement peut être nécessaire selon le cas d'application pour optimiser le temps de calcul d'une part et pour exprimer les contraintes liées à l'application en question d'autre part. La réduction de la borne supérieure à la borne inférieur revient à déclarer un symbole comme paramètre dans l'optimisation ou comme paramètre en simulation aussi. Le renseignement sous la forme d'un vecteur d'un symbole signifie que celui-ci est variable dans le temps. Ce qui revient à dire que le symbole est paramètre pour l'application optimisation et est entrée dans le cas simulation et estimation paramétrique.

La composition implique l'ajout de préfixes à l'ensemble des symboles et des contraintes de chaque modèle élémentaire selon les compositions dans lesquelles ils sont impliqués. Cette approche permet de rendre chaque symbole unique dans la composition finale qui peut faire appel à plusieurs instances d'un même modèle ou encore qu'il y ait des symboles similaires dans différents modèles. Le scénario proprement dit est aussi chargé à partir des fichiers de composition. Le scénario s'illustre par le resserrement des bornes des domaines de définitions voire les nature ou la temporalité. Dans le scénario, l'orientation de chaque symbole est aussi définie selon l'application.

Les contraintes sont chargées dans le conteneur sous forme d'arbres de représentation. Le choix de cette représentation XML par rapport à d'autres types de représentations répond à l'impératif d'adaptabilité avec d'autres outils parmi lesquels les CAS. Les règles sont implémentées de façon à s'appliquer sur ces arbres de représentation. Une règle traite chaque contrainte et ses symboles séparément. Le résultat du traitement est rechargé dans le conteneur à la place du précédent contenu jusqu'à l'application de la prochaine règle. La règle qui s'appliquera l'itération d'après s'appliquera sur le même conteneur. On peut enclencher un système d'archivage de l'historique du conteneur avant transformation mais cela augmente les temps de traitement.

Enfin, le séquencement des règles est défini dans la partie processus de transformation (figure VI.5). Cette partie constitue le centre de contrôle du process de transformation depuis les modèles descriptifs contenus dans les classes établies par le développeur de modèles jusqu'au modèle applicatif qui est transféré directement à l'application concernée. En ce sens nous rappelons que le travail présenté a vocation à être intégré à l'outil de gestion énergétique GHomeTech. Cela implique que le résultat de la transformation est un objet informatique dont on ne peut faire l'illustration dans ce manuscrit.

```
class PredisSimulatorValidation():
    def __init__(self,modelInstance,dictionaryOfVariablesWithValuesSubstitution):
        predisModel=modelInstance

        #Time projection

        timerProblemTool = TimeDiscretization(predisModel.container)
        timerProblemTool.problemTimeDisretization()

        #Constraints sort
        sortTool = SortTool(predisModel.container)
        sortTool.sortEquations()
        #Dulmage Mendelsohn part

        dulmageMendelsohnSystemBuilder=SystemFactory()
        var=predisModel.container.getDulmageMendeshonModelForm()
        for form in var :
            print('%%%%%%%',form,'%%%%%%%%%',var[form])
        dulmageMendelsohnSystemBuilder.setModel(predisModel.container.getDulmageMendeshonModelForm())
        self.system=dulmageMendelsohnSystemBuilder.getSystem()
        #print(self.system)
        print('underdetermined variables')
        print(self.system.getUnderdeterminedVariables())
        print('underdetermined equations')
        print(self.system.getUnderdeterminedEquations())
        print('justdetermined variables')
        print(self.system.getJustdeterminedVariables())
        print('justdetermined equations')
        print(self.system.getJustdeterminedEquations())
        print('overdetermined variables')
        print(self.system.getOverdeterminedVariables())
        print('overdetermined equations')
        print(self.system.getOverdeterminedEquations())
        if self.system.getUnderdeterminedEquations() is None and self.system.getOverdeterminedVariables() i
            causalModel=dulmageMendelsohnSystemBuilder.getCausality()
            return(causalModel)
```

FIGURE *VI.5 – processus de transformation vers un modèle de simulation*

## VI.2 Modélisation descriptive de PREDIS MHI

PREDIS est un centre d'innovation, de formation et de valorisation dans le secteur de l'énergie qui offre un support d'expérimentation pour la recherche, l'enseignement et l'industrie [PRED 2012][9]. Implanté sur 2500 m2 de locaux de l'école ENSE3 (école de l'énergie, l'eau et l'environnement), ce centre regroupe un ensemble de plateformes technologiques thématiques. Les axes de recherche autour de ces plateformes se répartissent sur trois principaux thèmes selon les équipements installés :

**La production décentralisée** : une centrale de cogénération à base de micro turbine à gaz, une pile à combustible et deux ensembles de 10 panneaux photovoltaiques sont à disposition.

**Les réseaux intelligents** : un simulateur temps réel hybride, un réseau industriel local (échelle 1/10ème), un réseau de distribution (échelle 1/1000ème) et un système de conduite et de supervision (SCADA) sont installés.

**Le bâtiment intelligent** :c'est la plate-forme PREDIS MHI que nous détaillerons plus bas.

La plateforme PREDIS MHI est située au premier étage du bâtiment PREDIS UHT en occupant une surface de 300 m2. La plate-forme est constituée d'une salle informatique pour les cours, un espace bureau dédié à la recherche et un local technique. Le bâtiment PREDIS UHT se trouve à l'intérieur d'un hangar au toit translucide et aux parois non-isolées. C'est un bâtiment existant qui a été rénové en ne gardant que la structure du

bâtiment précèdent. Les parois, les planchers du RDC et du premier étage ainsi que le toit intérieur ont été re-dessinés et ré-étudiés, le but étant d'aboutir à un bâtiment à la fois passif (par ses pertes limitées à travers l'enveloppe) et basse consommation (par la consommation de ses équipements).

L'équipement principal, en terme de consommation, composant cet ouvrage est un système de ventilation double flux, installé au niveau de la salle PREDIS MHI. Celui-ci permet un complément d'isolation actif de l'enveloppe, ce, en canalisant au maximum l'échange d'air avec l'extérieur par le biais de ce système. Ce moyen permet aussi un contrôle et un monitoring plus strict des échanges de flux d'air et de chaleur. Ce même système intègre un chauffage par eau chaude qui circule à travers des cassettes (échangeur eau/air) en contact avec le flux d'air entrant vers les salles.

Un système de récupération de calories appelé échangeur statique rotatif est intégré dans la boucle afin de recycler une partie des calories perdues pendant le renouvellement d'air.

Il est à noter qu'il n'existe aucune autre ouverture, vers l'extérieur à la portée des usagers, que les portes d'accès vers chacun des deux compartiments : salle de cours et l'espace de bureaux. Il y a aussi une porte communicante entre les deux salles et une baie vitrée de séparation phonique mais dont l'isolation thermique n'est pas optimale.

L'instrumentation du système de ventilation intègre des capteurs de flux et de température des courants d'air circulants. Il intègre aussi des actionneurs déviateurs de flux d'air et des vannes d'eau commandables pour le réseau d'eau chaude.

Dans les deux salles, un réseau de capteur d'ambiance (température, hygrométrie), des prises commandables équipées de watt-metre sont installées. Un capteur de CO2 et des capteurs de position des portes sont installés dans chacune des salles.

L'ensemble des données capteurs et statuts des actionneurs remonte à un centraliseur d'information : HAL. Le centraliseur "HAL" (pour Home Abstraction Layer) est un logiciel informatique développé par nos équipes qui permet de centraliser les informations remontées sous différents protocoles via des pilotes dédiés. HAL met à disposition l'ensemble des informations récoltées dans un serveur web sous forme d'une arborescence XML mise à jour en temps réel (latence négligeable par rapport à la fréquence de remontée de l'information).

La salle ainsi dotée d'un monitoring complet permet la création d'une base de données sur le comportement de l'enveloppe. Le système offre aussi des degrés de liberté pour la gestion énergétique à travers le système de conditionnement de l'air et des prises commandables.

### VI.2.1 Modèle du système de conditionnement de l'air

Le système a pour leviers de commande le débit d'air, la quantité de chaleur injectée à travers la cassette d'eau chaude, et un ensemble d'aiguillages des débits pour avoir des différents modes de ventilation. Le développement de modéles descriptifs consiste à formaliser les liens entre les degrés de liberté et les données en entrée nécessaires à la description des phénomènes intervenant. Nous rappellons que l'objectif premier des modèles est de fournir l'information nécessaire à la gestion énergétique uniquement. Les autres application sont considérées annexes à la gestion. C'est à partir de ce paradigme que l'ensemble des modèles sont construits.

HVAC

$$\rho_{Air} \times \phi_{air} \times cp_{air} \times (T_{in}^{exch} - T_{outdoor}) = \eta_{pass}^{exch} \times \phi_{air} \times Cp_{Air} \times (T_{in}^{room} - T_{out}^{exch}) \quad \text{(VI.1)}$$

$$\begin{aligned} P_{heating} =& \phi_{air} \times Cp_{Air} \times (T_{pulse} - T_{out}^{exch}) \times Mode_{recovery} + & \text{(VI.2)} \\ & \phi_{air} \times Cp_{Air} \times (T_{pulse} - T_{outdoor}) \times Mode_{shortcut} + & \text{(VI.3)} \\ & \phi_{air} \times Cp_{Air} \times (T_{pulse} - T_{in}^{room}) \times \times Mode_{loop} & \text{(VI.4)} \end{aligned}$$

avec :

$$T_{outdoor} \times Mode_{recovery} = T_{in}^{exch} \times Mode_{recovery} \quad \text{(VI.5)}$$

$$\begin{aligned} R_{Ventilation} =& Mode_{recovery}/((1-\eta) \times Cp_{Air} \times \rho_{Air} \times \phi) + & \text{(VI.6)} \\ & Mode_{shortcut}/(Cp_{Air} \times \rho_{Air} \times \phi) + & \text{(VI.7)} \\ & Mode_{loop} \times Value_{inf} & \text{(VI.8)} \end{aligned}$$

$$Mode_{shortcut} + Mode_{recovery} + Mode_{loop} = 1 \quad \text{(VI.9)}$$

avec :

$$\begin{aligned} Mode_{shortcut} : & \quad Binary & \text{(VI.10)} \\ Mode_{recovery} : & \quad Binary & \text{(VI.11)} \\ Mode_{loop} : & \quad Binary & \text{(VI.12)} \\ & & \text{(VI.13)} \end{aligned}$$

$$P_{ventilation} = \phi_{air} \times (w_{shortcut} \times Mode_{shortcut} + w_{recovery} \times Mode_{recovery} + w_{loop} \times Mode_{loop}) * P_{nominal} \quad \text{(VI.14)}$$

$$P_{airTreatementUnit} = P_{ventilation} + P_{heating} \quad \text{(VI.15)}$$

$$E_{airTreatementUnit} = \int_{t1}^{t2} P_{airTreatementUnit} \, \mathrm{d}t. \quad \text{(VI.16)}$$

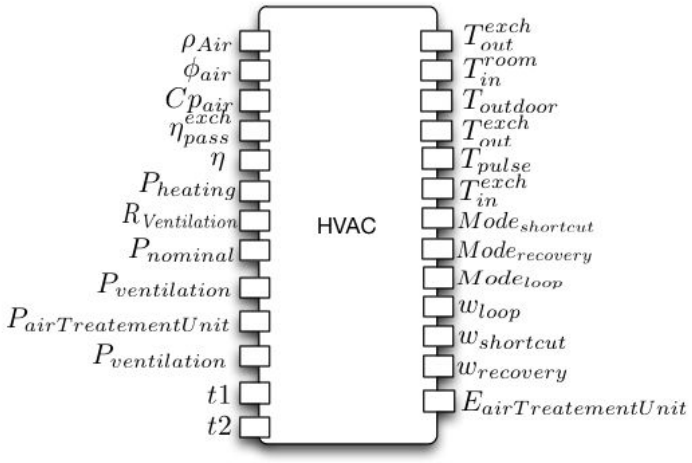



FIGURE *VI.6 – Modèle descriptif du système de conditionnement de l'air*

### VI.2.2 Modèle des balances énergétiques

energy balance

$$Phi_{Total} = Phi_{Sun} + Phi_{heating} + Phi_{Occup} \tag{VI.17}$$

$$P_{total} = P_{airTreatementUnit} + P_{lighting} + P_{computer} \tag{VI.18}$$

$$Total_{cost} = PricePerKwh \times \int_{t1}^{t2} P_{total}\,\mathrm{d}t. \tag{VI.19}$$

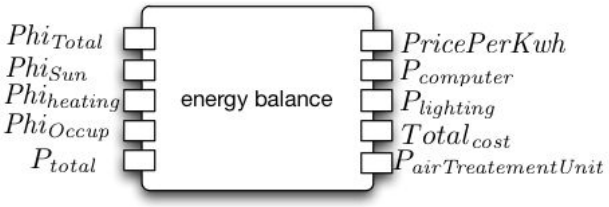



FIGURE *VI.7 – Modèle descriptif des balances énergétiques*

### VI.2.3 Modèle du confort thermique

confortThermique

*If presence* $= 1$ : (VI.20)

$$T_{felt} < T_{pref} \Rightarrow sigma_{incomfort} = 1/(T_{pref} - T_{max}) \times T_{felt} - T_{pref}/(T_{pref} - T_{max})$$

&

$$T_{felt} \geq T_{pref} \Rightarrow sigma_{incomfort} = 1/(T_{max} - T_{pref}) \times T_{felt} - T_{pref}/(T_{max} - T_{pref})$$

(VI.21)

*If presence* $= 0$ :

$$sigma_{incomfort} = 0$$

&

$$T_{felt} <= T_{max_{absence}} \tag{VI.22}$$

&

$$T_{felt} >= T_{min_{absence}} \tag{VI.23}$$

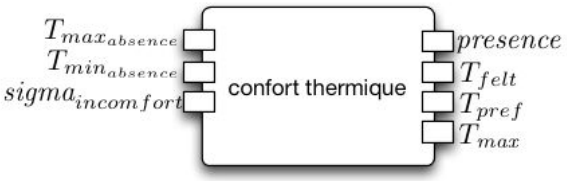



FIGURE *VI.8 – Modèle descriptif du confort thermique*

### VI.2.4 Modèle de l'enveloppe thermique

enveloppeThermique

$$R_{Eq} = 1/(1/(R_{Ventilation} + R_w) + \sum(1/R[neighborhood])) \tag{VI.24}$$

$$\begin{aligned}\frac{d}{dt}T_w = &- 1/(R_{Eq} \times C_w) \times T_w + 1/((R_{Ventilation} + R_w) \times C_w) \times T_{out} \\ &+ \sum(T[neighborhood]/(R[neighborhood] \times C_w)) \\ &+ R_{Ventilation} \times Phi_{total}/(C_w \times (R_{Ventilation} + R_w))\end{aligned} \tag{VI.25}$$

$$\begin{aligned}T_{In} = &R_{Ventilation} \times T_w/(R_{Ventilation} + R_w) \\ &+ R_w/(R_{Ventilation} + R_w) \times T_{Out} + \\ &R_{Ventilation} \times R_{Eq} \times Phi_{total}/(R_{Ventilation} + R_w)\end{aligned}$$

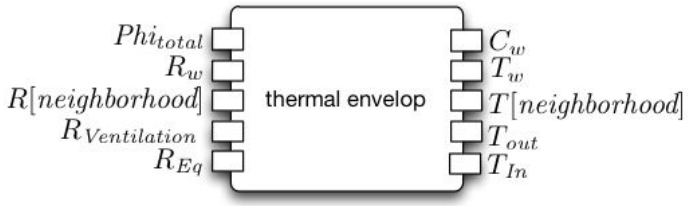



FIGURE *VI.9 – enveloppeThermique*

### VI.2.5 Modèle de l'évolution du $CO_2$

EvolutionCO2

$$\frac{dC_{inCO_2}}{dt} = \left(\frac{Q_{input} \times C_{outdoor}}{V_{space}} + \frac{S}{V_{space}}\right) - \frac{Q_{output}}{V_{space}} \times C_{inCO_2} \tag{VI.26}$$

avec :

- $Q_{input}$ : débit volumétrique d'air frais entrant
- $Q_{output}$ : débit volumétrique d'air rejeté
- $V_{space}$ : volume de l'espace
- $C_{inCO_2}$ : concentration intérieure du $CO_2$ à un instant donné

$C_{outdoor}$ **:** concentration extérieure de $CO_2$

$S$ **:** taux de génération de $CO_2$, masse par unité de temps à partir de l'occupation ou d'autres sources.

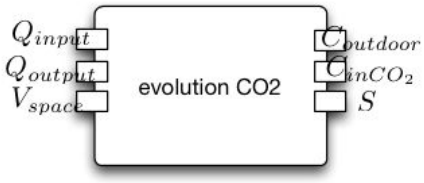



FIGURE *VI.10 – Modèle descriptif de l'évolution du $CO_2$*

### VI.2.6 Modèle du confort $CO_2$

confortCO2

$$sigma_{CO_2} = (C_{CO_2} - C_{fav})/(C_{max} - C_{fav}) \tag{VI.27}$$


FIGURE *VI.11 – Modèle descriptif du confort $CO_2$*

## VI.3 Composition

Nous aurions pu envisager la composition de deux manières différentes :

**Composition de modèles descriptifs** Cette composition présente l'avantage d'être au même niveau de développement que les modèles descriptifs, lesquels sont introduits par l'utilisateur. La composition à ce niveau offre plus d'ergonomie dans la mesure où le développeur de modèles devra manipuler les symboles qu'il a créés pour les modèles descriptifs.

**Composition de modèles applicatifs** Ce type de composition permet un gain en volume de traitement dans la phase de transformation. La transformation de chaque modèle descriptif se fait une seule fois. Les compositions des modèles applicatifs interviennent par la suite sur les modèles résultants. Cette approche présente néanmoins l'inconvénient d'être fastidieuse pour le développeur de modèles car implique de travailler avec les symboles projetés et les symboles créés par le système de linéarisation. D'autre part, en suivant ce choix, il faudra reprendre la composition pour chaque application et ainsi ne pas capitaliser cette partie du travail.

Nous avons adopté le premier type de composition dans l'approche logicielle que nous avons développée en Python et en Java.

Pour la composition du modèle PREDIS MHI, un fichier spécifique de composition VI.12 est intégré à la bibliothèque de modèles. Considérée comme modèle descriptif à part entière, la composition fait appel à d'autres modèles à travers le mot clé $import$.

# descriptive models instantication
EB=import ($energy_balance$)
Hvac=import ($HVAC$)
TC=import ($confortthermique$)
TE=import ($thermalenvelop$)
CO2Conf=import ($confortCO2$)
CO2Evol=import ($evolutionCO2$)
# connexions :
connect($TE.phi_{total}$, $EB.phi_{total}$ )
connect($TE.R_{ventilation}$,$Hvac.R_{ventilation}$)
connect($TE.T_{in}$,$Hvac.T_{in}^{room}$,$TC.T_{felt}$)
connect($CO2Evol.C_{inCO_2}$,$CO2conf.C_{CO2}$)
connect($CO2Evol.Q_{input}$,$CO2Evol.Q_{output}$,$Hvac.\phi_{air}$)
connect($EB.phi_{Total}$,$TE.phi_{total}$)
connect($EB.phi_{heating}$,$Hvac.P_{heating}$)
connect($EB.P_{airTreatementUnit}$,$Hvac.P_{airTreatementUnit}$)

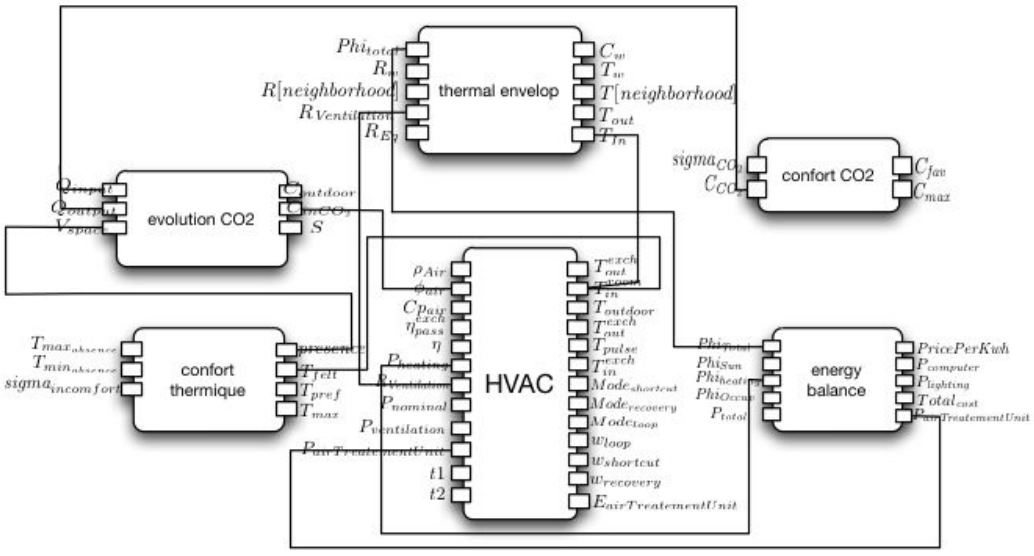



FIGURE *VI.12 – Composition pour PREDIS MHI*

## VI.4 Application gestion énergétique PLNE

Etant donné que l'exécution des parties automatiques ne produit pas de résultats intermédiaires lisibles, nous nous focaliserons sur certains mécanismes pour les expliquer en détail par des illustrations. Les interventions de l'utilisateur dans le processus seront aussi développées.

La première étape consiste à fournir au processus de transformation les éléments de départ schéma VI.13 que sont le modèle descriptif et le scénario d'usage.

Le modèle descriptif de Predis-MHI est contenu dans la composition (schéma VI.12).

Le scénario choisi pour notre étude est fonction des degrés de liberté effectivement possibles sur la plateforme.

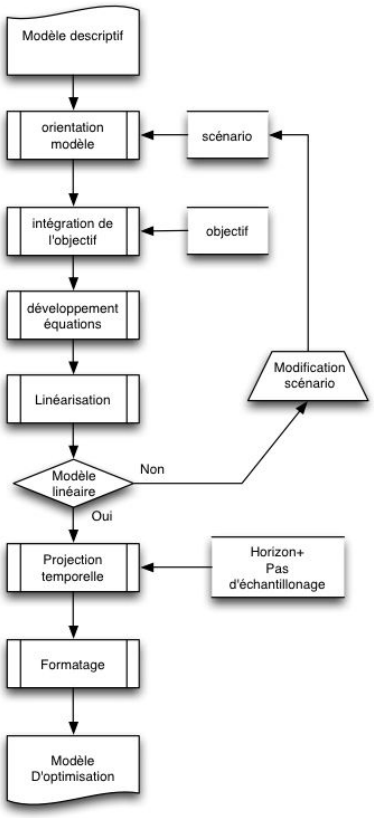



FIGURE *VI.13 – Processus de transformation de modèle pour l'optimisation PLNE*

# parameter definition $Hvac.\rho_{Air} = Value$
$Hvac.\phi_{air} = Value$
$Hvac.cp_{air} = Value$
$Hvac.\eta_{pass}^{exch} = Value$
$Hvac.T_{outdoor} = Value$
$Hvac.P_{nominal} = Value$
$Hvac.Value_{inf} = Value$
$Hvac.w_{shortcut} = Value$
$Hvac.w_{recovery} = Value$
$Hvac.w_{loop} = Value$
$Hvac.t1 = Value$
$Hvac.t2 = Value$
$EB.Phi_{Sun} = Value$
$EB.PricePerKwh = Value$
$EB.t1 = Value$
$EB.t2 = Value$
$TC.presence = Value$
$TC.T_{pref} = Value$
$TC.T_{max} = Value$
$TC.T_{max_{absence}} = Value$
$TC.T_{min_{absence}} = Value$
$TE.C_w = Value$
$TE.R_w = Value$
$TE.R[neighborhood] = Value$
$TE.T_{out} = Value$
$CO2Conf.C_{fav} = Value$
$CO2Conf.C_{max} = Value$
$CO2Evol.V_{space} = Value$
$CO2Evol.C_{outdoor} = Value$
# variables définition $Hvac.T_{in}^{room}$
$Hvac.T_{out}^{exch}$
$Hvac.T_{pulse}$

$Hvac.T_{in}^{exch}$
Binary : $Hvac.Mode_{recovery}$
Binary : $Hvac.Mode_{shortcut}$
Binary : $Hvac.Mode_{loop}$
$Hvac.P_{heating}$
$Hvac.P_{airTreatementUnit}$
$Hvac.P_{ventilation}$
$Hvac.E_{airTreatementUnit}$
$Hvac.R_{Ventilation}$
$EB.Phi_{Total}$
$EB.Phi_{heating}$
$EB.Phi_{Occup}$
$EB.P_{total}$
$EB.P_{airTreatementUnit}$
$EB.P_{lighting}$
$EB.P_{computer}$
$EB.Total_{cost}$
$TC.T_{felt}$
$TC.sigma_{incomfort}$
$TE.R_{Eq}$
$TE.T_{w}$
$TE.R_{Ventilation}$
$TE.T[neighborhood]$
$TE.Phi_{total}$
$TE.T_{In}$
$TE.T_{Out}$
$CO2Conf.sigma_{CO_2}$
$CO2Conf.C_{CO_2}$
$CO2Evol.Q_{input}$
$CO2Evol.Q_{output}$
$CO2Evol.CinCO_2$

S'ensuit l'introduction de l'objectif à optimiser :

# additional constraints objective : minimize Z

$$Z = sigma_{CO2} + sigma_{incomfort} + Total_{cost} \tag{VI.28}$$

Une fois les données rassemblées et l'objectif d'optimisation fixé, le traitement du modèle commence par une instanciation des données dans les équations puis la simplification et le développement de celles-ci.

Prenons l'exemple de l'équation VI.29 de qualité de l'air pour illustrer le travail sur le modèle.

$$sigma_{CO_2} = (C_{CO_2} - C_{fav})/(C_{max} - C_{fav}) \tag{VI.29}$$

$C_{max}$, $C_{fav}$ sont des paramètres et par conséquent l'expression $(C_{max} - C_{fav})$ est aussi paramètre.

La simplification puis le développement de l'équation donnent :

$$sigma_{CO_2} = \frac{1}{(C_{max} - C_{fav})} \times C_{CO_2} - \frac{C_{fav}}{(C_{max} - C_{fav})} \tag{VI.30}$$

La linéarisation est l'étape suivante dans le processus de transformation du modèle. Nous prendrons un cas de non linéarité qui n'a pu être traité automatiquement car il n'existe pas de pattern de traitement.

La linéarisation est l'étape suivante dans l'ordre du processus. Celle-ci a été largement discutée dans sa phase automatique au cours du chapitre II. L'étape consiste en la recherche de non-linéarités puis leur traitement automatique grâce aux patterns de linéarisation. L'algorithme de linéarisation est basé sur la recherche de non-linéarités prédéfinies. En ce sens, l'approche est perfectible car celle-ci ne garantit pas la linéarité absolue du modèle résultant mais garantit plutôt la non présence de certaines formes de non-linéarités prédéfinies par avance dans l'algorithme de recherche.

Dans le modèle de Predis MHI, l'ensemble des non-linéarités est détecté. Leur traitement suit les étapes du module développé dans Milp-workshop pour le traitement des non-linéarités.

Néanmoins, toutes les NL ne peuvent pas être traitées. Nous prenons comme exemple le modèle d'évolution du taux de $CO2$. Après simplification et réorganisation des équations nous avons la forme de l'équation VI.31.

$$\frac{dC_{inCO_2}}{dt} = -\frac{1}{V_{space}} \times Q_{output} \times C_{inCO_2} + \frac{Q_{input} \times C_{outdoor}}{V_{space}} + \frac{S}{V_{space}} \tag{VI.31}$$

Cette forme présente une non linéarité de type continu-continu $Q_{output} \times C_{inCO_2}$. Cette non-linéarité ne peut être traitée par les mécanismes automatiques. Le développeur doit alors traiter manuellement celle-ci. La solution conseillée est alors de discrétiser une variable continue afin de la rendre linéaire en considérant des paliers fixes pour le flux d'air possible. L'optimisation ne pourra choisir que dans un ensemble de valeurs restreintes le flux d'air renouvelé $Q_{output}$.

Le changement s'effectue en revenant au fichier de spécialisation et en déclarant la variable $Q_{output}$ comme "discrète". Cela implique d'initialiser les débits possibles sur un ensemble de valeurs ponctuelles : $0, 125m3/h, 250m3/h, 500m3/h, 1000m3/h$.

Le mécanisme de connexion des variables facilite la tâche. Il suffit d'effectuer le changement dans la composition en déclarant une restriction sur le symbole $Q_{output}$. Ainsi, toutes les variables connectées à $Q_{output}$ sont impactées.

La projection temporelle consiste à activer un mécanisme de démultiplication des éléments du modèle en fonction du pas d'échantillonnage et de l'horizon de temps choisis. Les variables ainsi que les équations et autres déclarations sont démultipliées pour chaque pas de temps. Les dérivées et les intégrales sont aussi traitées dans cette approche afin d'avoir un modèle complètement discrétisé à la fin car c'est le seul format de représentation possible pour un problème temporel dans le formalisme PLNE.

Nous illustrons cette étape sur le modèle PREDIS MHI sur une partie du modèle l'équation VI.31 car celui-ci devient très dense à cette étape. Cette partie contient une dérivée et le pattern de dérivation lui est donc appliqué. Nous remarquons aussi qu'il y a une forme de représentation d'état dans l'équation caractérisé par la variable d'état $C_{inCO_2}$. L'application du pattern de discretisation de la dérivée revient à considérer une discrétisation de l'équation d'état avec bloqueur d'ordre zéro (ROBINSON 2002). Le résultat du traitement est présenté dans l'équation VI.34 avec $t$ l'indice du pas de temps. Le résultat présenté est alors démultiplié pour chaque pas de temps sur l'horizon de travail entier.

$$\frac{C_{inCO_2}(t+1) - C_{inCO_2}(t)}{Te} = -\frac{1}{V_{space}} \times Z^{Q_{output}}_{C_{inCO2}}(t) + \frac{Q_{input}(t) \times C_{outdoor}}{V_{space}} + \frac{S(t)}{V_{space}} \tag{VI.32}$$

avec :

$$Z^{Q_{output}}_{C_{in_{CO2}}}(t) = Q_{output}(t) \times C_{inCO_2}(t) \tag{VI.33}$$

puis :

$$\begin{aligned} C_{inCO_2}(t+1) =& C_{inCO_2}(t) - \frac{Te}{V_{space}} \times Z^{Q_{output}}_{C_{in_{CO2}}}(t) + \frac{Te}{V_{space}} \times C_{outdoor} \times Q_{input}(t) \\ &+ \frac{Te}{V_{space}} \times S(t) \end{aligned} \tag{VI.34}$$

La dernière étape du processus avant la résolution consiste à construire un problème d'optimisation PLNE avec sa partie déclarative, ses contraintes égalités et inégalités, la déclaration de l'objectif et des données du problème. Cette étape dépend de l'interface utilisée avec le solveur PLNE. Globalement, l'ensemble des interfaces nécessite un conditionnement particulier des équations car celles-ci ne doivent être que sous forme polynomiale de premier ordre. Le travail dans cette partie consiste donc à assainir les équations de façon à les exprimer sous cette forme particulière. Le travail d'assainissement consiste à développer des équations puis, si nécessaire, la reformulation de l'équation de façon à éviter les divisions arithmétiques.

Le travail complémentaire dans cette étape est d'associer à chaque variable une borne supérieure et une borne inférieure en adéquation avec les contraintes de connexion. Ces bornes sont uniformisées lors de la connexion.

L'ensemble des données paramètres du problème est aussi rassemblé dans cette partie pour accompagner la structure du problème.

Après obtention du modèle PLNE, son exécution a donné comme résultat les courbes VI.14

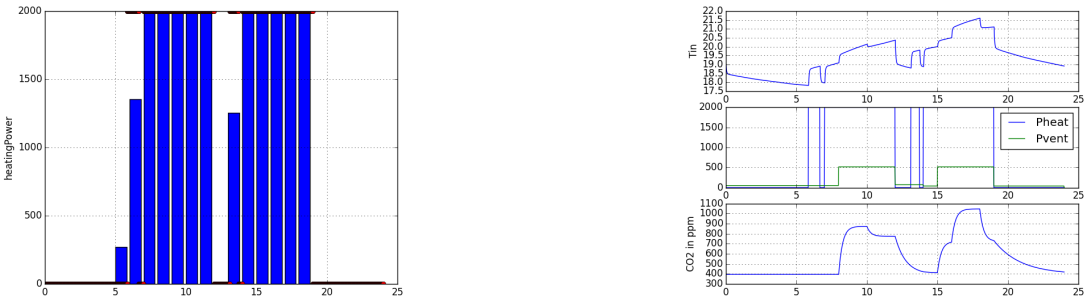



FIGURE *VI.14 – Plan initial avec les commandes initiales*

## VI.5 Application simulation

La simulation dynamique a des contraintes de modélisation différentes que celles de l'optimisation PLNE. La linéarité des modèles n'est plus une exigence dans la modélisation. Par contre la notion de causalité est une condition sine qua non pour la simulation d'un modèle. Cette condition garantit l'existence et l'unicité d'une solution au problème de simulation posé sur n'importe quel solveur.

Le travail de projection du modèle descriptif vers l'application simulation consiste essentiellement à traiter la question de causalité. La composition du modèle Predis MHI est la même que pour la partie optimisation PLNE. Le scénario, par contre, est fixé selon l'objectif de la simulation. Le scénario est ce qui constitue la causalité du modèle. Les algorithmes que nous appliquons ne font que vérifier la causalité des modèles de simulation et y définissent un ordre de résolution pour les équations.

La simulation a été utilisée dans l'algorithme du recuit-simulé tout comme elle a été utilisée en interaction et conseil à l'occupant. Dans ce cas, l'occupant, lui-même, constitue le scénario. Ce cas d'utilisation illustre parfaitement le besoin d'automatisation du traitement de la causalité car l'occupant n'est pas sensé être expert en modélisation. Les informations introduites peuvent donc être erronées ou incomplètes. Le système doit être en mesure de détecter les problèmes qui peuvent survenir et idéalement fournir des indices pour résoudre ces problèmes.

Dans ce qui suit, nous allons citer quelques exemples de scénarios envisagés durant nos tests et nous présenterons aussi les résultats de vérification de causalité.

**Example 1** Le premier exemple que nous choisissons est un exemple dans lequel nous voulons fixer à zéro les consommations électriques du système HVAC tout en essayant de jouer sur la température intérieure et en exigeant une qualité de l'air minimale (à travers un taux de CO2 maximum). Le schéma VI.15 montre par une encoche noire les symboles fixés en paramètres et par une encoche transparente les symboles maintenus variables pour la simulation.

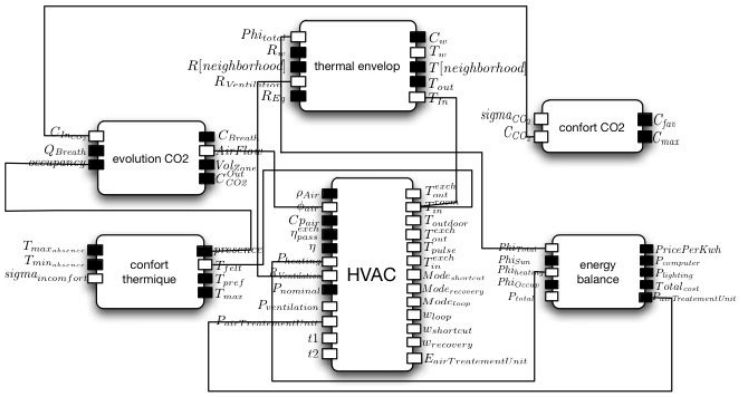



FIGURE *VI.15 – Exemple 1*

La vérification de causalité montre une sur-détermination du modèle orienté. En l'état, ce modèle ne peut être donc simulé car il n'y pas de solution.

**Example 2** Dans ce second exemple, nous choisissons d'avoir comme degrés de liberté l'ensemble des commandes de la VMC tout en permettant l'intégration de charges extérieures non controllée VI.16.

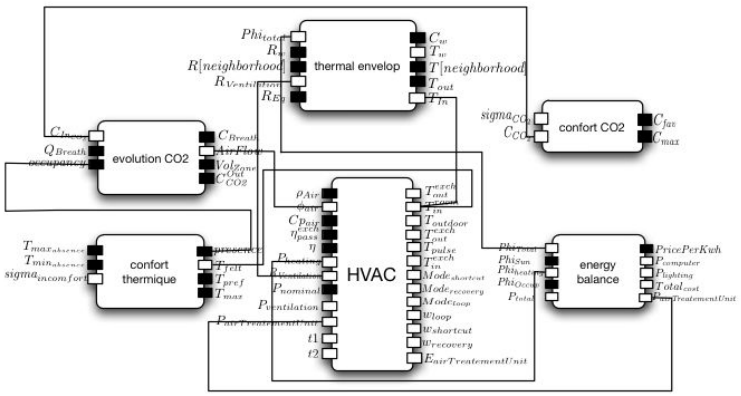



FIGURE *VI.16 – exemple 2*

La vérification de causalité de ce système montre une sous détermination du problème. Cette sous détermination conduit à une multitude de solutions possibles. Nous rappelons que la simulation n'a pas d'exigence d'optimalité et que, par conséquent, il n'y a pas de préférence de solution selon l'objectif choisi.

**Example 3** Dans ce cas, nous avons choisi de garder comme degrés de liberté les variables de contrôle de la VMC comme dans le schéma VI.17.

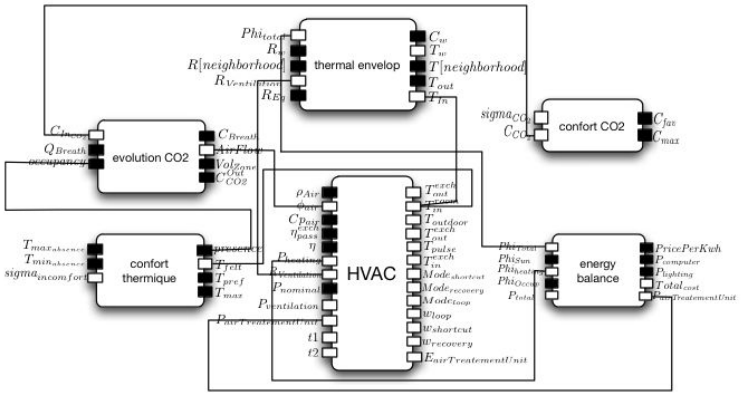



FIGURE *VI.17 – exemple 3*

Le résultat de l'étude de causalité montre que le système est juste déterminé et ainsi pouvant être simulé.

### VI.5.1 Utilisation des modèles de simulation dans SA

L'approche recuit simulé a été envisagée dans l'optique de fournir un outil d'optimisation complémentaire à l'optimisation PLNE.

L'initialisation pour l'optimisation par recuit simulé (en anglais Simulated annealing SA) est requise pour une efficacité de l'algorithme. Une solution initiale trop éloignée de l'optimum risque de perdre l'algorithme SA car celui-ci ne présente aucune garantie de convergence en temps acceptable surtout pour un problème multi dimensionnel.

En terme d'optimisation, le processus d'interaction est une succession de modifications mineures du problème initialement résolu. SA permet alors de rechercher une solution au problème posé par interaction à partir de la solution du problème initial. La

procédure d'interaction impose une première proposition de la part de la machine. Cette proposition est fournie par la couche prédictive du système GhomeTech grâce à l'optimisation PLNE. Cette proposition est ensuite discutée, modifiée ou validée par les utilisateurs à travers l'IHM pour adapter les contraintes du problème aux besoins de l'occupant.

Nous prenons comme exemple l'IHM développée pour le prototype CANOPEA tout en soulignant que les travaux développés dans cette thèse sont plus génériques. L'ensemble des interactions proposées par l'IHM de CANOPEA peut être techniquement interprété sous trois formes :

— L'ajout d'une ou plusieurs contraintes dans le problème d'optimisation à traiter.
— Le changement de valeurs de paramètres et donc d'ajustement de données existantes.
— La manipulation des compositions de modèles mais cette approche d'interaction parait complexe à mener par un utilisateur non expert.

L'approche d'optimisation par recuit simulé est une méthode d'exploration stochastique de solutions à partir d'une solution initiale dans un environnement mono-objectif. Cette approche n'est pas exhaustive dans la mesure ou elle traite des espaces infinis d'une manière semi-aléatoire contrairement à l'approche PLNE où l'exploration se fait à travers les sommets d'un polytope représentant l'ensemble des solutions aux limites jusqu'à l'optimum.

L'utilité de l'optimisation SA est principalement l'exploration d'un voisinage d'une solution existante et le temps de calcul paramètrable en fonction du besoin. Cette approche offre la possibilité de choisir le compromis acceptable entre optimalité de la solution et temps de calcul. L'algorithme SA est un algorithme glouton itératif lequel est sensé s'améliorer continuellement au cours des itérations. Il y a un mécanisme de diversification fonction de la température de recuit. Mettre comme condition d'arrêt du processus le temps de calcul permet donc de privilégier le temps de calcul à la qualité de résultat. De plus, l'algorithme permet au pire de garder la solution de départ s'il n'y a pas d'améliorations trouvées.

L'idée de coupler l'algorithme avec un solveur PLNE vient du constat de proximité de solution dans le cas de changements minimes. Cette approche est pertinente en interaction car l'utilisateur, en interagissant avec la machine, a comme premier besoin des résultats rapides à des changements mineurs. Il pourra toujours ré-enclencher une résolution complète PLNE si ces résultats ne lui conviennent pas en terme de qualité. Cette résolution prendra néanmoins plus de temps.

Pour la mise en oeuvre, le couplage entre les approches peut se faire à travers la plateforme de manipulation de modèles. SA n'a pas besoin d'un processus de transformation dédié. En effet, l'algorithme SA est basé sur de la simulation récursive. En ce sens, il suffit alors de déterminer un scénario associé au modèle descriptif pour avoir le modèle de simulation. Celui-ci peut être alors déployé dans l'environnement de simulation appelé par SA.

#### VI.5.1.1 L'algorithme d'optimisation SA

L'algorithme du recuit simulé tire son nom de l'industrie métallurgique dans laquelle a été observé l'amélioration des performances grace à des recuits (réchauffement de la matière à certains paliers de température).

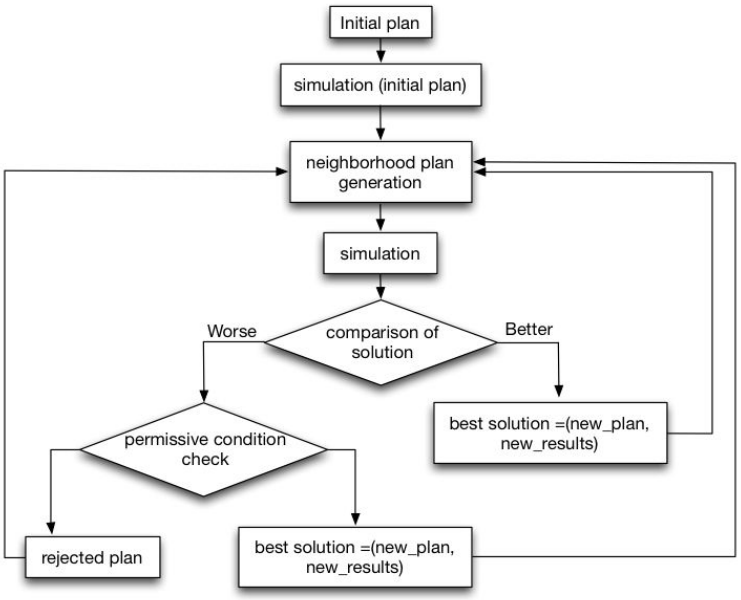



FIGURE *VI.18 – Algorithme du recuit simulé*

L'algorithme d'optimisation SA est basé sur la simulation récursive d'une représentation virtuelle de l'enveloppe associée à ses équipements avec calcul d'un objectif. L'algorithme SA itère des simulations avec des jeux d'entrées définis par l'algorithme lui-même. Celui-ci sélectionne à chaque itération la base de calcul des entrées de la prochaine itération en choisissant entre la simulation courante et la simulation retenue précédemment dans un espace de stockage tampon que nous appelons *buffer*.

Le choix de cette sélection est basé sur la comparaison entre la valeur de l'objectif courant et l'objectif de la solution retenue dans le buffer. Si l'objectif courant est meilleur que l'objectif de la simulation contenue dans le buffer alors le nouveau résultat est retenu, le buffer est mis à jour. Dans le cas contraire le résultat n'est pas systématiquement rejeté mais est soumis à une condition dite de permissivité donnée par l'inégalité VI.36 à travers la fonction de permissivité donnée par l'équation VI.35.

Fonction de permissivité :

$$f(T_n, Obj_{new}, Obj_{best}) = exp(\frac{Obj_{new} - Obj_{best}}{T_n}) \tag{VI.35}$$

Condition de permissivité :

$$N(P) \leq N(f(T_n, Obj_{new}, Obj_{best})) \tag{VI.36}$$

avec

$$P = T_n \times rand(0, 1) \tag{VI.37}$$

$N$ : pour valeur normalisée $0 \leq N(f) \leq 1$

La condition de permissivité consiste à comparer une variable aléatoire $P$ à la valeur retournée par la fonction de permissivité. Cette dernière dépendant de la variable température $T_n$, à ne pas confondre avec les températures dans l'enveloppe thermique ou d'autres modèles, et de l'écart entre les deux objectifs comparés.

L'association entre condition et fonction de permissivité consiste à positionner l'état d'avancement itératif de l'algorithme SA en jouant le rôle d'entonnoir. Au début de l'algorithme, la permissivité aux objectifs moins performants est plus grande. Cette approche permet une exploitation large de l'espace de recherche. Plus l'algorithme itératif se rapproche de la limite du nombre d'itérations maximal plus l'entonnoir se resserre et l'algorithme devient plus sélectif en terme de solutions acceptées.

La variable température $T_n$ évolue selon une fonction exponentielle du nombre d'itération. Cette variable donne la forme entonnoir à la condition de permissivité et caractérise l'approche du recuit-simulé.

$$T_n = T_0 \times \alpha^n \tag{VI.38}$$

avec : $\alpha$ and $T_0$ : définissent l'incurvation de la courbe

L'autre point important du recuit simulé concerne la recherche de nouveaux jeux de données. Comme expliqué au départ, le recuit simuté de l'objectif atteint durant l'une des précédentes itérations et ses jeux de données correspondants. Le nouveau jeu de données est fonction du choix de solution courante retenue et de la fonction d'exploration. La fonction d'exploration définit le jeu de données à tester pour l'itération d'après dans un voisinage de la solution courante.

Le problème de la dimension de l'espace d'exploitation se pose dans une optimisation plus large que l'exemple traité comme c'est le cas à travers Predis MHI. Chaque degré de liberté du problème d'optimisation (variable de simulation) constitue une dimension supplémentaire dans l'espace de recherche. L'ordre d'exploration des différents degrés de liberté doit obéir à un ordre de déroulement afin de trouver l'approche la plus optimale dans le temps imparti.

Nous avons recensé trois approches d'exploration :

**Traitement séquentiel des variables** Cette méthode consiste à explorer chaque dimension du problème d'optimisation en fixant les autres. Cette méthode est efficace lorsque les variables sont complètement décorrélées.

**Traitement global des variables** Il est question de développer une fonction de calcul de voisinage pouvant fixer pour chaque itération l'ensemble des degrés de liberté pour l'itération suivante. Cette méthode peut être complètement aléatoire ou bien complètement déterministe mais ne peut pas offrir un compromis intéressant entre les deux

**Traitement des variables par grappes** Le traitement de variables par grappe consiste à exploiter la particularité des problèmes d'optimisation pour la gestion énergétique. Les degrés de liberté représentent des variables physiques projetées temporellement. Certains degrés de liberté appartiennent donc à des familles la variable de base avant projection temporelle. Le traitement de ces variables devient plus simple car il s'agit d'une réduction virtuelle de la dimension de l'espace de recherche. D'un point de vue SA, chaque grappe représente alors un seul degré de liberté. La fonction d'exploration est adaptée localement à chaque type de grappe en fonction du type de variable, du nombre d'éléments dans la grappe et de la connaissance experte liée à l'évolution de ces variables et au ciblage préalable des optimums.

Le processus de projection du modèle descriptif pour l'optimisation SA est le processus de projection vers la simulation. Un scénario d'utilisation fixe est décrit pour cette

partie. Dans ce scénario, nous considérons que les degrés de liberté sont le taux de ventilation et la température ambiante. Le taux de ventilation est considéré discret dans notre cas d'utilisation afin d'illustrer l'optimisation avec des variables discrètes et non par obligation de linéarité qui n'existe pas dans cette application. La température sera considéré continue.

L'initialisation de l'algorithme s'est faite à travers un résultat dans les mêmes conditions et avec le même modèle que celui de l'application PLNE (figureVI.14).

L'objectif utilisé pour générer la solution PLNE est composé, contrairement à l'objectif fixé dans SA. L'objectif composé est une agrégation de plusieurs objectifs à travers des pondérations dans une une seule variable à optimiser. L'objectif économique consiste à demander à l'outil de gestion la rationalisation de l'énergie. L'objectif confort sert à rapprocher le scénario proposé vers les critères de confort idéaux en terme de qualité de l'air (taux de CO2) et de température idéale. Les objectifs sont agrégés en un seul via des pondérations personnalisables en fonction du besoin de l'occupant. La personnalisation de ces pondérations constitue un enjeu majeur dans la réussite et l'acceptation du plan proposé à l'occupant car il est très difficile voir impossible d'anticiper les préférences de l'occupant. Celui-ci dépend de son humeur, de ses intentions non énoncés à l'outil voir dans certains cas ou le fait même d'observer les différentes situations. En guise d'illustration, nous avons choisi de mettre à zéro les pondérations des objectifs économiques. Cette différence vis à vis du problème de départ permet d'illustrer un souhait de l'utilisateur post génération de plans sous PLNE.

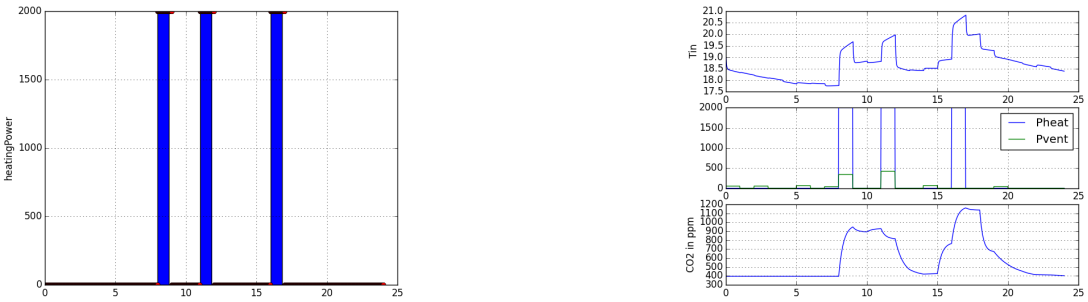



FIGURE *VI.19 – Plan d'usage résultant de l'execution de l'algorithme du recuit simulé (1/2)*

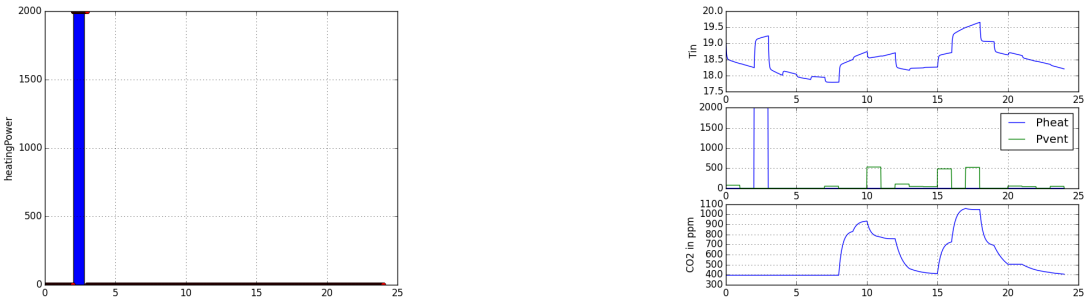



FIGURE *VI.20 – Plan d'usage résultant de l'execution de l'algorithme du recuit simulé (2/2)*

Les résultats (figure VI.19) montrent une différence dans la consommation de chauf-

fage entre le plan initial et les plans SA à travers deux instances d'optimisation. L'optimum absolu pour la question posée est l'arrêt total du chauffage sur l'horizon de simulation. Celui-ci n'est atteint dans aucune des deux instances mais l'instance (figure VI.20) se rapproche de cette solution. La solution s'est donc améliorée sans atteindre l'optimum en un temps limité, ce qui est recherché à travers la démarche SA.

La transformation de modèles a permis pour l'exemple un gain de temps dans le passage du modèle d'optimisation au modèle de simulation via le modèle descriptif. Le processus de transformation vers un modèle de simulation a été utilisé pour le scénario prédéfinit. C'est la seule tâche manuelle dans le cas présent.

La source étant commune à l'optimisation PLNE et à l'optimisation SA, il devient alors plus crédible de considérer les résultats de l'optimisation PLNE comme point d'initialisation pour SA. Le résultat était évident, extinction totale du chauffage, selon les hypothèses posées mais qui ne pouvait être décelé par l'algorithme directement. Néanmoins le point de départ issu de l'optimisation PLNE a rapproché la solution dans les deux instances. Ce point de départ a été probant car les modèles PLNE et SA découlent de la même base commune : le modèle descriptif.

## VI.6 Conclusion

Les résultats présentés dans ce chapitre ont été conformes à nos attentes en terme de systématisation. Le processus entier n'a pas été automatisé mais les tâches que nous avons jugés répétitives l'ont été : la linéarisation, la projection temporelle, la génération de causalité et la manipulation d'équations. Ces transformations ne sont pas encore exhaustives en terme de traitement de particularités et de fonctions spécifiques. Nous avons apporté la preuve de faisabilité à travers notre système. Le temps de traitement des tâches de transformation a été considérablement réduit.

L'approche semi-automatique de résolution permet de pallier à ces manques. La flexibilité et l'unicité du modèle descriptif sont aussi d'une aide conséquente dans notre travail. Cela permet de transformer le modèle vers les formalismes d'applications selon des règles prédéfinies et communes pour l'ensemble des modèles. Cette approche permet de combiner les résultats de chacune des applications de manière cohérente. Pour générer les résultats ci-dessus dans l'optimisation PLNE et recuit simulé, nous nous sommes basés sur le seul modèle descriptif.

Notre approche consiste à définir chaque transformation, depuis le modèle descriptif vers le modèle d'application. Elle permet de générer des modèles pour d'autres applications en utilisant les outils implémentés.

Nous avons construit le cahier de charge de ce qui a été développé pour cette thèse par rapport aux besoins liés à l'optimisation anticipative rencontrés dans le projet CANOPEA. Nous avons testé notre concept sur le modèle de PREDIS MHI. L'approche par recuit simulé a été introduite pour faciliter la manipulation systématique de modèles de simulation.

# Conclusion Générale

Cette thèse apporte des éléments de solution à la problématique de complexité qui revêt 2 aspects : la complexité liée au nombre d'éléments dans un modèle et celle liée à la diversité des applications cibles.

Pour la gestion de la complexité liée au nombre, nous avons présenté un ensemble de solutions issues de la littérature scientifique pour la gestion et la manipulation de modèles dans différents domaines afin d'établir un panorama de l'existant. Pour l'optimisation PLNE, la complexité apparaît nettement à travers le nombre de variables et de contraintes à appréhender. L'optimisation PLNE est une approche d'optimisation globale, qui requière donc un unique modèle du problème à résoudre. Nous avons proposé un principe de composition d'éléments de modèles conduisant progressivement à un modèle global. Chaque élément de modèle, lorsqu'il est habilement formulé, peut être réutilisé dans d'autres compositions de modèles. Il s'agit alors de constituer une bibliothèque de modèles réutilisables moyennant des adaptations de valeurs de paramètres. La complexité liée au nombre rend délicat les étapes de transformation, tels que l'application de patterns de linéarisation, du fait la démultiplication des transformations de modèles dans un contexte de projection temporelle. Nous avons proposé une systématisation des processus de transformation.

Une implémentation informatique de l'outil de composition a été validé sur le prototype de bâtiment CANOPEA : il s'appuie sur une logique de modélisation inspiré d'autres language à l'instar de Modelica. La modélisation de CANOPEA s'est avérée efficace de par son évolutivité durant le processus d'élaboration. Il s'agissait de pouvoir s'adapter à chaque changement décidé avec un minimum d'effort et de temps de développement. La composition a rendu aisé des modifications partielles du modèle. L'outil a aussi permis le test de plusieurs modélisations de l'enveloppe thermique afin d'obtenir un bon compromis entre le degré de précision du modèle d'enveloppe et la complexité calculatoire induite par un surcroît de précision.

La seconde partie de cette thèse traite de la gestion de la complexité liée à la diversité des applications de gestion énergétique. Les besoins de modèles pour les différentes applications ont été mis en évidence. Les applications simulation, estimation paramétrique et optimisation PLNE sont examinées à travers un exemple. Différentes natures de modèles apparaissent : acausal/causal, linéaire/non-linéaire et orienté paramètres/orienté variables. La problématique de transformation de modèles descriptifs vers des modèles pour différentes applications de gestion énergétique s'appuie sur la formalisation *model driven architecture*. Une méta-modélisation des différents types de modèles applicatifs est proposée selon les besoins constatés. Les principes de transformation sont énoncés suivant les méta-modèles. Le principe de base de l'approche proposée représente chaque grandeur intervenant dans le modèle initial par un symbole, qu'il s'agisse d'une variable physique ou d'un paramètre de modèle. Les transformations se traduisent par un ensemble de recettes s'appuyant sur un outil de calcul symbolique. Le modèle descriptif est le socle

commun permettant de générer les modèles d'applications dont le méta-modèle, représentant les éléments possibles d'un type de modèle, est défini. Pour déduire un modèle applicatif à partir du modèle descriptif, il suffit alors d'appliquer le processus de transformation adéquat. Le processus de transformation fait appel à une série de manipulations formelles, de spécialisation, d'ordonnancement causal,...mais aussi la composition, la linéarisation et la projection temporelle.

Les processus de transformation ont été validés sur la plateforme PREDIS MHI. Les transformations d'un modèle descriptif vers un modèle d'optimisation PLNE puis vers un modèle de simulation pour le recuit simulé sont détaillées. Les transformations ont été développées dans l'optique de mettre en place un gestionnaire énergétique capable d'interagir avec les occupants.

L'interaction est une perspective à ce travail de thèse. Cette dernière requiert une manipulation aisé de modèles pour recourir selon la nature de l'interaction, à des applications de simulation/re-simulation, de calcul de stratégies optimisées pour le futur mais aussi dans le passé, d'estimation de variables physiques non-mesurées,... Le passage de l'optimisation PLNE avec un modèle linéaire en nombres entiers, à un recuit simulé s'appuyant sur un modèle de simulation, permet d'embarquer dans un smart-device des algorithmes d'ajustement de stratégies optimisées à des demandes occupants collectées à travers des interactions. Le passage entre les modèles de simulation, d'optimisation, d'estimation paramétrique permet de créer une continuité entre la phase de développement et la phase d'exploitation. Aujourd'hui ce lien n'existe pas : le modèle de simulation thermique dynamique de conception est souvent peu représentatif du comportement réel du bâtiment en phase exploitation. Un re-calibrage des paramètres est indispensable.

Une autre perspective ouverte par cette thèse concerne le conseil à travers les modèles qualitatifs. Il s'agit de déduire via un processus de transformation, simple à mettre en oeuvre dans notre approche, le modèle symbolique pour en déduire les causalités permettant aux occupants d'un bâtiment de mieux s'approprier les stratégies énergétiques suggérées. Ces causalités pourront être alors traduites en un langage simple pour démontrer à l'occupant l'impact de ses décisions sur le système global, rendant ainsi plus persuasives les stratégies proposées.

# Bibliographie


AFGAN, Naim H et Maria G CARVALHO (2002). “Multi-criteria assessment of new and renewable energy power plants”. In : *Energy* 27.8, p. 739–755.

— (2004). “Sustainability assessment of hydrogen energy systems”. In : *International Journal of Hydrogen Energy* 29.13, p. 1327–1342.

AFGAN, Nain H et Maria G CARVALHO (2008). “Sustainability assessment of a hybrid energy system”. In : *Energy Policy* 36.8, p. 2903–2910.

AGGARWAL, Sanjeev Kumar, Lalit Mohan SAINI et Ashwani KUMAR (2009). “Electricity price forecasting in deregulated markets : A review and evaluation”. In : *International Journal of Electrical Power & Energy Systems* 31.1, p. 13–22. ISSN : 01420615. DOI : 10.1016/j.ijepes.2008.09.003. URL : http://linkinghub.elsevier.com/retrieve/pii/S0142061508000884.

ALLAIN, Loig (2003). “CAPITALISATION ET TRAITEMENT DES MODELES POUR LA CONCEPTION EN GENIE ELECTRIQUE”. In :

ALLEN, B (1996). “An integrated approach to smart house technology for people with disabilities”. In : *Medical engineering & physics* 18.3, p. 203–206.

ALLISON, James T (2004). “Complex System Optimization : A Review of Analytical Target Cascading , Collaborative Optimization , and Other Formulations by by”. In :

ANG, Kiam Heong, Gregory CHONG, Student MEMBER et Yun LI (2005). “PID Control System Analysis , Design , and Technology”. In : 13.4, p. 559–576.

ARI, S, I A COSDEN et C ISIK (2005). “Constrained Optimization”. In : p. 500–504.

BADREDDINE, Rim (2012). “atiment principes de validations Gestion Énergétique optimisée pour un bâtiment intelligent différents principes de validations”. Thèse de doct.

BARBIER, Franck, Corine CAUVET, Mourad OUSSALAH, Dominique RIEU, Sondes BENNASRI et Carine SOUVEYET (2002). “Composants dans l’ingénierie des systèmes d’information : concepts clés et techniques de réutilisation”. In : *Actes des deuxièmes assises nationales du GDR-I3*.

BASU, Kaustav, Lamis HAWARAH, Nicoleta ARGHIRA, Hussein JOUMAA et Stephane PLOIX (2013). “A prediction system for home appliance usage”. In : *Energy and Buildings* 67, p. 668–679. ISSN : 03787788. DOI : 10.1016/j.enbuild.2013.02.008. URL : http://linkinghub.elsevier.com/retrieve/pii/S0378778813000789.

BELTON, Valerie et Theodor STEWART (2002). *Multiple criteria decision analysis : an integrated approach*. Springer.

BIBLIOGRAPHIE BACHA, S et D CHATROUX (2006). “Nouvelles technologies de l’énergie, chapter 2 : Sytèmes photovoltaïque couplés au réseau”. In : *Hermes Science.(cité page 135.) Bakos, G., Soursos, M., et NF Tsagas (2003). Technoeconomic assessment of a buildingintegrated pv system for electrical energy saving in residential sector. Energy and Buildings* 35, p. 757–762.

BIEN, Z Zenn, Hyong-Euk LEE, Jun-Hyeong DO, Yong-Hwi KIM, Kwang-Hyun PARK et Seung-Eun YANG (2008). "Intelligent interaction for human-friendly service robot in smart house environment". In : *International Journal of Computational Intelligence Systems* 1.1, p. 77–93.

BLANC, Xavier et Olivier SALVATORI (2011). *MDA en action : Ingénierie logicielle guidée par les modèles*. Editions Eyrolles.

BLOCHWITZ, T, M OTTER, Germany I T I GMBH, D L R OBERPFAFFENHOFEN, Dassault SYSTÈMES et L M S IMAGINE. "Functional Mockup Interface 2 . 0 : The Standard for Tool independent Exchange of Simulation Models". In :

BLOCHWITZ, T., M. OTTER, M. ARNOLD, C. BAUSCH, C. CLAUSS, H. ELMQVIST, a. JUNGHANNS, J. MAUSS, M. MONTEIRO, T. NEIDHOLD, D. NEUMERKEL, H. OLSSON, J.-V. PEETZ et S. WOLF (2011). "The Functional Mockup Interface for Tool independent Exchange of Simulation Models". In : October 2015, p. 105–114. DOI : 10.3384/ecp11063105. URL : http://www.ep.liu.se/ecp{_}article/index.en.aspx?issue=63;article=13.

BRAUN, Martin, Kathrin BÜDENBENDER, Dirk MAGNOR et Andreas JOSSEN (2009). "Photovoltaic self-consumption in Germany : using lithium-ion storage to increase self-consumed photovoltaic energy". In : *24th European Photovoltaic Solar Energy Conference (PVSEC), Hamburg, Germany*.

BRUN, A., E. WURTZ, P. HOLLMULLER et D. QUENARD (2013). "Summer comfort in a low-inertia building with a new free-cooling system". In : *Applied Energy* 112, p. 338–349. ISSN : 03062619. DOI : 10.1016/j.apenergy.2013.05.052. URL : http://www.sciencedirect.com/science/article/pii/S0306261913004649.

CAMPANIÇO, Hugo, Pierre HOLLMULLER et Pedro M.M. SOARES (2014). "Assessing energy savings in cooling demand of buildings using passive cooling systems based on ventilation". In : *Applied Energy* 134, p. 426–438. ISSN : 03062619. DOI : 10.1016/j.apenergy.2014.08.053. URL : http://www.sciencedirect.com/science/article/pii/S0306261914008642.

CEBASEK, Gregory B, Jeffrey J GLOUDEMAN, Donald A GOTTSCHALK et David E RASMUSSEN (2000). *Communication system for distributed-object building automation system*. US Patent 6,104,963.

CETIN, K.S., P.C. TABARES-VELASCO et a. NOVOSELAC (2014). "Appliance daily energy use in new residential buildings : Use profiles and variation in time-of-use". In : *Energy and Buildings* 84, p. 716–726. ISSN : 03787788. DOI : 10.1016/j.enbuild.2014.07.045. URL : http://linkinghub.elsevier.com/retrieve/pii/S0378778814005854.

CHOUIKH, Sana Gaaloul (2012). "Interopérabilité basée sur les standards Modelica et composant logiciel pour la simulation énergétique des systèmes de bâtiment". Thèse de doct.

DARGAHI, Ardavan (2014). "Gestion des flux multi-énergie pour les systèmes V2H". Thèse de doct. Universite de Grenoble.

DAS, Debosmita, Reza ESMAILI, Longya XU et Dave NICHOLS (2005). "An optimal design of a grid connected hybrid wind/photovoltaic/fuel cell system for distributed energy production". In : *Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE*. IEEE, 6–pp.

DIRECTIONS, Future. "PID Control Control System System Analysis and Design". In : February 2006.

DOUKAS, Haris, Konstantinos D. PATLITZIANAS, Konstantinos IATROPOULOS et John PSARRAS (2007). "Intelligent building energy management system using rule sets". In : *Building and Environment* 42.10, p. 3562–3569. ISSN : 03601323. DOI : 10.1016/j.buildenv.2006.10.024. URL : http://linkinghub.elsevier.com/retrieve/pii/S036013230600312X.

DOUKAS, Haris Ch, Botsikas M ANDREAS et John E PSARRAS (2007). "Multi-criteria decision aid for the formulation of sustainable technological energy priorities using linguistic variables". In : *European Journal of Operational Research* 182.2, p. 844–855.

DOUNIS, a.I. et C. CARAISCOS (2009). "Advanced control systems engineering for energy and comfort management in a building environment—A review". In : *Renewable and Sustainable Energy Reviews* 13.6-7, p. 1246–1261. ISSN : 13640321. DOI : 10.1016/j.rser.2008.09.015. URL : http://linkinghub.elsevier.com/retrieve/pii/S1364032108001457.

DUFFIE, John A et William A BECKMAN (1980). *Solar engineering of thermal processes*. T. 3. Wiley New York etc.

EDWARDS, Richard E., Joshua NEW et Lynne E. PARKER (2012). "Predicting future hourly residential electrical consumption : A machine learning case study". In : *Energy and Buildings* 49, p. 591–603. ISSN : 03787788. DOI : 10.1016/j.enbuild.2012.03.010. URL : http://linkinghub.elsevier.com/retrieve/pii/S0378778812001582.

EHLERS, Gregory A, Robert D HOWERTON et Gary E SPEEGLE (1996). *Engery management and building automation system*. US Patent 5,572,438.

ELMQVIST, Hilding, François E CELLIER et Martin OTTER (1993). "Object-oriented modeling of hybrid systems". In :

FÁBIÁN, G, DA VAN BEEK et JE ROODA (1998). "Integration of the discrete and the continuous behaviour in the hybrid chi simulator". In : *1998 European Simulation Multiconference, Manchester*, p. 252–257.

FANGER, Poul O et al. (1970). "Thermal comfort. Analysis and applications in environmental engineering." In : *Thermal comfort. Analysis and applications in environmental engineering.*

FISCHER, Vincent, N I E ELECTRIQUE, Application A LA et Resolution D EQUATIONS (2004). "COMPOSANTS LOGICIELS POUR LE DIMENSIONNEMENT EN GENIE ELECTRIQUE . APPLICATION A LA RESOLUTION D ' EQUATIONS DIFFERENTIELLES . To cite this version :" in :

— (2009). "COMPOSANTS LOGICIELS POUR LE DIMENSIONNEMENT EN GENIE ELECTRIQUE . APPLICATION A LA RESOLUTION D ' EQUATIONS DIFFERENTIELLES . To cite this version :" in :

FOURER, Robert, D M GAY et B W KERNIGHAN (2004). "Design Principles and New Developments in the AMPL Modeling Language". In : *Modeling Languages in Mathematical Optimization*, p. 105–135.

FRITZSON, Peter (2010). *Principles of object-oriented modeling and simulation with Modelica 2.1*. John Wiley & Sons.

GAMS. *No Title*. URL : https://www.gams.com/.

— *No Title*. URL : http://www.omg.org/mda/.

GAY, David M (2014). "The AMPL Modeling Language — an Aid to Formulating and Solving Optimization Problems". In : URL : http://ampl.com/REFS/muscat14.pdf.

GEORGOPOULOU, Ekaterini, Yainnis SARAFIDIS et Danae DIAKOULAKI (1998). "Design and implementation of a group DSS for sustaining renewable energies exploitation". In : *European Journal of Operational Research* 109.2, p. 483–500.

GLOUDEMAN, Jeffrey J, Donald A GOTTSCHALK, David E RASMUSSEN et Michael E WAGNER (2000). *Distributed object-oriented building automation system with reliable asynchronous communication*. US Patent 6,167,316.

GOUMAS, M et V LYGEROU (2000). "An extension of the PROMETHEE method for decision making in fuzzy environment : Ranking of alternative energy exploitation projects". In : *European Journal of Operational Research* 123.3, p. 606–613.

HA, Duy Long (2008). "un système avancé de gestion de l'énergie dans le bâtiment". In :

HA, Duy Long, Minh Hoang LE et Stéphane PLOIX (2008). "An approach for home load energy management problem in uncertain context". In : *Industrial Engineering and Engineering Management, 2008. IEEM 2008. IEEE International Conference on*. IEEE, p. 336–339.

HAGRAS, Hani, Ian PACKHAM, Yann VANDERSTOCKT, Nicholas MCNULTY, Abhay VADHER et Faiyaz DOCTOR (2008). "an intelligent agent based approach for energy management in commercial building". In : *fuzzy systems*, p. 156–162. ISBN : 9781424418190.

HALLIKAINEN, Petri, Hannu KIVIJÄRVI et Kari NURMIMÄKI (2002). "Evaluating Strategic IT Investments : An Assessment of Investment Alternatives for a Web Content Management System". In : 00.c, p. 1–10.

HOLLMULLER, Pierre (2003). "Analytical characterisation of amplitude-dampening and phase-shifting in air/soil heat-exchangers". In : *International Journal of Heat and Mass Transfer* 46.22, p. 4303–4317. ISSN : 00179310. DOI : 10.1016/S0017-9310(03)00199-6. URL : http://linkinghub.elsevier.com/retrieve/pii/S0017931003001996.

HOLLMULLER, Pierre et Bernard LACHAL (2014). "Air–soil heat exchangers for heating and cooling of buildings : Design guidelines, potentials and constraints, system integration and global energy balance". In : *Applied Energy* 119, p. 476–487. ISSN : 03062619. DOI : 10.1016/j.apenergy.2014.01.042. URL : http://www.sciencedirect.com/science/article/pii/S0306261914000610.

HOLMBERG, Fredrik (2001). "Implementation of a PID Controller for Building Automation". In : March.

HUANG, H.-Y., J.-Y. YEN, S.-L. CHEN et F.-C. OU (2004). "Development of an Intelligent Energy Management Network for Building Automation". In : *IEEE Transactions on Automation Science and Engineering* 1.1, p. 14–25. ISSN : 1545-5955. DOI : 10.1109/TASE.2004.829346. URL : http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1309664.

JANG, W.S. et W.M. HEALY (2010). "Wireless sensor network performance metrics for building applications". In : *Energy and Buildings* 42.6, p. 862–868. ISSN : 03787788. DOI : 10.1016/j.enbuild.2009.12.008. URL : http://linkinghub.elsevier.com/retrieve/pii/S0378778809003314.

JUN, Zeng, Liu JUNFENG, Wu JIE et H.W. NGAN (2011). "A multi-agent solution to energy management in hybrid renewable energy generation system". In : *Renewable Energy* 36.5, p. 1352–1363. ISSN : 09601481. DOI : 10.1016/j.renene.2010.11.032. URL : http://linkinghub.elsevier.com/retrieve/pii/S0960148110005458.

KABANZA, Froduald, Michel BARBEAU et Richard ST-DENIS (1997). "Planning control rules for reactive agents". In : *Artificial Intelligence* 95.1, p. 67–113.

KASTNER, Wolfgang, Georg NEUGSCHWANDTNER, Stefan SOUCEK et HM NEWMANN (2005). "Communication systems for building automation and control". In : *Proceedings of the IEEE* 93.6, p. 1178–1203.

KAYA, İbrahim, Nusret TAN et Derek P.ATHERTON (2007). "improved cascade control structure for enhanced performance". In : *Journal of Process Control* 17.

KELMAN, Anthony et Francesco BORRELLI (2012). "Parallel nonlinear predictive control". In : *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*. IEEE, p. 71–78.

KETATA, Raouf (1992). "Méthodologies de régulation numérique incluant la logique floue". Thèse de doct.

KING, Jeffrey et Erwin JANSEN (2005). "The Gator Tech Smart House : A Programmable Pervasive Space". In : *Computer*.

KNIBBE, Engel J (1996). *Building management system*. US Patent 5,565,855.

KOLDA, Tamara G., Robert Michael LEWIS et Virginia TORCZON (2003). "Optimization by Direct Search : New Perspectives on Some Classical and Modern Methods". In : *SIAM Review* 45.3, p. 385–482. ISSN : 0036-1445. DOI : 10.1137/S003614450242889. URL : http://epubs.siam.org/doi/abs/10.1137/S003614450242889.

KOLOKOTSA, D, A POULIEZOS, G STAVRAKAKIS et C LAZOS (2009). "Predictive control techniques for energy and indoor environmental quality management in buildings". In : *Building and Environment* 44.9, p. 1850–1863.

LAUMANNS, Marco, Lothar THIELE et Eckart ZITZLER (2006). "An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method". In : *European Journal of Operational Research* 169.3, p. 932–942.

LE, Khang, Romain BOURDAIS et Hervé GUÉGUEN (2014). "From hybrid model predictive control to logical control for shading system : A support vector machine approach". In : *Energy and Buildings* 84, p. 352–359. ISSN : 03787788. DOI : 10.1016/j.enbuild.2014.07.084. URL : http://linkinghub.elsevier.com/retrieve/pii/S0378778814006331.

LE MINH, Hoang, Mireille JACOMINO et Stephane PLOIX (2011). "Prise en compte des incertitudes de pr{é}diction dans la gestion des flux de l'énergie dans l'habitat". Thèse de doct.

LE MOUNIER, Audrey, Benoît DELINCHANT et Stéphane PLOIX. "Determination of relevant model structures for self-learning energy management system". In : *Building Simulation Optimisation*.

LEE, Heyoung, Yong-Tae KIM, Jin-Woo JUNG, KH PARK, DJ KIM, B BANG et ZZ BIEN (2008). "A 24-hour health monitoring system in a smart house". In : *Gerontechnology* 7.1, p. 22–35.

LEFORT, Antoine, Romain BOURDAIS, Guillaume ANSANAY-ALEX et Hervé GUÉGUEN (2013). "Hierarchical control method applied to energy management of a residential house". In : *Energy and Buildings* 64, p. 53–61. ISSN : 03787788. DOI : 10.1016/j.enbuild.2013.04.010. URL : http://linkinghub.elsevier.com/retrieve/pii/S0378778813002454.

LOKEN, E (2007). "Use of multicriteria decision analysis methods for energy planning problems". In : *Renewable and Sustainable Energy Reviews* 11.7, p. 1584–1595. ISSN : 13640321. DOI : 10.1016/j.rser.2005.11.005. URL : http://linkinghub.elsevier.com/retrieve/pii/S1364032105001280.

LOVÁSZ, László (2010). "Discrete and continuous : two sides of the same ?" In : *Visions in Mathematics*. Springer, p. 359–382.

MAHENDRA, Singh, Ploix STÉPHANE et Wurtz FREDERIC (2015). "Modeling for Reactive Building Energy Management". In : *Energy Procedia* 83, p. 207–215.

MARINAKIS, Vangelis, Haris DOUKAS, Charikleia KARAKOSTA et John PSARRAS (2013). "An integrated system for buildings' energy-efficient automation : Application in the tertiary sector". In : *Applied Energy* 101, p. 6–14. ISSN : 03062619. DOI : 10.1016/j.apenergy.2012.05.032. URL : http://linkinghub.elsevier.com/retrieve/pii/S0306261912004059.

*mathworks*. URL : http://www.mathworks.fr/products/matlab/..

MAURER, Peter M (2000). "Components : What if they gave a revolution and nobody came ?" In : *Computer* 33.6, p. 28–34.

MEIJLER, Theo Dirk et Oscar NIERSTRASZ (1997). "Beyond objects : components". In : *Cooperative information systems : current trends and directions*, p. 49–78.

MISSAOUI, Rim, Hussein JOUMAA, Stephane PLOIX et Seddik BACHA (2014). "Managing energy Smart Homes according to energy prices : Analysis of a Building Energy Management System". In : *Energy and Buildings* 71, p. 155–167. ISSN : 03787788. DOI : 10.1016/j.enbuild.2013.12.018. URL : http://linkinghub.elsevier.com/retrieve/pii/S0378778813008335.

MONCRIEFF, Simon, Svetha VENKATESH, Geoff WEST et Stewart GREENHILL (2007). "Multi-modal emotive computing in a smart house environment". In : *Pervasive and Mobile Computing* 3.2, p. 74–94.

MOUNIER, Audrey Le, Benoît DELINCHANT et Stéphane PLOIX (2014). "DETERMINATION OF RELEVANT MODEL STRUCTURES FOR SELF-LEARNING ENERGY MANAGEMENT SYSTEM Grenoble Electrical Engineering lab , Grenoble-INP , Grenoble , France G-SCOP , Grenoble-INP , Grenoble , France The platform". In : *Building simulation optimization*.

OOKA, Ryozo et Kazuhiko KOMAMURA (2009). "Optimal design method for building energy systems using genetic algorithms". In : *Building and Environment* 44.7, p. 1538–1544.

PARK, Kwang-Hyun, Zeungnam BIEN, Ju-Jang LEE, Byung Kook KIM, Jong-Tae LIM, Jin-Oh KIM, Heyoung LEE, Dimitar H STEFANOV, Dae-Jin KIM, Jin-Woo JUNG et al. (2007). "Robotic smart house to assist people with movement disabilities". In : *Autonomous Robots* 22.2, p. 183–198.

PÉREZ-LOMBARD, Luis, José ORTIZ et Christine POUT (2008). "A review on buildings energy consumption information". In : *Energy and buildings* 40.3, p. 394–398.

POSTOLACHE, Roxana (2007). "Linear and Nonlinear Optimization Programming".

RICHARDSON, Leonard et Sam RUBY (2008). *RESTful web services*. " O'Reilly Media, Inc.".

RIGOS, Analandos, T KORONIDES, P CHARAMOGLIS et JM CACHET (1993). "A modern energy management system for the control of the hellenic generation and transmission system". In : *Athens Power Tech, 1993. APT 93. Proceedings. Joint International Power Conference*. T. 1. IEEE, p. 477–481.

ROBINSON, L E Plessis (2002). *WP7 : OUTILS POUR LA DISCRETISATION DE LA REPRESENTATION CONTINUE*. Rapp. tech. Acotris, p. 1–36.

SALSBURY, T. I. (1998). "A Temperature Controller for VAV Air-Handling Units Based on Simplified Physical Models". In : *HVAC&R Research* 4.3, p. 265–279. ISSN : 1078-9669. DOI : 10.1080/10789669.1998.10391404. URL : http://www.tandfonline.com/doi/abs/10.1080/10789669.1998.10391404.

SARJEANT, WJ, FW MACDOUGALL, DW LARSON et I KOHLBERG (1997). "Energy storage capacitors : aging, and diagnostic approaches for life validation". In : *Magnetics, IEEE Transactions on* 33.1, p. 501–506.

SASSOON, Jacques (1998). *Urbanisation des systèmes d'information*. Broché.

SPIGEL, Lynn (2005). "Designing the Smart House Posthuman Domesticity and Conspicuous Production". In : *European Journal of Cultural Studies* 8.4, p. 403–426.

SRINIVAS, Nidamarthi et Kalyanmoy DEB (1994). "Muiltiobjective optimization using nondominated sorting in genetic algorithms". In : *Evolutionary computation* 2.3, p. 221–248.

STÉPHANE, Gibout, Franquet ERWIN, Bédécarrats JEAN-PIERRE et Dumas JEAN-PIERRE (2012). "Comparison of different modelings of pure substances during melting in a DSC experiment". In : *Thermochimica Acta* 528, p. 1–8. ISSN : 00406031. DOI : 10.1016/j.tca.2011.10.019. URL : http://www.sciencedirect.com/science/article/pii/S0040603111005181.

STORN, Rainer (1997). "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces". In : *Journal of Global Optimization*, p. 341–359.

SZYPERSKI, Clemens (2002). *Component software : beyond object-oriented programming*. Pearson Education.

THEVENON, Luc (2000). "Representation des systemes hybrides complexes par flux de donnees : developpement d'un outil de modelisation et de simulation des procedes Batch". Thèse de doct.

THOMAS, Bertil, Mohsen SOLEIMANI-MOHSENI et Per FAHLÉN (2005). "Feed-forward in temperature control of buildings". In : *Energy and Buildings* 37.7, p. 755–761. ISSN : 03787788. DOI : 10.1016/j.enbuild.2004.10.002. URL : http://linkinghub.elsevier.com/retrieve/pii/S0378778804003378.

TSANAS, Athanasios et Angeliki XIFARA (2012). "Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools". In : *Energy and Buildings* 49, p. 560–567. ISSN : 03787788. DOI : 10.1016/j.enbuild.2012.03.003. URL : http://linkinghub.elsevier.com/retrieve/pii/S037877881200151X.

UPTON, Eben et Gareth HALFACREE (2013). *Raspberry Pi user guide*. John Wiley & Sons.

VIRK, GS, JYM CHEUNG et DL LOVEDAY (1990). "The role of control technology for energy savings in buildings". In : *Advanced SCADA and Energy Management Systems, IEE Colloquium on*. IET, p. 3–1.

WANG, Jiang-Jiang, You-Yin JING, Chun-Fa ZHANG et Jun-Hong ZHAO (2009). "Review on multi-criteria decision analysis aid in sustainable energy decision-making". In : *Renewable and Sustainable Energy Reviews* 13.9, p. 2263–2278. ISSN : 13640321. DOI : 10.1016/j.rser.2009.06.021. URL : http://linkinghub.elsevier.com/retrieve/pii/S1364032109001166.

WARKOZEK, Ghaith (2011). "Génération automatique de problèmes d ' optimisations pour la conception et la gestion des réseaux électriques de bâtiments intelligents multi-sources multi-charges". In :

WILLIAMS, Hilary Paul (1999). "Model building in mathematical programming". In :

YASSINE, Abed Al (2008). "Généération des tests et placement de capteurs pour le diagnostic des systèmes physiques s ' appuyant sur une modélisation structurelle". In :

YERRAPRAGADA, C et PS FISHER (1993). “Voice controlled smart house”. In : *Consumer Electronics, 1993. Digest of Technical Papers. ICCE., IEEE 1993 International Conference on*. IEEE, p. 154–155.

ZAMORA-MARTÍNEZ, F., P. ROMEU, P. BOTELLA-ROCAMORA et J. PARDO (2014). “On-line learning of indoor temperature forecasting models towards energy efficiency”. In : *Energy and Buildings* 83, p. 162–172. ISSN : 03787788. DOI : 10.1016/j.enbuild.2014.04.034. URL : http://linkinghub.elsevier.com/retrieve/pii/S0378778814003569.

ZONDAG, H.a., D.W. de VRIES, W.G.J. van HELDEN, R.J.C. van ZOLINGEN et a.a. van STEENHOVEN (2002). “The thermal and electrical yield of a PV-thermal collector”. In : *Solar Energy* 72.2, p. 113–128. ISSN : 0038092X. DOI : 10.1016/S0038-092X(01)00094-9. URL : http://linkinghub.elsevier.com/retrieve/pii/S0038092X01000949.