Etude prospective d'un environnement d'aide à la décision rendu configurable par des modèles

Liwen Zhang, Hervé Pingaud, Elyes Lamine, Franck Fontanili, Christophe Bortolaso, Mustapha Derras

CIGI 2021
Etude prospective d'un environnement d'aide à la décision rendu configurable par des modèles

LIWEN ZHANG[1,3], HERVE PINGAUD[2], ELYES LAMINE[1], FRANCK FONTANILI[1], CHRISTOPHE BORTOLASO[3], MUSTAPHA DERRAS[3],

[1] Centre Génie Industriel, Université de Toulouse-IMT Mines Albi
Campus Jarlard, 81000 Albi, France
{prénom.nom}@mines-albi.fr

[2] CNRS LGC, Université de Toulouse-INU Champollion
Place de Verdun, 81012 Albi, France
herve.pingaud@univ-jfc.fr

[3] Berger-Levrault
Rue Jean Rostand, 31670 Labège, France
{prénom.nom}@berger-levrault.com

***Résumé* – Les circuits de décision, le rythme de prise de décision, autant que le poids de celles-ci, prennent de nouvelles dimensions dans les organisations. Nous nous intéressons à la capacité du décideur à inclure les caractéristiques propres à son écosystème, son contexte et ses enjeux, souvent changeants, dans ses pratiques. Les décideurs sont souvent des experts métiers à qui nous essayons de procurer la capacité de concevoir et adapter à moindre effort leurs outils de prise de décision. A cette fin, la création d'un environnement d'aide à la prise de décision utilisant une approche d'Ingénierie Dirigé par les Modèle (IDM) est proposée. Ce travail exploratoire cherche à lier des modèles de besoin de prise de décision à des modèles logiques d'appel à un solveur en optimisation combinatoire. Les architectures fonctionnelles et logiques de cet environnement sont décrites sur un cas d'études, le problème du voyageur de commerce. Nous expliquons les hypothèses et les connaissances véhiculées par les modèles peuplant la chaîne d'ingénierie en suivant les principes de l'architecture MDA. Plusieurs règles de transformation entre modèles y sont définies. Un prototype de cet environnement a été développé à l'aide d'un outil de méta-modélisation. La preuve de concept est sa capacité à reformuler des problèmes en ligne.**

***Abstract* – Decision-making paths, frequency of decision-making, as much as their critical weight, have new prominences in today organizations. We are interested here in the decision-maker's capacity to include the characteristics specific to his/her ecosystem, its context and its often-changing issues in the way he/she practices. Decision-makers are often business experts to whom we want to help the design and adaption of their decision-making tools. The creation of a decision support framework using a Model Driven Engineering (MDE) approach is proposed. It seeks to link models of decision-making needs to logical models required by a combinatorial optimization solver. To do this, the functional and logical architectures of this framework are described on a case study, the traveling salesman problem. The assumptions and knowledge conveyed by the models all along the engineering chain are proposed. The design of the environment follows the principles of the MDA architecture. Several transformation rules between models are then explained. A prototype of this environment was developed using a meta modeler software. It provides a meaningful proof of concept, we mean the capacity to change problem formulations on the fly.**

***Mots clés* - Ingénierie Dirigée par les Modèles, Aide à la décision, Gestion des opérations, Recherche Opérationnelle, Problème du Voyageur de Commerce, Outils de méta modélisation**
***Keywords* – Model Driven Engineering, Decision making, Operations Management, Operations Research, Travelling Salesman Problem, Meta modeler.**

1 ANALYSE DES ATTENTES EN AIDE A LA DECISION

Nos travaux s'inscrivent dans le faisceau des transitions sociétale et numérique des systèmes de production de biens ou de services, et en particulier le mouvement de fond que constitue Industrie 4.0. Nous le revisitons sous l'angle d'une évolution fournie par le numérique sur la capacité décisionnelle des systèmes.

Actuellement, dans une société connectée à de multiples échelles territoriales, une forte pression s'exerce sur les organisations productives pour régulièrement coordonner leurs activités et piloter leurs flux de produits et services. Poursuivre les bons objectifs est toujours le cap, mais ces objectifs sont souvent évolutifs (accès aux marchés, maîtrise de performance, réduction des risques, …) et portent sur des périmètres qui sont à l'échelle de réseaux collaboratifs ce qui dresse parfois des barrières en termes d'autonomie de décision. Ce constat vaut pour de multiples domaines d'application, citons la chaine logistique manufacturière ou encore la gestion de tournées de professionnels libéraux dans les services à la personne, à titre d'exemples. Le caractère stratégique de tels sujets ouvre nécessairement un débat récurrent sur la culture du pilotage dans sa propre organisation et sur l'aptitude à répondre aux sollicitations des multiples réseaux dans lesquels elle s'inscrit. C'est là que se trouve une partie de l'art du management. Dans ce contexte, l'amélioration continue qui irrigue nos démarches qualité n'est pas seulement confinée dans les processus opérationnels et supports, elle impacte les processus de pilotage. Elle interroge sur les modes de pensée, les activités de prises de décision, les pratiques et les outils d'aide à la décision.

Nous savons de longue date que les avantages concurrentiels, tout comme la quête de valeur ajoutée, ne peuvent prendre une réalité tangible que par un savoir-faire en aide à la décision. Le génie industriel a joué un rôle important dans cette prise de conscience. Par exemple, la planification ne fait pas que traduire une volonté stratégique, c'est une fonction qui peut devenir une arme stratégique. Rappelons si besoin est l'importance donnée à la tension des flux, au juste à temps, ou encore à la satisfaction des usagers ou des agents de production dans les travaux de recherche en gestion des opérations. C'est la raison pour laquelle cette planification est abordée avec une formulation « maison », tenant compte de spécificités locales dans une organisation donnée.

La tendance est à introduire de plus en plus de connaissances métier dans la prise de décision.

Enfin, la société est devenue une société de l'exigence immédiate. Les circuits d'approvisionnement, de production et de distribution ne doivent pas seulement poursuivre des critères de performance économique et sociale, ils doivent accepter des changements de la demande et savoir y répondre. L'agilité est devenue un maître-mot. Mais le poids de ces incertitudes dans les espaces de prise de décision conduit inexorablement à plus de flexibilité dans les usages de différents outils de calcul et invite à développer un nouvel esprit de continuité des activités de prise de décision, prolongeant ainsi logiquement le concept de continuité des activités opérationnelles.

Pour toutes ces raisons, un décideur appréciera d'avoir le contrôle sur la formulation de ses modèles de prise de décision et de pouvoir faire bon usage de cette capacité de contrôle lorsque le contexte est très dynamique. Cela est d'autant plus factuel que beaucoup d'organisations ont gagné en maturité dans cet exercice. Toujours en planification, ce sont les progrès enregistrés grâce à la diffusion du MRP II (Manufacturing Ressource Planning II) et des facilités prodiguées par l'usage courant des progiciels de gestion intégrés qui sont à l'origine de cela. Le progiciel est paramétré pour représenter les processus relatifs à une organisation connue de la production. Modifier les paramètres est une opération lourde et à risque. D'autre outils se sont imposés dans le paysage de l'aide à la décision. Ce sont ceux qui sont mobilisés par les experts et chercheurs en aide à la décision. Ils leur offrent des services numériques encapsulant des techniques de recherche opérationnelle, appelés solveurs. Beaucoup de solveurs sont aujourd'hui pris en main en quelques heures et équipés de bibliothèques d'algorithmes dits classiques qui permettent de travailler par réutilisation de code existant.

Ces deux catégories d'outils encadrent un périmètre dans lequel le décideur en gestion des opérations évolue actuellement. D'un côté des outils accédant facilement aux données d'entreprise, mais pour lesquels les formulations de problème sont relativement fermées, même s'il existe quelques paramétrages possibles. De l'autre, des outils ouverts qui permettent de reformuler les problèmes d'aide à la décision, de faire face à une certaine complexité, mais qui requièrent une connaissance plus fine de la recherche opérationnelle, sans être vraiment branchés en ligne sur les données de l'organisation. A ce stade de notre analyse, l'idée a germé d'étudier une troisième voie ouvrant de nouvelles perspectives de travail pour le décideur.

Une analyse bibliographique nous a montré qu'il n'existe pas à ce jour de travaux de recherche équivalents sur une application de l'Ingénierie Dirigée par les Modèles (IDM) allant de la génération personnalisée de modèle d'aide à la décision à leur résolution passant avec un couplage systématique. Notre travail est donc prospectif, nous ouvrons une voie.

Cette nouvelle voie ne dispose pas de la maturité et de l'expérience des deux autres, évoquées ci-dessus. Par contre, il cherche à ouvrir au maximum les possibilités de maitriser l'instruction des problèmes d'aide à la décision à partir de connaissances métier et sans à avoir à traiter directement la conception de code pour l'appel au solveur.

Ainsi, nous émettons l'hypothèse qu'avec le temps, le verrou scientifique se déplace d'un savoir-faire factuel en aide à la décision vers un besoin nouveau de prise de décision rapide. Les outils doivent être conçus pour faire face à la variété des situations à traiter et pour pouvoir prendre des décisions circonstanciées en contexte de pression temporelle (ce qui revient à considérer que le nombre d'interlocuteurs dans la chaine décisionnelle est réduit).

L'objectif poursuivi est donc de chercher comment donner au décideur non expert en recherche opérationnelle la faculté de faire une spécification en ligne et à moindre effort de ses problèmes, puis de les résoudre avec les bonnes données.

Une telle ligne de pensée invite à revisiter la place de la décision dans les modèles d'entreprise. Car cette ambition ne peut s'affranchir de la quantité et de la qualité des connaissances métier représentatives de la réalité du fonctionnement de l'organisation. Elles sont la source du raisonnement d'un décideur. Or celles-ci étant nombreuses, une solution viable ne peut pas s'imaginer sous la seule forme de l'émancipation d'une solution technique existante, aussi riche soit elle. Cette émancipation se heurterait à des interfaces trop sophistiquées quand il faudrait mobiliser ces connaissances métier et les données afférentes. Il en découle une exigence qui est de bien structurer le passage de la connaissance métier à la connaissance logique dont un solveur a besoin pour réaliser des calculs afin de trouver des solutions. A cette fin de prendre une décision, conscients de l'impératif d'utilisation structurée des connaissances métier, nous proposons une nouvelle option dans la pratique d'un décideur avec un environnement d'aide à la décision dont le fonctionnement relève des théories de l'ingénierie dirigée par les modèles (IDM).

Puisque cette pratique exigera de conjuguer une relation subtile entre (1) les connaissances et l'expertise métier et (2) la capacité à décider, c'est d'une logique de flux de connaissances entre modèle orienté métier, modèle mathématique d'aide à la décision et modèle logique de calcul de solutions dont il sera question dans cet article. Notre posture consiste à étudier un enchainement de ces trois modèles pour relier par voie numérique l'espace de formulation de problèmes et l'espace de résolution

Notre recherche vise d'abord à imaginer l'architecture fonctionnelle d'un tel environnement de travail. Du fait du caractère très exploratoire du sujet, nous l'expliquons au moyen d'un cas d'études très simple, à vocation uniquement pédagogique. Il s'agit de différentes variantes du fameux voyageur de commerce. Ces variantes forment autant de problèmes différents sur lesquels nous avons raisonné. Il constituera un fil rouge dans la suite de cet article. Cette architecture fonctionnelle est détaillée au paragraphe 2. Cette présentation nous conduit à discuter les différents sens donnés au terme de modèle pour chacun des trois modèles qui sont utilisés par l'IDM. Puis, au chapitre 3, nous montrons comment différents mécanismes de transformation entre ces modèles permettent de rendre assez systématique le passage de la connaissance métier à la connaissance logique. Nous démontrons la faisabilité de cette proposition en faisant une preuve de concept sur notre prototype au chapitre 4. Enfin, nous tirons les enseignements de cette expérience au chapitre 5.

2 DES MODELES POUR ENCADRER LES VARIANTES DE PROBLEMES DE PRISE DE DECISION

Les problèmes de routage et d'ordonnancement sont des problèmes de prise de décision. Ils nécessitent la détermination de séquences optimales d'activités opérées par des ressources qualifiées. L'optimalité se traduit par des critères d'appréciation à la source de l'écriture d'une fonction objectif. L'ordre entre les activités et le choix des ressources allouées sont assujettis à la satisfaction de règles de fonctionnement qui forment un jeu de contraintes pesant sur l'espace des solutions possibles.

Le problème du voyageur de commerce (abréviation de sa traduction anglaise : TSP) est une classe connue dans l'ensemble de ces problèmes. Rappelons simplement son énoncé le plus répandu. Un voyageur doit visiter une et une seule fois un ensemble donné de villes. Sa tournée est conçue pour le satisfaire, au sens où le TSP est souvent présenté comme un problème de minimisation de la distance parcourue par le voyageur. Cette forme du plus court chemin est un problème NP complet. La figure 1 présente un cas d'études simple, à titre d'illustration de notre théorie. Un ensemble de quatre villes (Paris, Lyon, Nice, Toulouse) sont à visiter. Les distances entre ces villes sont connues. Les données de distance fournies permettent de faire le trajet dans les deux sens entre chaque paire de villes. Il n'y a donc aucun interdit en termes de connectivité entre les villes pour ce voyageur.

Figure 1- Le contexte de notre cas d'étude du TSP

Il existe de nombreuses variations dans la prise de décision autour du TSP. Comme le précisent (Gutin & Punnen, 2006), l'état de l'art couvre un ensemble d'au moins dix configurations différentes. Evoquons de manière synthétique les sources de ces changements de problème en trois parties :

1- La fonction traduisant l'objectif à optimiser peut varier pour traduire des moyens de transport en compétition avec des temps de transport différents, voire des coûts de transport.
2- Les règles de fonctionnement peuvent changer. Ainsi dans une forme d'évolution, chaque ville est associée à un poids non négatif. La tournée doit alors trouver un ensemble minimal de villes à visiter pour lequel la somme des poids est supérieure à un seuil donné. Dans le même ordre d'idée, le fonctionnement peut relaxer la notion de visite unique en spécifiant au moins une visite par ville. Ou encore, l'alternative à une tournée unique de plusieurs tournées partielles, indépendantes et complémentaires, conduit à des décisions différentes.
3- La nature du problème peut changer quand on touche aux hypothèses fondamentales. Ainsi, dans une forme clustérisée du TSP, plusieurs sous problèmes coexistent avec une recherche de coût minimal dans un ensemble des villes partitionné en plusieurs classes indépendantes. Ce même problème peut se décliner dans un autre but, cherchant une seule ville dans chacune de ces classes. Les auteurs citent aussi une publication où les classes de villes ne sont pas totalement disjointes. Enfin, la couverture des villes peut être imaginée en mobilisant plusieurs voyageurs partant de la même ville.

En recherche opérationnelle, une solution traditionnelle pour résoudre un problème de décision de type TSP est de passer par une formulation mathématique en optimisation sous contraintes. Il s'agit de poser les termes d'une programmation linéaire en nombre entiers. Cette écriture repose sur un graphe orienté traduisant les connaissances métier. Un point symbolise une ville à visiter et un arc représente le trajet entre deux villes. Les variables entières sont des booléens associés aux arcs. La solution se traduit par des variables nulles (=0) pour les arcs non retenus au final, les autres forment la tournée choisie et auront une valeur strictement positive (=1). Sur cette base, la formulation de la fonction traduisant l'objectif qualifie les performances attendues, et les relations de contraintes entre les variables traduisent les règles de fonctionnement. Cette première partie des variations est donc significative d'une nouvelle écriture de la fonction à optimiser. La modification dans l'écriture d'une contrainte, l'ajout ou la suppression d'une contrainte sont les changements induits par la deuxième partie. Dans la troisième partie, le changement est plus important. Il concerne tout à la fois les données, la fonction à optimiser et les contraintes.

Toutefois, nous avons choisi d'aborder sur un même front toutes ces sources de changement du TSP. Ainsi, les variations évoquées devraient pouvoir être traitées dans un même environnement de travail. Ce faisant, et ayant posé une première hypothèse sur la formulation mathématique du problème à traiter, nous devons intégrer les conséquences de celle-ci sur la recherche de solutions. Nous supposons qu'un solveur en programmation linéaire en nombres entiers est disponible au sein de l'environnement de travail. L'appel de ce solveur se fait par un langage spécifique à ce solveur qui permet de lui transmettre les données et les relations mathématiques dont il a besoin.

A ce stade de l'exposé et au moyen du cas d'études, nous avons donc décrit littéralement la nature des décisions à prendre sur un système qui est connu. Ce volet constitue les connaissances métier. Nous allons lui associer un modèle métier. Puis, nous avons expliqué comment traduire ces connaissances en un problème mathématique bien formulé en parlant de données, de variables, d'objectif et de contraintes. Cet ensemble mobilise un vocabulaire de recherche opérationnelle. Ce sont ces connaissances mathématiques que nous consignons dans un modèle d'aide à la décision. Enfin, pour prendre la décision, l'espace des solutions du problème mathématique est parcouru par un solveur qui cherche à assurer la solution, faisable avec un objectif poursuivi optimisé. L'appel du solveur est aussi associé à un modèle que nous appelons modèle logique de calcul des solutions.

Nous décrivons donc le mécanisme de l'aide à la décision au moyen d'une chaine de connaissances qui évolue depuis un stade préliminaire d'analyse des besoins métier jusqu'à une proposition de décisions satisfaisantes au regard de ces besoins. Trois formes de modèles jalonnent donc séquentiellement cette chaine. Cette logique d'enchainement correspond à un processus cognitif que l'expert en recherche opérationnelle exécute. Nous le résumons sur la figure 2. Il est composé de quatre étapes : définir le contexte, exprimer les besoins de prise de décision, identifier et formuler le problème d'aide à la décision, préparer le calcul de la solution. A l'issue de chaque étape, le raisonnement a produit un modèle qui constitue l'entrée de l'étape suivante.

C'est la façon dont ce processus linéaire s'exécute durant tout l'acte de prise de décision.

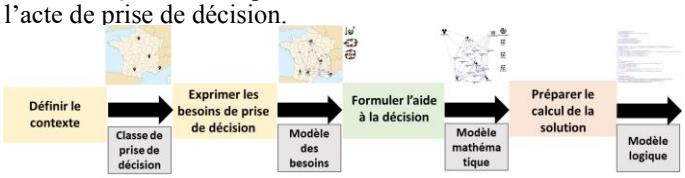

Figure 2- Modèle du processus cognitif de prise de décision

Notre environnement de travail est calqué sur ce processus cognitif. En cela, il cherche à l'expliciter et à le soutenir. Pour faciliter une vue d'ensemble de son architecture fonctionnelle, nous le décrivons en analogie avec une chaîne de production industrielle. Une chaîne de production est « l'ensemble des moyens qualifiés pour opérer une fabrication de produits manufacturés dans un volume et un temps maîtrisés. Elle s'étend des stocks de matières premières jusqu'à ceux de produits finis ». En plaquant cette définition sur notre sujet, nous identifions des stades d'avancement de la fabrication qui va aller de la définition du système et de ses besoins de prise de décision à la production d'un résultat par un logiciel de recherche de solutions appelé en toute connaissance de cause. Nous faisons reposer l'architecture fonctionnelle sur une approche modulaire. Une série de modules assimilés à des ateliers de transformation réalisent les transformations de connaissance. Chaque module correspond à une étape du processus cognitif. Sauf qu'ici, ce n'est pas un bien matériel qui est concerné, mais quelque chose de plus symbolique : une connaissance contenue dans un modèle. La transformation de la connaissance sera effectivement assimilée à une transformation de modèles. Chaque atelier a son propre objectif de transformation. Les flux entre ateliers sont des flux de modèles.

Attention, ce caractère un peu mécanistique des transformations de connaissance ne doit pas laisser supposer que l'humain est hors de la boucle. A chaque étape, il est possible de solliciter l'humain pour apporter les connaissances métier complémentaires en cas de blocage des règles de transformation. C'est une approche classique en IDM, l'apport est normalement minimisé, mais pas forcément inexistant.

Car, en IDM, par principe, l'usage fait d'un modèle quitte la vocation descriptive ou prescriptive qui lui donne la communauté du génie industriel. Il entre dans un rôle très opératoire parce qu'il est une source et un résultat d'un traitement informatique. Nos trois formes de modèles deviennent des artefacts d'ingénierie de la connaissance. Et en filant la métaphore, chaque atelier est équipé d'une ou plusieurs « machines » qui sont des services numériques opérant des traitements à base d'algorithmes, consommant et produisant des flux de connaissance via ces modèles. Tout au long du cycle de vie de l'ingénierie, le contrôle à exercer sur cette chaîne, module par module, correspondra aux règles régissant une transformation en IDM, de modèle à modèle (M à M).

Reprenons l'exemple du TSP pour illustrer le contenu de ces modèles et également leur variété. Les quatre flux de modèles entre les ateliers sont le modèle du contexte, le modèle des besoins de prise de décision, le modèle mathématique d'aide à la décision, le modèle logique de calcul. Dans le cas du TSP, ils seront décrits au prochain paragraphe. Nous choisissons de les présenter dans l'ordre inverse de ce flux. Le type de TSP choisi ici est un calcul cherchant la distance minimale à parcourir par un voyageur qui souhaite faire une seule tournée incluant une et une seule visite par ville.

Puisque nous avons défini le contexte d'un problème de routage et ordonnancement en faisant ce choix d'illustration, la première étape du processus de la figure 2 n'a pas d'objet autre que de référencer un besoin à exprimer dans cette classe de prise de décision avec un langage métier adapté à ce contexte.

a- Modèle logique de calcul de la décision

Le solveur que nous avons retenu pour notre prototype est IBM CPLEX [1] qui présente un avantage déterminant en IDM, il met à disposition un langage, appelé OPL (pour Optimisation Programming Language) pour que l'utilisateur définisse les problèmes à traiter. Le langage OPL a une structure classique de langage de programmation avec une partie déclarative et une partie exécutable incluses dans une fichier (.mod). Les données d'un calcul sont également renseignées dans un fichier de données (.dat).

Un exemple du code est fourni sur la figure 3. On y distingue une partie déclarative où données (*int, range, tuple, setof, int*, lignes 1 à 10) et variables (dvar, lignes 11&12) sont distinguées de manière abstraite. Puis, il possède une partie exécutable où les relations explicitant la fonction (*minimize*, lignes 15 à 17) et les contraintes sont définies à l'aide de mots clés (*subject to*, lignes 18 à 25). L'écriture en OPL est vraiment très proche des formulations mathématiques traditionnelles recommandées par *l'American Mathematical Society,* et que la recherche opérationnelle pratique avec rigueur.

L'appel que nous faisons de ce solveur nécessite de fournir ces deux fichiers. La figure 4 décrit le fichier des valeurs des données déclarées. Il est facile de repérer les caractéristiques de notre cas d'études avec les 4 villes et la spécification des distances entre les villes est faite par une liste en bijection avec la liste des arcs.

```
1  int totalNumberOfCityNode=...;
2  int beginNodeIndex=...;
3  range cityNodeSet=beginNodeIndex..totalNumberOfCityNode;
4  tuple routingEdge{
5      int startNodeID;
6      int endNodeID;
7  }
8  setof(routingEdge) totalEdgeTuple=...;
9  int distanceMatrix[totalEdgeTuple]=...;
10 int travellingTimeMatrix[totalEdgeTuple]=...;
11 dvar boolean dvarAssignment[totalEdgeTuple];
12 dvar int+ dvarNoSubTour[cityNodeSet];
13
14
15 dexpr int totalDist = sum ( <i,j> in totalEdgeTuple)distanceMatrix[<i,j>] * dvarAssignment[<i,j>];
16 dexpr int totalTime = sum ( <i,j> in totalEdgeTuple) travellingTimeMatrix[<i,j>] * dvarAssignment[<i,j>];
17 minimize totalDist+totalTime;
18 subject to {
19 forall(i in cityNodeSet: i>1, j in cityNodeSet:j>1 && j!=i )
20     dvarNoSubTour[i] - dvarNoSubTour[j] + (totalNumberOfCityNode - 1) * dvarAssignment[<i,j>] <= totalNumberOfCityNode - 2;
21 forall( j in cityNodeSet, <i,j> in totalEdgeTuple )
22     sum( i in cityNodeSet:i!=j ) dvarAssignment[<i,j>]==1;
23 forall( i in cityNodeSet, <i,j> in totalEdgeTuple )
24     sum( j in cityNodeSet:j!=i ) dvarAssignment[<i,j>]==1;
25 }
26
27
28 main {
29     thisOplModel.generate();
30 cplex.solve();
31 var ofile = new IloOplOutputFile("D:\\Users\\liwen.zhang\\Desktop\\tspOFResult.txt");
32 ofile.write(thisOplModel.totalTime+" ")
33 ofile.write(thisOplModel.totalDist+" ")
34 ofile.writeln(cplex.getSolvedTime())
35 ofile.close();
36 var ofCSV = new IloOplOutputFile("D:\\Users\\liwen.zhang\\Desktop\\tspAssResult.csv");
37 for(var i in thisOplModel.dvarAssignment){
38     ofCSV.writeln(i.startNodeID+","+i.endNodeID+","+thisOplModel.dvarAssignment[i]);
39 }
40 ofCSV.close();
41 }
```

Figure 3 – Modèle logique de calcul en langage OPL – Partie code du modèle (fichier .mod)

```
1 beginNodeIndex=1;
2 totalNumberOfCityNode=4;
3 totalEdgeTuple={<1,2><1,4><1,3><2,4><2,1><2,3><4,2><4,1><4,3><3,1><3,2><3,4>};
4 distanceMatrix=[550,175,750,115,600,450,135,230,200,800,430,180,];
5 travellingTimeMatrix=[0,0,0,0,0,0,0,0,0,0,0,0,];
```

Figure 4 – Modèle logique de calcul en langage OPL - Partie données du modèle (fichier .dat)

b- Modèle mathématique d'aide à la décision

La formulation du problème de TSP repose sur un graphe orienté duquel sera déduite la formulation mathématique. Pour les besoins de notre environnement, ce graphe est annoté de

[1] Solveur programmation mathématique : https://www.ibm.com/analytics/cplex-optimizer

toutes les connaissances dont il sera fait usage pour générer le modèle logique de la figure 3.

Ainsi, le graphe présenté sur la figure 5 est enrichi par ces annotations qui permettent de mémoriser la structure du graphe au niveau de ses arcs (routing edge). Les points sont numérotés à cette fin. Les ensembles homogènes sont aussi présentés (CityNodeSet, routingEdgeSet) et sont reliés à leurs éléments par des lignes pointillées. Enfin, les objets qui sont positionnés sur la droite correspondent à des relations mathématiques dont l'évaluation mobilise des éléments du graphe. On y trouve la fonction à minimiser (min. travel distance), des ensembles de relations portant sur les entrées et sorties de ville (flow in, flow out) qui serviront à brider le nombre de visites autorisées, et un ensemble de relations contraignant la solution à une seule et unique tournée (No Subtour).

A ce stade de développement de notre prototype, nous n'avons pas fait appel à des outils d'écriture formelle, comme Maple, par exemple, pour présenter ces objets mathématiques. Nous avons choisi de mobiliser une bibliothèque de formules pré-écrites où ces relations sont codées. Ce n'est qu'un stade intermédiaire qui nous permet d'entreprendre les premiers tests. La structure modulaire de l'environnement permettra de faire ultérieurement des développements plus détaillés de cette partie. Nous gardons à l'esprit une évidence, l'écriture des contraintes est un savoir-faire et il sera difficile de prétendre à en faire une génération systématique. La philosophie d'une récupération de l'existant n'est qu'une première voie pragmatique que nous retenons.

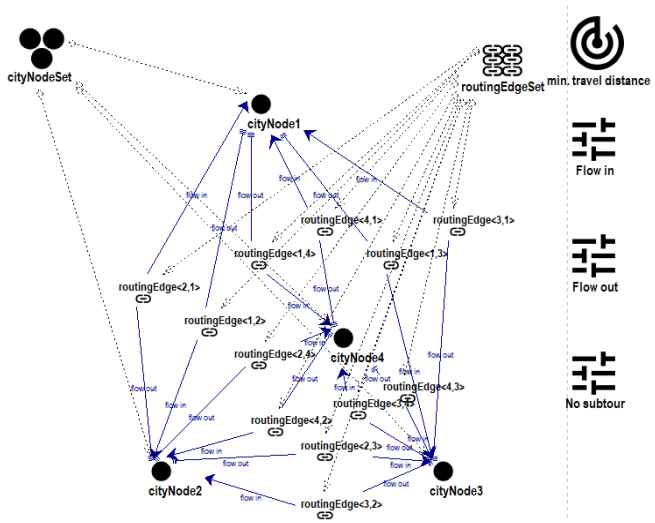

Figure 5 – Modèle mathématique d'aide à la décision inspiré de la théorie des graphes

c- Modèle des besoins de prise de décision

L'utilisation de langage spécifique à un domaine (DSL) a tendance à croître. Elle est une voie différente de la voie classique mobilisant des langages de modélisation d'entreprise, souvent promus par des logiciels BPM. Afin d'éviter toute surcharge en termes de syntaxe de langage, mais aussi avec le souci d'avoir tout degré de liberté sur le plan technique pour écrire et utiliser les connaissances portées par un tel modèle, nous avons retenu cette idée de mettre progressivement au point un langage spécifique au domaine de la prise de décision dans un contexte connu. En faisant cela, nous marchons sur les traces de la méthode GRAI (Graphs with Results and Actions Inter-related) (Chen et al., 1997) qui a de longue date gravé dans le marbre le concept de centre de décision appliqué à la gestion de production.

Récemment, l'OMG a publié la spécification du langage DMN (Decision Model and Notation) (Biard et al., 2015) qui permet d'enrichir BPMN 2.0 (Allweyer, 2016) . Dans la mesure où DMN est encore peu adapté à une prise de décision en gestion des opérations, mais qu'il vise plus à automatiser la gestion de flux d'informations dématérialisés, nous ne nous en sommes pas inspirés. En effet, si ce langage propose des concepts de type critère et variable de décision, il ne propose pas de primitives spécifier explicitement des contraintes de type égalité ou inégalité dans son formalisme de base (Biard, 2015).

Il faut souligner ici que le DSL dépendra vraisemblablement du type de problèmes abordés dans une classification qui est celle de la recherche opérationnelle. C'est cette reconnaissance préliminaire qui doit conduire à sélectionner un DSL adapté à la description du besoin de prendre un certain type de décisions selon le contexte (première étape du processus cognitif de la figure 2). Ici, le contexte est celui d'un problème de routage et d'ordonnancement, le DSL propose des éléments de syntaxe concrète pour aider à représenter ce besoin. Dans un premier temps, nous avons surtout abordé des décisions de nature opérationnelles, les choix tactiques restent un chantier à ouvrir qui dépendait des résultats acquis dans ce premier temps.

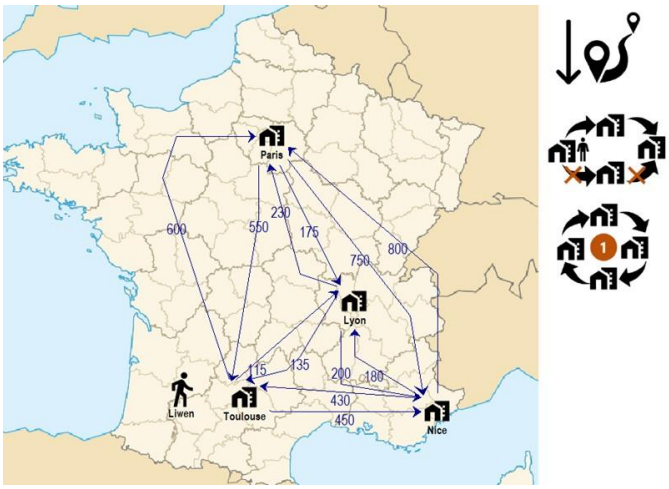

Figure 6 – Modèle des besoins de prise de décision inspiré du centre de décision de la méthode GRAI

Sur la figure 6, la toile de fond est une carte géographique du territoire identique à la figure 1, où on peut positionner de manière réaliste les villes qui prennent la syntaxe concrète d'un symbole de type « maison » avec une sémantique claire (Paris, Lyon, ...). Les flèches qui relient les villes sont des associations de type connexion routière et portent l'attribut de la distance. Le voyageur est symbolisé par une icône « forme humaine en marche », avec le prénom de ce voyageur. Cette icône peut se déplacer, elle est toujours attachée à une ville et cette connexion n'est pas matérialisée graphiquement. Cette partie gauche sur la figure 6 permet de désigner les variables de décision, c'est-à-dire l'ordre dans lequel l'icône humaine va se déplacer pour parcourir les quatre villes. Et par conséquent, les routes qui seront empruntées par lui.

De nouveau, comme pour le paragraphe b, des éléments graphiques sont disposés à droite de la figure. L'objet en tête de liste précise qu'il faut trouver une route entre chaque paire de villes avec une distance minimale à parcourir. Le suivant explique qu'il s'agit d'un flux traversant une et une seule fois toute ville. Le dernier explique qu'il ne fera qu'une et une seule tournée. Ces objets forment le corps explicite du centre de décision (variables, objectif et contraintes).

3 PRINCIPE DES TRANSFORMATIONS DE MODELES

Nos modules au sein de cette architecture produisent des modèles. Mais comment se réalise cette production ? L'IDM a développé une théorie qui permet d'emprunter cette voie et de réaliser de tels pontages entre modèles. Les modèles y ont été d'abord considérés comme des artefacts techniques peuplant le cycle de vie de l'ingénierie logicielle (c'est-à-dire à un niveau applicatif, nous ne sommes plus au niveau fonctionnel). Les modèles prennent ce caractère opérationnel quand ils permettent d'élaborer et de contrôler la production de contenus informatiques. Le génie logiciel a utilisé ces techniques pour la génération de codes d'haute-fidélité sur des systèmes critiques (référence au langage Lustre, par exemple). Mais dans le cadre d'une architecture orientée services, il est aussi possible de maîtriser une exécution conforme à une spécification donnée par des appels réalisés intelligemment grâce à des modèles de services numériques. En visant la production d'un code en langage OPL sur notre prototype, nous adoptons les mêmes démarches.

Ce sont des techniques dites de transformation des modèles (M to M) qui sous-tendent ces capacités de production logicielle. Une transformation des modèles fait d'abord recours à la méta-modélisation. Un méta-modèle est en quelque sorte le moule formel du modèle. Selon (Benaben, 2012), il est : « un modèle qui définit le langage utilisé pour exprimer un modèle ». (Chapron, 2006) est en phase avec cette vision quand il définit un méta-modèle comme : « une description de tous les concepts d'un langage. La méta-modélisation permet de définir directement les concepts manipulés, la sémantique et la syntaxe du langage liée à l'utilisation de ces concepts ». Cette vision est confirmée par (James et al., 2013) et (Karagiannis et al., 2016). Les méthodes de transformation des modèles ont fait l'objet de nombreuses études. (Czarnecki & Helsen, 2003) proposent une classification des approches générales existantes. Proche de notre problématique, (Cuadrado et al., 2014), en mobilisant le langage de transformation ATL, transforment un modèle source qui est un modèle de traitement dématérialisé de l'information (un *workflow* spécifié avec plusieurs options de langage possibles : diagrammes d'activité UML, YAWL et BPMN) en un Réseau de Petri. (Gómez et al., 2018) expliquent comment transformer un modèle physique exprimé en Modelica pour produire le fichier de lancement de sa simulation dynamique. L'environnement EMF (Eclipse Modeling Framework) est utilisé par ces auteurs comme cadre supportant la transformation.

L'OMG a proposé une structuration de l'avancement de processus d'ingénierie dirigée par les modèles, gradué en trois niveaux de modélisation quand on passe du besoin à la solution :

- Le CIM pour Computer Independent Model,
- Le PIM pour Platform Independent Model,
- Le PM pour Platform Model, et le PSM pour Platform Specific Model.

Cette représentation porte le nom d'architecture MDA (Model Driven Architecture) (Mellor et al., 2004). Chaque modèle est écrit avec un langage approprié.

Les problèmes d'interopérabilité entre partenaires impliqués dans différentes configurations de réseaux de collaboration ont été traités selon ces principes architecturaux. Nous citons l'ingénierie de système d'information de médiation comme un domaine où la conception part de modèles métier de type CIM (Mu et al., 2017). Le PIM y représente les connaissances techniques par un modèle de processus en BPMN. Enfin, le PSM est une implantation dans une architecture de services avec une technologie de bus d'entreprise. En amont d'un tel problème, la conception même du réseau de collaboration peut également mobiliser une structure de connaissances métier supplémentaires apportées par des ontologies (Wang et al., 2018).

Dans notre étude, nous avons suivi les mêmes raisonnements :

- Le niveau CIM est une spécification de haut niveau d'une fonction de prise de décision à mettre en œuvre et véhiculant les connaissances métier. C'est le sens que nous donnons à notre modèle de besoins de prise de décision,
- Le niveau PIM est une étape intermédiaire allant vers l'exécution de la fonction décrite par le CIM et mobilisant des connaissances de recherche opérationnelle. Mais à ce stade, aucun environnement de développement n'a été choisi. Toutes les options restent ouvertes. C'est le sens donné à notre modèle mathématique pour l'aide à la décision.
- Le niveau PSM dicte l'implémentation du PIM dans un environnement d'exécution connu qui est nommé PM. En faisant le choix du solveur IBM CPLEX, et, nous instancions le concept de PM et donc du langage OPL. Le PSM correspond aux deux fichiers du modèle logique écrits en OPL.

A chaque étape dans la logique de fonctionnement de notre environnement, c'est-à-dire dans chaque module de son architecture fonctionnelle, une transformation de modèles va donc être opérée (passage CIM-PIM, puis PIM-PSM). En quelque sorte, le module devient un atelier où des opérations sont exécutées. Pour les comprendre, il nous faut d'abord définir les méta-modèles pour nos 3 types de modèles. A partir de cette base, une série de règles de correspondance entre les méta-modèles permet de faire ladite transformation, comme illustrée sur la figure 7.

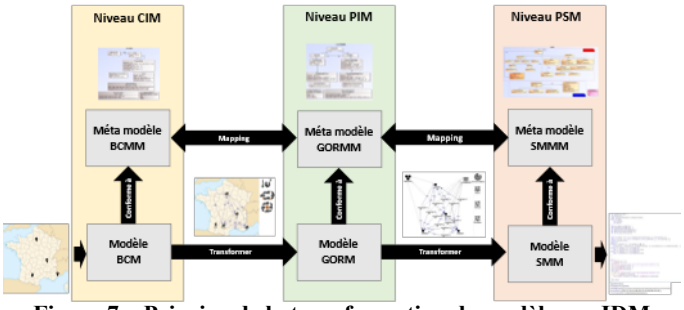

Figure 7 – Principe de la transformation de modèles en IDM

Dans l'écosystème de l'IDM, le principal ingrédient pour soutenir la transformation des modèles est le langage de transformation des modèles (MTL). Ainsi, (Agrawal et al., 2006) introduisent un langage basé sur une représentation graphique pour soutenir la mise en œuvre de transformation particulièrement efficaces qui assurent des traductions entre modèles, ou entre modèles et codes. VMTL est un MTL (Acreţoaie et al., 2018). ATL en est un autre, largement cité dans la littérature (Jouault et al., 2008).

Nous avons choisi la plate-forme de méta-modélisation ADOxx pour réaliser nos développements. Elle possède son propre MTL appelé AdoScript. Cette plateforme est largement utilisée pour construire des preuves de concept (PoC) dans les activités de recherche en ingénierie dirigée par les modèles. Elle a créé une communauté OMILAB (Fill & Karagiannis, 2013) qui partage des expériences et des applications ainsi développées dans une optique de réutilisabilité. Ayant rejoint cette communauté, nous avons pu dresser le constat d'une absence d'application, dans sa panoplie des projets finis ou en cours, ayant un voisinage thématique avec notre travail.

Toutefois, une recherche récente traite de l'usage potentiel de MTL dans la pratique (Burgueño et al., 2019), ainsi que des rôles actuel et futur d'un MTL en intelligence artificielle. Nous nous inscrivons dans la même veine de travaux.

4 APPLICATION SUR LE CAS D'ETUDES

Ce sont ces méta-modèles et ces transformations de modèles que nous décrivons dans ce paragraphe. Le contenu est représentatif du travail fourni pour construire un prototype expérimental dans l'outil ADOxx et en démontrer la pertinence sur notre cas d'études. C'est un mécanisme à base de règles qui régit le MTL.

Nous mobilisons trois types de règles de transformation, règle de « mapping », règle de navigation et règle de « parsing ». Une règle de mapping traite une mise en correspondance entre les deux méta-modèles. Elle pointe des concepts du méta-modèle source et décrit explicitement l'effet de leur combinaison sur des concepts du méta-modèle cible. Une règle de navigation opère des traitements locaux dans le méta-modèle cible pour créer des éléments nouveaux en exploitant des dépendances locales. Une règle de parsing permet de faire des modifications du modèle qui sont induites par des annotations dans le méta-modèle.

La description complète et détaillée des transformations dépasse largement la taille raisonnable de cet article et a, de plus, un caractère très technique. Nous avons donc décidé de présenter ces règles dans leurs grandes lignes.

- **Transformation CIM-PIM**

Le premier atelier transforme le modèle de besoin de prise de décision (niveau CIM), que nous nommons de manière synthétique BCM, en un modèle mathématique d'aide à la décision (niveau PIM) appelé GORM. La palette graphique du DSL utilisé pour élaborer le BCM est une expression directe de ce qu'est le méta-modèle (cf. figure 6). Nous le présentons de manière plus rigoureuse avec le premier méta-modèle à la source de la transformation, qui est consigné sur la figure 8. Nous l'avons appelé BCMM (Zhang et al., 2020).

Le résultat de la transformation est notre modèle cible GORM (cf. figure 5). Il doit être conforme au méta-modèle de la figure 9, appelé GORMM.

Ce sont des diagrammes de classe UML qui sont utilisés pour décrire ces méta-modèles.

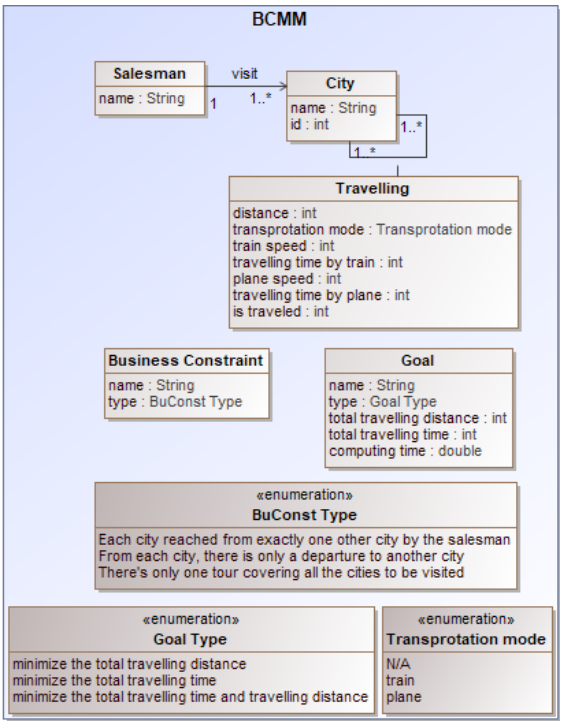

Figure 8 – Méta modèle BCMM (niveau CIM)

Algorithme de transformation CIM-PIM

Règles de mapping entre BCMM et GORMM

Travail sur les relations directes entre concepts

- Chaque ville dans le modèle métier devient un nœud dans le graphe. Son nom (id de City) devient un numéro (id de Node).

Travail sur les maillages de concepts

- Chaque trajet pouvant être emprunté par le voyageur (relation réflexive sur le concept City avec la classe association Travelling) devient un arc dans le graphe. Un arc est un lien dirigé entre deux nœuds. Il prend pour attribut la donnée qui sera utilisée pour poursuivre l'objectif (edgeLength).
- L'objet qui spécifie la traversée d'une et une seule ville dans le modèle métier est désagrégé. Il induit la désignation des arcs comme devenant des variables de décision avec deux indices (dvar Assignement).
- Le « flow in » n'autorisera qu'un trajet à être fait parmi tous les arcs qui entrent dans une ville. Il en va de même pour le « flow out » en sortie de ville.
- L'exigence d'une tournée unique induit la création de variables de décision permettant de tracer la tournée en cours de calcul et donc de contrôler l'absence de sous tournée (dvar NoSubTour).
- Chaque trajet est inducteur d'une contrainte qui n'est que pointée dans son existence à ce stade, mais qui servira à trouver le formulation mathématique associée dans la bibliothèque de formules. Il en va de même pour la tournée unique et pour le choix de la fonction objectif.

Règle de navigation dans le GORMM (niveau PIM)

- Regroupement des données portées par les concepts qui composent le graphe (CityNode et RoutingEdge) dans des ensembles homogènes (CityNodeSet et RoutingEdgeSet) qui seront ensuite utiles pour écrire le code logique dans sa partie déclarative.

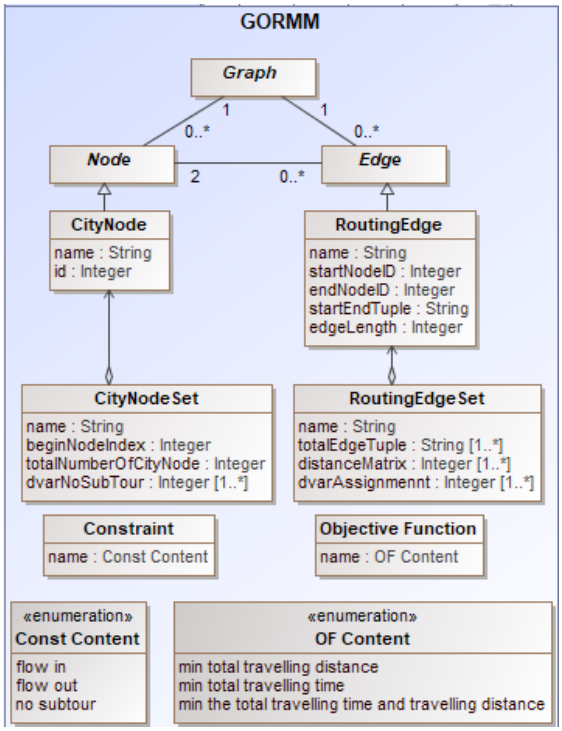

Figure 9 – Méta modèle GORMM (niveau PIM)

- **Transformation PIM-PSM**

Le deuxième atelier de transformation concerne la génération du modèle logique de calcul de la décision à partir du GORM. Ce modèle est appelé SMM. Nous présentons le méta-modèle

auquel il doit être conforme, appelé SMMM (Assouroko & Denno, 2016), sur la figure 10.

Algorithme de transformation PIM-PSM

Règles de mapping entre GORMM et SMMM

- Développement des types de données dans la partie déclarative du code OPL avec une distinction faite entre paramètres du modèle et variables de décision, chacun ayant ensuite son mot clé dans le langage OPL.
- Utilisation de pointeurs pour construire les lignes de codes des relations mathématiques de la partie exécutable à partir des patrons de formulation extrait d'une bibliothèque de formules en OPL (fonction à optimiser et contraintes).
- Une règle de transformation permet de générer des composants en instanciant des patrons pour lesquels les paramètres attendus sont renseignés à partir des données du GORM.

Règle de navigation dans le SMMM (niveau PSM)

- Assemblage des lignes de déclaration et des lignes d'exécution selon les règles de bonne construction d'un code OPL

Règle de parsing du SMM

- Ajout des mots clés du langage OPL aux parties déclaratives et aux parties exécutables

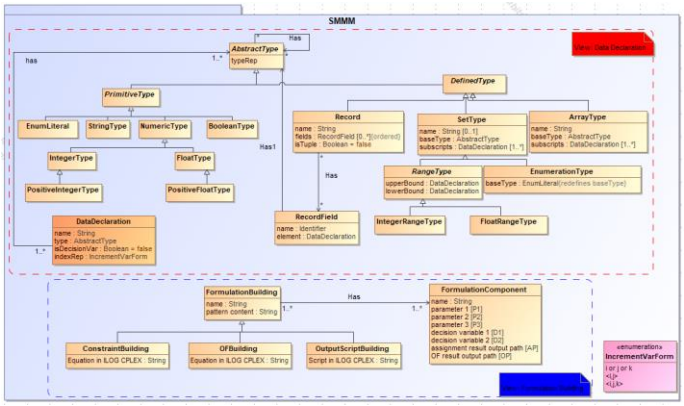

Figure 10 - Méta modèle SMMM (niveau PSM)

L'ensemble de la chaine de production est résumé sur la figure 11.

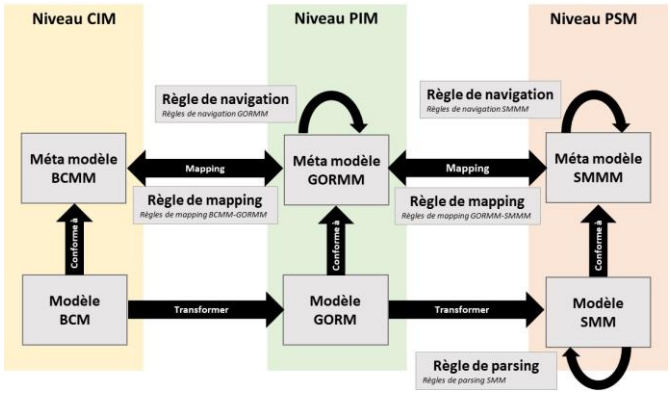

Figure 11- Schématisation de la chaine de transformations des modèles du niveau CIM à celui du PSM

Le SMM permet de lancer le calcul sur IBM CPLEX. Avec la solution ainsi obtenue, les chemins retenus sont (1,4), (4,3), (3,2), (2,1), présenté sous le forme (id de ville de départ, id de ville d'arrivé). Il en découle une tournée (Toulouse, Paris, Lyon, Nice, Toulouse).

Un décideur ayant contribué à l'élaboration du BCM aura naturellement envie de retrouver des propositions dans le même univers de discours. Il appréciera certainement de disposer d'une interface homme-machine (IHM) afin d'obtenir de manière efficace et compréhensible des connaissances concrètes sur le TSP ainsi résolu. Nous avons donc débuté le développement d'un retour vers le modèle BCM pour visualiser ces résultats de manière graphique en changeant simplement la couleur des chemins retenus entre la série de villes formant la tournée.

5 CHANGEMENT DE PROBLEME

Afin de mettre à l'épreuve notre prototype développé sous ADOxx, nous lui avons soumis un autre problème TSP. Il s'agit maintenant d'une formulation où la fonction à,optimiser est un temps minimal de voyage. Ce passage à une caractérisation de chaque trajet entre deux villes par un temps de voyage plutôt qu'une distance est accompagné par une interaction homme-machine dans la conception du BCM. Cela permet de spécifier le mode de transport employé pour chaque trajet. Ainsi, nous avons mixé les modes de transport en choisissant l'avion pour les chemins de plus longue distance, c'est-à-dire entre Paris et les villes du sud, et le train pour les autres. Sur la figure 12, l'icône de la fonction à optimiser a changé. De plus, la contrainte de la tournée unique a été supprimée.

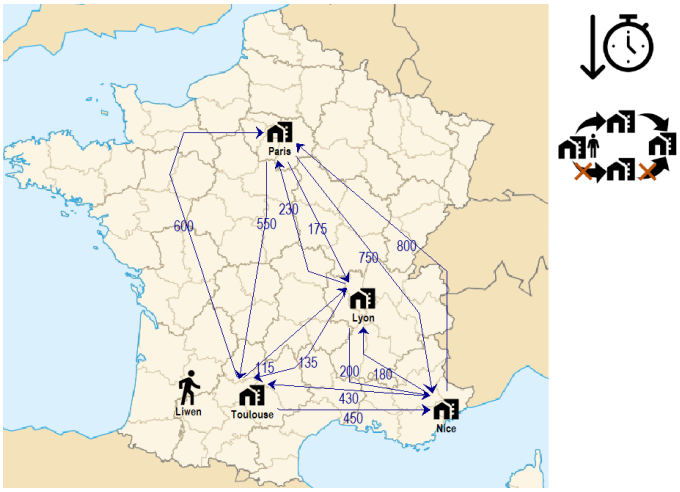

Figure 12- Spécification d'un TSP différent

Le résultat est présenté sur la figure 13. Nous avons maintenant deux tournées distinctes. Une se fait en avion sur l'aller-retour Paris Toulouse, l'autre se fait en train sur l'aller-retour Lyon-Nice.

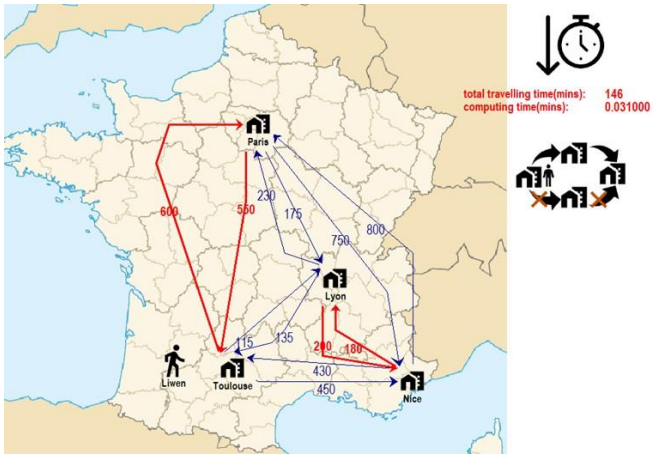

Figure 13- Spécification d'un TSP différent

Les deux modèles intermédiaires GORM et SMM ont été contrôlés et ont permis de valider notre mise en œuvre des règles de transformation.

6 CONCLUSIONS ET PERSPECTIVES

Nous avons présenté deux niveaux d'architecture de notre environnement d'aide à la prise de décision. La couche fonctionnelle est inspirée du processus cognitif de l'expert en recherche opérationnelle, décrit en en quatre phases. La couche logique implémente ces fonctions attendues et a permis de décrire une suite des modules de traitement de trois modèles dans le respect des principes du MDA (CIM, PIM et PSM).

Le développement du prototype sous ADOXX a été réalisé dans le contexte d'un langage spécifique à un problème de routage et ordonnancement. La preuve de concept est obtenue à partir d'une variation du problème du voyageur de commerce. L'implémentation de cette idée d'une « usine » de la connaissance sur le problème du voyageur de commerce (TSP) prouve la faisabilité d'une telle chaine IDM en pratique.

Rappelons que notre but était d'explorer cette voie et de défricher un terrain vierge, celui de l'emploi de l'IDM dans les problèmes de prise de décision. Nous avons fait de nombreuses hypothèses dans cette prospection que nous avons soulignées dans cette présentation (DSL, décision opérationnelle, bibliothèque de formules, solveur). Nous nous attacherons à les lever progressivement pour donner plus de capacité à cet environnement de prise de décision, et c'est la perspective des chantiers à venir.

Le test de sensibilité des modèles à une modification du problème à traiter reste une piste à explorer afin d'estimer l'impact sur les différents stades du processus de la figure 2 et donc la capacité de l'IDM à faire face à la demande de changement du problème à résoudre. Ne perdons pas le cap.

Fort de cette première avancée, la suite de notre travail s'inscrira aussi dans un cadre plus difficile à traiter. Il s'agit du problème de coordination opérationnelle d'une prise en charge à domicile de patients. Ce problème est connu et intitulé « *Home HealthCare Routing and Scheduling Problem* » (HHCRSP) dans la communauté de recherche opérationnelle. Il est lui aussi sujet à de nombreuses variations liées à la différence entre les organisations en charge de coordination, leurs pratiques, leur territoire de chalandise, leur réseau d'intervenants, les moyens de transport, les horaires de travail, etc. Nous avons choisi d'étendre les compétences de notre prototype sur ce nouveau cas d'études. Nous avons produit un modèle mathématique d'aide à la décision qui est en fait un TSP clustérisé multi-voyageurs. Pour cette nouvelle étude, une partie de ce que nous venons de présenter est *de facto* réutilisable.

7 REFERENCES

Acreţoaie, V., Störrle, H., & Strüber, D. (2018). VMTL: A language for end-user model transformation. *Software & Systems Modeling*, *17*(4), 1139–1167. https://doi.org/10.1007/s10270-016-0546-9

Agrawal, A., Karsai, G., Neema, S., Shi, F., & Vizhanyo, A. (2006). The design of a language for model transformations. *Software & Systems Modeling*, *5*(3), 261–288. https://doi.org/10.1007/s10270-006-0027-7

Allweyer, T. (2016). *BPMN 2.0: Introduction to the standard for business process modeling*. BoD–Books on Demand.

Assouroko, I., & Denno, P. O. (2016). *A metamodel for optimization problems* (NIST IR 8096; p. NIST IR 8096). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.IR.8096

Benaben, F. (2012). *Conception de Système d'Information de Médiation pour la prise en charge de l'Interopérabilité dans les Collaborations d'Organisations* [Thesis, Institut National Polytechnique de Toulouse]. https://hal-mines-albi.archives-ouvertes.fr/tel-01206234/document

Biard, T., Le Mauff, A., Bigand, M., & Bourey, J.-P. (2015). Separation of decision modeling from business process modeling using new "Decision Model and Notation"(DMN) for automating operational decision-making. *Working Conference on Virtual Enterprises*, 489–496.

Burgueño, L., Cabot, J., & Gérard, S. (2019). The Future of Model Transformation Languages: An Open Community Discussion. *The Journal of Object Technology*, *18*(3), 7:1. https://doi.org/10.5381/jot.2019.18.3.a7

Chapron, J. (2006). *L'urbanisme organisationnel: Méthode et aides à la décision pour piloter l'évolution du système d'information de l'entreprise* [Phdthesis, Ecole Nationale Supérieure des Mines de Saint-Etienne]. https://tel.archives-ouvertes.fr/tel-00796061/document

Chen, D., Vallespir, B., & Doumeingts, G. (1997). GRAI integrated methodology and its mapping onto generic enterprise reference architecture and methodology. *Computers in Industry*, *33*(2–3), 387–394.

Cuadrado, J. S., Guerra, E., & de Lara, J. (2014). A Component Model for Model Transformations. *IEEE Transactions on Software Engineering*, *40*(11), 1042–1060. https://doi.org/10.1109/TSE.2014.2339852

Czarnecki, K., & Helsen, S. (2003). Classification of model transformation approaches. *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*, *45*, 1–17.

Fill, H.-G., & Karagiannis, D. (2013). On the Conceptualisation of Modelling Methods Using the ADOxx Meta Modelling Platform. *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, *8*(1), 4–25. https://doi.org/10.18417/emisa.8.1.1

Gómez, F. J., Vanfretti, L., & Olsen, S. H. (2018). CIM-Compliant Power System Dynamic Model-to-Model Transformation and Modelica Simulation. *IEEE Transactions on Industrial Informatics*, *14*(9), 3989–3996. https://doi.org/10.1109/TII.2017.2785439

Gutin, G., & Punnen, A. P. (2006). *The traveling salesman problem and its variations* (Vol. 12). Springer Science & Business Media.

James, W., Athansios, Z., Nicholas, M., Louis, R., Dimitios, K., Richard, P., & Fiona, P. (2013). What do metamodels really look like. *Frontiers of Computer Science*.

Jouault, F., Allilaire, F., Bézivin, J., & Kurtev, I. (2008). ATL: A model transformation tool. *Science of Computer Programming*, *72*(1–2), 31–39. https://doi.org/10.1016/j.scico.2007.08.002

Karagiannis, D., Mayr, H. C., & Mylopoulos, J. (2016). *Domain-Specific Conceptual Modeling: Concepts, Methods and Tools*. Springer International Publishing. http://public.eblib.com/choice/publicfullrecord.aspx?p=4587053

Mellor, S. J., Scott, K., Uhl, A., & Weise, D. (2004). *MDA distilled: Principles of model-driven architecture*. Addison-Wesley Professional.

Mu, W., Benaben, F., Boissel-Dallier, N., & Pingaud, H. (2017). Collaborative Knowledge Framework for Mediation Information System Engineering. *Scientific Programming*, *2017*, 1–18. https://doi.org/10.1155/2017/9026387

Wang, T., Montarnal, A., Truptil, S., Benaben, F., Lauras, M., & Lamothe, J. (2018). A Semantic-checking based Model-driven Approach to Serve Multi-organization Collaboration. *Procedia Computer Science*, *126*, 136–145. https://doi.org/10.1016/j.procs.2018.07.217

Zhang, L., Fontanili, F., Lamine, E., Bortolaso, C., Derras, M., & Pingaud, H. (2020). A Systematic Model to Model Transformation for Knowledge-Based Planning Generation Problems. In H. Fujita, P. Fournier-Viger, M. Ali, & J. Sasaki (Eds.), *Trends in Artificial Intelligence Theory and Applications. Artificial Intelligence Practices* (pp. 140–152). Springer International Publishing. https://doi.org/10.1007/978-3-030-55789-8_13