Table des matières

Première partie

Etat de l'Art

Chapitre 1

Motivations et problématique

Sommaire

1.1 Contexte industriel

1.1.1 Qu'est ce qu'un Smart Grid ?

Le terme « Smart Grid » est une appellation générale désignant les technologies « intelligentes » qui « augmentent » [1] les réseaux électriques actuels en améliorant leurs performances.

Cette « augmentation » peut servir différents objectifs dépendant des limites du système électrique existant, du cadre de régulation, ainsi que de l'orientation politique des états. Il existe autant de définitions du terme « Smart Grid » que d'objectifs motivant son implémentation.

En Europe, les exigences de régulation et les volontés politiques des états membres en matière d'écologie favorisent l'intégration des énergies renouvelables et la participation

1. À la manière de « l'augmentation de l'humain » qui désigne l'amélioration technique des performances humaines, aussi bien physiques, intellectuelles qu'émotionnelles [2013-1].

active des consommateurs. Les Smart Grids européens sont donc synonymes de producteurs d'énergie renouvelables et de compteurs intelligents. Pour l'*European Technology Platform SmartGrids* (ETP), les Smart Grids sont ainsi « des réseaux électriques qui intègrent de manière intelligente les comportements et les actions de tous les acteurs connectés — les producteurs, les consommateurs et ceux qui consomment et produisent en même temps — pour garantir une fourniture d'électricité efficiente, durable, économique et sûre. » [2].

Les Smart Grids américains mettent, quant à eux, l'accent sur la modernisation des réseaux de transport d'énergie souvent très mal entretenus. Les États Unis doivent en effet faire face un nombre de coupures de courant qui ne cesse d'augmenter, passant de 76 pannes en 2007 à plus de 300 en 2011 [2014-3]. Ces pannes sont aussi fréquentes qu'importantes. En 2003, un *blackout* dans l'Ohio prive 50 millions de personnes d'électricité et coûte 6 milliards de dollars [2005-4]. Ces coupures sont dues à la combinaison de trois facteurs [2014-5] :

- une infrastructure électrique vieillissante, en grande partie mécanisée et souffrant du manque d'investissement ;
- une forte croissance démographique ;
- des conditions climatiques de plus en plus extrêmes.

Aux États Unis, moderniser le réseau électrique en investissant dans les Smart Grids est ainsi essentiellement guidé par un impératif de fiabilité. Pour définir les Smart Grids, le département de l'énergie américain (*United States Department of Energy*) dresse une liste d'exigences mettant en avant la sûreté des réseaux électriques[6]. Selon cette définition, un Smart Grid doit ainsi répondre aux exigences suivantes :

- auto-réparation en cas d'événements perturbateurs ;
- permettre la participation active des consommateurs ;
- résilience face aux attaques physiques et cybernétiques ;
- fournir une électricité de qualité face aux besoins du 21$^{\text{ème}}$ siècle ;
- intégration de moyens de production et de stockage d'électricité ;
- exploitation optimisée des infrastructures et conduite efficace des réseaux électriques.

À partir de ces deux définitions, nous constatons que l'obsolescence du système électrique (dans le cas des États Unis) et les impératifs régulatoires et écologiques (dans le cas de l'Europe) font de la mise à niveau des réseaux électriques un enjeu crucial. Cette mise à niveau passe d'abord par le déploiement des Technologies de l'Information et de la Communication (TIC) et donc par l'implémentation des Smart Grids, plutôt que par le remplacement massif des infrastructures électriques existantes.

Envisager uniquement le renforcement et de remplacement massif des infrastructures électriques du réseau n'est en effet pas une solution optimale et semble difficilement réalisable compte tenu de la croissance démographique constante des zones urbaines et du coût important des investissements à consentir [7].

1.1.2 Contexte économique, cadre législatif et modes de consommation en constante mutation

Le système électrique actuel est mis à l'épreuve par l'entrée en jeu de nouveaux acteurs économiques et de nouveaux cadres législatifs.

D'une part, la libéralisation du marché de l'électricité permet à un producteur quelconque de produire et de vendre son électricité après s'être convenablement raccordé au réseau électrique de distribution ou de transport (donc entre les centrales de production et les consommateurs). Contrairement aux centrales de production localisées en amont du réseau électrique de transport, ces sources d'énergie sont dites distribuées. Les sources d'énergie distribuées perturbent fortement la stabilité des réseaux électriques. En injectant du courant, elles peuvent déséquilibrer le niveau de tension du réseau, endommageant ainsi les équipements du système électrique tels que les transformateurs, les lignes et les protections.

D'autre part, les enjeux environnementaux encouragent le recours aux énergies renouvelables. Dans sa directive du 23 avril 2009, la commission européenne fixe à 20% la part de contribution des ressources renouvelables dans la production totale d'énergie à l'horizon de 2020. Les réseaux électriques sont de ce fait amenés à connaître une croissance constante de ces producteurs décentralisés dans les années à venir.

Cette même directive fixe à 20% la réduction des émissions de gaz à effet de serre par rapport à leur niveau en 1990 et à 20% l'augmentation de l'efficacité énergétique. Or les pics de consommation en période de pointe sont coûteux et émetteurs de CO_2. Ces pics correspondent typiquement la fin de journée d'un jour de semaine lorsque les gens allument leurs télévisions, plaques électriques et autres outils électroménagers en rentrant chez eux. En effet, pour maintenir l'équilibre entre l'offre et la demande d'électricité en cas de pics de consommation, les producteurs recourent à des centrales à charbon, à fioul et à gaz comme l'illustre la figure 1.1.

Ces pics s'intensifient significativement avec l'émergence de nouveaux usages de consommation d'énergie dont, notamment, la mobilité électrique. En France, les pouvoirs publics estiment à deux millions le nombre de véhicules électriques en circulation en 2020. L'impact de ces véhicules sur l'équilibre entre l'offre et la demande n'est pas négligeable. En effet, la recharge complète d'un véhicule électrique ayant 150 km d'autonomie est équivalente en termes d'appel de puissance à :

- un chauffe-eau si la recharge s'effectue en 8 h (recharge normale) ;
- un immeuble si la recharge s'effectue en 1 h (recharge accélérée) ;
- un quartier urbain si la recharge s'effectue en 3 min (recharge rapide).

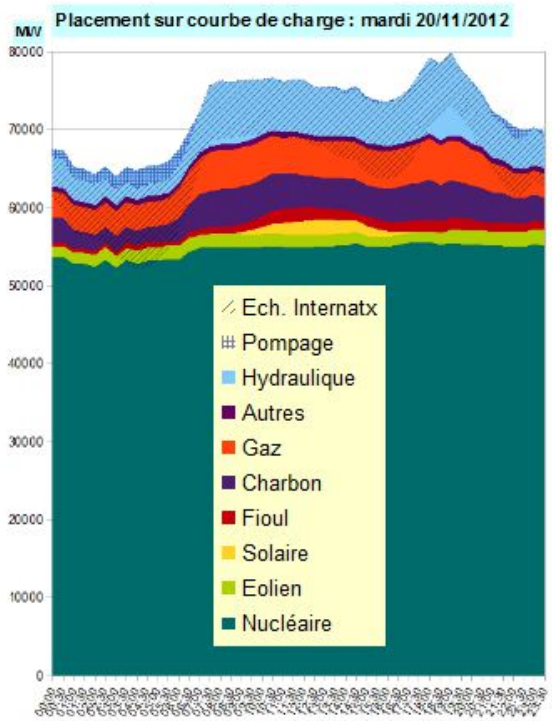

FIGURE 1.1 – Placement du type d'énergie sollicitée sur la courbe de charge française du mardi 20 novembre 2012 (à remplacer par une plus belle/récente)(source RTE)

1.1.3 Architecture des Smart Grids : vers des réseaux électriques flexibles et communicants

Pour faire face aux mutations du contexte énergétique, les gestionnaires de réseaux électriques ne peuvent plus compter uniquement sur la conduite prévisionnelle du réseau électrique (peu réactive face à l'intermittence des énergies renouvelable par exemple) ni envisager le redimensionnement du réseau (onéreux et non optimal).

La solution réside dans l'automatisation de la conduite des reseaux electrique, grace a l'acquisition et l'exploitation en temps réel d'informations sur l'etat des reseaux. Cela passe par le deploiement d'un reseau informatique au niveau des infrastructures electriques, et la mise en place dans du Système d'Information (SI) d'outils pour l'exploiter.

Ainsi équipés, les réseaux électriques s'apparentent à une toile d'araignée où les mailles interagissent constamment via des liens de communication. Ces mailles correspondent aux acteurs du système électrique : consommateurs, producteurs ou les deux à la fois. Outre l'électricité, ces acteurs produisent et consomment de l'information en temps réel grâce aux modules logiciels dont ils sont équipés et à divers moyens de télécommunication, tels que les réseaux mobiles ou le Courant Porteur de Ligne (CPL). Ce partage permanent et instantané d'informations entre les équipements préserve la stabilité du système électrique tout en augmentant son efficacité énergétique.

Pour illustrer les possibilités offertes par les TIC, prenons l'exemple du pic de consommation de la fin de journée d'un jour en semaine. Grâce aux TIC, il devient possible d'agir sur la demande plutôt que sur l'offre. Le distributeur d'électricité, s'appuyant sur des points de contrôle distants et des compteurs intelligents installés chez les clients pour envoyer et recevoir des informations et des consignes, peut alors addresser des demandes d'effacement aux consommateurs moyennant des incitations tarifaires. Il peut s'agir d'une coupure du chauffage pendant 15 min à 30 min d'un foyer ou d'un bureau bien isolé. Sans incidence sur le confort des consommateurs, ces demandes d'effacement aident à lisser la courbe de charge aux heures de pointe tout en évitant de mobiliser des centrales de production.

Nés de la convergence des réseaux électriques et des TIC, les Smart Grids se composent de trois couches que nous retrouvons dans la figure 1.2 :

1. le premier niveau correspond à l'infrastructure et aux équipements électriques acheminant l'électricité tels que les lignes et les transformateurs ;
2. le second niveau correspond à l'infrastructure de communication composée de différentes technologies de télécommunication comme la fibre optique, le CPL, ou encore la Troisième Génération (3G) ;
3. le troisième niveau correspond aux applications informatiques qui incarnent « l'intelligence » du réseau. En utilisant des informations délivrées en temps réel, ces applications calculent des consignes à envoyer aux équipements concernés et automatisent ainsi la conduite du système électrique. Cette intelligence est centralisée au niveau des centres de conduites du réseau ou distribuée sur les équipements électriques.

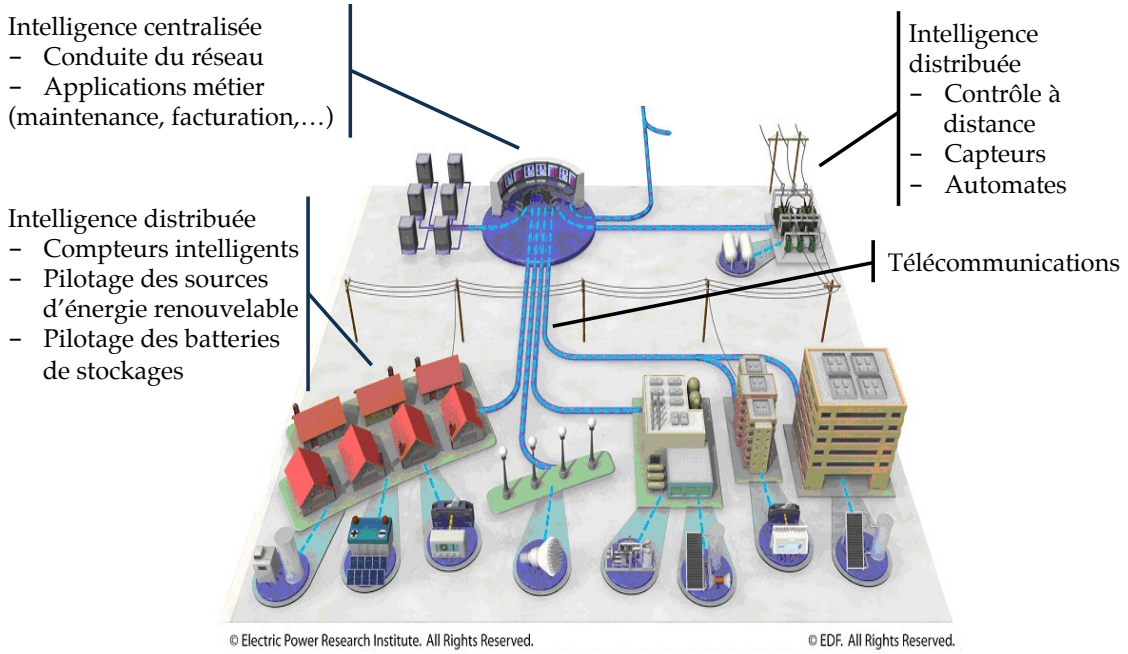

FIGURE 1.2 – Architecture des Smart Grids [2006-8]

1.2 Problématique industrielle

En traitant les données envoyées en temps réel par les capteurs installés sur les équipements électriques et chez les consommateurs, les SI calculent des consignes destinées à des organes télécommandés permettant ainsi de piloter les réseaux électriques à distance. Cette automatisation de la gestion des réseaux est une solution pour les adapter rapidement face aux contraintes qu'introduit l'intégration des énergies renouvelables et des nouveaux usages [7]. Les SI sont donc au cœur des enjeux Smart Grids.

L'implémentation des Smart Grids va ainsi de pair avec la mise à niveau des SI des gestionnaires du réseau électrique. En effet, ces SI doivent pleinement intégrer les évolutions qu'induisent les Smart Grids au niveau des processus métier du gestionnaire de réseau, des acteurs impactés, des informations échangées ainsi que des applications informatiques et des infrastructures techniques sous-jacentes. Parmi ces évolutions nous citons :

- les nouveaux flux d'information provenant du réseau électrique ;
- l'entrée en jeu de nouveaux acteurs tels que les producteurs décentralisés (éolien, photovoltaïque) ;
- les nouveaux équipements communicants comme le compteur Linky (Électricite Réseau de Distribution France (ERDF) annonce le déploiement de 30 millions de compteurs

d'ici 2020 [2]) ;

- les nouvelles réglementations et directives européennes (dans le cas des gestionnaires de réseaux européens) ;
- les nouveaux usages comme les véhicules électriques ou encore les maisons connectées.

Outre leurs SI, les gestionnaires de réseaux électriques doivent faire évoluer leurs stratégies de développement en envisageant de nouveaux modèles métier et de nouveaux partenaires, tout en tenant compte de l'émergence des nouvelles technologies et des exigences du législateur. Une étude américaine, menée par IBM, CISCO, EPRI et South Carolina Edison, fait état de cinq thèmes stratégiques clés pour l'implémentation des Smart Grids :

- **permettre au consommateur de contrôler sa consommation d'énergie** et de réduire son empreinte carbone en utilisant des équipements intelligents et des véhicules électriques et en produisant de l'énergie renouvelable à domicile ;
- **améliorer la sécurité et la productivité des employés** en mettant par exemple à leur disposition des outils performants pour le contrôle à distance, des équipements de protection, et des applications mobiles ;
- **intégrer des sources d'énergie renouvelables** distribuées sur le réseau en assurant la protection des équipements électriques, le stockage de l'énergie et la stabilité du réseau ;
- **améliorer l'efficience et la résilience du réseau** à travers les systèmes de mesure en temps réel, l'analyse et le contrôle à distance ;
- **fournir les informations et la connectivité nécessaires** en developpant une infrastructure TIC pour répondre aux besoins d'informatisation du réseau électrique. Ce dernier point est une condition *sine qua non* de la réalisation des quatre précédents.

Compte tenu de ces thèmes clés, les gestionnaires de réseaux envisagent des stratégies impliquant l'adoption de technologies Smart Grids. La mise en œuvre effective de ces stratégies demande l'automatisation des actions sur le réseau et du traitement des données entrainent donc le déploiement de nouveaux SI. Afin d'appréhender ces paradigmes naissants, des scénarios métier sont élaborés mais il est indispensable de les éprouver et de les valider avant d'envisager leur implémentation finale.

Plusieurs démonstrateurs physiques ont été déployés sur le terrain [3]. Ces projets pilotes permettent de mener des expérimentations en conditions réelles pour tester des fonctions et des services, comme par exemple le démonstrateur InfiniDrive [4] pour le pilotage des infrastructures de recharge de véhicules électriques ou le démonstrateur Venteea [5] pour l'intégration de forte capacité éolienne dans un réseau rural. Cependant, les démonstrateurs nécessitent que le gestionnaire de réseau de distribution recrute des clients industriels et/ou

2. www.erdf.fr/Linky
3. www.erdf.fr/Carte_demonstrateurs_Smart_Grids
4. avem.fr/actualite-erdf-et-le-groupe-la-poste-lancent-le-projet-infini-drive-a-nice-3450.html
5. www.venteea.fr

domestiques qui acceptent d'avoir du matériel à tester chez eux. De plus, leur exploitation reste limitée par les réglementations en cours. Enfin, leur mise en place se révèle souvent longue et coûteuse.

En plus de ces démonstrateurs, des réseaux de distribution d'expérimentation grandeur nature tel que Concept Grid [6], implanté à Électricite De France (EDF) Lab, permettent de tester les nouveaux équipements avant leur installation sur les réseaux du distributeur ERDF. Ces réseaux ont l'avantage de permettre la conduite de stress tests en conditions perturbées, impossibles à réaliser dans le cadre de démonstrateurs, ceux-ci impliquant de vrais clients. Cependant, la taille réduite de ces réseaux reste limitante.

Pour pallier toutes ces limitations, une troisième voie est la simulation. La simulation intègre les trois couches qui composent les Smart Grids : l'infrastructure électrique (transformateurs, lignes, charges, sources), l'infrastructure de télécommunication (réseaux mobiles, CPL) et enfin les SI qui les pilotent. Des simulateurs spécialisés dans la simulation de réseaux électriques (EMTP-RV, Dymola, PowerFactory, Eurostag, etc.) ainsi que des simulateurs de réseaux de télécommunication (OPNET, NS-3, OMNeT ++, etc.) ont déjà validé l'apport de la simulation dans leurs domaines respectifs. Toutefois, les SI sont souvent relégués à de simples modèles de calcul de consignes souvent développés en Matlab ou en C++ [2014-9].

Dans ce contexte, la problématique industrielle dans laquelle s'inscrivent nos travaux de recherche est donc la suivante :

Comment valider une stratégie de développement orientée Smart Grids à travers la simulation de sa déclinaison au niveau du SI du gestionnaire du réseau électrique ?

1.3 Problématique de recherche

Les technologies Smart Grids illustrent le défi que représente l'évolution des TIC et l'intégration des énergies renouvelables pour les gestionnaires de réseaux électriques. Jeremy Rifkin annonce même « une troisième révolution industrielle, fondée sur le couplage des technologies de l'Internet et des énergies nouvelles » [2012-10].

Mais à l'ère numérique, l'adaptation au changement relève des préoccupations de toute entreprise s'appuyant sur les TIC pour mener ses activités. Le très haut niveau de concurrence que connait le secteur des technologies de l'information stimule l'innovation. Or ces technologies sont de plus en plus le moteur qui fait progresser les métier de l'entreprise. Être capables de s'adapter continuellement à l'évolution rapide et constante des TIC représente donc un véritable challenge pour les entreprise d'aujourd'hui.

Pour mener à bien tout changement, il est primordial de commencer par une description représentative de « l'objet » à changer, qu'il s'agisse d'une nouvelle version d'un avion,

6. chercheurs.edf.com

d'une voiture, d'un ordinateur ou encore d'une entreprise [1997-11]. Cette description représentative revient à concevoir l'architecture de l'objet en question. L'architecture en tant qu'activité est centrale dans plusieurs disciplines, allant de l'architecture du bâtiment à l'architecture logicielle, en ce qu'elle est un outil indispensable à la construction d'artefacts respectant les qualités attendues de l'objet final. En précurseur, Zachman [1997-11] recommande d'appliquer les principes d'architecture à l'entreprise pour faire face aux changements dictés par l'innovation technologique.

Les SI sont en première ligne dès qu'il s'agit de l'évolution des TIC. Nos travaux se sont donc d'abord portés sur l'évaluation de l'impact de cette évolution sur les SI de l'entreprise. De ce fait, nous nous sommes d'abord appliqués à décrire l'architecture de ces SI [2014-12].

Les changements apportés par ces technologies impactent cependant, non seulement les SI, mais l'entreprise dans son ensemble : de sa stratégie à ses partenaires, en passant par ses objectifs, ses clients et ses processus métier. Par exemple, l'utilisation des véhicules électriques fait évoluer le SI du gestionnaire du réseau électrique qui doit mettre en place de nouvelles applications capables de bien gérer leur recharge. Mais il doit aussi faire évoluer son modèle métier en mettant à disposition de nouveaux contrats client favorisant la recharge hors de la période des pics de charge par exemple, ou encore créer de nouveaux partenariats avec les constructeurs automobiles comme dans le cas de EDF et Renault-Nissan qui collaborent sur un système de recharge pour véhicule électrique permettant à ce dernier de communiquer avec les bornes de recharge.

Évaluer l'adoption de nouvelles technologies de l'information oblige à prendre en compte l'entreprise dans son ensemble afin de garantir une cohérence entre la stratégie d'évolution adoptée et les SI qui implémentent cette stratégie. Parce que l'alignement entre la stratégie métier et le SI est au cœur de l'*Enterprise Architecture* (EA) [7] [1997-11], nous adoptons l'EA pour évaluer l'impact des technologies Smart Grids sur les gestionnaires des réseaux électriques. Il est en effet indispensable de concevoir une architecture cible pour avoir « une vision générale de comment une entreprise va mettre en œuvre sa stratégie » [2006-13].

L'EA participe à l'alignement des composants d'une entreprise en offrant une vision globale et transverse [1987-14]. Dans le contexte des Smart Grids par exemple, elle permet d'aligner efficacement les intérêts des acteurs impliqués dans leur implémentation tels que les experts métier, les conseillers stratégiques, les experts environnementaux ou encore les experts en normalisation.

L'exécution effective d'une stratégie est cependant confrontée à des barrières de communication au sein de l'entreprise [2010-15]. Le recours à l'EA est d'autant plus justifié qu'elle représente un outil pour la transmission des objectifs stratégiques à tous les niveaux hiérarchiques de l'organisation en question [2008-16].

Pour toutes ces raisons, nous souhaitons mettre à profit les principes d'EA pour évaluer l'impact de l'adoption des technologies Smart Grids sur les SI des gestionnaires de réseaux

7. L'acronyme EA (*Enterprise Architecture*) est souvent utilisé même dans la communauté francophone.

électriques, tout en garantissant la cohérence entre la stratégie adoptée et les SI qui les implémentent.

Les Smart Grids correspondent à des systèmes dynamiques et complexes [2010-17] étant donné le grand nombre de parties prenantes qui interagissent eux (tels que les producteurs d'énergie renouvelable ou les consommateurs actifs sur le réseau), tout en ayant des comportements autonomes et des objectifs différents. Les SI pilotant les Smart Grids correspondent par conséquence à des systèmes dynamiques aux comportements complexes. Borshchev et Filippov[2004-18] affirment que le seul moyen d'adresser cette complexité est de simuler ces systèmes. La simulation est une technique connue pour valider ou critiquer la conception d'un système dès les premières étapes de son cycle de développement. Les acteurs impliqués dans le déploiement des Smart Grids acquièrent, par la simulation, une connaissance approfondie et directe des modèles créés pour valider ou critiquer leur choix d'implémentation.

Néanmoins, les approches d'EA se focalisent le plus souvent sur des aspects statiques et structurels tels que les interconnexions entre les différentes applications métier [2008-19]. De plus, les modèles issus de ces approches sont ne sont pas exécutables et sont conçus en priorité pour la documentation et la communication entre parties prenantes [2013-20]. Notre problématique de recherche se résume de ce fait dans la question suivante :

Quels méthodes, modèles et outils adopter pour simuler une architecture d'entreprise afin de la valider ?

Chapitre 2

Architecture d'Entreprise

Sommaire

2.1 Notions fondamentales de l'Architecture d'Entreprise

Comme précédemment exposé dans notre problématique de recherche (cf. chapitre 1), nos travaux se sont d'abord portés sur la modélisation et la simulation du SI des Smart Grids. Cependant, le SI est sensé répondre de manière pertinente aux besoins de l'entreprise et implémenter efficacement sa stratégie. Avant toute simulation, nous devons donc de facto prendre en compte la stratégie de l'entreprise et ses objectifs métier et la mettre en cohérence avec le SI qui l'implémente.

L'alignement métier/SI est au cœur de l'EA, une discipline à part entière qui traite le SI en le corrélant au reste de l'entreprise. En effet, l'EA offre une vision globale des composants de l'entreprise tels que les processus métier, les parties prenantes, les informations, les fonctions, les applications informatiques, les infrastructures techniques.

Nous consacrons cette première partie de l'état de l'art à l'EA en commençant par définir les termes de SI et d'EA.

2.1.1 Terminologie

2.1.1.1 Système d'Information

Selon Robert Reix, le SI est « un ensemble organisé de ressources : matériel, logiciel, personnel, données, procédures permettant d'acquérir, de traiter, de stocker des informations (sous forme de donnée, textes, images, sons, etc.) dans et entre des organisations » [1995-21].

Nous adoptons cette définition car elle a l'avantage de ne pas réduire le SI d'une organisation à son système informatique. Le système informatique est constitué de l'ensemble du patrimoine matériel (hardware) et applicatif (software) de la dite organisation et a pour objectif d'automatiser le traitement de l'information. Nous adoptons l'acronyme IT (*Information Technologies*) pour le différencier du SI.

On suppose souvent que les SI sont totalement informatisés et c'est une des raisons qui mènent à confondre SI et Cependant, le SI comprend non seulement le système informatique mais aussi des ressources humaines telles que les partenaires ou le personnel et des ressources matérielles comme les procédures de gestion ou encore le savoir-faire métier.

2.1.1.2 Architecture d'Entreprise

Selon Zachman, une architecture d'entreprise est « un ensemble pertinent d'artefacts de conception ou de représentations descriptives pour décrire une entreprise de manière à ce cette entreprise soit créée en respectant certaines exigences et à ce qu'elle soit facilement maintenue tout au long de son cycle de vie » [1997-11].

Zachman voit ainsi dans l'architecture un gage de qualité et de maintenabilité. L'EA revient à appliquer à l'entreprise les principes de l'architecture telle qu'elle est pratiquée dans de nombreuses autres disciplines comme par exemple l'architecture du bâtiment. Construire une maison en procédant chambre par chambre sans plan d'architecture général peut en effet mener à un résultat peu probant. De même, le développement préalable d'une organisation sans architecture de référence risque de mener à une duplication de ses ressources et altère par conséquent son efficacité, sa cohérence interne rend fastidieuse toute entreprise de changement [1997-11] [2012-22].

Il est important de noter que le terme architecture d'entreprise peut prêter à confusion car il est à la fois utilisé pour désigner (1) l'activité de conception d'une architecture i.e. la description des éléments composant l'organisation en question et leurs relations mais aussi (2) l'ensemble des artefacts résultant de cette activité. Pour éviter toute confusion, nous désignons l'activité de conception par l'acronyme EA et le résultat de cette activité, c'est-à-dire les artefacts qui en sont issus, comme étant l'architecture de l'entreprise. L'EA est apparue en tant que discipline dans les années 1980 suite à l'informatisation accrue des entreprises mais son périmètre ne cesse d'évoluer depuis. La section suivante écrit les raisons et les grandes étapes de cette évolution.

2.1.2 Évolution de l'Architecture d'Entreprise

L'EA est ancrée dans l'architecture de systèmes informatiques [2008-16]. Mais le périmètre de l'EA ne cesse de s'étendre en comprenant d'abord l'IT et certains aspects métier de l'entreprise [2006-23], évoluant ensuite en adressant l'ensemble de l'entreprise en intégrant sa stratégie et ses processus décisionnels [2006-13], et allant même jusqu'à aborder l'environnement dans lequel elle évolue [2012-24].

L'origine de l'EA [1] remonte en effet aux travaux de Zachman, souvent considérés comme précurseurs. Il y propose un cadre d'architecture pour l'IT [1987-14] afin d'optimiser la gestion du patrimoine applicatif et de l'infrastructure technique de l'entreprise. À ses débuts, l'EA se focalise donc sur des artefacts purement IT (data, logiciels, équipements) pour rationaliser l'utilisation des ressources informatiques [2006-23], tout en répondant aux besoins métier de l'entreprise. L'EA est alors guidée par les pratiques de l'ingénierie logicielle.

Cependant, l'accroissement de la complexité de l'IT et son rôle de plus en plus prégnant dans le cœur de métier des entreprises [2005-26] ont fait de l'architecture IT une problématique inhérente à l'ensemble des composants de l'entreprise. Pour y faire face, l'EA a commencé à inclure quelques aspects métier tels les processus les acteurs impliqués [2006-23]. En intégrant ainsi des problématiques métier, l'EA ne relève plus de l'architecture IT mais de l'architecture du SI dans son ensemble.

Pour Scott Bernard, l'EA doit même aller plus loin en intégrant la stratégie de l'entreprise dans son périmètre pour offrir une vision globale de l'ensemble des ressources de l'entreprise [2012-22]. Le terme « entreprise » implique ainsi une vue à haut niveau de l'ensemble de l'organisation. Le terme « architecture » fait référence à la mise en place d'un cadre structuré et cohérent pour l'analyse, le planning, et le l'exploitation de toutes les ressources dont dispose l'entreprise pour atteindre ses objectifs.

Enfin, certains auteurs insistent sur le fait que l'entreprise évolue dans un environnement inconstant [2012-24]. L'EA doit par conséquence inclure les relations de l'entreprise avec son

1. D'après Scott Bernard [2012-22], le terme Architecture d'Entreprise a fait sa première apparition dans le livre de Steven Spewak intitulé « Enterprise Archietcture Planning : developing a blueprint for data, applications and technology » [1993-25].

environnement pour mesurer l'impact de ce dernier et faciliter les processus d'adaptation et d'innovation. Les différentes phases d'évolution de l'EA sont à l'origine de l'apparition de plusieurs écoles que nous détaillons dans la section suivante.

2.1.3 Écoles de pensée de l'Architecture d'Entreprise

Aucune définition de l'EA n'a été universellement adoptée [2012-27] [2005-26]. Il existe en effet une pléiade de définitions émanant aussi bien du milieu académique que du milieu industriel donnant ainsi lieu à plusieurs écoles de pensée. Ce manque de consensus est du à la nature intrinsèque de l'activité d'EA : celle-ci est régie par un ensemble de préceptes et de bonnes pratiques que l'architecte reste libre d'adapter au contexte de l'entreprise.

Il est toutefois possible d'identifier un thème commun à toutes les définitions proposées : l'architecture d'entreprise décrit les composants interdépendants d'une organisation et guide leurs évolutions [2012-24]. En revanche, le *périmètre* de cette description ainsi que les *préoccupations* adressées diffèrent d'une définition à l'autre.

S'agissant du *périmètre*, le terme « entreprise » peut couvrir uniquement l'IT ou s'étendre à tous ses composants humains, stratégiques, économiques et techniques. Les *préoccupations* sous-jacentes à l'EA peuvent quant à elles couvrir des objectifs allant de l'optimisation des investissements dans l'infrastructure technique à l'implémentation de la stratégie de l'entreprise en passant par l'alignement métier/IT.

Partant de ce constat, James Lapalme identifie trois écoles de pensée en EA [2012-24] : l'architecture de l'IT d'entreprise (*Enterprise IT Architecting*), l'architecture intégrative de l'entreprise (*Enterprise Integrating*) et l'architecture de l'entreprise dans son environnement (*Enterprise Ecological Adaptation*). Chaque école a son propre système de croyance (devise, préoccupations et objectifs, principes et postulats).

Il est important de noter que cette une catégorisation est épurée voire idéalisée, dans la mesure où le majorité des auteurs gravitent autour d'une école plutôt que de se conformer complètement à une seule école. *Enterprise IT Architecting* réduit le périmètre de l'architecture à l'IT de l'entreprise. *Enterprise Integration* adopte une approche intégrative en incluant toute l'entreprise dans son périmètre, de sa stratégie métier à la gestion de son patrimoine applicatif. *Enterprise Ecological Adaptation* inclue l'environnement dans lequel évolue l'entreprise pour en déterminer les impacts potentiels et mettre en place une stratégie d'adaptation adéquate. Le tableau 2.1 résume ces écoles de pensée en termes de devises, objectifs et principes.

En définissant sa taxonomie, Lapalme insiste sur le fait que ces écoles de pensée se sont formées par héritage : l'*Enterprise Ecological Adaptation* hérite de l'*Enterprise Integration*, qui elle-même hérite l'*Enterprise IT architecting*. Cependant, cet l'héritage implique une transcendance car il existe des différences fondamentales entre ces trois écoles de pensée. Par exemple, l'approche réductionniste de l'*Enterprise IT architecting* est fondamentalement opposée à l'approche holistique des deux autres écoles. Mais quelle qu'en soit l'école de

	Enterprise IT Architecting	Enterprise Integration	Enterprise Ecological Adaptation
Devise	L'architecture d'entreprise raccorde l'IT au métier de l'entreprise	L'architecture d'entreprise lie la stratégie et son exécution	L'architecture d'entreprise est un moyen d'innovation organisationnelle et une garantie de durabilité
Objectifs et préoccupations	Planifier l'IT et en optimiser les coûts	Implémenter efficacement la stratégie de l'entreprise	Adapter et innover
	Appuyer le métier	Assurer la cohésion de l'organisation	Assurer la cohésion de l'organisation
			Encourager la co-évolution entre l'entreprise et son environnement
Principes et postulats	Appliquer une approche réductionniste	Appliquer une approche holistique	Appliquer une approche holistique
	Ne pas remettre en question la stratégie et les objectifs métier	Ne pas remettre en question la stratégie et les objectifs métier	Créer la stratégie de l'entreprise est une priorité
	Concevoir les composants de l'organisation de manière indépendante	Concevoir les différents aspects de l'entreprise de manière intégrative	Concevoir les différents aspects de l'entreprise de manière intégrative
	Ne pas se préoccuper des aspects non IT	Tenir compte de l'environnement comme source de changement	L'environnement peut être transformé

TABLE 2.1 – Écoles de pensée de l'Architecture d'Entrerpise selon [2012-24]

pensée, l'EA présente des avantages certains pour l'entreprise que nous rapportons dans la section suivante.

2.1.4 Avantages de l'Architecture d'Entreprise

L'EA organise et structure les informations à l'échelle de l'entreprise tout en fournissant les détails appropriés à chacune des parties prenantes et en définissant le schéma directeur nécessaire à la construction de SI évolutifs et pertinents pour le métier.

Pour Zachman, manier l'architecture de l'entreprise avec agilité et l'adapter rapidement à un contexte économique et technologique en constante évolution est un facteur de survie déterminant pour les entreprises du 21ème siècle [1997-11]. Afin de souligner l'importance de l'EA, Ross [28] donne l'exemple d'une problématique d'architecture que l'entreprise américaine *Jonhson&Johnson* a rencontré en 1995. Le succès international de cette entreprise repose surtout sur l'autonomie de ses 170 filiales. Ses managers sont parfaitement satisfaits des processus métier mis en place mais ce n'est pas le cas de tous ses clients. Les très grands clients reçoivent en effet de nombreux bons de commande et plusieurs factures des différentes filiales de *Johnson&Johnson* et doivent donc les traiter séparément. Un jour, ces clients exigent de ne recevoir qu'une seule facture. Or *Johnson&Johnson* n'en est pas

capable. Ni ses processus métier, ni sa structure organisationnelle et encore moins son IT ne lui permet d'accéder à la demande de ses clients. Ceci est un cas d'école typique que permet d'adresser l'EA.

L'EA présente plusieurs avantages liés autant aux aspects purement IT qu'aux aspects métier. D'abord, en capturant l'essence du métier, de l'IT et de son évolution [2013-29], l'EA permet d'abstraire la complexité d'un système telle qu'une entreprise. Ensuite, en tant que référentiel et support de communication, l'EA facilite la coordination entre les projets IT d'une entreprise, la supervision des ressources techniques ainsi que la suppression des redondances applicatives [2007-30]. Enfin, l'EA est un moyen efficace pour représenter les composants d'une entreprise dans son état courant et désiré. Comme tableau de bord, l'EA facilite l'accès à l'information nécessaire à l'optimisation des processus métier et à l'alignement effectif entre l'IT et la stratégie adoptée.

L'EA peut faciliter le processus de prise de décision si elle aboutit à une vision à la fois globale et adaptée aux décideurs. Dans la section suivante, nous présentons quelques approches et cadres d'EA prenant en compte le processus de prise de décision et les acteurs qu'il implique.

2.2 Conception d'une architecture d'entreprise

2.2.1 Approches orientées points de vue

Quelle qu'en soit l'école, l'EA reste une tâche complexe [2004-31] car elle implique un grand nombre de parties prenantes. Chaque partie prenante a des préoccupations et des systèmes de notation propres et relatives à son domaine d'expertise et aux responsabilités lui incombant.

L'EA doit capturer une grande variété de composants difficiles à représenter dans un seul et unique modèle. En procédant par analogie avec l'architecture d'une vile, la multitudes d'acteurs concernés peuvent difficilement lire un plan où l'on représente à la fois les rues et les bâtiments ainsi que les réseaux de transports, d'électricité, de gaz et d'eau.

Il en va de même pour l'architecture d'entreprise. La nature mutli-facettes inhérente à l'entreprise rend inappropriée toute approche monolithique [1999-32]. En effet, un analyste métier est concerné par les processus et les fonctions métier tandis qu'un administrateur de bases de données est concerné par les données manipulées. Pour cette raison, la plupart des cadres d'EA adoptent une approche par points de vue.

Les approches orientées points de vue sont d'abord utilisées pour la spécification des besoins en ingénierie logicielle [1979-33]. Les chercheurs s'intéressent alors aux systèmes à « perspectives multiples » [1992-34] [1996-35] [1994-36] [1993-37]. Ces travaux précurseurs contribuent à l'émergence de plusieurs normes pour les systèmes logiciels, proposant des cadres d'architecture orientés points de vue. C'est le cas de la norme IEEE-1471 [2000-38],

du standard *Reference Model of Open Distributed Processing* (RM-ODP) [1995-39] ou encore du standard MDA (*Model Driven Architecture*) [2003-40].

Dans la norme IEEE1471 [2000-38], une vue correspond à une représentation du système selon une certaine perspective à laquelle est associé un ensemble de préoccupations. Les vues permettent ainsi de séparer les préoccupations des différentes parties prenantes. Un point de vue correspond, quant à lui, à un *template* pour la création de vues. Il formalise les objectifs des parties prenantes concernées par la vue ainsi que les techniques qui permettent de la créer et de l'analyser. Pour décrire un point de vue, la norme IEEE1471 [2000-38] exige de spécifier les attributs suivants :

- le nom du point de vue ;
- la partie prenante ciblée ;
- les préoccupations de celle-ci ;
- le langage, les techniques de modélisation ou encore les méthodes d'analyse à utiliser pour la création de la vue.

2.2.2 Cadres d'Architecture d'Entreprise

Les approches orientées points de vue sont largement utilisées en ingénierie logicielle comme moyen d'adresser la complexité des architectures [2004-31]. Comme l'EA trouve ses racines dans l'architecture IT [2008-41], de nombreux cadres d'EA recourent aux points de vue en transférant les concepts développés pour architecture IT à l'EA. Parmi les cadres d'EA les plus utilisés nous citons le cadre Zachman, *The Open Group Architectural Framework* (TOGAF). Nous citons aussi d'autres cadres d'architecture spécifiques à leurs domaines comme RM-ODP et *Smart Grid Reference Architecture* (SGAM).

2.2.2.1 Le cadre Zachman

Zachman [1987-14] propose de structurer et d'organiser les différentes représentations intervenant dans la description d'une entreprise en les classant selon une matrice à deux dimensions. La dimension verticale correspond aux points de vue et la dimension horizontale aux abstractions. Comme l'illustre le tableau 2.2, chaque cellule de la matrice correspond à l'intersection entre une partie prenante impliquée dans le processus de conception de l'architecture et une abstraction présentée sous forme d'une question. Chaque représentation est alors adaptée aux acteurs impliqués.

Encore largement utilisé, le cadre Zachman est le premier à adresser l'entreprise dans son ensemble. Il est de plus facile à comprendre et ne dépend ni d'une méthode ni d'un outil en particulier. Sa mise en pratique reste cependant fastidieuse à cause du grand nombre de cellules à modéliser. En outre, la mise en cohérence entre les différents artefacts n'est pas évidente car les relations entre les cellules ne sont pas explicitement spécifiées. Selon

Perspectives (lignes) \ Abstractions (colonnes)	**Données** *Quoi*	**Fonctions** *Comment*	**Personnel** *Qui*	**Temps** *Quand*	**Motivation** *Pourquoi*
Portee *Planificateur*					
Modèle métier *Propriétaire*					
Modèle système *Concepteur*					
Modèle Technologique *Réalisateur*					
Modèle détaillé *Sous-traitant*					

TABLE 2.2 – Cadre Zachman [1987-14]

Lankhorst [2013-29], le cadre Zachman offre un schéma de classification bien structuré mais ne spécifie aucune méthode pour mener les différentes activités d'architecture.

2.2.2.2 TOGAF

TOGAF, le plus connu et le plusutilisé des cadres d'architecture [2008-41], est comme sonnom l'indique un standard de l'*Open Group*. TOGAF est doté de quatre points de vue — métier, information, applicatif et technique — et d'une méthode de conception — *The Architecture Development Method* (ADM).

La méthode ADM correspond à un processus de conception cyclique. Elle préconise de piloter l'EA par la gestion des exigences. Les exigences sont dérivées de la stratégie et des objectifs métier de l'entreprise. Elles sont de ce fait considérées comme le centre névralgique des activités d'architecture et font le lien entre les différentes étapes :de la conception de l'architecture à sa mise en œuvre comme l'illustre la figure 2.1.

La méthode ADM commence par une phase préliminaire qui consiste à initialiser ou encore contextualiser l'EA. Pendant cette phase, la disposition de l'entreprise à engager une démarche d'EA est évaluée et les grands principes d'EA sont définis en accord avec le métier. Le cycle ADM se poursuit avec les huit phases, notées de A à G, relatives à la création des vues métier, applicative, information et technique. Il s'agit ensuite de planifier le déploiement de l'architecture avant de l'implémenter. La dernière phase (notée H) consiste à gérer les changements qui peuvent survenir suite aux évolutions métier ou technologiques pour assurer une mise à jour continue de l'architecture.

Générique, TOGAF peut être appliqué à des entreprises diverses indifféremment de leurs secteurs d'activité. Flexible et largement paramétrable, ce cadre d'architecturepréconise un ensemble de bonnes pratiques à adapter selon les cas. Il est par exemple possible de recourir au schéma de classification de Zachman dans le cadre d'une démarche ADM.

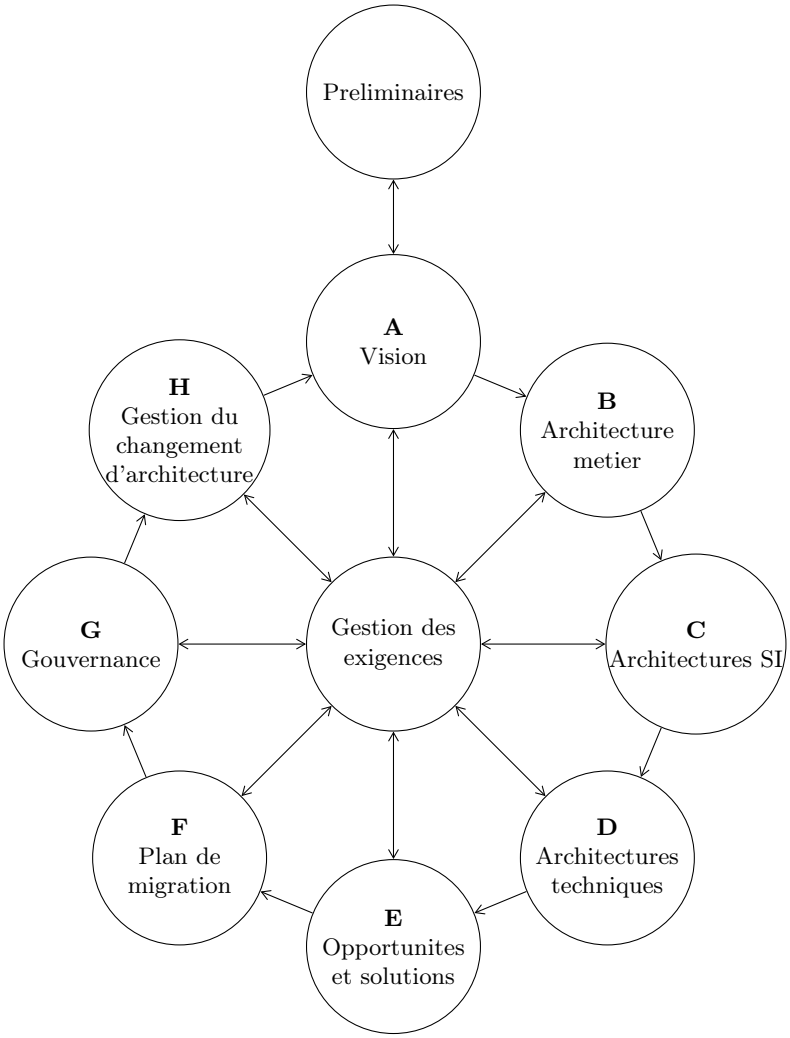

FIGURE 2.1 – TOGAF ADM [2009-42]

TOGAF comme Zachman laisse aux praticiens la liberté de choisir les langages de modélisation et le niveau de détail requis pour la conception des vues de l'architecture.

2.2.2.3 RM-ODP

RM-ODP est un standard ISO/ITU [2] qui définit un cadre d'architecture pour la spécification des systèmes distribués ouverts. Il se réfère au paradigme de l'orienté objet et identifie cinq points de vue :

– le point de vue Entreprise, qui décrit les activités métier du système ;

2. International Organization for Standardization/International Telecommunication Union

- le point de vue Information, qui définit l'information traitée par le système et la façon dont elle est traitée par les différents composants ;
- le point de vue Traitement, qui spécifie les traitements effectués par les différents composants en termes de fonctions et en faisant abstraction de toute plate-forme technique ;
- le point de vue Ingénierie, qui décrit les mécanismes logiciels permettant la distribution des composants et leur exécution sur les plates-formes d'exécution ;
- le point de vue Technologie, qui définit les technologies matérielles et logicielles utilisées pour l'infrastructure d'exécution, leur configuration.

RM-ODP préconise de découpler les préoccupations métier des contraintes liées à une plate-forme technique donnée. En effet, les cinq points de vue sont séparés mais corrélés. Chaque objet d'un point de vue donné correspond à un autre objet dans un autre point de vue.

RM-ODP offre un cadre de référence mais ne préconise pas de méthode d'architecture. Cette correspondance entre les différents points de vue n'est donc pas nécessairement respectée en spécifiant, par exemple, chaque point de vue de façon isolée. Pour y remédier, EDF R&D a proposé DASIBAO, une Démarche d'Architecture des Systèmes d'Information Basée sur RM-ODP. DASIBAO préconise le langage UML dans la spécification des cinq points de vues qu'il aborde dans l'ordre itératif suivant :

$$\text{Entreprise} \rightarrow \text{Information} \rightarrow \text{Traitement} \rightarrow \text{Ingénierie} \rightarrow \text{Technique}.$$

C'est donc par construction que DASIBAO se propose d'assurer la cohérence des différents points de vue.

2.2.2.4 SGAM

Le SGAM [2012-43] adresse l'architecture du Smart Grid en englobant les trois domaines : SI, réseau électrique et réseau de télécommunication. (Le reste à récupérer d'une note H rédigée pour EDF.)

2.2.3 Points de vue retenus

Les cadres d'EA n'utilisent pas tous les mêmes points de vue : leurs nombres, leurs noms ainsi que les préoccupations qu'ils adressent varient. Néanmoins, ces cadres recourent souvent, implicitement ou explicitement, aux points de vue ci-après.

Point de vue métier : ce point de vue reflète la vision métier de l'entreprise. On y retrouve ses objectifs ainsi que ses processus. Ces derniers sont représentés selon la structure organisationnelle de l'entreprise en termes d'acteurs internes et externes ;

Point de vue fonctionnel : ce point de vue organise l'entreprise en blocs fonctionnels implémentant les processus de la vue métier. Cette structuration implique souvent une grande cohérence au sein d'un même bloc et une forte décorrélation entre blocs dans un souci de modularité et d'évolutivité. À l'échelle d'une entreprise, cette structuration devient vite complexe à cause du caractère étendu, transverse et interdépendant des processus métier impactés ;

Point de vue applicatif : ce point de vue structure l'entreprise en blocs applicatifs. Chaque bloc implémente un ou plusieurs blocs fonctionnels. Il est aussi important de spécifier les échanges entre blocs applicatifs. Comme pour la vue fonctionnelle, la vue applicative des très grandes entreprises souffre souvent du syndrome du plat à spaghetti : les nombreuses applications fortement couplées deviennent difficiles et chères à maintenir ;

Point de vue technique : ce point de vue correspond à l'infrastructure technique nécessaire à l'exécution des blocs applicatifs. Le point de vue technique spécifie ainsi les machines physiques et liens de communication utiles au déploiement des applications informatiques.

En outre, ces cadres d'architecture sont orientés composants car ils utilisent des concepts tels que les macros processus, les blocs fonctionnels ou encore les blocs applicatifs. Les informations sont modélisées soit implicitement et de manière diffuse à l'intérieur des vues (TOGAF), soit séparément dans une vue dédiée et décorrélée des autres vues (RM-ODP, SGAM). Une troisième méthode consiste à les modéliser sous forme d'aspect pour chaque vue (Zachman, Archimate).

De plus, ces cadres d'architecture organisent hiérarchiquement les différentes vues en appliquant *« IT follows business »* comme principe : commencer par la vue métier et la dériver progressivement jusqu'à l'infrastructure technique en passant par les fonctions et les applications [2006-23].

Les cadres d'architecture sont certes indispensables pour aboutir à une représentation pertinente des composants de l'entreprise mais ne suffisent pas à appréhender la complexité du système entreprise. Pour cela, il est primordiale de disposer d'outils et de méthodes appropriés à l'analyse des modèles d'entreprise obtenus. La section suivante offre un tour d'horizon de l'activité d'analyse en EA.

2.3 Analyse en Architecture d'Entreprise

La valeur ajoutée de l'EA réside dans sa capacité à adresser le changement en offrant une vue holistique de l'entreprise. En effet, l'efficacité de l'entreprise dépend de l'orchestration effective de ses différents composants et entités plutôt que d'optimisations locales et isolées [1992-44].

La documentation et la description ne suffisent cependant pas à assurer une architecture

à la fois cohérente et pertinente pour le métier. Les techniques d'analyse de modèles sont indispensables à l'optimisation globale et effective d'une architecture [2013-29]. Les techniques d'analyse de modèles jouent donc un rôle crucial dans tout processus de changement affectant l'entreprise en éclairant efficacement la prise de décision. En effet, l'analyse de l'architecture d'entreprise ne doit pas se résumer pas à la revue mentale ou manuelle d'une vue d'ensemble étant données la taille et la complexité des architectures impliquées.

2.3.1 Au-delà des modèles « contemplatifs »

Les entreprises recourent aux cadres d'EA pour les guider dans la création et la maintenance de leurs architectures et pour avoir ainsi une vue globale et cohérente de leurs stratégies, leurs processus métier et leurs IT

Les artefacts issus d'une démarche d'EA se résument souvent à un ensemble de documents utilisés comme supports de communication et comme schéma directeur au sein de l'organisation [2013-45] [2014-46]. Ces modèles fournissent certes un vocabulaire commun aux différentes parties prenantes mais ne sont ni manipulables ni interprétables par une machine. Ce sont des modèles purement « contemplatifs ». Cette terminologie est introduite par Bézivin [2001-47] en qualifiant les modèles de spécification utilisés pendant les premières phases de conception en génie logiciel.

Aussi, l'EA s'intéresse-elle d'avantage aux aspects structuraux de l'entreprise. Pour cette raison, les modèles utilisés, bien qu'offrant l'abstraction nécessaire pour adresser la complexité de l'entreprise, sont le plus souvent statiques. Ce genre de modèles ne permet pas d'appréhender les comportements de l'entreprise et sont donc insuffisants pour éclairer efficacement les prises de décision au niveau stratégique.

L'EA, à travers les différents cadres d'architecture proposés lors de ces trente dernières années, a certes contribué à traiter de manière intégrée les différents aspects d'une entreprise tels que les processus, le personnel, les services et l'IT. Cependant, la gestion des artefacts issus de l'EA reste un défi malgré l'existence d'outils sur étagère. En effet, les architectes d'entreprise expérimentés ainsi que les autres parties prenantes impliquées dans les activités d'architecture sont supposés se fier à leur bon jugement pour créer une architecture adéquate.

L'architecture créée est donc correcte par définition et dépend essentiellement des capacités de l'architecte et de son expertise. Certains travaux proposent de rendre l'EA moins dépendante de l'expertise de la personne en charge en traduisant les représentations d'architecture en ontologies [2013-48]. Les ontologies étant exécutables, permettent en effet de bénéficier des capacités d'analyse des raisonneurs disponibles .

Certaines méthodes d'EA sont accompagnées de langages de modélisation comme le langage Archimate pour TOGAF. Dans leur quête de généricité, ces langages deviennent rapidement très larges et difficiles à manipuler. Les modèles produits sont d'autant plus difficiles à gérer qu'ils ne sont pas manipulables par une machine. L'activité d'analyse en EA, bien que cruciale, est donc compromise par toutes ces limitations. Et bien que des modèles et

des techniques destinés à l'analyse des architectures d'entreprise existent, le recours aux modèles exécutables reste encore marginal dans le domaine de l'EA [2013-20].

Parmi ces travaux nous citons le langage LEAP [2011-49]. Léger, générique et exécutable, LEAP est destiné à l'analyse des modèles issus de l'EA pour valider l'alignement des modèles métier et IT

Les deux sections suivantes présentent deux schémas de classification des approches d'analyse en EA. Le premier est proposé par Lankhorst [2013-29] et et le deuxième par Buckl et al. [2009-50]. Ces schémas de classification nous permettent de (1) comparer entre elles les différentes approches d'analyse recensées dans cet état de l'art et (2) positionner nos travaux.

2.3.2 Classification des approches d'analyse selon Lankhorst

Lankhorst [2013-29] utilise deux dimensions pour classifier les différentes approches d'analyse d'architecture d'entreprise : le type d'analyse et la technique employée. Ceci donne lieu à quatre catégories comme l'illustre la figure 2.2. La première dimension fait la distinction entre deux types d'analyse.

L'analyse fonctionnelle concerne les aspects fonctionnels de l'architecture. Elle permet par exemple de valider la structure ou de comprendre le comportement d'une architecture.

L'analyse quantitative concerne les aspects non fonctionnels de l'architecture comme la performance ou le coût.

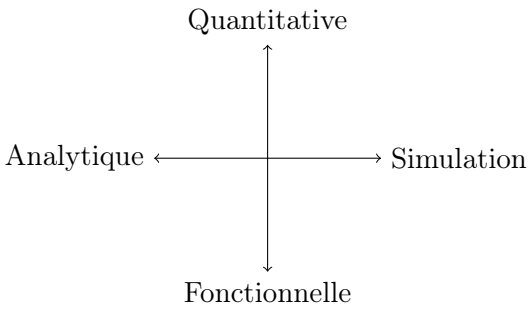

FIGURE 2.2 – Classification des approches d'analyse selon Lankhorst [2013-29]

La deuxième dimension identifie deux techniques pouvant être employées pour l'analyse fonctionnelle ou quantitative :

La technique de simulation des modèles revient à les exécuter. La simulation dans le cas de l'analyse fonctionnelle est utilisée pour mieux appréhender les aspects dynamiques d'une architecture. La simulation quantitative permet de mesurer des paramètres

quantitatifs tel que le temps d'exécution d'un processus métier par exemple à travers plusieurs itérations de la simulation.

La technique analytique est plus formelle que la simulation. Ici le terme analytique signifie plutôt « mathématique ». Cette technique est plus efficace que la simulation quantitative pour fournir des indicateurs de performance.

2.3.3 Classification des approches d'analyse selon Buckl

Le schéma de classification de Lankhorst [2013-29] offre un premier aperçu des différentes approches d'analyse d'architecture cependant il ne révèle pas toutes les variantes et les subtilités d'une démarche d'analyse en EA. Nous considérons donc les travaux de Buckl et al. [2009-50] qui définissent un système de classification plus détaillé. Ce système classification peut même être considéré comme un cadre d'analyse d'architectures.

Buckl et al. [2009-50] font intervenir cinq dimensions pour catégoriser les approches d'analyse d'architecture d'entreprise. Ces dimensions sont illustrées par la figure 2.3.

2.3.3.1 Sujet de l'analyse

L'analyse peut concerner trois aspects différents de l'architecture : sa structure, son comportement dynamique ou son comportement statistique. D'abord, l'analyse de la structure est nécessaire pour appréhender la complexité structurelle des entreprises. Celle-ci est due à la densité des interconnexions entres ses composants. Ensuite, la complexité de la structure de l'entreprise induit une complexité au niveau de son comportement d'où la nécessité d'analyser l'aspect comportemental de l'architecture pour évaluer l'impact d'une anomalie sur le déroulement d'un processus par exemple. Enfin, l'analyse et l'agrégation des mesures statistiques provenant du comportement offrent une meilleure compréhension de l'architecture.

2.3.3.2 Référence temporelle

L'analyse peut se porter sur l'architecture d'une entreprise dans son état courant ou telle qu'elle est planifiée. Une analyse *ex post* concerne les modèles d'une architecture déjà mise en place alors qu'une analyse *ex ante* se réfère à différents scénarios élaborés pour une future implémentation.

2.3.3.3 Technique d'analyse

Les techniques d'analyse employées peuvent s'appuyer sur des experts, sur des règles ou sur des indicateurs. Les analyses orientées experts sont les moins formelles et dépendent du niveau d'expertise de l'expert impliqué. Cette technique d'analyse est certes chronophage

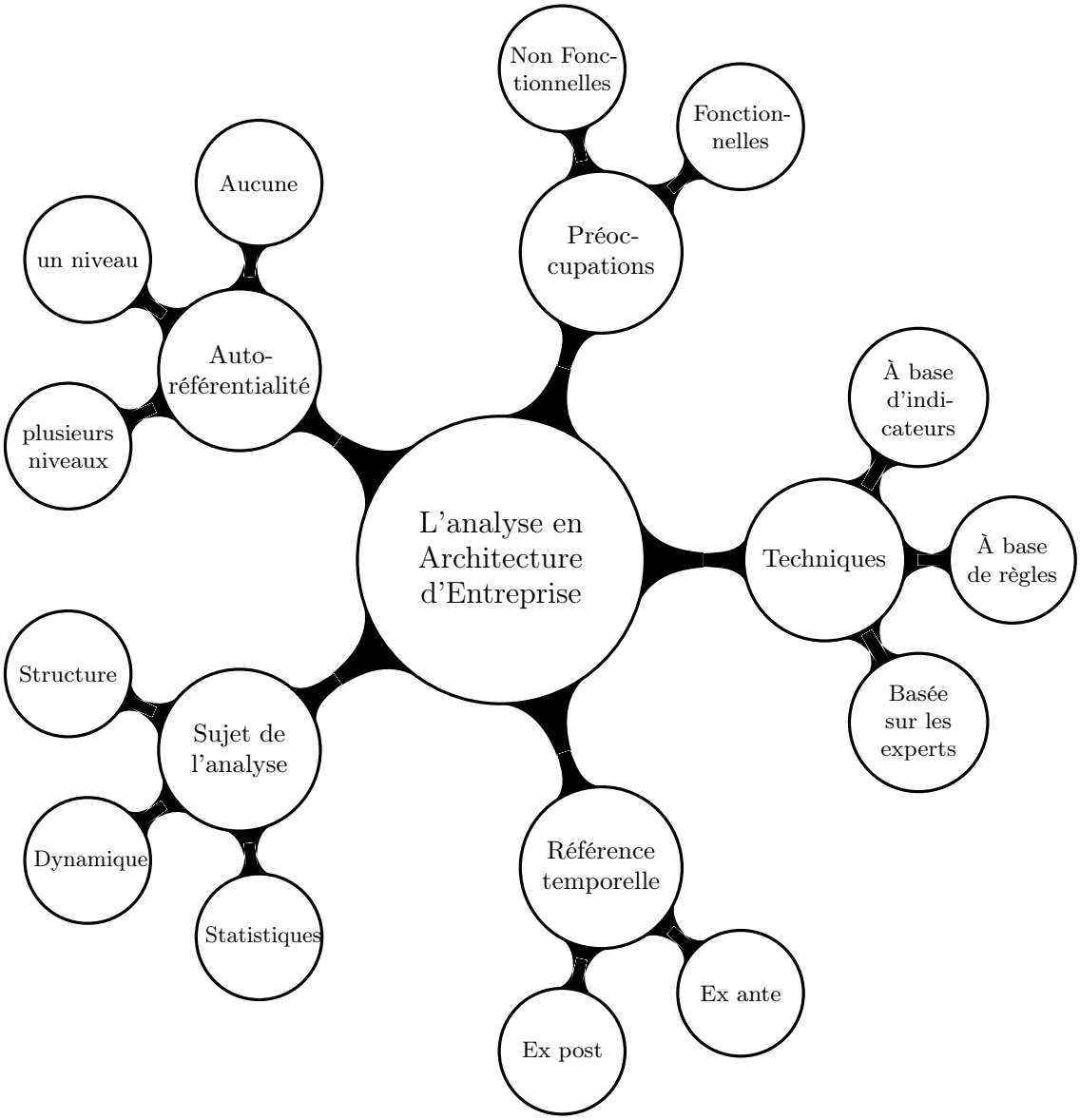

FIGURE 2.3 – Schéma de classification des approches d'analyse selon Buckl et al. [2009-50]

mais c'est celle qui offre le plus de flexibilité. Les résultats d'une telle analyse prennent la forme de conseils concrets ou d'idées et de directives générales concernant l'architecture en question.

Les techniques d'analyse s'appuyant sur des règles sont plus formelles que celles s'appuyant sur des experts et peuvent être plus facilement automatisées. Ces règles décrivent les modèles de conception que l'architecture doit respecter ou éviter.

Les techniques d'analyse s'appuyant sur des indicateurs sont encore plus formelles que les deux précédentes et servent à évaluer en les quantifiant certaines propriétés de l'architecture. Les résultats d'une telle analyse doivent cependant être interprétés avec prudence car ils dépendent d'hypothèses souvent amenées à évoluer.

2.3.3.4 Préoccupations de l'analyse

Buckl et al. [2009-50] font la différence entre les analyses fonctionnelles et les analyses non fonctionnelles à la manière de Lankhorst [2013-29]. L'analyse fonctionnelle évalue si l'architecture remplie les fonctions métier de l'entreprise telles que la production ou la vente. L'analyse non fonctionnelle évalue des aspects comme le temps d'exécution d'un processus ou le coût de l'implémentation et de maintenance d'une architecture. Contrairement à Lankhorst [2013-29], Buckl et al. [2009-50] emploient le terme *non fonctionnelle* plutôt que *quantitative* en arguant que certains aspects non fonctionnels telle que la sécurité peuvent être analysés de manière non quantitative.

2.3.3.5 Autoréférentialité

Les personnes en charge de l'EA peuvent faire partie l'architecture créée car appartiennent aussi à l'entreprise. Qui plus est, l'activité de décrire et de planifier l'architecture de l'entreprise peut elle-même être décrite et planifiée et fait par conséquent partie de l'activité d'EA. Selon Buckl et al. [1974-51] l'entreprise est en effet un système vivant capable de faire sa propre autopsie [1974-51].

L'analyse peut considérer un seul niveau d'autoréférentialité en intégrant les activités de management d'architecture. Une analyse à plusieurs niveaux d'autoréférentialité incorporent des activités de meta-management d'architecture comme par exemple la gouvernance des activités de management d'architecture. L'autoréférentialité augmente la complexité de l'analyse. Peu de travaux considèrent des niveaux d'autoréférentialité multiples [2014-52]. Nous citons parmi celles-ci les travaux de [2014-53] qui traitent la gestion et la gouvernance d'architecture en définissant des stratégies d'évolution pour les architectures actuelles vers les architectures cible dans un framework dédié.

2.3.4 Analyse de la structure

2.3.4.1 Finalités

Maintenir une cohérence entre l'infrastructure informatique de l'entreprise avec ses processus métier est au cœur des activités d'EA[2013-29]. L'objectif de cet alignement est d'améliorer l'efficacité de l'entreprise et de maximiser ses bénéfices. L'alignement métier/IT reste une problématique cruciale pour les entreprises qui s'appuient sur les technologies de l'information dans la réalisation de leurs objectifs métier [2005-54]. Zachman affirme même

que seules les entreprises capables d'aligner rapidement leurs SI à leurs stratégies métier sont en mesure de survivre dans un environnement hautement concurrentiel [1997-11].

Pour Lankhorst [2013-29], l'EA a pour vocation d'offrir une vue générale et homogène du métier de l'entreprise, de son patrimoine applicatif, de son infrastructure technique et de son évolution pour en faciliter l'analyse via un ensemble cohérents de principes, méthodes et modèles. De plus, l'EA explicite et documente les relations entre les processus métier et l'IT de l'entreprise [2005-54].

En offrant une vision globale de l'entreprise et en documentant les relations entre ses différents composants, l'EA est un outil incontournable pour les architectes dans leur quête d'alignement métier/IT. Cependant, les méthodes et techniques actuelles offerts par l'EA ne suffisent pas à atteindre cet objectif [2013-55] car :

- les processus métier, les technologiques, les structures organisationnelles ainsi que l'environnement sont en constante évolution. Les changements sont si rapides et nombreux qu'un réel alignement relève du vœux pieux [2013-29]. L'alignement métier/IT correspond plutôt un idéal kantien vers lequel l'entreprise doit tendre à défaut de le réaliser complètement ;
- l'architecture en tant que discipline tient d'avantage de l'art que de la science. En effet, l'architecte d'entreprise analyse souvent de la documentation (tels que des tableurs Excel ou des illustrations Power Point) même si quelques logiciels permettent désormais de visualiser ces modèles. Cette tâche devient rapidement ardue dès qu'il s'agit de grandes entreprises dont l'important patrimoine applicatif s'apparente à un plat de spaghetti.

2.3.4.2 Approches d'analyse de la structures existantes et leurs limites

Les modèles exécutables peuvent assister les architectes d'entreprise à surmonter les obstacles cités ci-dessus. Les modèles exécutables augmentent l'agilité de l'architecture en détectant automatiquement les incohérences dès les premières phases de conception.

Les approches de modélisation en EA recourent souvent à des langages semi-formels qui ne permettent pas de vérifier dynamiquement et automatiquement certaines propriétés requises pour l'architecture comme la cohérence entres vues. Partant de ce constat, [2013-48] proposent de recourir aux ontologies tirant ainsi profit des raisonneurs disponibles pour analyser la structure des architecture d'entreprise. Ils arrivent à analyser l'impact du changement ou encore les relations et dépendances entre les opérations métier et les applications informatiques de l'entreprise. Cependant, les architectes d'entreprise sont peu familiarisés avec les ontologies. Les modèles d'architecture sont donc traduits en ontologies, risquant de faire apparaître un gap sémantique entre les modèles utilisés pour la représentation de l'architecture et les ontologies utilisées pour mener l'analyse.

La diversité des acteurs impliqués dans l'EA comme l'architecte d'entreprise, le manager ou encore l'ingénieur IT engendre une hétérogénéité au niveau des modèles utilisés. [2013-56]

proposent de créer un mapping entre des modèles hétérogènes (tels que des tableurs Excel, de la documentation ou des bases de données). Pour cde faire, ils recourent à des modèles manipulables par machine et mettent à profit les techniques de tissages de modèles issus de l'IDM. Ces travaux adressent l'hétérogénéité des modèles et offrent une vue intégrée de l'architecture de l'entreprise en l'adaptant aux acteurs concernés. Mais elle ne permet pas de mener des analyses concernant l'impact du changement par exemple.

2.3.5 Analyse du comportement

2.3.5.1 Finalités

Pour Shannon [1975-57], la simulation est « un processus consistant à modéliser un système réel et à mener des expérimentations sur le modèle obtenu dans le but de comprendre le comportement du système et/ou d'évaluer différentes stratégies concernant son fonctionnement ». Quel qu'en soit le domaine d'application, la simulation est un moyen d'apprécier les choix des concepteurs sur le comportement du système modélisé. Elle peut se traduire par par l'animation d'un modèle (représentant notre perception du système, qu'il soit existant ou à construire) et l'étude du comportement de ce modèle en fonction des variables en entrée.

La simulation des architectures d'entreprise permet de modifier localement des stratégies et d'observer l'impact de ces modification sur le comportement globale du système [2008-19]. La simulation des architectures d'entreprise est d'autant plus cruciale dans le contexte des Smart Grids. Ces derniers sont en constante et rapide transformation : évolution des cadres législatifs, apparition de nouveaux partenaires, hétérogénéité des interactions avec les clients finaux (les compteurs intelligents, Internet, les téléphones ou encore les tablettes).

Ainsi, le recours à la simulation dès les premières phases du cycle de vie des architectures d'entreprise dans le contexte des Smart Grids augmente leur évolutivité en apportant une aide supplémentaire à leur validation. La simulation des modèles facilite leurexploration par les experts métier et lève les ambiguïtés engendrées par les modèles purement contemplatifs. Elle permet en outre un prototypage rapide et une analyse itérative des modèles par les parties prenantes tels que les architectes d'entreprise, les expert métier, les analystes ou les architectes IT.

2.3.5.2 Approches de simulation existantes et leurs limites

[2015-58] recensent quatorze approches d'analyse d'architectures d'entreprise et les classent selon les quatre dimension du schéma de classification de Lankhorst [2013-29] (précédemment illustré par la figure 2.2). Seulement quatre approches parmi les quatorze recourent à la simulation comme outil d'analyse d'une architecture d'entreprise.

Le nombre limité d'approches utilisant la simulation pour l'analyse d'architecture d'entreprise est du au fait que l'EA est initialement conçue comme une description statique des composants essentiels de l'entreprise et de leurs interconnexions [2013-59]. Les cadres d'EA standards ne prennent pas en compte les informations nécessaires pour analyser les aspects comportementaux d'une entreprise et mettent d'abord l'accent sur ses aspects structuraux tels que les liens entre les processus et les applications métier. Il en résulte que les outils d'EA n'adressent souvent que les aspects statiques de l'entreprise car ils se basent sur les mêmes cadres d'architecture.

Buckle et al. [2009-50] classifient les approches d'analyse d'architecture existantes selon leur propre système de classification illustré par la figure 2.3. Nous tirons le même constat que pointe l'état de l'art de l'analyse en EA de [2015-58].

Dans le présent état de l'art, nous nous intéressons donc aux travaux qui analysent la dimension dynamique d'une architecture d'entreprise. Parmi ces approches, celles développées par [2011-60], [2011-61] et [2015-58] soulignent l'importance de simuler le comportement d'une architecture d'entreprise.

Glazner [2011-60] propose une approche de simulation hybride pour évaluer le comportement d'une entrepris en combinant une simulation à événements discrets et une simulation multi-agents. Il met à profit les capacité d'abstraction de l'EA pour adresser la complexité d'un système telle que l'entreprise et s'en sert comme socle pour structurer les modèles de simulation. Les techniques et les langages utilisés pour la simulation sont décorrélés des langages de représentation utilisés par les architectes d'entreprise entrainant ainsi un gap sémantique entre l'aspect statique de l'entreprise (les composants de l'entreprise et leurs relations) et son aspect dynamique (le comportement des composants).

Ludwig et al. [2011-61] simulent une architecture d'entreprise en fonction de la configuration organisationnelle d'une entreprise. Ils développent ainsi un langage exécutable pour décrire les relations entre des concepts de l'ordre organisationnels comme l'entité, le rôle qu'elle joue, les services qu'elle procure et les processus dans lesquels elle intervient. La méthode et l'outil développés permettent de reconfigurer les modèles organisationnels lors de l'exécution. Ces travaux n'adressent cependant que la vue métier et une partie de la vue fonctionnelle d'une architecture d'entreprise.

Manzur et al. [2015-58] proposent une plateforme de simulation qui s'appuie sur deux métamodèles différents : un premier métamodèle, correspondant au langage Archimate, pour spécifier les composants structurels de l'architecture et un deuxième métamodèle complémentaire pour spécifier les comportements des composants structurels. Les modèles sont ensuite simulés afin d'observer le comportement de l'architecture et de l'évaluer selon les indicateurs établis. Cependant cette approche évalue uniquement les aspects non fonctionnels d'une architecture d'entreprise.

Chapitre 3

Ingénierie Dirigée par les Modèles

Sommaire

3.1 Genèse et objectifs

L'Ingénierie Dirigée par les Modèles (IDM) est née du constat que le paradigme du « tout est objet », prôné dans les années 1980, a atteint ses limites avec ce début de siècle [2004-62]. En effet, face à la croissance de la complexité des systèmes logiciels, au coût de la main d'œuvre

et de maintenance, une approche centrée sur le code, jugé alors seul représentant fiable du système, suscitait de moins en moins l'adhésion des industriels et du milieu académique.

Partant de ce constat, l'*Object Management Group* (OMG) a proposé en novembre 2000, l'approche *Model Driven Architecture* (MDA) qui s'inscrit dans le cadre plus général de l'IDM et se réalise autour d'un certain nombre de standards tels qu'UML, *Meta-Object Facility* (MOF), XML, QVT, etc. Le monde de la recherche s'y est aussitôt intéressé pour dégager les principes fondamentaux de l'IDM [2001-63][2002-64] [2002-65] et déjouer le piège des définitions parfois trop floues qui qui mènent à confondre les concepts du paradigme objet et ceux de l'IDM [2004-66]. Par ailleurs, des industriels comme *International Business Machine*s (IBM) [2004-67] et Microsoft [2004-62] ont aussi rendu publiques leur vision de l'IDM. Ainsi, l'IDM prend son origine dans la convergence de toutes ces visions et des avancées techniques de chacun.

L'originalité de l'IDM ne réside pas dans le recours systématique aux modèles pour le développement logiciel comme le laisserait entendre sa terminologie [2004-68]. Plusieurs méthodes de modélisation, telles que Merise ou SSADM, préconisent aussi l'utilisation de modèles mais dont le rôle s'achève aux phases amont du développement logiciel :l'analyse et la conception. Les modèles servent alors à faciliter la communication et compréhension entre les différents acteurs mais n'interviennent pas dans la phase de production, de maintien et d'évolution. Nous parlons dans ce cas de modèles « contemplatifs ».

L'IDM a pour objectif de rendre les modèles « productifs » sur tout le cycle de vie du système et à tous les niveaux d'abstraction. Pour y parvenir, les modèles doivent être décrits formellement afin d'être interprétés et exécutés par une machine. Dès lors, ces modèles permettent d'industrialiser la production logicielle, jusque-là centrée sur le code produit par l'informaticien [2005-69].

En mettant à profit des disciplines telles que la modélisation par objets, l'ingénierie des langages, la compilation de langages, les méthodes formelles ou encore la programmation par composants, l'IDM offre un cadre intégrateur reposant sur quelques concepts fondamentaux : la notion de modèle et la relation *ReprésentationDe*, la notion de métamodèle et la relation *ConformeÀ*.

3.2 Concepts fondamentaux

3.2.1 Modèle et Représentation

La notion de modèle est centrale dans l'IDM car, comme nous venons de l'évoquer, l'enjeu de cette approche est de rendre les modèles productifs sur tout le cycle de vie du système. Il n'existe pas de définition universelle de la notion de modèle. En nous appuyant sur les définitions données dans les travaux [1967-70] [2001-63] et [2003-71], nous adoptons la définition suivante du terme modèle :

Definition 1. *Un modèle est une abstraction d'un système, selon le bon point de vue, qui permet de répondre à des questions prédéfinies sur ce système en lieu et place de celui-ci.*

De cette définition découle la première relation fondamentale de l'IDM qui lie le modèle et le système qu'il représente. Celle-ci est nommée *Représente* et notée μ. Bien que la relation *Représente* ne soit pas nouvelle dans l'ingénierie logicielle (Merise, UML), l'IDM a permis d'en définir les contours [2003-72] [2003-71] [2004-66].

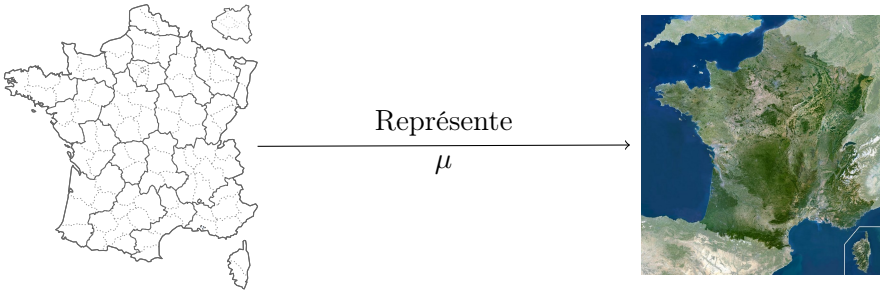

FIGURE 3.1 – Relation entre système et modèle [2006-8]

Cette définition n'est pas restreinte à l'informatique et pourrait s'appliquer à n'importe quel système. La figure 3.1 reprend l'exemple connu de la cartographie où une carte géographique joue le rôle de modèle pour le territoire français qui joue le rôle su système réel.

L'intérêt de l'IDM est de produire des modèles exploitables informatiquement. Ceci n'est possible que si ces modèles sont décrits par des langages formels. Il devient alors important de bien définir ces langages à l'aide de métamodèles.

3.2.2 Métamodèle et Conformité

L'originalité de l'IDM ne réside pas dans la relation de représentation qui trouve plutôt son origine dans les méthodes de modélisation telles que Merise. L'apport de l'IDM est dans l'utilisation systématique de métamodèles pour la description des langages de modélisation.

Il existe plusieurs définitions de la notion de métamodèle dans la littérature. Cependant la définition suivante est communément admise [2004-68].

Definition 2. *Un métamodèle est un modèle du langage de modélisation qui sert à exprimer les modèles.*

Une autre définition courante mais erronée de la notion de métamodèle suppose qu'un métamodèle est un modèle de modèle. La figure 3.2 reprend l'exemple de la cartographie évoquée plus haut. Nous appliquons récursivement la relation *Représente* (μ) au territoire français. Ici une carte de la France joue le rôle de modèle du territoire français et un fichier

XML joue le rôle de modèle de la carte. Dans ce contre exemple, le fichier XML n'est pas un métamodèle de la France. Un métamodèle n'est donc pas un modèle d'un modèle.

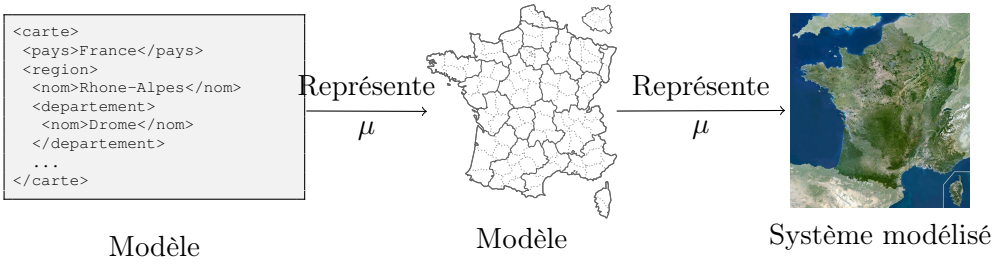

FIGURE 3.2 – Modèle de modèle selon l'exemple de la cartographie [2006-8]

Par ailleurs, le concept de métamodèle induit la deuxième relation fondamentale de l'IDM liant un modèle à son métamodèle. Cette relation est nommée *ConformeÀ* et notée χ [2004-66] [2004-73]. La figure 3.3 reprend l'exemple de la cartographie où la légende de la carte joue le rôle de métamodèle pour une carte de la France. En effet, pour être lisible, la carte doit être conforme à la légende.

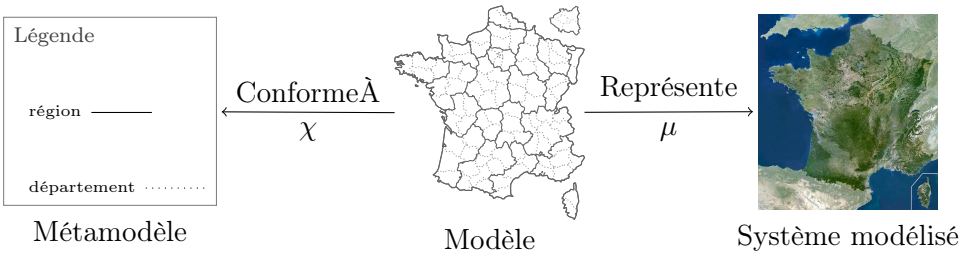

FIGURE 3.3 – Relations entre système, modèles, métamodèle et langage de modélisation [2006-8]

3.3 Transformation de modèle

Comme expliqué précédemment, la préoccupation majeure de l'IDM est de rendre les modèles opérationnels sur tout le cycle de vie des systèmes logiciels, depuis l'analyse et la conception jusqu'à la maintenance et l'évolution. La transformation de modèle se retrouve au cœur de l'IDM car c'est à travers elle que se fait l'automatisation des traitements apportés aux modèles. Dans cette section, nous donnons une définition de la transformation de modèle avant d'en présenter les types et les usages.

3.3.1 Définition de la transformation de modèle

L'OMG définit une transformation de modèle comme « le processus consistant à convertir un modèle en un autre modèle d'un même système » [2011-74].

Kleppe et al. [2003-40] proposent une définition moins générique en insistant sur l'aspect automatique de ce processus : « une transformation de modèle consiste en la génération automatique d'un modèle source en un modèle cible, selon une description établie de cette transformation ». Cette définition implique qu'une transformation est décrite à un niveau d'abstraction au dessus de celui des modèles : au niveau d'un métamodèle auquel elle doit se conformer.

Mens et Van Gorp [2006-75] étendent cette définition en considérant qu'une transformation est une opération qui peut avoir en entrée un ou plusieurs modèles source et en sortie un ou plusieurs modèles cible :
Definition 3. *Une transformation génère automatiquement un ou plusieurs modèles cible à partir d'un ou plusieurs modèles source, selon une description établie de la transformation.*

C'est cette dernière définition que nous adoptons dans ce document. Par ailleurs, notons que, si les métamodèles source et cible sont différents, la transformation est dite exogène. Si les métamodèles source et cible correspondent au même métamodèle, la transformation est dite endogène. Ces termes sont introduits par Mens et Van Gorp [2006-75].

3.3.2 Composants d'une transformation de modèle

La figure 3.4 illustre les composants d'une transformation de modèle : les modèles source, les modèles cible, la définition de la transformation et le moteur qui va opérer la transformation selon sa définition.

La description de la transformation définit la manière de transformer un ou plusieurs modèles source en un ou plusieurs modèles cible. Elle est créé dans un langage de transformation de modèle. Par exemple, si c'est un langage à base de règles, la description de la transformation consiste en un ensemble de règles de transformation à opérer sur les modèles cible [2003-40].

Un moteur de transformation exécute ou interprète la description. Il applique donc la description aux modèles source pour produire les modèles cible en suivant les étapes ci-dessous [2005-76] :

- identifier l'élément du ou des modèles source à transformer ;
- pour chaque élément identifié, produire l'élément cible qui lui est associé dans le ou les modèles cible ;
- produire une trace de la transformation qui lie les éléments du ou des modèles cibles aux éléments du ou des modèles source.

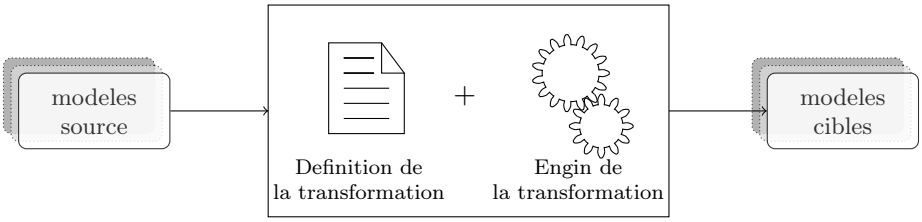

FIGURE 3.4 – Composants d'une transformation de modèle

3.3.3 Usages de la transformation de modèles

Les transformations de modèles sont au cœur d'une démarche dirigée par les modèles : elles permettent d'automatiser les manipulations subies par les modèles telles que la modification, la création, l'adaptation, la composition ou encore le filtrage de modèles, à travers la réutilisation systématique d'informations contenues dans les modèles existants.

Il est possible de recourir aux transformations de modèles sur tout le cycle de vie d'un système. Les usages les plus répondus sont le raffinement, l'intégration d'outils, la composition, l'analyse, la simulation et l'optimisation que nous présentons dans la suite de ce document.

3.3.3.1 Raffinement

Le raffinement consiste à rajouter plus de détails au modèle initial. Ce type de transformation peut aussi bien être endogène (métamodèles source et cible identique) ou exogène (métamodèle source et cible différents). Le raffinement se prête parfaitement à toute la partie descendante du cycle en V où les modèles passent à des niveaux d'abstraction plus bas. Ceci revient à faire des transformations successives de type modèle-à-modèle et une transformation de type modèle-à-texte pour aboutir au code final.

Raffiner un modèle revient à décomposer des concepts de haut niveau, à choisir un algorithme particulier, à spécialiser un concept pour un contexte donné ou encore à le concrétiser sous forme d'une solution exécutable par une machine en générant le code à partir de modèles de plus haut niveau d'abstraction [2000-77].

3.3.3.2 Intégration d'outil

Il existe une panoplie d'outils disponibles pour créer, manipuler, analyser ou encore simuler des modèles. Souvent ces outils utilisent des métamodèles internes et des espaces techniques qui leurs sont propres. Ainsi, l'échange de modèle entre ces outils est compromis et l'interopérabilité est fortement entravée. L'utilisateur se trouve obligé d'utiliser un seul et même outil sur tout le cycle de vie du système et ne peut donc pas tirer avantage des possibilités offertes par d'autres outils plus adaptés à ses besoins à certaines étapes.

L'intégration d'outil est une solution pour palier la divergence syntaxique et sémantique des outils et des langages de modélisation par le biais la transformation de modèle [2005-76]. Ce type de transformation permet de naviguer entre deux métamodèles, de synchroniser des modèles qui évoluent séparément sur des outils distincts, de faire des mapping entre métamodèles pour maintenir la cohérence des modèles conformes à ces métamodèles. Il sera donc possible de faire appel à des outils mieux adaptés à chaque étape du cycle de vie.

3.3.3.3 Composition

Pour réduire la complexité inhérente à la modélisation et à l'analyse de grands systèmes, tels que les Smart Grids par exemple, il est possible d'adopter une approche par points de vue qui permet de séparer les préoccupations. Les modèles produits correspondent donc à ces différents points de vue qu'on peut ainsi valider séparément dans un premier temps. A l'issue de cette approche modulaire, on pourra composer ces modèles, c'est-à-dire les assembler, pour aboutir un modèle global du système.

Dans le cas le plus simple, les deux modèles à composer sont conformes à un même métamodèle. Cependant, il est aussi possible de composer deux modèles conformes à deux métamodèles différents.

Les deux modèles à composer peuvent aussi présenter des concepts en commun. Deux techniques existent pour composer des modèles :

- la première technique consiste à les fusionner. Dans ce cas, le modèle final résultant de la composition doit contenir toutes les informations issues des modèles initiaux, sans duplication des informations communes [2006-78]. [2008-79] présente un framework générique capable de composer des modèles indépendamment de leurs langages de modélisation. L'approche consiste à identifier les éléments qui représentent le même concept dans les deux modèles à composer et à les fusionner dans un nouveau modèle qui représente une vue intégrée de ces concepts. Il est aussi possible de spécialiser le framework pour un métamodèle particulier mais qui reste conforme au MOF ;
- la deuxième technique consiste à les tisser. Dans ce cas, on crée des correspondances entre les éléments qui représentent un même concept. Un métamodèle générique est créé pour définir les correspondances qui sont donc modélisées dans le modèle final. On y retrouve donc les éléments en commun dupliqués mais liés par un lien de correspondance.

Il est à noter que le modèle issu du tissage de deux modèles M_A et M_B peut être utilisé comme modèle intermédiaire que l'on note M_T pour la fusion de M_A et M_B. Dans ce cas L'opération de fusion consiste à produire un modèle M_{AB} en prenant comme entrée M_A, M_B et M_T. Cette technique est notamment utilisée par [2007-80] pour la composition semi-automatique de modèles.

3.3.3.4 Simulation

La transformation de modèle peut être utilisée pour simuler des modèles. En effet, une transformation de modèle peut mettre à jour le système modélisé. Dans ce cas, le modèle cible est une mise à jour du modèle source et la transformation est de type sur-place (modèles source et cible confondus).

Par exemple, [2011-81] simule un comportement simple d'un jeu de Pacman en utilisant la transformation de modèle. La transformation spécifie les règles de transition qu'une instance du jeu peut prendre (Pacman et fantôme se trouvant dans la même case, Pacman et pomme se trouvant dans la même case, etc.). En ingénierie des langages, ceci revient à définir la sémantique opérationnelle d'un langage de modélisation. L'exécution de la transformation anime le modèle en fonction du comportement qu'on lui confère.

La transformation peut aussi être utilisée comme intermédiaire dans la simulation de modèle. Des modèles en entrée d'un outil de simulation externe sont produits par une transformation des modèles qu'on souhaite simuler. Cette technique permet de tirer profit d'outils de simulation existant sur le marché en utilisant l'intégration d'outils.

3.3.3.5 Analyse et optimisation

La transformation de modèle peut être utilisée pour les activités d'analyse de modèle. Une analyse simple telle que le calcul de métrique de similarité entre deux modèles via la transformation de modèle est donnée dans [2007-80] avec un modèle de transformation écrit en *Atlas Transformation Language* (ATL) [2006-82].

Des analyses plus complexes sont possibles grâce à l'intégration d'outils d'analyse externes vers lesquels les modèles source sont transformés.

[2010-83] propose d'utiliser la transformation de modèle pour l'analyse de sûreté de fonctionnement dans le domaine de l'automobile. Les modèles source sont transformés en modèles conformes au métamodèle de l'outil d'analyse de sûreté de fonctionnement retenu.

L'optimisation vise à améliorer les propriétés non fonctionnelles des modèles telle que l'évolutivité, la fiabilité, la modularité, etc. L'optimisation est typiquement utilisée sur les modèles d'architecture. Les transformations utilisées pour l'optimisation sont de types endogènes car on cherche à affiner la conception de modèles existants. La réingénierie est un exemple de transformation utilisée pour optimiser les modèles : on cherche à améliorer la maintenabilité, la lisibilité et l'évolutivité des modèles.

3.3.4 Approches existantes pour la transformation de modèle

Le recours à la transformation de modèle est l'objet de recherches informatiques antérieures à l'apparition de l'approche IDM. Par exemple, les compilateurs utilisent la transformation

pour passer du code source au fichier binaire [1985-84]. Ce type de transformation est restreint au domaine de la programmation informatique. La transformation de modèle embrasse un domaine plus large encore.

Nous trouvons dans la littérature plus d'une trentaine d'approches différentes de transformation de modèle [2011-81]. Czarnecki et Helsen proposent une classification de ces approches selon plusieurs critères tels que le paradigme retenu pour définir la transformation, la relation entre les modèles sources et cibles, la directivité de la transformation, le nombre de modèles cible et source, l'orchestration et l'ordonnancement des règles de transformation, etc. [2006-85]. [2011-86] définit les trois grandes catégories d'approches suivantes.

Par programmation : Les modèles offrent une interface qui permet d'écrire les transformations dans un langage de programmation. Mais cette technique relève plus de la programmation que de la modélisation. Ce sont en fait des applications informatiques qui ont la particularité de manipuler des modèles. L'avantage de cette approche est qu'elle utilise un langage de programmation généraliste tel que Java ou C++ pour écrire les transformations. Ainsi le programmeur n'a pas besoin d'apprendre un nouveau langage. Cependant ces applications ont tendance à devenir difficilement maintenables.

Par template : Dans cette approche, des canevas des modèles cibles sont définis. Ces modèles contiennent des paramètres qui sont remplacés par les informations contenues dans les modèles source. Ce type de transformation est souvent utilisé pour les transformations qui génère des modèles dont la syntaxe concrète est textuel. Cette approche est associée au visitor-pattern qui va traverser la structure interne du modèle source.

Par modélisation : Cette approche vise à appliquer les principes de l'IDM aux transformations de modèle elles-mêmes. Les modèles de transformation sont ainsi prennes, réutilisables et indépendants des plates-formes d'exécution [2006-87]. Pour cela, des langages de modélisation dédiés à l'activité de transformation de modèles sont utilisés. Cette approche considère donc la transformation comme un modèle à part entière conforme à un métamodèle de transformation. La figure 3.5 illustre cette approche en positionnant la transformation par rapport aux niveaux d'abstraction de l'IDM. Elle corrobore ainsi la vision unificatrice de l'IDM à travers le paradigme du « tout est modèle [2005-69].

3.4 Langages et outils pour la transformation de modèle

Sans viser à l'exhaustivité, nous introduisons succinctement quelques langages et outils dédiés à la transformation de modèles.

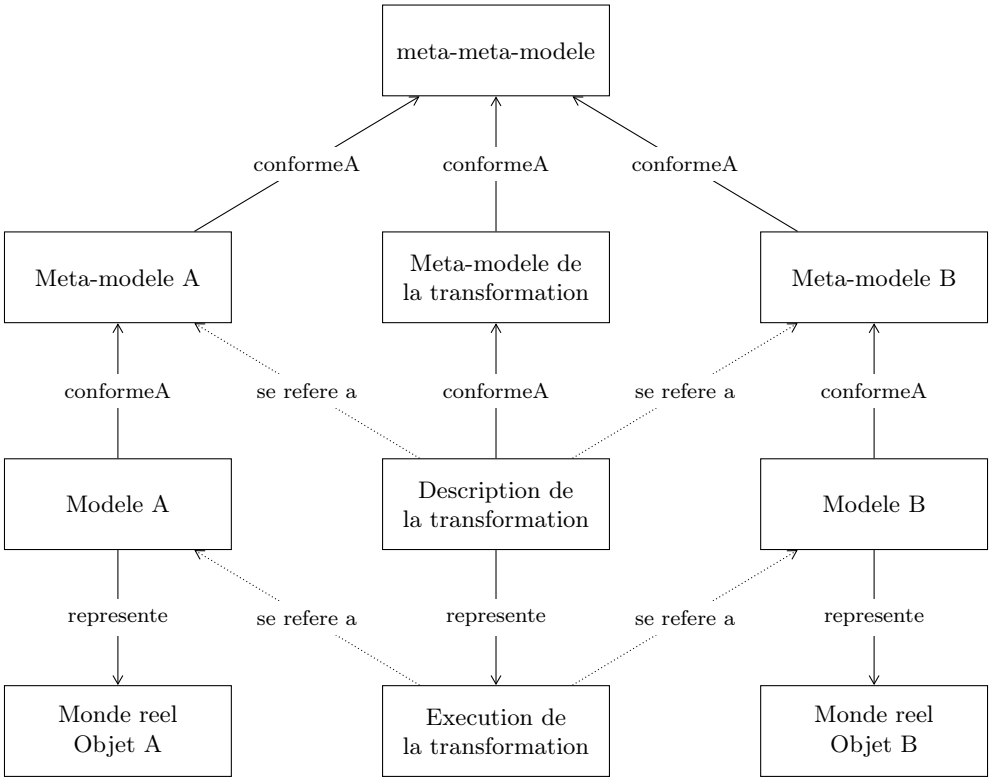

FIGURE 3.5 – Méta niveaux d'une transformation de modèle

3.4.1 ATL

ATL [1] est né de la volonté de proposer des langages de modélisation dédiés à la transformation de modèle en définissant un métamodèle et des outils pour l'exécution des transformations.

ATL est un langage hybride (déclaratif et impératif) à base de contraintes *Object Constraint Language* (OCL) [2006-82].

ATL peut réaliser des transformations sur place, c'est-à-dire, une transformation où le modèle source et le modèle cible sont confondus en utilisant le mode raffinement de modèle.

3.4.2 QVT

Query View Transformation (QVT) [2008-88] [2011-74] est un standard de l'OMG. Le métamodèle de QVT est conforme au MOF. Comme ATL, QVT se base sur OCL pour accéder aux éléments des modèles. QVT définit deux langages de transformation de type modèle-à-modèle. QVT *Declarative* (QVTd) [2] est un langage déclaratif. QVT *Operational*

1. http ://www.eclipse.org/atl/
2. http ://projects.eclipse.org/projects/modeling.mmt.qvtd

(QVTo) est un langage impératif [3].

3.4.3 Kermeta

Kermeta [4] est un langage généraliste de méta-modélisation exécutable et de méta-programmation orientée objet qui peut aussi décrire des transformations de modèle. Intégré à EMF, il est doté d'un métamodèle conforme au MOF qu'il étend avec un langage d'action impératif utilisé pour écrire le corps des opérations définies sur les concepts d'une syntaxe abstraite (ce qui revient à doter une syntaxe abstraite d'une sémantique opérationnelle). On peut ainsi décrire n'importe quel traitement sur un modèle ce qui est assimilé à une transformation de modèle.

3.4.4 Acceleo

Acceleo [5] est un langage de transformation de modèles pour lequel les modèles cible ont une syntaxe concrète textuelle. Il suit une approche de transformation par *template*. Le template correspond au modèle cible qui prend la forme d'un texte avec des espaces réservés aux informations provenant des modèles source. Accelo utilise OCL pour accéder au modèles source.

3.5 L'Ingénierie Dirigée par les Modèles pour l'Architecture d'Entreprise

L'IDM se cantonne souvent au processus de développement logiciel en offrant une aide supplémentaire à l'analyse et la validation des modèles de spécification et en automatisant certaines tâches à travers la génération de code par exemple. Mais des disciplines plus orientées métier telle que l'EA peuvent aussi tirer partie des avantages qu'apporte l'IDM à l'ingénierie logicielle.

3.5.1 Méta-modélisation en Architecture d'Entreprise

Les cadres d'EA contribuent à réduire la complexité liée à un système telle que l'entreprise en décomposant celle-ci en plusieurs vues. Les architectes d'entreprise doivent ensuite représenter les artefacts qui composent ces différentes vue. Le recours aux techniques de modélisation n'est donc pas étranger à l'EA. Les architectes d'entreprise ont besoin d'une représentation claire et pertinente des composants de l'entreprise pour leur propre

3. http ://projects.eclipse.org/projects/modeling.mmt.qvt-oml
4. http ://www.kermeta.org/
5. http ://www.eclipse.org/acceleo/

compréhension mais aussi pour faciliter la communication avec les autres parties prenantes comme les experts métier, les architectes fonctionnels, les architectes applicatifs, etc.

Souvent, les cadres d'architecture ne préconisent pas de langages de modélisation en particulier et se contentent d'ériger un ensemble de bonnes pratiques. Pour représenter l'entreprise, les architectes utilisent alors leurs propres systèmes de notations. Si deux architectes appartenant à des filiales différentes d'une même entreprise utilisent des notations différentes, des problèmes de communication, de collaboration et d'interopérabilité à l'échelle de l'entreprise risquent d'apparaître. L'architecture de l'entreprise perd alors de sa crédibilité en tant que référentiel d'entreprise.

Graphiques ou textuelles, ces notations manquent de plus d'une sémantique précise et formalisée entrainant la création de modèles purement contemplatifs. Les outils de visualisation et d'analyse sont d'autant plus difficiles à développer.

L'*Open Group* propose de mettre en cohérence les notations utilisées en EA en créant un langage de modélisation standardisé nommé Archimate. Archimate sert ainsi à décrire l'entreprise, sa structure organisationnelle, ses processus, ses règles métier ou encore son SI. Archimate est un langage de représentation graphique utilisé pour documenter et à communier les architectures d'entreprise au sein d'une même organisation ou entre différentes organisations. Archimate est en outre étroitement lié au cadre d'architecture TOGAF.

Bien que concis, le métamodèle d'Archimate vise à couvrir la majorité des concepts intervenants dans la création d'une architecture d'entreprise. Archimate utilise trois vues : métier, applicative et technique. Pour chacune des vues, trois types d'éléments différents sont spécifiés : structure active, comportement et structure passive (voir figure 3.1). Les structures actives correspondent aux éléments dotés d'un comportement. Il peut s'agir d'un acteur métier, d'un module applicatif ou d'un élément de l'infrastructure technique. Le comportement est défini comme un ensemble d'activités ou de tâches poursuivies par un ou plusieurs éléments de la structure active. Il peut s'agit par exemple d'un processus métier. La structure passive comportent les éléments manipulés par les éléments de la structure active, comme par exemple les objets métier.

	Structure passive	**Comportement**	**Structure active**
Metier			
Application			
Techonologie			

Table 3.1 – Composants du langage Archimate

Le métamodèle du langage Archimate est un adossé d'une syntaxe concrète graphique permettant visualiser les modèles. Il associe par exemple des couleurs différentes aux éléments en fonction de leur type. Les éléments de type structure active sont bleus, ceux de type comportement sont jaunes et enfin ceux de type structure passive sont verts. Mais

aucune sémantique standard d'exécution n'est formalisée. Les modèles d'architecture sont ainsi souvent purement contemplatifs et les outils implémentant Archimate n'offrent pas la possibilité d'exécuter ces modèles à des fins d'analyse de structure et de comportement.

3.5.2 Approches d'Architecture d'Entreprise recourant à l'Ingénierie Dirigée par les Modèles

L'IDM a prouvé sa capacité à adresser des systèmes complexes [2007-89]. L'application des techniques de l'IDM aux approches d'EA font l'objet de quelques rares travaux [2013-56]. Les travaux de [2003-90] font figure de précurseurs. Ces derniers décrivent un mapping entre le cadre Zachman et les vues MDA (figure 3.2). Mais le périmètre de ces travaux se réduit à l'architecture IT de l'entreprise plutôt qu'à l'entreprise dans son ensemble.

Abstractions (colonnes) ⟷

Perspectives (lignes) ↕	**Donnees** *Quoi*	**Fonctions** *Comment*	**Personnel** *Qui*	**Temps** *Quand*	**Motivation** *Pourquoi*	
Portee *Planificateur*						CIM
Modele metier *Proprietaire*						CIM
Modele systeme *Concepteur*						PIM
Modele Technologique *Realisateur*						PSM
Modele detaille *Sous-traitant*						Code

TABLE 3.2 – Mapping Zachman/MDA [2003-90]

[2014-46] proposent d'étendre la démarche MDA, traditionnellement appliquée au développement de systèmes informatiques, à l'ensemble de l'entreprise pour mettre en place une organisation dirigée par les modèles ou MDO (*Model Driven Organisation*). Comme l'illustre la figure 3.6, une MDO comprend un modèle de l'organisation (*Model of the Organization*), un modèle spécifique à une plateforme (*Platform Specific Model*) et une plateforme d'entreprise (*Platform for Organization*).

Le modèle de l'organisation correspond à l'ensemble des modèles métier présentant les processus, les objectifs, les acteurs, les ressources. L'organisation fait appel à son IT pour réaliser ses objectifs métier. L'ensemble du système informatique correspond alors à la plateforme opérationnelle de l'entreprise. Comme pour MDA, le modèle spécifique à une plateforme est dérivé du modèle de l'organisation et sert de pivot pour générer automatiquement la plateforme de l'organisation par transformation de modèles.

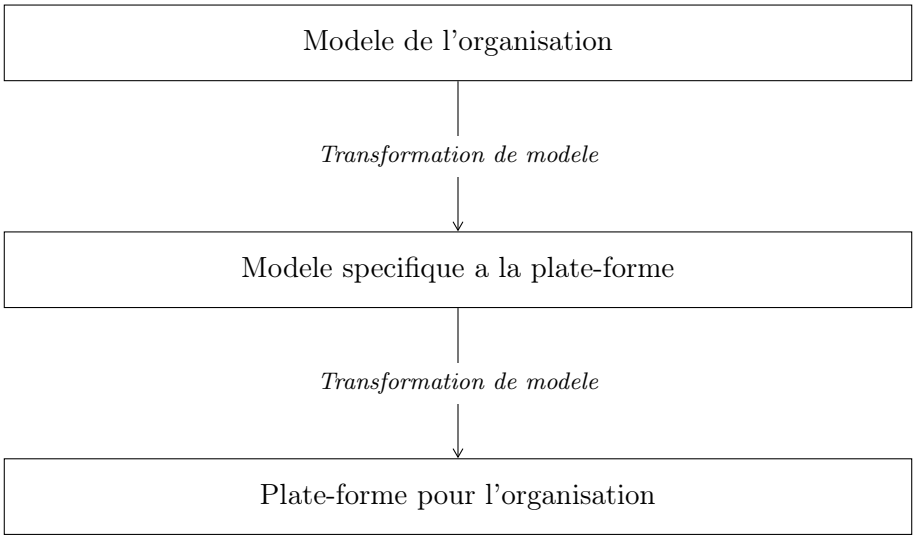

FIGURE 3.6 – Architecture d'une organisation dirigée par les modèles (à remplacer par une belle figure) [2014-46]

Dans leurs travaux, [2014-46] proposent un cadre théorique et une étude d'opportunité quant à la généralisation de MDA à l'échelle de l'entreprise. Cependant MDA reflète la vision particulière de l'OMG concernant l'IDM. L'approche MDA est de ce fait restreinte aux standards de l'OMG. Les travaux de Clark et al. sont donc limités par l'approche MDA (celle-ci étant un sous-ensemble de l'IDM).

L'IDM préconise le recours aux modèles exécutables pendant tout le cycle de vie du système à implémenter. Cependant, à l'échelle d'une entreprise, l'utilisation de ce type de modèles est souvent limité à l'implémentation des systèmes informatiques, c'est à dire à l'implémentation de la vue applicative et la vue technique d'une architecture d'entreprise. L'usage des modèles en tant artefacts exécutables est peu commun en EA [2013-45].

Parmi les travaux mettant à contribution les techniques de l'IDM nous citons ceux de [2011-49]. Ces derniers proposent un langage concis et exécutable pour modéliser et simuler les différentes vues d'une architecture d'entreprise. Cependant ce langage est principalement textuel. Il offre une visualisation graphique des aspects structuraux d'une organisation sous la forme d'un diagramme de classes. Mais la syntaxe concrète pour la spécification du comportement est essentiellement textuelle. L'adoption de ce langage par les parties prenantes à profil non technique n'est pas évidente. Or l'EA a pour vocation d'offrir un référentiel compréhensible et partagé par toutes les parties prenantes.

Les travaux de [2013-91] tirent profit des capacités de l'IDM à automatiser la manipulation des modèles pour offrir une assistance dans la création et la visualisation d'architectures d'entreprise. La plateforme proposée s'appuie sur TOGAF et fournit un support pour la gouvernance et la prise de décision souvent gérées manuellement. Ces travaux se focalisent sur les techniques IDM permettent de (1) concevoir une cartographie de l'existant par

rétro ingénierie à partir des données disponibles dans le SI, (2) adapter la représentation graphique des modèles au besoin des utilisateurs concernés, (3) gérer plusieurs vues d'un même système et automatiser leur manipulation. [2013-91] démontrent que les techniques de l'IDM apportent des réponses à certaines limitations liées aux pratiques actuelles d'EA mais ne supportent pas la simulation des modèles obtenus.

[2015-58] utilisent les techniques de l'IDM (en particulier la métamodélisation) pour analyser la composante dynamique des modèles Archimate à travers la simulation de propriétés non fonctionnelles. Dans un premier temps, ils réduisent le métamodèle d'Archimate en ne retenant que les concepts indispensables à la modélisation des propriétés non fonctionnelles. Comme Archimate est d'abord conçu pour modéliser la structure des architectures d'entreprise, [2015-58] enrichissent dans un deuxième temps le métamodèle obtenu avec deux autres types de concepts (1) des concepts pour décrire le comportement et (2) des concepts uniquement liés à l'exécution.

Les travaux de [2015-58] illustrent les avantages de la métamodélisation et la définition d'une sémantique formelle pour l'exécution de modèles d'architecture d'entreprise. Néanmoins, ces travaux ne concernent que les propriétés non fonctionnelles des modèles d'architecture. Ces travaux formalisent aussi bien le métamodèle de la vue métier que ceux des vues applicative et technique. Cependant la simulation des différentes se fait séparément les unes des autres. L'approche ne permet donc pas de simuler simultanément l'ensemble de l'architecture.

3.5.3 Langages exécutables pour l'Architecture d'Entreprise

L'automatisation de l'analyse des modèles d'architecture d'entreprise s'avère particulièrement cruciale dans le contexte particulier des Smart Grids : évolution des cadres législatifs, apparition de nouveaux partenaires, hétérogénéité des interactions avec les clients finaux via les compteurs intelligents, les smartphones, les tablettes tactiles, etc. Or l'exécution des modèles est indispensable à l'automatisation de l'analyse de leur structure et de leur comportement. Exécuter des modèles apporte ainsi une aide précieuse à la validation d'une architecture d'entreprise dès les premières phases de son cycle de vie et augmente son agilité et son évolutivité. D'une part, l'exécution des modèles facilite leur exploration par les experts métier en levant les ambiguïtés engendrées par les modèles purement contemplatifs. D'autre part, exécuter des modèles permet d'évaluer et de comparer des architectures de manière objective en définissant des métriques et des indicateurs d'évaluation.

L'exécution des modèles est rendue possible par la définition d'une sémantique exécutoire du langage dans lequel ces modèles sont exprimés. La sémantique d'un langage correspond au sens que peuvent prendre les concepts manipulés et leurs agencements lorsqu'ils sont instanciés au niveau des modèles [2012-92]. La définition de la sémantique du langage dépend de l'objectif poursuivi : simulation, génération de code, vérification, compilation, etc. L'expression de la sémantique d'un langage fait l'objet d'intenses recherches en ingénierie des langages, et en particulier sous la thématique des langages formels [2007-93].

Comme l'atteste le manifeste d'IBM [2006-94], les axes principaux de l'IDM sont (1) les

standards ouverts, (2) l'automatisation et (3) la représentation directe. Compte tenu de ces axes, pour modéliser les différentes vues d'une architecture d'entreprise, nous préconisons l'utilisation de **langages standardisés, exécutables et compréhensibles** par les acteurs concernés par la modélisation d'architectures d'entreprise pour les Smart Grids.

Nous identifions plusieurs langages, issus de l'ingénierie des langages et en particulier de l'IDM, pouvant satisfaire ces critères :

- Un sous-ensemble de UML [6] limité au diagramme de classes et au diagramme d'activité possède désormais une sémantique d'exécution décrite par le standard fUML [7] (*Foundational UML*). Les diagrammes de classes conviennent pour la description des modèles d'information tandis que les diagrammes d'activité sont adaptés à la description de la dynamique d'un modèle et au comportement attendu des fonctions ;
- Le standard BPMN [8] (*Business Process Model and Notation*) est un langage de modélisation graphique permettant de décrire tous les aspects d'un processus métier à l'aide d'un seul type de diagramme. Ce formalisme présente l'avantage d'avoir une sémantique d'exécution bien définie permettant le développement d'outils pour la simulation de modèles de processus métier. BPMN est parfaitement adapté à la description de processus métier ;
- Le standard OCL [9] est un langage textuel standard d'expression de contraintes. Il est adosser à UML pour exprimer les propriétés difficiles à capturer dans des diagrammes UML. L'exécution d'OCL se fait à travers la transformation de modèle en ciblant un langage d'expression de contrainte de plus bas niveau qui soit exécutable comme MiniZinc [2007-95], ou à travers son utilisation au niveau du métamodèle avec OCLinEcore [10].

6. http ://www.omg.org/spec/UML/
7. http ://www.omg.org/spec/FUML/
8. http ://www.omg.org/spec/BPMN/2.0/
9. http ://www.omg.org/spec/OCL/
10. wiki.eclipse.org/OCL/OCLinEcore

Deuxième partie

Contribution

Chapitre 4

Démarche

Sommaire

Créer des modèles d'architecture d'entreprise à des fins d'analyse par simulation implique de suivre un processus précis. L'objectif de notre travaux est de définir le processus de modélisation et de simulation à adopter, les rôles impliqués, les artefacts conceptuels requis, ainsi que les outils et les langages adéquats pour mener des analyses de structure et de comportement.

Notre contribution est double. D'abord, nous proposons une démarche de modélisation et d'analyse qui s'appuie sur un cadre d'architecture multi-vues. Nous dotons ce cadre d'architecture d'une vue supplémentaire qui est est la vue intégration. Cette vue a pour but d'adresser les problématiques de cohérence et d'alignement. Ensuite, nous mettons à profit les langages et standards de l'IDM permettant de modéliser et de simuler les architectures d'entreprise.

4.1 Approche conceptuelle pour la modélisation et l'analyse d'architectures d'entreprise

L'EA peut avoir différents objectifs. Kurpjuweit et Winter [2007-96] identifient trois objectifs de l'EA par rapport aux SI de l'entreprise : (1) la documentation et la communication, (2) l'analyse et la compréhension et (3) la conception. Nous étendons cette vision centrée

sur les SI à l'ensemble de l'entreprise en adoptant l'école de pensée « intégrative » comme décrite par la taxonomie de Lapalme (voir section 2.1.3, page 16 de l'état de l'art).

Nous considérons donc que la documentation et la communication, l'analyse et la compréhension et enfin la conception concernent tout le système entreprise et pas seulement sa composante SI. Par exemple, contrairement à une EA centrée sur les SI, les processus métier sont modélisés et évalués tout autant que l'architecture applicative. La figure 4.1 illustre notre approche conceptuelle. Celle-ci fait intervenir plusieurs types d'acteurs intervenant à plusieurs niveaux de l'entreprise :

- un analyste métier qui conçoit la vue métier à travers la modélisation et l'évaluation des processus métier mettant en œuvre la stratégie de l'entreprise ;
- un architecte fonctionnel qui traduit les processus métier en termes de fonctions logiques ;
- un architecte applicatif qui traduit la vue fonctionnelle en un ensemble structuré d'applications implémentant les différentes fonctions ;
- un architecte d'entreprise dont la mission est de mettre en cohérence l'ensemble des vues.

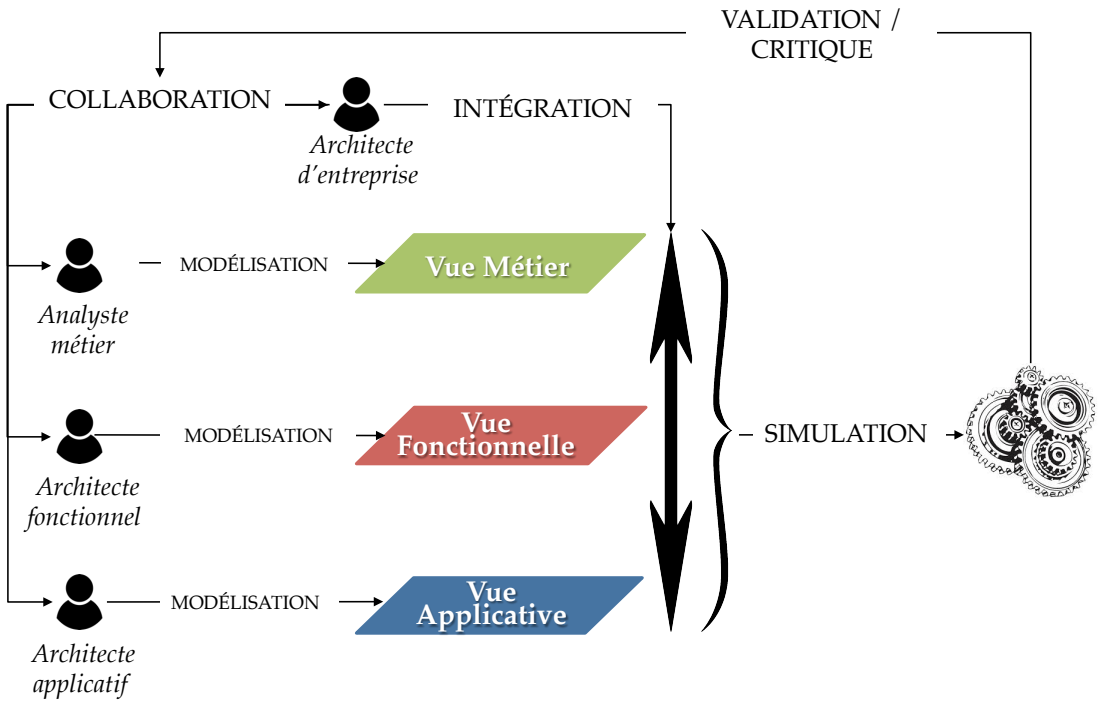

FIGURE 4.1 – Approche conceptuelle pour l'analyse d'une architecture d'entreprise

L'analyste métier, l'architecte fonctionnel et l'architecte applicatif collaborent entre eux et avec l'architecture d'entreprise pendant tout le processus de modélisation. Par la suite, l'activité d'intégration incombe à l'architecte d'entreprise, détenteur de la vision globale.

Mais l'intégration des vues implique de rebouclage avec les autres acteurs (analyste métier, architecte fonctionnel et architecte applicatif) pour garantir l'alignement business/IT. Les étapes de modélisation et d'intégration de notre approche répondent donc aux objectifs de conception et de communication tels que définis par Kurpjuweit et Winter [2007-96].

Comme relaté dans l'état de l'art, l'analyse fait partie des activités les moins courantes de l'EA quelle qu'en soit l'école de pensée [2008-97] [2013-55]. Et même lorsqu'une architecture d'entreprise est analysée très peu d'approches recourent à la simulation pour analyser l'aspect comportemental [2011-60] [2015-58]. La simulation est pourtant une technique reconnue pour évaluer le comportement d'un système et/ou évaluer plusieurs stratégies concernant son fonctionnement [1975-57]. Notre approche préconise de simuler les modèles issus de activités de modélisation et d'intégration afin de les valider ou les critiquer par l'ensemble des acteurs impliqués dans l'EA. Nous couvrons ainsi l'objectif d'analyse tel que préconisé par Kurpjuweit et Winter [2007-96].

4.2 Un cadre d'architecture dirigé par les modèles exécutables

4.2.1 Approche par points de vue

Toute simulation d'un système commence par sa modélisation. Pour modéliser des systèmes complexes comme les architectures d'entreprise, nous adoptons une approche par points de vue. Celle-ci facilite la conception des modèles par les acteurs impliqués en séparant leurs préoccupations respectives. Elle permet également de présenter les modèles obtenus, ainsi que les résultats de simulation de manière plus compréhensible à ces acteurs, car chaque point de vue n'utilise que les concepts métier propres à chaque acteur, selon sa perspective.

Dans notre approche, nous nous intéressons aux vues métier, fonctionnelle et applicative comme l'illustre la figure 4.2. Mais nous souhaitons aussi étendre nos travaux à la vue technique. Plusieurs raisons motivent l'utilisation de ces points de vue. D'abord, la vue métier et la vue applicative sont incontournables pour n'importe quel cadre d'architecture. Ensuite, selon les cadres d'architecture, la vue fonctionnelle est modélisée de deux manière : elle est soit intégrée à la vue applicative sous forme de services (Archimate, TOGAF, RM-ODP), soit modélisée à part entière dans une vue dédiée (Club Urba, SGAM, Zachman). Nous prenons le parti de modéliser explicitement les fonctions dans une vue dédiée. En effet, passer directement de la vue du métier à la vue applicative peut être en quelque sorte brutal pour l'analyste métier mais aussi pour l'architecte applicatif. La vue fonctionnelle permet une transition progressive de la logique métier vers l'architecture logicielle.

Nous ne modélisons pas les informations dans une vue dédiée contrairement aux cadre RM-ODP ou SGAM. Nous explicitons les informations en tant qu'aspect pour chacune des autres vues comme recommandé par le cadre Zachman (Le quoi de la dimension horizontale).

L'aspect « *information* » permet d'avoir un modèle explicite des données utilisées dans chacune des vues métier, fonctionnelle et applicative :

- **L'aspect information du point de vue métier**
  Cet aspect établit le modèle de données métier qui décrit les objets ou concepts métier manipulés par le processus métier. Ce modèle est peu sujet au changement, sauf évolution importante des pratiques métier. Il est aussi à l'origine du découpage en bloc par entité métier de la vue fonctionnelle ;
- **L'aspect information du point de vue fonctionnel**
  Cet aspect établit le modèle de données fonctionnelles qui donne le type des données utilisées par les blocs fonctionnels nécessaires à la réalisation de processus métier. Elle décrit leurs caractéristiques et leurs relations sous forme de diagrammes de classes par exemple ;
- **L'aspect information du point de vue applicatif**
  Cet aspect établit le modèle de données applicatives qui dépend fortement des applications choisies : il décrit les formats de données compatibles avec les modules applicatifs.

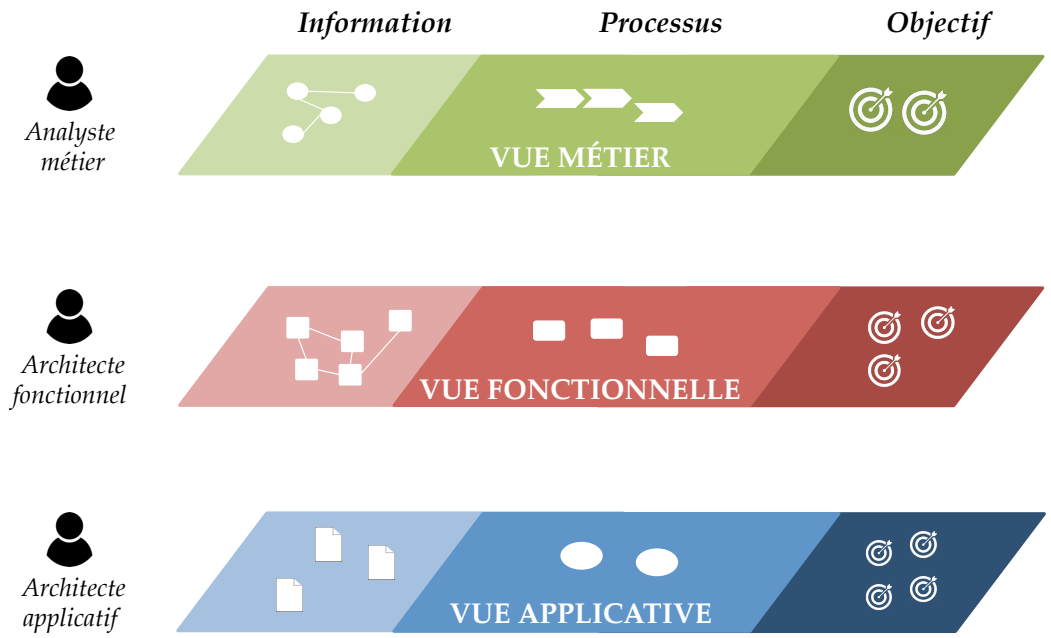

FIGURE 4.2 – Points de vue et aspects utilisés

De plus, nous modélisons le comportement de chaque vue dans l'aspect « *processus* ».[1]Ainsi nous retrouvons :

- **L'aspect processus du point de vue métier**

Cet aspect correspond aux processus métier de l'entreprise, décrits en utilisant les concepts métier, sans référence aux détails d'implémentation. Nous préconisons l'utilisation de formalismes standard pour la modélisation de processus métier qui soient exécutables, tels que les diagrammes d'activité fUML ou les diagrammes BPMN dans une perspective de simulation. Des langages spécifiques à un domaine (DSML) peuvent également être utilisés ;

- **L'aspect processus d'un point de vue fonctionnel**
  Cet aspect décrit les fonctions qui réalisent les processus métier ainsi que leur orchestration en tant que processus fonctionnels. Ces fonctions sont regroupées en blocs. Chaque objet métier identifié dans l'aspect Information de la vue métier correspond à un unique bloc fonctionnel. Ceci garantit la construction de blocs fonctionnels fortement décorrélés, avec une forte cohésion interne. Dans chaque bloc fonctionnel, on retrouve les opérations correspondant à une tâche donnée du processus qui impacte l'objet métier impliqué ;
- **L'aspect processus du point de vue applicatif**
  Cet aspect décrit les modules logiciels qui implantent les blocs fonctionnels ainsi que leur orchestration en processus applicatifs. Dans un premier temps, il est conseillé de dresser un inventaire de l'existant applicatif et d'en extraire les modules capables de réaliser les opérations des blocs fonctionnels. Ensuite, si aucune application ou module existant ne peut répondre au besoin des nouveaux processus métier, l'architecte technique fait le choix des nouveaux composants applicatifs à mettre en place. En plus d'identifier les composants applicatifs existants ou à développer, l'architecte applicatif spécifie leurs interconnexions tels que échange de messages, synchronisation de données, transfert de fichiers périodique.

Nous étendons chacune des vues par l'aspect « *objectif* ». Cet aspect correspond au « *pourqoi* » du cadre Zachman qui spécifie les motivations de l'architecture. D'une part, modéliser cet aspect permet de garder une traçabilité d'une entre les processus modélisés et les objectifs qu'ils sont censés remplir. D'autre part, il permet de décliner les objectifs métier en objectifs applicatifs, et les objectifs applicatifs en objectifs fonctionnels. Comme nos travaux adoptent l'école de pensée « Architecture du Système Entreprise », nous souhaitons évaluer non seulement la composante SI mais l'ensemble de l'entreprise dont la stratégie. L'aspect objectif est un moyen de modéliser explicitement la stratégie de l'entreprise, traduite en un ensemble cohérent d'objectifs métier et ce pour évaluer la capacité des processus mis en place à y répondre et pour évaluer la stratégie elle-même par rapport à la réalité de l'entreprise.

4.2.2 Intégration d'une architecture d'entreprise

L'alignement et la cohérence sont des problématiques centrales en EA [2005-54]. Notre contribution essentielle est de dédier un point de vue spécifique pour adresser ce type de problématique : le **point de vue intégration** illustré par la figure 4.3. Ce point de

vue définit un *mapping* d'alignement en spécifiant (1) les entités à aligner, (2) les liens de cohérence entre ces éléments et (3) les transformations de modèles nécessaires au raffinement des ces entités. Les transformations de modèles facilitent et automatisent les passages d'une vue à l'autre. Ce point de vue permet une intégration « IntraVue » et « InterVues » (entres les aspects d'une seule vue). La figure 4.4 donne un métamodèle de notre framework en explicitant les concepts abordés et leurs relations.

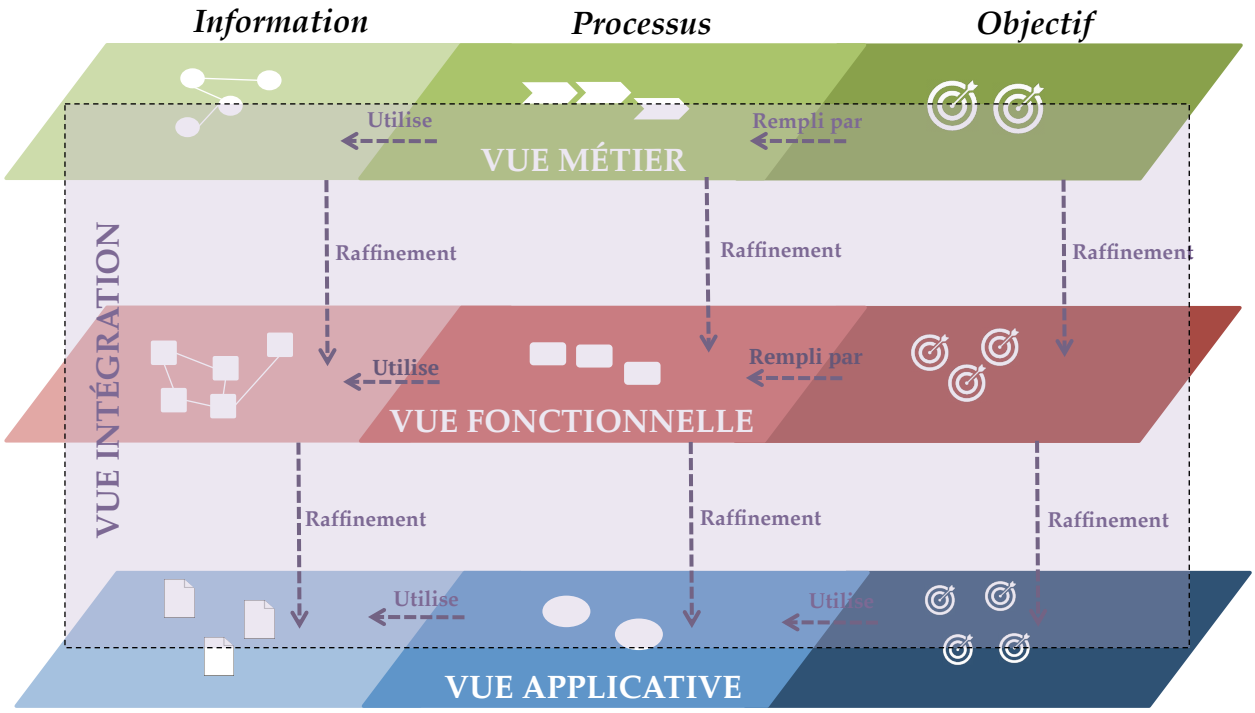

FIGURE 4.3 – Framework proposé avec la vue Intégration

L'intégration inter-vues décrit les relations entre les vues (sauf la vue intégration). La vue intégration est donc transverse à toutes les autres vues et permet de modéliser explicitement les notions de raffinement à travers la classe « InterVues » (voir figure 4.4), tant pour les aspects *processus* et *information* que sur les aspects *objectif*.

Nous donnons le métamodèle de la vue intégration dans la figure 4.5. Cette vue permet des vérifications horizontales à l'intérieur de chacune des vues. En effet, l'association « utilise » assure donc la compatibilité des données échangées entre les tâches d'un processus métier, les fonctions d'un bloc fonctionnel ou entre les modules d'une application (voir figure 4.5). L'association « rempli par » hérite aussi de la classe « IntraVue » et associe explicitement une entité à l'objectif qui lui est assigné. De cette manière, il est possible de tracer l'implémentation effective d'une stratégie métier à travers l'ensemble de l'entreprise,

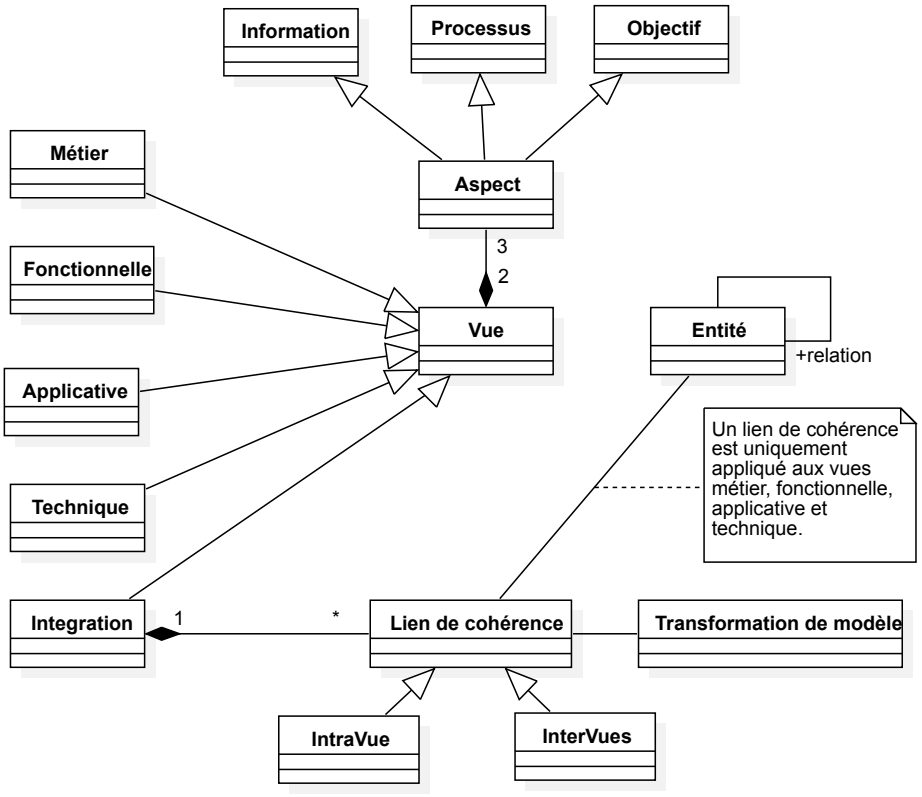

FIGURE 4.4 – Métamodèle du framework proposé

des entités métier à l'IT.

La définition d'un métamodèle précis pour le framework permet de plus de contrôler la conformité des modèles d'architecture. Ainsi, une entité du modèle (telle qu'une fonction, une tâche du processus métier, un objectif, etc.) appartient toujours à deux vues : une vue intégration en plus d'une autre vue parmi les vues métier, fonctionnelle, applicative et technique. Autre exemple, tous les objectifs de la vue métier doivent avoir des liens de cohérence inter-vues avec les objectifs de la vue fonctionnelle et de même pour ces dernier avec les objectif de la vue applicative et ainsi de suite. De même pour les autres entités des aspects information et processus. Une fois le modèle d'architecture créé, il est possible de vérifier sa conformité au métamodèle et de s'assurer que la cohérence entre les différentes entités de modèles est bien maintenue.

La vue intégration permet en outre de vérifier qu'une application implémente bien tous les blocs fonctionnels nécessaires au déroulement d'un processus métier. Cette vue donne ainsi accès aux informations de traçabilité qui permettent de déterminer l'impact d'une modification ou d'une défaillance d'un module applicatif sur les processus métier. Elle permet aussi de vérifier que les formats applicatifs permettent d'encoder les types de données fonctionnelles requis, qui eux-même raffinent les concepts métiers. Cette vue détermine les

éventuelles transformations de modèle nécessaires au déploiement en spécialisant l'association « raffine ». Le choix des modèles à transformer dépend fortement de leur nature (modèles graphiques, textuels, etc.) et mais aussi de leur niveau d'abstraction. Par exemple, la génération de code demande un modèle en entrée suffisamment détaillé pour exécuter une transformation pertinente. L'état de l'art actuel des langages de transformation de modèle privilégie l'usage des transformations de modèle entre la vue fonctionnelle et et la vue applicative.

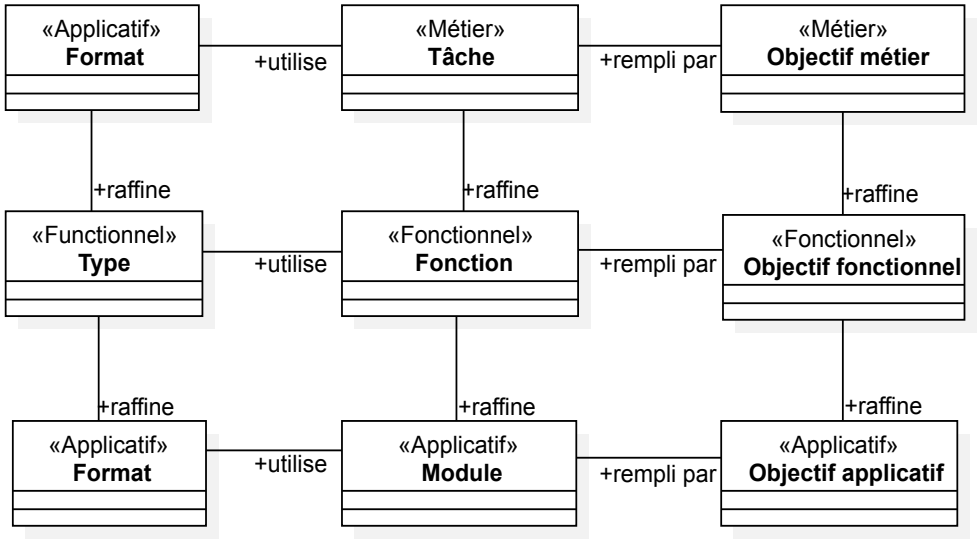

FIGURE 4.5 – Métamodèle de la vue intégration

4.2.3 Analyse de l'architecture d'entreprise

4.2.3.1 Analyse du comportement : simulation dirigée par les processus métier

Pour simuler le comportement d'une entreprise nous nous appuyant sur les modèles d'architecture d'entreprise préalablement définis par les différentes parties prenantes. L'EA permettent de capturer l'essentiel des composants d'une entreprise sous forme d'abstraction. Une approche par points de vue guide la décomposition d'une entreprise en vues pertinentes pour les différents acteurs. Ces vues apportent une aide supplémentaire à la définition du périmètre des modèles de simulation. La vue intégration permet en particulier de définir les liens entres les différentes vues, donc entre les différents modèles de simulation et de garantir la cohérence et donc la pertinence de la simulation. Comme les modèles de simulation sont directement dérivés de l'architecture d'entreprise telle qu'elle est définie par les parties prenantes, elle est d'autant plus facile à appréhender à comprendre. Ces derniers peuvent

aussi aisément communiquer et échanger autour des résultats de la simulation.

Les modèles d'entreprise doivent offrir un niveau d'abstraction suffisant à la compréhension, l'analyse et la communication. Les modèles doivent donc permettre d'abstraire les détails techniques et les nombreuses interconnexions tout en garantissant la traçabilité et la cohérence de l'ensemble de l'architecture. Notre approche consiste donc de mettre en lumière les composants et les relations qui sont critiques pour le comportement de l'ensemble de l'architecture. En effet, modéliser l'architecture d'une entreprise revient à la modélisation de systèmes complexes. Herbet Simon [1990-98] dans ses travaux de modélisation de systèmes complexes affirme que « l'approximation judicieuse et non la puissance de calcul d'une machine » reste la manière la plus effective d'adresser des systèmes complexes.

La simulation des processus métier est souvent réduite à de la simple animation visuelle de diagrammes pour vérifier l'orchestration des taches métier. Nous proposons de piloter la simulation du comportement de l'architecture par le processus métier. Dans ce cas, le calcul d'une valeur ne se fait pas au niveau de la tache métier mais du module applicatif qui l'implémente. Le processus métier est modélisé sous forme de diagrammes d'activité fUML. Dans ce cas les simulations du comportement de l'architecture d'entreprise est pilotée par les processus métier qui sont alors responsable d'orchestrer l'ensemble des modèles comme l'illustre la figure 4.6. La simulation est lancée après l'intégration de l'architecture à travers la création de lien de cohérences intra-vue et inter-vues. Ces liens sont par la suite utilisés pour mettre en œuvre la simulation. D'une part, la *Tâche A* de la figure 4.6 appelle le *Module A* car le *Module A* raffine la *Fonction A* qui elle-même raffine la *Tâche A*. D'autre part, les liens de cohérences intravue garantisse une compatibilité entre les informations envoyés par la *Tâche A* et celles attendues par la *Tâche B*, de même entre la *Fonction A* et la *Fonction B* et entre le *Module A* et le *Module B*.

En plus de s'appuyer sur les processus métier pour piloter la simulation de l'ensemble du comportement de l'architecture d'entreprise, notre approche consiste à mettre à profit les techniques et les langages de l'IDM pour faciliter l'automatisation de l'activité d'analyse du comportement. Nous nous appuyant sur le manifesto de IBM [2006-94] concernat l'IDM dans la sélection de techniques et langages qui soient pertinents pour notre approche. Le mnifesto de IBM recommande l'utilisation de langages (1) exécutables, (2) standardisés et (3) compréhensibles par les experts du domaine. C'est le cas de fUML, BPMN et OCL. La figure 4.1 fait la correspondance entre les langages et la possibilité de les utiliser selon les vues.

Ces langages permettent une exécution directe des modèles créés. Contrairement à d'autres méthodes de simulation de processus métier qui utilisent l'IDM pour isoler la définition du processus de son exécution. Ces méthodes font ensuite appel aux transformations de modèle pour automatiser la conversion entre les modèles de représentation et leur exécution.

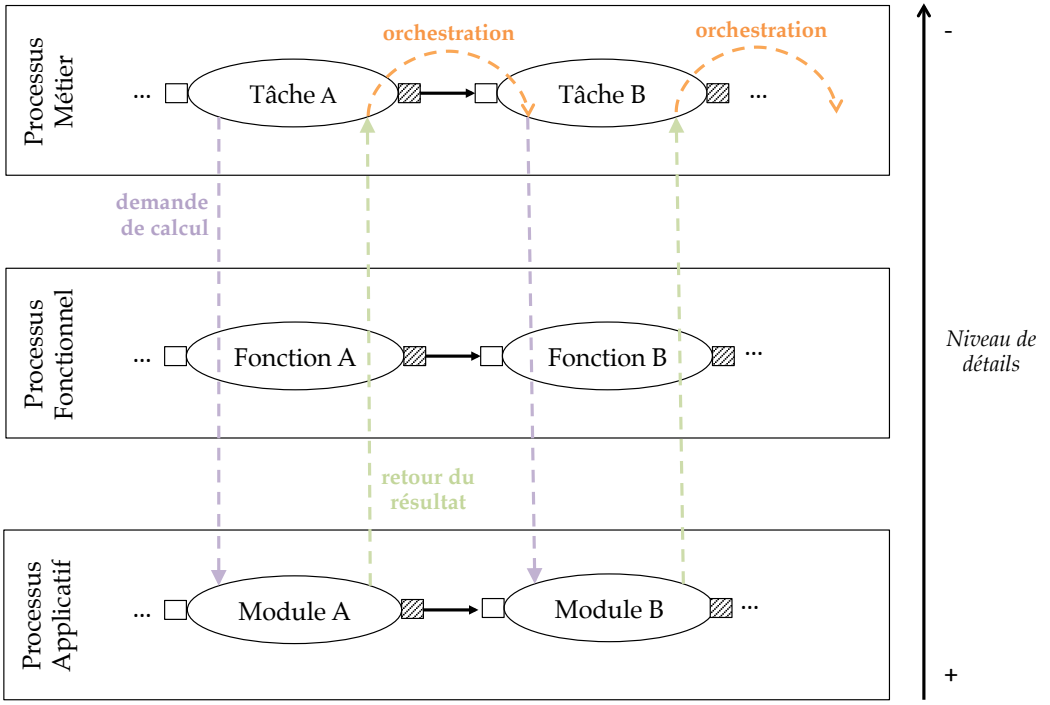

FIGURE 4.6 – Simulation dirigée par les processus métier

	Metier	Fonctionnel	Applicatif
BPMN	✓		
fUML	✓	✓	
OCL		✓	
MiniZinc			✓

TABLE 4.1 – Langages de l'IDM pour l'EA

4.2.3.2 Analyse de la structure

Tout comme l'analyse du comportement, l'analyse de la structure de l'architecture d'entreprise est étroitement lié à l'impératif de cohérence. Il est indispensable s'assurer de la cohérence entres les modèles des différentes vues avant d'initier une analyse structurelle qui soit pertinente pour les parties prenantes et en particulier pour l'architecte d'entreprise.

Notre contribution consiste à définir la manière dont les langages et techniques de l'IDM peuvent être utilisés pour mener une analyse structurelle des modèles d'entreprise. Le cadre d'EA que nous proposons permet d'acquérir une vision globale et cohérente de l'ensemble des artefacts qui la composent. La taille de plus en pls importantes des entreprises actuelle

fait que la complexité de l'entreprise en tant que système se retrouve dans les modèles d'architecture qui le représentent.

L'analyse automatisée de ces modèles facilite leur appréhension par l'acteur qui mène ces analyses (en l'occurrence l'architecte d'entreprise). Elle permet de révéler la structure de l'entreprise et le fonctionnement de cette structure. Un intérêt typique de l'analyse de la structure est la mesure de l'impact du changement [2005-99] sur l'architecture. L'analyse de l'impact du changement consiste à dévoiler les effets de bord d'un changement apporté à un élément de l'architecture.

Les langages de modélisation doivent donc permettre de représenter convenablement les différents composants de l'entreprise en plus d'offrir la possibilité d'analyser la structure des modèles créés dans l'objectif de mieux comprendre le système réel, qui est dans ce cas l'entreprise. Nous proposons donc de tirer profit des méthodes IDM telle la métamodélisation et de les associer à des langages capables d'exprimer et d'exécuter des contraintes et des requêtes sur les modèles tels que OCLinEcore ou QVT. Grâce à ce type de langage il est possible de :

1. modéliser des règles de structure supplémentaires qui précisent d'avantage méta-modèle d'architecture. Une fois que le modèle d'architecture est conforme à ce métamodèle, il est possible de vérifier qu'il respecte bien toutes les contraintes exprimées au niveau du métamodèle ;
2. analyser l'impact du changement pour évaluer par exemple l'impact de l'indisponibilité d'un module applicatif sur l'architecture globale. En mettant à profit les liens de cohérence de la vue intégration, il est alors possible de déterminer quels sont les processus métier ou fonctionnels touchés par cette défaillance applicative. De même, il est possible d'identifier les modules applicatifs existants pouvant participer à la réalisation d'un nouveau métier si ces processus fait intervenir des taches métier déjà implémentées dans le SI.

Chapitre 5

Implémentation et Validation

Sommaire

5.1 Modèles d'architecture d'entreprise pour le cas métier de la gestion d'une flotte de véhicule électrqiue

Nous éprouvons notre démarche au cas métier de la gestion d'une flotte de véhicules électriques. Nous construisons les modèles adéquats pour les vues métier, fonctionnelle et applicative en adoptant des langages exécutables. La cohérence est modélisée dans la vue intégration. L'architecture globale du cas métier est illustrée dans la 5.1.

5.1.1 Vue métier

Nous utilisons fUML comme langage exécutable pour modéliser cette vue. Le processus métier consiste à collecter les données relatives aux véhicules (électriques et thermiques) ainsi qu'aux tournées à effectuer, de calculer l'énergie nécessaire à chaque tournée et l'affectation véhicule/tournée avant de faire valider cette dernière par le manager de flotte. Nous modélisons ce processus métier sous forme de diagramme d'activité fUML et le simulons

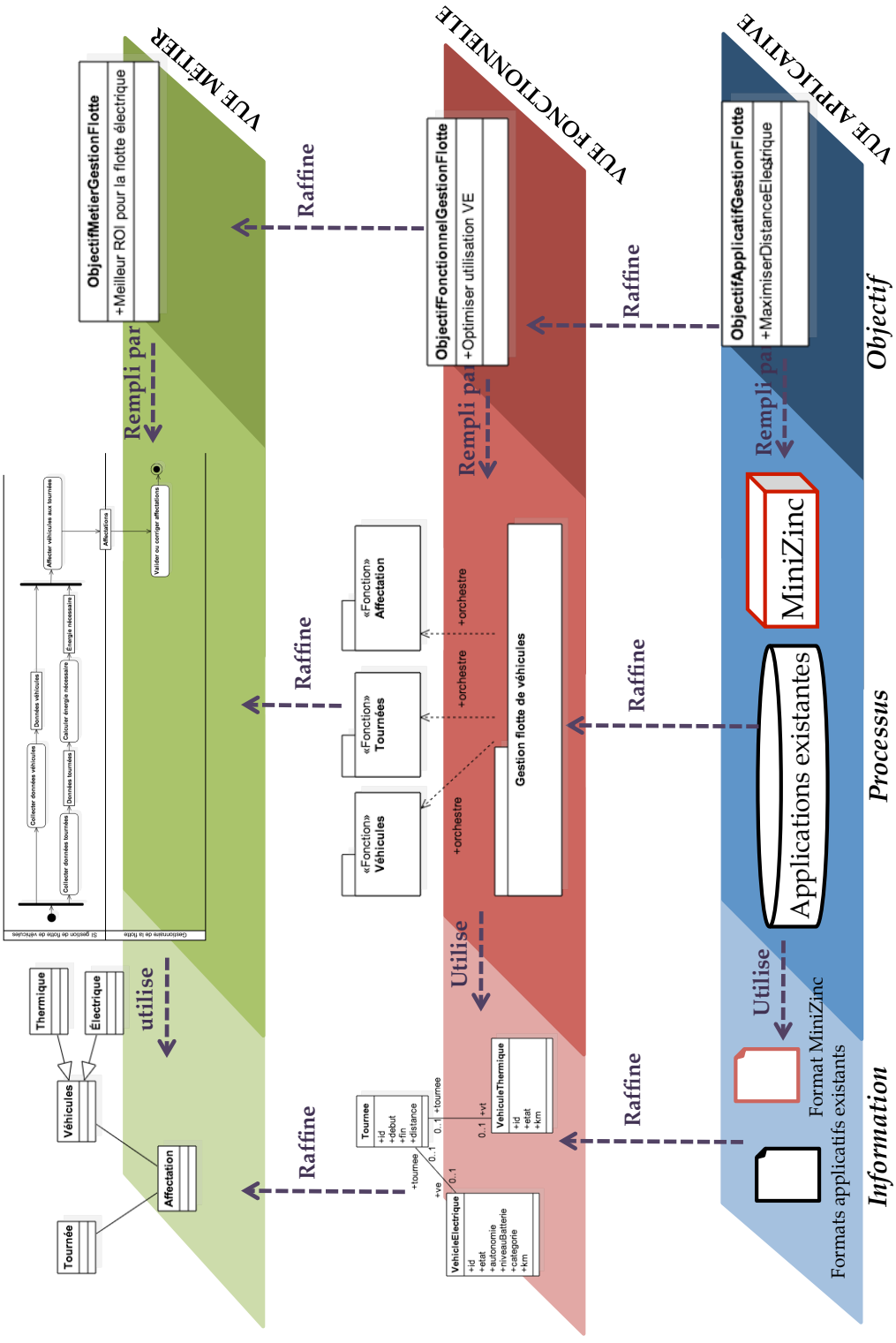

FIGURE 5.1 – Implémentation de l'approche proposé à la gestion de flotte de véhicules

en utilisant Papyrus. Selon notre cadre d'architecture, les modèles créés représentent donc l'aspect processus de la vue métier.

Pour l'aspect information, nous utilisons des diagrammes de classe UML pour représenter les concepts métier et leurs relations. Nous modélisons ainsi les concepts de Véhicule, Tournée et Affectation. Gérer une flotte de véhicules peut avoir plusieurs objectifs métier. Dans notre cas, obtenir le meilleur retour sur investissement suite à l'intégration de véhicules électriques dans la flotte de véhicules. Nous modélisons l'objectif métier sous la forme d'une classe UML.

Le choix des langages de modélisation et de l'outil de simulation est motivé par les pratiques du domaine. En effet, la Commission Électrique Internationale a adopté Enterprise Architect comme outil pour maintenir et distribuer le CIM [1][2012-43], un modèle d'information commun pour le domaine électrique [2].

5.1.2 Vue fonctionnelle

Nous modélisons l'aspect information sous la forme d'une diagramme de classe fUML. Ce modèle raffine les concepts métier en spécifiant leurs types. Dans la vue fonctionnelle, le concept d'allocation prend la forme d'une association entre les véhicules et les tournées. L'objectif fonctionnel est modélisé sous la forme d'une classe UML. L'objectif fonctionnel spécifie qu'il faut optimiser l'utilisation des véhicules électriques pour atteindre l'objectif métier qui est d'avoir un meilleur retour sur investissement.

Pour l'aspect processus, nous commençons par identifier trois blocs fonctionnels : un bloc pour la gestion de la flotte de véhicules (électriques et thermiques), un bloc pour la gestion des tournées, un bloc pour la gestion de l'affectation (voir figure 5.1. Ces blocs contiennent les fonctions qui raffinent les tâches du processus métier. Comme expliqué plus tôt dans la démarche, le fait de les rassembler dans des blocs selon les concepts métier augmente la modularité et l'évoluvilité de l'architecture. De plus, nous consacrons un bloc à la gestion des processus fonctionnels. Ce bloc est responsable de l'orchestration des fonctions du processus fonctionnel.

Les blocs fonctionnels offrent une vue plus détaillée des tâches métier. Il est possible de modéliser les différentes fonctions (calcul des tournées à partir de bon de travaux, calcul de l'énergie nécessaire à une tournée, etc.) à l'aide de diagrammes d'activité exécutables. Le choix du langage de modélisation dépend du cas d'application. Par exemple, nous modélisons l'affectation véhicule/tournée sous la forme de contraintes OCL : pour affecter un véhicule à une tournée, il faut que l'énergie nécessaire à celle-ci soit inférieure à l'autonomie de la batterie. Dans notre cas d'application, nous considérons qu'il n'est pas possible de recharge le véhicule pendant la tournée de l'agent.

1. Common Information Model
2. www.sparxsystems.com.au/press/articles/iec.html

Nous considérons un premier cas où il n'est pas possible de recharger la batterie au cours de la tournée. L'aspect *information* de la vue fonctionnelle prend la forme de données fonctionnelles modélisées par un diagramme de classes sur lequel s'appliquent les contraintes OCL (5.1). OCL est adapté aux diagrammes de classes. De plus, OCL est un langage standardisé et exécutable pour l'expression de contraintes. Il est possible de modéliser les autres algorithmes de traitement (calcul des tournées à partir de bon de travaux, calcul de l'énergie nécessaire à une tournée, etc.) à l'aide de diagrammes d'activité exécutables.

En utilisant le langage OCL, nous modélisons la fonction d'affectation sous la forme de deux contraintes et d'une requête. La première contrainte OCL signifie que si un véhicule électrique est affecté à une tournée alors l'énergie dont il dispose permet d'assurer la totalité de la tournée. La deuxième contrainte signifie que si aucun véhicule électrique n'est capable d'assurer une tournée donnée alors c'est un véhicule thermique qui lui est associé. Enfin, la requête calcule le nombre total de kilomètres électriques correspondant à la distance parcourue par les véhicules électrique après l'affectation. Cette requête permet d'évaluer l'utilisation des véhicules électriques dans l'optique d'atteindre l'objectif fonctionnel.

```
context Tournee
inv : self.ve.autonomie * self.ve.niveauBatterie > self.energieRequise

context Tournee
inv : self.ve <> undefined xor self.vt <> undefined

context Tournee :: elecKm() : int
body : (Tournee::allInstances() -> collect(t.ve <> undefined|t.distance)) -> sum()
```

5.1.3 Vue applicative

Pour les processus applicatifs, nous commençons par identifier les applications nécessaires à l'implantation des blocs fonctionnels. Dans notre cas, le patrimoine applicatif de l'entreprise dispose déjà d'applications pour la gestion de tournées (calcul de tournées optimisé à partir de bons de travaux) et la gestion de véhicules (administration, maintenance, etc.). Pour la fonction d'allocation, nous faisons le choix d'utiliser MiniZinc pour modéliser les contraintes au niveau applicatif 5.2.

```
constraint forall (i in Tournees) (
%affectation de vehicule electrique si l autonomie l autorise
constraint forall(i in  Tournees, j in VehiculesElec)
(tourneeVehicule[i]= identifiantVE[j] -> (autonomie[j]*niveauBatterie[j] > distanceTournee[i] /\
kmElec[i]=distanceTournee[i]));

% affectation d un vehicule thermique et dans ce cas kmElec est nul
constraint forall(i in  Tournees, k in VehiculesTherm)
(tourneeVehicule[i]= identifiantVT[k] -> kmElec[i]=0);

%maximiser le nombre de km de tounrnee fait par les vehicules electriques
solve maximize sum(i in Tournees)(kmElec[i]);
```

FIGURE 5.2 – Contraintes du module MiniZinc

MiniZinc est un langage de modélisation et de résolution de contraintes de niveau intermédiaire qui a pour vocation de devenir un langage de modélisation standard dans le domaine

de la programmation par contraintes. L'aspect information contient les formats de données nécessaires aux différentes applications. La figure 5.3 représente le fichier de données (le format .dzn) nécessaire à l'application MiniZinc pour calculer l'affectation.

```
nbreVE = 4;
identifiantVE = [1, 2, 3, 4];
nbreVT = 4;
identifiantVT = [11, 22, 33, 44];
autonomie = [1500, 1200, 1500, 1200];
niveauBatterie = [8, 6, 5, 2];
nbreTournees = 4;
distanceTournee = [1000, 700, 800, 1600];
```

FIGURE 5.3 – Fichier de données pour le module MiniZinc

L'objectif applicatif est modélisé sous la forme d'une classe fUML. Un véhicule électrique devient rentable par rapport à un véhicule thermique à partir d'un certain nombre de kilomètre parcouru. C'est pourquoi l'objectif fonctionnel qui est d'optimiser l'usage de la flotte électrique se traduit par la maximisation du nombre de kilomètres électriques, c'est à dire affecter aussi souvent que possible un véhicule électrique aux tournées. Ainsi, le module MiniZinc prend en compte cet objectif en résolvant les contraintes tout en maximisant la distance électrique.

5.1.4 Vue intégration

Nous modélisons la vue intégration en spécifiant les liens de cohérence intra-vue et et inter-vues. Les contraintes que ces liens doivent respecter sont spécifiés dans le métamodèle du cadre d'architecture présenté dans la démarche. La figure 5.4 offre une vue partielle des modèles d'intégration. Nous y modélisons à titre d'illustration les liens de cohérence intravue entre la fonction d'affection, ses inputs et output et son objectifs fonctionnel. Nous faisons de même pour le module d'optimisation Minizinc, ses inputs et outputs ainsi que l'objectif applicatif qu'il remplit. Le même principe d'intégration « horizontale », c'est à dire entre les aspects d'une même vue, est appliqué à tous les autres éléments de la vue métier, fonctionnelle et applicative. Nous mettons l'ensemble des modèles de la vue intégration en annexe de ce manuscrit.

De la même manière, nous modélisons les liens de cohérence inter-vues. Par exemple, le lien « inputs » exprime qu'un type de données est compatible avec la fonction qui l'utilise et qu'un format est compatible avec le module applicatif qui l'utilise en entrée. Pour garantir une bonne orchestration des processus, il faut que le « outputs » d'une tâche (respectivement une fonction, un module applicatif) soit compatible avec le « inputs » de la tâche suivante (respectivement une fonction, un module applicatif).

Les éventuelles transformation de modèles sont aussi identifiées dans la vue intégration. Pour le cas métier de la gestion de flotte de véhicules, nous utilisons une transformation de modèle pour générer les contraintes pour le module de calcul d'affectation MiniZinc de la

vue applicative à partir des contraintes OCL exprimées dans la fonction affectation de la vue fonctionnelle. La transformation est écrite dans le langage de transformation Acceleo (Annexe Acceleo). En plus de transformer les contraintes décrite dans l'aspect métier, cette transformation de modèle permet aussi de transformer les instances des types fonctionnels en instances dans le format dzn utilisé par le module MiniZinc comme l'illustre la figure 5.4.

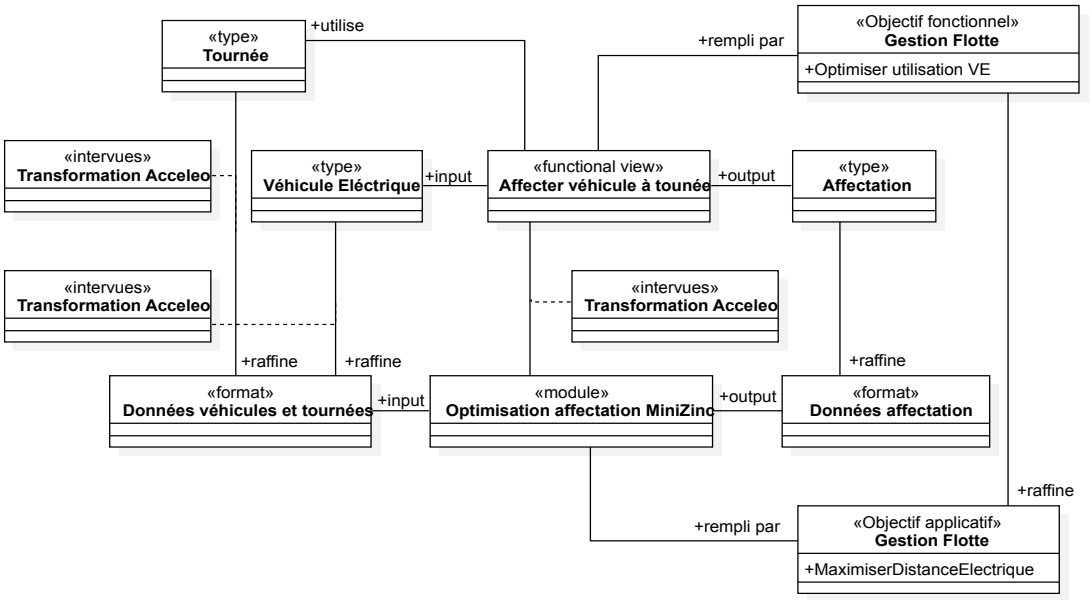

FIGURE 5.4 – Une vue partielle des modèles d'intégration de la vue fonctionnelle et de la vue applicative

5.2 Analyse par simulation des modèles d'architecture

Analyse du comportement des modèles d'architecture se fait directement dans les langages de modélisation. Comme nous l'expliquons dans notre démarche, la simulation de l'ensemble de l'architecture est pilotée par l'exécution du processus métier. La simulation du processus métier se traduit par l'exécution du diagramme d'activité fUML. Nous utilisons Papyrus comme outil de modélisation et de simulation du processus métier de gestion de flotte de véhicule. En effet, Papyrus implémente la sémantique d'exécution fUML telle qu'elle est spécifiée par l'OMG.

Notre démarche préconise de modéliser les détails des taches dans les vues inférieure afin de respecter le niveau d'abstraction requis par chaque point de vue et ainsi de ne pas altérer

la compréhension des parties prenantes de la vue qui leur destinée. Par exemple, l'analyse métier n'aura ainsi pas à discuter du détails des applications implémentant les taches métier avec l'architecte applicatif.

Papyrus offre la possibilité d'étendre la sémantique d'exécution de fUML à travers les « Opaque Actions ». Celles-ci permettent d'invoquer des modules d'applications extérieures au moment de l'exécution du diagramme d'activité fUML. Un stagiaire a développé cette extension en collaboration avec l'équipe Triskel. L'extension permet d'invoquer directement le module MiniZinc pour optimiser l'affectation des véhicules aux tournées.

La simulation prend la forme d'une animation de diagramme. La figure 5.5 est une capture d'écran montrant une simulation en court d'exécution. Papyrus permet aussi de mettre des « breakpoints » sur certaines activités et de paramétrer le pas de temps pour contrôler le déroulement du processus.

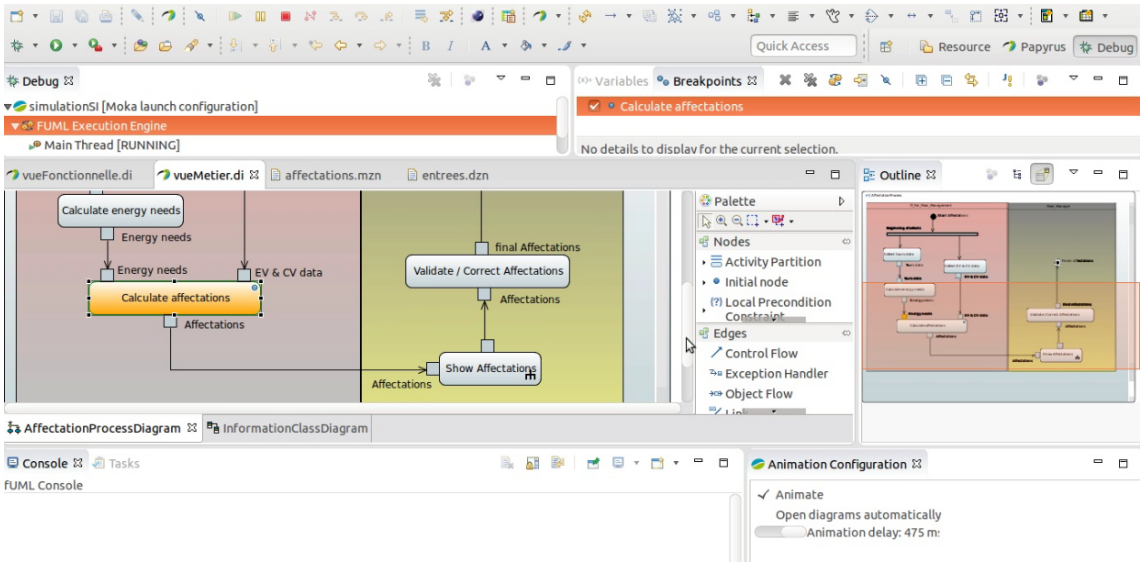

FIGURE 5.5 – Simulation de l'architecture sous Papyrus

La simulation retourne comme résultat les affectations des véhicles aux tournées. Ce résultat est affiché dans la console comme l'illustre la figure 5.6. La simulation a pour objectif de voir si l'utilisation des véhicules électrique est rentable en comparant la distance parcourue par le véhicule électrique et la distance minimale permettant de le rentabiliser. Dans ce cas, il est par exemple envisageable de reconfigurer les tournées de manière à ce que plus de véhicules électriques soient affectés. En effet, notre démarche a pour but l'analyse fonctionnelle. La validation s'appuie sur les indicateurs dérivés de l'aspect objectif et sur l'avis des experts. Des analyses statistiques peuvent être conduites mais elles ne rentrent pas dans le périmètre de nos travaux.

```
include "alldifferent.mzn";

%ensemble des vehicules therminques
int : nbreVehiculeThermique;
```

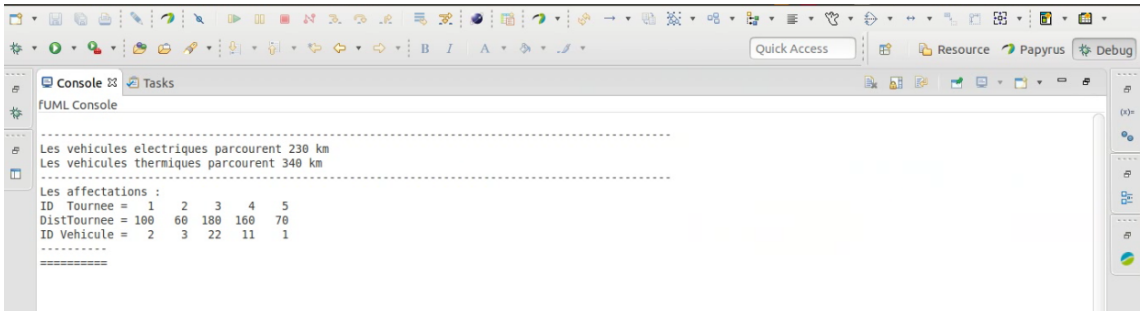

FIGURE 5.6 – Résultat retournée par la simulation l'architecture sous Papyrus

```
set of int: VehiculeThermiques = 1..nbreVehiculeThermique;
array[VehiculeThermiques] of int : identifiantVT;

%L'ensemble de tous les vehicules
set of int : lesVehicules;

% variables : quel vehicule affecter a quelle tournee
array[Tournees] of var lesVehicules : tourneeVehicule;
array[Tournees] of var 0..max(distanceTournee) : kmElec;

%Les contraintes minizinc sur les variables

%nbre de tournees
int : nbreTournee;
set of int : Tournees = 1..nbreTournee;
%distance de chaque tournee
array[Tournees] of int : distanceTournee;

%ensemble des vehicules electriques
int : nbreVehiculeElectrique;
set of int : VehiculeElectriques = 1..nbreVehiculeElectrique;
array[VehiculeElectriques] of int : identifiantVE;
%autonomie de chaque VE
array[VehiculeElectriques] of int : autonomie;
%niveau de batterie de chaque VE
array[VehiculeElectriques] of int : niveauBatterie;


% affectation d'un vehicule thermique et dans ce cas kmElec est nul
constraint forall(t in Tournees)(
  forall(vt in VehiculeThermiques)(
    tourneeVehicule[t] = identifiantVT[vt] -> kmElec[t] = 0
  )
);


% Un vehicule par tournee
constraint alldifferent(tourneeVehicule);
% L'autonomie et le niveau de batterie du VE sont suffisants pour effectuer la tournee
constraint forall(t in Tournees)(
  forall(ve in VehiculeElectriques)(
    (tourneeVehicule[t] = identifiantVE[ve]) ->
    ( autonomie[ve] * niveauBatterie[ve] > distanceTournee[t]
     /\ kmElec[t] = distanceTournee[t])
  )
);
%maximiser le nombre de km de tounrnee fait par les vehicules electriques
solve maximize sum([kmElec[t] | t in Tournees]);

int: dist_tournees = sum(t in Tournees)(distanceTournee[t])
int: dist_electrique = sum(t in Tournees)(kmElec[t])
int: dist_thermique = dist_tournees - dist_thermique

%affichage
output
[show("\nLes vehicules electriques parcourent "),
 show_int(dist_electrique),
 show(" km")] ++
[show("\nLes vehicules thermiques parcourent "),
 show_int(dist_thermique),
```

```
 show(" km")] ++
[show("\nLes affectations :")]++
[show("\nID  Tournee = ")]++[show_int(3, idTournee) ++ "  " | idTournee in Tournees, ]++[show("\n")]++
[show("\nDistTournee = ")]++[show_int(3, distanceTournee[t]) ++ "  " | t in Tournees, ]++[show("\n")]++
[show("\nID Vehicule = ")]++[show_int(3, tourneeVehicule[t]) ++ "  " | t in Tournees]
;
```

Annexe A

Exemple d'annexe

A.1 Exemple d'annexe

.

Acronymes

3G Troisième Génération. 7

ADM *The Architecture Development Method*. 20
ATL *Atlas Transformation Language*. 40, 42

CIM *Computation Independant Model*. 45
CPL Courant Porteur de Ligne. 7, 10

EA *Enterprise Architecture*. 11, 12
EDF Électricite De France. 10, 11
ERDF Électricite Réseau de Distribution France. 8, 10
ETP *European Technology Platform SmartGrids*. 4

IBM *International Business Machine*s. 34
IDM Ingénierie Dirigée par les Modèles. 33–36, 40, 41, 43, 45–47

MDA *Model Driven Architecture*. 34, 45, 46
MOF *Meta-Object Facility*. 34, 39, 42, 43

OCL *Object Constraint Language*. 42, 48
OMG *Object Management Group*. 34, 37, 42, 46

PIM *Platform Independant Model*. 45
PSM *Platform Specific Model*. 45

QVT *Query View Transformation*. 42

RM-ODP *Reference Model of Open Distributed Processing*. 19, 21–23

SGAM *Smart Grid Reference Architecture*. 19, 22, 23
SI Système d'Information. 7–12

TIC Technologies de l'Information et de la Communication. 4, 7, 9–11
TOGAF *The Open Group Architectural Framewor*k. 19–21, 23, 46

Bibliographie

[2013-1] Nicolas Le Dévédec and Fany Guis. L'humain augmenté, un enjeu social. *SociologieS*, 2013. Association internationales des sociologues de langue française (AISLF), 2013. 3

[2] European technology platform for the electricity networks of the future. Définition des Smart grids. `http://www.smartgrids.eu`. 4

[2014-3] L'énergie en question EDF. La panne d'électricité à Détroit, symbole de la vétusté du réseau électrique américain. 2014, 2014. `https://www.lenergieenquestions.fr/la-panne-delectricite-a-detroit-symbole-de-la-vetuste-du-reseau-electr` 4

[2005-4] G Andersson, P Donalek, R Farmer, N Hatziargyriou, I Kamwa, P Kundur, N Martins, J Paserba, P Pourbeik, J Sanchez-Gasca, et al. Causes of the 2003 major grid blackouts in North America and Europe, and recommended means to improve system dynamic performance. *Power Systems, IEEE Transactions on*, 20(4) :1922–1928, 2005. IEEE, 2005. 4

[2014-5] JORDAN WIRFS-BROCK Inside Energy. Power Outages On The Rise Across The U.S. 2014, 2014. `http://insideenergy.org/2014/08/18/power-outages-on-the-rise-across-the-u-s/`. 4

[6] US department of enrergy. Introduction to Smart grids. `energy.gov/oe/downloads/smart-grid-introduction-0`. 4

[7] Smart Grids CRE. Définition des Smart grids. `http://www.smartgrids-cre.fr/index.php?p=definition-smart-grids`. 4, 8

[2006-8] Jean-Marie FAVRE, Jacky ESTUBLIER, and Mireille BLAY-FORNARINO. L'ingénierie dirigée par les modèles. Au-delà du MDA (Traité IC2, série Informatique et Systèmes d'Information), 2006. 8, 35, 36

[2014-9] Peter Palensky, Edmund Widl, and Atiyah Elsheikh. Simulating cyber-physical energy systems : challenges, tools and methods. *Systems, Man, and Cybernetics : Systems, IEEE Transactions on*, 44(3) :318–326, 2014. IEEE, 2014. 10

[2012-10] Jeremy Rifkin. *La troisième révolution industrielle : comment le pouvoir latéral va transformer l'énergie, l'économie et le monde.* Éditions Les liens qui libèrent, 2012. 10

[1997-11] John A Zachman. Enterprise architecture : The issue of the century. *Database Programming and Design*, 10(3) :44–53, 1997, 1997. 11, 14, 17, 29

[2014-12] Rachida Seghiri, Frédéric Boulanger, Claire Lecocq, and Vincent Godefroy. Simulation des Systèmes d'Information des Smart Grids. 2014, 2014. 11

[2006-13] Jeanne W Ross, Peter Weill, and David Robertson. *Enterprise architecture as strategy : Creating a foundation for business execution.* Harvard Business Press, 2006. 11, 15

[1987-14] John Zachman et al. A framework for information systems architecture. *IBM systems journal*, 26(3) :276–292, 1987. IBM, 1987. 11, 15, 19, 20

[2010-15] Tomaž Čater and Danijel Pučko. Factors of effective strategy implementation : Empirical evidence from Slovenian business practice. *Journal for east european Management Studies*, pages 207–236, 2010. JSTOR, 2010. 11

[2008-16] Leon Kappelman, Tom McGinnis, Alex Pettite, and Anna Sidorova. Enterprise architecture : Charting the territory for academic research. *AMCIS 2008 Proceedings*, page 162, 2008, 2008. 11, 15

[2010-17] Antonello Monti and Ferdinanda Ponci. Power grids of the future : Why smart means complex. pages 7–11. IEEE, 2010, ISBN : 1424459826. 12

[2004-18] Andrei Borshchev and Alexei Filippov. From system dynamics and discrete event to practical agent based modeling : reasons, techniques, tools. In *Proceedings of the 22nd international conference of the system dynamics society*, volume 22, 2004. 12

[2008-19] Sabine Buckl, Florian Matthes, Wolfgang Renz, Christian M Schweda, and Jan Sudeikat. Towards Simulation-supported Enterprise Architecture Management. In *MobIS*, pages 131–145. Citeseer, 2008. 12, 30

[2013-20] Vinay Kulkarni, Suman Roychoudhury, Sagar Sunkle, Tony Clark, and Balbir Barn. Modelling and Enterprises-The Past, the Present and the Future. In *MODELSWARD*, pages 95–100, 2013. 12, 25

[1995-21] Robert Reix, Bernard Fallery, Michel Kalika, and Frantz Rowe. *Systèmes d'information et management des organisations.* Vuibert, 1995. 14

[2012-22] Scott A Bernard. *An introduction to enterprise architecture.* AuthorHouse, 2012. 14, 15

[2006-23] Robert Winter and Ronny Fischer. Essential layers, artifacts, and dependencies of enterprise architecture. In *Enterprise Distributed Object Computing Conference Workshops, 2006. EDOCW'06. 10th IEEE International*, pages 30–30. IEEE, 2006. 15, 23

[2012-24] James Lapalme. Three schools of thought on enterprise architecture. *IT professional*, (6) :37–43, 2012. IEEE, 2012. 15, 16, 17

[1993-25] Steven H Spewak and Steven C Hill. *Enterprise architecture planning : developing a blueprint for data, applications and technology.* QED Information Sciences, Inc., 1993. 15

[2005-26] Parthasarathy Ranganathan and Norman Jouppi. Enterprise IT trends and implications for architecture research. In *High-Performance Computer Architecture, 2005. HPCA-11. 11th International Symposium on*, pages 253–256. IEEE, 2005. 15, 16

[2012-27] Jan Mentz, Paula Kotzé, and Alta van der Merwe. A comparison of practitioner and researcher definitions of enterprise architecture using an interpretation method. *Advances in Enterprise Information Systems II*, pages 11–26, 2012. CRC Press, 2012. 16

[28] Jeanne Ross on Enterprise Architecture. Poscast from the MIT CISR. `https://www.youtube.com/watch?v=feI6_-v10Dk`. 17

[2013-29] Marc Lankhorst. {Enterprise Architecture at Work : Modelling, Communication and Analysis (The Enterprise Engineering Series)}. 2013. Springer, 2013. 18, 20, 24, 25, 26, 28, 29, 30

[2007-30] Hanifa Shah and Mohamed El Kourdi. Frameworks for enterprise architecture. *It Professional*, 9(5) :36–41, 2007. IEEE, 2007. 18

[2004-31] Maarten WA Steen, David H Akehurst, HWLT Doest, and Marc M Lankhorst. Supporting viewpoint-oriented enterprise architecture. In *Enterprise Distributed Object Computing Conference, 2004. EDOC 2004. Proceedings. Eighth IEEE International*, pages 201–211. IEEE, 2004. 18, 19

[1999-32] Frank J Armour, Stephen H Kaisler, and Simon Y Liu. A big-picture look at enterprise architectures. *IT professional*, 1(1) :35–42, 1999, ISSN : 1520-9202, 1999. 18

[1979-33] Geoff P Mullery. CORE-a method for controlled requirement specification. In *Proceedings of the 4th international conference on Software engineering*, pages 126–135. IEEE Press, 1979. 18

[1992-34] Anthony Finkelstein, Jeff Kramer, Bashar Nuseibeh, Ludwik Finkelstein, and Michael Goedicke. Viewpoints : A framework for integrating multiple perspectives in system development. *International Journal of Software Engineering and Knowledge Engineering*, 2(01) :31–57, 1992. World Scientific, 1992. 18

[1996-35] Gerald Kotonya and Ian Sommerville. Requirements engineering with viewpoints. *Software Engineering Journal*, 11(1) :5–18, 1996. IET, 1996. 18

[1994-36] Bashar Ahmad Nuseibeh. *A multi-perspective framework for method integration.* PhD thesis, Imperial College, 1994. 18

[1993-37] Scott Douglas Meyers. *Representing software systems in multiple-view development environments.* PhD thesis, Brown University, 1993. 18

[2000-38] Rich Hilliard. Ieee-std-1471-2000 recommended practice for architectural description of software-intensive systems. *IEEE, http ://standards. ieee. org*, 12 :16–20, 2000, 2000. 18, 19

[1995-39] Kerry Raymond. Reference model of open distributed processing (RM-ODP) : Introduction. In *Open distributed processing*, pages 3–14. Springer, 1995. 19

[2003-40] Anneke G Kleppe, Jos B Warmer, and Wim Bast. *MDA explained : the model driven architecture : practice and promise*. Addison-Wesley Professional, 2003. 19, 37

[2008-41] Robert Winter and Joachim Schelp. Enterprise architecture governance : the need for a business-to-IT approach. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 548–552. ACM, 2008. 19, 20

[2009-42] TOGAF Version 9. The Open Group Architecture Framework (TOGAF). *The Open Group*, 1, 2009, 2009. `http://www.opengroup.org/togaf`. 21

[2012-43] Mathias Uslar, Michael Specht, Christian Dänekas, Jörn Trefke, Sebastian Rohjans, José M González, Christine Rosinger, and Robert Bleiker. *Standardization in Smart Grids : Introduction to IT-Related Methodologies, Architectures and Standards*. Springer Science & Business Media, 2012. 22, 65

[1992-44] David Nadler, Marc S Gerstein, and Robert B Shaw. *Organizational architecture : Designs for changing organizations*, volume 192. Jossey-Bass Inc Pub, 1992. 23

[2013-45] Vinay Kulkarni, Suman Roychoudhury, Sagar Sunkle, Tony Clark, and Balbir Barn. Modelling and Enterprises-The Past, the Present and the Future. pages 95–100, 2013. 24, 46

[2014-46] Therese Clark, Vaishali Kulkarni, Balbir Barn, Robert France, Ulrich Frank, and Dan Turk. Towards the Model Driven Organization. pages 4817–4826. IEEE, 2014. 24, 45, 46

[2001-47] Jean Bézivin and Olivier Gerbé. Towards a precise definition of the OMG/MDA framework. pages 273–280. IEEE, 2001, ISBN : 076951426X. 24

[2013-48] Sagar Sunkle, Vinay Kulkarni, and Suman Roychoudhury. Analyzing enterprise models using enterprise architecture-based ontology. In *Model-Driven Engineering Languages and Systems*, pages 622–638. Springer, 2013, ISBN : 3642415326. 24, 29

[2011-49] Tony Clark, Balbir S Barn, and Samia Oussena. Leap : a precise lightweight framework for enterprise architecture. In *Proceedings of the 4th India Software Engineering Conference*, pages 85–94. ACM, 2011. 25, 46

[2009-50] Sabine Buckl, Florian Matthes, and Christian M Schweda. Classifying enterprise architecture analysis approaches. In *Enterprise Interoperability*, pages 66–79. Springer, 2009. 25, 26, 27, 28, 31

[1974-51] Francisco G Varela, Humberto R Maturana, and Ricardo Uribe. Autopoiesis : the organization of living systems, its characterization and a model. *Biosystems*, 5(4) :187–196, 1974. Elsevier, 1974. 28

[2014-52] Roland Smook. Executable enterprise architecture models : enabling business analytics using the Monte Carlo method. 2014. University of Twente, 2014. 28

[2014-53] Alexandre Métrailler and Thibault Estier. EVOLIS Framework : A Method to Study Information Systems Evolution Records. pages 3798–3807. IEEE, 2014. 28

[2005-54] Stephen H Kaisler, Frank Armour, and Michael Valivullah. Enterprise architecting : Critical problems. pages 224b–224b. IEEE, 2005, ISBN : 0769522688. 28, 29, 55

[2013-55] Balbir S Barn, Tony Clark, and Martin Loomes. Enterprise architecture coherence and the model driven enterprise : is simulation the answer or are we flying kites ? In *Proceedings of the 6th India Software Engineering Conference*, pages 97–102. ACM, 2013. 29, 53

[2013-56] Hugo Brunelière, Jordi Cabot, Stéphane Drapeau, Flavien Somda, William Piers, Juan David Villa Calle, and Jean-Christophe Lafaurie. Un support IDM pour l'architecture d'entreprise dans un contexte industriel : l'exemple du framework TEAP. *Génie logiciel*, (107) :33–38, 2013, 2013. 29, 45

[1975-57] Robert E Shannon. Systems simulation. 1975. Prentice-Hall, 1975. 30, 53

[2015-58] Laura Manzur, Jorge Mario Ulloa, Mario Sánchez, and Jorge Villalobos. xArchiMate : Enterprise Architecture simulation, experimentation and analysis. *Simulation*, 91(3) :276–301, 2015. SAGE Publications, 2015. 30, 31, 47, 53

[2013-59] Kenneth C Hoffman, Christopher G Glazner, William J Bunting, Leonard A Wojcik, and Anne Cady. *Enterprise Dynamics Sourcebook*. CRC Press, 2013. 31

[2011-60] Christopher G Glazner. Enterprise transformation using a simulation of enterprise architecture. *Journal of Enterprise Transformation*, 1(3) :231–260, 2011. Taylor & Francis, 2011. 31, 53

[2011-61] Marie Ludwig, Nicolas Farcet, Jean-Philippe Babau, and Joël Champeau. Organizational configurations in executable Enterprise Architecture models. In *poster session of Complex Systems Design and Management 2011*, 2011. 31

[2004-62] Jack Greenfield, Keith Short, Steve Cook, Stuart Kent, and John Crupi. Software factories : assembling applications with patterns, models, frameworks, and tools. 2004. Wiley Pub., 2004. 33, 34

[2001-63] Jean Bézivin and Olivier Gerbé. Towards a precise definition of the OMG/MDA framework. In *Automated Software Engineering, 2001.(ASE 2001). Proceedings. 16th Annual International Conference on*, pages 273–280. IEEE, 2001. 34

[2002-64] Stuart Kent. Model driven engineering. In *Integrated formal methods*, pages 286–298. Springer, 2002. 34

[2002-65] Juan De Lara and Hans Vangheluwe. Using Meta-Modelling and Graph Grammars to Process GPSS Models. In *ESM*, pages 100–107, 2002. 34

[2004-66] Jean Bézivin. In search of a basic principle for model driven engineering. *Novatica Journal, Special Issue*, 5(2) :21–24, 2004. Citeseer, 2004. 34, 35, 36

[2004-67] Grady Booch, Alan W Brown, Sridhar Iyengar, James Rumbaugh, and Bran Selic. An MDA manifesto. *Business Process Trends/MDA Journal*, 2004, 2004. 34

[2004-68] Jean Bézivin, Mireille Blay, Mokrane Bouzhegoub, Jacky Estublier, Jean-Marie Favre, Sébastien Gérard, and Jean Marc Jézéquel. Rapport de Synthèse de l ?AS

CNRS sur le MDA (Model Driven Architecture). *CNRS, novembre*, 2004, 2004. 34, 35

[2005-69] Jean Bézivin. On the unification power of models. *Software & Systems Modeling*, 4(2) :171–188, 2005. Springer, 2005. 34, 41

[1967-70] Marvin L Minsky. *Computation : finite and infinite machines.* Prentice-Hall, Inc., 1967. 34

[2003-71] Ed Seidewitz. What models mean. *IEEE software*, (5) :26–32, 2003. IEEE, 2003. 34, 35

[2003-72] Colin Atkinson and Thomas Kühne. Model-driven development : a metamodeling foundation. *Software, IEEE*, 20(5) :36–41, 2003. IEEE, 2003. 35

[2004-73] Jean-Marie Favre. Towards a basic theory to model model driven engineering. In *3rd Workshop in Software Model Engineering, WiSME*, pages 262–271. Citeseer, 2004. 36

[2011-74] QVT OMG. Meta Object Facility (MOF) 2.0 Query/View/Transformation, v1. 1, 2011. 37, 42

[2006-75] Tom Mens and Pieter Van Gorp. A taxonomy of model transformation. *Electronic Notes in Theoretical Computer Science*, 152 :125–142, 2006. Elsevier, 2006. 37

[2005-76] Laurence Tratt. Model transformations and tool integration. *Software & Systems Modeling*, 4(2) :112–122, 2005. Springer, 2005. 37, 39

[2000-77] Krzysztof Czarnecki and Ulrich W Eisenecker. Intentional programming. *Generative Programming. Methods, Tools, and Applications*, 2000, 2000. 38

[2006-78] Jean Bézivin, Salim Bouzitouna, Marcos Didonet Del Fabro, Marie-Pierre Gervais, Frédéric Jouault, Dimitrios Kolovos, Ivan Kurtev, and Richard F Paige. A canonical scheme for model composition. In *Model Driven Architecture–Foundations and Applications*, pages 346–360. Springer, 2006. 39

[2008-79] Franck Fleurey, Benoit Baudry, Robert France, and Sudipto Ghosh. A generic approach for automatic model composition. In *Models in Software Engineering*, pages 7–15. Springer, 2008. 39

[2007-80] Marcos Didonet Del Fabro and Patrick Valduriez. Semi-automatic model integration using matching transformations and weaving models. In *Proceedings of the 2007 ACM symposium on Applied computing*, pages 963–970. ACM, 2007. 39, 40

[2011-81] Eugene Syriani. *A multi-paradigm foundation for model transformation language engineering.* PhD thesis, McGill University, 2011. 40, 41

[2006-82] Frédéric Jouault and Ivan Kurtev. Transforming models with ATL. In *satellite events at the MoDELS 2005 Conference*, pages 128–138. Springer, 2006. 40, 42

[2010-83] Matthias Biehl, Chen DeJiu, and Martin Törngren. Integrating safety analysis into the model-based development toolchain of automotive embedded systems. In *ACM Sigplan Notices*, volume 45, pages 125–132. ACM, 2010. 40

[1985-84] Alfred V Aho, Ravi Sethi, and Jeffry D Ullman. Compilers Principles, Techniques, and Tools Addison-Wesley, 1986. *QA76*, 76 :C65A37, 1985, 1985. 41

[2006-85] Krzysztof Czarnecki and Simon Helsen. Feature-based survey of model transformation approaches. *IBM Systems Journal*, 45(3) :621–645, 2006. IBM, 2006. 41

[2011-86] Xavier Blanc and Olivier Salvatori. *MDA en action : Ingénierie logicielle guidée par les modèles*. Editions Eyrolles, 2011. 41

[2006-87] Jean Bézivin, Fabian Büttner, Martin Gogolla, Frédéric Jouault, Ivan Kurtev, and Arne Lindow. Model transformations ? transformation models ! In *Model driven engineering languages and systems*, pages 440–453. Springer, 2006. 41

[2008-88] Ivan Kurtev. State of the art of QVT : A model transformation language standard. In *Applications of graph transformations with industrial relevance*, pages 377–393. Springer, 2008. 42

[2007-89] Robert France and Bernhard Rumpe. Model-driven development of complex software : A research roadmap. In *2007 Future of Software Engineering*, pages 37–54. IEEE Computer Society, 2007. 45

[2003-90] David S Frankel, Paul Harmon, Jishnu Mukerji, James Odell, Martin Owen, Pete Rivitt, Mike Rosen, and Richard Mark Soley. The Zachman framework and the OMG's model driven architecture. *Business Process Trends*, 14, 2003, 2003. 45

[2013-91] Hugo Bruneliere, Jordi Cabot, Stéphane Drapeau, Flavien Somda, William Piers, Juan David Villa Calle, and Jean-Christophe Lafaurie. MDE Support for Enterprise Architecture in an Industrial Context : the TEAP Framework Experience. In *TowArds the Model DrIveN Organization (AMINO 2013) workshop-a MODELS 2013 Satellite Event*, 2013. 46, 47

[2012-92] Jean-Marc Jézéquel, Benoit Combemale, and Didier Vojtisek. *Ingénierie Dirigée par les Modèles : des concepts à la pratique...* Ellipses, 2012. 47

[2007-93] Anneke G Kleppe. A language description is more than a metamodel. 2007. megaplanet. org, 2007. 47

[2006-94] Henry Chesbrough and Jim Spohrer. A research manifesto for services science. *Communications of the ACM*, 49(7) :35–40, 2006. ACM, 2006. 47, 59

[2007-95] Nicholas Nethercote, Peter J Stuckey, Ralph Becket, Sebastian Brand, Gregory J Duck, and Guido Tack. Minizinc : Towards a standard CP modelling language. In *Principles and Practice of Constraint Programming–CP 2007*, pages 529–543. Springer, 2007. 48

[2007-96] Stephan Kurpjuweit and Robert Winter. Viewpoint-based Meta Model Engineering. In *EMISA*, volume 143, page 2007, 2007. 51, 53

[2008-97] David Chen, Guy Doumeingts, and François Vernadat. Architectures for enterprise integration and interoperability : Past, present and future. *Computers in industry*, 59(7) :647–659, 2008. Elsevier, 2008. 53

[1990-98] Herbert A Simon. Prediction and prescription in systems modeling. *Operations Research*, 38(1) :7–14, 1990. INFORMS, 1990. 59

[2005-99] Frank S de Boer, Marcello M Bonsangue, LPJ Groenewegen, AW Stam, L van der Torre, et al. Change impact analysis of enterprise architectures. In *Information Reuse and Integration, Conf, 2005. IRI-2005 IEEE International Conference on.*, pages 177–181. IEEE, 2005. 61