Architecture d'entreprise : alignement des cartographies métiers et applicatives du système d'information

Jonathan Pepin

UNIVERSITE BRETAGNE LOIRE

UNIVERSITÉ DE NANTES

Thèse de Doctorat

Jonathan PEPIN

Mémoire présenté en vue de l'obtention du
grade de Docteur de l'Université de Nantes
sous le sceau de l'Université Bretagne Loire

École doctorale : Sciences et technologies de l'information, et mathématiques

Discipline : Informatique et applications, section CNU 27
Unité de recherche : Laboratoire d'informatique de Nantes-Atlantique (LINA)

Soutenue le 5 décembre 2016

Architecture d'entreprise : alignement des cartographies métiers et applicatives du système d'information

JURY

Présidente :	**M^me^ Marianne HUCHARD**, Professeur, Université de Montpellier - LIRMM
Rapporteurs :	**M^me^ Florence SEDES**, Professeure, Université de Toulouse - IRIT
	M. Xavier BLANC, Professeur, Université de Bordeaux - LABRI
Examinateurs :	**M^me^ Anne ETIEN**, Maître de Conférences, Université de Lille - INRIA
	M. Jean-Claude ROYER, Professeur, École des Mines de Nantes - LINA
	M. Pascal ANDRÉ, Maître de conférences, Université de Nantes - LINA
Invités :	**M. Erwan BRETON**, Directeur technique, Mia-Software
	M. Laurent PICHIERRI, Ingénieur étude et recherche, Mia-Software
Directeur de thèse :	**M. Christian ATTIOGBÉ**, Professeur, Université de Nantes - LINA

Remerciements

Cette thèse a été réalisée dans le cadre d'une collaboration entre le laboratoire d'informatique de l'**Université de Nantes** (LINA) et le département Recherche et Développement **Mia-Software**, filiale de l'entreprise Sodifrance. Je remercie l'entreprise Sodifrance de m'avoir fait confiance et d'avoir financé ces trois années et demie de recherche.

Au **Lina**. J'adresse mon premier remerciement à Pascal André qui est à l'initiative de ce projet de thèse. Il m'a proposé de réaliser un doctorat après m'avoir encadré durant les stages de mes deux années de Master *méthodes informatiques appliquées à la gestion des entreprises* (MIAGE). Puis, Pascal m'a suivi assidument durant ces trois années au cours de nos rendez-vous hebdomadaire. J'ai beaucoup appris à ses côtés tant sur la méthodologie de recherche, la vulgarisation et encore la rigueur de rédaction. Nos longues digressions durant nos conversations et réflexions communes ont été particulièrement enrichissantes tant du point de vue intellectuel que humain. Je te remercie de m'avoir motivé et soutenu de façon amicale même dans les moments personnels délicats.

Je remercie également mon directeur de thèse, Christian Attiogbé. Christian, Pascal et moi avons formé un trio au sein de l'équipe AeLos. Les corrections précises des publications en anglais, souvent très tard la nuit, m'ont permis de progresser de façon continue et de participer ainsi à des conférences de qualité.

Je remercie le reste de l'équipe d'AeLos pour leur bonne humeur et les conseils avisés pendant les réunions ou entre deux portes. Notamment Benoît pour ses nombreux encouragements. Sans oublier tous les doctorants que j'ai pu croiser au moment du goûter ou avec qui j'ai partagé un bureau, je pense notamment à Matthieu Perrin pour sa bienveillance, mais aussi à David, Nicolas, Amine, Georges, Matula et Pauline.

Au cours de ces années passées à l'Université de Nantes, nombreux de mes anciens professeurs du LINA sont devenus des collègues, j'ai une pensée particulière à Marie Catalo, j'ai aussi une pensée pour mes deux anciens camarades de Master, Oussama et Florian.

À **Sodifrance**. Je tiens à adresser mes remerciements à Erwan Breton, docteur en informatique, directeur de l'innovation à Sodifrance, qui a obtenu l'accord de financement de cette thèse auprès de la direction et m'a soutenu avec confiance dans le travail que j'ai mené. Le sujet de la thèse est issu de sa vision sur les besoins de nos entreprises clientes dans le but de moderniser leur système d'information.

La recommandation auprès d'Erwan a été permise par la collaboration de Jean-Pierre Morgant que je remercie chaleureusement pour les deux années de stage et d'alternance que j'ai menées à ses côtés à Sodifrance avant de démarrer la thèse.

Au quotidien, ma montée en compétences sur les outils et cadriciels pour réaliser les prototypes a été permise par la patience, la pertinence, la pédagogie et la rigueur de Laurent Pichierri et Grégoire Dupé.

J'adresse mes remerciements également à tous les collaborateurs de la R&D qui ont

partagé mon quotidien (parsemé de nombreux éclats de rire) à l'entreprise : David Cam, Thomas Cicognani, David Couvrant, Fabien Gicquel, David Launay, François Lelièvre, Jean-Michel Petit, Olivier Remaud, Patrick Talbot, Fabien Tréguer, Yann Tréguer... et toutes les personnes qui sont passées dans l'équipe ITM-Factory.

Je tiens à remercier Christian Chabot qui a été la première personne à m'initier sur la problématique d'architecture logicielle et la veille technologique, ainsi, que les consultants Sodifrance, Hugues Van Eylen et Marc Divet pour avoir partagé avec moi leur expérience en architecture d'entreprise.

J'ai rencontré beaucoup de collaborateurs chaleureux à Sodifrance durant toutes ces années, je ne pourrais tous les citer, ils se reconnaîtront.

Je remercie Xavier Blanc et Jean-Claude Royer qui ont jugé de l'avancement des travaux dans le cadre du **comité de suivi** de thèse et qui ont été de bons conseils. Je remercie les **rapporteurs** Xavier Blanc et Florence Sèdes pour leur analyse avisée du mémoire de thèse. Je remercie également les **examinateurs**. Anne Etien pour son regard éclairé sur les problématiques d'évolution et d'alignement du SI, qui ont constitué également le sujet de sa thèse ; nous avons eu l'occasion d'échanger lors de conférences où nous nous sommes rencontrés. Marianne Huchard d'avoir accepté de participer au jury de cette thèse.

Je ne pourrais terminer sans remercier intensément **ma famille et mes proches** qui m'ont soutenu, encouragé et aidé durant cette épreuve. J'embrasse mes parents Corinne et Denis, mon frère Benjamin, et son épouse Via, mon neveu et ma nièce, mes oncles et tantes, mes grand-mères, Cécile et Loïc, et mon affection particulière à Marie et Gabrielle pour leur précieux soutien. Mes plus fidèles et anciens amis, Pierre, Mathieu, Xavier et Claire, Gaëtan et Marie, Guillaume, Cyril et Marlène, Pierre-Alexandre et Mari, Florent, Benoît, Eric. Et les bandes de copains des associations House 2 Couette et MacBidouille.

Cette thèse a été une expérience exceptionnelle dans ma vie.
À tous, mes très sincères remerciements.

Table des matières

Préambule

La présente thèse vise à la **connaissance**, la **compréhension** et au **contrôle** d'un système dans le but de le faire **évoluer** et de l'améliorer en lui appliquant des modifications par **transformation**. Plus précisément, ici il s'agit d'étudier un **système d'information** (SI) d'**entreprise** qui est un ensemble organisé de ressources socio-techniques :

- **social** : structure organisationnelle des personnes liées à l'activité de l'entreprise, aussi bien les décideurs que les opérateurs ;
- **technique** : technologies matérielles et logicielles ainsi que les processus mettant en œuvre l'activité de l'entreprise.

Les modifications à apporter au SI sont issues d'**objectifs** (besoins et contraintes) stratégiques pour améliorer l'**activité** de l'entreprise. L'étude de l'organisation et du fonctionnement existant et à jour de l'entreprise est réalisée par **cartographie**. La cartographie structure les différentes natures d'informations par **points de vue** : processus, fonctionnel, applicatif, infrastructure... Les différents points de vue doivent être mis en cohérence par **alignement**. La discipline qui vise à aligner les objectifs aux composantes du système d'information lors d'évolutions se nomme **architecture d'entreprise** ou **urbanisation**.

La vie du SI est similaire à la vie d'une cité (une ville). L'urbanisation des cités est une pratique bien antérieure à l'urbanisation du SI, elle remonte au 19ème siècle suite à l'industrialisation et à l'augmentation de l'exode rural. Pour comprendre le principe d'urbanisation, nous proposons une analogie avec le jeu de simulation urbain SimCity®.

SimCity a été créé en 1989 par Will Wright de la société Maxis. En 2008, le code source de SimCity a été diffusé sous licence publique générale GNU (GNU GPL) sous le nom de Micropolis[1]. Nous présentons cette version *open source* en figure 1.

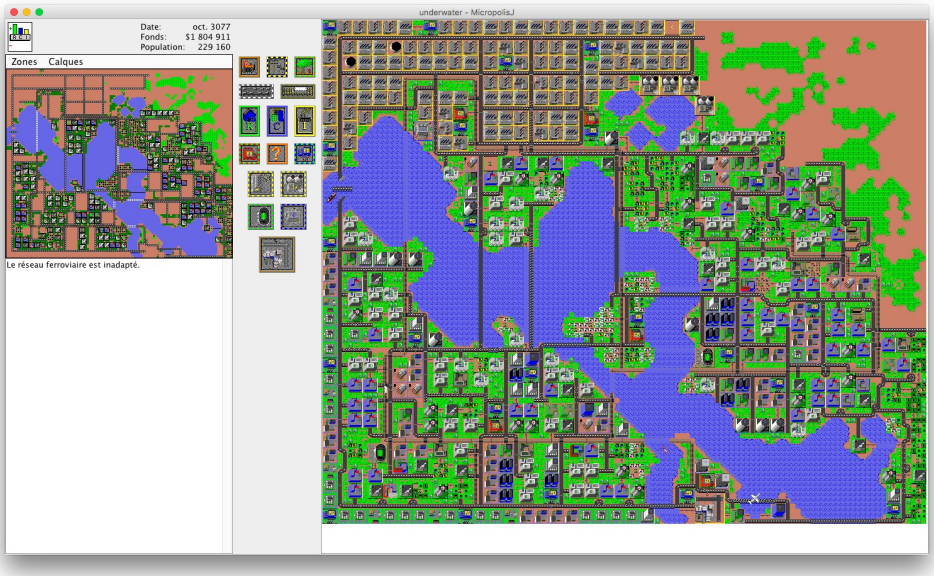


FIGURE 1 – Capture d'écran du jeu Micropolis

[1]La marque SimCity est détenue par l'éditeur de jeux Electronic Arts®

Le joueur incarne le maire de la ville. Le conseil municipal fixe à chaque période budgétaire un taux d'imposition afin de collecter les fonds permettant le fonctionnement de la ville (infrastructure et services) et son aménagement. Plusieurs outils permettent de structurer et développer la ville en différentes zones (résidentielles, commerciales et industrielles) et de définir des infrastructures (route, électricité, réseau ferroviaire et parc) pour relier ces zones. Au début de la partie le terrain de jeu est vierge, trois types de parcelles naturelles subdivisent la carte : parcelle de terre, parcelle aquatique et parcelle arborée. L'intégralité de la carte permet de développer son agglomération à la taille souhaitée, à l'exception des parcelles aquatiques où il est impossible de construire des zones urbaines. Cependant, il est possible d'ériger des ponts au-dessus de l'eau afin de connecter des terres distantes, ainsi que d'accueillir un port maritime en bordure de littoral. L'outil bulldozer est disponible pour restructurer l'agglomération au cours de l'avancement de la partie.

Le simulateur SimCity est pourvu d'un système de cartographie en figure 2 qui permet de sélectionner une vue relative aux différents types qui structurent la ville : résidentielle, commerciale, industrielle ; elles-mêmes liées aux infrastructures de transport.

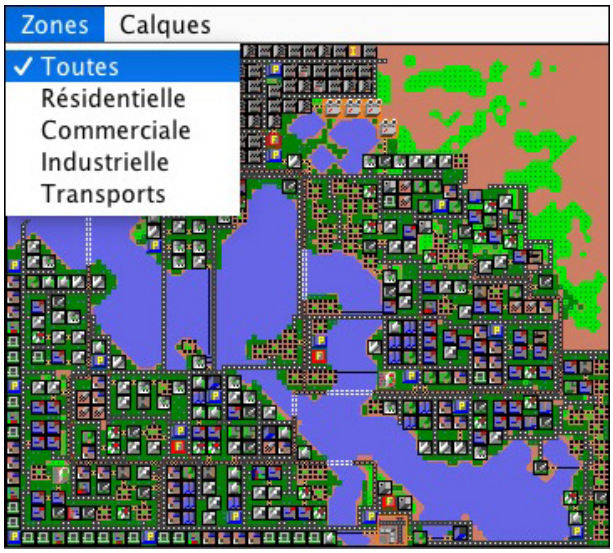


FIGURE 2 – Sélection du point de vue

Chacune des vues permet à l'urbaniste d'obtenir une représentation abstraite sur un aspect du système qui l'intéresse. La vue transport est toujours affichée lors de la sélection de la vue correspondante à chacune des trois zones : résidentielle, commerciale et industrielle. La sélection de la seule vue transport permet de s'abstraire de toutes les zones.

Par exemple, la vue résidentielle en figure 3 permet de localiser les zones d'habitations de la population, la vue commerciale en figure 4 permet de localiser les zones de consommation et de travail de la population, la vue industrielle en figure 5 indique les zones de production et de travail de la population, enfin la vue infrastructure en figure 6 permet de visualiser les voies de circulation entre les différentes zones et parcelles naturelles du terrain. Ces différentes vues permettent de situer où sont construites les zones sur la carte afin de structurer de façon logique les différents **services** liés à la logistique et au transport nécessaire. Par exemple, pour relier les zones d'habitation et de travail sans problème de trafic, il est nécessaire de proposer aux habitants un réseau de routes dimensionné en rapport avec la superficie des zones construites.

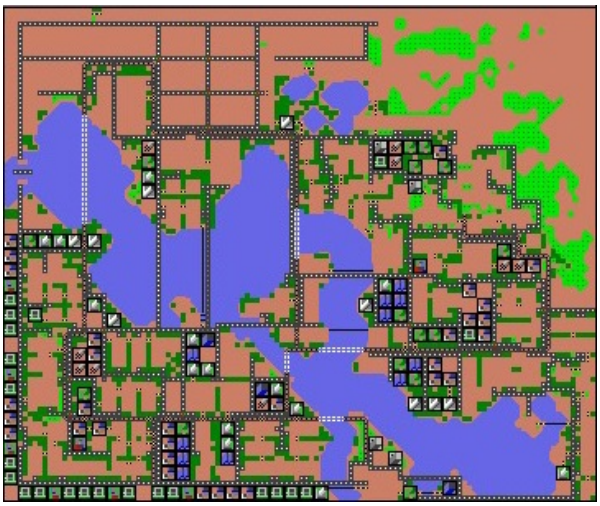


FIGURE 3 – Vue résidentielle

FIGURE 4 – Vue commerciale

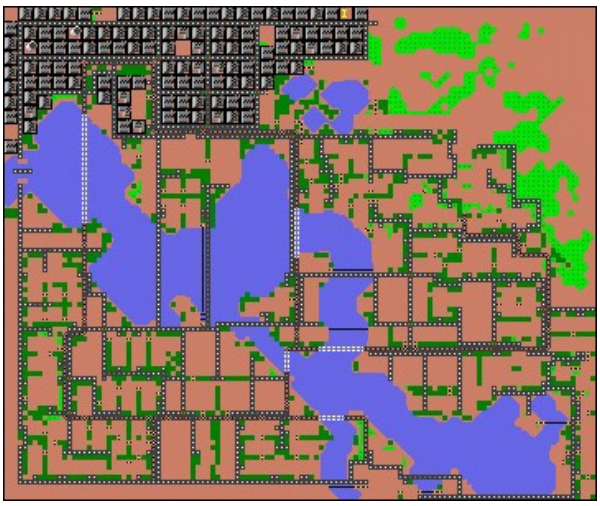


FIGURE 5 – Vue industrielle

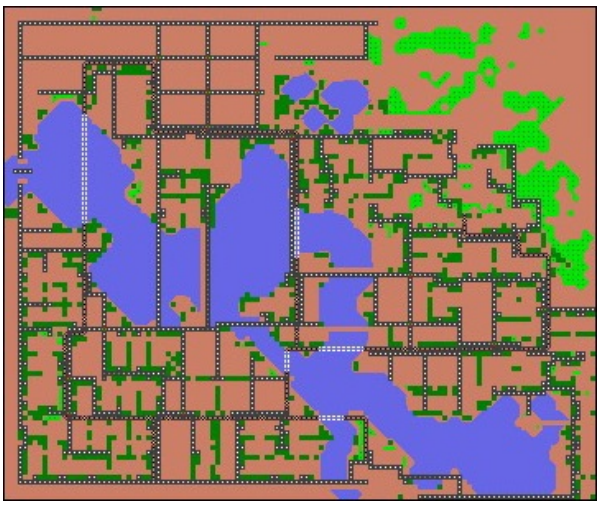


FIGURE 6 – Vue infrastructure

Micropolis propose deux types de métriques pour donner des indications au maire sur l'état de la ville :

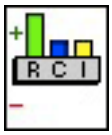


FIGURE 7 – Indicateur de demande RCI

- Les **contraintes** imposent la couverture en postes de police pour lutter contre la criminalité, la couverture en casernes de pompier pour lutter contre l'incendie, le respect de l'environnement en limitant la pollution, une électrification de chaque zone, et la garantie d'un trafic routier fluide.
- La **demande** est formalisée par l'indicateur Résidence-Commerce-Industrie **RCI** en figure 7. L'indicateur permet de connaître pour chaque zone si elle a besoin d'être développée (demande positive), est suffisamment développée (demande nulle), ou au contraire est surdimensionnée (demande négative).

L'évolution étant permanente, il est nécessaire de réaliser à chaque période un alignement de tous ces critères : modifier le système sur l'un de ces aspects induit des impacts sur les autres. Les décisions sur l'évolution à appliquer sur la ville doivent être en respect du budget, des contraintes et de la demande. Par exemple, la création de résidences nécessite de nouveaux commerces, industries et infrastructures, tout en étant vigilant au coût d'entretien de l'infrastructure pour boucler son budget positivement en fin de période. Ces évolutions sont suivies à l'aide de graphiques (en figure 8).

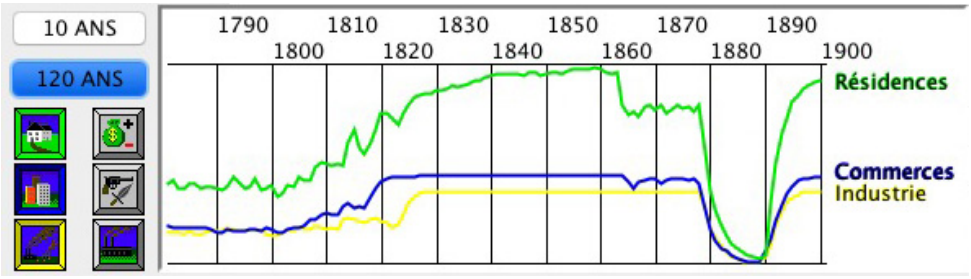


FIGURE 8 – Évolution de chaque zone

Le résultat d'un développement harmonieux de la ville est caractérisé par une croissance de la population, des finances issues des impôts en augmentation, une opinion publique satisfaisante, et un bon score calculé par l'algorithme de simulation du jeu (entre 0 et 1000) (cf. figure 9).

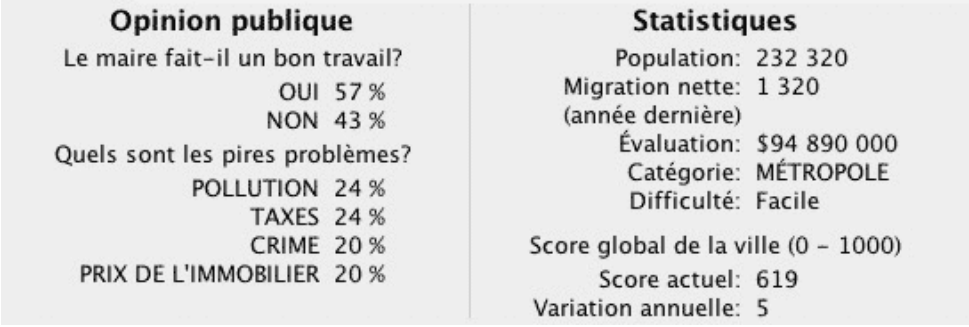


FIGURE 9 – Exemple d'opinion publique et statistiques

L'urbanisation de la cité virtuelle de la simulation Micropolis est simplifiée par rapport à la gestion d'une ville réelle ou au travail d'urbanisation d'un SI réel d'entreprise. Le système est de plus en plus complexe à contrôler selon son niveau de taille : métropole (plus de 400 000 habitants) ou mégapole (plus de 10 millions d'habitants) pour une ville, entreprise de taille intermédiaire (ETI, 251 à 5 000 salariés) ou grande entreprise (GE, plus de 5 000 salariés) pour un SI. Un SI ou une ville est perçu selon différents points de vue avec des objectifs, des méthodes, et des outils différents. De multiples acteurs participent au SI mais avec des points de vue propres, par exemple : le décideur (point de vue stratégique), l'analyste métier (point de vue processus et fonctionnel), l'informaticien (point de vue applicatif), le technicien (point de vue infrastructure). Dans une ville, nous retrouvons cette même multiplicité au niveau des acteurs, par exemple : architectes, économistes, géographes, ingénieurs, juristes, paysagistes, politiques, sociologues, etc. Or dans Micropolis nous avons uniquement le point de vue du maire : l'urbaniste. Ainsi, nous avons abordé la notion de vues multiples et non de point de vues (regroupement de vues d'un même type).

Néanmoins, les grands principes sont présents dans l'exemple de Micropolis : **cartographie, points de vue, alignement, indicateur, impact, évolution**.

La mise en œuvre de l'urbanisation sur les SI pose quelques difficultés qu'il faut surmonter afin de gérer au mieux l'inévitable évolution des SI. Dans la présente thèse, toutes ces notions seront détaillées dans les différents chapitres. Nous présenterons nos propres outils de visualisation du système, similaires à Micropolis, pour suivre l'évolution de l'alignement et l'analyser à l'aide d'indicateurs.

1

Introduction

Sommaire

Introduction

L'**architecture d'entreprise** (ou **urbanisation**) est un domaine qui vise à fournir aux décideurs une vue d'ensemble de l'entreprise à travers son système d'information. Elle inclut de bonnes pratiques pour mettre en place des plans de transitions permettant l'**évolution du système d'information** [21, 62]. L'architecture d'entreprise exploite les **points de vue** du système d'information, notamment les points de vue **métier** et **informatique**.

Ces deux points de vue ont des objectifs, des méthodes, des acteurs, des représentations et des pratiques différents. Chacune de ces différences est une difficulté pour réaliser la cartographie complète du SI.

Par exemple, il est nécessaire de structurer les différentes sources supports de la cartographie ; or elles sont de nature hétérogène même au sein de chaque point de vue. D'une part, la stratégie et le fonctionnement de l'entreprise sont renseignés à travers des documents le plus souvent informels : traitements de texte, présentations, feuilles de calcul ; et dans le meilleur des cas à travers une modélisation métier décrite dans un langage plus ou moins formalisé et normé. D'autre part, le patrimoine applicatif est composé de programmes et infrastructures physiques, s'il n'est pas cartographié la documentation est constituée au mieux de modèles d'architecture, sinon le code source est l'unique référence possible.

Le premier enjeu est donc de définir les concepts de chaque point de vue pour unifier le vocabulaire de travail.

Il peut être difficile de garder ces points de vue cohérents au fur et à mesure des **maintenances**, des **changements technologiques** ou des **évolutions stratégiques et organisationnelles**. L'évolution des points de vue du système d'information est un problème complexe et un enjeu important pour l'entreprise, c'est ce qui permet à l'entreprise d'être à la fois **performante et flexible**. Pour réaliser la transformation du SI, la démarche d'architecture d'entreprise est un projet nécessitant de mobiliser du temps conséquent et du personnel en nombre ce qui peut **engendrer des coûts** importants.

Pour réduire l'écart entre deux points de vue, il est nécessaire de rapprocher leurs modèles, on parle alors d'**alignement**, et notamment d'alignement entre la stratégie métier d'entreprise et le support informatique[1]. Par exemple, un processus de l'entreprise est implémenté[2] par un ou plusieurs programmes. L'alignement permet d'établir un lien entre des éléments de points de vue différents. Il est ensuite possible d'analyser les dépendances entre éléments et de mesurer les impacts de changements lors de l'évolution de ces points de vue.

Notre objectif est de proposer une **méthode pragmatique d'alignement par les modèles** pour les systèmes d'information existants qui nécessitent des évolutions. Aligner de bout en bout chaque élément du SI, du code binaire jusqu'à la stratégie d'entreprise, tient du mythe. Nous définissons alors un triplet de points de vue : processus métier, fonctionnel et applicatif. Chacun d'entre eux est alignable avec l'autre. Selon le niveau de maturité, la préoccupation, ou l'importance de l'entreprise, seulement deux des trois points de vue peuvent être mis en évidence.

La tâche est alors d'obtenir les modèles abstraits et structurés qui sont comparables

[1] BITA : Business-IT alignment en anglais

[2] Ne pas confondre avec ***Implanter*** *« Installer un logiciel ou un sous-système donné en réalisant les adaptations nécessaires à leur fonctionnement dans un environnement défini. »* - Legifrance

et alignables ; or ils n'existent pas toujours. Pour obtenir le modèle applicatif, nous proposons un processus de remontée en **abstraction** depuis le code source des applications. Une série de transformations permet d'abstraire les concepts techniques pour ne conserver que les éléments architecturaux. Pour obtenir les modèles métiers, c'est l'inverse, on doit matérialiser la stratégie d'entreprise, on parle de **concrétisation** ou de raffinement. Si la documentation métier est inexistante ou partielle, les différents acteurs (analystes métier, analystes fonctionnels et utilisateurs finaux) doivent définir les modèles fonctionnels et processus à partir de leurs savoirs et connaissances du fonctionnement et de l'organisation de l'entreprise.

Techniquement pour réaliser l'alignement des modèles applicatifs avec des modèles métiers (processus et fonctionnel), nous proposons une solution non-intrusive de **tissage** par l'implémentation de facettes permettant l'extension virtuelle de méta-modèles. L'ensemble de nos propositions forment une méthode **outillée** en utilisant des standards de l'ingénierie des modèles. Des **expérimentations** sont menées sur des cas concrets d'entreprise du secteur de l'assurance et valident la proposition.

Revenons au problème initial qui est l'alignement ; les modèles sont construits et alignés pour être **analysés** et répondre au questionnement des décideurs. Nous proposons des **indicateurs** afin de permettre à l'architecte de décider des évolutions à réaliser sur le SI.

1.1 Problématique

La stratégie de l'entreprise évolue sans cesse, ce qui nécessite de mettre en place rapidement de nouvelles fonctionnalités pour répondre aux modifications de l'activité. Ces changements ne peuvent être atteints sans rapprochement des différents points de vue du système d'information par alignement, parce qu'on doit mesurer les conséquences des décisions avant d'agir. L'alignement pose et traite la question fondamentale de la cohérence des informations situées à différents niveaux d'abstraction. Une fois l'alignement effectué, son analyse permet d'identifier les actions d'évolution à réaliser. Les échanges nécessitant toujours d'être de plus en plus rapides et d'être capables de transmettre des volumes d'informations de plus en plus importants, la maîtrise des points de vue du SI répond par conséquent à l'amélioration de l'efficacité et des performances du SI.

Nous détaillons les besoins de développement du SI et les motivations pour réaliser l'alignement.

1.1.1 Répondre à des besoins d'évolution du SI

L'évolution de l'entreprise est permanente que ce soit pour s'améliorer en faisant des choix stratégiques internes (nouveau produit, baisse de coût...), ou pour répondre à des contraintes multiples venant de l'extérieur (concurrence, inflation législative...). Nous détaillons quelques-uns des facteurs nécessitant un besoin d'évolution du SI de l'entreprise.

Le facteur **concurrentiel** (figure 1.1) est lié à la mondialisation et au progrès technologique qui ont entraîné la libéralisation de la nouvelle économie, des monopoles et l'ouverture des marchés nationaux vers l'international. Ces changements entraînent une nouvelle concurrence sur les marchés traditionnels. Par exemple, l'ouverture en France entre 2000 et 2007 des fournisseurs d'énergie (gaz et électricité) à la concurrence a aboli les monopoles d'État de EDF et GDF. EDF a dû s'adapter et faire évoluer l'activité de son SI pour proposer des contrats correspondants à la fois aux tarifs réglementés historiques et à la nouvelle offre de marché.

FIGURE 1.1 – Le marché concurrentiel

FIGURE 1.2 – Le ROI et l'actionnariat

Le facteur **financier** (figure 1.2) regroupe l'actionnariat et la distribution de dividendes qui entraînent la recherche de retour sur investissement (ROI[3]) le plus optimal. L'impact sur le SI est important, l'optimisation des processus pour effectuer la chasse aux coûts modifie profondément l'activité de l'entreprise. Par exemple, la délocalisation pour transférer une partie des activités à l'international par la recherche d'une main d'œuvre ou d'une infrastructure moins coûteuse.

Le facteur **légal** (figure 1.3) est la promulgation de lois nationales et internationales qui a une influence sur l'activité des entreprises. Les lois encadrent les activités avec par exemple des obligations de transparence financière, des interdictions d'anticoncurrence, une fixation du prix de vente, etc. L'entrée en vigueur de la loi peut entraîner un coût pour modifier les données du système d'information. Par exemple, la loi sur le mariage pour tous a impacté le champ sexe saisi pour un couple marié dans les applications de gestion. Des milliers d'organismes privés et publics ont dû prendre en charge cette modification, ce coût n'est pas financé par des subventions publiques.

FIGURE 1.3 – Les réformes gouvernementales et nouvelles lois

FIGURE 1.4 – Les évolutions technologiques, la "digitalisation"

Le facteur **technologique** en (figure 1.4) intervient lors de la modernisation du SI pour améliorer son efficacité, sa qualité et les fonctionnalités des services rendus. L'adoption de nouvelles plateformes technologiques est une préoccupation interne à l'entreprise, mais elles doivent être interopérables avec les systèmes externes. L'évolution technologique de ces deux dernières décennies a rendu désuets les systèmes centralisés[4] construits autour de bases de données monolithiques pour laisser la place à des systèmes répartis en réseaux. Cette évolution a permis la transformation des activités vers le tout numérique[5] qui entraîne une refonte des métiers. Par exemple, les agences de voyage physiques n'existent plus, la réservation d'hôtel, la location, l'achat de billet de train ou d'avion se font via des comparateurs et subissent une tarification différenciée en temps réel en fonction de l'offre et de la demande[6].

[3] *Return On Investment* en anglais
[4] *Mainframe* en anglais
[5] Encore appelé *digitalisation*
[6] Le concept du *yield management* a été créé par la compagnie aérienne américaine ***Delta Air Lines***

Les besoins d'évolution du SI ne sont pas tous prévisibles et ils peuvent survenir à tout moment au cours de la vie d'une entreprise. Pour que l'entreprise devienne ou reste *leader* d'un marché, la flexibilité doit être au cœur de sa stratégie : innover en premier, modifier pour améliorer, abandonner juste à temps.

Pour permettre aux décideurs de "décider", il faut leur fournir les éléments pour comprendre l'état actuel et prévoir des scénarios d'évolution chiffrés.

1.1.2 Aligner les points de vue du SI

Pour avoir une meilleure compréhension du système d'information par les différents acteurs, nous souhaitons réaliser une cartographie des différents points de vue qui le composent. Puis, nous souhaitons rendre cohérents les différents points de vue à l'aide de l'alignement. Cet alignement va permettre à l'architecte d'obtenir de nouveaux indicateurs pour décider de la maintenance et des évolutions à appliquer.

Modéliser

Centraliser les documents et connaissances du système d'information est un problème essentiel de l'alignement du SI qui peut être résolu par l'utilisation d'un **référentiel**. Le référentiel a pour but d'obtenir et de maintenir une **image complète du SI** avec les différents points de vue métiers et informatique (figure 1.5). Ainsi, le référentiel permet de connecter les différents supports décrivant l'activité de l'entreprise (PDF, document, présentation, tableur) ainsi que les modèles et patrons de conception (données, objets, comportements) des logiciels. Le principal défi du référentiel est de le maintenir à jour à chaque évolution du SI.

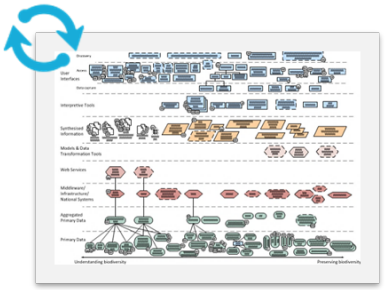


FIGURE 1.5 – Obtenir une image complète et à jour du SI

FIGURE 1.6 – Résoudre la communication

La **cartographie** vise à modéliser les différents points de vue hétérogènes de l'entreprise. Il est nécessaire de **définir les concepts** (traitements et données) de chaque point de vue pour définir un langage adapté aux informations à modéliser. Un langage unifié permet de résoudre la communication entre les différents acteurs et parties prenantes de chaque point de vue (figure 1.6) : décideurs, architectes, développeurs, analystes métier, etc.

L'**alignement** entre les points de vue permet d'établir une **traçabilité** entre les concepts et de vérifier la cohérence du SI : par exemple les applications répondent aux différentes fonctionnalités organisationnelles. À partir de l'alignement, la **navigation** ascendante et descendante (figure 1.7) permet de représenter et confronter les différents points de vue du SI. Le SI n'est plus une collection de différents points de vue sans relation, mais devient un ensemble logique, structuré et cohérent.

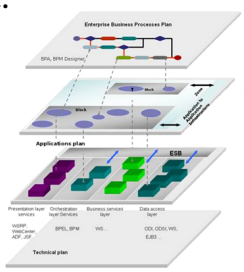


FIGURE 1.7 – Naviguer dans le SI

Évaluer

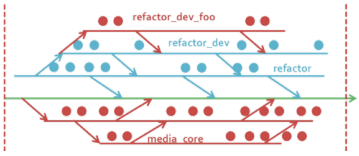


FIGURE 1.8 – Décider à l'aide d'indicateurs

L'alignement du SI permet de calculer des **indicateurs** pour assister l'architecte d'entreprise dans le pilotage et la prise de décision sur les évolutions à appliquer au système d'information (cf. figure 1.8). Les indicateurs doivent permettre de simuler les **impacts** des changements selon les points de vue étudiés et de choisir un scénario de changement. Par exemple, dans la couche applicative, une analyse des interdépendances entre les sous-systèmes permet de mettre en place une architecture orientée service (SOA) pour éviter l'informatique dite "*spaghetti*".

Les indicateurs permettent d'obtenir l'état d'une application au cours de son cycle de vie, et ainsi de décider de l'avenir de ses composants :

- Le **développement** est l'ajout au SI d'un composant applicatif implémentant une nouvelle fonctionnalité spécifiée par un nouveau besoin ;
- La **maintenance** est la correction technique (débogage) ou l'évolution des besoins ;
- La **dépréciation** d'un composant applicatif est la décision de ne plus le maintenir (mettre à jour), il est tout de même conservé en l'état. Une autre version du composant peut être exécutée en parallèle avant le débranchement ;
- La **suppression** d'un composant est sa désinstallation définitive et l'archivage du code source correspondant.

Chacun des problèmes d'alignement sera abordé dans la présente thèse, nous apporterons des solutions outillées afin d'assister l'architecte dans les différentes tâches : définition des points de vue et de leurs concepts, modélisation et extraction des informations, définition de l'alignement, tissage entre les instances de concepts, exploitation et analyse du résultat d'alignement.

1.2 Méthodologie

Le sujet de la thèse "*l'alignement des cartographies métiers et applicatives du système d'information*" découle d'une problématique industrielle constatée auprès des clients de l'entreprise de service du numérique (ESN)[7] Sodifrance. Les entreprises dont le SI comporte une dimension et un historique important ont des problèmes à le faire évoluer tout en respectant les problématiques de coûts. Ainsi, elles s'intéressent de plus en plus aux domaines qui fournissent des réponses, notamment la gouvernance, l'architecture d'entreprise et plus précisément l'alignement du SI. À cette préoccupation industrielle, s'ajoute l'identification des enjeux scientifiques du traitement de l'information : cohérence, performance, évolutivité, qualité, etc.

La nature des points de vue du SI a conduit à étudier des sujets variés et interdisciplinaires au cours de cette thèse : gestion des entreprises, stratégie, architecture logicielle, ingénierie des modèles. Les disciplines utilisent un vocabulaire commun autour de *l'alignement*, mais avec une sémantique différente ; ce qui a posé une difficulté supplémentaire : il a fallu déterminer à chaque lecture si le sujet concernait notre contexte ou un autre.

[7] Appellation Syntec depuis 2013, anciennement *Société de services en ingénierie informatique* (SSII)

Pour conduire cette thèse nous avons réalisé les activités suivantes :

État de l'art : Une étude de l'état de l'art globale a été menée par la lecture d'ouvrages de référence afin d'aborder les notions générales de *système d'information*, d'*architecture d'entreprise*, d'*urbanisation* et de *gouvernance*. Ces lectures ont été complétées par la rencontre de consultants de Sodifrance qui ont pour activité la restructuration d'applications au sein de SI de leurs clients, les restructurations sont induites par des besoins d'évolution aussi bien techniques que métiers. Puis, nous avons conduit une étude plus approfondie pour chaque problème scientifique abordé au cours de la thèse. La prise de connaissance des travaux connexes existants a été faite par la lecture ou le parcours de nombreuses publications, la bibliographie en fin de thèse n'en relève qu'un extrait de nos lectures.

Méthode et définition : À partir de l'état de l'art, des défis identifiés et aussi de nos propres convictions, nous avons élaboré des méthodes, des processus et des méta-modèles. Nous nous sommes aussi appuyés sur nos connaissances dans le domaine de l'ingénierie des modèles.

Développement de prototypes : Avant de commencer le développement logiciel, une montée en compétence a été effectuée sur un socle d'outils d'ingénierie des modèles, notamment le cadriciel[8] Eclipse EMF qui permet de réaliser des méta-modèles dédiés par extension[9]. Le développement de prototypes a été réalisé à partir de composants logiciels existants, améliorés et nouvellement conçus.

Expérimentation de cas d'études : Pour exploiter les prototypes, une recherche de cas d'étude d'entreprise réels a été effectuée auprès des projets clients de Sodifrance et notamment dans le secteur de l'assurance. Bénéficier de ces cas nous a permis de mettre en œuvre nos techniques à l'échelle significative des défis posés. L'expérimentation a permis de conforter nos choix et d'élaborer un processus d'alignement générique et adaptable à chaque cas d'étude.

TABLE 1.1 – Méthodologie et sujets abordés

Méthodologie \ Sujets	Architecture d'entreprise, urbanisation et gouvernance	Points de vue du système d'information	Alignement	Analyse de modèles par indicateurs	Matrice DSM et algorithme MCL
État de l'art	x	x	x	x	x
Méthode et Définition		x	x	x	x
Développement de prototypes		x	x	x	x
Expérimentation de cas d'études		x	x	x	x

Les différentes étapes de recherche ont été réalisées selon le tableau 1.1 pour chaque sujet nouvellement abordé.

[8] *Framework* en anglais
[9] *Plug-in* en anglais

1.3 Synthèse de la démarche et des résultats de la thèse

La démarche complète proposée est résumée par la figure 1.9. Chaque contribution est représentée avec un guide vers le chapitre correspondant. De gauche vers la droite, de haut en bas :

- Les **points de vue du SI** colorés sont ceux que nous avons sélectionnés et que nous souhaitons aligner : le processus métier représenté en bleu, le fonctionnel représenté en rouge, et l'applicatif représenté en jaune.
- Les **méta-modèles** et leurs concepts définissent les trois points de vue sélectionnés. Nous les avons nommés : **BPM** pour le méta-modèle du point de vue processus métier, **Fun** pour le méta-modèle du point de vue fonctionnel et **App** pour le méta-modèle du point de vue applicatif.
- Le **processus d'architecture d'entreprise dirigée par les modèles** permet d'obtenir les modèles conformes aux méta-modèles. Les informations en entrée proviennent du code source ou d'un référentiel d'entreprise. Elles sont exploitées par transformation automatique ou traduction manuelle pour construire les modèles en sortie.
- La **définition de l'alignement** spécifie la mise en correspondance des concepts des trois méta-modèles (BPM, Fun et App) par des liens de deux types : traitement et donnée. Elle est implémentée à l'aide de la technique d'extension virtuelle de méta-modèles par facettes.
- L'**alignement par tissage** des modèles est concrétisé par un assistant sous la forme d'un éditeur logiciel. L'éditeur permet d'ouvrir les modèles conformes à nos trois méta-modèles (BPM, Fun et App). La création de liens entre les modèles repose sur le mécanisme de facettes implémentant notre définition.
- Le **tableau de bord de l'alignement** réunit les indicateurs pour permettre à l'architecte d'entreprise de suivre l'état de l'alignement par point de vue : nombre d'éléments alignés, cohérence, etc.
- La **matrice de dépendances** permet de visualiser les dépendances des modèles représentant les liens d'alignement élaborés durant l'étape de tissage. Un algorithme de regroupement des dépendances améliore la visualisation et donc l'analyse.

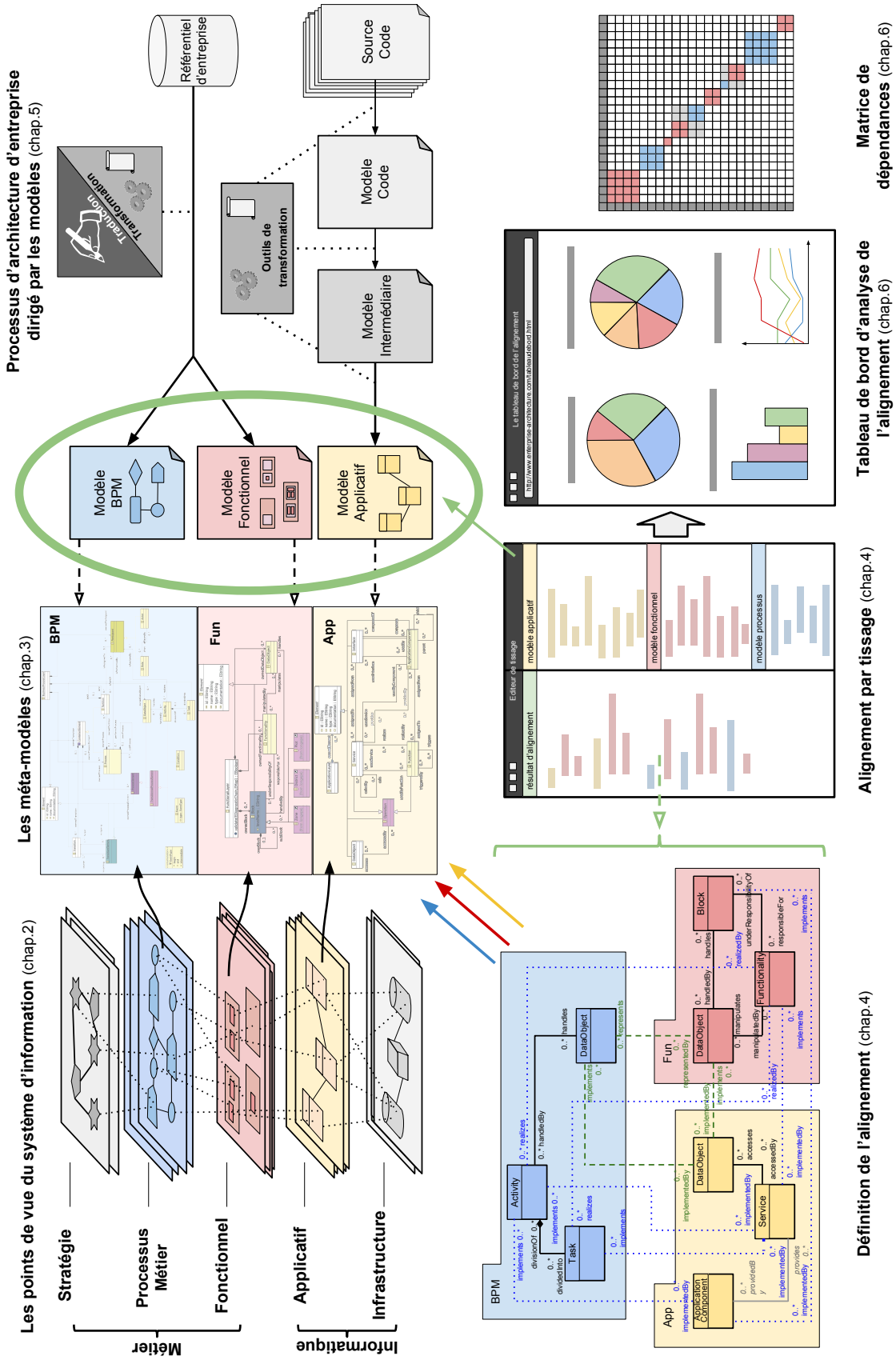


FIGURE 1.9 – Démarche complète de la thèse

1.4 Plan de la thèse

Le chapitre 2 concerne l'**état de l'art** et vise à introduire les différentes notions utiles à la compréhension des disciplines et méthodes environnantes au **système d'information**. Une première partie aborde les normes et bonnes pratiques de **gouvernance** utiles pour la conduite de SI. Puis, nous passons en revue les différents cadres d'**architecture d'entreprise** avec un historique, et présentons une comparaison réalisée entre les cadres que nous avons retenus comme les plus importants : urbanisation, Zachmann, TOGAF et CCU.

Le chapitre 3 est consacré à la **cartographie** du système d'information. Nous sélectionnons les **trois points de vue** que nous retenons pour réaliser l'alignement : processus métier, fonctionnel et applicatif. Pour chacun de ces points de vue, nous illustrons par des exemples comparatifs de notations existantes, puis nous donnons une définition des concepts par la spécification d'un méta-modèle. Les **trois méta-modèles** se nomment : **BPM** (processus métier), **Fun** (fonctionnel) et **App** (applicatif).

Le chapitre 4 introduit notre **définition de l'alignement** à partir des concepts des trois méta-modèles (BPM, Fun et App) définis précédemment. Nous détaillons une typologie de liens selon deux préoccupations : les traitements et les données. Puis, pour réaliser l'alignement, nous faisons un comparatif des différentes techniques de **composition de modèles** existantes. La technique sélectionnée est le tissage par facettes que nous utilisons pour implémenter notre définition de l'alignement. Pour assister l'architecte d'entreprise à effectuer l'alignement des modèles, nous proposons un jeu d'outils : atelier de tissage et éditeur de requêtes.

Le chapitre 5 décrit la mise en œuvre d'un **processus** complet d'alignement à partir des sources d'un patrimoine de système d'information hétérogène : code source, modèles métiers, référentiel... Ce processus vise d'une part à transformer le code source par **rétro-ingénierie** pour obtenir un modèle applicatif abstrait des concepts techniques ; d'autre part il vise à exploiter les informations métiers existants pour obtenir des modèles conformes à notre spécification (BPM et Fun). Le jeu de modèles composé est prêt à être aligné avec la méthode du chapitre précédent. Chaque étape est illustrée par des **expérimentations** tirées de cas d'étude de trois entreprises du secteur de la mutuelle d'assurance que nous nommons : SAMM, SAMI et SAMUT.

Le chapitre 6 est l'**exploitation du résultat de l'alignement** pour suivre l'évolution du SI. Nous définissons des **indicateurs** dans une maquette d'un **tableau de bord** destiné à l'architecte d'entreprise afin de suivre l'alignement : statistique du nombre d'éléments ; taux d'alignement ; détection des éléments alignés et non alignés ; liste des incohérences ; calcul du couplage entre éléments. Pour chacun des indicateurs, nous donnons une définition détaillée et les **requêtes** pour effectuer le calcul à partir de l'alignement de modèles conforme à nos trois méta-modèles (BPM, Fun et App). Enfin pour analyser plus finement le couplage, nous proposons une **visualisation des dépendances** entre les points de vue alignés du SI à l'aide d'une matrice d'adjacence appelée **DSM**, à laquelle nous appliquons l'algorithme de regroupement **MCL** pour déterminer des grappes de dépendances.

Le chapitre 7 de conclusion fait un rappel du travail réalisé par rapport aux objectifs, établit une liste des contributions et donne des perspectives scientifiques et techniques sur le sujet avec des opportunités de valorisation.

2

État de l'art

Sommaire

Introduction

Dans ce chapitre, nous allons introduire les notions essentielles pour comprendre comment une entreprise et son environnement évoluent, et comment elle adapte son système d'information pour répondre à ses nouveaux besoins.

Le **système d'information (SI)** est organisé, évalué, contrôlé, maîtrisé, optimisé et rationalisé sous l'impulsion de la **gouvernance** de l'entreprise, tandis que le SI est cartographié, **aligné**, évalué, mis à jour et transformé à l'aide de l'**architecture ou l'urbanisation**.

2.1 Le système d'information

Le système d'information est un élément majeur dans le fonctionnement d'une entreprise, car il établit des liens entre les différentes parties **internes** de l'entreprise et aussi des échanges avec l'**extérieur** (filiales et autres SI des clients / fournisseurs). Le SI couvre les niveaux stratégiques, décisionnels et opérationnels de l'entreprise.

La complexité du SI est proportionnelle à la taille de l'entreprise ou de l'organisation. Le SI d'une **PME**[1] est **simple** et **unique** comparé à une **multinationale** qui possède un SI global **complexe** et de **multiples** sous-SI. De par son historique, la multinationale vit différentes phases : croissance, acquisitions, fusion. Il existe des divergences selon chaque site et pays d'implantation[2] : droit, culture et normes. Il subsiste différents SI dans la multinationale qui tendent vers la normalisation à l'aide de la mutualisation des pratiques et l'élaboration du SI global pour supporter les flux d'inter-échanges.

DÉFINITION 2.1 – Le système d'information

Le système d'information est un ensemble organisé de ressources : matériel, logiciel, personnel, données, procédures, etc. permettant d'acquérir, de traiter, de stocker des informations (sous forme de données, textes, images, sons, etc.) dans et entre des organisations [87].

La constante évolution du SI complexifie sa gestion. Longépé [67] recommande de la rigueur et de la cohérence, et de trouver un équilibre entre : identification des changements pour mettre en place la stratégie de l'entreprise ; amélioration de l'efficacité et sauvegarde de la cohérence du SI en place ; mises à jour technologiques pour assurer une qualité de service et limiter les coûts.

2.1.1 Propriétés du SI

Le SI manipule un ensemble d'informations (hétérogènes) qui évolue dans le temps. Le SI est la colonne vertébrale de l'entreprise. Il est nécessaire d'organiser et structurer le SI à partir d'objectifs de qualité afin de garantir sa cohérence et pouvoir être exploité par ses utilisateurs.

Caseau [21] définit les propriétés suivantes essentielles au SI :

La flexibilité La flexibilité consiste à pouvoir modifier ou étendre facilement des fonctions rendues par le SI. On parle également d'alignement stratégique qui se concrétise par des petites retouches des outils informatiques : traitements et données.

[1] Petite ou moyenne entreprise
[2] Appelé la localisation en économie

La maintenabilité C'est la capacité à faire évoluer le SI à coût modéré. Quand le système devient trop complexe, il ne s'agit plus de maintenance mais de contournement, le système n'est plus flexible.

La mutualisation C'est la réutilisation de fonctionnalités déjà présentes pour éviter la duplication au niveau des logiciels qui peut entraîner des problèmes de cohérence et de qualité si les fonctions dupliquées évoluent de façon non synchronisée.

L'extensibilité L'extensibilité[3] est le passage à l'échelle de volumes de traitements. Une entreprise doit être capable par exemple de traiter plus de commandes sur une période de l'année. Les applications doivent tenir la montée en charge par une augmentation de ressources sans interruption de service.

La résilience C'est la capacité du SI à continuer de fonctionner en cas de panne à l'aide d'une infrastructure redondante et de sauvegarde, ou de reprendre son activité en cas d'interruption.

Wieringa [106] propose une classification par les aspects d'un système d'information (figure 2.1). Cette hiérarchie étend la taxonomie couramment rencontrée pour décrire les critères de qualité nécessaires à l'élaboration d'une architecture logicielle [85]. On retrouve des critères proches des propriétés précédemment énumérées par Caseau.

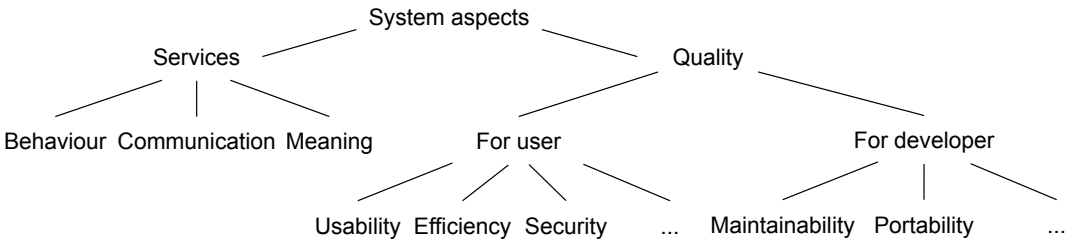


FIGURE 2.1 – Les aspects du système d'information [106]

Au-delà des critères de qualité pour les utilisateurs et les développeurs, un niveau d'abstraction supplémentaire est évoqué : les services. Le système délivre des services à son environnement en respectant différents aspects [62] :

le comportement est l'ordonnancement de l'exécution des services et de ses fonctions ;

la communication est l'interaction entre les entités (personnes, matériel, logiciels) pendant l'exécution du service ;

la signification est la sémantique et la nature des échanges opérés durant le service.

Le système d'information est perçu par ses acteurs selon deux aspects complémentaires : l'aspect **gestion ou métier** (*business*) par les utilisateurs et l'aspect **informatique et technique** (*Information Technology IT*) par les développeurs. Cette dualité *business/IT* est la caractéristique majeure des systèmes d'information d'entreprise. Elle implique des visions différentes des acteurs en jeu, des méthodes et techniques différentes qu'il faut rapprocher. De plus pour chaque aspect, en fonction des acteurs concernés, **plusieurs points de vue** (des niveaux de préoccupation selon [99]) se confrontent. Par exemple dans l'entreprise on distingue entre autres la vue **stratégie** (pilotage, gouvernance), la vue **métier** (processus, fonctionnelle) et la vue **opérationnelle** (organisationnelle, procédure). Du côté informatique, citons la vue **applicative** (le patrimoine des applications), la vue **technique** (les composants), la vue **physique** (infrastructure). Cette classification varie d'une approche à l'autre dans les nombreux cadres d'architecture d'entreprise (*frameworks*) proposés tels que Zachman, DODAF, TOGAF [35].

[3] *Scalability* en anglais

2.1.2 Cartographie du SI

Une cartographie est une **photo du SI** à un instant t [7]. Cet aspect temporel statique limite cette représentation, elle n'est pas vivante et ne permet pas de rendre compte du SI lors de son exécution (nombre d'utilisateurs connectés, nombre d'appels à une fonction, nombre d'instances dans une table de données...). Pour voir le film (historique du SI), il faut juxtaposer les photos (les instantanés), ce point de vue en fonction du temps permet de voir l'**évolution du SI**.

DÉFINITION 2.2 – Cartographie

La cartographie est l'ensemble des études et des opérations scientifiques, artistiques et techniques, intervenant à partir des résultats d'observations ou de l'exploitation d'une documentation, en vue de l'élaboration et de l'établissement de cartes, plans et autres modèles d'expression, ainsi que de leur utilisation.

Dictionnaire de l'urbanisme et de l'aménagement - Pierre Merlin et Françoise Choay [73]

L'utilisation du terme cartographie provient essentiellement de la discipline de l'urbanisme[4]. Dans la littérature anglo-saxonne, le terme employé est **point de vue** : *Viewpoint*. De nombreuses méthodologies de spécifications de points de vue [61] ont été élaborées au cours des années 70 à 90 : *Structured Analysis and Design Technique* (SADT), *Controlled Requirements Expression* (CORE), *ViewPoint Oriented System Engineering* (VOSE) et *Viewpoint-Oriented Requirements Definition* (VORD). La technique de visualisation et création des points de vue à l'aide de modèles s'appelle *Landscape Map* [97] que l'on peut traduire par « paysage ou panorama ».

Le SI est communément structuré sous la forme d'une **pile** de différents niveaux d'abstraction, de perception et de préoccupation. Pour représenter chaque niveau de la pile, les termes couramment employés sont : la vue, le point de vue, et la couche. Nous privilégierons le terme point de vue. Le point de vue est composé de différentes vues. La couche est synonyme de point de vue selon le niveau d'abstraction ou de détail. La cartographie désignera la démarche globale d'élaboration des différents points de vue.

Une cartographie a une **granularité** correspondant au niveau de détail des informations à représenter. Plus la granularité est fine, plus l'effort du travail de mise à jour sera important [7] (coûteux en temps et nombre de personnes dédiées). De plus, une cartographie complexe sera compliquée à lire et à maîtriser. À l'inverse une granularité trop large peut omettre des informations essentielles. Au-delà de la granularité, qui est une préoccupation verticale, le périmètre qui est une préoccupation horizontale est à prendre en considération : où la cartographie commence-t-elle et où s'arrête-t-elle ? Telle information fait-elle partie de cette cartographie ou d'une autre ?

Les **modèles** d'architecture présentent les points de vue en couche qui mélangent abstractions et préoccupations. Les couches hautes se focalisent sur la stratégie dans une vision métier, tandis que les couches basses se focalisent sur la mise en œuvre des données et de leurs traitements où les niveaux bas sont les plus concrets et opérationnels. Par exemple, le cadre de Longépé [67], schématisé dans la figure 2.2 propose **cinq couches**. On distingue chaque couche par des symboles différents, par exemple : étoile sur la couche stratégique, rectangle sur la couche fonctionnelle. Ces différences correspondent à la nature variée des informations collectées pour représenter la totalité du SI.

[4] Étude et action de l'urbanisation

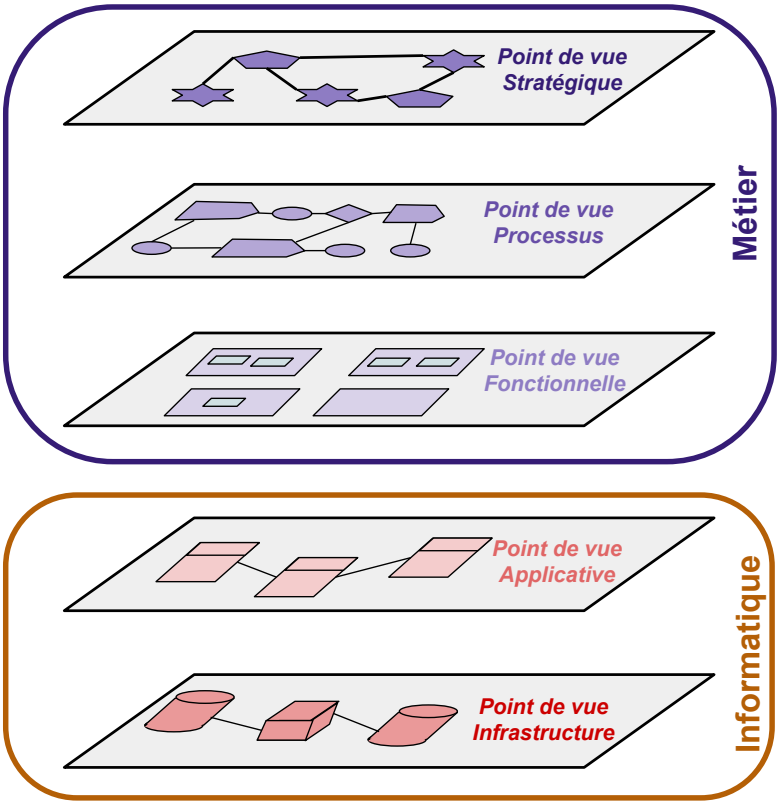


FIGURE 2.2 – Couches du système d'information selon Longépé

Stratégie : la couche stratégie donne le positionnement de l'entreprise sur le marché concurrentiel : les **objectifs** métier de l'entreprise [105] (performance à atteindre, mise sur le marché d'une nouvelle offre, etc.) ; mais aussi sur son fonctionnement et son **organisation** interne (exemple : plan de coupe budgétaire, restructuration, choix d'une nouvelle plateforme technologique, etc.)

Processus : la couche processus prend en charge l'organisation à travers des **processus**, des **activités** et des acteurs [104]. Le processus permet d'exécuter les activités en identifiant leurs enchaînements et leurs conditions d'enchaînement. La couche processus permet de gérer la qualité au moyen d'indicateurs de performance sur les processus : temps et coût pour réaliser une activité, ou encore fluidité des échanges d'information et de contrôle. La couche processus permet aussi de comprendre comment fonctionnent les métiers de l'entreprise et se décomposent les activités de l'entreprise, et permet ainsi de déterminer quelles compétences sont requises pour les accomplir.

Fonctionnelle : la couche fonctionnelle représente la structure des différentes **fonctions** d'une organisation. Les fonctions réalisent les activités de la couche processus. Exemple, la fonction marketing d'une entreprise a pour activités : faire une étude de marché, modéliser une offre de produit / service, établir une tarification, trouver des distributeurs, etc. La couche fonctionnelle organise les fonctions sous forme hiérarchique, en termes de blocs fonctionnels, le découpage le plus commun est la granularité : **zone, quartier, îlot**. Zone est le bloc le plus haut. Un découpage fonctionnel à jour permet à l'entreprise de se séparer ou de sous-traiter un de ses blocs (exemple : vente, production, etc.)

Applicative : la couche applicative modélise la partie automatisée du SI à savoir les informations stockées et manipulées, ainsi que leurs traitements. Elle est composée d'**applications**, de **composants logiciels** et de **services** implémentant des fonctionnalités et leurs relations. La couche applicative décrit également les structures de données et flux (ou messages) qui transitent entre les différents composants applicatifs.

Infrastructure : la couche infrastructure représente toutes les ressources nécessaires au stockage, à la communication et à l'exécution des unités logicielles : bases de données, **réseau**, intergiciels, jusqu'au **matériel**. La couche infrastructure enregistre les coordonnées où sont installés physiquement les logiciels (IP, nom du serveur) ou les matériels (adresse du site d'info-gérance, bâtiment, étage, baie, etc.) Cette couche permet de suivre le cycle de vie et la bonne exécution de chaque ressource, et le cas échéant de localiser rapidement le logiciel ou le matériel en panne. La couche infrastructure peut être enrichie de critères pour déterminer le coût, le taux d'utilisation ou encore l'âge des logiciels.

Note

Nombre de points de vue Il existe d'autres points de vue transversaux et secondaires sur le SI. Par exemple, la gestion des risques, ou la sécurité qui influent sur l'organisation du SI. Nous ne les aborderons pas dans la thèse, ces points de vue sont en dehors de notre préoccupation d'alignement qui se situe essentiellement à la frontière entre couches métier et applicative. Cette frontière est visible sur la figure 2.2 qui représente deux domaines distincts : métier et informatique.

Plan d'occupation des sols Dans la littérature liée à l'urbanisation, le terme plan d'occupation des sols (POS) est souvent utilisé, il peut être employé à chaque niveau de cartographie du SI. Néanmoins, dans un contexte où la nature de la cartographie du POS n'est précisée (processus, fonctionnelle, applicative), il s'agit de la cartographie fonctionnelle. Nous donnerons une définition plus précise du POS dans la section 2.3.2.

Nous détaillerons davantage les points de vue sélectionnés (processus, fonctionnel et applicatif) dans le chapitre 3 pour répondre à nos objectifs d'alignement du SI formulés dans la section 2.5.

2.1.3 Problème de la maîtrise des évolutions du SI

Le SI a un volume d'informations souvent proportionnel à l'importance de l'entreprise et à la complexité de son métier. Nous pouvons évoquer quelques-uns de ces paramètres de complexité :

- nombre de collaborateurs (TPE, PME, GE, multinationale)
- nombre de sites géographiques (local, national, continental, international)
- historique (entreprise âgée de quelques années ou plusieurs décennies)
- étendue des prestations (catalogue d'offres : services et produits)
- nombre de partenaires et intermédiaires (fournisseurs, clients, sous-traitants)
- contraintes du secteur d'activités (lois et normes)
- cadence des évolutions (degré de concurrence)
- etc.

Chaque entreprise est singulière, elle a un fonctionnement propre et un contexte propre, ce qui entraîne des problèmes pour maîtriser le SI. Il faut donc le personnaliser :

- Quelles sont les informations à **cartographier** ? Quelles couches faut-il adopter, toute ou en partie (métier, fonctionnel, applicative, infrastructure) ? Quelle **granularité** (niveau de détail) est nécessaire ? Ces choix posent un problème de compromis entre exhaustivité pour permettre une analyse du SI suffisante et volume d'informations à traiter qui allonge le temps de cartographie du SI.
- Comment **aligner** les différentes cartographies ? La cartographie d'aujourd'hui et de demain ont des informations différentes. Les couches peuvent être réalisées par différents acteurs de l'entreprise. Comment mettre en correspondance des couches de **nature différente** ?
- Comment **décider** quelles parties du SI sont à faire **évoluer** de façon prioritaire ? Quels **choix** technologiques et stratégiques s'imposent ?
- Comment **normer et contrôler** le SI ? Quelles pratiques de sécurité, performance gestion des risques, gestion de production ?

Pour couvrir et répondre aux problèmes du SI deux branches d'étude existent : la gouvernance et l'architecture d'entreprise. Nous allons introduire un peu plus dans le détail ces deux disciplines dans les sections suivantes.

2.2 La gouvernance des SI

L'élaboration de normes de contrôle des entreprises a débuté dès les années 2000 suite aux scandales financiers Enron, Worldcom, Madoff... Le Congrès américain promulgue alors la loi Sarbanes-Oxley [31], illustrée par l'exemple 2.1 définissant des normes de comptabilité et mesures de contrôle. La mondialisation de la finance et des industries entraîne les autres continents à prendre des mesures similaires : la France en 2003 avec la loi de sécurité financière LSF [46] et plus largement l'accord européen BALE II en 2004. Ces lois édictent les principes de transparence [49] suivants : une responsabilité accrue des dirigeants, un renforcement du contrôle interne et une réduction des sources de conflits d'intérêts. Ces nouvelles préoccupations ont impacté les entreprises sur le contrôle de leur SI, pour en assurer une plus grande maîtrise.

EXEMPLE 2.1 – Exemple d'impact sur le SI de la loi Sarbanes-Oxley

- **409 - *Real Time Issuer Disclosure*** La section 409 oblige, entre autres, les entreprises à être en mesure de clôturer leurs comptes le plus rapidement possible (deux jours).
- **404 - *Management Assessment of Internal Controls*** La section 404, beaucoup plus contraignante, impose aux entreprises de mettre en place des contrôles internes dont l'efficacité devra être démontrée.

DÉFINITION 2.3 – La gouvernance du système d'information

La Gouvernance du système d'information décrit comment un système d'information est dirigé et contrôlé. C'est à la fois l'association du pilotage[5]*, pour s'assurer que les décisions d'aujourd'hui préparent convenablement demain, et du contrôle afin de mesurer l'écart par rapport à ce qui était prévu.* Gouvernance du système d'information, p.11 [84].

[5] Le contrôle des performances d'une entreprise grâce à des tableaux de bord [48]

Il n'existe pas réellement de cadriciel[6] qui **couvre** en totalité **la gouvernance** du SI. Du point de vue des standards, le SI est abordé selon des facettes très différentes : le **service** et le **management de production** (Bibliothèque pour l'infrastructure des technologies de l'information - ITIL[7]), le **développement** et l'**organisation de projet** (Modèle intégré du niveau de maturité - CMMI[8], Guide du corpus des connaissances en management de projet - PMBOK[9]), la **technologie** et le **pilotage des processus** (Objectifs de contrôle de l'Information et des Technologies Associées - CobiT[10] et ISO 38500), la **sécurité** (ISO 27000). Les auteurs du livre "Les services agiles" [22] ont représenté les différents services de référentiels sous forme d'un temple que nous avons adapté et illustré par la figure 2.3. Elle permet de comprendre les différents **rôles de chaque norme**. Il est nécessaire de choisir les normes et certifications pour répondre aux principales préoccupations de gouvernance du SI représentées sur la figure 2.4.

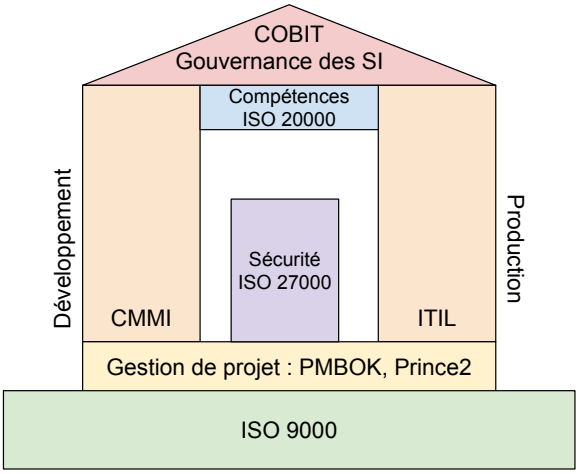


FIGURE 2.3 – Le temple du service

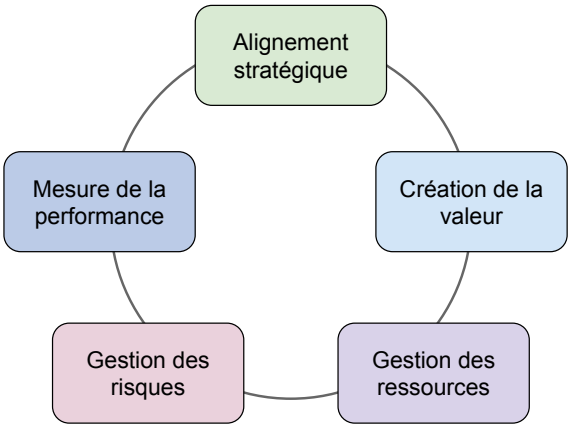


FIGURE 2.4 – Les 5 piliers de la gouvernance

Chaque norme a tendance à étendre son champ de compétences, aussi peuvent-elles finir par se recouvrir, ou faire **doublon**. La clef est donc l'intégration et l'adaptation en choisissant de construire sa propre démarche et en implémentant quelques parties des normes plutôt que d'avoir pour objectif de tout mettre en œuvre. Le Cigref [28] définit les objectifs de la gouvernance à travers trois questions : Comment sont prises les décisions concernant le système d'information ? Comment faire pour améliorer et faire accepter la prise de ces décisions ? Comment s'assurer que ces décisions seront convenablement mises en œuvre ? Ainsi, la mise en place de la gouvernance doit permettre la **remontée d'indicateurs de performance** compréhensibles, utilisés par la direction afin d'évaluer quotidiennement le bon fonctionnement des services informatiques, en réponse aux besoins stratégiques de l'entreprise [14].

Nous passons en revue les normes de gouvernance de SI les plus communément utilisées.

[6] *Framework* en anglais
[7] *Information Technology Infrastructure Library* en anglais
[8] *Capability Maturity Model Integration* en anglais
[9] *Project Management Body of Knowledge* en anglais
[10] *Control Objectives for Information and related Technology* en anglais

2.2.1 ITIL

Information Technologiy Infrastructure Library (ITIL), ou Bibliothèque pour l'infrastructure des technologies de l'information est un référentiel ouvert et public de bonnes pratiques visant à améliorer la **fourniture de services** informatiques d'une entreprise. L'*Office of Gouvernment Commerce* (OGC), ou le bureau du commerce du gouvernement est un cabinet du gouvernement britannique à l'origine d'ITIL en 1988. Le but est d'améliorer le service public et l'utilisation de l'argent du contribuable. Depuis la version 3 de 2007, des experts de grandes entreprises interviennent : EY, HP, CGI... c'est à partir de cette version que sont traitées la gouvernance des SI et l'alignement stratégique (cf. section 2.4.3) du SI sur les processus métier. A l'origine une norme européenne, ITIL s'est ensuite implantée début 2000 sur le marché nord-américain. Désormais **présente au niveau mondial**, elle est devenue incontournable par la promotion du réseau *itSMF*. Les thématiques de ITIL v3 sont : *Stratégie des services, Conception des services, Transition des services, Exploitation des services, et Amélioration continue des services*. Le client est au cœur de l'approche ITIL, l'implémentation commence fréquemment par la gestion du référentiel de configuration[11], la gestion des changements, et la gestion des incidents et problèmes.

2.2.2 CobiT

Control Objectives for information and technology (CobiT), ou Objectifs de contrôle de l'Information et des Technologies Associées, est un cadre de référence pour assurer la **maîtrise de la gouvernance** du SI dans la durée par la certification d'une personne physique [91]. L'*Information Systems Audit and Control Association* (ISACA) a développé CobiT à partir de 1994 comme référence pour l'audit des SI, le cadre est représenté en France depuis 1967 par l'association française de l'audit et du conseil informatique (AFAI). La formation est destinée aux directeurs des SI, consultants, formateurs et experts en sécurité. Issue des bonnes pratiques des experts mondiaux, CobiT vise à contrôler la gouvernance plutôt que son exécution. Ainsi CobiT aide à optimiser et évaluer les risques liés aux investissements informatiques ; contrôler la sécurité des services informatiques internes ou sous-traités ; assurer la fourniture de services et leur bon fonctionnement. Il existe plusieurs versions de CobiT ; elles sont complémentaires. CobiT 4.1 traite les thèmes suivants : *Planification et Organisation ; Acquisition et Installation ; Livraison et Support ; Monitoring*. Tandis que CobiT 5 aborde les sujets suivants : *Évaluer, diriger, et surveiller ; Aligner, planifier et organiser ; Bâtir, acquérir, et implanter ; Livrer, servir et soutenir ; Surveiller, évaluer et mesurer*. Une des préoccupations de CobiT (également une des notre) est de contribuer à l'alignement des technologies sur la stratégie d'entreprise. Cet alignement s'occupe spécifiquement des budgets d'investissement et de fonctionnement pour que la valeur ajoutée de l'informatique soit conforme au métier de l'entreprise.

2.2.3 ISO 38500

Publiée en 2008, ISO 38500 est une **norme de gouvernance** des technologies de l'information par l'entreprise, elle est issue de la reprise de la norme australienne AS8015. Cette norme a trois principaux objectifs :

1. L'évaluation de l'utilisation de l'informatique actuelle et future.

[11] *Configuration Management DataBase* (CMDB)

2. La préparation directe et la mise en place de plans et de politiques pour assurer que les applications informatiques concourent effectivement à la réalisation des objectifs généraux de l'entreprise.
3. Le pilotage de la conformité des politiques et des performances permises grâce à la mise en place des plans.

2.2.4 CMMI

Capability Maturity Model Integrated (CMMI), ou Modèle intégré du niveau de maturité, est un ensemble de bonnes pratiques et de mesures de **qualité des services informatiques**. CMMI a été développé par le *Software Engineering Institute* de l'Université de Carnegie-Mellon à Pitssburgh (Pennsylvanie - USA) à la demande du département de la défense des États-Unis (DoD) pour appréhender et mesurer la qualité des services rendus par leurs fournisseurs de logiciels. CMMI regroupe 3 activités avec un socle commun à 60% : CMMI-DEV pour le développement des systèmes, CMMI-ACQ pour la maîtrise des services achats, CMMI-SVC pour la fourniture de services. CMMI est composé de 5 niveaux : initial (1), discipliné (2), ajusté (3), maîtrisé (4), et optimisé (5). Le niveau 1 est proche d'un état dans lequel où aucune bonne pratique n'est encore mise en place. CMMI est une démarche qualité pour le développement logiciel qui permet une meilleure allocation des ressources, un meilleur pilotage et donc une réduction de coût et une tenue supérieure de délais [22]. La démarche s'intègre dans la transformation de SI pour encadrer le développement de ses évolutions.

2.2.5 PMBOK

Project Management Body of Knowledge (PMBOK), ou Guide du corpus des connaissances en management de projet, dont les origines remontent à 1983 avec une première publication par l'organisme mondial sans but lucratif *Project Management Institute* (PMI) a pour but de **standardiser l'approche et les procédures de gestion de projet**. Le premier PMBOK est publié en 1996 et reconnu comme un standard officiel de l'ANSI depuis 1999. Cette norme rencontre une très forte adhésion ; en 2015 sont recensés 476 000 membres dans 204 pays. PMBOK définit un ensemble de 47 processus regroupés autour de ces 5 processus parents : 1 - Démarrage, 2- Planification, 3 - Exécution, 4 - Surveillance et maîtrise, 5 - Clôture. Chaque processus détaille des mécanismes de transformation d'intrants (documents de planification, de conception) en extrants (documents, produits...) [29]. PMBOK est complémentaire de la démarche d'amélioration continue CMMI.

2.2.6 ISO 27000

ISO 27000 est une suite de bonnes pratiques regroupées en norme qui traite de la **sécurité de l'information**. Rédigée à partir de 2005, elle est en permanente évolution. La version d'ISO 27000 publiée en 2009 est un court document introductif de 38 pages définissant la famille des normes et les termes. La seule norme menant à la certification est ISO 27001, elle définit un cahier d'exigences et des points de contrôle afin de protéger les actifs informatiques contre tout perte, vol, intrusion ou altération du système informatique. Tandis que la norme ISO 207002 est un guide de bonnes pratiques énumérant les mesures pour la mise en place ou le maintien d'un Système de Management de la Sécurité de l'Information (SMSI). Les bonnes pratiques sont appliquées au travers de règles de sécurité au SI pour garantir la protection de son

infrastructure. La sécurité protège des **attaques informatiques** qui viserait le vol de données par intrusion ou le déni de service[12] interrompant l'activité de l'entreprise.

2.2.7 Bilan

Les six différentes approches de gouvernance passées en revue couvrent chacune des propriétés spécifiques du SI. La gouvernance du SI est un projet d'envergure qui nécessite une évaluation au préalable afin d'éviter un échec lors de sa mise en œuvre.

Applicabilité de la gouvernance

La gouvernance du SI concerne des entreprises de taille intermédiaire et supérieure. Initier et entreprendre un travail de gouvernance est complexe, long et coûteux. Il nécessite un investissement permanent des parties prenantes et la mise en place d'équipes pour mettre en place le projet de gouvernance. Le coût de mise en œuvre peut être important : formation des personnes concernées dans l'entreprise ; appel à un consultant externe ; certification de la norme. La gouvernance ne résout pas tous les problèmes, comme nous avons pu le voir, chacune des approches est dédiée à des sujets précis. De par son coût de mise en place, un retour sur investissement est encore moins garanti.

Alignement et gouvernance

La gouvernance concerne spécifiquement la discipline de l'informatique de gestion. La présente thèse se préoccupe de l'alignement entre les métiers et les architectures logicielles. Un des sujets de la gouvernance semblerait pouvoir répondre à notre préoccupation, en effet, la gouvernance aborde la notion d'alignement. Cependant, il s'agit d'un seul type d'alignement, l'alignement stratégique. Nous définirons cet alignement dans la section 2.4.3, nous expliquerons en quoi cet alignement ne correspond pas encore au champ d'étude visé qui ne s'arrête pas qu'aux domaines métier et stratégique.

Pour aller plus loin sur ce sujet, Claudepierre a réalisé une thèse [29] sur les concepts manipulés par la gouvernance des SI. Il a dressé une comparaison (figure 2.5) multi-critères (sujet, usage, système et développement) entre plusieurs approches de gouvernance (CobiT, COSO, ITIL, CMMI, PMBOK).

Cet aperçu des principales méthodes de gouvernance de SI nous a permis d'appréhender plus amplement ce que signifie la gouvernance et de prendre connaissance du rôle de chacune d'entre elles : sécurité, qualité, offre, services, normalisation, conduite de projet, coûts. L'application de différentes bonnes pratiques de gouvernance influe sur les aspects opérationnels de la transformation du SI, elle facilite sa conduite et son contrôle. Connaître la fonction de la gouvernance pour le SI c'est également éviter de la confondre avec l'architecture d'entreprise qui se préoccupe aussi du SI. Nous abordons l'architecture d'entreprise dans la prochaine section.

[12] *Denial of Service* (DoS) en anglais

Cadre des quatre mondes		*Approches de la gouvernance des systèmes d'information*				
Monde	*Facette*	CobiT	COSO	ITIL	CMMI	PMBOK
Sujet	ORGANISATION DE LA GOUVERNANCE	*	*	-	-	*
	DECISION	-	-	*Infrastructure*	-	*Plan. projet*
	PROCESSUS IT	*	*	*	*	*
	PROCESSUS METIER	-	*	*	-	*
	CHANGEMENT	*	-	*évolutif*	*	*
	PORTEFEUILLE DE PROJETS SI	-	-	*	*Clas. : Multi-critère Transfo. : **	*Clas. : Multi-critère Transfo. : **
Usage	MINIMISER LES RISQUES	*	*	*qualité*	*	*
	ATTEINDRE L'ETAT D'ALIGNEMENT	-	-	*Evolution SI*	-	*Evolution SI, Evolution métier*
	OBTENIR LA PERFORMANCE	-	-	-	*Maturité des processus*	*Maturité des processus*
	CREER DE LA VALEUR	-	*Patrimoine métier*	*Patrimoine SI, usage*	*Patrimoine SI*	*Patrimoine SI, Patrimoine métier*
Système	CONTENU	*document*	*document*	*document*	*document*	*document*
	MODELE	*Processus, objet*	*Processus, objet*	*Processus, objet*	*Processus, objet*	*Processus, objet, décision*
	METRIQUES	*Risque, performance, valeur*	*risque*	*Alignement, performance*	*performance*	*Risque, Performance, Valeur*
Développement	NATURE DES PROCESSUS	*systématique*	*systématique*	*systématique*	*systématique*	*systématique*
	MATURITE DES PROCESSUS	*	-	-	*	*
	CAPITALISATION DE LA CONNAISSANCE	*externalisation*	*externalisation*	*externalisation*	*externalisation*	*externalisation*
	LOGICIEL	*	*	*	*	*

* Facette pleinement couverte, - facette non couverte

FIGURE 2.5 – Comparaison des approches de gouvernance des SI [29]

2.3 L'architecture d'entreprise et urbanisation

L'architecture d'entreprise vise à **modéliser** le système d'information de l'entreprise pour en **contrôler l'évolution** [20]. L'urbanisation cible plus spécifiquement le système d'information, et la démarche d'urbanisation vise à en **améliorer le fonctionnement** en le rationalisant dans un "cercle vertueux de transformation et d'amélioration continue" [99]. Pour Y. Caseau [21], l'urbanisation est en premier lieu une démarche technique d'accompagnement du SI utilisant des principes simples (**décomposition, découplage, intermédiation**) pour répondre à des objectifs de **flexibilité**, de **mutualisation**, de **maintenabilité**, etc.

2.3.1 Historique

Dans les années 70, les informaticiens et les analystes métiers ont commencé à se préoccuper et investir les problématiques SI, telles que celles exposées précédemment dans la section 2.1.3. Il faut attendre les années 80 pour voir émerger les premières méthodes et concepts d'architecture d'entreprise. IBM, à l'époque leader du marché informatique, fera la promotion de la méthode *Business Systems Planning* (**BSP**) considérée comme pionnière de l'Architecture d'Entreprise. John **Zachmman** un des principaux concepteurs de la méthode BSP et employé d'IBM utilisera pour la première fois en 1982 le terme "**Architecture d'Entreprise**", avant de présenter la première version de *Information Systems Architecture Framework* en 1987 qui est nommée plus communément comme méthode Zachmman.

Au fil du temps, le cadriciel Zachmman a inspiré d'autres méthodes [71], notamment *Enterprise architecture planning* (**EAP**) en 1992 par Steven H. Spewak, puis *Integrated Architecture Framework* (**IAF**) méthode développée en 1996 par la société informatique Cap Gemini rejointe par le cabinet d'audit financier Ernst & Young en 2001. En parallèle, une autre branche de méthodologie d'architecture d'Entreprise est issue des travaux du gouvernement des États-Unis d'Amérique (USA) et particulièrement du département

de la défense (DoD). À partir de 1986 la méthode *Technical Architecture Framework for Information Management* (**TAFIM**) est développée, il s'agit de l'ancêtre de la méthode *The Open Group Architecture Framework* (**TOGAF**) qui apparaît en 1995. Mais le DoD développera simultanément à TAFIM, la méthode *Department of Defense Architecture Framework* (**DoDAF**), anciennement appelée *Command, Control, Communications, Computers, Intelligence*[13]*, Surveillance, Reconnaissance* (**C4ISR**). TAFIM est abandonnée en 1995, TOGAF lui succèdera. Les USA ne sont pas le seul pays à développer des méthodes, en Europe des partenariats industriels ont établis les cadriciels *Computer Integrated Manufacturing Open Systems Architecture* (**CIMOSA**) puis *Generalised Enterprise Reference Architecture and Methodology* (**GERAM**) qui auront influencé l'éditeur de logiciel commercial ARIS. En France, la **méthode d'Urbanisation** définit sa propre méthodologie à partir de la **métaphore de la cité**. L'état français développe ses propres méthodes influencées par les existantes : **AGATE** pour le département de la défense dérivé de DODAF, NAF et MODAF ; le cadre commun en 2012 qui résulte de travaux du **DISIC** (Direction interministérielle des systèmes d'information et de communication de l'État).

Nous ne détaillerons pas toutes les différentes démarches et évolutions historiques, ni ne pourrons en faire un listing complet. La figure 2.6 donne un aperçu des principales démarches pour permettre de comprendre leurs inspirations, leurs affiliations et leur généalogie.

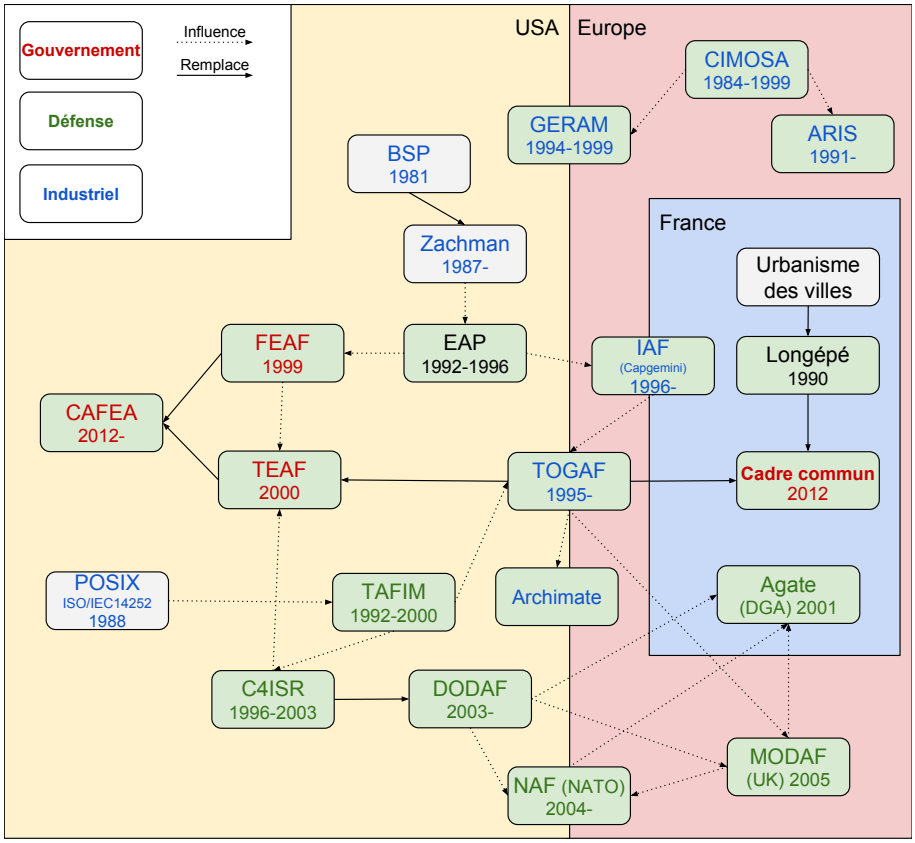


FIGURE 2.6 – Affiliation non exhaustive des méthodes d'AE aux USA et en Europe

Par exemple : la norme POSIX de 1988 a influencé TAFIM développé dans les années 90, ce dernier a influencé lui-même TOGAF créé en 1995. L'organisation du schéma est géographique : à gauche les **USA**, à droite l'**Europe**, avec un zoom sur la **France**. Ce choix de représentation permet de voir que les racines étaient originellement propres à

[13] Traduisible de l'anglais par *Renseignement militaire*

chaque région du monde pour peu à peu converger. Nous avons réalisé une classification par nature des organismes à l'origine des démarches : **gouvernemental**, **département de la défense**, et **industriel**. Les organismes industriels n'ont pas systématiquement une participation liée à un but commercial, par exemple TOGAF est un groupement pour avoir une norme et des pratiques communes, on peut utiliser TOGAF sans licence. À contrario, ARIS propose un cadre utilisable uniquement avec ses logiciels. Nous remarquons que toutes les démarches, malgré les différences de culture entre continents, ont fini par se recroiser et s'influencer : **TOGAF est la méthode cœur**, au centre de la figure 2.6. TOGAF a su fédérer les bonnes pratiques de toutes les démarches.

Dans la suite, nous donnerons un aperçu des méthodes contemporaines, ainsi que la définition des principaux termes dans la littérature environnante de l'architecture d'entreprise.

2.3.2 Urbanisation : la métaphore de la cité

L'urbanisation est une méthode qui a les mêmes objectifs que l'architecture d'entreprise : **encadrer** et **assister** l'évolution du système d'information, **aligner** la stratégie au système. Bien que l'urbanisation caractérise fortement la méthode définie par Christophe **Longépé** et les travaux de l'association Club Urba SI créée en 2000 par Axa, RATP, Lyonnaise des eaux, FNAC et ORESYS. Le terme urbanisation est devenu un **synonyme** d'Architecture d'entreprise, et dans la littérature nous pouvons retrouver le terme urbanisation dans le contexte d'utilisation de la méthode anglo-saxonne TOGAF. Une preuve significative est la juxtaposition des deux termes lors du changement du nom du Club Urba SI « club des urbanistes et architectes des systèmes d'information » en club URBA-EA l'année 2006 « URBA Urbanisme des SI - Enterprise Architecture ». Dans cette thèse nous avons choisi le terme Architecture d'Entreprise que nous considérons plus générique, englobant la méthode initiale d'urbanisation franco-française.

L'urbanisation du système d'information est issue de l'appropriation de termes et méthodes de la discipline d'urbanisme de la ville et de l'aménagement des territoires.

DÉFINITION 2.4 – Plan d'occupation des sols (POS)

Les POS ont pour objet premier de définir de façon précise les droits attachés à chaque parcelle, mais aussi d'organiser le tissu urbain en définissant la destination des constructions, les densités et éventuellement les formules applicables, de localiser les emplacements réservés pour la réalisation des équipements et de protéger les espaces naturels ou agricoles.

Dictionnaire de l'urbanisme et de l'aménagement - Pierre Merlin et Françoise Choay [73]

Le POS vise à **cartographier** l'ensemble du système d'information à l'aide d'une **nomenclature granulaire en blocs** et sous-blocs. La nomenclature proposée par Longépé [67] suit la hiérarchie : **Zone, Quartier et Îlot**. Ainsi, nous retrouvons une analogie avec le découpage de nos villes : zone commerciale, zone industrielle, zone tertiaire, quartier résidentiel...

Un exemple de POS provenant du SI de l'état français est illustré par la figure 2.12 dans la section 2.3.4.

Note

- Nous proposons notre propre méta-modèle dans le chapitre 3 que nous appelons fonctionnel en référence au découpage de l'entreprise en fonctions.
- Le terme POS n'existe plus en terminologie d'urbanisme de la cité, il a été remplacé par le PLU (Plan local d'urbanisme) en 2000 par la loi SRU, mais a été conservé pour la discipline des systèmes d'information.

L'urbanisation ne se résume pas à une simple cartographie du SI, mais elle est un cadre complet d'architecture d'entreprise. Il s'agit du POS qui vise à formaliser le SI par la cartographie. Le POS est utilisé dans les étapes d'**analyse de l'existant** et de **définition de la cible**. Ces étapes s'inscrivent dans une démarche méthodologique globale : de la capture de la stratégie, au **plan directeur** (ou de convergence) qui définit un scénario d'actions sur le SI, jusqu'à l'exécution de la transformation. On retrouve la démarche d'urbanisation sous le terme **IT City Planning** dans la littérature anglophone [77].

L'urbanisation aborde la notion de **cycle de vie** du SI. Chaque étape compose un **processus global** qui décrit des activités distinctes où **différents acteurs** interviennent, illustré par le tableau 2.1 extrait de Longépé [67].

TABLE 2.1 – Rôles et responsabilités des acteurs de l'urbanisation [67]

Phases		Axes stratégiques		Analyse de l'existant				Définition de la stratégie				Plan de convergence			
Activités / Acteurs	Planification de l'étude	Compréhension de la stratégie métier de l'entreprise	Définition de la vision cible	Cartographie de l'existant	Bilan de l'existant	Étude des opportunités technologiques	Orientation et déclinaison de la stratégie	Plan d'occupation des sols	Prévision des performances	Organisation Cible	Évaluation des scénarios et choix	Finalisation du plan de convergence	Définition et mise en place de la stratégie suivie	Publication de la stratégie	Mise à jour de la stratégie
Chef de Projet	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Directeur de l'étude	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Équipe d'étude	I	A	A	A	A	A	A	A	A	A	A	A	A	I	I
Comité de direction	I	I	I	I	I	I	I	I	I	I	C	I	I	I	C
Direction métier	I	C	C	C	C	C	C	C	I	I	C	C	I	I	C
Utilisateur	I	C	C	C	C	I	C	I	I	I	C	C	I	I	I
Direction des SI	C	C	C	C	C	C	C	C	C	C	C	C	A	I	C
Responsable qualité	I	C	C	C	C	C	C	C	C	C	C	C	C	I	I
Équipe étude et développement	I			A	I	I	I	I	A	C	C	A	I	I	A
DRH				A			A			A		A		I	
Comité de direction du SI	I	I	I	I	I	I	I	I	I	I	I	I	I	R	R
Sponsor de l'étude	A	C	C	C	C	C	C	C	C	C	C	C	C	A	C
Chef de projet métier	C	C	C	C	C	I	C	I	I	I	C	C	I	I	C
Pôle urbanisme														A	A

R : *est responsable*, A : *est acteur*, C : *est consulté*, I : *est informé*

La figure 2.7 présente les 7 étapes de la démarche d'urbanisation. Elles se réalisent de gauche à droite. Cette représentation en blocs est commune dans la littérature sur l'urbanisation, les blocs étant de même taille, aucune indication ne nous permet de savoir quelle étape demande un travail plus conséquent que les autres.

Pour exécuter le processus d'urbanisation du SI, les étapes nécessitent d'être effectuées de façon successive ou en simultané selon leur nature. Par exemple : la définition de la stratégie peut se réaliser en même temps que l'analyse de l'existant, mais ce n'est qu'une fois que ces deux étapes sont terminées que l'on peut passer au plan de convergence [66].

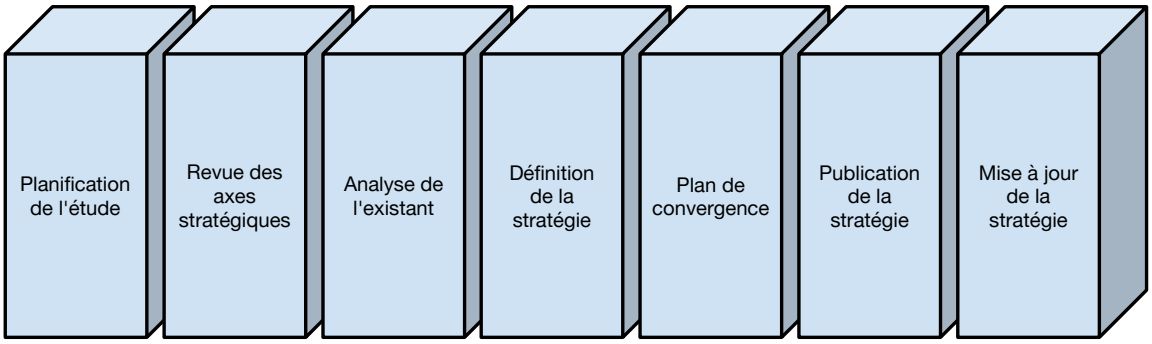


FIGURE 2.7 – Démarche méthodologique de l'urbanisation à la française

DÉFINITION 2.5 – Plan directeur

Le plan directeur, plan d'urbanisation, plan de transformation, plan de convergence (Longépé) [67], planning de migration (TOGAF) [35] ou encore trajectoire. Il s'agit du planning et de l'organisation de la mise en œuvre de la nouvelle architecture. Le plan définit les nouvelles applications, fonctions ou services à développer ou à modifier. Y sont détaillées : la faisabilité technique, organisationnelle, une estimation des coûts, des ressources nécessaires et du temps. Y sont étudiées les contraintes d'intégration avec l'existant et la migration pour basculer d'un système à l'autre.

Les entreprises peuvent éprouver des difficultés à mettre en place le processus d'urbanisation. Yves Caseau [21] constate que les entreprises s'intéressent très souvent aux premières étapes, mais finissent par abandonner le processus une fois l'analyse de l'existant terminée. La documentation générée peut être trop complexe et trop importante.

Note

Nous avons rencontré un cas similaire lors de nos expérimentations industrielles. Une cartographie d'une partie du SI avait été réalisée de façon très détaillée, mais n'a ensuite jamais été utilisée, ni même mise à jour.

Il est difficile de s'attaquer au SI en entier, il est plutôt recommandé d'appliquer la méthode d'urbanisation de façon itérative. Pour que l'urbanisation soit un succès, tous les acteurs de l'entreprise, métier et informatique, doivent être partie-prenante.

Nous abordons maintenant la démarche consœur de l'urbanisation : l'architecture d'entreprise (AE). L'AE était une démarche d'origine anglo-saxonne avant d'émerger également en France et de concurrencer l'urbanisation du SI.

2.3.3 Différents cadres d'architecture d'entreprise

L'historique de la section 2.3.1 nous a permis de constater que différents courants de cadre d'architecture ont émergé et évolué. L'étude des principales démarches dans les sections suivantes va nous permettre une meilleure compréhension des objectifs et des étapes d'exécution d'un cadre d'architecture.

Zachmann : le cadre précurseur

John Zachman propose dans son framework (*Zachman Framework for Enterprise Architecture* - ZFEA) une représentation de l'architecture d'entreprise organisée sous forme **matricielle** (cf. figure 2.8) avec en ligne les différents **points de vue** et les acteurs responsables, puis en colonne les différents **concepts** répondant aux questions circonstancielles : Quoi ? Comment ? Où ? Qui ? Quand ? Pourquoi ? [74]. La genèse de cette approche par questions remonte à la méthode *Five W's* utilisée dans le journalisme américain, un article de journal doit systématiquement répondre aux questions : *Who, What, Where, When, Why* ? Le choix d'ajouter *How* par Zachman est une influence de la méthode BSP créée par son professeur Duane Dewey Walker qui fait la distinction entre *how* (le processus), *who* (l'organisation) et *what* (la donnée) [53]. En effet une organisation peut être modifiée sans impact sur les processus, de plus les données peuvent avoir des sens différents selon leur utilisation. En français la variante QQOQCCP (Qui ? Quoi ? Où ? Quand ? Comment ? Combien ? Pourquoi ?) est utilisée comme méthode empirique en sciences de l'information et de la communication : *benchmarking*, roue de Demming, analyse qualité TQM, méthodes agiles...

Note Pour aller plus loin, un domaine d'étude ontologique existe autour de la terminologie 5W1H [59].

	Quoi ? Donnée	Comment ? Fonction	Où ? Réseau	Qui ? Personne	Quand ? Temps	Pourquoi ? Objectif	
Périmètre *Contexte*							Stratège
Modèle métier *Conceptuel*							Propriétaire
Modèle du système *Logique*							Concepteur
Modèle technologique *Physique*							Développeur
Représentation détaillée *Intégration*							Sous-traitant

FIGURE 2.8 – Le cadriciel Zachman

La force du cadriciel ZFEA fut d'ajouter les différents points de vue en plus de ces questions. La figure 2.8 illustre les différents points de vue en en-tête de ligne, l'acteur en fin de ligne, et les questions en colonne.

Le périmètre est le point de vue du *stratège*, il délimite l'architecture et les représentations sur tout ou partie de l'entreprise.

Le modèle métier est le point de vue du propriétaire, il décrit des processus et des objets métier.

Le modèle système est le point de vue du concepteur, il définit des modèles d'architectures.

Le modèle technologique est le point de vue du développeur, il représente les modèles utilisés pour développer le système.

La représentation détaillée est le point de vue des sous-traitants, il décrit le produit final et ses composants extérieurs pour l'intégration à l'environnement de production.

Un dernier point de vue (non représenté sur la figure) décrit le système en lui-même.

Chaque cellule est donc un modèle, un livrable, un document ; elle peut être indépendante ou en lien avec une autre. En colonne, il s'agit de niveaux d'abstraction différents, un même élément est présenté avec un niveau de détail différent, un lien est donc implicite. Néanmoins, un composant d'architecture du système doit figurer dans une seule et même cellule.

La limite du framework ZFEA est qu'il ne dispose d'aucun méta-modèle, ni même d'une méthodologie pour élaborer les modèles [67]. Bien qu'il ait inspiré les autres cadres d'architecture (par exemple EAP, cf. figure 2.6), il est nécessaire de compléter ZFEA avec des notations standards et de bonnes pratiques issues d'autres cadres. ZFEA est aujourd'hui peu à peu abandonné pour ses défauts : pas de description en détail des étapes ; pas de modèles proposés. Zachman répond à ces attaques et réintègre ses travaux dans le contexte actuel de l'architecture d'entreprise [63]. Il semble conscient que TOGAF est désormais leader sur le marché des cadriciels AE. Lapalme, promoteur de ZFEA pointe vers un article de Gerber démontrant des faiblesses de TOGAF 9 par « *l'ambiguïté et l'inconsistance* » de ses méta-modèles.

Dans la suite, nous présentons le cadre d'architecture qui a de nos jours le plus d'adhésion : TOGAF.

TOGAF : le cadre contemporain

The Open Group Architecture Framework (TOGAF) est un ensemble de concepts développés depuis le milieu des années 1990 par le consortium industriel et international *Open Group*. TOGAF est une méthode générique répondant au besoin de disposer d'un **cadre commun d'architecture d'entreprise** ; le document de référence est composé de 750 pages. Son succès se prouve par la constitution d'une **véritable communauté** d'utilisateurs au fil des années et la croissance du nombre de certifications **à travers le monde** : 50 000 atteintes en 2015, 12 000 uniquement lors de cette même année (figure 2.9)[14]. La France a passé le chiffre de 2 000 professionnels certifiés TOGAF.

TOGAF couvre une **vision** large de l'architecture d'entreprise depuis les aspects **stratégique**, **métier** et **organisationnel** jusqu'aux préoccupations liées au système informatique. TOGAF détaille ainsi la collaboration entre les différents **acteurs** de l'architecture d'entreprise : la gestion et la mise en place des différentes équipes. Cette collaboration suit le cycle *Architecture Development Method* (**ADM**) qui est le cœur de la méthode TOGAF.

[14] Source : http://www.opengroup.org

Les différentes étapes du cycle ADM sont présentées sur la figure 2.10. Le cycle comprend **huit phases** nommées de A à H, précédées par une étape préliminaire de **définition**. L'ensemble de ces étapes constitue le cadre de progression du projet de transformation de l'architecture. Chaque étape définit les **objectifs**, les **cibles**, les **documents** nécessaires ou produits, et le **rôle** des participants.

Note

Nous remarquerons sur la figure 2.10 que les étapes B, C et D sont les étapes de cartographie correspondant à différents points de vue du SI : métier, applicatif et technique. Ainsi, nous retrouvons aussi dans la démarche TOGAF les différents points de vue de la pile du SI présentée initialement en figure 2.2.

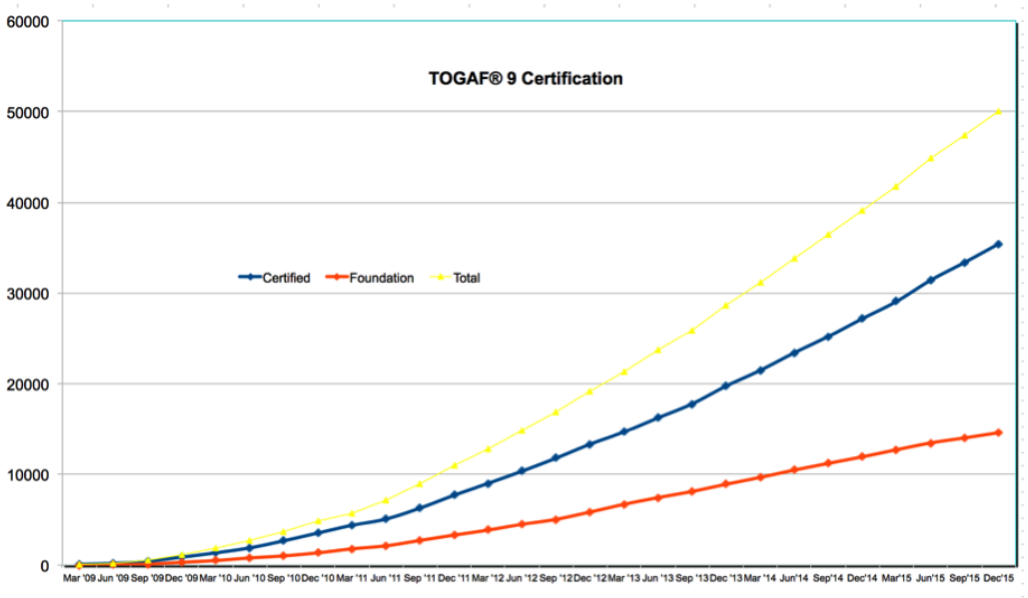


FIGURE 2.9 – Évolution du nombre de certifications TOGAF 9 entre 2009 et 2015

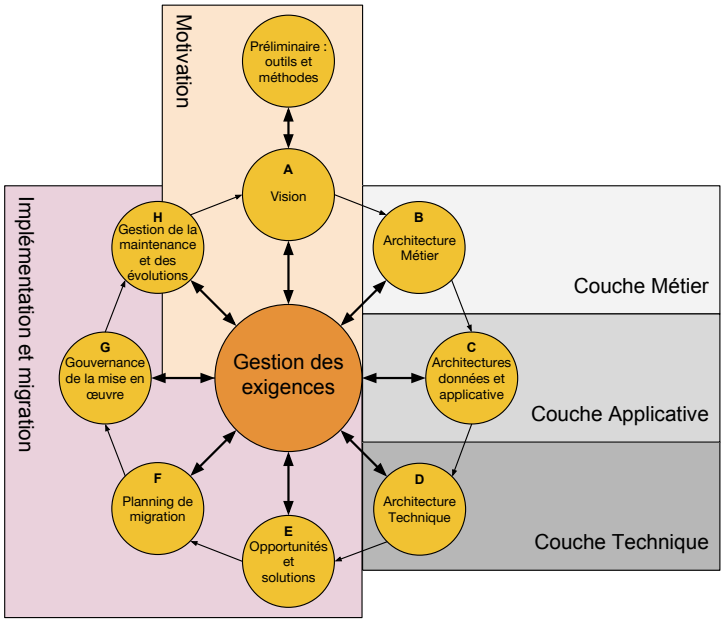


FIGURE 2.10 – Le cycle ADM de TOGAF [35]

TOGAF est souvent nommé par l'expression "*référentiel TOGAF*", car la méthode détaille de nombreux modèles, normes, règles, exemples, guides à exploiter selon le

contexte. La gestion du référentiel est une pratique continue qui s'inscrit dans l'ADM [35] ; à chaque fin de phase, certains éléments sont sélectionnés pour alimenter le référentiel.

TOGAF propose une structure prédéfinie pour le référentiel :

- le méta-modèle des éléments et de leurs relations ;
- la cartographie décrivant l'architecture existante, la bibliothèque qui emmagasine les guides et patrons de conception réutilisables ;
- la base de standards avec les normes et outils à respecter ;
- la gouvernance composée du journal des activités et l'organisation. Elle est sous la responsabilité d'un comité architectural restreint pour un fonctionnement optimisé.

Le document TOGAF est découpé en sept parties pour une lecture thématique en fonction des besoins de l'architecte d'entreprise : Introduction (1), ADM (2), ADM guidelines (3), Architecture Content (4), Enterprise continuum and tools (5), Reference Models (6), Architecture Capability Framework (7). En partie 2, on retrouve la description de la méthode principale, accompagnée de bonnes pratiques et d'exemples en partie 3. La partie 4 contient les supports de travail et livrables qui vont être produits au cours du processus d'élaboration de l'architecture. Les parties 5 et 6 constituent le référentiel avec des typologies, classements et outils.

La suite est la présentation d'un autre cadre commun : le cadre commun de l'urbanisation de la direction interministérielle française des systèmes d'information.

2.3.4 Cadre commun d'urbanisation, l'essentiel

Paru fin 2012, le *Cadre Commun d'Urbanisation du Système d'Information de l'État* (CCU) a été mis en place par la *Direction interministérielle des systèmes d'information et de communication* (**DISIC**) de l'état français [40]. L'objectif étant d'**uniformiser les pratiques** d'architecture de SI au sein des ministères, puisque chacun avait ses propres pratiques et choix technologiques. Il y a également une volonté d'**inter-connexion** entre les différents ministères, ainsi qu'avec les différentes instances territoriales, nationales, européennes, et même jusqu'aux usagers : privés, professionnels et associatifs.

EXEMPLE 2.2 – Déclaration des revenus
Un exemple de nécessité d'inter-connexion est la collecte de l'impôt sur le revenu. Elle est centralisée par la *Direction générale des Finances publiques* dépendant du ministère des économies et des finances, chaque année le contribuable reçoit sa déclaration pré-remplie. Les données proviennent de parties tierces : les salaires sont communiqués par les employeurs, et les revenus de capitaux sont transmis par les banques.

Le constat est issu de 30 années de construction sans concertation des SI avec des **technologies variées** et encore souvent **dépassées**. La mise en place de la DISIC se réalise dans un contexte de **restriction budgétaire** et vise ainsi à normaliser toutes les pratiques. Le document du cadre commun réalise une synthèse des **innovations** des 10 dernières années en terme d'architecture de SI. Il est précisé que les termes «urbanisation» et «architecture d'entreprise» sont convergents, néanmoins «urbanisation» a été conservé dans le CCU par son utilisation historique au sein de l'administration française, cependant en cas de traduction en anglais du cadre commun, le terme anglais "entreprise architecture" serait adopté.

La démarche contribue à la **transformation** du SI de l'état et tente de permettre d'**aligner** le système d'information et de communication sur la **stratégie** de l'état. La **trajectoire** (cf. définition 2.3.2) du SI est considérée comme l'un des principaux livrables, elle s'appuie sur une nomenclature précise définie par le cadre : le plan d'occupation des sols et la cartographie fonctionnelle.

Le CCU met à disposition le POS fonctionnel de l'état en exemple, figure 2.12. Il y a 5 zones principales :

Échange en vert correspond à la communication interne et externe : messagerie et téléphonie

Pilotage & Contrôle en orange, désigne les blocs fonctionnels dédiés à la stratégie et aux décisions. Par exemple, les aspects contrôle du budget et politique de sécurité sont inclus dans cette zone.

Données en gris, il s'agit de toutes les données échangées entre les ministères et les interlocuteurs : entreprises, collectivités, usagers...

Ressource & Support en bleu, il s'agit de toutes les ressources permettant le fonctionnement des ministères : ressources humaines, finance, immobilier, juridique, documentation, etc.

Fonctionnalités en couleur cyan, chaque colonne correspond à un ministère (MAE - *Ministère des Affaires étrangères* ; MEN - *Ministère de l'éducation nationale* ; MAAF - *Ministère de l'agriculture, de l'agroalimentaire et de la forêt*, etc.) avec le détail des blocs fonctionnels dont il est responsable.

Cet exemple donne un bon aperçu de la taille d'un POS fonctionnel d'un SI d'envergure : l'état français composé de ses ministères. Le POS permet de donner une vue d'ensemble complète en un seul diagramme. Néanmoins, le niveau de détail est assez faible, il est nécessaire de faire le même travail pour chaque secteur fonctionnel de chaque ministère.

Le cadre commun est articulé autour de sept parties et trois questions : «Pourquoi ?», «Comment ?», et «Quoi ?» ; comme décrit dans la figure 2.11.

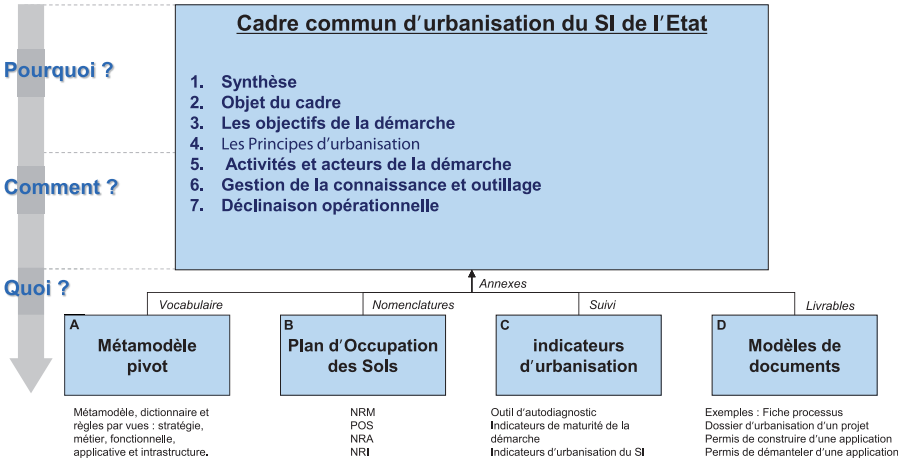


FIGURE 2.11 – Organisation et plan du Cadre Commun d'Urbanisation [40]

La septième partie du CCU est composée d'annexes (rectangles A, B, C et D sur la figure 2.11), dont une partie nommée "***méta-modèle pivot***" comportant la description des différents points de vues du SI : métier, fonctionnel, applicatif et technique. La pile du SI est identique à notre figure 2.2.

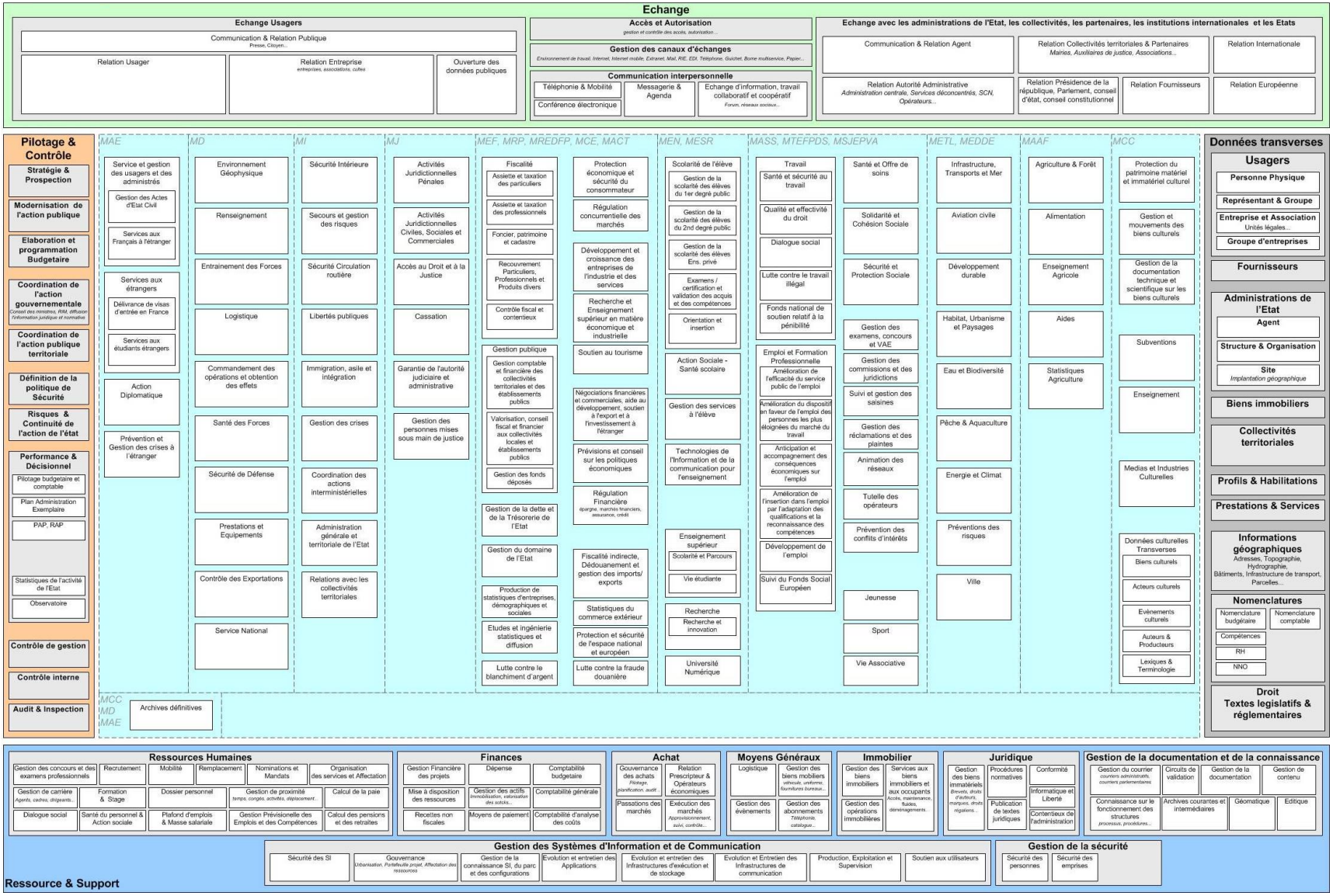


FIGURE 2.12 – Le POS de l'état [40]

Le CCU donne une **définition** précise de tous les objets contenus dans chaque **méta-modèle**, et les liens entre les différents objets qui peuvent traverser une ou plusieurs couches. La définition par le CCU de la vue fonctionnelle, en interaction avec les vues métier et applicative, est illustrée par l'exemple 2.3.

EXEMPLE 2.3 – La définition de la vue fonctionnelle

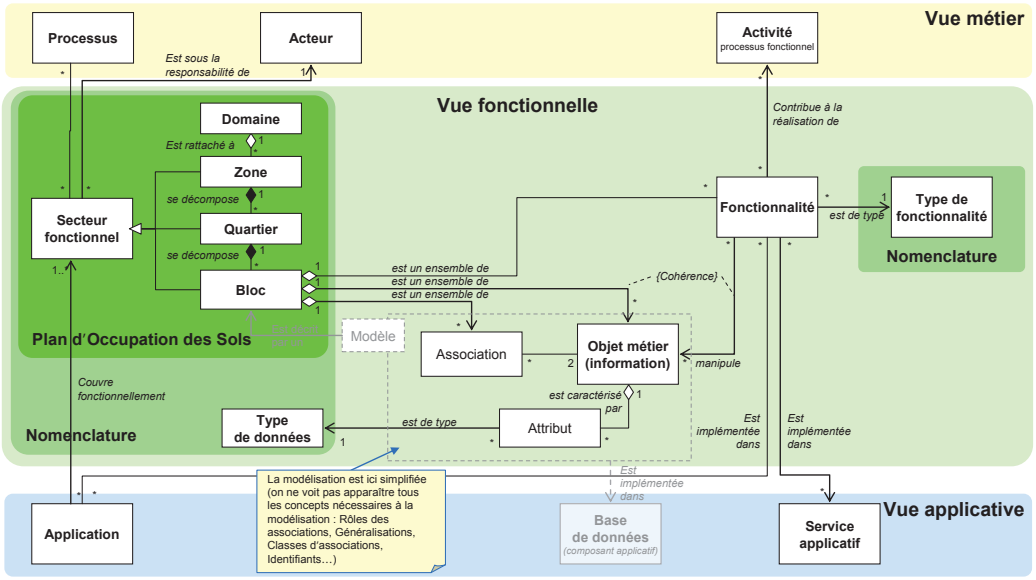


FIGURE 2.13 – Les concepts de la vue fonctionnelle [40]

Les enseignements et recommandations du document, bien qu'il cible les instances de l'état, peuvent être transposés pour un SI d'entreprise privée. En effet, les problématiques budgétaires, organisationnelles, règles d'urbanismes, points de vue du SI et les technologies sont en tous points partagées. Dans la figure 2.12 représentant le POS de l'état, la plupart des blocs ont une fonction présente dans une structure d'entreprise : contrôle de gestion, achat, fournisseurs, ressources humaines, etc. Seuls les blocs fonctionnels représentant les ministères sont très spécifiques à l'organisation de l'état français, mais on peut trouver aisément une structure en directions équivalentes dans une entreprise, par exemple : une grande entreprise est découpée en filiales, un groupe multinational en sociétés...

2.3.5 Comparaison des approches

Zachman permet d'organiser les modèles que l'on souhaite générer selon les différents **points de vue** et **questions** que l'on se pose par rapport au SI. Néanmoins, le cycle de vie du SI et la mise en pratique de l'architecture ne sont pas abordés.

L'**urbanisation à la française** est la démarche dont la **méthodologie** et le **cycle de vie** sont les plus faciles à comprendre et à mettre en place. Il s'agit de la démarche détaillant le plus la modélisation de la **couche fonctionnelle** et ses règles d'urbanisme. Bien que cette couche permette un niveau d'abstraction très utile pour représenter le SI, elle est utilisée en majorité par les francophones (IT City planning pour les anglophones). TOGAF ne spécifie pas de vue fonctionnelle distincte, les concepts fonctionnels sont dilués dans la couche métier.

Concernant la démarche d'urbanisation, nous émettons des réserves sur les étapes de cartographie processus et applicative : la démarche date d'avant l'émergence forte

de la notation BPMN et des architectures orientées services bien que des adaptations soient possibles et décrites. L'urbanisation décrit les architectures applicatives de la même manière qu'un POS fonctionnel ce qui ne permet d'obtenir qu'une vue très macroscopique. Pour une mise en place à partir de zéro, des choix plus contemporains sont possibles à puiser dans les autres démarches ou standards de méta-modèle.

Note Nous proposerons notre propre méta-modèle applicatif dans le chapitre 3.

TOGAF est devenu le **cadre** d'architecture d'entreprise **universel** et un **standard** incontournable pour mettre en place une nouvelle pratique de l'architecture dans une entreprise. La littérature et le nombre d'outils compatibles permettent d'appliquer TOGAF dans toute situation. Cependant, il est possible de se perdre très rapidement dans tous les concepts proposés. Une **montée en compétence** importante et des choix s'imposent, il n'est pas possible de mettre en place la totalité de TOGAF. À la différence de la démarche d'urbanisation, TOGAF distingue une couche du SI représentant à part les **données**, tandis que ces dernières sont diluées dans les différentes couches dans l'urbanisation. Un choix s'impose. Archimate représente les données à différents niveaux ; la donnée a ainsi une abstraction différente selon le point de vue et suit un raisonnement d'abstraction similaire à la représentation Zachman.

Note Nous avons fait le choix de représenter les données dans chacun de nos méta-modèles (cf. section 3).

Une autre différence entre TOGAF et l'urbanisation se situe sur le cycle de vie, la roue TOGAF spécialise les différentes couches à modéliser en autant d'étapes, au contraire de l'urbanisation qui ne propose qu'une seule phase d'analyse de l'existant. TOGAF permet des itérations plus courtes pour l'évolution du SI et une comparaison systématique entre existant et cible[15].

Le cadre commun d'urbanisation fait la **synthèse** des principales pratiques d'architecture d'entreprise notamment TOGAF, Archimate, Club Urba, mais aussi les équivalents gouvernementaux des États-Unis d'Amérique (*The common approach to Federal Enterprise Architecture*), du Royaume-Uni (*HM Government Information Principles*) et de l'Australie (*Australian Government Architecture Reference Models and Interoperability Framework*). Le CCU se révèle être une synthèse **facile à aborder** pour pratiquer l'architecture d'entreprise avec un **niveau de détail important**. Néanmoins, le CCU n'est pas totalement exhaustif, il sera nécessaire d'étendre des points précis avec d'autres bonnes pratiques comme TOGAF. Le deuxième inconvénient, ou force selon la position de l'utilisateur, est que le CCU n'est proposé qu'en français. Les racines du CCU proviennent du Club Urba et Longépé, ce qui leur confère une orientation franco-française.

En ce qui concerne l'application des méthodes dans l'industrie, jusque dans les années 2000 l'urbanisation était la référence en France, notamment par la promotion faite par le CIGREF et le Club Urba. Mais depuis la décennie en cours ce n'est plus le cas, l'intensification des échanges électroniques à l'international, mais aussi la nouvelle génération d'experts informatiques plus familière à la culture anglo-saxonne ont permis une **adoption accélérée** de TOGAF. Néanmoins, l'urbanisation restera encore majoritaire pour de nombreuses années dans les secteurs au **patrimoine**[16] important et dont les

[15] *As-is / to-be* en anglais
[16] *Legacy* en anglais

cycles de changement sont longs : mutuelle, bancaire, GIE, administration publique... Pour exemple, Nicolas Dieudonné utilise la démarche d'urbanisation dans la grande entreprise AREVA [38].

Toutes les méthodes de transformation du SI exposées précédemment (urbanisation ou architecture d'entreprise) sont complémentaires. Elles sont le fruit de décennies d'expérimentation et tendent à converger. Il est nécessaire de tirer profit de chaque démarche. L'entreprise ne doit pas s'adapter à un cadre d'architecture, ce sont les bonnes pratiques qui doivent être collectées à partir des différents cadres pour s'adapter à la philosophie, à l'organisation et au fonctionnement de l'entreprise et de ses acteurs.

2.4 L'alignement du SI

L'alignement vise à mettre en cohérence et correspondance les différents points de vue du SI. Cette notion d'alignement a été évoquée lorsque nous avons précédemment abordé les outils de la gouvernance et les cadres d'architecture d'entreprise. Nous allons donc définir à présent l'alignement. Derrière cette désignation on distingue deux problématiques très différentes mais complémentaires : l'alignement spatial et l'alignement temporel.

2.4.1 Alignement spatial

L'alignement spatial est l'alignement entre les **différentes couches** du SI. Comme évoqué lors de la présentation de la pile de point de vues du SI (cf. section 2.2) et du cadre de Zachman, le passage d'un point de vue à l'autre est un problème d'**abstraction** (du bas vers le haut, *bottom-up* en anglais) ou de **concrétisation** (du haut vers le bas, *top-down* en anglais). Chaque couche représente bien le même SI mais vu sous des **angles différents**. Ainsi à travers les différents objets de chaque couche, l'alignement consiste à créer des **liens** pour mettre en correspondance des **objets similaires** disséminés dans les différentes couches.

L'alignement spatial du SI est couvert par un domaine d'étude de l'alignement largement rencontré dans la littérature anglophone sous la terminologie "*Business-IT aligment*" (**BITA**) [9, 24, 68, 98]. Ce domaine d'étude vise principalement à réconcilier les domaines **métier** et **informatique**, il est constaté de par la nature des informations, de par le vocabulaire des acteurs qu'un **fossé** sépare ces deux visions du SI.

Un exemple de méthode BITA est celle de Chen *et al. Business IT Alignment Method* (BITAM) [25] composée de 12 étapes de la modélisation de la nouvelle stratégie, choix des scénarios opérationnels, réalignement du SI, jusqu'à l'évaluation de l'investissement provoquée par la nouvelle stratégie.

Le terme BITA est néanmoins réutilisé à des niveaux d'abstraction différents et n'a pas de définition universelle, ainsi on pourra également retrouver la notion de BITA sur des sujets de gouvernance du SI.

2.4.2 Alignement temporel

L'alignement temporel est une **comparaison** entre une seule et même cartographie figée à deux instants différents du SI : **source** / *As-Is* et **cible** / *To-Be*. La comparaison isole les différences et les contenus communs pour **détecter** ce qui a été modifié, ajouté ou supprimé. Les outils pour réaliser cet alignement réalisent une **analyse d'écart**.

Anne Etien parle de **co-évolution** [44] et propose la méthode ACEM résumée par la figure 2.14. ACEM nécessite une première étape d'unification des cartographies cible et source vers un modèle pivot unique pour permettre la comparaison. Cette première étape se rapporte à la problématique d'alignement spatial (abstraction / concrétisation). Un mauvais alignement spatial met en péril l'analyse de l'évolution. De plus le fait d'unifier les modèles de chaque côté (source et cible) complexifie l'identification des concepts réellement impactés.

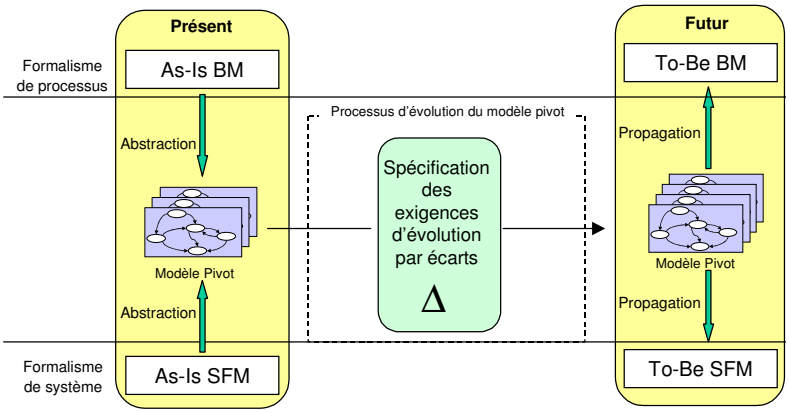


FIGURE 2.14 – L'approche ACEM de co-évolution [44]

Plusieurs solutions techniques sont proposées pour résoudre le problème de co-évolution. Flores propose la plateforme ASIMOV [47] pour la création de modèles et méta-modèles qui intègrent d'origine des concepts liés à la problématique de co-évolution. ASIMOV fournit deux langages *ASIMOV Evolution* et *ASIMOV Assistance* pour spécifier les changements à réaliser dans le méta-modèle à l'aide de scripts. *ASIMOV Engine* permet de les exécuter. Mantz propose une solution totalement différente basée sur la transformation de graphes [69].

2.4.3 Alignement stratégique

En dehors des définitions qui couvrent l'évolution spatio-temporelle du SI, nous retrouvons dans la littérature le terme d'alignement stratégique [83]. Selon Caseau [21], lors de la démarche d'urbanisation, le poids de l'étape de définition de la stratégie d'entreprise peut être tel que cette modélisation peut occulter facilement les **enjeux techniques et opérationnels**. La mise en place de **processus** dans une entreprise est un **changement culturel** qui peut engendrer un risque dans le démarrage des travaux de transformation du SI. Il est donc de nouveau nécessaire de rappeler l'importance de travailler par **itération**. Réduire l'alignement à un alignement stratégique est donc dangereux dans un projet d'architecture. Caseau [21] rappelle que les enjeux stratégiques du DSI : achats, compétences, coûts, renouvellements du parc, infrastructures ne font pas partie de l'architecture d'entreprise, mais sont des problèmes de gouvernance du SI. G. Jean [56] distingue à ce propos l'urbanisation de l'urbanisme : l'**urbanisation** est la **phase opérationnelle** et technique sous la responsabilité des informaticiens, l'**urbanisme** est la **phase amont** d'organisation des processus, activités, fonctions et objets métier. La déconnexion de ces deux phases provoque une nouvelle fois des risques de réaliser un urbanisme trop théorique et irréalisable.

L'alignement stratégique peut donc être considéré de deux manières :

- soit il s'agit d'une phase de l'alignement du SI, où l'on aligne les points de vue stratégiques avec les points de vue fonctionnel et applicatif.
- soit il s'agit d'une phase en amont pour aligner la stratégie à terme (court, moyen, long) avec les acteurs internes et externes au SI.

Dans le second cas, Thevenet [96] propose le modèle *Stategic MAP* et définit le méta-modèle à la figure 2.15. La vue stratégique définit des visions pour satisfaire les acteurs. La vision est élaborée à l'aide d'un plan de stratégie décrivant des buts pour atteindre les objectifs.

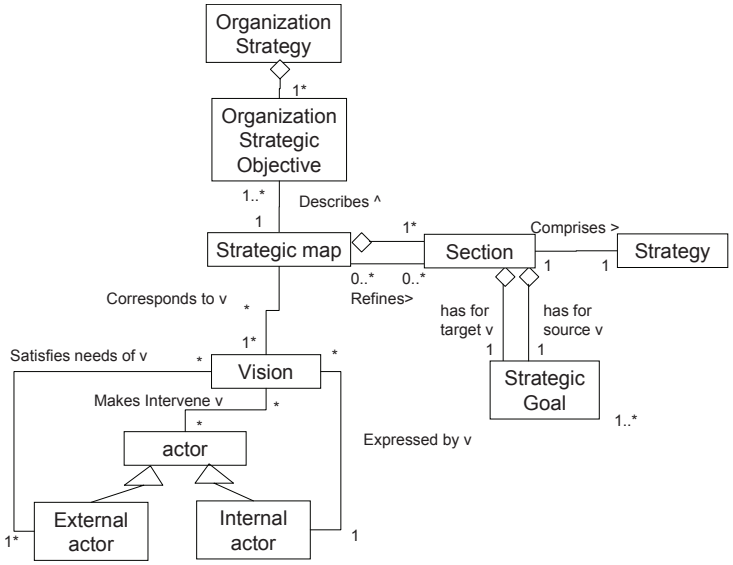


FIGURE 2.15 – Le méta-modèle Strategic MAP [96]

2.4.4 Méthodes d'alignement mixte

L'alignement est un sujet complexe qui peut aborder plusieurs préoccupations du SI : stratégie, métier, informatique.

Par exemple l'alignement doit permettre de sortir du cadre du SI unique et intégrer plusieurs SI, notamment dans le contexte d'une fusion acquisition de plusieurs entreprises. Baker et Niederman ont étudié l'intégration des fonctions du SI par alignement BITA auprès d'entreprises américaines [11]. Il existe plusieurs stratégies de fusion : la co-existence avec réplication des données, ou l'absorption pour garder uniquement l'un des SI. Mais l'alignement ne consiste pas seulement à aligner les deux SI, une fusion modifie la stratégie en elle-même de la nouvelle entité. Ainsi, deux alignements sont nécessaires, l'un pour accorder les SI, l'autre pour s'adapter à la nouvelle organisation.

D'autres classifications d'alignement peuvent être considérées. Dans la littérature anglophone, l'approche la plus remarquée et étudiée est le Modèle Stratégique d'Alignement (SAM) Henderson et Venkatraman [52, 17] illustré par la figure 2.16 qui distingue d'une part la stratégie métier de la stratégie informatique et d'autre part l'organisation du SI de son architecture technique. De nombreux travaux ont été réalisés autour de SAM afin de le revisiter [10] ou de l'étendre [33] avec les préoccupations techniques actuelles.

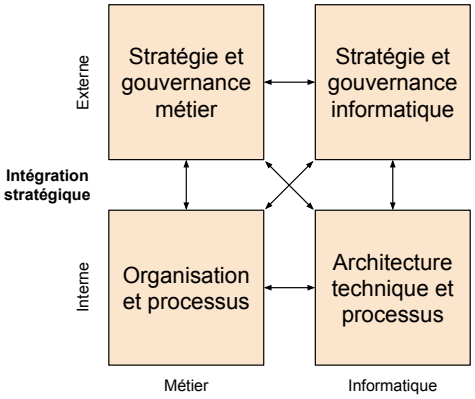


FIGURE 2.16 – Méthode SAM

Thevenet propose la méthode *INtentional STrategic ALignment* (INSTAL) [95] à la frontière entre les niveaux stratégique et opérationnel par une modélisation explicite de l'alignement afin de guider l'évolution et les corrections à effectuer sur les nouvelles exigences de façon itérative.

L'approche *Systemic Enterprise Architecture Method* (SEAM) de Wegmann [102] s'intéresse à l'alignement entre trois entités : l'environnement, les processus métier et le système. L'approche SEAM s'inscrit dans un contexte d'évolution, le but étant de construire la cible dans laquelle le SI est aligné au marché.

Svärdström et Magoulas [83] distinguent dans *Framework for Understanding Enterprise Morphology* (FEM) illustré par la figure 2.17, cinq types d'alignement : info-logiciel, socio-structurel, fonctionnel, socio-culturel et contextuel.

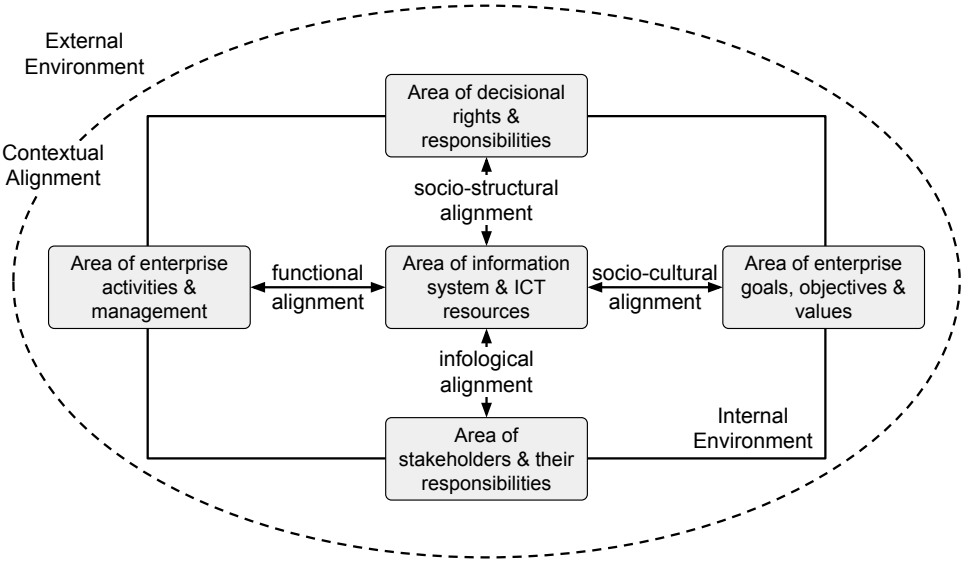


FIGURE 2.17 – FEM - Svärdström et Magoulas [83]

La méthode GRAAL définit également une dimension sociale et réinterprète le modèle SAM [41]. GRAAL distingue trois dimensions : sociale (l'entreprise), physique (infrastructure), et symbolique (logicielle). La méthode GRAAL propose un alignement deux à deux entre chaque monde [62].

2.4.5 Bilan

L'alignement est un terme générique qui désigne la mise en correspondance de concepts :

- Soit les concepts sont de même nature mais portent sur des versions différentes, il s'agit alors d'une évolution du concept. Sur la figure 2.18 cet alignement temporel est caractérisé par les flèches horizontales vertes. À chaque itération, une nouvelle version du SI est réalisée, l'alignement permet de garder une trace des évolutions entre deux versions de SI.

- Soit les concepts ont des natures différentes et n'appartiennent pas à la même couche, mais ont des similitudes par approximation. Ces alignements spatiaux sont représentés par les flèches verticales sur la figure 2.18. Selon l'objectif, l'alignement couvre plus ou moins de couches. Par exemple, l'alignement stratégique (flèche mauve) s'intéresse uniquement aux couches stratégie et métier.

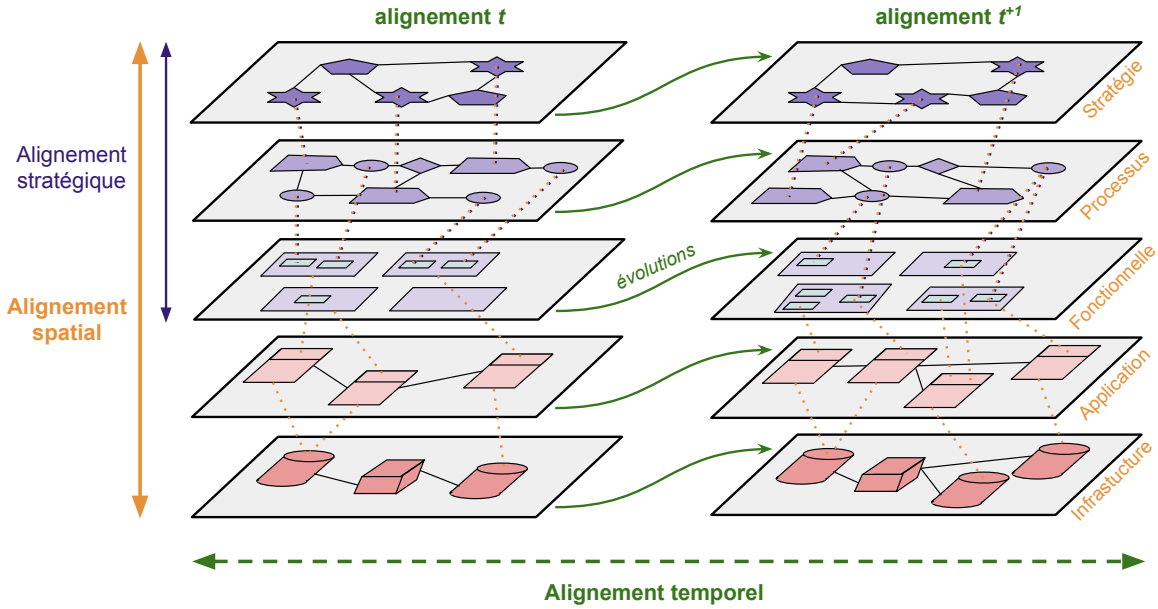


FIGURE 2.18 – Le SI et des alignements possibles

2.5 Le positionnement de la thèse

Dans cette **thèse** nous nous focalisons essentiellement sur l'**alignement** au sens **spatial** et **BITA**, dont nous préciserons notre définition dans le chapitre 4. Le problème de co-évolution n'est ainsi pas abordé, nous considérons qu'il s'agit de l'étape d'après, une fois achevé l'alignement spatial du SI. L'alignement spatial permet une **analyse précise** et fine de l'**existant** en vue de **détecter** les **problèmes** et **tracer les perspectives** du SI **cible**, tandis que l'analyse temporelle permet d'obtenir la trajectoire entre existant et cible.

Dans un premier temps, nous nous concentrons sur les phases préliminaires d'audit par les architectes (phases B, C, D de TOGAF) qui permet de constituer les modèles du SI. Puis dans un second temps, nous réalisons l'alignement spatial de ces modèles. Enfin dans un dernier temps, nous analysons l'alignement pour permettre d'assister l'architecte dans la prise de décision des actions de transformation et donc l'élaboration du plan d'urbanisation ou du planning de migration (phase F de TOGAF, figure 2.10).

Nous ne nous préoccupons pas de l'alignement stratégique qui est une étape comportant la modélisation des processus métier, néanmoins notre alignement spatial réutilise les livrables de cette étape d'alignement stratégique.

Problème

Les cadres étudiés ne définissent pas comment exécuter les différentes étapes d'architecture d'entreprise. Chaque démarche reste conceptuelle et à un niveau organisationnel du projet d'architecture. Nous souhaitons répondre de façon opérationnelle aux problématiques rencontrées sur les étapes de cartographie et d'alignement. Ainsi, la présente thèse propose une **méthodologie appliquée** par les **outils d'ingénierie des modèles** pour assister les phases d'analyse de l'existant.

Solution proposée

Nous proposons une méthode qui consiste à outiller la démarche d'architecture d'entreprise dans les étapes de cartographie, d'alignement et de prise de décision à la transformation du SI. Notre méthode est **générique** et **compatible** avec les principaux cadres d'architecture abordés précédemment. Nous avons vu qu'il n'est pas nécessaire d'adopter un cadre unique, mais qu'il est possible, voire recommandé de prendre les **bonnes pratiques** adaptées à la situation dans différents cadres. Nos travaux n'ont donc aucune adhérence forte avec l'un des cadres présentés.

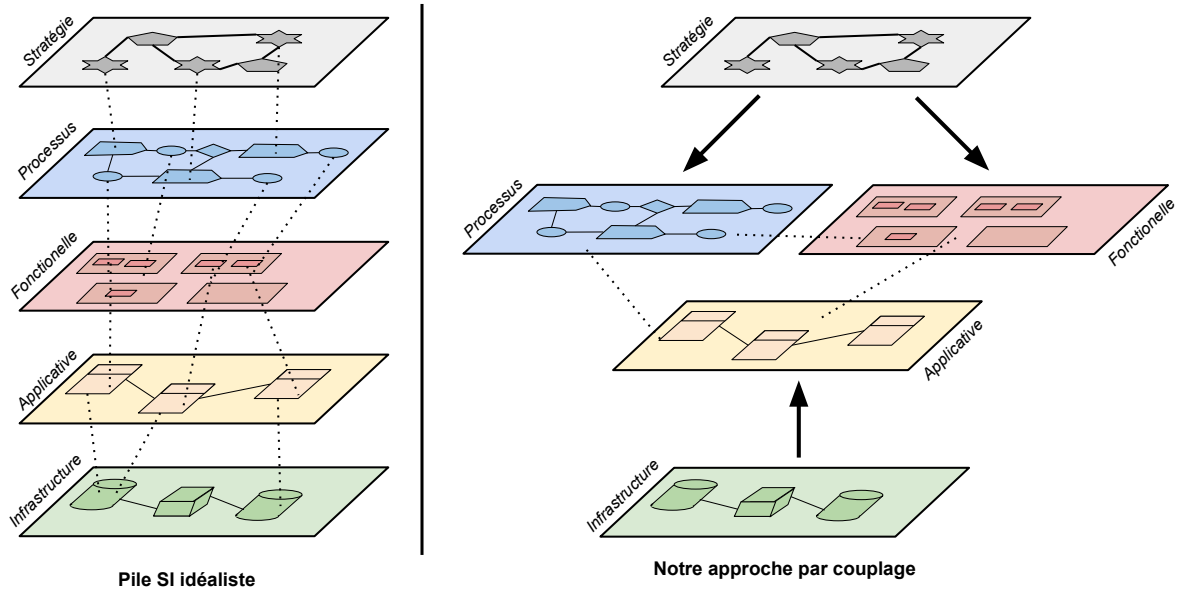


FIGURE 2.19 – Les couches du SI : la pile idéale et l'aperçu de notre approche

Concernant la pile du SI de la figure 2.2, nous verrons par l'expérimentation et l'étude de cas industriel que la **pile** est une **vision idéaliste** d'un SI. Les entreprises n'ont pas toutes le même niveau de **maturité** ni la capacité à s'investir dans un projet d'architecture du SI. Ainsi, certaines entreprises n'abordent que l'essentiel du SI avec les vues applicative et fonctionnelle, tandis que d'autres entreprises poussent l'exercice jusqu'aux couches de processus métier et technique. Un SI est rarement cartographié en entier, le travail d'architecture commence lorsque survient des problèmes : techniques, de maintenance ou de coût. Lorsque les problèmes deviennent majeurs et ne peuvent plus être réparés par l'application de pansement[17] ; un audit complet est nécessaire pour redéfinir une vision stratégique du SI par la cartographie de l'existant. Ainsi, l'architecture va s'attaquer à **une ou plusieurs applications** mises en cause dans l'entreprise et non à l'intégralité du SI.

Notre **vision** du SI est plus **flexible** comme décrit à droite sur la figure 2.19, nous présentons un **couplage** des différentes couches et non une traçabilité qui traverse les couches. Au fur et à mesure de l'avancée du projet d'architecture, il sera toujours possible d'ajouter une couche à l'alignement sans avoir besoin d'une couche intermédiaire comme considérée dans la représentation de la pile idéaliste (à gauche figure 2.19).

[17] *Patch* en anglais

3

Cartographie du SI : les points de vue et méta-modèles associés

Sommaire

Introduction

Dans le chapitre 2, nous avons présenté un état de l'art sur le domaine de l'architecture d'entreprise. Nous avons appris que le système d'information peut être représenté comme une pile composée de différents points de vue : stratégie, processus, fonctionnel, applicatif et infrastructure. L'étude du système d'information pour obtenir une photo du système d'information s'appelle la cartographie. Chaque point de vue peut être représenté par un modèle pour structurer les informations. Les concepts du modèle sont définis dans un méta-modèle (définition 3).

DÉFINITION 3.1 – Méta-modèle

Le méta-modèle est le «modèle du modèle», il est donc le modèle du langage de description de modèles. Le méta-modèle définit les concepts d'un système par le langage constitué de classes, attributs, relations... Il existe différents métalangages pour écrire un méta-modèle, tel le standard de l'OMG **Meta-Object Facility** (**MOF**). Le modèle est une instance du méta-modèle. Les instances de classes sont appelées objets.

Dans l'état de l'art, nous avons également abordé la notion d'alignement du système d'information. Nous avons restreint la problématique à l'alignement spatial entre les points de vue à la frontière entre domaine métier et domaine applicatif (figure 3.1).

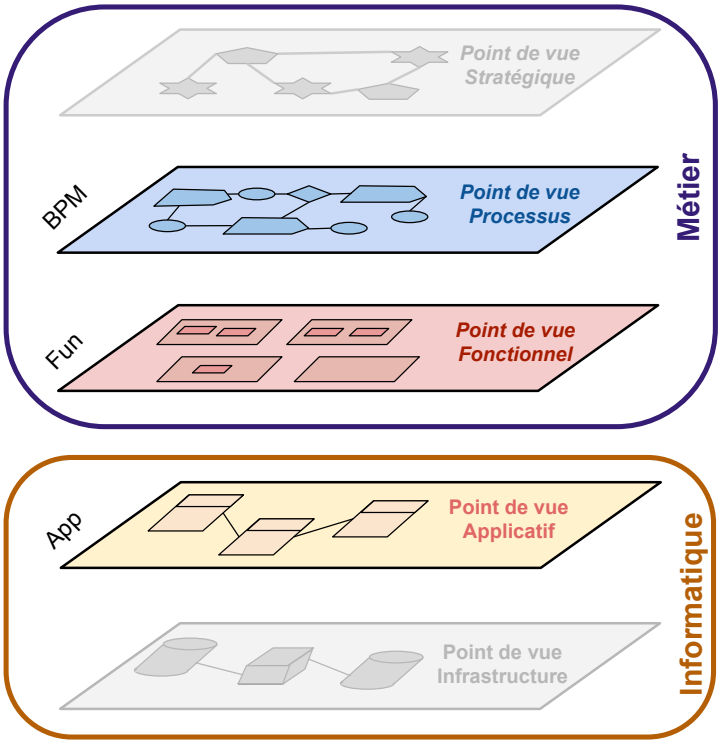


FIGURE 3.1 – Les trois points de vues du SI étudiés

Ainsi, nous réduisons le champ d'étude aux points de vue fonctionnel, processus et applicatif. Nous allons par la suite détailler chaque point de vue de la façon systématique suivante :

- une définition générale textuelle du point de vue ;
- des exemples de diagrammes modélisés dans des notations compatibles au point de vue ;

- un tableau comportant la définition des concepts et des associations ;
- la définition par un méta-modèle de notre conception ;
- un comparatif avec des langages et notations les plus couramment utilisés avec des exemples d'utilisations ;
- une expérimentation de modélisation à l'aide de notre propre méta-modèle.

Chaque point de vue est spécifié par des concepts et leurs relations. Chaque spécification permet ensuite de définir un méta-modèle que nous implémentons avec le projet open source Eclipse EMF. Ce cadriciel constitue le mécanisme de méta-modélisation, il permet de générer le code Java nécessaire à la manipulation des instances de méta-modèle : les modèles. Ces modèles sont le support pour modéliser les informations d'un point de vue. Par exemple, des activités sont créées dans un modèle correspondant au point de vue processus, des blocs sont créés dans un autre modèle correspondant au point de vue fonctionnel, enfin des composants applicatifs sont créés dans un modèle distinct correspondant au point de vue applicatif. Les méta-modèles présentés sont construits de manière à être adaptables et compatibles avec les principaux langages utilisés dans le domaine de l'architecture d'entreprise (ex. : Archimate, Longépé, SOA...) que nous aborderons dans les exemples et comparatifs respectifs.

Pour alimenter et construire les modèles de chaque point de vue, nous avons identifié différents scénarios :

une modélisation manuelle : lors du démarrage d'un nouveau projet d'architecture d'entreprise. L'architecte fait appel aux différentes parties prenantes du SI (direction SI, direction métier, utilisateurs, etc.) afin d'organiser un audit et de recueillir toutes les informations pour cartographier le SI ;

une importation : les informations existantes dans différents supports hétérogènes : un référentiel, une base de données, ou des modèles ; permettent d'alimenter nos points de vue. Cette importation peut nécessiter une étape de transformation de données ;

une rétro-ingénierie : il est possible d'extraire depuis un code source, les concepts du point de vue applicatif, abstraits des spécificités techniques.

La **modélisation manuelle** nécessite d'être réalisée avec l'assistance d'un éditeur graphique. Cet éditeur peut se présenter sous deux formes différentes : une **arborescence** pour créer les instances de concepts par composition ; ou un **diagramme** à plat pour créer les objets graphiquement. Le cadriciel **Eclipse EMF** permet de générer automatiquement l'éditeur par arborescence. En revanche, le diagramme ne peut être généré automatiquement. Nous avons implémenté l'éditeur de diagramme à l'aide du cadriciel **Eclipse GMF** qui permet de définir le comportement associant les éléments graphiques du diagramme et les instances du modèle. Une capture d'écran résultant de l'éditeur de diagramme est présentée pour chaque point de vue dans les sections "expérimentations" respectives.

L'**importation et la rétro-ingénierie** posent une problématique d'ingénierie des modèles : les descriptions des informations entre les supports sources et cibles sont différentes. Pour traduire l'information d'un modèle vers un autre, nous utiliserons la technique de transformation de modèles dans le chapitre 5. Pour réaliser une transformation de modèles, il est nécessaire d'écrire un script qui décrit la traduction des concepts

de modèles sources vers leurs équivalents dans les modèles cibles. Ainsi pour chaque point de vue, nous dressons dans un tableau comparatif les équivalences entre concepts pour passer des notations les plus communément utilisées vers nos méta-modèles.

Pour guider la lecture, nous avons fait le choix d'une nuance de couleur ainsi que d'un diminutif pour raccourcir l'identification, afin de différencier les différents points de vue :

- bleu : processus métier - BPM (*Business Process Model*)
- rouge : fonctionnel - Fun (*Functional*)
- jaune : applicatif - App (*Applicatif*)

Ces couleurs sont utilisées sur les différentes figures : schémas, méta-modèles, diagrammes... telle la figure 3.1. Ce choix est arbitraire, il n'existe ni de norme ni de bonne pratique d'usage sur ce sujet.

3.1 BPM : le point de vue des processus métier

Les points de vue métier sont déclinés en deux représentations (cf. figure 3.1) : description des activités en processus et classification en domaines fonctionnels. Nous allons définir le **point de vue processus métier** dans cette section, puis le point de vue fonctionnel dans la section suivante 3.2.

Les processus métiers sont tout d'abord illustrés par des exemples. Puis, nous dressons une description des différents concepts dans un tableau qui permet de présenter le méta-modèle que nous avons appelé **BPM**. Notre méta-modèle a pour but de faire une synthèse des concepts primordiaux pour modéliser des processus métier. Notre définition constitue une référence unifiée, ou commune. Nous ferons un comparatif pour dresser la compatibilité avec les concepts des normes les plus couramment rencontrées.

3.1.1 Introduction

Le point de vue processus reflète l'organisation des activités d'une entreprise privée ou d'une administration publique sous forme de processus. Le cycle de vie des activités du SI (création, modification, suppression) dépend d'une part des **décisions stratégiques** et d'autre part de l'évolution du **savoir-faire** de l'entreprise. L'entreprise répond stratégiquement à une **demande** des clients de son secteur économique, pour vendre des **services** ou des **produits**. L'offre se conforme à des **normes** ou standards de qualité, et des **lois** provenant des états ou des unions économiques internationales.

Les activités sont réalisées par des **acteurs** qui peuvent être des personnes : clients, législateurs, personnel de l'entreprise, sous-traitants, fournisseurs, distributeurs, etc. ; ou bien des machines ou **systèmes informatiques**. Ces acteurs jouent un rôle **interne** ou **externe** à l'entreprise et peuvent se décliner selon la précision que l'on applique aux processus à modéliser.

Le processus décrit l'enchaînement des **activités** entre les différents acteurs pour réaliser son objectif. Un processus possède un début et une fin, et est déclenché par un **évènement** interne ou externe, manuel ou automatique. Pour interagir avec leur environnement, les processus échangent des messages qui leur permettent de synchroniser

et communiquer des **données**. Par exemple, dans un processus de GRC[1], un client va demander un devis que lui envoie l'entreprise. Si le devis est accepté, l'entreprise fait parvenir un bon de commande. Le client à la suite de la signature du bon de commande, devra effectuer le paiement, et recevra en retour la facture. Ces différents messages comportent des données : référence produit, quantité, adresse, coordonnée bancaire, etc.

DÉFINITION 3.2 – Processus métier

Un processus est un ensemble d'activités entreprises dans un objectif déterminé. La responsabilité d'exécution de tout ou partie des activités par un acteur correspond à un rôle. Le déroulement du processus utilise des ressources et peut être conditionné par des évènements, d'origine interne ou externe. L'agencement des activités correspond à la structure du processus.

Processus Métier et SI, p49 [75]

Le lexique des concepts pour modéliser des traitements peut prêter à confusion : processus, procédure, activité, tâche, opération. Il convient de positionner ces différents termes.

Pour modéliser le système d'information, la méthode Merise [78, 39] distingue d'une part les données des traitements et d'autre part le niveau organisationnel (SIO) du niveau informatique (SII). La figure 3.2 illustre les différents niveaux d'abstraction de la méthode Merise, en rouge les modèles correspondant au contexte de la modélisation processus. Le modèle conceptuel de traitement (MCT) formalise l'activité de l'entreprise sans préciser les ressources ni leur organisation. À contrario, le modèle organisationnel de traitements (MOT) décrit le fonctionnement de l'entreprise en précisant les ressources humaines et matérielles mobilisées ainsi que l'organisation de ces ressources dans le temps et dans l'espace, en séparant les tâches manuelles des tâches semi-automatisées.

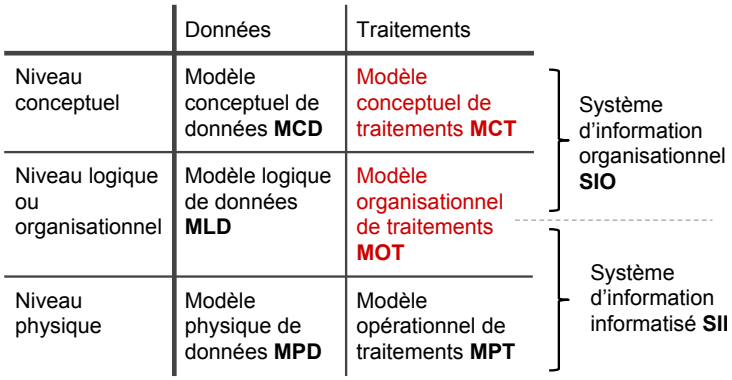


FIGURE 3.2 – Les différents modèles Merise du système d'information

Ainsi Merise distingue d'une part les termes processus et opération au niveau conceptuel (MCT), et les termes procédure et tâche au niveau organisationnel (MOT). Maintenant, définissons chacun des termes dans le tableau 3.1.

[1]Gestion de la Relation Client, en anglais *Customer Relationship Management* (CRM)

TABLE 3.1 – Définition des concepts de traitements

Concept	Merise [78]	Disic [40]
Processus	Le **processus** est un ensemble structuré d'évènements, **opérations** et résultats consécutifs qui concourent à un même but. Il représente généralement un sous-ensemble d'**activités** de l'entreprise.	Un processus est représenté fonctionnellement par un enchaînement d'activités permettant la délivrance d'un service ou d'un produit au client. Un **processus** est mis en œuvre opérationnellement par des **procédures** spécifiques.
Procédure	C'est un enchaînement de **tâches** d'intérêt pour l'organisation. La **procédure** organisationnelle prend en compte un évènement type (ou plusieurs évènements synchronisés), appelé évènement initial de la procédure, et produit tous les résultats types qui en découlent.	Une **procédure** décrit la mise en œuvre de tout ou partie d'un **processus**. Une procédure est un regroupement d'**opérations** qui produit un résultat contractuel. Les procédures sont des ensembles logiques de **tâches** élémentaires, permettant l'accomplissement d'un même objectif opérationnel. Elle constitue la brique de base des **processus**, et dépend généralement d'un seul acteur.
Opération	L'**opération** est la description du comportement du système d'information par rapport aux évènements types. L'opération comprend l'ensemble des **activités** que le domaine peut effectuer à partir des informations fournies par l'évènement.	Une **opération** est une étape d'une **procédure** correspondant à l'intervention d'un acteur de l'organisation dans le cadre de l'une des **activités** de la structure.
Tâche	La **tâche** modélise un ensemble homogène d'**activités** élémentaires. La tâche peut également être perçue comme la décomposition d'une **opération** conceptuelle.	Un ensemble de **tâches** élémentaires décompose une **opération**.
Activité		Une **activité** est un ensemble cohérent d'actions mobilisant une seule compétence ou un groupe de compétences (un métier) produisant un résultat dans la chaîne du **processus**. Une activité représente un maillon dans la chaîne de valeur du processus.

Nous résumons la granularité des concepts pour chaque modèle sur la figure 3.3.

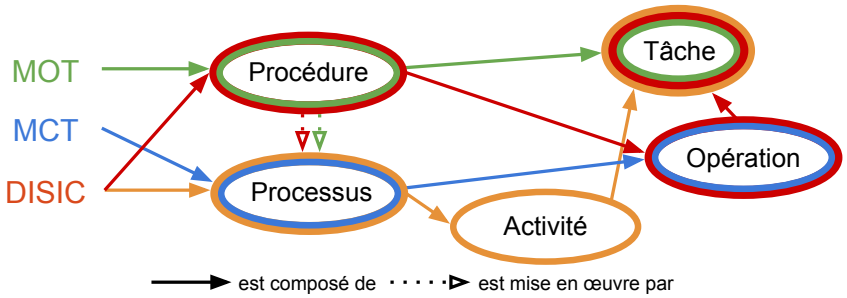


FIGURE 3.3 – Granularité des concepts de traitements

Cette étude lexicale nous permet de constater que pour un même terme, la définition peut être différente. Par exemple, tâche et opération sont variablement utilisées dans le contexte processus ou procédure. Pour la suite, nous retenons uniquement les termes processus, activité et tâche pour la modélisation de processus, nous en donnerons une définition en section 3.1.3.

3.1.2 Exemples

Les exemples sont extraits de plusieurs ouvrages : *Processus métiers et SI* de Morley, *Togaf en pratique* de Desfray [35] et *Le projet d'urbanisation du SI* de Longépé [67]. Ces ouvrages ont déjà réalisé un travail d'étude des différentes notations les plus courantes.

Les exemples que nous avons choisis sont de trois types différents pour représenter la réalisation de l'activité de l'entreprise : les processus, les diagrammes d'activité, et les cas d'utilisation. Tous utilisent des concepts similaires que nous allons détailler. Cependant, les niveaux de détail et les représentations graphiques sont très variés.

EXEMPLE 3.1 – Processus

Le processus de gestion d'inscriptions en figure 3.4 ne décrit que le contexte (rôles et acteurs) de cette activité. Nous remarquons qu'un acteur peut avoir différents rôles. Ainsi, le *Client* est à la fois *Opérateur de saisie*, *Contrôleur* et *Payeur*.

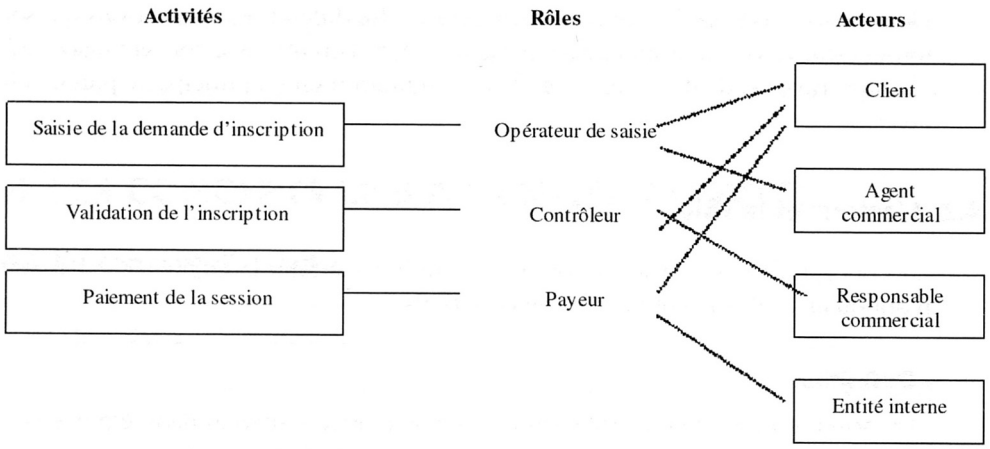


FIGURE 3.4 – Processus de gestion d'inscriptions [75]

Le processus de vente en figure 3.5 décompose l'activité en sous-processus. L'activité *Vente* se préoccupe de vendre deux types de produits : *Vente solution intégrée* et *Vente logicielle*. Et la *Vente logicielle* sur deux canaux de distribution différents : *Vente par commercial* et *Vente en ligne*.

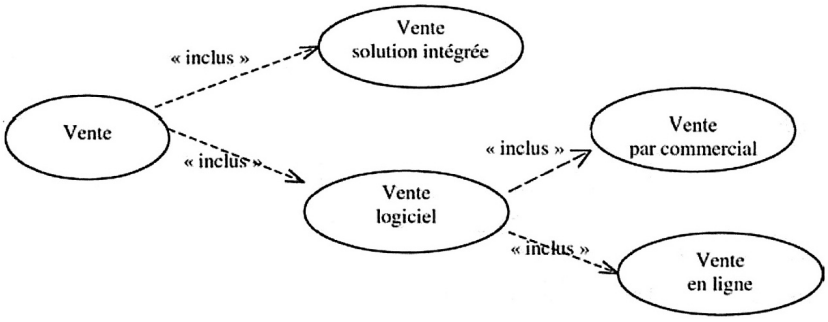


FIGURE 3.5 – Processus et sous-processus de vente modélisé en UML [75]

Les deux exemples suivants détaillent l'enchaînement des activités. La figure 3.6 décrit le processus d'une vente en ligne ; la figure 3.7 représente le processus d'une réservation de voyage. On remarque que les deux cas manipulent des évènements représentés par des symboles circulaires. Sur la vente en ligne, l'évènement de début est un évènement déclencheur extérieur : la réception d'une commande. Le déroulement des différentes tâches peut prendre plusieurs chemins possibles à l'aide de conditions. Par exemple, du choix du paiement par

chèque ou espèce ou *carte* lors de la vente en ligne a pour conséquence de suivre des tâches différentes. Le paiement par carte doit obtenir une autorisation de paiement par l'établissement financier.

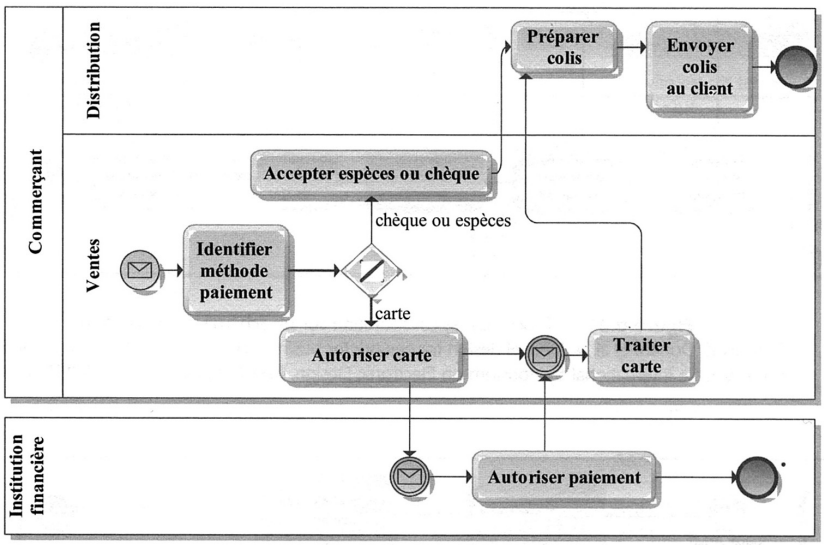


FIGURE 3.6 – Processus de paiement d'une vente en ligne modélisé en BPMN [75]

La figure 3.7 représente la donnée *Commande* par un symbole de fichier. Cette donnée est exploitée ou modifiée au fur et à mesure du processus : initialisation, vérification, lecture pour créer la facture.

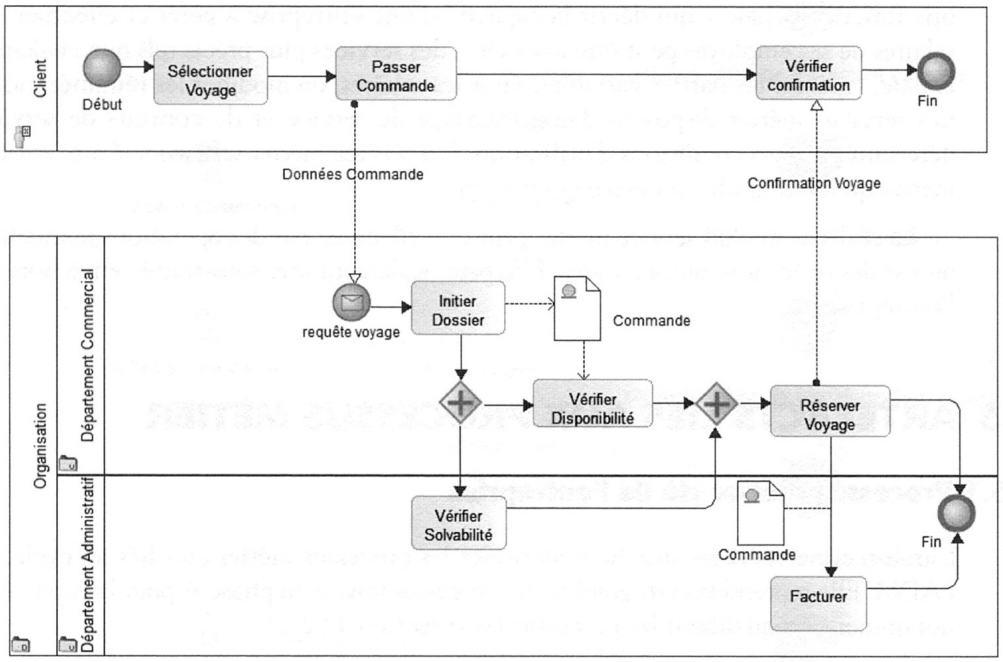


FIGURE 3.7 – Processus de réservation de voyage modélisé en BPMN [35]

Les figures 3.6 et 3.7 regroupent les tâches dans des couloirs[2]. Les couloirs peuvent être composés de sous-couloirs pour obtenir un niveau de détail supérieur. Chaque couloir fait intervenir un rôle ou un acteur qui réalise chaque tâche. Sur la réservation de voyage, l'évènement de début est déclenché par le rôle du couloir du processus : le client.

[2] *Lane* en anglais

EXEMPLE 3.2 – Diagrammes d'activité

Les diagrammes d'activité en figures 3.8 et 3.9 sont très similaires aux deux derniers processus détaillés. Ils diffèrent par leur orientation qui est verticale au lieu d'être horizontale, avec en en-tête de chaque couloir les acteurs. Une activité débute et termine sur un point unique, même si des sous-activités peuvent se dérouler en parallèle à l'intérieur de l'activité principale. Le flot de contrôle est déterminé par des conditions et des évènements.

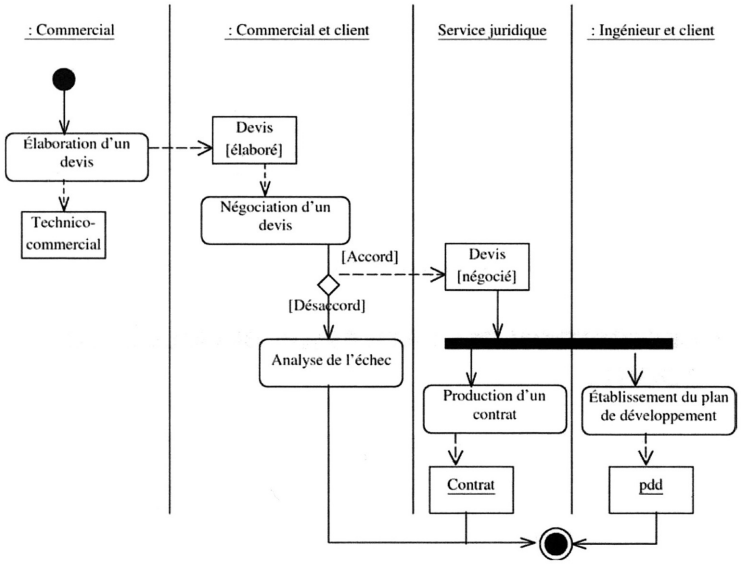


FIGURE 3.8 – Diagramme d'activité d'une vente d'un logiciel modélisé en UML [75]

Les symboles rectangulaires dans la figure 3.8 représentent les objets (ou encore appelés ressources) en interaction avec l'activité, par exemple le *Devis*. Lorsque l'objet est souligné il s'agit d'une donnée : *Contrat* et *pdd* (plan de développement). Les différentes activités modifient ou génèrent les données.

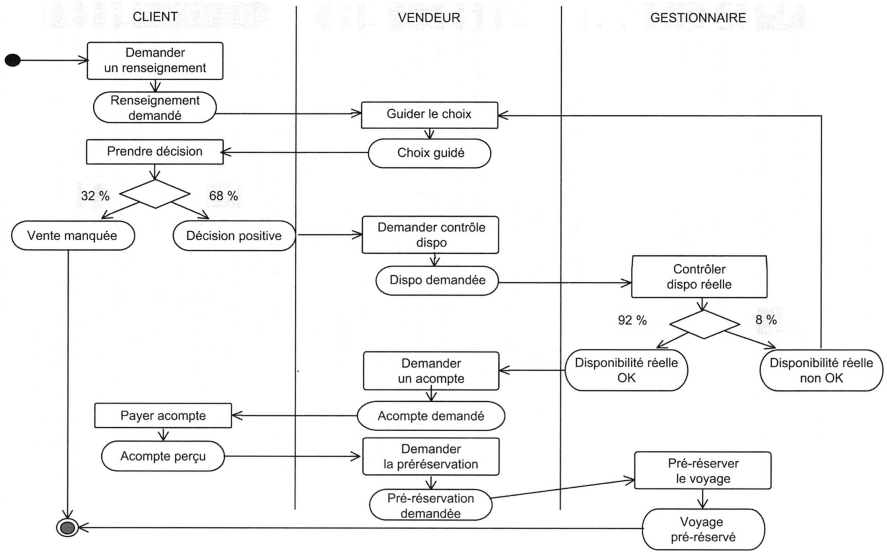


FIGURE 3.9 – Diagramme d'activité d'une réservation de voyages modélisé en UML [67]

La notation de la figure 3.9 a un symbolisme différent : les symboles rectangulaires ne sont pas des objets, mais des libellés de transition, exemple : *Demander un acompte*, *Payer un acompte*. Et les symboles rectangulaires arrondis sont les activités, exemple : *Acompte demandé*, *Acompte perçu*.

EXEMPLE 3.3 – Cas d'utilisation

Les cas d'utilisations ci-après ressemblent aux deux premiers exemples des processus non détaillés des figures 3.4 et 3.5.

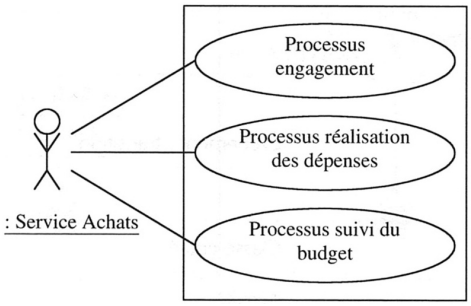


FIGURE 3.10 – Cas d'utilisation d'un service achats modélisé en UML [75]

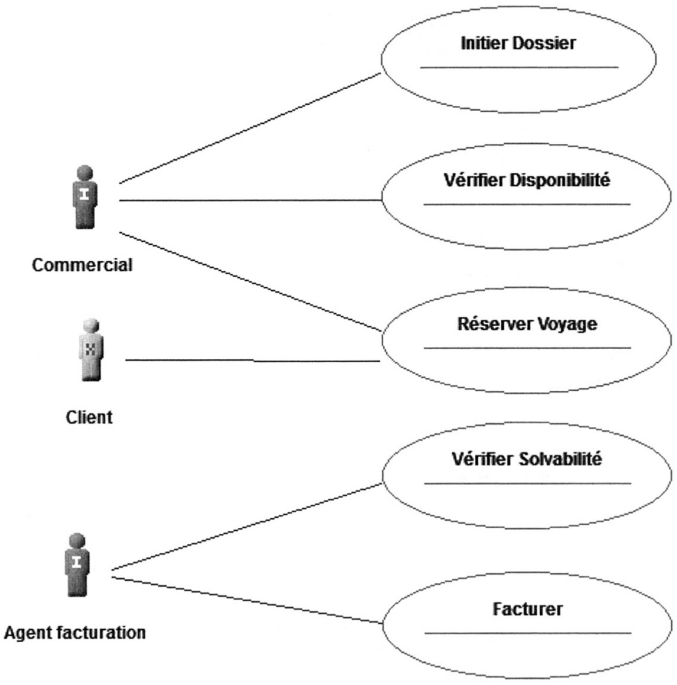


FIGURE 3.11 – Cas d'utilisation d'une réservation de voyages [35]

Le cas d'utilisation permet de mettre en relation les acteurs avec les processus (figure 3.10) ou avec les activités (figure 3.11).

À la suite de ces trois types d'exemples : processus, diagramme d'activité et cas d'utilisations. Nous remarquons que la notation processus peut servir à la fois pour représenter l'activité à un niveau abstrait ou détaillé :

- Soit les processus représentent un **niveau abstrait de l'activité** en représentant seulement les principaux processus et leurs acteurs. Cette représentation est celle utilisée par le cas d'utilisation. Ainsi, nous pouvons modéliser le cas d'utilisation à l'aide d'une notation dédiée au processus en utilisant les concepts élémentaires : processus et acteur.

- Soit les processus représentent un **niveau détaillé de l'activité** en représentant les différentes transitions entre les évènements et les activités. La représentation processus est très similaire au diagramme d'activité. Ainsi, nous pouvons modéliser les diagrammes d'activité à l'aide de processus détaillés.

Pour modéliser les activités de l'entreprise à des niveaux d'abstraction différents, nous unifions les concepts nécessaires dans la définition d'un méta-modèle à la section suivante.

3.1.3 Définition du méta-modèle

Les exemples proviennent de plusieurs standards de notation : BPMN et diagrammes d'activité UML. L'analyse de différents standards a permis de converger vers une définition générique des concepts pour établir une compatibilité.

Nous donnons une description de chaque concept du méta-modèle dans le tableau 3.2. Les concepts en italique dans la figure 3.12 et le tableau 3.2 sont purement techniques par choix de modélisation pour structurer les concepts.

TABLE 3.2 – Description des concepts de BPM

Symbole	Concept	Définition
	*BusinessProcessLayer*	Concept technique conteneur de tous les éléments de la cartographie. Il correspond au nœud racine de l'arbre XML lorsque le modèle est sérialisé en XMI.
	Element	Concept technique abstrait définissant les attributs en commun de tous les autres concepts du méta-modèle. Un nom (*name*), un identifiant unique (*id*) et des commentaires (*documentation*).
	ContainerElement	Concept technique abstrait qui généralise les concepts processus et partition. Le *ContainerElement* est composé de *ProcessElement* et de *Transition*.
	Process	Étend les propriétés de *ContainerElement* et *TransitionElement*. Le processus peut contenir des *Partitions* (*Pool* et *Lane*) et des sous-processus (*subProcess*). Le processus peut être enchaîné avec d'autres processus via des *Transition*. Le processus est effectué par (*isPerformedBy*) un *Participant* (acteur ou rôle).
	Partition	Spécialise le concept *ContainerElement*, une partition est un regroupement au sein d'un processus de *ProcessElement*. La partition est confiée à un Participant (rôle ou acteur). Une partition peut contenir des sous-partitions appelées *Lane* ; la partition mère est appelée *Pool*.
	Participant	Concept abstrait de rôle et acteur. Le participant peut réaliser une ou plusieurs activités et/ou peut effectuer un ou plusieurs processus. Le participant peut faire partie d'une partition.
	Role	Un rôle caractérise un comportement attendu qui peut être joué par un ou plusieurs acteurs jouant ce rôle.
	Actor	Dans une activité un acteur joue un rôle. Un acteur est une entité externe humaine ou non humaine (matérielle ou logicielle), il interagit avec le système.
	ProcessElement	Concept abstrait regroupant le comportement en commun des éléments contenus dans un processus : *DataObject*, *Event*, *Condition*, *Activity*. Un *ProcessElement* est contenu dans un *Process* ou une *Partition*.
	DataObject	Objet de donnée manipulé par les activités.
	TransitionElement	Concept abstrait de *Process* et *TransitionnalProcessElement*, définit l'enchaînement de ces éléments par des transitions.
	Transition	Une transition relie une association en entrée et une autre en sortie. Elle permet d'enchaîner *Activity*, *Event*, *Condition* et *Process*. Une transition est orientée.
	TransitionnalProcessElement	Concept abstrait de *Activity*, *Event* et *Condition*. Ces éléments peuvent posséder des transitions pour s'enchaîner.
	Event	Un évènement est un stimulus qui provoque une réaction dans le processus. L'évènement peut être de début, intermédiaire ou de fin.
	EventType	Énumération précisant le type de l'évènement : début, intermédiaire ou de fin.
	Condition	Une condition exprime une restriction dans l'enchaînement et le choix des activités, le déclenchement d'une transition ou l'effet d'un évènement.
	Activity	Une activité est un ensemble de tâches. Une tâche peut manipuler des objets de donnée (*DataObject*) et est réalisée par un *Participant* (acteur ou rôle).
	Task	Une tâche est le plus petit élément de décomposition d'une activité.

Nous avons défini le méta-modèle représenté en figure 3.12.

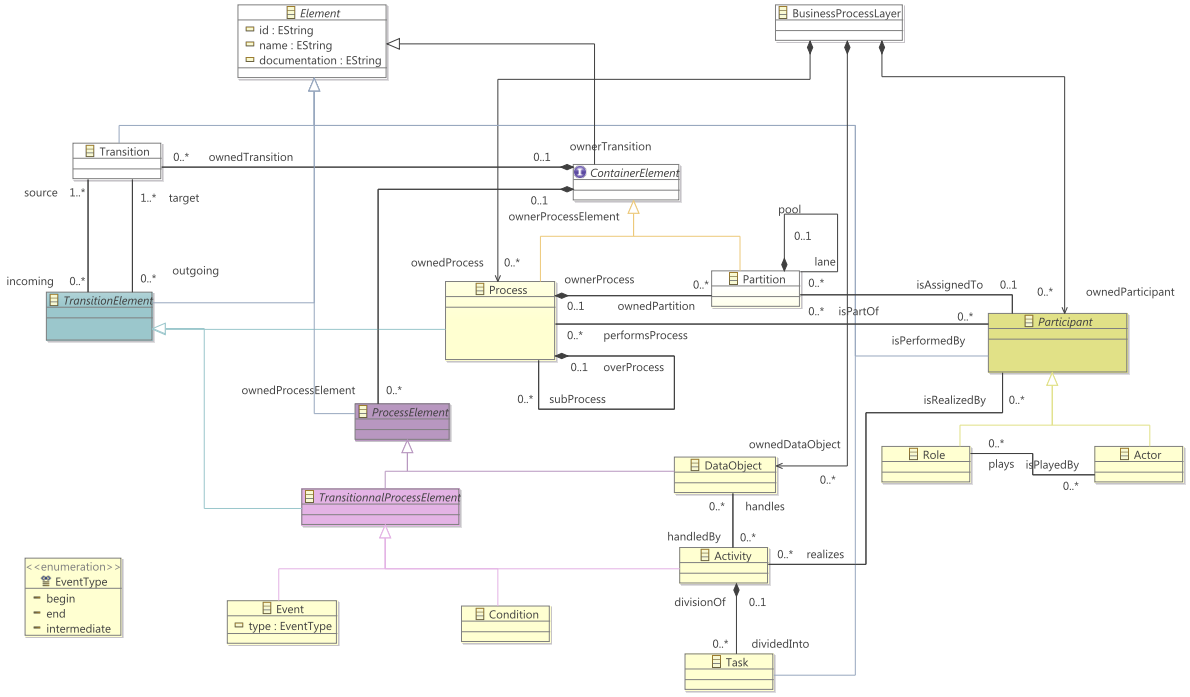


FIGURE 3.12 – Méta-modèle de processus (modélisé avec le cadriciel Eclipse EMF)

Nous retrouvons les principaux concepts communs à toutes les notations de processus : le processus (*Process*) déclinable en sous-processus composés d'activités (*Activity*) et de tâches (*Task*). Les activités se déclenchent par des évènements *(Event)* qui s'enchaînent à l'aide de conditions (*Condition*) et transitions (*Transition*). Les processus sont exécutés par des acteurs (*Acteur*) qui jouent différents rôles (*Role*). Les processus regroupent leurs activités par acteur à l'aide de couloirs et sous-couloirs *(Partition)*.

Nous donnerons des exemples issus de ce méta-modèle BPM dans la section expérimentation 3.1.5.

3.1.4 Comparaison

Plusieurs approches existent pour modéliser les processus métiers, nous avons déjà donné quelques exemples en section 3.1.2. Nous comparons notre méta-modèle avec les trois notations les plus couramment utilisées :

UML Activity Diagram : le diagramme d'activité UML permet de représenter des comportements et états du système, il est proche de l'organigramme avec une dimension temporelle ;

UML Use Case : le diagramme des cas d'utilisation, il permet de décrire l'interaction entre l'acteur et le système ;

BPMN 2 : la notation BPMN a été créée uniquement dans le but de modéliser le fonctionnement d'une entreprise par l'approche orientée processus.

BPMN est un langage de modélisation plus récent que UML, la première version de BPMN date de 2006, et la version 2.0 de 2011. La première norme UML 1.0 est apparue en 1997, et contenait le diagramme d'activité. BPMN répond davantage aux besoins pour modéliser un processus avec des concepts plus détaillés issus des bonnes pratiques déjà

existantes (UML Activity Diagram, IDEF, BPSS, Merise, OSSAD, etc.). À contrario, UML a une approche davantage dédiée à la conception logicielle et à l'architecture du système d'information. Avec l'intensification des projets de développement orientés processus et l'utilisation de moteur de workflow, BPMN est de plus en plus utilisé au détriment de *UML Activity Diagram*. D'ailleurs, BPMN depuis sa version 2.0 est directement utilisable comme langage de moteur d'exécution, ce qui évite de recourir à un langage tierce pour implémenter l'exécution des processus.

Nous avons listé les principaux concepts de chaque notation dans le tableau 3.3.

TABLE 3.3 – Comparaison de BPM avec BPMN2 et UML

BPM	BPMN 2[3]	UML Activity Diagram[4]	UML Use Case[4]
Process	Process, Activity	Activity	*
Activity	Activity, SubProcess	Action	Use Case
Actor	Pool, Lane	Partition	Actor
Role	Pool, Lane	Partition	Actor
Task	Task	*	Use Case (Include, Extend, Generalisation)
Partition	Swimlanes	Partition	Actor
Condition	Gateways	Decision, fork, join, merge node	*
Event	Event	initial node, final node, signal action, event action	*
Transition	Sequence Flow	Flow Edge	Interaction
DataObject	Data	Object	*

* Aucun concept équivalent n'est existant

Nous remarquons qu'entre notre propre notation BPM définie en section 3.1.3 et les notations BPMN 2 et UML Activity Diagram, nous retrouvons les principaux concepts, mais la dénomination est différente ou l'information est différemment représentée. Par exemple, le diagramme d'activité UML ne possède ni concept de rôle ni concept d'acteur, mais cette information est portée par le nom de la partition qui est en fait le nom de l'acteur réalisant les activités de la partition. Le diagramme de cas d'utilisation UML est une représentation avec beaucoup moins de concepts, mais qui a des correspondances avec BPM.

3.1.5 Expérimentations

Les quatre expérimentations suivantes reprennent les exemples (cf. section 3.1.2) tirés de l'ouvrage de Morley [75]. Nous avons réinterprété les cas d'études pour les modéliser avec notre propre notation. Ces expérimentations permettent de vérifier la compatibilité et la pertinence du choix des concepts primordiaux de notre méta-modèle. Afin de pouvoir représenter par une méthode générique les points de vue processus et la description des activités d'une entreprise.

Nous n'avons pas traduit tous les cas d'étude, mais un échantillon des deux catégories après nos constatations dans la section Exemples 3.1.2 :

- Processus simplifié ou cas d'utilisation : les processus et les acteurs associés ;
- Processus détaillé ou diagramme d'activité : la description temporelle et évènementielle des activités.

[3] BPMN 2.0 specification http://www.omg.org/spec/BPMN/2.0/
[4] UML 2.5 specification http://www.omg.org/spec/UML/2.5/

EXPÉRIMENTATION 3.1 – Processus simplifié vers BPM

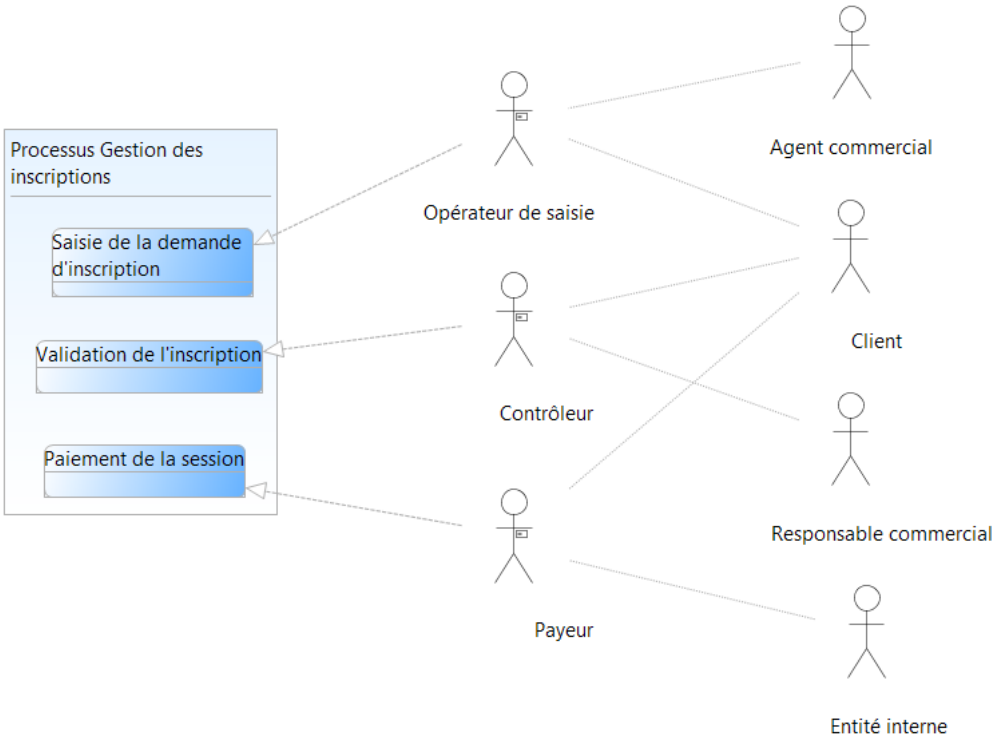


FIGURE 3.13 – Processus de Gestion des inscriptions en BPM

La figure 3.13 est la traduction de l'exemple en figure 3.4 du processus de *Gestion d'inscriptions*. Le processus de *Gestion d'inscriptions* a une session de formation comprenant trois **activités** : la *Saisie de la demande*, la *Validation de l'inscription* et le *Paiement de la session*. Trois **rôles** (symbole bonhomme avec un badge) sont identifiés et affectés à une activité. Les rôles sont joués par un ou plusieurs **acteurs** (symbole bonhomme sans badge).

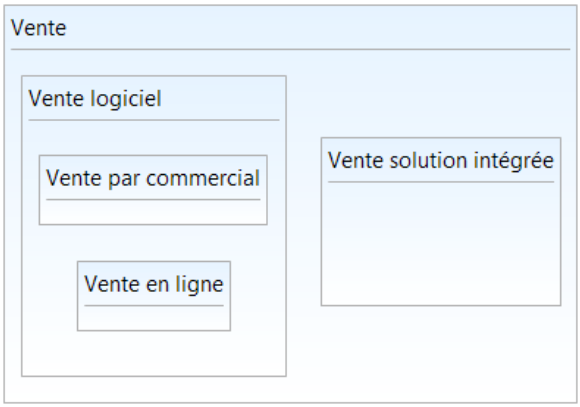


FIGURE 3.14 – Processus et sous-processus de Vente en BPM

La figure 3.14 est la traduction de l'exemple en figure 3.5 de processus et sous-processus de *Vente*.

Notre modélisation permet de réaliser une vue macroscopique et met bien en évidence l'imbrication des différentes activités.

Le processus **Vente** englobe la *Vente logicielle* et la *Vente de solution intégrée*. La *Vente de solution intégrée* est définie par un processus unique, tandis que la *Vente logicielle* est spécifiée par deux techniques de vente : la *Vente en ligne* et la *Vente par un commercial*.

EXPÉRIMENTATION 3.2 – Processus détaillé vers BPM

La figure 3.15 est la traduction de l'exemple en figure 3.6 de processus de *Paiement d'une vente en ligne*.

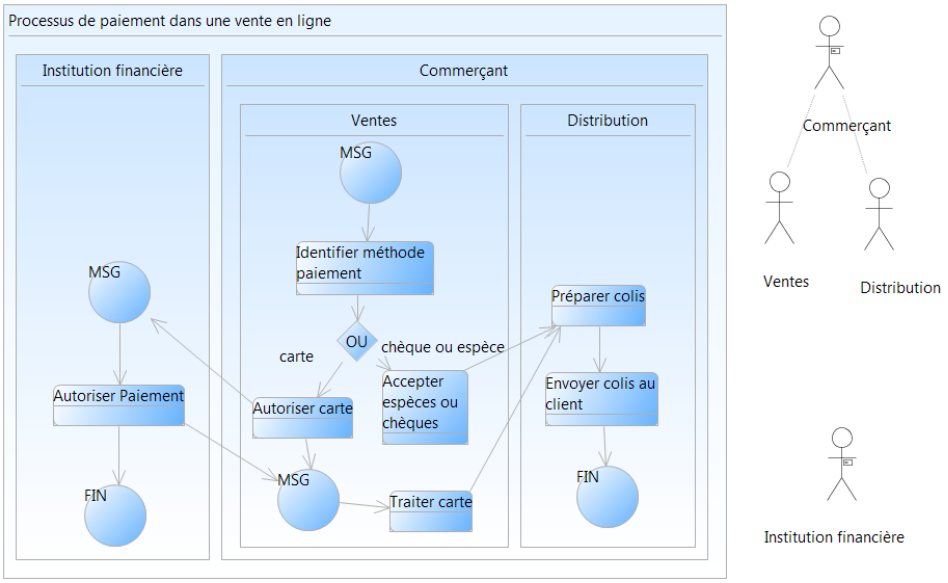


FIGURE 3.15 – Processus de paiement

Le processus de paiement dans une vente en ligne décrit la séquence de paiement qui conduit à l'envoi de l'article commandé lors d'une vente de produit en ligne. Les deux rôles *Commerçant* et *Institution financière* sont affectés aux deux couloirs. Les acteurs *Ventes* et *Distribution* sont eux affectés aux sous-couloirs du couloir *Commerçant*. La Vente est un traitement automatisé par le système informatique. La *Distribution* a en charge la préparation et l'envoi du colis au client (non représenté). Si le paiement s'effectue par carte, un message est envoyé à l'*Institution financière* : cet évènement déclenche l'activité d'autorisation du paiement qui envoie un message au *Commerçant*.

Nous n'avons pas rencontré de difficulté pour traduire les différents exemples. Notre méthode est suffisamment générique pour couvrir les cas étudiés.

3.1.6 Bilan

Le point de vue processus métier permet de représenter l'organisation et l'ordonnancement des activités d'une entreprise. L'étude d'exemples et de différents langages de modélisation dédiés à la représentation processus et activités nous ont permis d'élaborer un méta-modèle unifié. Les principaux concepts sont les processus, les activités, les tâches, les rôles, les acteurs, les évènements, les conditions et les objets de données.

Notre méta-modèle processus n'est néanmoins pas totalement complet, la présente thèse n'étant pas dédiée à la représentation processus, nous nous sommes intéressés aux concepts processus généraux et non aux détails qui pourraient être ajoutés par la suite à notre méta-modèle. Par exemple nous avons simplifié les types d'évènements au nombre de trois : début, intermédiaire et fin. Durant notre étude nous avons aperçu davantage de types existants d'évènements et de conditions. Dans la notation BPMN2 les conditions sont au nombre de 8, et les évènements au nombre de 63 comme illustrés sur la figure 3.16[5].

[5]Extrait de BPMN 2.0 Poster http://www.bpmb.de/images/BPMN2_0_Poster_FR.pdf

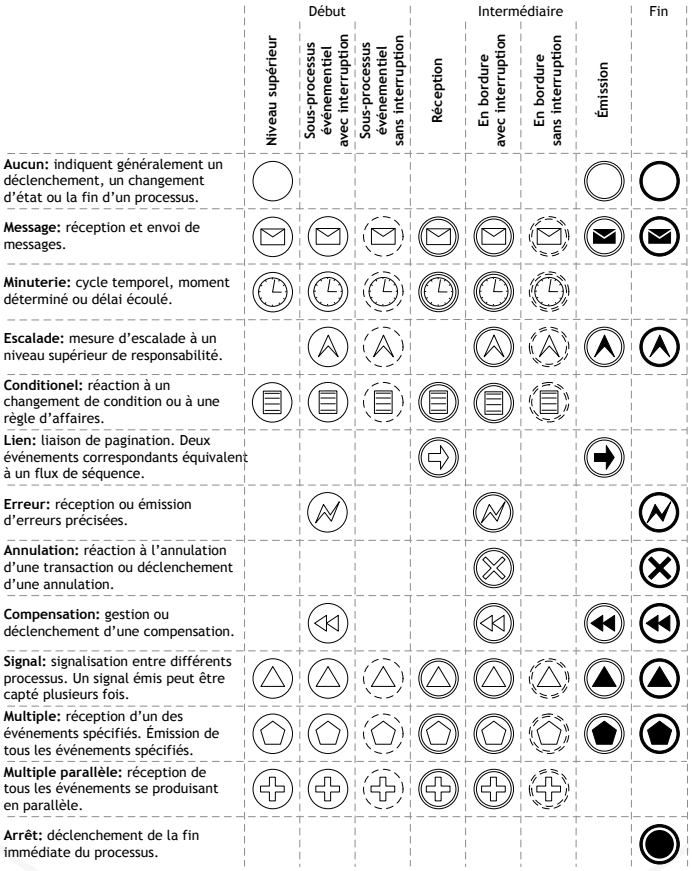


FIGURE 3.16 – Les évènements de BPMN 2

3.2 Fun : le point de vue fonctionnel

La présente section décrit le point de vue fonctionnel. Tout d'abord, nous donnons une définition générale, illustrée par des exemples existants. Ensuite, nous proposons notre méta-modèle fonctionnel (**Fun**) et la définition de ses concepts. Nous comparons notre méta-modèle à des notations existantes dans des logiciels commerciaux et donnons un exemple de modélisation.

3.2.1 Introduction

Le point de vue fonctionnel est un point de vue métier du SI qui vise à identifier les grandes fonctions de l'entreprise. Cette représentation est extraite de la méthode d'urbanisation des villes introduite dans l'état de l'art (section 2.3.2). Elle peut être nommée par des appellations différentes comme le plan d'occupation des sols, ou encore le lotissement. Le point de vue fonctionnel permet de réaliser le découpage structurel des activités, à savoir les services d'une entreprise. À la différence du point de vue processus métier, la fonction métier n'a pas de caractère temporel : pas de début, pas de fin, ni d'évènement déclencheur. La fonction est donc continue pour réaliser une activité de l'entreprise. La notion de fonction en entreprise a été introduite à la fin du 19ème siècle par le français Henri Fayol [45]. La fonction est un groupe d'opérations concentrant des expertises, métiers et savoir-faire communs ou voisins. On parle également de département, secteur, service ou pôle. Fayol distingue cinq grandes fonctions qui constituent l'entreprise :

- **fonction technique** : production, fabrication ou transformation qui constituent la valeur ajoutée de l'entreprise ;
- **fonction financière** : recherche et utilisation optimale des capitaux mis à la disposition de l'entreprise ;
- **fonction de sécurité** : protection des personnes, des biens et du patrimoine de l'entreprise ;
- **fonction comptable** : calcul de la paie et des statistiques, recensement des actifs et du patrimoine de l'entreprise ;
- **fonction commerciale** : achat, vente et échange.

Dans l'entreprise, on distingue souvent huit grandes fonctions de base : direction et administration, achat, finance et comptabilité, logistique, marketing et commerciale, production, recherche et développement et ressources humaines. Parfois, on ajoute des fonctions caractéristiques de l'activité et de l'organisation d'une entreprise, par exemple : une fonction juridique chez Google, une fonction *design* chez Apple et une fonction exploration chez Total.

DÉFINITION 3.3 – Point de vue fonctionnel

Le point de vue fonctionnel est la structuration du système d'information en blocs fonctionnels communicants. La cartographie fonctionnelle fait apparaître les liens entre les zones, quartiers, îlots, et les activités métier qu'ils assurent (ou assureront dans le cas de la conception du futur système d'information). Ces informations permettront de mesurer les impacts des modifications fonctionnelles et de repérer les parties d'application éventuellement réutilisables.

Le projet d'urbanisation du SI, p. 33 [67]

Le point de vue fonctionnel est représenté par une structure hiérarchique par décomposition des différentes fonctions. Le but est d'avoir une abstraction utile pour comprendre la structure de l'entreprise sans connaître le déroulement de l'activité (information réservée au point de vue des processus métiers). Le découpage se réalise en **blocs** et sous-blocs fonctionnels qui sont responsables d'une ou plusieurs **fonctionnalités** et manipulent des informations représentées par une **donnée**. Exemple : une entreprise est composée de plusieurs fonctions : marketing, achat, vente, commercial, etc. Ces entités sont représentées par des blocs. Puis, chaque bloc peut avoir un découpage en sous-blocs. La fonction vente peut être déclinée en un service vente sur le web, vente physique, etc. Plus l'entreprise est d'une taille importante et son organisation est complexe, plus le découpage aura de nombreux niveaux de granularité. La granularité la plus communément utilisée, en référence à l'urbanisation des villes, possède 3 niveaux : **zone, quartier, îlot** [67].

3.2.2 Exemples

Les exemples sont tirés de deux ouvrages : *Le projet d'urbanisation* de Longépé [67] et *Togaf en pratique* de Desfray [35]. Ils présentent tous deux le point de vue fonctionnel d'une agence de voyage. La représentation de deux entreprises du même secteur d'activité permet d'illustrer les différences d'organisation possibles, malgré un métier identique.

EXEMPLE 3.4 – Points de vue fonctionnel

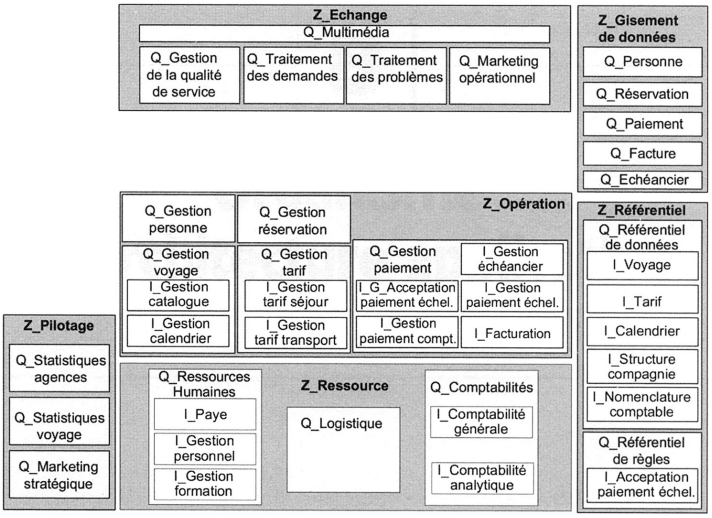


FIGURE 3.17 – Point de vue fonctionnel d'une agence de voyages [67]

Le premier exemple en figure 3.17 est extrait de l'ouvrage de Longépé [67]. Pour modéliser le fonctionnement de l'agence de voyage, il a utilisé la méthode décrite dans sa démarche d'urbanisation, à savoir :

- définition des **zones** à partir de la bonne pratique : échange, gisement de données, référentiel de données et de règles, pilotage, opérations, ressources ;
- identification des **quartiers** à partir des processus ou de la connaissance de l'entreprise le cas échéant ;
- niveau de détail dans les **îlots** toujours à partir des processus : spécialisation des activités.

Ainsi, les zones ont une nomenclature spécifique à la démarche : échange, données, référentiel, opérations, ressources, pilotage, tandis que les quartiers et les îlots sont caractéristiques des fonctions de l'agence de voyage. Exemple la zone *Opération* est composée du quartier *Gestion voyage*, lui-même composé des îlots *Gestion catalogue* et *Gestion calendrier*.

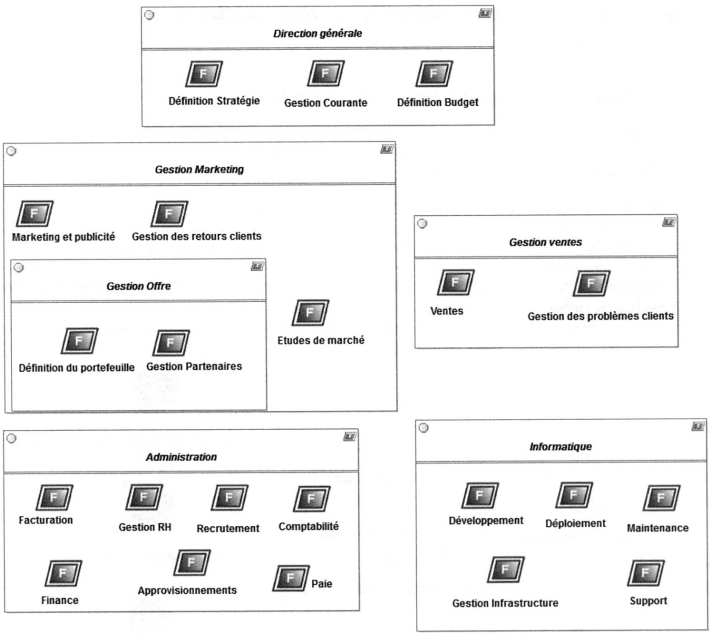


FIGURE 3.18 – Point de vue fonctionnel de la compagnie Discount Voyages [35]

Le second exemple en figure 3.18 est un découpage générique en blocs et sous-blocs fonctionnels. Il n'y a pas de définition de granularité : zone, quartier, îlot. Au premier niveau de

découpage, on retrouve les principales fonctions d'une entreprise : direction générale, gestion marketing, gestion ventes, administration, informatique. La fonction marketing est découpée en une fonction *Gestion offre* elle-même décomposée en *Définition du portefeuille* et *Gestion partenaires*.

En comparant les deux représentations d'agence de voyage, nous remarquons que les deux fonctions similaires Gestion de catalogue et Gestion de portefeuille ne font pas partie de la même fonction de plus haut niveau. Dans la modélisation de Longépé, cette fonction *Portefeuille/Catalogue* fait partie de **Gestion Voyage** de la **Zone Opération**. Tandis que dans l'ouvrage de Desfray, la fonction fait partie de *Gestion Offre* qui est similaire à *Gestion Voyage*, mais *Gestion offre* est incluse dans *Gestion Marketing*. Or la fonction *Marketing* dans la figure 3.17 fait partie de la *Zone Échange*.

Par ces deux exemples modélisant une activité similaire (l'agence de voyage), nous remarquons que le découpage est dépendant de la méthode d'analyse des fonctions du SI, de la granularité adoptée (bloc|sous-bloc ou zone|quartier|îlot) et de la nomenclature imposée par les règles d'urbanisation spécifiques à l'organisation de chaque entreprise.

3.2.3 Définition du méta-modèle

Le point de vue fonctionnel a une granularité en blocs qui peut être différente selon les organisations : 2, 3, 4 niveaux ou plus. Ainsi la représentation Zone, Quartier, Îlot n'est pas suffisante. Nous proposons donc une définition plus générique par une composition de blocs / sous-blocs sans limites de niveaux pour une compatibilité avec tous les choix de représentation d'organisation. Nous avons défini le méta-modèle fonctionnel (**Fun**) en figure 3.19.

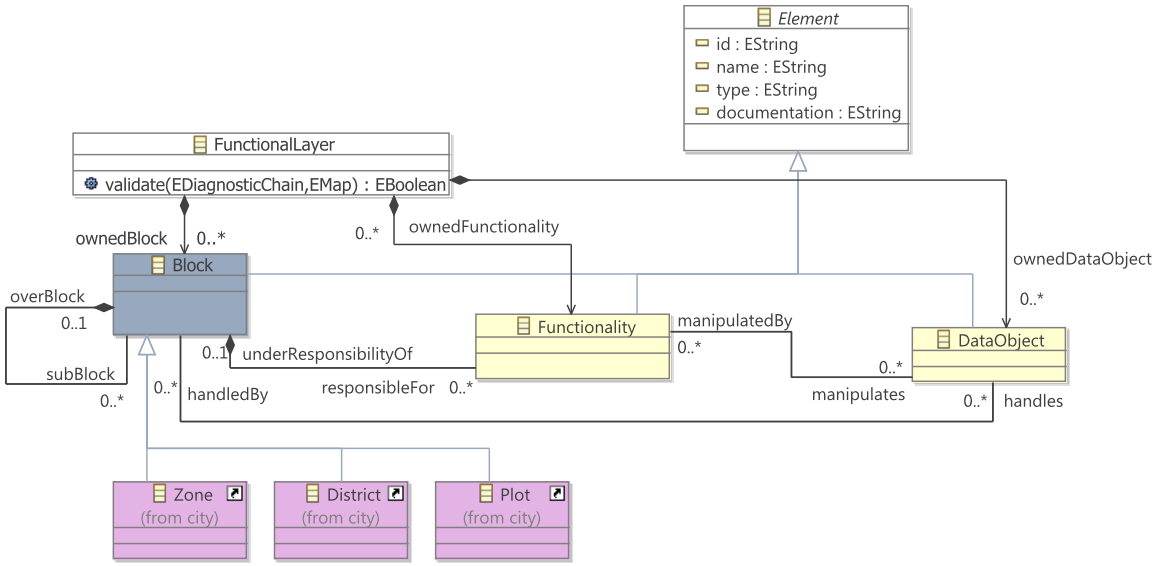


FIGURE 3.19 – Le méta-modèle fonctionnel modélisé avec le cadriciel Eclipse EMF

Un bloc (*Block*) est responsable de fonctionnalité (*Functionality*) ; est divisible en sous-blocs ; et manipule une donnée (*DataObject*). La granularité zone (*Zone*), quartier (*District*) et îlot (*Plot*) hérite du concept de bloc.

Nous donnons une description de chaque concept du méta-modèle dans le tableau 3.12. Les concepts en italique sont des classes techniques utilisées par la modélisation pour structurer les concepts.

TABLE 3.4 – Description des concepts fonctionnels

Symbole	Concept	Description
	*FunctionnalLayer*	Concept technique qui contient tous les éléments de la cartographie. Il correspond au nœud racine de l'arbre XML lorsque le modèle est persisté en XMI.
	Element	Concept technique définissant les attributs en commun de tous les autres concepts du méta-modèle. Un nom (name), un identifiant unique (id) et des commentaires (documentation).
	Block	Le bloc est le concept fondamental qui permet le découpage en blocs et sous-blocs. Un bloc peut manipuler un objet de donnée (DataObject) et peut posséder des fonctionnalités (Functionality).
	Zone	Premier niveau de découpage du SI, spécialise les propriétés de bloc. La liste des zones est définie à partir des règles de bonnes pratiques, exemple : Opération, Support, Pilotage et Contrôle, Données transverses, Échanges. Une zone se décompose en quartiers (District).
	District	Le quartier (District) spécialise les propriétés de bloc, il correspond à un sous-système, il est rattaché à une seule zone et se décompose en plusieurs îlots (Plot). Exemple : gestion des contrats, gestions des paiements, gestions du stock.
	Plot	Îlot (Plot) spécialise les propriétés de bloc, il est un ensemble de fonctionnalités et accède à des données. Il s'agit du niveau de granularité le plus fin du SI, un îlot est remplaçable, ses résultats peuvent être exploités par un autre îlot. Exemple : facturation, création client.
	Functionality	Une fonctionnalité représente un ensemble de traitements et de règles de gestion produisant un résultat. Une fonctionnalité appartient à un et un seul bloc. La fonctionnalité manipule des objets métiers appartenant uniquement à son bloc. Une fonctionnalité peut être réutilisable (par plusieurs processus) et est implémentée par des Services applicatif.
	DataObject	Objet métier est caractérisé par des propriétés : attributs et association. Il correspond à une entité du diagramme de classe.

Pour vérifier la cohérence de la granularité **Zone, Disctrict, Plot** établie dans ce méta-modèle, nous avons spécifié quatre règles de conformance du modèle en OCL, voir la définition en liste 3.1. La règle sur la zone (*Zone*) vérifie que ses sous-blocs sont des quartiers (*District*). Deux règles concernent le quartier, la première vérifie que son bloc parent est une zone, la deuxième contrôle si le sous-bloc fils est bien un îlot (*Plot*). La règle sur l'îlot (*Plot*) vérifie que ses blocs parents sont des quartiers (*District*).

Listing 3.1 – WFR Zone, Quartier, Îlot

```
context Zone
        inv: self.subBlock.oclIsTypeOf(city::District)
context District
        inv: self.overBlock.oclIsTypeOf(city::Zone)
context District
        inv: self.subBlock.oclIsTypeOf(city::Plot)
context Plot
        inv: self.overBlock.oclIsTypeOf(city::District)
```

Les règles ont été ajoutées au cœur du méta-modèle, EMF supportant nativement OCL comme mécanisme de validation. Les règles sont donc invoquées à chaque manipulation des concepts concernés.

3.2.4 Comparaison

Nous avons réalisé une comparaison des logiciels de référentiel d'entreprise les plus connus du commerce intégrant une modélisation fonctionnelle : MEGA®, Aris® et CaseWise®. Les éditeurs ne documentent pas leurs méta-modèles, néanmoins les concepts ont été déterminés par rétro-documentation.

Dans le manuel du logiciel de référentiel d'entreprise MEGA, nous avons extrait l'exemple d'une modélisation fonctionnelle d'une compagnie de croisière, la figure 3.20 illustre plusieurs niveaux de découpage visibles :

Area : sans icône, il s'agit du niveau le plus haut appelé **aire** : "*Front Office*", "*Partner communication*", "*Repositories*"... ;

District : avec une icône 9 carrés, il s'agit du niveau intermédiaire appelé **quartier** : "*Bank partner*", "*Travel partner*" et "*Boat equipment partners*" dans l'aire "*Partner communication*" ;

Block : avec une icône 4 carrés, il s'agit du niveau le plus bas appelé **bloc** : "Travel Desk", "Reservation" dans le quartier "Desk Services".

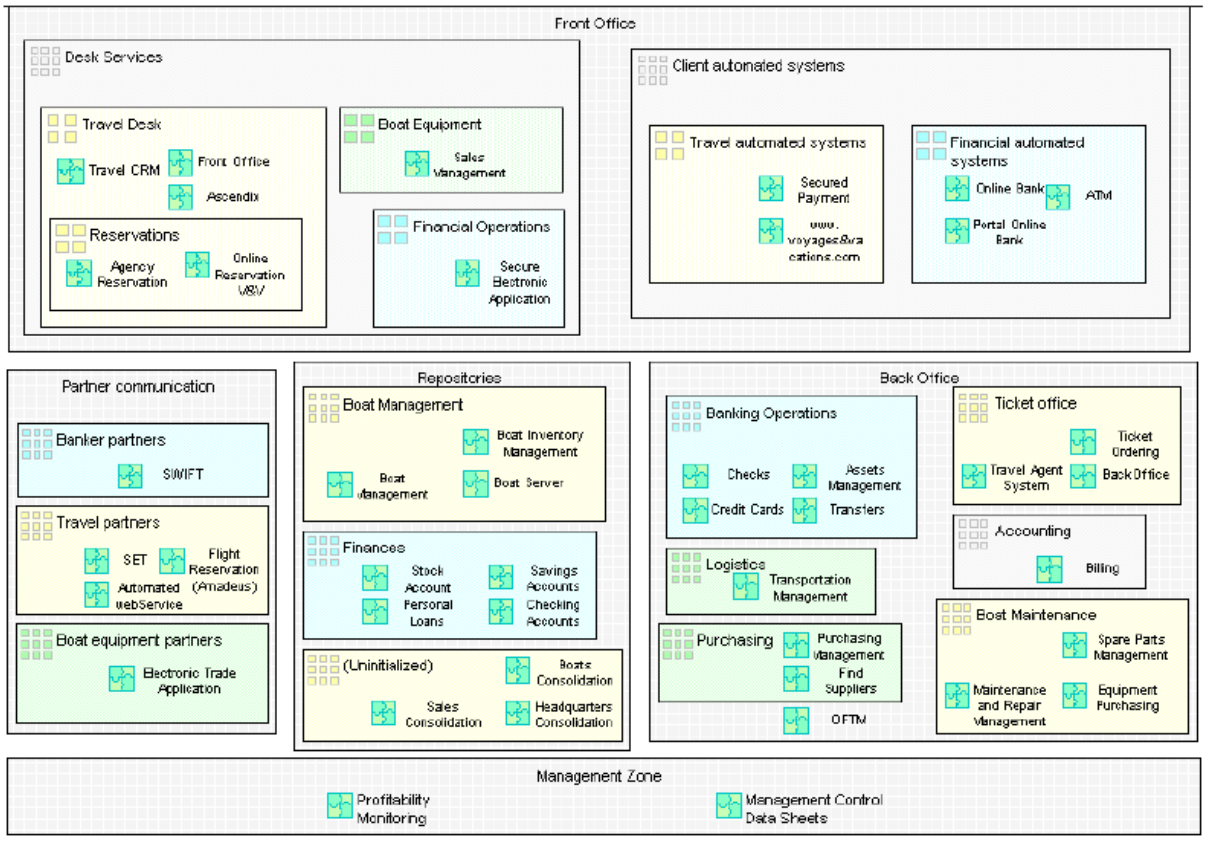


FIGURE 3.20 – Exemple de modélisation fonctionnel d'une compagnie de croisière avec Mega City Planning

Nous remarquons que chaque découpage peut recevoir des composants applicatifs : "*SET*", "*Flight Reservation*" et "*Automated webservice*" dans le quartier "*Travel partner*" de l'aire "*Partner communication*". La représentation de MEGA mélange des concepts en provenance d'une **représentation applicative** que nous aborderons dans la section 3.3 à la **représentation fonctionnelle**. À la différence de notre définition, les blocs sont responsables uniquement de fonctionnalités. Pour éviter la gestion de doublon de concepts entre différents points de vue, nous préférons dissocier les concepts dans chaque point séparément. Ainsi, dans la section 4 nous proposons d'associer les fonctionnalités du point de vue fonctionnel aux composants applicatifs du point de vue applicatif par alignement.

Le logiciel Aris (copie d'écran, figure 3.21) propose un découpage similaire à MEGA, on retrouve la zone (**Zone**) en rose sur fond blanc, puis le quartier (**District**) avec une bordure plus fine, et le bloc fonctionnel (**Functional Block**) avec le fond rose.

Chaque élément peut contenir trois types de composants en provenance du point de vue applicatif : développement interne (***Internal Development***), développement externe (***External Development***) et logiciel (***Software***).

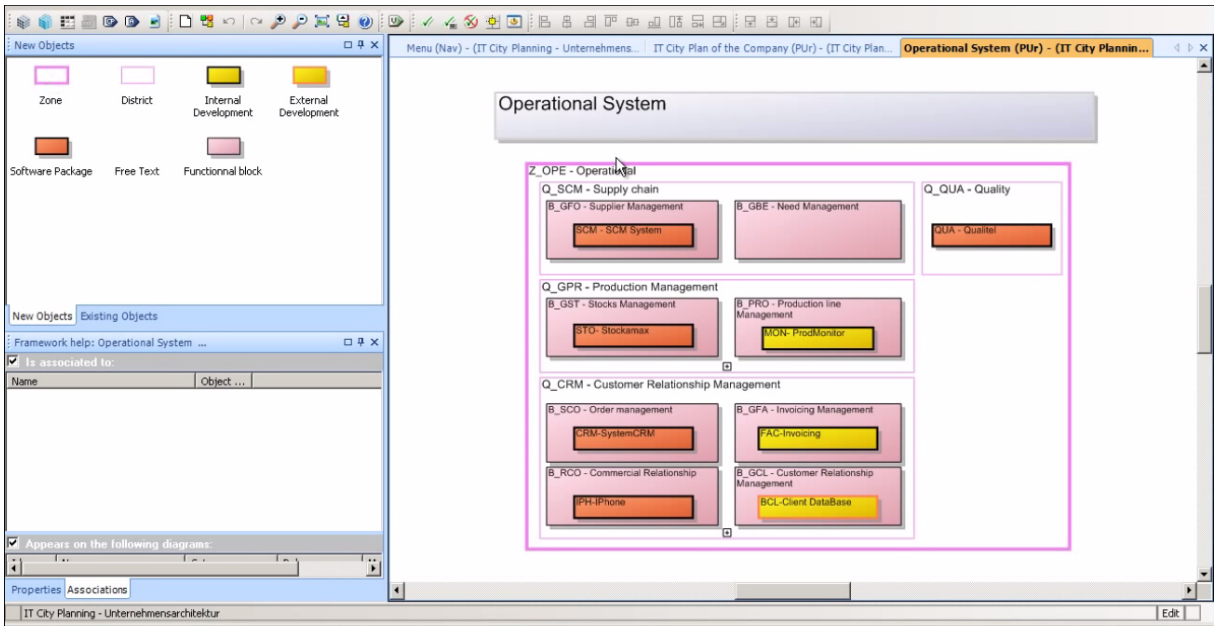


FIGURE 3.21 – Copie d'écran du logiciel ARIS

Nous avons compilé les différents concepts des différentes notations dans le tableau 3.5.

TABLE 3.5 – Comparaison de concepts fonctionnels

Fun	MEGA City Planning	CaseWise IT-City planning	Aris IS view / IT city planning
Zone	Area	Zone	Zone
District	District	District	District
Plot	Block	Building Cluster	Functional Block
*	*	Functional Block	*
Functionality, DataObject	Application Component	Capabilitiy, IS service	Software Package, Development

* Aucun concept équivalent n'est existant

Les noms des différents niveaux de bloc peuvent avoir des différences :

1. **Zone** ou **Aire** (*Area*) pour le niveau le plus haut
2. **Quartier** (*District*) pour le niveau intermédiaire
3. **Bloc** (*Block*) ou **Îlot** (*Plot*)

Notons que CaseWise propose un niveau intermédiaire supplémentaire : la grappe (*Building Cluster*). Chacun des trois logiciels commerciaux associe aux blocs des composants logiciels provenant de la vue applicative. Ces composants peuvent définir des traitements ou structures de données. Nous proposons une abstraction plus haute en définissant les notions de fonctionnalité (*Fonctionality*) et d'objet de donnée (*DataObject*).

3.2.5 Expérimentations

L'expérimentation en figure 3.22 est une ré-interprétation du cas d'étude de l'agence de voyage extrait de l'ouvrage de Longépé [67] à la figure 3.17. La modélisation avec notre méta-modèle fonctionnel permet de vérifier son fonctionnement.

EXPÉRIMENTATION 3.3 – Architecture fonctionnelle d'une agence de voyage avec Fun

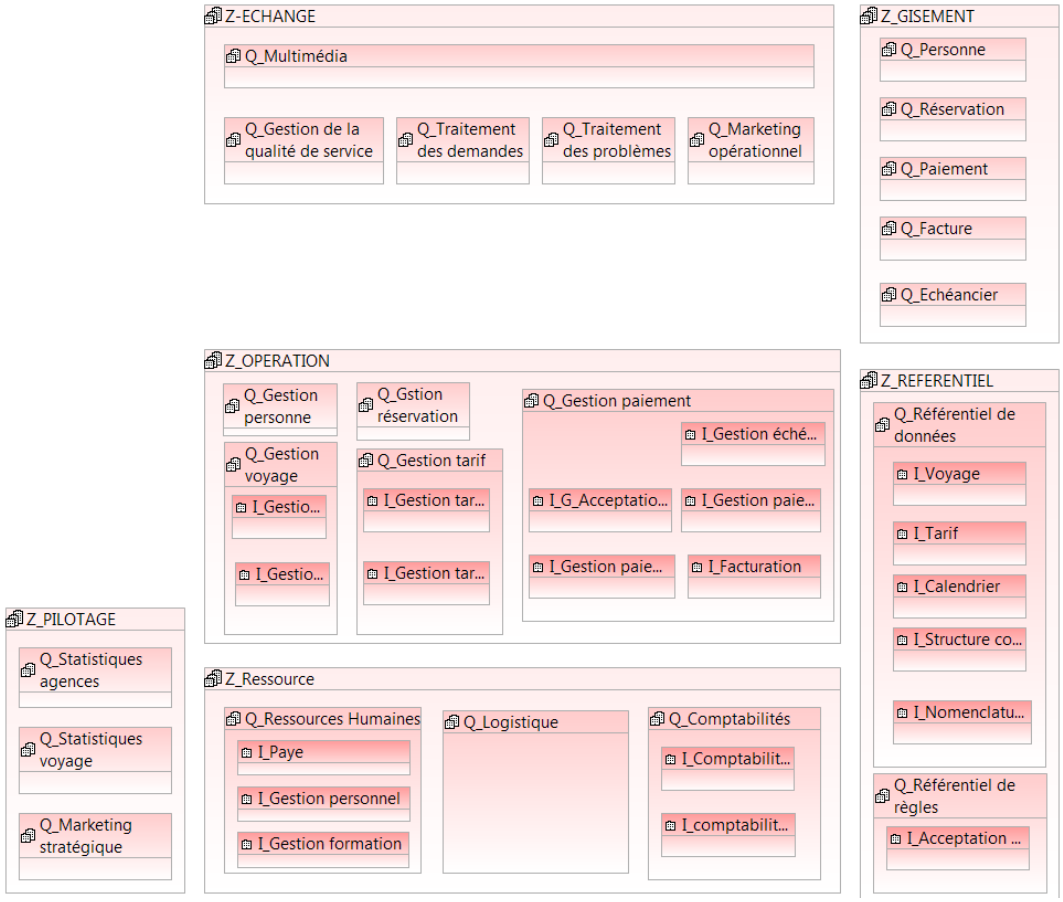


FIGURE 3.22 – Modélisation du point de vue fonctionnel de l'agence de voyage

Pour modéliser ce cas d'étude, il faut commencer par les blocs de plus haut niveau *Zone*, puis *Quartier*, et enfin *Îlot*. Le niveau de détail du cas d'étude est assez faible. Il ne décrit ni fonctionnalité ni donnée. Cette modélisation permet d'avoir une vue d'ensemble abstraite de l'entreprise pour avoir une compréhension de son organisation en un coup d'œil.

Le point de vue fonctionnel est la modélisation la plus simple à réaliser pour la cartographie du système d'information. La modélisation du diagramme fonctionnel se déroule en quelques étapes :

1. choix de la granularité : nombre de niveaux de blocs ;
2. création des blocs pour chaque niveau, ou de la fonctionnalité ;
3. affectation du nom de la fonction métier au titre du bloc ou de la fonctionnalité ;
4. ajout des objets de données.

3.2.6 Bilan

Le point de vue fonctionnel permet le découpage de l'organisation d'une entreprise en blocs fonctionnels. L'étude de différentes représentations fonctionnelles nous a permis d'établir un méta-modèle fonctionnel générique. Les principaux concepts sont les blocs, les fonctionnalités et les objets de données. En plus de ces trois concepts, nous avons spécifié la granularité la plus couramment rencontrée, celle de Longépé permet trois niveaux de détail : des zones contenant des quartiers contenant des îlots.

3.3 App : le point de vue applicatif

La présente section détaille le point de vue applicatif. Après une définition informelle complétée par des exemples, nous définirons les différents concepts spécifiant notre méta-modèle **App**. Notre méta-modèle est inspiré de Archimate®, un standard de référence du domaine. Nous proposons une comparaison avec cette représentation et avec d'autres langages ayant les notions orientées composants et services (UML et SCA). Enfin, nous donnerons plusieurs expérimentations dont un extrait d'une modélisation d'un logiciel avec notre point de vue applicatif, issu du cas d'étude étudié en chapitre 5.

3.3.1 Introduction

Le point de vue applicatif concerne la représentation du parc applicatif, c'est-à-dire les différentes **applications** utilisées dans le système d'information.

DÉFINITION 3.4 – Le point de vue applicatif

La vue applicative permet de décrire le système informatique et plus précisément les solutions technologiques retenues pour implémenter la vue fonctionnelle, mais ne comprend pas les équipements physiques. Il s'agit de décrire les composants logiciels du SI, principalement les applications et leurs architectures : les services offerts, les données physiquement gérées et les échanges et flux de données.

Cadre Commun d'Urbanisation du Système d'Information de l'État, p. 67 [40]

Les applications peuvent être de différentes natures : progiciel, scripts, batch, développements spécifiques... Nous considérons chaque application comme un **composant** applicatif qui peut lui-même être composé de sous-composants. Chaque composant rend des **services** exposés dans une **interface**. Les services sont réutilisables par d'autres services pour créer de nouveaux services étendus. Le service décrit un comportement implémenté par une **fonction** informatique fournie par un composant, un service peut utiliser une ou plusieurs fonctions. La *fonction informatique* doit bien être dissociée de la *fonction de management* définie dans la section 3.2. La fonction informatique est une routine qui prend des paramètres en entrée et retourne une seule valeur.

Pour transmettre des informations entre les composants, les fonctions implémentant les services accèdent à des **données**. Les données peuvent être internes à l'application ou externes dans une base de données. Les données sont créées, modifiées et supprimées au sein des fonctions.

3.3.2 Exemples

Nous avons choisi trois exemples utilisant trois notations différentes : Archimate, *Service Component Architecture Assembly Model* (SCA) et diagramme de composants UML. Pour les exemples concernant SCA et le diagramme de composants UML, les cas d'études sont tirés des documents de spécifications disponibles sur le site des normes, respectivement Oasis[6], et OMG[7]. L'exemple utilisant Archimate est tiré du cas d'étude Archisurance qui est publié par *The Open Group*, le promoteur d'Archimate et TOGAF. Néanmoins, nous

[6] SCA 1.1 specification http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-spec-v1.1-csd08.pdf

[7] UML 2.5 specification http://www.omg.org/spec/UML/2.5/

avons utilisé un seul exemple provenant d'Archisurance, les autres ne respectant pas les spécifications. Un des exemples comporte des composants applicatifs (*Application Component*) qui sont en relation directe avec des objets de données (*DataObject*) ; or aucune relation n'existe dans le méta-modèle entre les composants applicatifs et les objets de données (cf. figure 3.26).

EXEMPLE 3.5 – Archimate

L'exemple de la figure 3.23 décrit la réalisation du service de création d'une assurance par le composant appelé "*Home & Away*". Il est composé de quatre fonctions "*Calculate Risk*", "*Calculate Premium*", "*Create Policy*" et "*Store Policy*" regroupées en une fonction appelée "*Policy Creation*". Ces fonctions utilisent les données "*Insurance Request Data*", "*Insurance Policy Data*" et "*Customer File Data*".

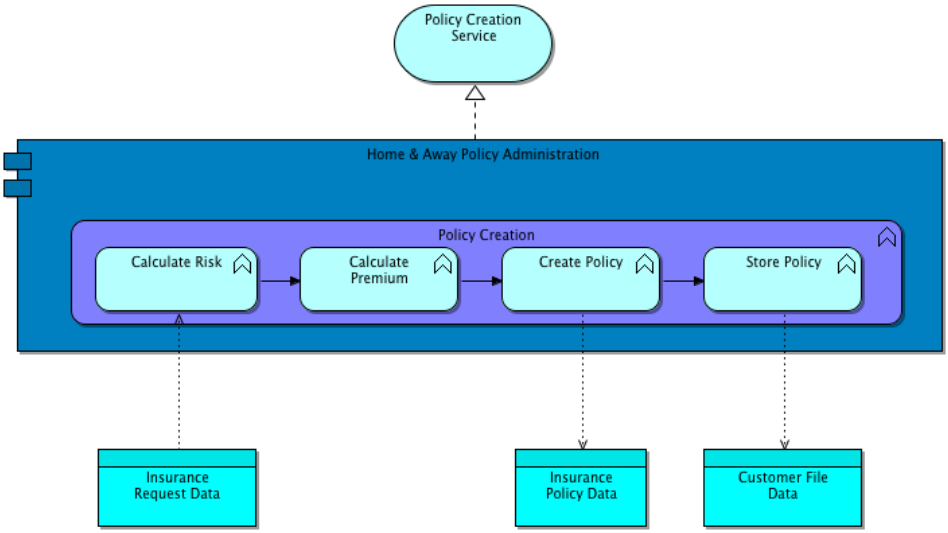


FIGURE 3.23 – Service applicatif de création d'une police d'assurance avec Archimate

On remarque que les fonctions s'enchaînent dans un ordre précis :

1. calcul du risque à partir du registre d'assurance ;
2. calcul de la prime ;
3. création de la police ;
4. enregistrement dans le fichier client.

Cet enchaînement décrit le fonctionnement de la création d'une police d'assurance.

EXEMPLE 3.6 – *UML Component Diagram*

La figure 3.24 montre une application de vente en ligne composée de sept composants : "*ShoppingCart*", "*Service*", "*Order*", "*Product*", "*BackOrder*", "*Customer*" et "*Organization*. Trois interfaces sont également décrites :

- **"OrderEntry"** entre *ShoppingCart* et *Order* pour permettre à la commande d'accéder aux éléments du panier ;
- **"Person"** entre *Customer*, *Order*, *Organization* et *BackOrder* pour que le client puisse retourner sa commande à l'entreprise ;
- **"OrderableItem"** entre *Service*, *Product* et *Order* pour fournir les éléments commandables.

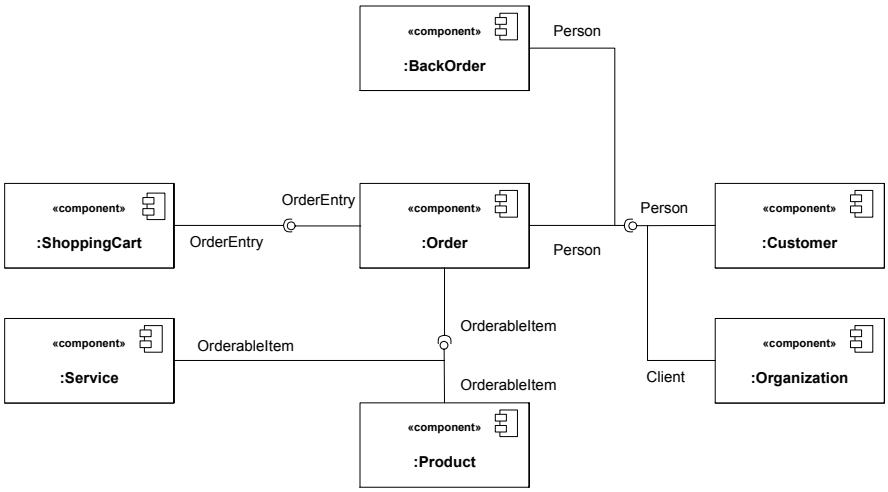


FIGURE 3.24 – Description des composants d'une vente en ligne avec UML

Cet exemple permet de connaître les interactions entre composants via les interfaces. Néanmoins, cette notation UML ne détaille pas les différentes fonctions des composants, il s'agit d'une vue structurelle et non comportementale. Et la nature du composant ne précise pas s'il réalise des traitements ou encapsule des données. Les composants *Service*, *Product*, *Customer* et *Organization* semblent être des objets de données.

EXEMPLE 3.7 – SCA

La figure 3.25 montre l'exemple d'un service de paiement d'une commande en ligne. Le service est réalisé à l'aide d'un composant nommé "Payments" faisant appel lui-même à trois composants "Customer Account", "Product Princing" et "Account Ledger" ; ainsi qu'à un service externe "External Banking".

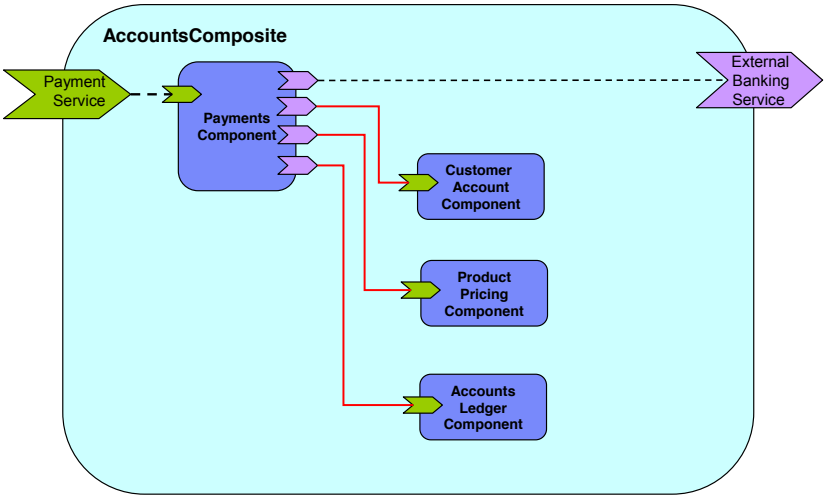


FIGURE 3.25 – Service applicatif de paiement d'une commande en ligne

Dans cet exemple aucune donnée n'est décrite.

Ces trois exemples permettent de comparer trois notations pour la description d'architecture applicative orientée composant. La représentation Archimate couvre les concepts de composants, de services, d'interfaces, de fonctions et données. De son côté, le diagramme à composants UML couvre les concepts de composants, d'interfaces, de fonctions et de ports. SCA ne représente pas les données. Une synthèse comparative est donnée dans le tableau 3.7.

3.3.3 Définition du méta-modèle

Notre méta-modèle est largement inspiré de Archimate (cf. figure 3.26), nous avons vu dans les exemples que cette notation a une bonne couverture des concepts pour décrire une architecture applicative. Nous avons réutilisé les concepts définis dans le méta-modèle de Archimate et fait des modifications au niveau de leurs relations. Archimate réutilise des liens avec une nomenclature générique pour ses différentes couches, ce qui impacte la compréhension des liens, leur ontologie n'est donc pas pleinement spécifique à chaque contexte.

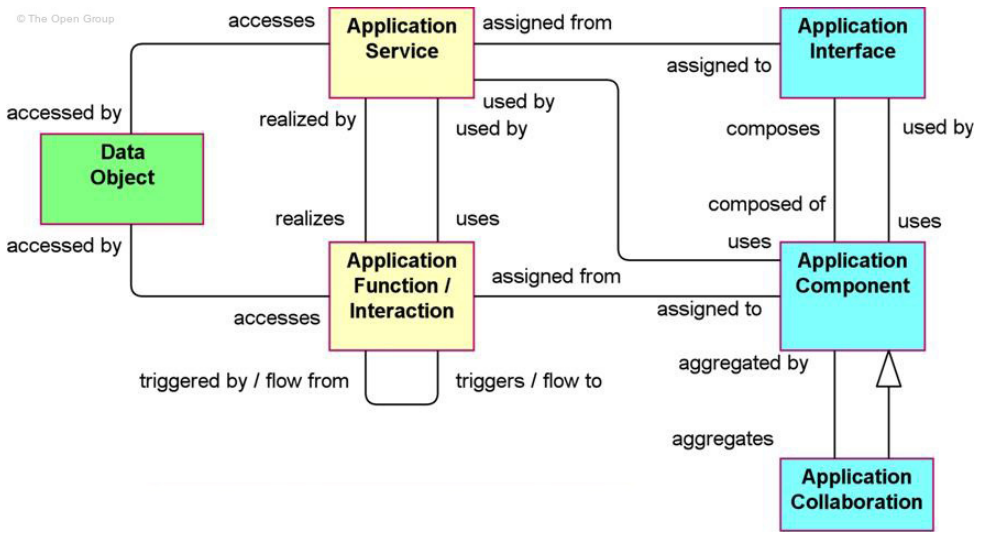


FIGURE 3.26 – Le méta-modèle applicatif de Archimate 2.1 [93]

Le méta-modèle en figure 3.27 est une ré-interprétation de Archimate. Le composant applicatif (*ApplicationComponent*) est utilisé par une interface (*Interface*). L'interface assigne des services (*Service*) utilisés par les composants application par la réalisation de fonctions (*Function*). Un service peut appeler un autre service. La fonction peut être appelée par une autre fonction. La fonction accède à des objets de données (*DataObject*).

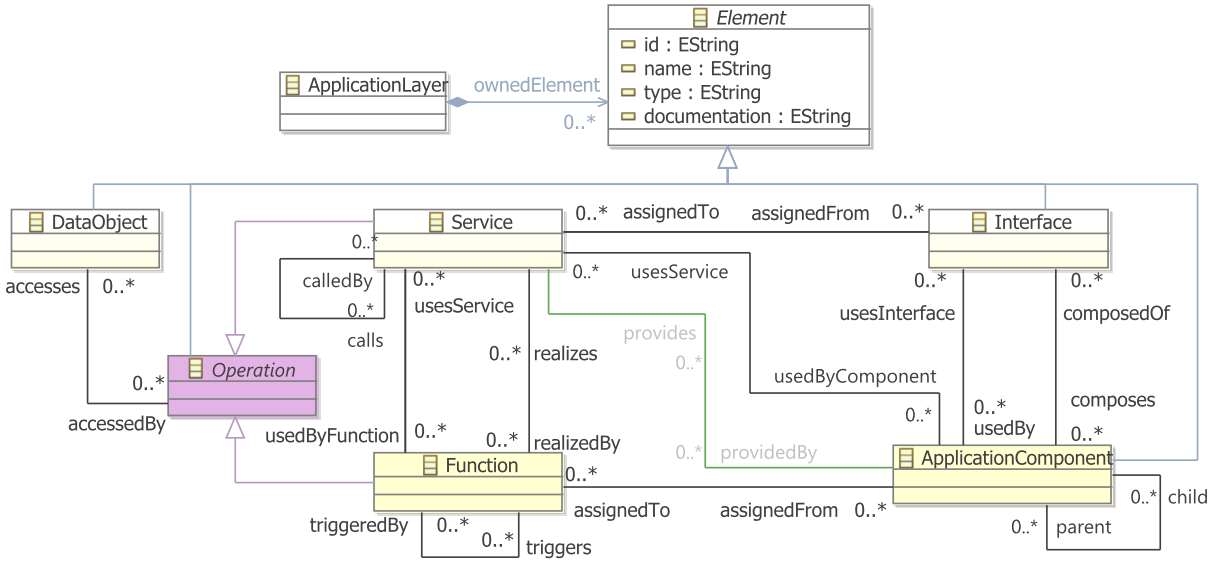


FIGURE 3.27 – Méta-modèle applicatif modélisé avec le cadriciel Eclipse EMF

La différence majeure entre Archimate et notre méta-modèle est la composition sur le composant applicatif qui permet d'avoir des composants fils et parents.

TABLE 3.6 – Description des concepts applicatifs

Symbole	Concept	Description
	*ApplicationLayer*	Concept technique contenant tous les éléments de la cartographie. Il correspond au nœud racine de l'arbre XML lorsque le modèle est sérialisé en XMI.
	Element	Concept technique définissant les attributs communs à tous les autres concepts du méta-modèle. Un nom (*name*), un type (*type*), un identifiant unique (*id*) et des commentaires (*documentation*).
	ApplicationComponent	Concept représentant la totalité, une partie, ou une sous-partie d'une application réutilisable et remplaçable qui fournit des fonctionnalités par exécution.
	Interface	Concept procurant un point d'accès aux services, utilisable par un utilisateur ou un autre composant applicatif. L'interface spécifie un contrat : paramètre, type de donnée, pre/-post condition.
	Operation	Concept abstrait de *Service* et *Function*. Une opération accède à une donnée (*DataObject*).
	Service	Concept mettant en œuvre la fonctionnalité d'un composant applicatif.
	Function	Comportement d'une fonctionnalité d'un composant exposé par un ou plusieurs services. La fonction peut utiliser des objets de données (DataObject).
	DataObject	Concept d'objet de donnée représentant un élément informatique passif utilisé par les fonctions et services. Il possède des propriétés : attributs et associations. Son cycle de vie peut être temporaire : flux de message ; ou permanent : persistance en base de données.

Pour vérifier la cohérence des instances de ce méta-modèle, nous avons écrit avec le langage OCL quatre règles de conformité[8] du modèle, voir la définition en liste 3.2. La règle portant la fonction (*Function*) vérifie qu'elle appartient bien à un composant applicatif. Deux règles concernent le service, la première vérifie qu'il est affecté à une interface, la seconde contrôle si le service est bien réalisé par la même fonction que les relations associant l'interface et le composant applicatif. La règle sur la donnée (*DataObject*) vérifie qu'elle est accédée à la fois par une fonction et le service réalisant cette fonction.

Listing 3.2 – WFR Application

```
context Function
inv: self.assignedFrom -> notEmpty()

context Service
inv: self.assignedFrom -> notEmpty()

context Service
self.realizedBy.assignedFrom.composedOf.assignedTo -> exists(s | s = self)

context DataObject
inv: self.accessedBy.realizes.accesses -> exists(s | s = self)
```

Les règles ont été ajoutées au cœur du méta-modèle, EMF supportant nativement OCL comme mécanisme de validation. Les règles sont donc invoquées à chaque manipulation des concepts concernés.

[8] Well-formed ou conformance en anglais

3.3.4 Comparaison

Le méta-modèle défini dans le présent document est inspiré de l'*Application Layer* provenant d'ArchiMate, ainsi les concepts sont très similaires. Nous avons également comparé notre méta-modèle avec deux notations orientées composants :

- **SCA (*Service Component Architecture*)** est une notation pour décrire des applications construites dans une architecture orientée service. Une application SCA est un assemblage de composants implémentant une logique métier qui offre et peut requérir d'autres services par références.
- **Diagramme de composants UML** est une notation qui permet de décrire les composants d'un système et leur interface ainsi que les interactions entre composants.

TABLE 3.7 – Comparaison du méta-modèle Applicatif avec Archimate, SCA et UML

App	Archimate	SCA	UML Component Diagram
ApplicationComponent	Component	Component	Component
Interface	Interface	Interface	Interface / Port
Service	Service	Service / Reference	Operation
Function	Function	Operation	Operation
DataObject	DataObject	*	*

* Aucun concept équivalent n'est existant

Les concepts de Archimate et de notre méta-modèle sont équivalents puisque nous nous en sommes inspirés. Dans SCA nous retrouvons des concepts équivalents à l'exception des objets de données qui ne sont pas présents. Enfin, le diagramme de composants d'UML est très minimaliste, seuls les composants et interfaces sont représentés. Notre méta-modèle couvre l'intégralité des concepts de ces notations, il est donc possible de traduire ces langages vers App.

3.3.5 Expérimentations

L'expérimentation décrite en figures 3.28, 3.29 et 3.30 est une ré-interprétation des exemples donnés en section 3.3.2, tandis que la figure 3.31 est un extrait d'un cas d'étude qui sera mis en œuvre dans le chapitre 5. La modélisation avec notre langage de modélisation permet de vérifier la pertinence de notre méta-modèle par l'expérimentation.

EXPÉRIMENTATION 3.4 – Archisurance

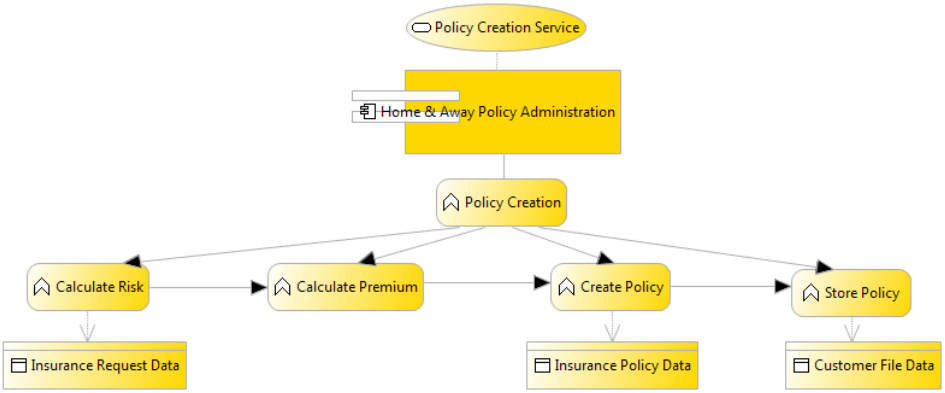


FIGURE 3.28 – Service applicatif de création d'une police d'assurance avec App

Notre méta-modèle étant très similaire à celui d'Archimate, la traduction n'a posé aucune difficulté. Seule la représentation visuelle présente des différences. Les fonctions sont graphiquement hors des composants, et la fonction principale "*Policy Creation*" ne contient pas graphiquement les autres composants qu'elle appelle.

EXPÉRIMENTATION 3.5 – UML Component

La ré-interprétation du diagramme UML à l'aide de notre méta-modèle a été simple. Une différence existe visuellement au niveau de chaque interface.

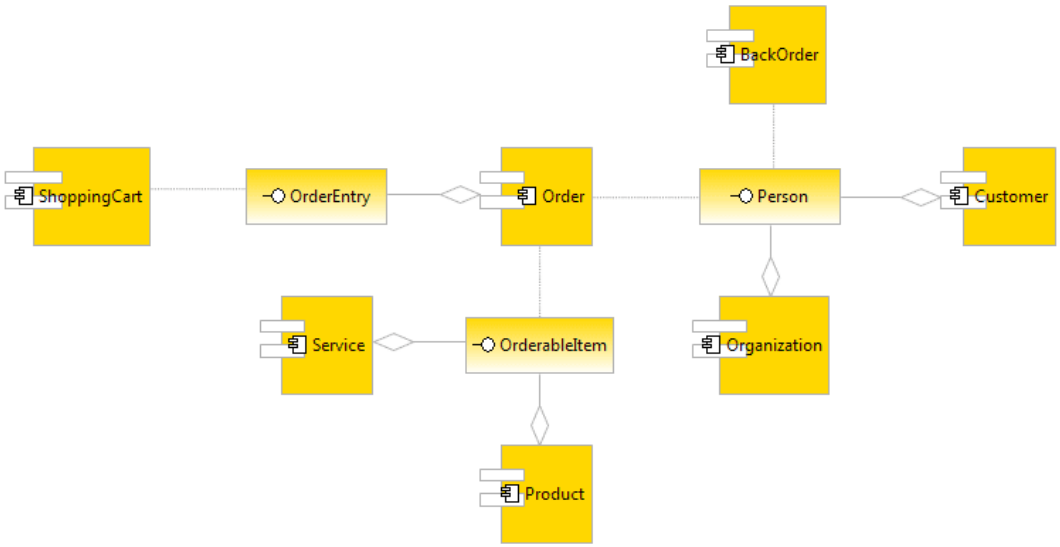


FIGURE 3.29 – Exemple de cartographie applicative

EXPÉRIMENTATION 3.6 – SCA

Notre méta-modèle est pleinement compatible avec les concepts issus de la notation SCA. Néanmoins, la représentation visuelle a une logique moins adaptée pour représenter un service et sa réalisation par les composants.

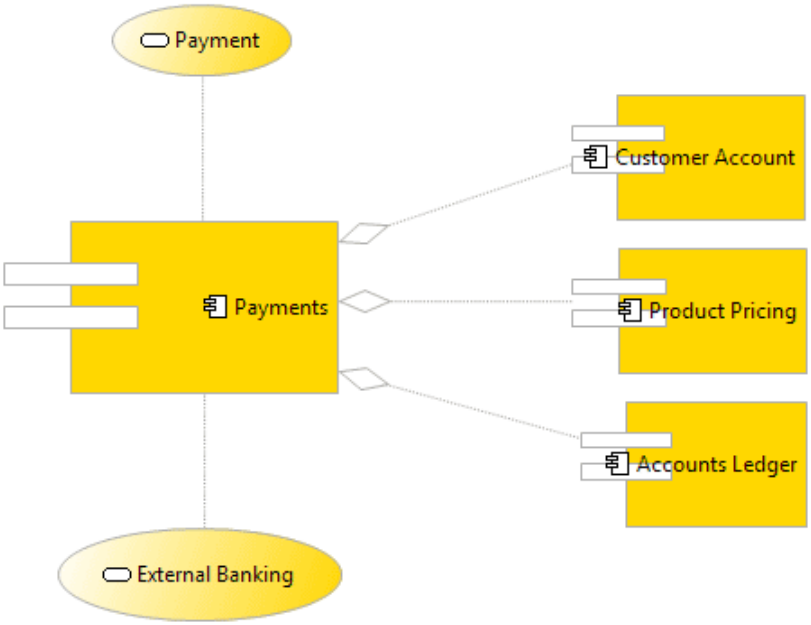


FIGURE 3.30 – Exemple de cartographie applicative

EXPÉRIMENTATION 3.7 – Extrait d'une rétro-ingénierie vers App

Cette expérimentation est un extrait d'un diagramme applicatif issu de la rétro-modélisation d'un cas provenant d'une assurance mutuelle. Le code source de l'application est volumineux, il est composé de plus de 30 0000 classes.

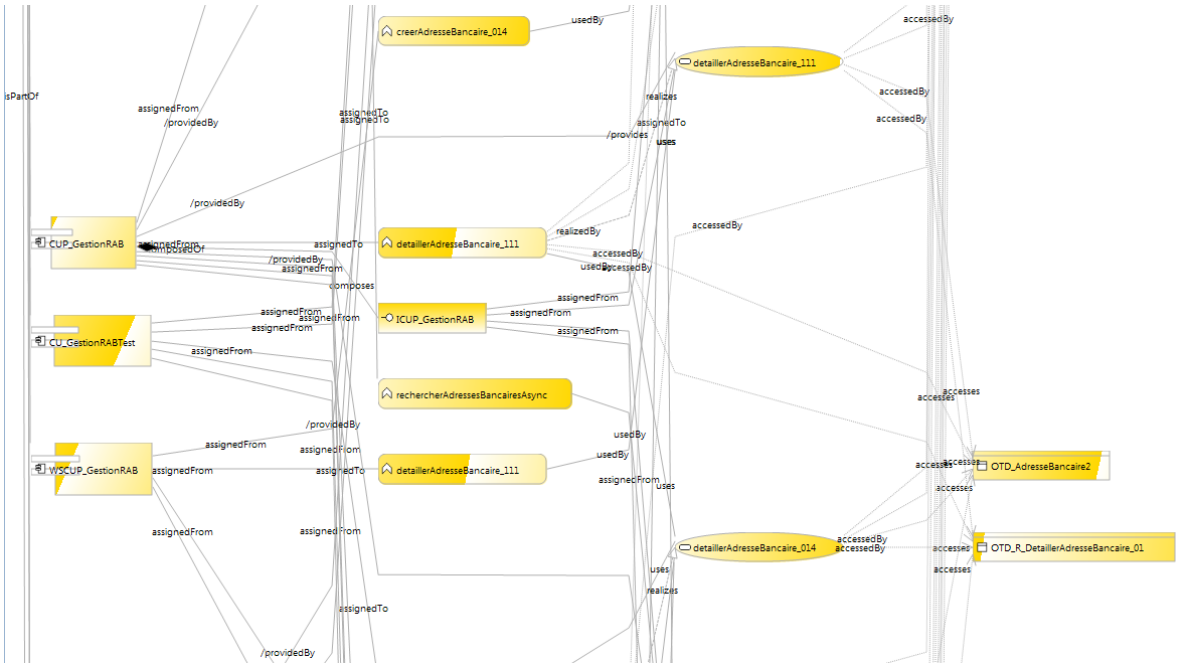


FIGURE 3.31 – Exemple de cartographie applicative

La représentation visuelle par un diagramme représenté dans la figure 3.31 est difficilement exploitable. Pour des modèles d'envergure, il faut donc privilégier d'autres modes de consultation/modification, par exemple : la vue arborescente, le filtrage par champs de recherche, ou la décomposition en sous-modèles si un découpage est possible.

Notre méta-modèle applicatif est suffisamment générique pour interpréter les exemples étudiés en début de cette section. Néanmoins, chaque notation adopte une représentation graphique bien plus adaptée et pertinente selon l'angle prépondérant des concepts modélisés : service ou composant.

3.3.6 Bilan

Le point de vue applicatif permet de représenter le système informatique et ses différents composants de traitements et de données. L'étude d'exemples et de différents langages dédiés à la représentation d'architecture applicative nous a permis d'élaborer un méta-modèle applicatif générique. Les principaux concepts sont les composants applicatifs, les fonctions, les interfaces, les services et les objets de données. Les architectures applicatives peuvent contenir pour chacun de ces concepts des déclinaisons spécifiques. Par exemple, les composants applicatifs peuvent être de natures différentes : script, batch, application monolithique, composant, etc. L'attribut type sur chacun des concepts permet de réaliser une nomenclature spécifique et de préciser leur nature.

Conclusion

Nous avons définis trois méta-modèles qui permettent de représenter les principaux points de vue d'un SI pour ce qui est des domaines métier et informatique : processus, fonctionnel et applicatif.

Le point de vue processus montre comment se réalisent les activités de l'entreprise. Le point de vue fonctionnel permet de décrire par découpage les fonctions principales qui composent l'entreprise. Le point de vue applicatif détaille l'implémentation de l'activité dans le domaine logiciel. Ces trois points de vue séparent les traitements (fonction dans Fun, service dans App, activité et tâche dans BPM) des données.

Les trois méta-modèles **BPM**, **Fun** et **App** ne constituent pas une simple proposition pour cartographier les couches processus, fonctionnel et applicative du SI ; il s'agit du socle essentiel pour notre définition de l'alignement.

Note

Généricité de l'approche Ces représentations sont compatibles avec les principales notations équivalentes, et il est possible de passer aisément de l'un à l'autre par équivalence des concepts en commun.

Dans le prochain chapitre, nous définirons les liens entre les méta-modèles permettant de "traverser" le SI. Cette traçabilité ouvre des possibilités pour apporter de la flexibilité au SI. Une analyse plus profonde de l'alignement permet de déterminer lors des changements stratégiques et organisationnels de l'entreprise les modifications à appliquer au parc applicatif.

4

Alignement du SI

Sommaire

Introduction

Dans le chapitre précédent, nous avons défini trois points de vue décrivant les concepts du SI aux interfaces entre domaine métier et informatique : processus métier, fonctionnel et applicatif. La modélisation de ces points de vue s'inscrit comme une des étapes du processus d'architecture d'entreprise. Ce dernier est composé de différentes étapes décrites par le cycle ADM de TOGAF ou par la méthode d'urbanisation de Longépé (cf. chapitre 2).

1. **étape d'analyse de l'existant** du SI par la cartographie ou mise à jour des différents points de vue ;
2. **étape d'alignement** des concepts des différents points de vue issus de la cartographie ;
3. **étape de migration et d'évolution** du SI à partir du plan de convergence des ajouts, modifications et suppressions à appliquer.

L'étape d'alignement fait le lien entre l'analyse de l'existant et la migration du SI. L'alignement permet la détection des incohérences et mesure la portée des modifications à apporter.

Problème

La vision usuelle, idéaliste et même utopiste illustrée par la figure 4.1 est couramment rencontrée dans la littérature décrivant les SI. Cette pile réduit l'alignement à une ligne idéale de traçabilité traversant les points de vue. L'alignement est défini alors comme un enchaînement de concepts d'un point de vue vers un autre point de vue.

Dans la pratique, le point de vue fonctionnel est plutôt orthogonal aux points de vue processus et applicatif.

Nous réduisons le problème d'alignement à la frontière entre domaine métier et informatique, car nous considérons les hypothèses suivantes :

- L'alignement entre l'organisation et l'infrastructure perd de l'importance compte tenu de la démocratisation de l'**externalisation** des supports informatiques par les services d'infogérance et de *cloud*[1]. Il n'est plus nécessaire d'avoir sur un même lieu le personnel et les machines contenant les traitements et données liés à l'activité de l'entreprise. Par exemple : Google Drive et Microsoft Office 365 proposent une suite bureautique (traitement de texte, tableur et présentation) entièrement externalisée dans le cloud. L'utilisateur a seulement besoin d'un compte (identifiant / mot de passe) et d'un navigateur Internet. Plus aucun logiciel ne nécessite d'être installé sur son poste de travail et les documents sont stockés en ligne.
- L'alignement idéal relie les éléments de la couche stratégie aux programmes par les modèles métiers. Mais souvent, le niveau stratégique n'est lui-même pas exprimé par des modèles stratégiques de besoins exploitables (cf. section 7.3.1) : *e3value*, *Resource Event Agent* (REA), *Business Model Ontology* (BMO), *i**, *Strategic Dependency Model* (SD), *Strategic Rationale Model* (SR) [86].
- Le code source des logiciels est de trop bas niveau et contient beaucoup trop de détails liés aux langages de programmation et aux patrons de conception.

[1] Informatique dématérialisée

Par exemple, le patron de conception d'objet d'accès aux données (DAO) est un ensemble de classes techniques pour lire, écrire et modifier des données persistantes, ce code source ne présente aucune règle de gestion métier.

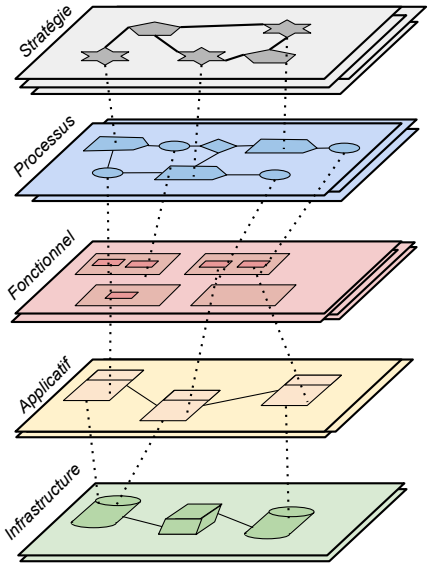


FIGURE 4.1 – La vision idéalisée du SI

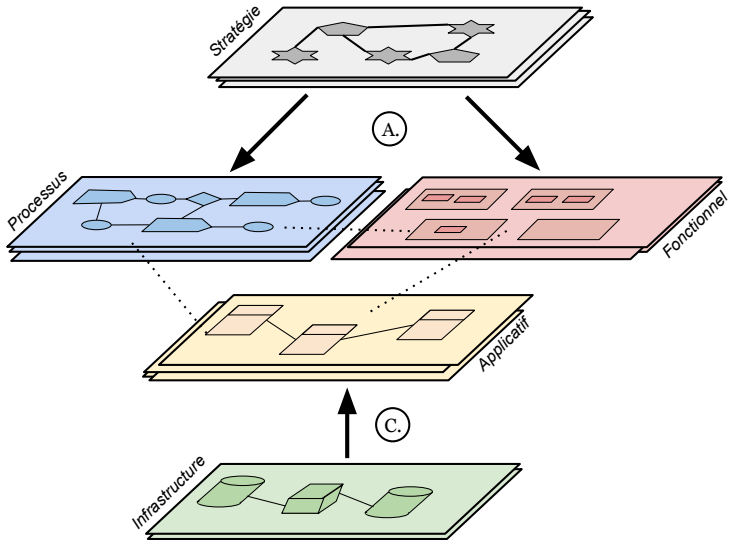


FIGURE 4.2 – Notre approche pragmatique par couplage

Solution proposée

À la différence de la vision idéaliste, nous proposons un rapprochement des trois points de vue considérés (processus, fonctionnel et applicatif) par couplage illustré par la figure 4.2. Cette vision est plus souple et permet de considérer le niveau de maturité différent des entreprises dans la cartographie de leur SI. En effet, pour modéliser les points de vue métier, certaines entreprises pratiquent uniquement la modélisation fonctionnelle qui est plus simple, tandis que d'autres entreprises, avec une plus grande expertise, pratiquent la modélisation de processus, plus complexe.

Nous proposons le rapprochement des points de vue métier et applicatif :

(C.) en **concrétisant** le niveau stratégie par des modèles de processus métier décrivant l'activité de l'entreprise, et par des modèles fonctionnels décrivant l'organisation de l'entreprise ;

(A.) en masquant les détails d'implémentation, par processus d'**abstraction** ou de rétro-ingénierie, jusqu'à un niveau acceptable et compréhensible pour un "langage commun" entre informatique et métier.

Le langage commun est défini comme un tissage entre le langage de chaque point de vue, c'est-à-dire une correspondance par alignement entre concepts. Dans l'état de l'art (cf. 2.4) nous avons donné une typologie des différents types d'alignement : spatial, temporel et *Business IT alignement* (BITA). Les considérations temporelles de co-évolution sont écartées de nos travaux à ce stade. L'objet de la section suivante 4.1 est de définir l'alignement spatial entre nos différents méta-modèles de chaque point de vue : BPM, Fun et App.

Les aspects techniques de la réalisation de l'alignement seront décrits dans la section 4.3, et les étapes de rapprochement et d'abstraction seront décrites dans le chapitre suivant 5.

4.1 Une définition de l'alignement par les méta-modèles

Aligner c'est mettre en correspondance des éléments issus de points de vue différents. On met en œuvre l'alignement par des liens entre les instances de modèles correspondant à chaque point de vue. Ces liens sont spécifiés au niveau méta-modèle par des associations. Notre définition de l'alignement vise à identifier les associations entre concepts des méta-modèles BPM, Fun et App.

DÉFINITION 4.1 – Relation d'association

Une association est la relation sémantique de deux concepts caractérisée par un nom. Exemple, un service applicatif implémente une activité d'un processus, le lien portera le nom « *implémentéPar* ». L'association possède une navigabilité : monodirectionnelle, le lien possède un sens représenté visuellement par une flèche ; ou bidirectionnelle, le lien est visuellement représenté par un simple trait. L'association possède également une multiplicité (ou cardinalité) qui est le nombre minimum et maximum d'instances des concepts composant la relation : zéro ou une instance (0-1), zéro ou plusieurs instances (0-n), une et une seule instance (1-1), une ou plusieurs instances (1,n).

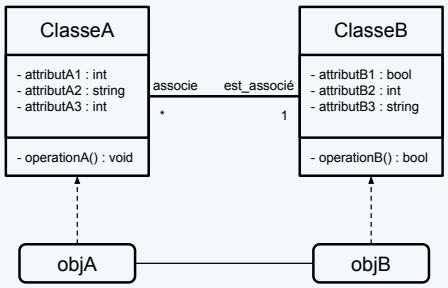


FIGURE 4.3 – Exemple de relation d'association

Nous avons utilisé le principe de séparation des préoccupations[2] [70]. Ce principe vise la séparation en points de vue (cf. section 2.1.2), le découpage en fonctions/services métier, mais aussi la séparation entre les données et les traitements afin d'éviter des couplages forts. La séparation permet la souplesse dans l'évolution des logiciels et donc du SI. En effet, les données et les traitements ont des cycles de vie différents. Il est possible de modifier les traitements sans altérer les données, mais à l'inverse une modification de structure de données engendre une modification des signatures de traitements. De plus, les données peuvent être utilisées par plusieurs traitements. Ainsi, les données sont manipulées par les traitements, et les traitements sont dépendants des données. Pour résoudre ce problème d'interdépendance, la méthode de conception et développement Merise par exemple propose également la séparation des modèles de conception : données (MCD) et traitements (MCT). Nous avons donc défini deux types de liens pour l'alignement : les **traitements** et les **données**.

DÉFINITION 4.2 – Traitement et donnée

Traitement *Le traitement est le fonctionnement du SI perçu à travers ses couplages avec le système opérant (applications et logiciels) et le système de pilotage (management stratégique). Décrire les traitements, c'est décrire les processus mis en œuvre dans le système en interaction avec son environnement.* - Ingénierie des systèmes d'information, p.73 [78]

Donnée *La donnée est la représentation des propriétés définissant les réalités de l'entreprise.* - L'essentiel sur Merise, p.48 [39]

Notre travail vise à définir un alignement générique à partir des liens constatés dans la variété des méthodes d'architecture étudiées supportant les points de vue métier

[2] *Separation of concerns* en anglais

et applicatif. Dans les cadres TOGAF [92] ou ArchiMate [94], un lien "*realizes*" est défini entre les services métiers et les services applicatifs (cf. figure 4.4). L'alignement devient plus simple si on dispose d'une notion de service dans les points de vue métier et applicatif. Cependant, toutes les applications ne suivent pas une architecture orientée service (SOA). En particulier, les applications anciennes programmées en Cobol issues de patrimoines dans les domaines de la banque ou de l'assurance.

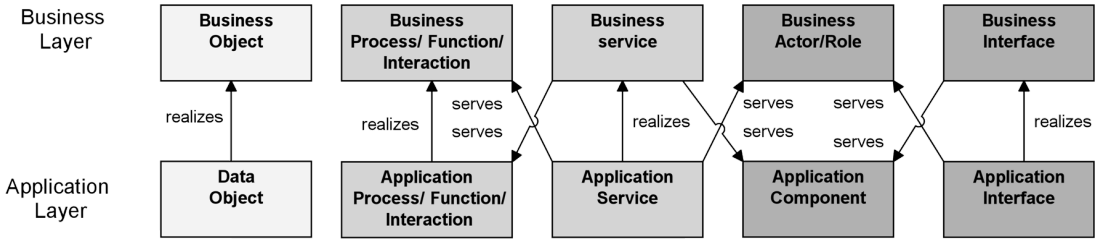


FIGURE 4.4 – Dépendances entre les points de vue métier et applicatif, Archimate 3.0 [94]

Dans d'autres travaux [40, 88] nous constatons que le lien est établi entre activités métiers (ou tâches) au sens du processus et des fonctionnalités d'applications (cf. figure 4.5).

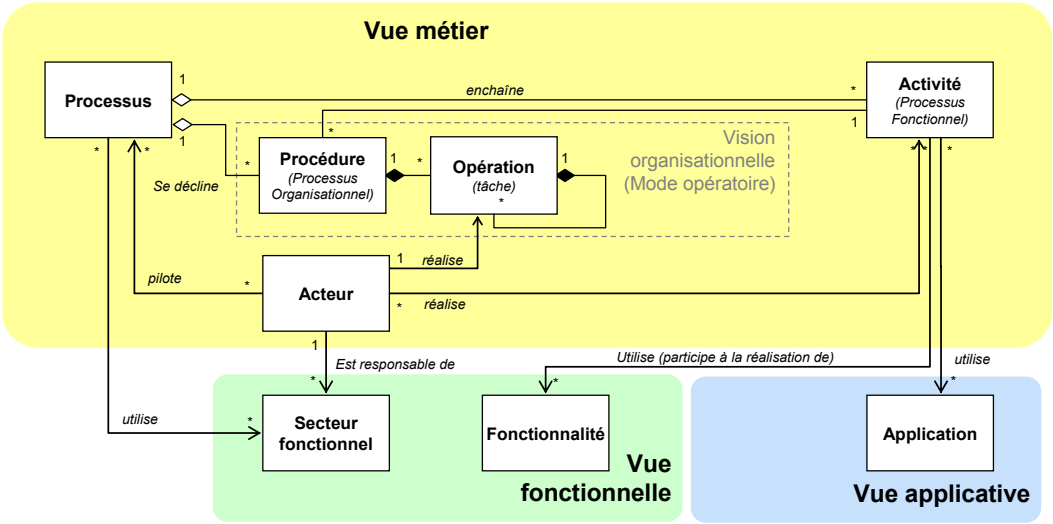


FIGURE 4.5 – Liens entre les points de vue processus, fonctionnel et applicatif du Cadre commun d'urbanisation de la DISIC [40]

Afin de réaliser l'alignement entre les trois points de vue définis dans le chapitre 3 : *BPM*, *Fun* et *App*, notre définition est modélisée en figure 4.6. Le schéma présente uniquement les concepts possédant des liens avec les autres méta-modèles, le reste des concepts est masqué, il faut se reporter aux méta-modèles respectifs. Les liens sont détaillés dans les deux sections à suivre par leur typologie : traitement et donnée.

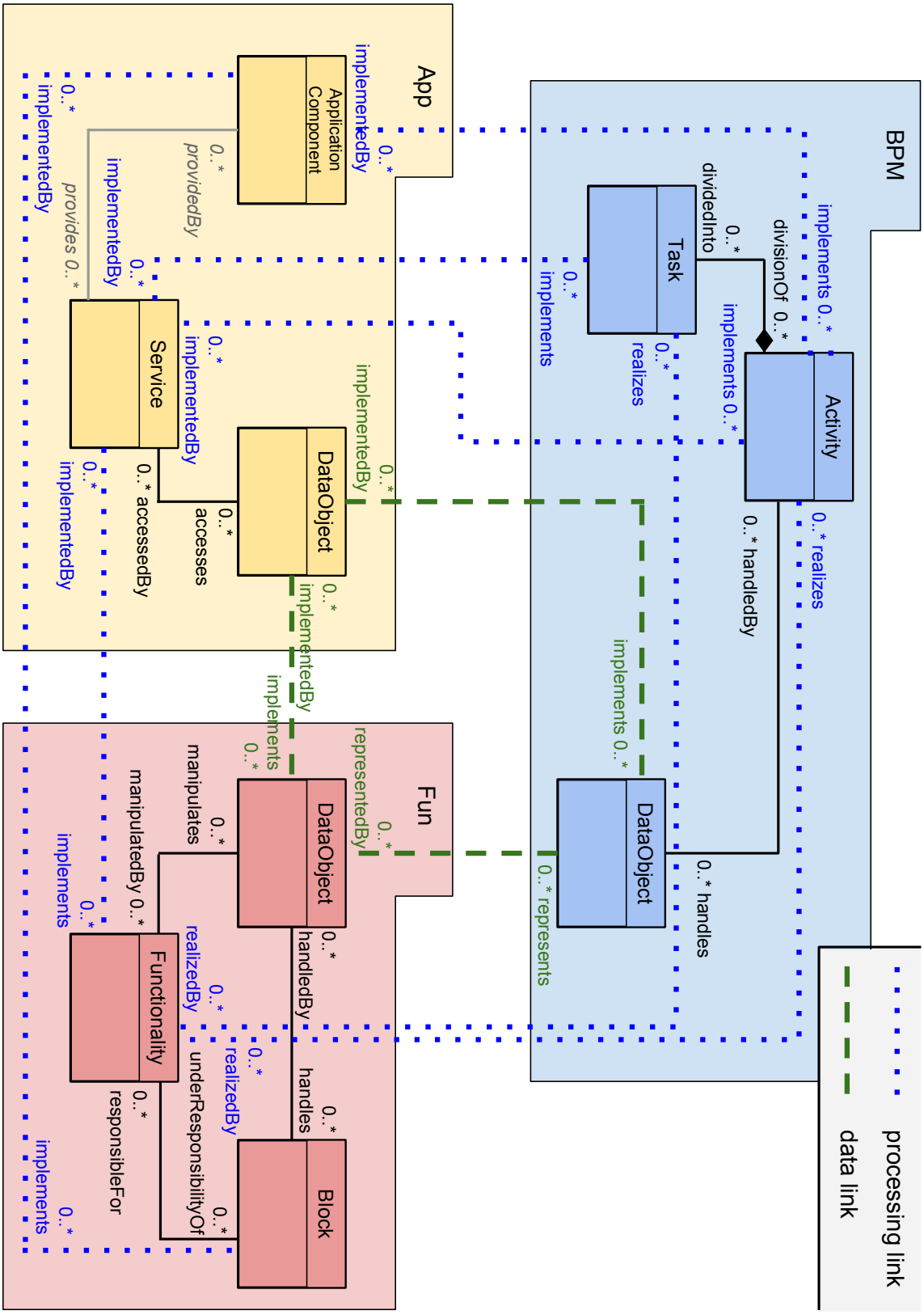


FIGURE 4.6 – Définition de l'alignement par liens inter méta-modèles

4.1.1 Liens des concepts de traitements

Nous définissons le terme de "*lien par les traitements*" pour les relations d'associations entre les concepts qui définissent des traitements du SI. Les concepts de traitement dans les différents méta-modèles sont : les tâches, les activités, les fonctionnalités, les services et les composants applicatifs. Ces concepts sont similaires, mais à des niveaux d'abstraction différents, correspondant au point de vue auquel ils appartiennent : processus métier, fonctionnel ou applicatif. Par exemple, les tâches ou activités (processus) rendent des services (applicatif) ou des fonctionnalités (fonctionnel). Deux concepts sont associés par deux types de lien possibles : `implémente`[3] ou `réalise`[4].

DÉFINITION 4.3 – Implémenter et réaliser

Implémenter *Effectuer l'ensemble des opérations qui permettent de définir un projet et de le réaliser, de l'analyse du besoin à l'installation et à la mise en service du système ou du produit.* - Journal officiel du 20/04/2007

Réaliser *Rendre réel et effectif ; construire, élaborer quelque chose qui était à l'état de projet, de pensée.*

La figure 4.7 synthétise les liens d'association entre les concepts qui définissent l'alignement par les traitements.

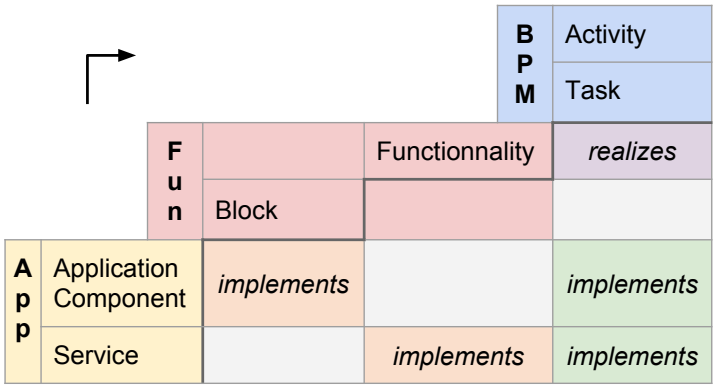


FIGURE 4.7 – Alignement des concepts de traitements

Le couplage de concepts se fait entre deux méta-modèles :

BPM ↔ Fun Les fonctionnalités du méta-modèle Fun réalisent des activités ou des tâches du méta-modèle BPM.

Fun ↔ App Les fonctionnalités du méta-modèle Fun sont implémentées par des Services du méta-modèle App.

BPM ↔ App En fonction du niveau de granularité de la modélisation processus à l'aide du méta-modèle BPM : soit les tâches sont détaillées et sont implémentées par les services ; soit uniquement les activités sont représentées et elles sont implémentées par les services. Dans un scénario où l'architecture ne serait pas orientée service, les activités peuvent être directement implémentées par un composant applicatif.

Le couplage entre les deux méta-modèles métier (BPM et Fun) est de type `réalise`, et le couplage entre un des deux modèles métier et le méta-modèle applicatif (App) est de type `implémente`.

[3] *Implements* en anglais
[4] *Realize* en anglais

4.1.2 Liens des concepts de données

Dans chaque modèle, les concepts de données sont définis par un *objet de donnée* (**DataObject**), le lien avec les autres modèles est une association entre objets de données conformes à leurs propriétés d'attributs. Entre les concepts des méta-modèles processus (BPM) ou fonctionnel (Fun) et le méta-modèle applicatif (App), nous retrouvons le couplage de type `implémente`. Tandis qu'entre les méta-modèles processus et fonctionnel, le lien est de type `correspond` (cf. définition 4.1.3). La figure 4.8 synthétise les liens d'association entre les concepts qui définissent l'alignement par les traitements.

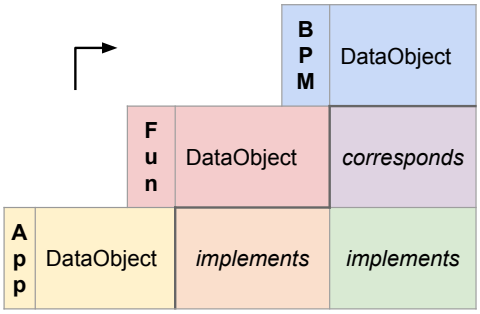


FIGURE 4.8 – Alignement par les données

Le couplage des concepts se fait par paire de méta-modèles :

BPM ↔ Fun Les objets de donnée du méta-modèle Fun correspondent aux objets de donnée du méta-modèle BPM.

Fun ↔ App Les objets de donnée du méta-modèle Fun sont implémentés par les objets de donnée du méta-modèle App.

BPM ↔ App Les objets de donnée du méta-modèle BPM sont implémentés par les objets de donnée du méta-modèle App.

Les objets de donnée alignables sont uniquement des objets de type métier (cf. définition 4.1.2).

DÉFINITION 4.4 – Objet métier

L'objet métier est un concept ayant du sens pour des acteurs d'une organisation. Il porte des données informatives (ses attributs) et il est manipulé par ces acteurs dans le cadre de l'exécution des processus. Il représente les concepts invariants du métier de l'entreprise tels que le client, le contrat ou la facture. En proposant une vision indépendante des aspects techniques (outils, application, base de données...), il permet de répondre aux enjeux et besoins des métiers.

Les référentiels du SI, p.67 [13]

Il ne faut pas confondre les objets de données métier avec les objets de données techniques. Ces objets de données sont présents dans les architectures logicielles. Ils sont utilisés par les couches d'inter-médiation construite par les API, typiquement dans les conceptions logicielles d'accès aux données. Ils servent uniquement au fonctionnement du logiciel lui-même. Ces objets de données techniques peuvent se retrouver dans le modèle applicatif si le niveau d'abstraction est proche du code source. Ces objets de données ne doivent pas être alignés avec les modèles métiers (processus et fonctionnel).

4.1.3 Autre définition de l'alignement

Un travail de définition au niveau méta-modèle a été proposé dans la thèse de Anne Etien [44]. Deux types de lien ont été déterminés à partir de la littérature : `correspond` et `représente`.

DÉFINITION 4.5 – Correspond et représente

Correspond *Exprime une égalité (à un isomorphisme près) entre des concepts similaires de méta-modèles différents.* [44]

Représente *Spécifie qu'un concept d'un méta-modèle a un impact sur un concept d'un autre méta-modèle : affecte le comportement, la valeur ou l'existence.* [44]

La figure 4.9 reprend la hiérarchie de niveaux de modélisation M0 à M3 communément utilisés dans la communauté d'ingénierie des modèles. Notre définition de lien inter méta-modèles se situe au niveau M2 (*Meta-Model EA*) et les types de liens `représente` et `correspond` sont à un niveau d'abstraction supérieur *Meta-Concept* toujours au niveau M2.

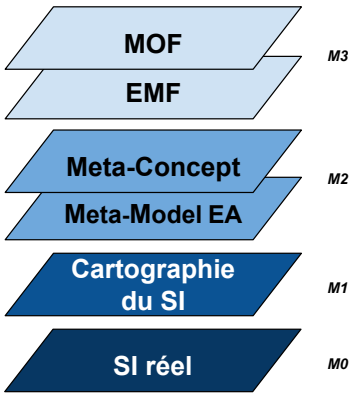


FIGURE 4.9 – Notre pile de modélisation EA

Nous constatons une relation d'instance entre les liens `correspond` et `représente`, et les liens de traitement `implémente` et `réalise` que nous avons définis :

- Nos six liens `implémente` entre les modèles métiers (processus et fonctionnel) et applicatif qu'ils soient de type traitement ou donnée sont des instances des liens de type `représente`.
- Notre lien `réalise` entre les modèles métiers eux-mêmes (processus et fonctionnel) de type traitement ou donnée sont des instances du lien de type `correspond`, tandis que le lien de type donnée entre les modèles métier est déjà de type `correspond`.

Cette différence entre les travaux d'Anne Etien et les nôtres est due à ces deux définitions de l'alignement qui se situent à des niveaux d'abstraction différents. Notre définition est au plus proche des concepts des différents points de vue du SI, nous spécifions lien par lien l'association possible entre deux concepts provenant de deux des trois méta-modèles (BPM, Fun et App). Tandis que les travaux d'Anne Etien définissent des liens d'alignement entre concepts plus génériques quels que soient les points de vue étudiés.

4.1.4 Définition de l'alignement adaptable et extensible

La définition de l'alignement que nous avons synthétisée en figure 4.6 est dépendante de nos trois méta-modèles : App, Fun et BPM. Toutefois, la démarche menant à cette définition est générique et adaptable. Il est possible d'adapter ou d'étendre notre définition de l'alignement en fonction des besoins.

Adaptation

- une entreprise possède déjà ses **propres méta-modèles** dans un référentiel d'entreprise ;
- une entreprise utilise une **démarche d'architecture** incluant des méta-modèles spécifiques (Togaf / Archimate...) ;
- une entreprise souhaite utiliser des notations standards (UML, BPMN, etc.).

Comme exemple d'adaptation de notre définition vers des méta-modèles équivalents aux points de vue processus métier, fonctionnel ou applicatif. Nous proposons pour chacun de nos méta-modèles, en section 3, les correspondances avec d'autres notations standards.

Extension

- une entreprise souhaite **étendre la représentation du SI** à davantage de points de vue (stratégie, infrastructure...).

Pour illustrer l'extension, nous proposons l'ajout d'un quatrième point de vue : le diagramme de classes ; en figure 4.10 pour représenter les objets de données. Sur la gauche de la figure, le paquetage EA correspond à une représentation générique des méta-modèles BPM, App, Fun auxquels appartient l'élément DataObject.

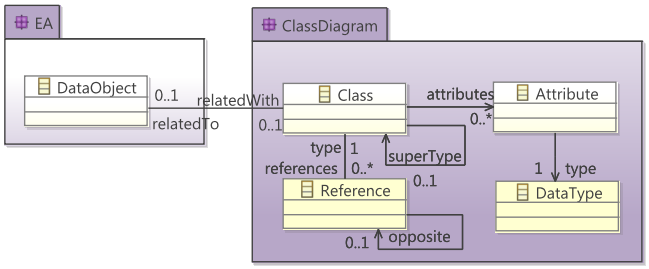


FIGURE 4.10 – Extension de l'alignement avec le diagramme de classe

Le diagramme de classes est un modèle couramment utilisé par les développeurs lors de la conception d'un logiciel orienté objet. Une classe est un ensemble de fonctions et de données. Les classes ont des relations entre elles (héritage, association, composition, dépendance...). En cas d'existence de ce diagramme, il est pertinent de l'aligner aux autres points de vue du SI grâce au lien par les données.

4.1.5 Résumé

Notre définition de l'alignement a déterminé des liens associant les concepts de nos trois méta-modèles : BPM, Fun et App. Ces liens sont de deux types : données et traitements. L'alignement permet de compléter la cartographie du SI, on ne raisonne plus en point de vue isolé, mais il est désormais possible de naviguer à travers les concepts d'un point de vue à l'autre. Par exemple, le parcours à partir d'une fonctionnalité ou d'une activité vers un composant applicatif (et réciproquement) est désormais possible. Ce cheminement ouvre des perspectives d'analyse du SI que nous verrons au chapitre 6. En effet, il sera possible de connaître les dépendances entre concepts de point de vue différents.

4.2 L'alignement par composition des modèles

Maintenant que nous avons éclairci le concept d'alignement, attachons-nous à étudier sa mise en œuvre.

Problème

D'une part, la cartographie du SI nécessite d'utiliser un langage dédié à chaque point de vue ; d'autre part, l'implémentation de l'alignement nécessite d'associer les différents concepts de chaque langage par des liens (cf. section 4.1). Or les langages n'ont aucune référence entre eux. Il est donc nécessaire d'avoir à disposition une technique de **composition de modèles** pour réaliser l'alignement.

Solution proposée

L'ingénierie des modèles (IDM) est un domaine de recherche très actif qui propose diverses techniques de composition de modèles [30]. Lors de nos recherches, nous avons sélectionné cinq techniques de composition : fusion, extension, annotation, tissage et facet.

Ces techniques disposent d'avantages et d'inconvénients. Le tableau 4.1 récapitulatif est dressé en fin de comparatif. Afin de sélectionner la technique la plus appropriée pour l'alignement, nous avons déterminé quatre critères de comparaison :

Non-intrusion : la composition ne doit pas modifier individuellement chaque modèle, car les modèles évoluent de façon itérative et indépendante.

Ontologie : la composition doit supporter des relations sémantiques pour associer les différents concepts via leur équivalence et notamment notre définition de l'alignement.

Navigation : la composition doit fournir un mécanisme pour naviguer directement du modèle source au modèle cible, et réciproquement.

Persistance : la composition doit permettre d'enregistrer les liens obtenus par sérialisation.

Pour illustrer les différentes techniques, nous utilisons un exemple en fil rouge illustré par la figure 4.11. Cet exemple simplifie l'alignement en une association entre deux concepts établie par un lien, l'alignement peut ici être considéré comme une opération binaire entre deux éléments.

Nous souhaitons aligner deux de nos méta-modèles : App et BPM ; et plus précisément des activités (*Activity*) du méta-modèle BPM avec des services du méta-modèle App. Les deux concepts sont associés par le lien de traitement `implémentéPar` (cf. section 4.1.1)

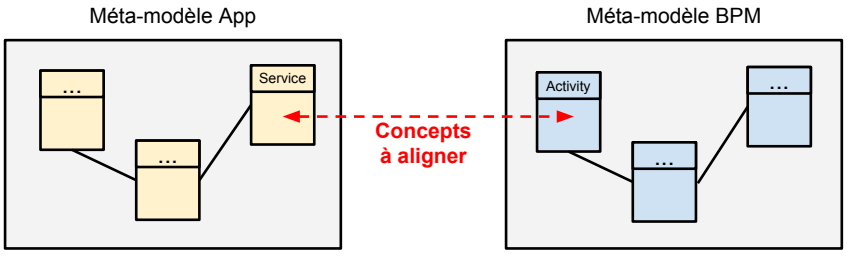


FIGURE 4.11 – Alignement de deux concepts des méta-modèles App et BPM

Dans les schémas des exemples qui suivent : un trait est un lien d'association, unidirectionnel si terminé d'une flèche noire ; un trait terminé d'une flèche blanche symbolise un lien d'héritage.

4.2.1 Fusion de méta-modèle

La première technique de composition est la fusion de méta-modèle. La fusion consiste à créer un méta-modèle global qui inclut tous les concepts des modèles à composer, ainsi que les nouveaux liens qui associent leurs concepts.

EXEMPLE 4.1 – Fusion des méta-modèles App et BPM

Pour réaliser la fusion de App et BPM, il faut créer un nouveau méta-modèle que nous nommons AppBPM en figure 4.12. AppBPM contient les concepts des deux méta-modèles, dont service et activité (*Activity*), ainsi que la nouvelle relation `implémente` entre ces deux concepts.

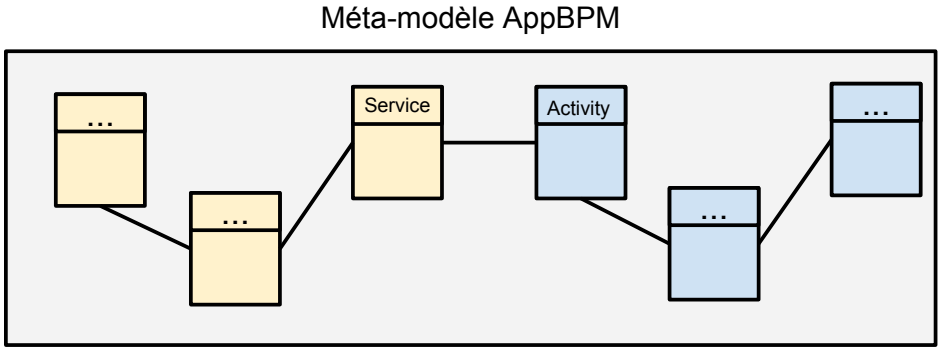


FIGURE 4.12 – Fusion de App et BPM en un unique méta-modèle

Avantages / Inconvénients

+ Le nouveau méta-modèle fonctionne sans aucune spécificité technique supplémentaire.
− Nécessite la création de "super méta-modèles" pour chaque nouveau cas rencontré, ce qui viole l'approche de création d'un langage dédié[5].
− La maintenance est complexifiée, les deux méta-modèles n'ayant plus un cycle de vie indépendant.
− Au niveau instance, il est nécessaire de fusionner aussi les modèles existants avec un outillage adapté, par exemple : *Eclipse EMF Compare*.
− Risque de conflit entre des concepts de même nom, avec ou non les mêmes attributs. Ce conflit est à régler au cas par cas, ce qui nécessite une connaissance approfondie, y compris des éléments qui ne sont pas à aligner.

La technique de fusion n'est pas adaptée pour mettre en œuvre l'alignement, elle ne respecte pas le critère de non-intrusion.

4.2.2 Extension de méta-modèle

La seconde technique de composition consiste à étendre chaque méta-modèle à composer pour inclure les liens des concepts à aligner. En effet, la modélisation permet de

[5]DSL : *Domain Specific Langage* en anglais

réaliser une référence vers le concept d'un autre méta-modèle, notamment par un lien d'association.

EXEMPLE 4.2 – Extension des méta-modèles App et BPM

Les deux méta-modèles : App et BPM sont modifiés. Un nouveau lien d'association est créé dans le méta-modèle App sur le service pour référencer l'activité (*Activity*) du méta-modèle BPM. De même, un lien opposé est créé dans le méta-modèle BPM pour associer l'activité au service.

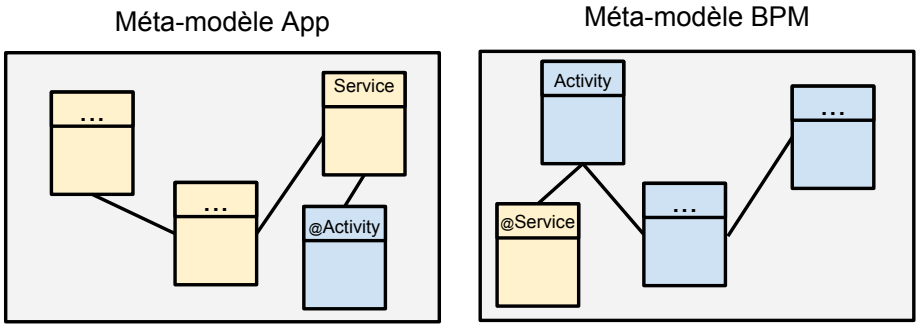


FIGURE 4.13 – Extension de App et BPM

Avantages / Inconvénients

+ Le nouveau méta-modèle fonctionne sans aucune spécificité technique supplémentaire.
− Chaque cas d'étude n'utilisera pas les mêmes méta-modèles, ainsi il faut réaliser la modification pour chaque cas.
− Si un méta-modèle normé est utilisé, sa modification le rend non conforme : par exemple, ajouter des associations aux concepts de BPMN 2.0.

La technique d'extension n'est pas adaptée pour mettre en œuvre l'alignement, elle ne respecte pas le critère de non-intrusion.

4.2.3 Annotations

La troisième technique de composition consiste à étendre chaque concept, que l'on souhaite aligner à des concepts d'autres méta-modèles, grâce à un méta-modèle d'annotation. Les relations entre concepts de modèles différents sont référencées par les annotations. Une fois le concept du méta-modèle aligné au concept d'annotation, il est possible de créer des annotations sur l'instance au niveau modèle. L'annotation permet de créer différentes étiquettes pour référencer n'importe quelle instance d'un autre modèle.

Nous avons expérimenté cette technique grâce au méta-modèle d'annotation appelé *MAnnotation* en figure 4.14. Ce méta-modèle a été développé par Mia-Software[6], il a été inspiré par le mécanisme *Eannotation* du méta-modèle Ecore de Eclipse EMF. Le

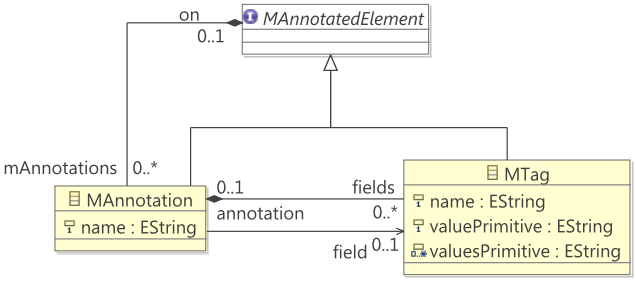


FIGURE 4.14 – Le méta-modèle Mannotations

[6] http://www.mia-software.com/

méta-modèle *MAnnotation* offre l'interface *MAnnotedElement* extensible par un concept de n'importe quel méta-modèle. Au niveau instance, un concept qui étend *MAnnotedElement* peut créer une ou plusieurs annotations *MAnnotation*, et chaque annotation peut avoir plusieurs étiquettes *MTag*. L'étiquette possède des attributs et des liens dont le lien d'association *references* pour associer une instance d'un autre modèle.

EXEMPLE 4.3 – Annotation des méta-modèles App et BPM

Les concepts à aligner des méta-modèles BPM et App sont étendus par le concept *Annoted Element* du méta-modèle d'annotation. Par exemple, le service du méta-modèle App et l'activité (*Activity*) étendent *MAnnotedElement*. Au niveau modèle, une annotation *Annot 1* est créée sur le service *Service 1* du modèle App. Sur cette annotation est créée une étiquette *Tag 1* qui référence l'activité *Activity A* du modèle BPM. De même, une annotation *Annot A* est créée sur l'activité *Activity A* du modèle BPM, l'étiquette *Tag A* référence le service *Service 1* du modèle App.

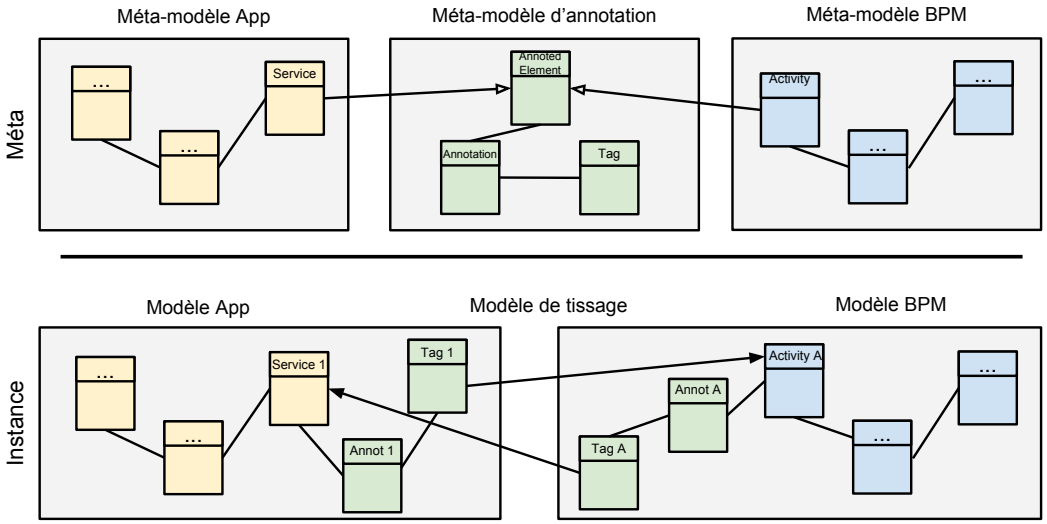


FIGURE 4.15 – Annotation entre App et BPM

Avantages / Inconvénients

+ Il est possible de naviguer entre les modèles par la résolution de proxy native de EMF.
– Il faut réaliser la modification pour chaque concept à aligner, cette méthode est donc intrusive.
– Si un méta-modèle normé est utilisé, sa modification le rend non conforme : par exemple, modifier des concepts de BPMN 2.0 pour hériter du méta-modèle d'annotation.
– Les liens créés par annotation n'ont aucune contrainte de type : notre définition de l'alignement de la section 4.1 n'est pas respectée.

La technique d'annotation n'est pas adaptée pour mettre en œuvre l'alignement, elle ne respecte pas les critères de non-intrusion et d'ontologie.

4.2.4 Tissage de modèles

La quatrième technique de composition est le tissage. Le tissage permet de créer un modèle indépendant référençant plusieurs modèles issus de méta-modèles différents, et de créer de nouveaux liens entre les éléments sources et cibles des différents modèles [50].

Cette technologie est non intrusive puisque les liens de tissage restent externes aux modèles.

EXEMPLE 4.4 – Tissage des méta-modèles App et BPM

Le tissage s'utilise au niveau modèle. Sur la figure 4.16, nous créons un lien *Link 1* qui associe le service *Service 1* du modèle (App) à l'activité *Activity A* du modèle BPM.

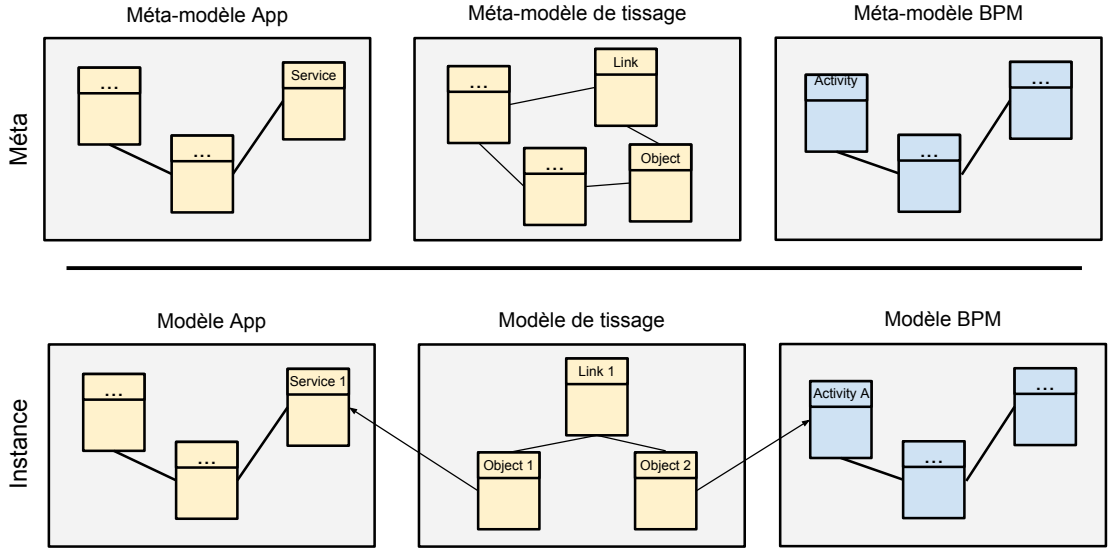


FIGURE 4.16 – Tissage entre App et BPM

Des outils ont été développés pour Eclipse EMF, proposant un méta-modèle de tissage et un éditeur graphique notamment : *Atlas Model Weaver* (AMW) [37] et *Virtual EMF* [16]. AMW inclut un mécanisme de transformation ATL pour réaliser des tissages de façon automatique. Virtual EMF est plus simple, il propose une interface permettant d'éditer deux modèles pour créer des liens via le mécanisme de glisser-déposer. Néanmoins, ces deux éditeurs ne sont plus maintenus depuis plusieurs années et n'ont pas été rendus compatibles avec les nouvelles versions d'Eclipse depuis la version 4.

La technique de tissage étant non intrusive, elle est une candidate potentielle comme technique de composition pour implémenter l'alignement. Nous avons donc expérimenté davantage cette technique. Nous avons créé notre propre méta-modèle illustré par la figure 4.17 pour mettre en correspondance nos différents modèles. Elipse EMF

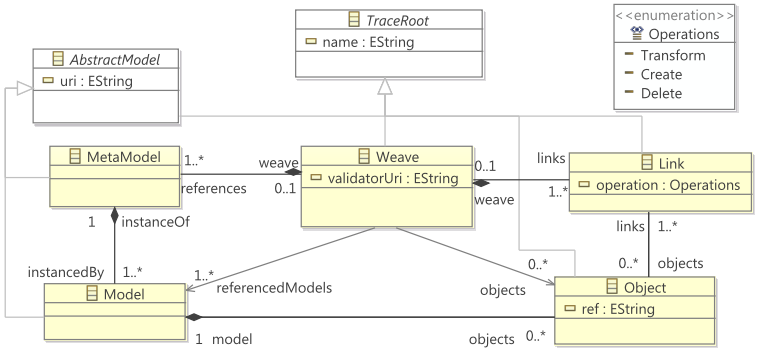


FIGURE 4.17 – Le méta-modèle de tissage modélisé avec

Pour tester cette technique, nous avons créé un éditeur de tissage, illustré en figure 4.18, sous la forme d'un module d'extension (plug-in) pour Eclipse.
L'interface homme-machine (IHM) de ce prototype est composée de deux parties :

À droite les différents modèles que l'on souhaite tisser sont chargés et regroupés par méta-modèles.

À gauche la création de nouveaux liens se réalise à l'aide du menu contextuel. Les différents éléments de modèles sont ajoutés au lien par mécanisme de glisser-déposer de la droite vers la gauche. Le résultat du tissage est visible une fois le lien déplié.

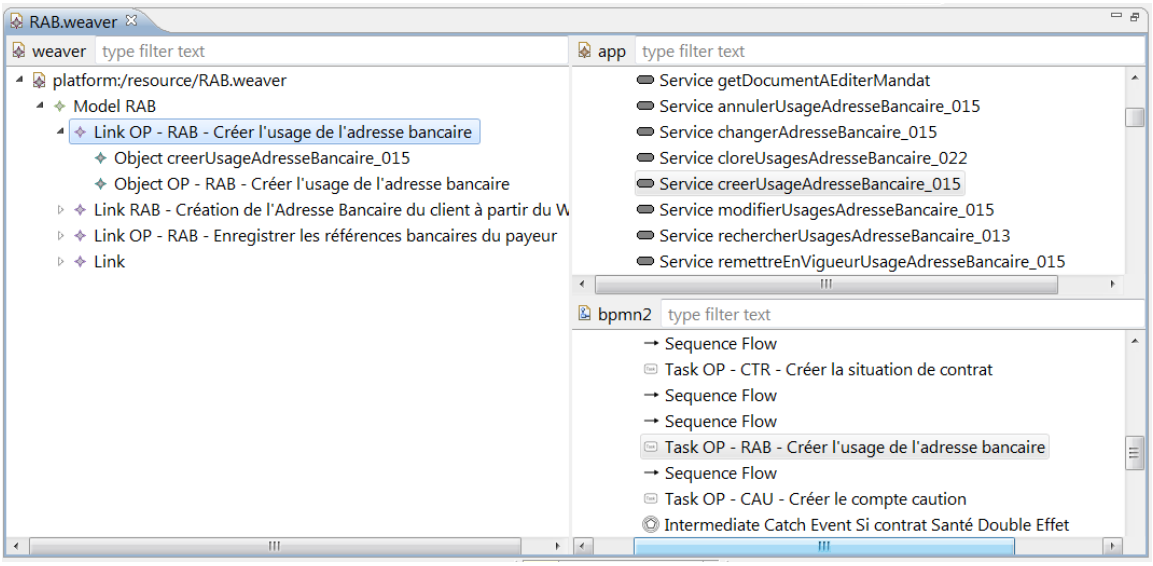


FIGURE 4.18 – Capture d'écran de notre plug-in de tissage

L'éditeur apporte quelques assistances à l'utilisateur :

La recherche permet de filtrer uniquement les éléments correspondant à une saisie alphanumérique. C'est une fonctionnalité essentielle pour l'architecte d'entreprise, car les volumes des modèles peuvent devenir importants.

La surbrillance colore les éléments contenus dans un lien tissé. En cliquant sur un lien à gauche, les éléments des modèles de droite qui composent le lien sont automatiquement mis en surbrillance.

Avantages / Inconvénients

+ Les modèles originaux ne subissent aucune modification.
− Le lien créé entre deux modèles ne se résout qu'avec un éditeur compatible avec le méta-modèle de tissage.
− Il n'est pas possible de naviguer d'un concept d'un modèle vers le concept d'un autre modèle à partir du lien sans parcourir intégralement le modèle de tissage.
− Les liens créés n'ont aucune contrainte de type, ils peuvent recevoir des instances de n'importe quel concept, de n'importe quel modèle : notre définition de l'alignement de la section 4.1 n'est pas respectée.

La technique de tissage n'est pas pleinement adaptée pour mettre en œuvre l'alignement, elle ne respecte pas les critères d'ontologie et de navigation.

4.2.5 Facettes

La cinquième et dernière technique de composition utilise des facettes. Cette technique, également non intrusive, est une évolution de la technique de tissage. La technique de facettes permet d'étendre un méta-modèle en ajoutant virtuellement des attributs, des références et des opérations à un concept préexistant d'un méta-modèle.

EXEMPLE 4.5 – Facettes des méta-modèles App et BPM

Nous créons un nouveau *FacetSet* qui est le support pour mémoriser les nouvelles extensions que nous voulons définir aux méta-modèles App et BPM. Nous ajoutons au FacetSet une propriété virtuelle sur le service du méta-modèle App qui référence l'activité (*Activity*) du méta-modèle BPM. De même, nous ajoutons au FacetSet une propriété virtuelle sur l'activité (*Activity*) du méta-modèle BPM qui référence le service du méta-modèle App. Les deux nouvelles références sont liées entre elles pour que l'association soit bi-directionnelle.

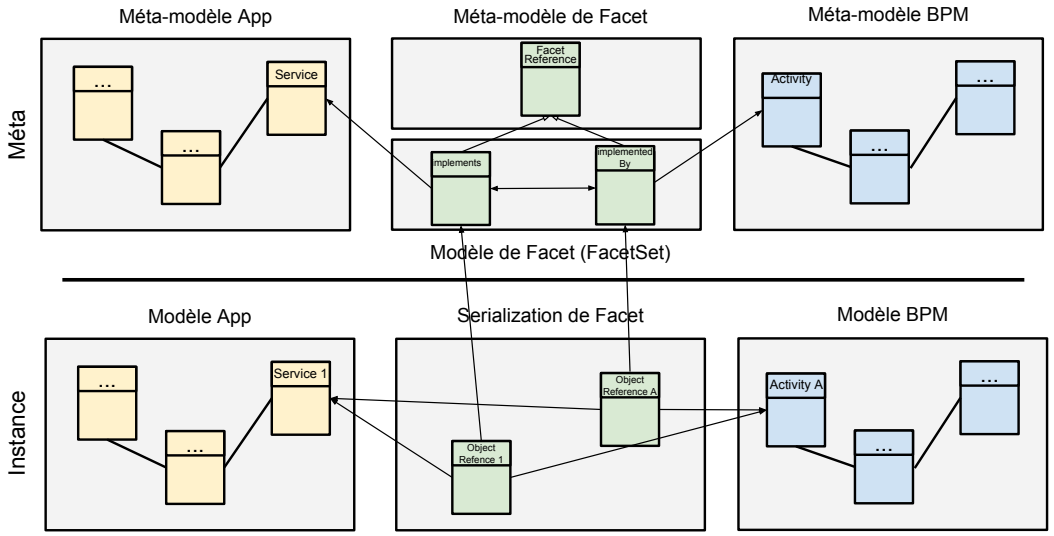


FIGURE 4.19 – Extension de méta-modèle par facettes entre App et BPM

Au niveau instance, nous pouvons désormais créer des liens entre App et BPM comme s'ils étaient natifs dans leurs méta-modèles d'origine.

Cette technique a été expérimentée grâce au projet au code source libre (*open source*) : *Eclipse EMF Facet*.

Pour fonctionner, EMF Facet nécessite l'écriture au préalable de la définition des nouvelles extensions des méta-modèles conformément à son méta-modèle en figure 4.20 : nouveaux attributs, références et opérations ; dans un FacetSet. Le FacetSet est chargé au niveau instance sur un modèle existant afin d'obtenir et manipuler virtuellement les nouvelles extensions. La facette fonctionne grâce à l'implémentation de requêtes qui peuvent être écrites en Java, en OCL, ou en Javascript.

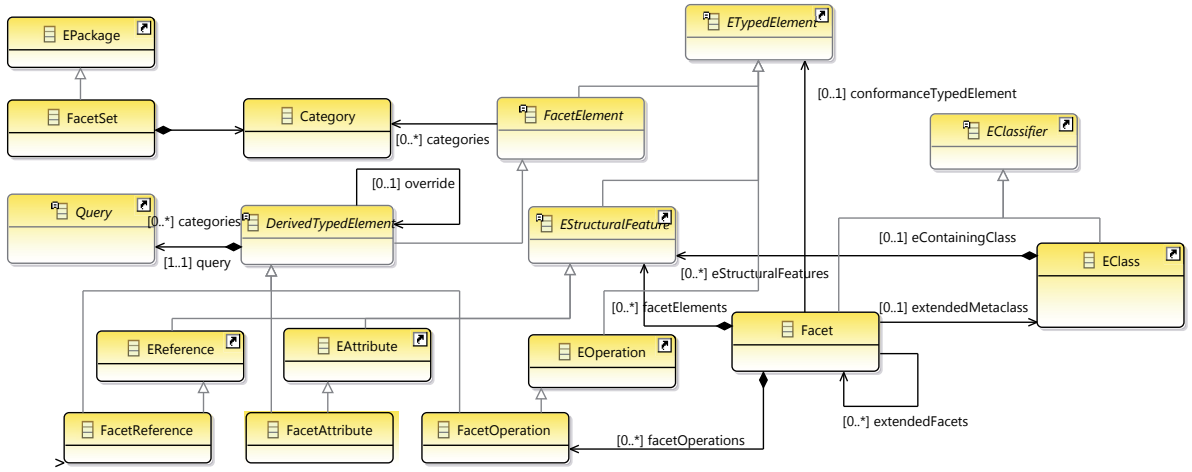


FIGURE 4.20 – Le méta-modèle EMF Facet

Avantages / Inconvénients

+ Les modèles originaux ne subissent aucune modification.
+ Il est possible de naviguer d'un concept d'un modèle vers le concept d'un autre modèle à partir du lien.
+ Les liens sont typés et peuvent respecter la définition exacte de l'alignement en section 4.1.
− Les requêtes s'exécutent durant le chargement du modèle, le temps de calcul impacte le temps de réponse particulièrement si le modèle est volumineux.
− La valeur des extensions n'est évaluée que par des requêtes et ne peut être assignée manuellement comme pour des références ou attributs classiques d'un méta-modèle ordinaire.

La technique de facettes n'est pas pleinement adaptée pour mettre en œuvre l'alignement, car elle ne respecte pas le critère de persistance, mais elle est cependant la technique la plus prometteuse comme nous le verrons dans la section suivante.

4.2.6 Comparaison des différentes techniques de composition de modèles

Nous avons expérimenté toutes les techniques dans le contexte de l'alignement de cartographies d'architecture d'entreprise. Nous résumons notre résultat de comparaison dans le tableau 4.1.

TABLE 4.1 – Comparaison entre les différentes techniques de composition

Technique	Non-intrusivité	Ontologie	Navigation	Persistance
Fusion	✗	✓	✓	✓
Extension	✗	✓	✓	✓
Annotations	✗	✗	✓	✓
Tissage	✓	✗	✗	✓
Facettes	✓	✓	✓	✗

Les trois techniques de fusion, extension et annotations ont été très vite écartées, car elles nécessitent de modifier les méta-modèles pour les mettre en œuvre. Le tissage est une technique que nous avons longuement expérimentée, mais elle ne permet pas d'implémenter notre définition de l'alignement. La technique de facettes reprend les bases du tissage en permettant d'étendre virtuellement un méta-modèle. Elle est la technique qui répond le plus majoritairement à nos critères initiaux : non-intrusive, respecte une définition conceptuelle, navigation entre les concepts de modèles différents. Néanmoins, dans l'état actuel, il n'est pas possible d'assigner une valeur par tissage manuel et de rendre persistant le résultat dans un modèle intermédiaire. Dans la suite, nous allons approfondir le fonctionnement de facettes et proposer une contribution au projet "*open-source*" EMF afin d'améliorer cette technique.

4.3 Une implémentation de l'alignement à l'aide de facettes

Précédemment, nous avons comparé à l'aide de critères plusieurs techniques de composition de modèles. À l'issue de cette comparaison, nous avons sélectionné une technique de tissage évoluée par facettes. Nous avons expérimenté cette technique au travers du projet *open-source EMF Facet*.

Nous allons d'abord faire la présentation des fonctionnalités de EMF Facet dans la section 4.3.1. EMF Facet présente quelques lacunes pour être pleinement utilisable pour aligner des modèles, nous présentons les modifications effectuées en section 4.3.2. À partir des améliorations effectuées sur EMF Facet, nous pouvons désormais implémenter notre définition de l'alignement en section 4.3.3. Enfin, nous présenterons l'utilisation de EMF Facet à l'aide de l'atelier de tissage que nous avons conçu en section 4.3.4 et l'exploitation du résultat de tissage en section 4.3.5.

4.3.1 Présentation du projet Eclipse EMF Facet

Le projet Eclipse EMF Facet est un cadriciel[7] d'extension de méta-modèle qui ajoute des attributs et des références virtuelles. Les concepts originaux du méta-modèle ne sont pas modifiés, c'est lors de l'exécution que les nouvelles propriétés sont disponibles et opérationnelles.

Techniquement, le projet est composé de deux principales parties : *Facet* et *Customization*.

Facet La partie facette permet d'étendre les méta-modèles existants par la définition de nouveaux attributs et références. Les nouveaux attributs permettent d'ajouter de l'information pour créer une nouvelle classification des éléments du modèle. Les nouvelles références permettent d'associer les éléments d'un même modèle ou les éléments de plusieurs modèles différents. Cette dernière fonctionnalité va nous permettre d'implémenter notre définition de l'alignement. L'ajout de nouveaux attributs et références ne modifie pas le méta-modèle d'origine, la définition est "virtuelle" et réalisée dans un modèle tiers appelé *FacetSet*. Il est possible de définir autant de facettes que souhaité, elles peuvent co-exister et être surchargées (une facette étend les attributs d'une autre facette). L'activation/désactivation d'une facette se fait "à la volée" et ne nécessite pas de réouvrir le modèle.

Customization La partie personnalisation permet de modifier le libellé et la représentation graphique d'un concept. Pour réaliser la cartographie du SI, une entreprise peut souhaiter choisir une norme et son méta-modèle existant. Mais les concepts peuvent être synonymes (pas le même nom, mais le même sens) aux concepts couramment employés dans sa culture d'entreprise. Ce facteur de changement peut être un frein à son adoption par les utilisateurs. Pour éviter ce problème, la partie *Customization* de EMF Facet permet de renommer les concepts d'un méta-modèle et également d'en changer l'apparence : icône, couleur du texte, couleur d'arrière-plan, et police de caractères.

[7] *Framework* en anglais

4.3.2 Contribution au projet Eclipse EMF Facet

La composition de modèles à l'aide de facettes est prometteuse (cf. section 4.2.6), mais l'obligation d'une requête pour calculer la valeur d'une facette ne permet pas d'atteindre notre but : l'alignement.

Problème

Nous souhaitons réaliser l'alignement par tissage en associant des concepts de différents modèles. Par exemple, des activités d'un modèle processus métier avec des services d'un modèle applicatif. Ce tissage est manuel à partir des connaissances d'un architecte d'entreprise lors de l'audit du SI. Or les valeurs des nouvelles propriétés (attributs et références) ajoutées à l'aide de EMF Facet ne peuvent être affectées manuellement. Elles sont nécessairement calculées à l'aide de requêtes. Un alignement automatique par requête implique d'être réalisé à partir d'informations déjà existantes dans un modèle tiers. Ce cas est possible, mais rare, nous avons rencontré un cas d'étude en section 5 où l'alignement peut se baser sur le nom des concepts. L'alignement effectué, les valeurs doivent être conservées dans un modèle qui sera enregistré sur le disque dur ou dans une base de données, les valeurs doivent donc être "serialisables".

Solution proposée

Pour résoudre ces problèmes, nous avons modifié le méta-modèle et les codes sources qui implémentent le gestionnaire de EMF Facet (*FacetManager*) et le mécanisme de sérialisation. Afin que des valeurs puissent être affectées aux propriétés, il faut supprimer la contrainte qui rend obligatoire la présence d'une requête sur les propriétés. Les propriétés *FacetReference* et *FacetAttribute* héritent de *DerivedTypedElement* qui possède une association de composition avec la requête *Query*.

La cardinalité est à l'origine 1..1 (figure 4.20) ce qui impose la présence d'une requête *Query* sur toutes les classes du méta-modèle qui étendent *DerivedTypedElement*. Comme illustré par la figure 4.21, nous avons changé la cardinalité vers 0..1, ainsi la requête *Query* devient optionnelle. Ensuite, le code source du moteur de facettes *FacetManager* doit être modifié pour adapter le comportement de l'algorithme. En cas d'absence de requête, l'accesseur et le mutateur[8] sont disponibles pour lire/modifier/écrire la valeur de la propriété.

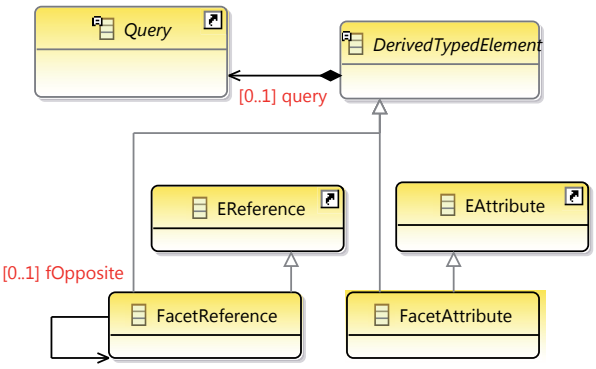


FIGURE 4.21 – Extrait du méta-modèle EMF Facet et les modifications appliquées

Pour permettre de créer des associations bi-directionnelles, nous avons ajouté sur la propriété *FacetReference* la nouvelle référence *fOpposite* (cf. figure 4.21). Ce mécanisme est similaire au méta-modèle natif d'EMF (Ecore) ; avec la référence *eOpposite* sur *EReference*. Enfin, un mécanisme de sérialisation existait déjà au sein de Facet, nous l'avons étendu pour prendre en compte le nouveau comportement des deux propriétés : *FacetReference* et *FacetAttribut*.

[8] Respectivement *getter* et *setter* en anglais

Après ces modifications, EMF Facet permet d'étendre les méta-modèles et de réaliser le tissage de modèles. Nous pouvons mettre à jour le critère *Persistance* du tableau 4.1. Les facettes deviennent l'approche qui répond à toutes nos exigences dans le but de réaliser l'alignement. Nos modifications ont été soumises au projet open source Eclipse EMF Facet, cette soumission a été acceptée et fait maintenant partie de la version en cours disponible.

4.3.3 Implémentation de la définition d'alignement avec Facet

Maintenant que nous avons amélioré le cadriciel EMF Facet, nous avons la technique pour réaliser un tissage entre modèles à partir d'une définition conceptuelle au niveau méta-modèle. Nous implémentons à l'aide de EMF Facet la définition de l'alignement de nos méta-modèles d'architecture d'entreprise (BPM, Fun et App), à partir de la figure 4.6. Nous créons à l'aide des outils de EMF Facet la définition de l'alignement dans un nouveau **FacetSet** en figure 4.22. Le FacetSet contient toutes les **facettes** qui définissent l'alignement : une facette par concept à aligner.

La facette étend les concepts existants de BPM, Fun et App : service, activité (*Activity*), tâche (*Task*), objet de donnée (*DataObject*), bloc (*Block*)... Nous nommons chaque facette à partir du nom du concept à étendre en ajoutant le suffixe "Ext". Puis, pour chaque facette nous créons une nouvelle propriété de type **référence** (*FacetReference*). Cette référence a comme rôle de mémoriser le lien d'association entre le **concept source** et le **concept cible** à aligner. La référence est définie par un nom, une cardinalité, un type (le concept cible) et la référence opposée s'il s'agit d'un lien bi-directionnel.

Chaque référence (*FacetReference*) implémente le lien d'alignement, nous choisissons de les nommer par le nom du type du concept comme sur la figure 4.23, plutôt que par le nom de l'association défini en figure 4.6. Ce choix est guidé par des facilités de suivi de la navigation. Sur le listing 4.1 écrit en OCL, il est plus simple d'atteindre les composants applicatifs en exécutant la deuxième requête (lignes 4-5) plutôt que la première requête (lignes 1-2), car le nom du type recherché est explicite.

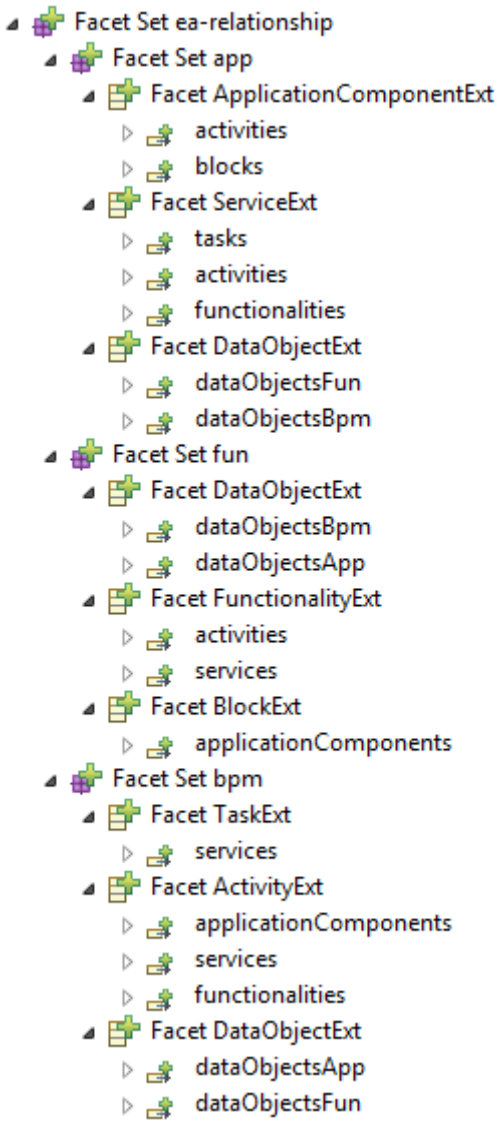


FIGURE 4.22 – Capture d'écran de la définition du FacetSet

Listing 4.1 – Exemple de navigation OCL

```
1 context BusinessProcessLayer:
2 bpm::Activity.allInstances().implementedBy
3
4 context BusinessProcessLayer:
5 bpm::Activity.allInstances().applicationComponents
```

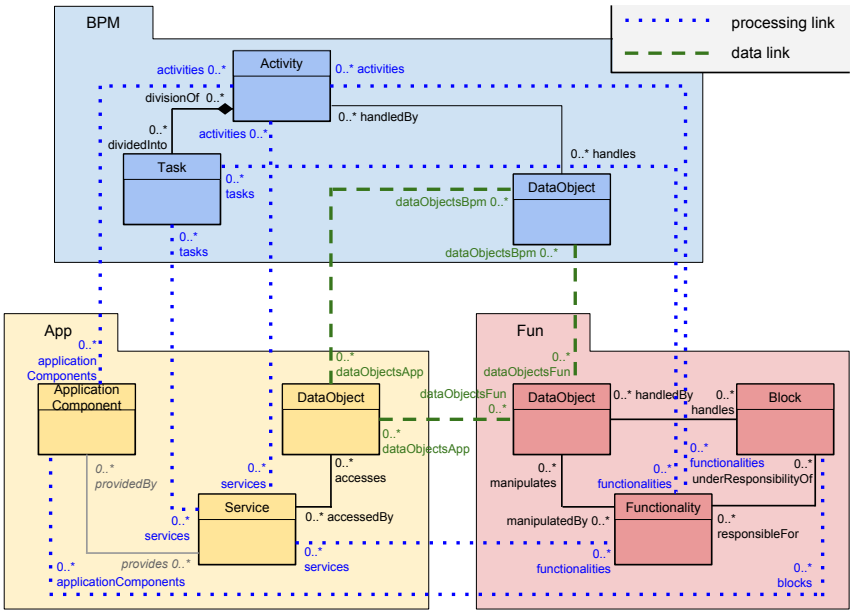


FIGURE 4.23 – Définition de l'alignement avec les liens nommés selon les noms des concepts cibles

Nous reprenons notre exemple fil rouge de la section précédente (figure 4.11) pour effectuer l'alignement entre l'activité *Activity* du méta-modèle BPM et le service du méta-modèle App.

Le premier exemple est la création d'une facette pour étendre le concept activité (*Activity*) du méta-modèle BPM. À partir de cette facette, nous ajoutons une référence pour tisser le lien entre l'activité (*Activity*) du méta-modèle BPM et le service du méta-modèle App.

EXEMPLE 4.6 – Création d'une facette pour aligner l'activité au service

Au FacetSet, nous ajoutons une nouvelle facette *ActivityExt* pour étendre le concept activité *Activity* du méta-modèle BPM. La vue propriété permet d'affecter les différents attributs de la *Facet* : le nom et la méta-classe. Sur la facette activité *ActivityExt* précédemment créée, nous ajoutons une référence *FacetReference* pour implémenter le lien d'alignement entre l'activité *Activity* du méta-modèle BPM vers le service du méta-modèle App. Nous nommons la référence *services* avec comme cardinalité 0..*, ce qui signifie que la relation est n-aire : l'activité est alignée avec aucun, un ou plusieurs services.

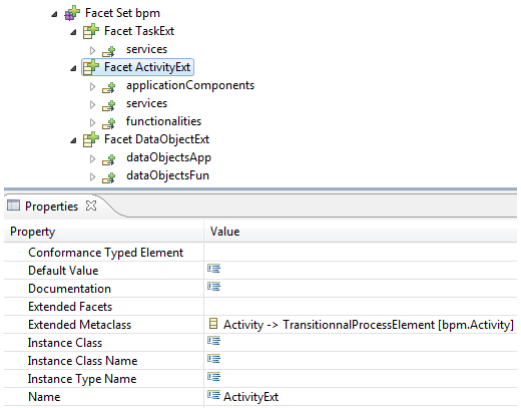


FIGURE 4.24 – Capture d'écran de la Facet ActivityExt

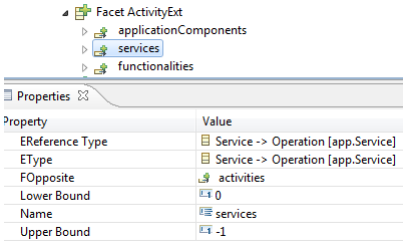


FIGURE 4.25 – Capture d'écran de la FacetReference services

Nous réitérons la procédure, le second exemple est la création d'une facette pour étendre le concept service du méta-modèle App. Sur cette nouvelle facette, nous ajoutons une référence pour tisser le lien entre le service du méta-modèle App et l'activité (*Activity*) du méta-modèle BPM.

EXEMPLE 4.7 – Création d'une facette pour aligner le service à l'activité

Au FacetSet, nous ajoutons une nouvelle facette *ServiceExt* pour étendre le concept service du méta-modèle App. La vue propriété permet d'affecter les différents attributs de la *Facet* : le nom et la méta-classe.

Sur la facette service *ServiceExt* précédemment créée, nous ajoutons une référence *FacetReference* pour implémenter le lien d'alignement entre le service du méta-modèle App vers l'activité (*Activity*) du méta-modèle BPM. Nous nommons la référence *activities* avec comme cardinalité 0..*, ce qui signifie que la relation est n-aire : le service est aligné avec aucune, une ou plusieurs activités.

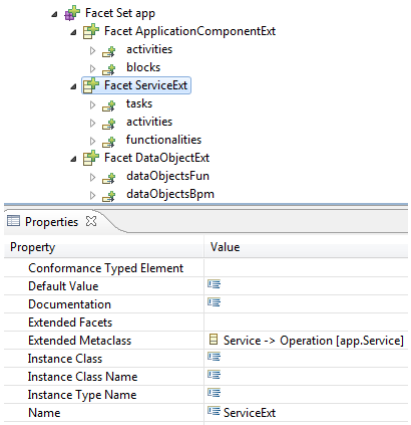


FIGURE 4.26 – Capture d'écran de la Facet ServiceExt

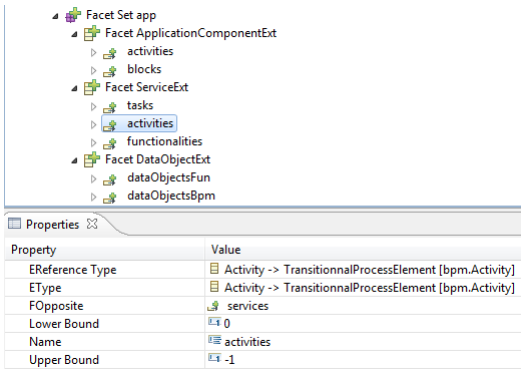


FIGURE 4.27 – Capture d'écran de la FacetReference activities

4.3.4 Atelier de tissage

Pour réaliser l'alignement, nous avons désormais le cadriciel EMF Facet pour réaliser le tissage des modèles, et l'implémentation de notre définition de l'alignement dans un FacetSet. Il est maintenant nécessaire d'avoir un outil de travail destiné à l'utilisateur final : architecte ou analyste métier. L'outil doit posséder les fonctionnalités suivantes pour permettre de réaliser l'alignement :

1. Chargement des différents modèles d'architecture d'entreprise (processus, fonctionnel et applicatif).

2. Réalisation du tissage entre les instances de concepts (activités, tâches, services, composants applicatifs, objets de données, bloc...).

Nous avons acquis une première expérience lors du développement du premier prototype de tissage en section 4.2.4. À partir de ces travaux, nous avons repensé à un nouvel éditeur adapté à la philosophie de facettes.

Comme l'ancien éditeur, la nouvelle interface utilisateur illustrée par la figure 4.28 se présente en deux parties :

À droite les différents modèles que l'on souhaite tisser sont chargés et regroupés par méta-modèle. La différence avec l'ancien éditeur montre que ce panneau est maintenant indépendant et s'affiche désormais dans la vue Outline de Eclipse. L'ergonomie s'améliore : la vue peut être déplacée ou fermée selon les usages. La fonction de recherche textuelle est toujours présente pour filtrer les concepts par leur nom.

À gauche une vue repensée utilisant le classement par granularité de la figure 4.29

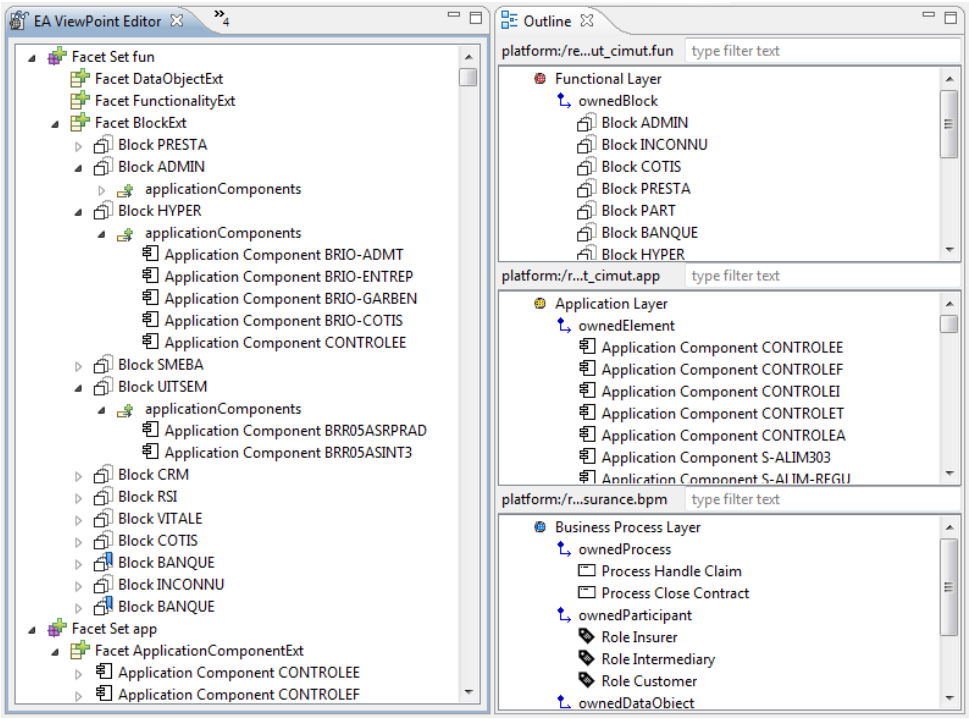


FIGURE 4.28 – Capture d'écran de l'éditeur de tissage par Facet

La nouvelle arborescence présente un rangement par la définition des liens d'alignement ce qui permet une navigation aisée et l'affichage direct des éléments cibles alignés avec l'élément source parent. Le classement est basé sur une nouvelle granularité présentée en figure 4.29.

(1) FacetSet
(2) └ Facet
(3) └ instance source
(4) └ FacetReference
(5) └ instance cible

FIGURE 4.29 – Arborescence d'affichage des concepts dans le nouvel éditeur

L'arborescence présente en niveau (1) la FacetSet par méta-modèle, puis en niveau (2) toutes les facettes que nous avons définies. Nos facettes finissent par le suffixe "Ext". L'arborescence affiche donc tous les concepts alignables de BPM (tâche, activité et objet de donnée), de Fun (bloc, fonctionnalité et objet de donnée), et de App (composants applicatifs, service et objet de donnée). Pour chacun de ces concepts les éléments fils en niveau (3) sont les instances de concepts source que nous souhaitons aligner. Puis, pour chacune de ces instances, les références définies par notre alignement sont disponibles en niveau (4), par exemple la référence *applicationComponents* pour les blocs. Enfin, si la référence possède des éléments fils, ils sont le résultat de l'alignement en niveau (5) : les instances de concepts cibles.

Tissage par glisser-déposer

L'alignement se fait par le mécanisme de glisser-déposer. Les éléments cibles provenant des modèles chargés dans la vue "Outline" de droite sont ajoutés à une référence d'un élément source dans l'arborescence à gauche sur l'éditeur en figure 4.28.

L'exemple suivant est le résultat d'un tissage par glisser-déposer entre un bloc du modèle fonctionnel et des composants applicatifs du modèle applicatif.

EXEMPLE 4.8 – Résultat d'alignement entre un bloc et des composants applicatifs

Nous réalisons un agrandissement sur la capture d'écran de la figure 4.28. Le bloc (*Block*) HYPER du modèle fonctionnel est aligné avec 5 composants applicatifs (*ApplicationComponenent*) du modèle Applicatif : BRIO-ADMT, BRIO-ENTREP, BRIO-GARBEN, BRIO-COTIS et CONTROLEE par la référence *applicationComponents*.

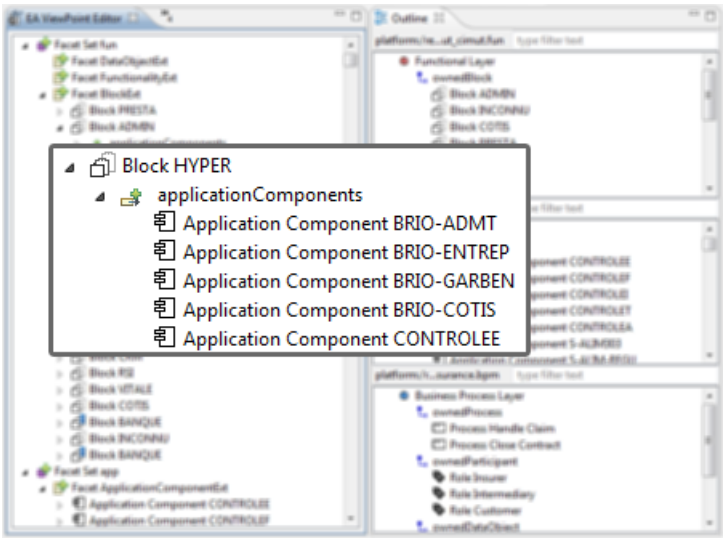


FIGURE 4.30 – Exemple d'alignement entre un Block et des ApplicationComponent

Néanmoins, l'ajout par glisser-déposer n'est pas la façon la plus ergonomique et productive, car elle nécessite beaucoup d'allers-retours avec le pointeur de la souris. Nous proposons une variante pour réaliser le tissage à l'aide de la vue propriété.

Tissage par l'édition des propriétés

Il est commun dans Eclipse d'éditer les propriétés d'un élément en **sélection** à l'aide de la vue Propriété. La vue Propriété de EMF permet par défaut d'éditer les différentes propriétés d'un élément de modèle EMF. Mais la vue de EMF est uniquement compatible avec ses propriétés natives uniquement depuis la définition d'un méta-modèle Ecore : les propriétés virtuelles de Facet ne s'affichent donc pas. Nous avons développé une nouvelle vue Propriété qui étend celle de EMF pour ajouter l'affichage et l'édition des propriétés de Facet.

L'exemple suivant est le résultat d'un tissage par l'édition des propriétés entre un bloc du modèle fonctionnel et des composants applicatifs du modèle applicatif.

EXEMPLE 4.9 – Édition des propriétés pour aligner des instances

Nous affichons les propriétés du bloc (*Block*) “HYPER” en le sélectionnant. Puis, nous éditons la propriété de la référence “*applicationComponents*” qui permet d’aligner des composants applicatifs au bloc HYPER. Une fenêtre modale s’affiche permettant d’ajouter ou supprimer les composants applicatifs.

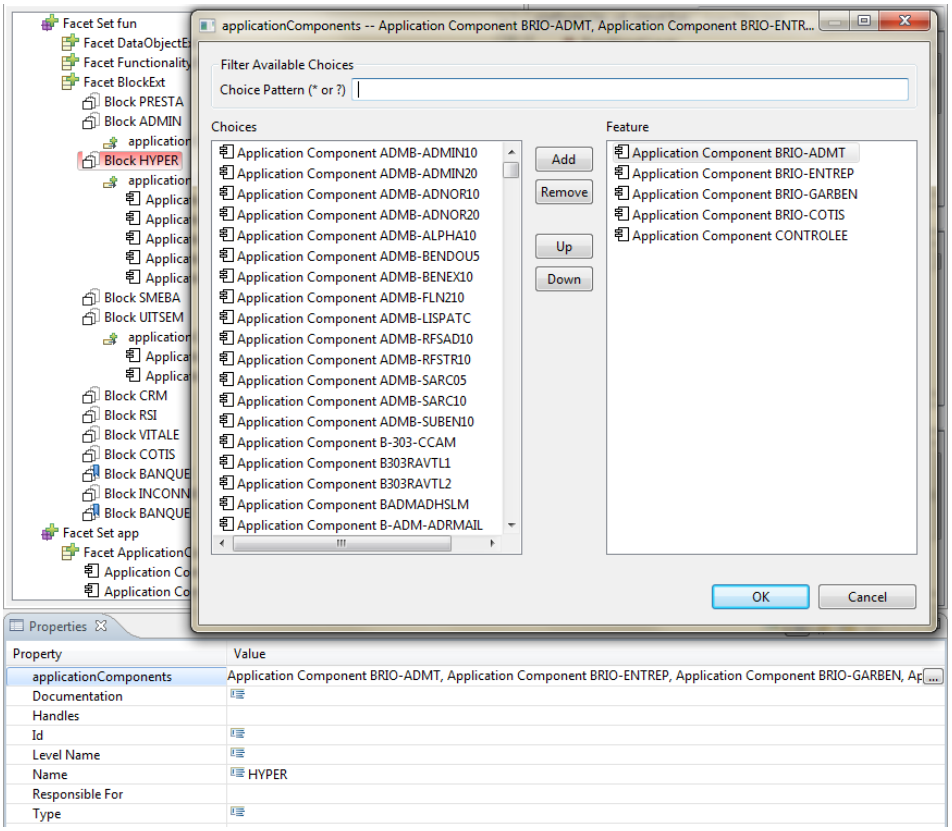


FIGURE 4.31 – Sélection des composants applicatifs à aligner

Cette amélioration est également une contribution acceptée dans le projet *open source EMF Facet*. Il est possible d’utiliser la nouvelle vue Propriété dans d’autres composants utilisant EMF Facet, comme le *Tree Resource Editor* de Eclipse Modisco en figure 4.32.

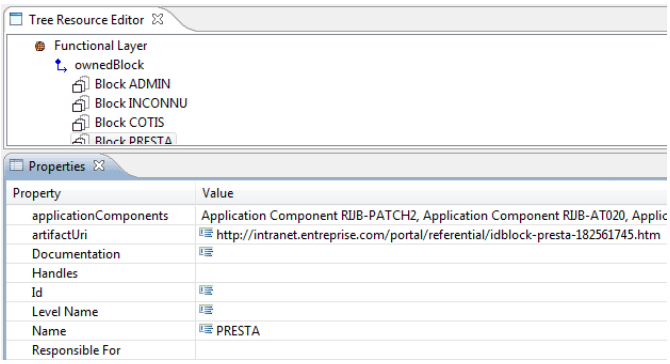


FIGURE 4.32 – La vue Propriété pour éditer les attributs natifs et de Facet

Ce mode d’édition est très efficace pour l’alignement : il évite la manipulation de la souris, il est possible de faire des recherches multiples sur le nom des instances, et on aperçoit le résultat avant de valider sur la partie de droite de la figure.

Notre outillage nous permet maintenant de réaliser l’alignement de modèles d’architecture d’entreprise de façon ergonomique à l’aide d’un éditeur de notre propre conception et d’enregistrer le résultat dans un modèle. Dans la section suivante, nous présentons d’autres outils pour exploiter le résultat de l’alignement.

4.3.5 Exploitation de l'alignement à l'aide de requêtes

À partir d'un résultat d'alignement, nous souhaitons accomplir deux enjeux de l'architecture d'entreprise présentés en introduction (section 1) :

Navigation : l'architecte d'entreprise souhaite naviguer de bas en haut et de haut en bas[9] à travers les différents modèles d'architecture d'entreprise. La navigation est importante pour explorer les éléments qui composent le SI et ses dépendances. À partir de l'alignement réalisé avec l'aide de EMF Facet, il est possible de naviguer à travers les modèles de référence en référence. Par exemple l'éditeur de Eclipse Modisco "*Tree Resource Editor*" permet la navigation de modèles par arborescence. Ce dernier est compatible avec EMF Facet et permet ainsi de naviguer au travers des références "FacetReference". Néanmoins, la navigation par interface utilisateur n'est pas toujours aisée, nous souhaitons également naviguer par recherche à partir de requêtes constituées du chemin d'accès à partir des noms des références de concepts.

Analyse statistique : l'architecte d'entreprise souhaite réaliser des statistiques sur le résultat de l'alignement. Le chapitre 6 décrit les types d'analyses souhaités et possibles. Les actuelles interfaces utilisateur de visualisation permettent uniquement un filtrage par le nom des concepts. Il serait plus fonctionnel de pouvoir filtrer par de multiples critères et conditions, et de pouvoir agréger les résultats à l'aide d'opérateurs ensemblistes classiques (union, intersection, complémentaire, différence, produit cartésien, etc.) pour obtenir des résultats de mesures.

Nous avons étudié les différentes solutions dans la sphère Eclipse EMF pour analyser les modèles. Une API est proposée pour exécuter des requêtes écrites avec le langage OCL au travers du projet Eclipse OCL.

Note

OCL est un langage formel simple d'accès créé par IBM initialement pour l'expression de contraintes sur les modèles UML. Il est désormais un standard de l'OMG depuis la version 1.4 et non réservé à UML mais toutes les méta-modélisations compatibles MOF. OCL permet de décrire : des invariants dans un modèle, pré et post-conditions pour une opération ; expressions de navigation (attributs dérivés) ; expressions booléennes ; etc.

Le projet Eclipse OCL propose tout un panel d'outils permettant d'analyser grammaticalement[10] un modèle EMF à partir d'une requête OCL. Une console permet d'écrire et d'exécuter des requêtes. Un assistant par auto-complétion permet d'aider l'écriture des requêtes en proposant sur le contexte en cours : les expressions du langage OCL et les propriétés du modèle. L'éditeur permet ainsi une navigation aisée dans le modèle par le nom des références. Cependant, l'éditeur est uniquement compatible avec les références et attributs EMF natifs ; les propriétés virtuelles ajoutées par Facet sont invisibles.

Nous avons développé une extension au moteur OCL pour rendre compatibles les extensions de Facet et utiliser les propriétés de Facet dans les expressions OCL[11]. Cette nouvelle fonctionnalité a été écrite dans un nouveau plug-in, le code source a été accepté au sein du projet communautaire EMF Facet.

[9] Respectivement *bottom-up* et *top-down* en anglais
[10] *Parser* en anglais
[11] *Statements* en anglais

L'exemple suivant montre la **navigation** par requêtes OCL à partir d'un modèle fonctionnel, pour atteindre par référence les instances alignées à un modèle applicatif.

EXEMPLE 4.10 – Navigation OCL à travers les références

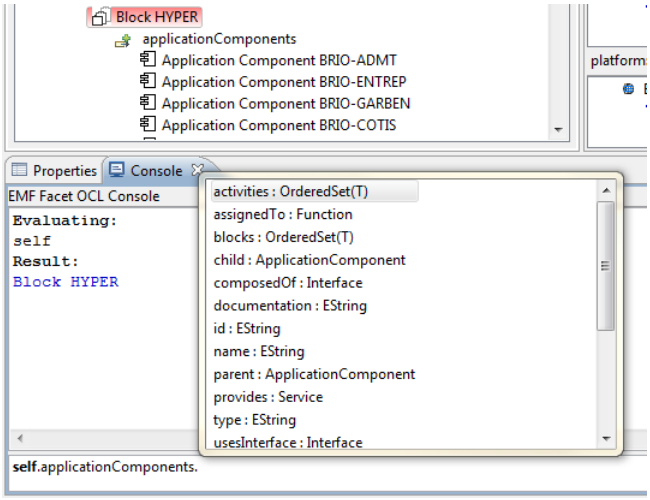


FIGURE 4.33 – Navigation à travers les modèles via les FacetReference

Pour réaliser une requête, il faut sélectionner un nœud dans l'arborescence de l'éditeur afin de définir le contexte OCL. Ensuite, lors de l'écriture de la requête une auto-complétion apparaît pour compléter l'expression en cours de frappe. À partir du bloc (*Block*) "HYPER" du modèle fonctionnel en contexte, nous pouvons descendre dans l'arborescence via la référence *applicationComponents* pour atteindre les composants applicatifs alignés depuis le modèle applicatif. Puis, nous remontons vers les activités (*Activity*) du modèle BPM par la référence *activities*. Cette navigation permet d'obtenir par lien dérivé : les activités alignées au bloc "HYPER".

Le second exemple montre l'**analyse statistique** par requête OCL à partir d'un modèle fonctionnel pour calculer le nombre d'instances alignées à travers une référence.

EXEMPLE 4.11 – Calcul du nombre de dépendances d'un bloc fonctionnel

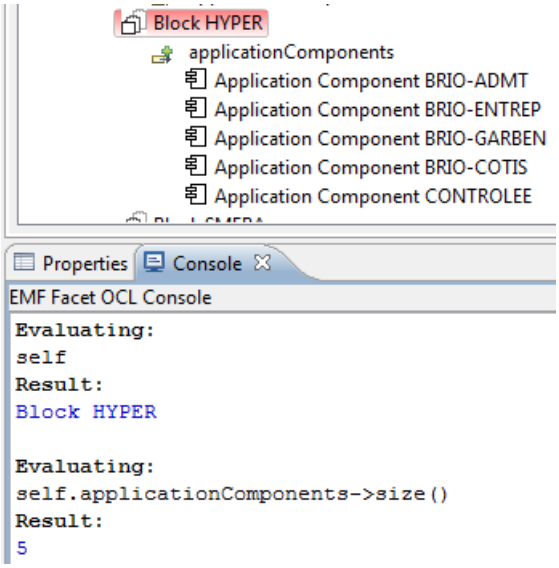


FIGURE 4.34 – Requête qui calcule le nombre ApplicationComponent du Block HYPER

Cette requête triviale permet de calculer le nombre de composants applicatifs *ApplicationComponent* alignés au bloc (*Block*) "HYPER". Le résultat est **5**.

Les possibilités de navigation et de calcul sont très nombreuses, OCL est un langage complet [19] et la console avec l'assistance par auto-complétion permet d'écrire aisément les requêtes.

Conclusion

Nous avons proposé une définition de l'alignement (cf. section 4.1) et avons sélectionné une technique de composition de modèles par facettes (cf. section 4.2.6). Le projet EMF Facet permet d'implémenter notre définition de l'alignement (cf. section 4.3.3), et nous avons développé des outils ergonomiques pour assister l'utilisateur final dans la réalisation de l'alignement de modèles (cf. section 4.3.4). Enfin, nous proposons d'exploiter le résultat de l'alignement à l'aide de construction de requêtes par une console (cf. section 4.3.5). Les travaux qui concernent EMF Facet ont tous été soumis à la communauté du projet open source.

Concrétisation de l'alignement

Les liens entre concepts des modèles processus, fonctionnel et applicatif s'obtiennent à partir de connaissances sur l'organisation de l'entreprise. Ces connaissances sont détenues collaborativement par des acteurs informaticiens (développeurs, architectes) et issus du management (direction métier, chef de projet).

L'information pour réaliser l'alignement peut être obtenue selon deux scénarios possibles :

Automatique Les données qui permettent de réaliser l'alignement entre les points de vue existant dans un référentiel, base de données, tableur... ; il est donc possible de l'exploiter par rétro-ingénierie et plus précisément par une transformation qui constitue le modèle de tissage.

Manuel L'information n'existe pas et doit être reconstituée à partir de la connaissance des utilisateurs, l'assistant permet de créer le tissage avec une approche pratique.

Obtention des modèles en entrée

Nous sommes outillés pour rapprocher, c'est-à-dire aligner concrètement le triplet de modèle : BPM, Fun et App. Il faut désormais peupler ces modèles : les instancier à partir des informations disponibles. Si cette information existe de façon formelle, alors nous pouvons l'utiliser comme des modèles sources. Cependant, ces modèles sont hétérogènes et écrits avec des langages variés. Par transformations successives, nous obtenons des modèles conformes à nos trois méta-modèles qui sont exploitables pour réaliser le tissage.

Dans le chapitre 5, nous présentons un processus automatisé de rétro-ingénierie pour obtenir le modèle applicatif à partir d'un code source.

5

Proposition d'un processus global d'alignement et expérimentations

Sommaire

Introduction

Dans les précédents chapitres nous avons vu que le nœud du problème d'alignement se situe entre les points de vue métier (processus et fonctionnel) et applicatif. Nous avons défini l'alignement à l'aide de **trois méta-modèles** (BPM, fonctionnel et applicatif) pour couvrir les notions relatives aux domaines métier et informatique. Nous avons ensuite proposé une **technique de tissage par facettes** pour aligner les différentes instances de concepts de chaque modèle. Les éléments sont en place, il faut maintenant passer à la mise en œuvre et traiter les problèmes inhérents, notamment la construction de chaque modèle à partir de la collecte des informations correspondant à chaque point de vue.

Problème

Pour réaliser la cartographie du SI, les entrées sont disséminées dans divers supports de stockage de l'information : modèles de données, diagrammes, tableurs, PDF, diaporamas, cahiers des charges, référentiels d'entreprise, etc. L'entreprise doit mettre en place une méthodologie pour classer et conserver ces documents, tel un outil de gestion électronique de documents (GED).

Pour le point de vue applicatif, il est parfois difficile de retrouver les modèles – d'architectures, de classes ou de données – élaborés durant la phase de conception. Le changement de collaborateur dans une équipe projet est fréquent, ainsi les documents et connaissances ne sont pas transmis correctement ou sont perdus. Dans le meilleur cas, celui où la documentation existe, il se pose le problème de sa pertinence : elle peut être obsolète car non maintenue. Un écart a pu se creuser entre les spécifications d'implémentation et ce lors des différentes évolutions du logiciel. Pour résoudre ce problème de documentation obsolète, nous proposons de partir du code source du logiciel : véritable référence de l'état concret du patrimoine applicatif du SI.

Extraire un modèle applicatif à partir du code n'est pas une tâche triviale, le code source est écrit avec un langage de programmation avec des aspects techniques spécifiques ; or seule la description des traitements et des données du programme nous intéresse. Par exemple le code concernant l'interface utilisateur (IHM) ne nous intéresse pas explicitement.

De plus, il existe différentes possibilités indénombrables pour implémenter un logiciel. Prenons quelques exemples de typologie pour illustrer les différents choix qu'un architecte logiciel doit prendre. Nous supposons connue la signification des différents termes techniques.

- **langage de programmation** : impératif, fonctionnel, orienté objet, web...
- **typologie d'architecture logicielle** : monolithique, à composants, client-serveur, orientée services (SOA), distribuée, en couches...
- **patrons de conceptions** : proxy, factory, decorator, observer, singleton, MVC...
- **bibliothèques** : framework, interface (API), plug-in, snippet...
- **typologie de base de données** : fichier, hiérarchique, relationnel, graphe, objet...
- **etc**.

Il semble impossible de proposer une seule technique pour extraire les informations de traitements et de données du logiciel. À chaque situation, des règles spécifiques d'interprétation doivent être élaborées. Ainsi, nous proposons dans ce chapitre un **processus global générique** composé de **sous-processus spécifiques** pour chaque type de support.

Solution proposée

Pour faire face au problème d'obtention des modèles en entrée, nous proposons un **processus global** dans le but de réaliser l'alignement d'un point de vue opérationnel :

- par **abstraction** en remontant du code source écrit par les développeurs vers un modèle applicatif, avec abstraction des détails d'implémentation, exploitable par les architectes d'entreprise ;
- par **concrétisation** en exploitant les modèles métiers réalisés par les architectes et analystes métier durant la phase de cartographie de l'existant durant la démarche d'architecture d'entreprise.

Le processus global définissant les étapes d'abstraction et de concrétisation est détaillé étape par étape dans la section 5.3. Pour illustrer les étapes du processus, nous réalisons des expérimentations en provenance de sociétés de mutuelle d'assurance française. Les cas d'études sont présentés en section 5.2. Chaque étape manipule des modèles ; la section suivante aborde les approches et techniques autour des modèles.

5.1 L'ingénierie des modèles

Pour l'exploitation des modèles, nous faisons appel à l'approche d'**ingénierie dirigée par les modèles** et la pratique de l'**architecture dirigée par les modèles**.

L'ingénierie dirigée par les modèles (**IDM**)[1] est une approche visant à concevoir le développement en uniformisant les représentations par des modèles. Chaque préoccupation est séparée et constitue un modèle [32]. Les instances des concepts et relations du modèle sont définies dans un langage de modélisation (DSML) par la conception du méta-modèle (cf. section 3).

L'architecture dirigée par les modèles (**MDA**) est un ensemble de bonnes pratiques et de standards (UML, MOF, XMI) définis par l'OMG dans les années 2000. Le MDA propose, pour le cycle de vie de développement logiciel, trois niveaux de modèles :

- *Platfom Specific Model* (**PSM**) est le modèle de code correspondant au langage de programmation et l'architecture logicielle ;
- *Platform Independent Model* (**PIM**) est le modèle d'analyse et conception qui s'abstrait de l'architecture logicielle ;
- *Computation Independent Model* (**CIM**) est le modèle métier ou d'exigence indépendant des considérations informatiques.

Le passage de PIM à PSM fait intervenir différents mécanismes de transformation autour des modèles. Les transformations construisent de nouveaux modèles à partir d'autres modèles :

- La transformation modèle vers texte (**M2T**) encore appelé génération est la création d'un fichier texte à partir d'un modèle. Il s'agit généralement de code source, de scripts ou de fichiers de paramétrage.
- La transformation texte vers modèle (**T2M**) encore appelée extraction est l'analyse grammaticale[2] d'un texte pour en déduire le modèle par la construction de son arbre syntaxique. Par exemple, l'analyse d'un code source applicatif pour en déduire son modèle.

[1] *Model Driven Engineering* (MDE) en anglais
[2] *Parser* en anglais

- La transformation modèle vers modèle (**M2M**) est le passage d'un ou plusieurs modèles à d'autres par conversion de concepts de méta-modèle vers d'autres. Il peut s'agir d'une modification du formatage des propriétés, d'un enrichissement par ajout d'informations, ou d'abstraction par omission volontaire d'informations non désirées.

D'autres techniques ont été créées pour manipuler les modèles d'IDM comme la **composition** pour mettre en relation plusieurs modèles, par exemple nous avons abordé la technique de tissage dans la section 4.3.

À l'origine, l'approche **MDA** semblait pertinente et promettait des gains de temps et d'argent pour **générer** une architecture, des patrons de conception, ou un modèle de données. Cependant, l'approche MDA s'avère problématique lors d'une volonté de générer un comportement et des traitements fonctionnels dérivant des spécifications métiers.

Avec le recul, l'IDM requiert des **connaissances et des outils spécifiques**. Sur un projet de développement faisant appel à l'approche MDA, les ingénieurs doivent être formés ; or la connaissance est souvent perdue lorsque l'architecte ayant mis en place la chaîne automatique par l'approche MDA sort du projet. La chaîne de MDA ne pouvant plus être mise à jour est alors progressivement **abandonnée**, et l'investissement d'origine perdu. Un autre inconvénient de l'approche MDA est que les modifications manuelles du code source ne peuvent être remontées au niveau modèle[3]. Ainsi, un **décalage** se crée au fur et à mesure, le modèle métier ne correspond plus à la réalité du programme. Le constat est que l'approche MDA est en perte de vitesse dans les projets de développement du fait de ces difficultés et de l'apparition de cadriciels embarquant les mécanismes techniques qui étaient précédemment générés.

L'approche MDA est habituellement utilisée dans un contexte descendant "*top-down*" ; on part du modèle métier (CIM) vers le code source (PSM) dans une logique de génération de code. Dans le cas présent, notre démarche d'abstraction est montante "*bottom-up*", on part du code source (PSM) pour progressivement s'abstraire de l'architecture logicielle et obtenir le modèle applicatif (PIM) dans le but de réaliser un tissage avec le modèle métier (CIM). Dans la section 5.3.1 nous détaillerons l'étape d'abstraction, elle-même composée de trois sous-étapes correspondant ainsi aux trois modèles définis par le MDA : PSM, PIM et CIM.

Avant d'aborder la présentation du processus global illustré par des expérimentations, nous listons dans la section suivante les trois études de cas support de nos expérimentations.

5.2 La présentation des études de cas d'entreprises

Expérimenter c'est valider notre processus par la pratique avec des **cas réels** issus d'entreprises. Un cas réel signifie une application extraite d'un patrimoine applicatif avec les conditions et scénarios typiques rencontrés en entreprise : code source important, documentation partielle, perte de la traçabilité entre le code et les modèles de spécification, etc.

Nous avons eu l'opportunité de tester notre démarche sur **trois études de cas** concrets, provenant de sociétés d'Assurance Mutuelle françaises que nous nommerons

[3] À l'exception de l'utilisation de l'ingénierie synchronisée, en anglais *Round Trip Engineering* (**RTE**)

SAMM, **SAMI** et **SAMUT**. Chaque étude de cas possède ses spécificités et couvre tout ou partie des modèles du processus. L'expérimentation a pour but de **vérifier la faisabilité** de notre démarche : pertinence des modèles, automatisation des transformations et de leur enchaînement ; ainsi que la pertinence du tissage et de sa mise en œuvre.

Nous présentons succinctement les trois études de cas par une brève description, la nature du code source, le support d'information disponible et les parties prenantes.

SAMM

Description Logiciel de gestion des contrats d'assurance installé sur les postes de chargé de clientèle de chaque agence sur le territoire français.

Code source Un code source complet écrit en Java avec 33 400 classes, environ 3 400 000 lignes de codes. Le poids de l'application est conséquent, c'est un véritable défi pour traiter ce volume d'informations.

Support d'information disponible Un référentiel d'entreprise sous la forme d'un portail HTML exporté depuis le logiciel MEGA Enterprise Architecture[4]. Le référentiel d'entreprise contient 360 diagrammes de processus métier couvrant la totalité du SI.

Parties prenantes Développeur (partie code), analyste métier (partie référentiel).

SAMI

Description Partie du SI de l'entreprise de mutuelle d'assurance qui concerne la gestion des contrats "incendie, accidents et risques divers" (**I.A.R.D.**).

Code source Aucun

Support d'information disponible Le cas SAMI est composé uniquement d'un référentiel d'entreprise MEGA comportant à la fois des points de vue applicatif, fonctionnel et processus dont la source est disponible.

Parties prenantes Analyste métier.

SAMUT

Description Progiciel dédié à la gestion du régime obligatoire et du régime complémentaire.

Code source Les objets de données ont été extraits par rétro-ingénierie d'une ancienne base de données hiérarchique et les composants applicatifs de l'architecture existante à partir de procédures stockées (*SQL Stored procedures*).

Support d'information disponible Audit par un architecte d'entreprise.

Parties prenantes Analyste métier et architecte logiciel.

Ces trois études de cas ont tous des caractéristiques particulières, les supports sources sont totalement hétérogènes et représentatifs de la disparité de maturité des SI. Un cas ne possède aucun code source (SAMI), un cas ne possède aucune représentation métier (SAMUT). Seul (SAMM) possèdent les deux, mais la source du référentiel a été perdue, il reste uniquement l'export HTML.

Dans la suite, nous abordons le processus étape par étape, un ou plusieurs cas présentés illustrent la mise en œuvre expérimentale de l'étape selon le critère *Support d'information disponible*.

[4] http://www.mega.com/fr/solution/business-architecture

5.3 Le processus proposé et ses outils

Le processus global est organisé en **4 étapes** que nous présentons par la figure 5.1.

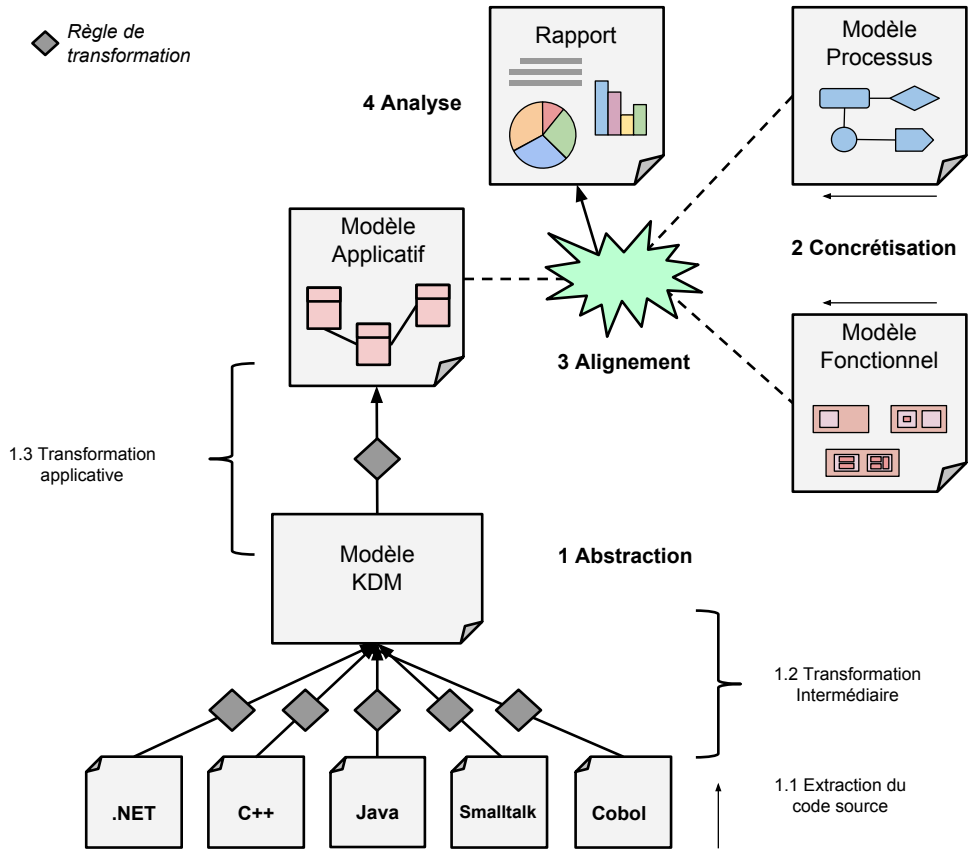


FIGURE 5.1 – Les étapes du processus global

1 **Abstraction** : Obtention du modèle applicatif par rétro-ingénierie du code source ;

2 **Concrétisation** : Exploitation des supports pour construire les modèles métier ;

3 **Alignement** : Tissage à l'aide de facettes ;

4 **Analyse** : Calcul de métriques à partir l'alignement.

La première étape du processus est l'**abstraction**, elle consiste à remonter les informations provenant du code source par rétro-ingénierie, afin d'**obtenir un modèle applicatif** (cf. section 5.3.1). Cette étape est facultative si la cartographie applicative a déjà été réalisée par un autre outil, par exemple un référentiel de SI.

Note

La rétro-ingénierie est un processus d'analyse d'un système :
1. identifier les composants et leurs relations ;
2. créer une représentation du système sous une forme plus condensée ou à un niveau d'abstraction plus. élevé

Reverse engineering and design recovery : A taxonomy - Chikofsky [27]

La deuxième étape du processus est la **concrétisation**, elle consiste à exploiter les différents **modèles métiers** déjà élaborés par les responsables métier de l'entreprise (cf.

section 5.3.2). Il convient de traduire ou de transformer ces modèles selon nos méta-modèles processus (BPM) et fonctionnel (Fun) dans le but de les aligner conformément à la définition de la section 4.1.

La troisième étape du processus est l'**alignement**. Pour réaliser cette étape, nous mettons en œuvre les techniques du chapitre 4. L'alignement exploite les modèles obtenus aux étapes d'abstraction et de concrétisation, nous relions les concepts de ces modèles à mettre en relation par **tissage des modèles**.

La dernière et quatrième étape du processus est l'**analyse** du résultat de l'alignement. L'analyse a pour but de **détecter les incohérences** de l'alignement. Les différentes mesures sont détaillées et mises en place dans le chapitre suivant.

Dans la suite, nous détaillons et illustrons par les études de cas les **trois sous-processus** : abstraction, concrétisation et alignement. Pour chacun, nous présentons les données en entrée et sortie, le but et la technique mise en œuvre.

5.3.1 Étape d'abstraction pour obtenir un modèle applicatif

Pour répondre au problème d'obtention du modèle correspondant au point de vue applicatif, nous proposons le sous-processus d'**abstraction** afin d'obtenir un modèle applicatif à partir du code source. Les trois sous-étapes nécessaires sont rappelées sur la figure 5.2. Ces étapes utilisent toutes des techniques d'ingénierie dirigées par les modèles.

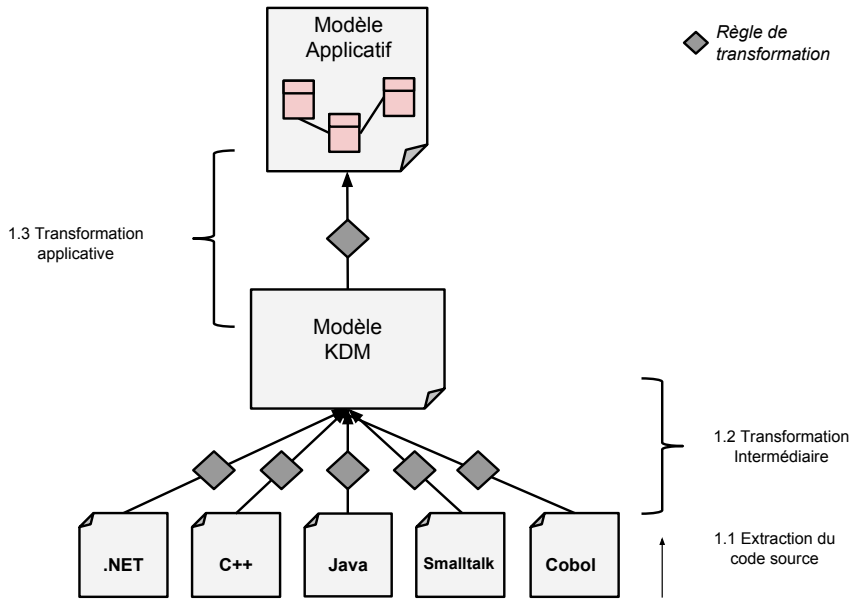


FIGURE 5.2 – Les étapes du sous-processus d'abstraction

1.1 **Extraction du code source** : Découverte du code source afin de réaliser l'analyse du code et de détecter les différentes structures du langage : syntaxe, instruction, type, variable, déclaration, etc. ;

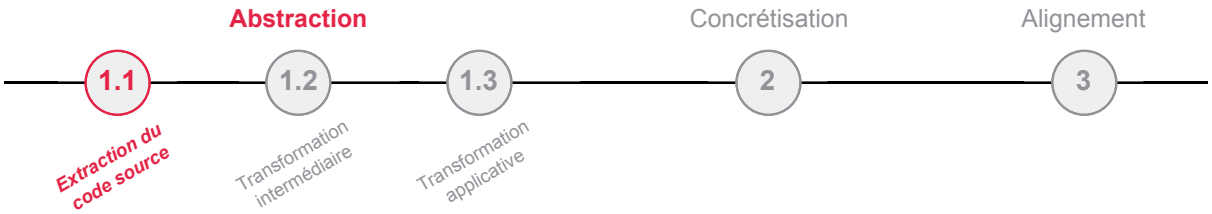
1.2 **Transformation intermédiaire** : Transformation vers un modèle pivot pour s'abstraire des spécificités du langage de programmation ;

1.3 **Transformation applicative** : Transformation vers le modèle applicatif conservant uniquement les concepts conformes à la spécification du méta-modèle *App*.

Il serait possible de réaliser toutes les étapes d'**extraction du code source** et de **transformation** en une seule étape. Or en IDM, il est plus simple de découper une transformation complexe en un processus de transformations élémentaires et on gagne aussi en généricité et réutilisation. Ainsi pour l'étape de transformation texte vers modèle, l'analyseur syntaxique est réutilisable pour tous les programmes utilisant la même version d'un langage de programmation, exemple Java 7. Tandis que pour nos transformations de modèle vers modèle, elles peuvent requérir des spécificités pour chaque patron de conception et architecture logicielle.

Nous allons détailler ces trois sous-étapes en illustrant leur mise en œuvre et les outils utilisés au travers l'expérimentation d'entreprise SAMM, car il s'agit de la seule mettant à disposition un code source.

Extraction du code source



L'étape d'**extraction du code source** est dépendante du langage de programmation et de l'architecture employée. Or les systèmes d'information sont hétérogènes et composés de technologies diverses. Les langages les plus communément rencontrés en entreprise sont Java, C++, .Net, Smalltalk, Cobol, etc. Mais les cadriciels, technologies et langages de programmation évoluent rapidement dans le temps. D'une version de plateforme à l'autre, l'extraction du code source peut nécessiter des adaptations. Le code source est représenté par un fichier texte où sont écrites les instructions du programme. Les instructions suivent une syntaxe définie par une grammaire (ensemble de mots-clefs).

La rétro-ingénierie consiste à analyser les fichiers de code source à l'aide d'un analyseur syntaxique (*parser*). Un analyseur différent est nécessaire pour chaque technologie rencontrée. L'analyseur permet à partir de la grammaire du langage d'obtenir l'arbre syntaxique abstrait (**AST**) dont les nœuds sont les opérateurs du langage et les feuilles les variables ou constantes. L'arbre peut ensuite être enregistré à l'aide du méta-modèle du langage vers un modèle représentant le code source (modèle PSM).

La première expérimentation est l'extraction du code Java du cas d'étude SAMM pour obtenir un modèle Java.

EXPÉRIMENTATION 5.1 – Extraction du code Java de SAMM avec Modisco

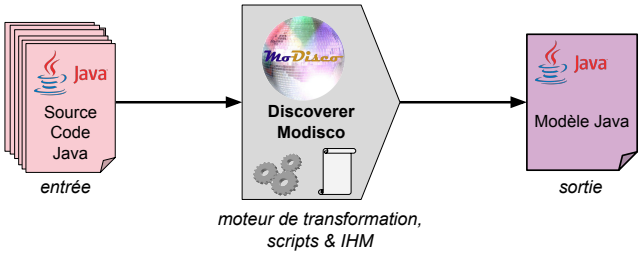


FIGURE 5.3 – Schéma du processus d'extraction de code source

Pour cette étape d'extraction de code source, nous utilisons le projet open-source **Eclipse Mo-**

disco – projet supporté et maintenu par l'entreprise Mia-Software – et notamment l'analyseur Java Discovery intégré à l'outil Modisco. Il s'appuie sur le plug-in *Java Development Toolkit* (**JDT**) qui permet de programmer les applications Java sur Eclipse. Le JDT permet de construire l'arbre syntaxique (**AST**) des éléments du langage Java contenus dans les fichiers de code source. Cette découverte produit le modèle du code Java en sortie à partir du méta-modèle Java apporté par Modisco.

La figure 5.4 illustre l'interface utilisateur du découvreur Java de Modisco. À partir du menu contextuel et de la sélection d'un projet Java dans l'espace de travail (*Workspace*) d'Eclipse, il est possible d'obtenir le modèle Java par transformation.

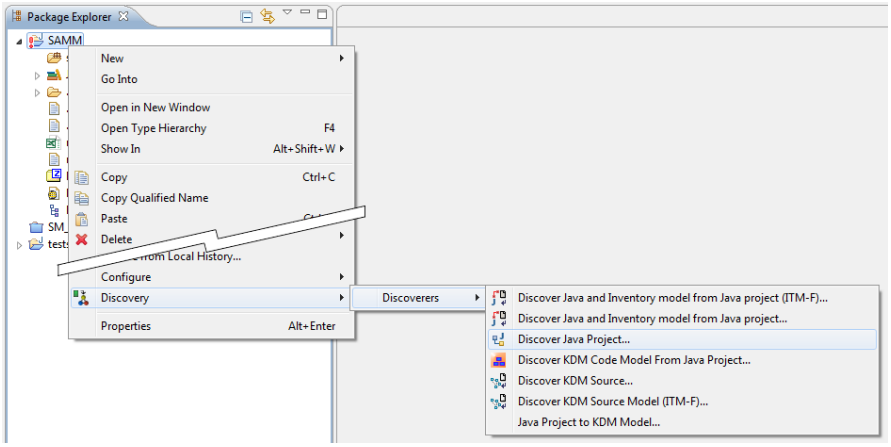


FIGURE 5.4 – Sélection du découvreur Java à l'aide du menu contextuel sur le projet

La figure 5.5 illustre l'interface de sélection des paramètres proposés avant de lancer la transformation qui permet de définir le nom du modèle de sortie, des filtres d'expression régulière (regex) sur les éléments à analyser, etc.

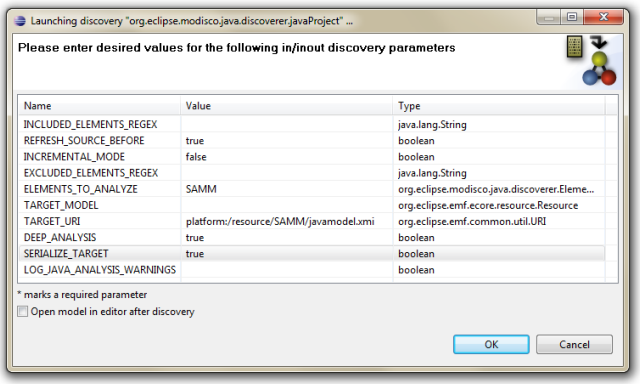


FIGURE 5.5 – Paramètres du découvreur Java avant son exécution

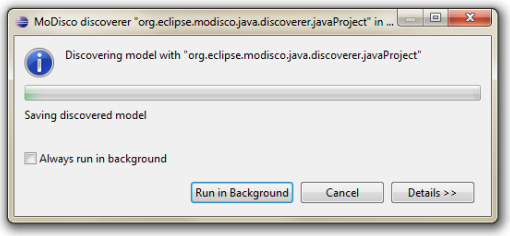


FIGURE 5.6 – Dialogue de progression durant la découverte

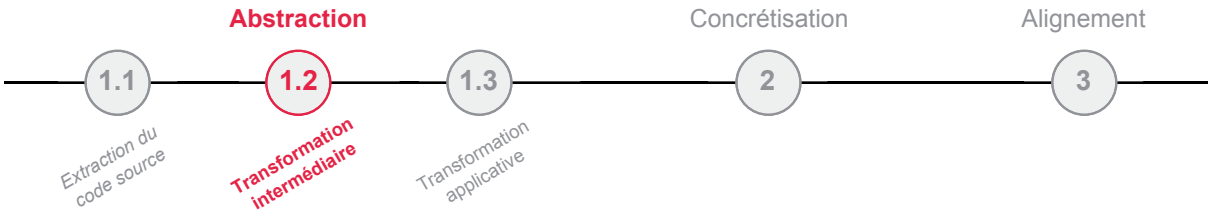
Le temps de traitement (figure 5.6) et l'occupation mémoire dépendent du nombre de projets Java en dépendance, du nombre de fichiers de code source, et du nombre de dépendances entre classes. En effet, un import en en-tête d'une classe code source Java exprime une dépendance. Ainsi, l'analyseur doit découvrir les dépendances avant de découvrir le fichier de programme courant.

Sur le cas d'étude SAMM utilisé pour l'exemple, le code est composé de 33 400 classes Java pour environ 3 400 000 lignes de codes. Malgré la taille du code source, la rétro-ingénierie est rapide : environ 10 minutes sur un ordinateur équipé d'un processeur 4 cœurs et 8Go de mémoire. Le modèle généré en sortie est lui très volumineux, le fichier *XML Metadata Interchange* (XMI) généré en sortie a une taille de 1,4 Go.

L'expérimentation 5.1 a permis de vérifier la faisabilité de l'extraction d'un volumineux code source d'une application vers une représentation en un modèle du langage de programmation de ce code source.

L'étape suivante consiste à transformer le modèle obtenu dont les concepts sont liés à un langage de programmation vers la représentation indépendante KDM.

Transformation intermédiaire



Dans l'étape précédente d'extraction, nous avons obtenu le modèle de code (PSM). La seconde étape de **transformation intermédiaire** de notre processus est la transformation vers le modèle intermédiaire (PIM) pour s'abstraire des détails spécifiques liés au langage de programmation. Le modèle intermédiaire est défini par un méta-modèle capable de recevoir les concepts des différentes technologies issues de la première étape d'extraction. Cette transformation n'abstrait pas tous les aspects d'une architecture logicielle, elle permet par exemple de passer du langage Java ou C++ vers une représentation unifiée d'un programme écrit en orienté objet.

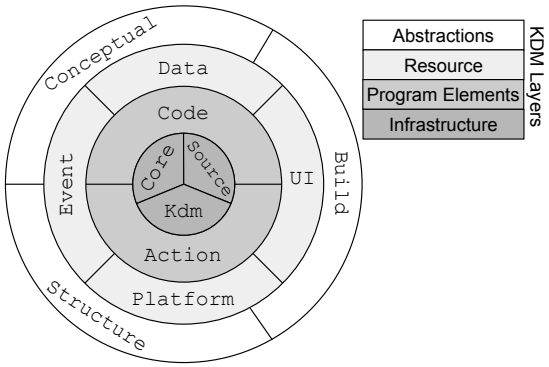


FIGURE 5.7 – Les couches et paquetages du méta-modèle KDM

Nous avons choisi le méta-modèle intermédiaire *Knowledge Discovery Metamodel* (**KDM**), standard spécifié par l'OMG utilisé dans de nombreux outils informatiques de modernisation. Le méta-modèle KDM inclut 12 paquetages (*package*) regroupés en 4 couches pour stocker les différents aspects des langages de programmation communs [79]. Nous utilisons principalement le paquetage `Code` et `Core` qui sont les seuls utiles pour nos préoccupations de modélisation d'une application. Néanmoins, le niveau de description très complet du méta-modèle KDM ouvre des perspectives pour représenter des programmes avec de nouvelles informations (interface utilisateur, base de données, paramétrage de compilation), sans pour autant changer de méta-modèle.

La deuxième expérimentation est la transformation du modèle Java, obtenu durant la première expérimentation du cas d'étude SAMM, pour obtenir le modèle KDM.

EXPÉRIMENTATION 5.2 – Transformation du modèle Java de SAMM vers KDM

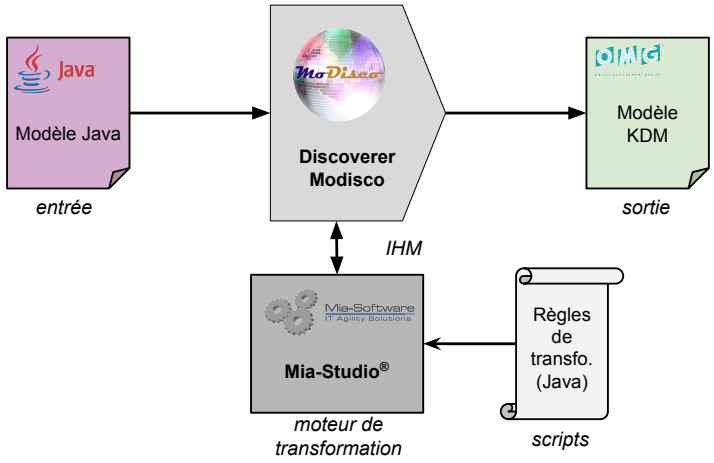


FIGURE 5.8 – Schéma du processus de transformation intermédiaire

Le projet open-source Modisco précédemment utilisé à l'étape d'extraction propose déjà une transformation de modèle Java vers KDM. Nous avons tenté d'utiliser la transformation Java embarquée dans Modisco. Cependant, nous sommes face à un problème de passage à l'échelle. Sur de petites expérimentations correspondant à un projet Java de quelques dizaines de classes Java, la transformation se déroule correctement. Mais à partir d'un modèle Java en entrée dépassant les 100 Mo la transformation ne finit jamais (exécution laissée durant plusieurs jours). Or le modèle Java du cas d'étude SAMM obtenu à la première étape a une taille de 1,4 Go, et ne passe donc pas dans l'outil. La transformation de modèle Java vers un modèle KDM embarqué dans Modisco est écrite avec le langage *Atlas Transformation Language* (ATL). ATL est un langage et un moteur de transformation open source initié par l'équipe Atlanmod [57]. Nous avons tenté de détecter l'erreur et de mettre à jour le moteur d'exécution, mais faute de maîtriser de façon poussée l'outillage ATL, nous avons décidé d'écrire notre propre transformation avec le logiciel *Mia-Transformation*.

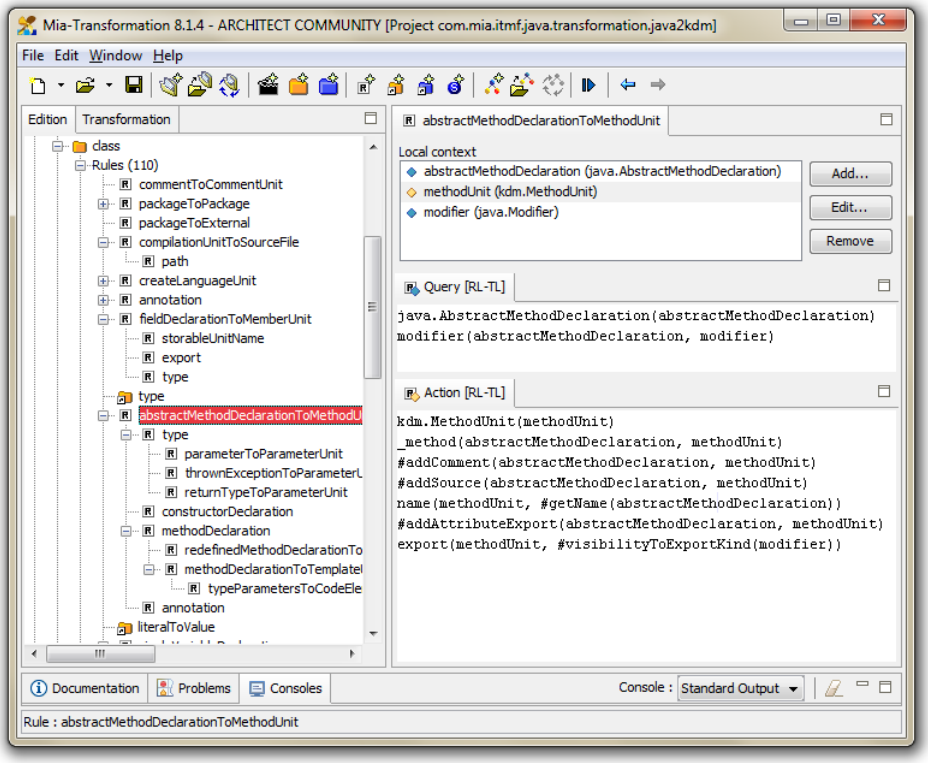


FIGURE 5.9 – Interface développeur de Mia-Transformation

Les transformations de modèle Java vers un modèle KDM ont été réécrites en nous inspirant largement des règles ATL de Modisco. Toutefois, quelques modifications par rapport aux règles ATL originales ont été apportées pour améliorer le résultat de la transformation, notamment en enrichissant les informations des concepts du paquetage du méta-modèle KDM `Source`. La complexité est importante, la transformation composée de plus de 150 règles se déroule en deux étapes :

1. Une première étape traduit tous les concepts (paquetages, classes, interfaces, attributs, méthodes, annotations, déclarations, types, etc.), il est nécessaire de traduire tous les concepts avant de les associer ;
2. Une seconde étape transcrit les associations entre concepts (une classe est composée de méthodes et variables, une classe hérite d'une autre classe et/ou implémente une interface, etc.).

La figure 5.9 est une capture d'écran du logiciel Mia-Transformation. Sur l'interface, dans la partie de droite les éléments avec une icône "losange jaune" sont les entrées :les concepts à transformer du modèle Java ; les éléments avec une icône "losange bleu" sont les sorties : les concepts transformés dans le modèle KDM). L'interface est ouverte avec une sélection sur un exemple de règle de la première étape. La règle « *abstractMethodDeclarationToMethodUnit* » permet de transformer une méthode de classe Java en une méthode du modèle KDM.

L'étape 2 utilise une table de correspondance (*Map*) entre concept Java d'origine et concept KDM nouvellement créé. En parcourant les liens entre concepts Java, la transformation recrée des liens équivalents sur les concepts atteints par la table de correspondance.

Afin de conserver la même ergonomie pour l'utilisateur dans les différentes transformations, nous avons créé un plug-in pour Modisco similaire à la première expérimentation (cf. figure 5.4) pour lancer le moteur de Mia-transformation et les différents scripts de transformation Java vers KDM que nous avons écrits.

Les premières tentatives d'exécution de la nouvelle transformation ont échoué, faute de mémoire vive suffisante. Nous avons dû utiliser une machine pourvue de 10 Go de mémoire vive. Après un temps de traitement acceptable d'environ 2 heures, nous obtenons avec succès le modèle KDM, sa taille est de 780Mo.

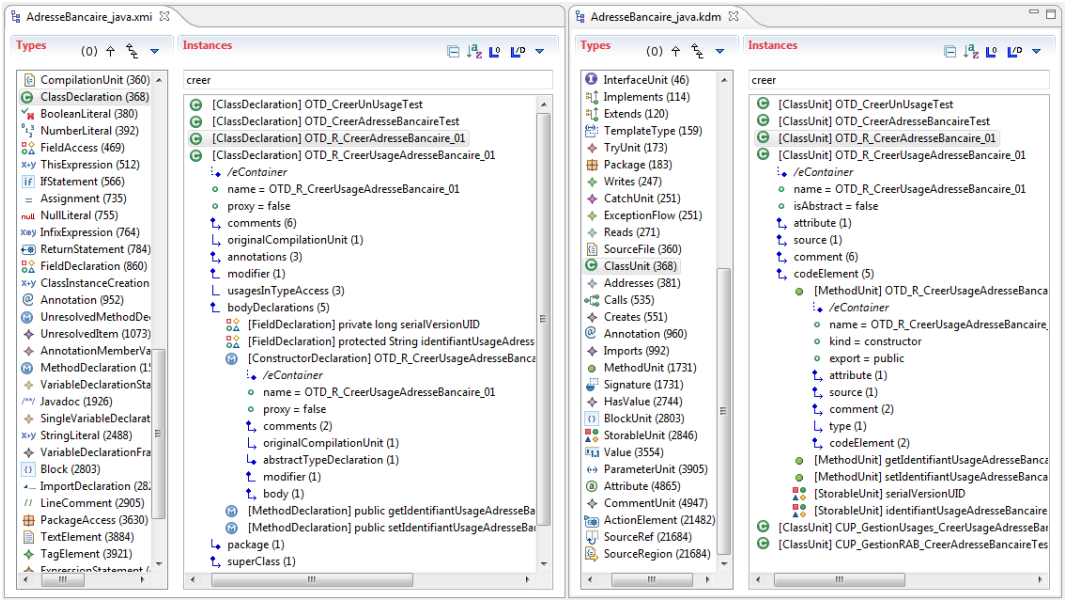


FIGURE 5.10 – Comparaison de la représentation d'une classe en KDM et Java

Comparé au modèle Java, la différence de taille est due au langage qui est moins détaillé dans le paquetage code du méta-modèle KDM que dans le méta-modèle Java, ainsi toutes les spécificités du langage Java ne sont pas traduites en détail. Par exemple sur la figure 5.10 la classe "OTD_R_CreerAdresseBancaire_01" est représentée à gauche par le modèle Java et à

droite dans le modèle KDM. Nous pouvons remarquer que le nombre d'attributs et propriétés est inférieur en KDM, cependant les principaux sont présents : source, commentaire, méthodes et corps des méthodes, visibilité. Ainsi, les concepts essentiels (classes, interfaces, méthodes, routines, variables, etc.) sont présents et sont correctement retranscrits après vérification par comparaison des modèles en entrée et sortie.

L'expérimentation 5.2 a permis de vérifier la faisabilité de la transformation d'un modèle de taille importante représentant le code source d'une application vers le modèle intermédiaire KDM. Cette abstraction a permis de réduire de façon significative le volume d'informations par deux.

Note

Mia-Transformation est un logiciel élaboré à partir de l'expérience de Mia-Software. La filiale Mia-Software a été crée en 2004, il s'agit du département R&D de l'ESN Sodifrance. Mia-Software édite la suite logicielle *Mia-Studio*, destinée au développement d'applicatif à l'aide de la démarche d'ingénierie dirigée par les modèles. Mia-Studio intègre notamment les logiciels Mia-Génération (génération de code) et Mia-Transformation (transformation de modèles).

Le logiciel Mia-Transformation se base sur les standards de l'OMG : MDA, MOF et XMI ; et est également pleinement compatible avec les méta-modèles écrits en UML et EMF. Trois étapes sont nécessaires pour réaliser la transformation :

1. Chargement du modèle source
2. Transformation des concepts du modèle source vers les concepts du modèle cible
3. Sauvegarde du modèle cible

Il est possible de définir un ou plusieurs modèles sources et cibles.

Un projet Mia-Transformation est composé de trois parties :

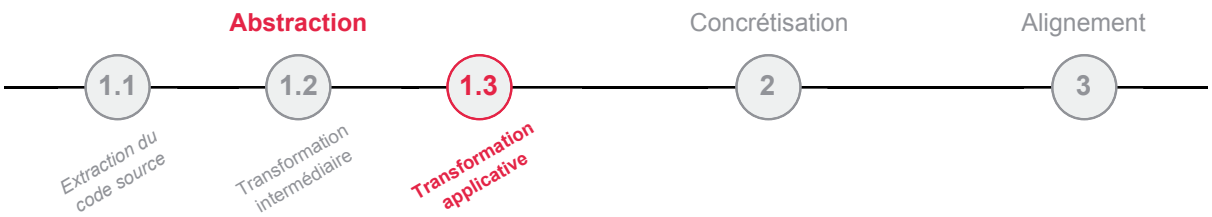
- les scénarios permettent de déclarer les méta-modèles de travail et de définir les actions de transformation : modèle à charger, règle initiale, enregistrement ;
- les règles définissent une requête de lecture sur le modèle d'entrée et une action à effectuer sur le modèle de sortie ;
- les services permettent de créer une action réutilisable à partir de plusieurs règles.

Pour écrire les règles et services de transformation Mia propose trois langages :

- MIA-TL : syntaxe réduite, proche du langage du méta-modèle, utilisé pour des transformations simples sur les attributs ;
- RL-TL : syntaxe très simpliste, proche du langage du méta-modèle, utilisé pour des transformations très simples ;
- Java : permet d'élaborer les algorithmes les plus complexes.

L'étape suivante consiste à s'abstraire des concepts architecturaux de KDM vers notre méta-modèle App.

Transformation applicative



L'étape de transformation applicative procède à l'abstraction la plus forte pour se détacher de la logique de code programme et aller vers une logique applicative [5]. Dans la section 3.3 nous avons défini le méta-modèle App constitué de composants, services, interfaces, fonctions et objets de données. La transformation doit détecter les concepts du méta-modèle App à partir du modèle KDM. La représentation de code source KDM contient des informations de différentes natures : métier, utilitaire et technique. La transformation va tamiser les différents concepts de KDM pour ne capter que les concepts utiles dans l'objectif de constituer le modèle applicatif. L'algorithme de détection est composé de scripts de transformation avec une complexité dépendante de l'architecture utilisée pour l'écriture du code source.

La transformation du modèle KDM vers le modèle App que l'on propose est une transformation spécifique à chaque cas d'étude et à partir de règles conformes à la connaissance de l'application : règles d'architecture ou convention de nommage. Nous ne prétendons pas pouvoir résoudre le problème de détection automatique et générique des éléments d'un code source : composants, objets de données, services, interface, etc. Chaque façon d'implémenter et le choix des patrons de conception induisent une définition des éléments différents.

Note

Pour aller plus loin sur la **détection de composants**, de nombreux travaux tentent de résoudre ce problème [5], deux catégories d'études existent [51] :

- Le ***component recovery*** tente de retrouver au moins un composant dans une application [60] ;
- La ***componentisation*** cherche à appliquer une architecture à composants sur une application [2, 90].

Chardigny [23] propose un processus quasi-automatique d'identification d'architectures à partir d'un système orienté objet existant. Le processus d'abstraction repose sur des propriétés sémantiques, de qualité architecturale, et encore sur l'architecture intentionnelle du système à partir de la documentation et des recommandations de l'architecte.

La troisième expérimentation est la transformation du modèle KDM, obtenu durant la deuxième expérimentation du cas d'étude SAMM, pour obtenir le modèle App.

EXPÉRIMENTATION 5.3 – Transformation du modèle KDM de SAMM vers App

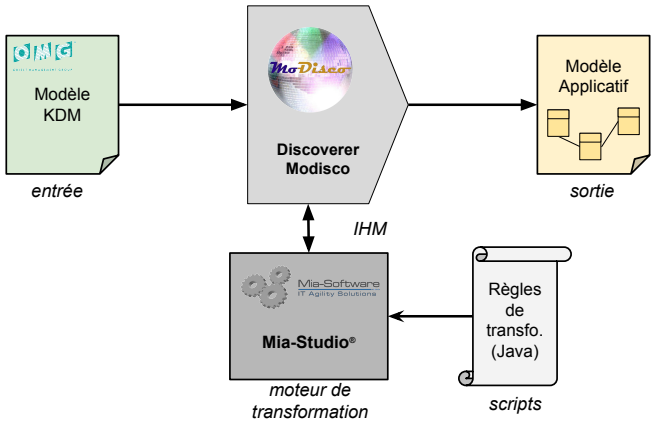


FIGURE 5.11 – Schéma du processus de transformation applicative

Pour la dernière étape, nous avons écrit la transformation du modèle KDM vers le modèle applicatif avec le logiciel Mia-Transformation. Les règles de transformations sont regroupées en deux collections :

- Une première collection structurelle correspond aux langages de programmation orientés objet : classes, interfaces, méthodes, paquetages...
- Une seconde collection sémantique détecte les composants et les objets de données du programme. Cette analyse est la plus complexe.

Les règles de transformation sémantique ont été adaptées au cas d'étude qui présente une particularité singulière : les concepts d'architecture sont intégrés dans le nom des classes et des interfaces Java via un préfixe. Ainsi, la reconnaissance des services, des interfaces de services et des objets de données est relativement aisée. En sortie de la transformation nous obtenons un fichier XMI de 2 Mo ; cette taille est bien éloignée des modèles précédents. C'est en cela que l'abstraction est intéressante, nous obtenons un nouveau point de vue de l'application plus pratique à lire et à manipuler.

Listing 5.1 – Règles de conformance OCL sur App

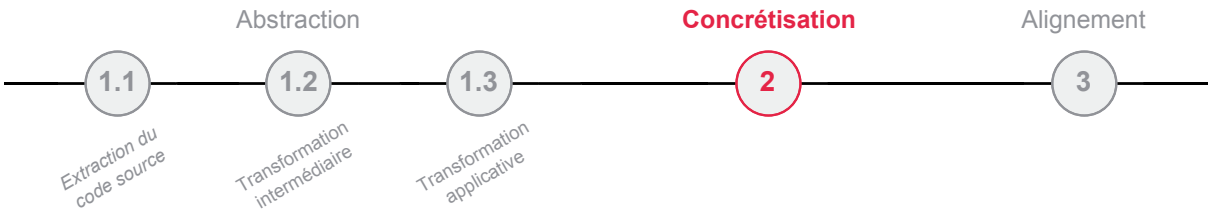
```
context DataObject
  inv: self.accessedByFunction.realizes.accesses -> exists(s | s = self)
context Service
  inv: self.realizedBy.assignedFrom.composedOf.assignedTo ->
                                                    exists(s | s = self)
```

Enfin, nous avons vérifié la structure des instances d'objets de données et de services du modèle obtenu avec les règles OCL incorporées dans le méta-modèle (listing 5.1). Les règles sont vérifiées avec succès, aucune erreur n'est remontée. La transformation est non seulement correcte, mais le code source est également complet pour peupler le modèle applicatif.

L'expérimentation 5.3 a permis de vérifier la faisabilité de la transformation d'un modèle représentant le code source d'une application à l'aide de KDM vers un modèle applicatif. Cette abstraction a permis la réduction du volume d'informations en détectant les éléments conformes à la définition des concepts du méta-modèle App.

L'étape suivante consiste à exploiter les modèles métiers dans le but de réaliser l'alignement.

5.3.2 Étape de concrétisation des modèles du SI



L'élaboration des modèles de processus métier et fonctionnel est une tâche manuelle effectuée par les responsables métiers. Si ces modèles existent déjà, nous pouvons créer une transformation depuis les modèles disponibles ou le référentiel d'entreprise, vers des modèles conformes à la définition de nos méta-modèles. Dans la section 3, nous avons proposé le tableau 3.3 de traduction des concepts des méta-modèles processus normés les plus couramment rencontrés (UML Activité et BPMN) vers notre méta-modèle BPM ; et d'un méta-modèle fonctionnel de référentiels commerciaux les plus utilisés (MEGA, Aris, CaseWise) vers notre méta-modèle Fun. Si des méta-modèles métiers n'ont pas été choisis dans l'entreprise, nos méta-modèles sont de bons candidats pour commencer la cartographie à partir de zéro.

Dans la suite, nous présentons trois expérimentations correspondant à trois scénarios spécifiques :

1. dans le cas d'étude **SAMM**, l'étape de concrétisation est la traduction des concepts des diagrammes processus existants vers les concepts de notre méta-modèle BPM ;
2. dans le cas d'étude **SAMI**, l'étape de concrétisation est réalisée par l'extraction et la transformation depuis les sources d'un référentiel d'entreprise ;
3. dans le cas d'étude **SAMUT**, l'étape de concrétisation est réalisée manuellement par un architecte d'entreprise.

L'expérimentation suivante est la traduction de diagramme processus du référentiel MEGA du cas d'étude SAMM pour obtenir un modèle BPM.

EXPÉRIMENTATION 5.4 – Traduction des diagrammes processus de SAMM

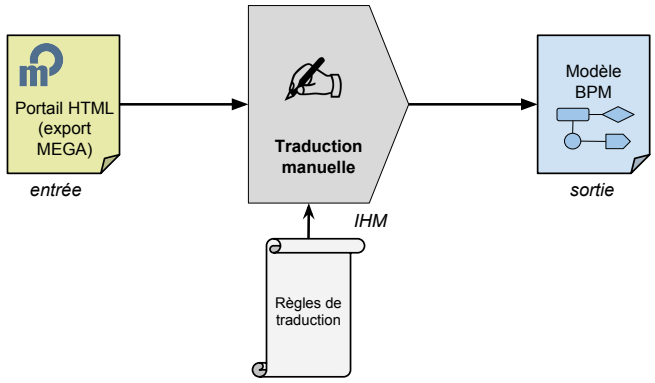


FIGURE 5.12 – Schéma du processus S.2 manuel

Le référentiel d'entreprise MEGA fourni dans le cas SAMM n'est pas exploitable directement, car il s'agit d'un export HTML statique et les diagrammes sont sous forme d'images. Ainsi, il n'y a ni modèle XMI à charger ni d'extraction possible.

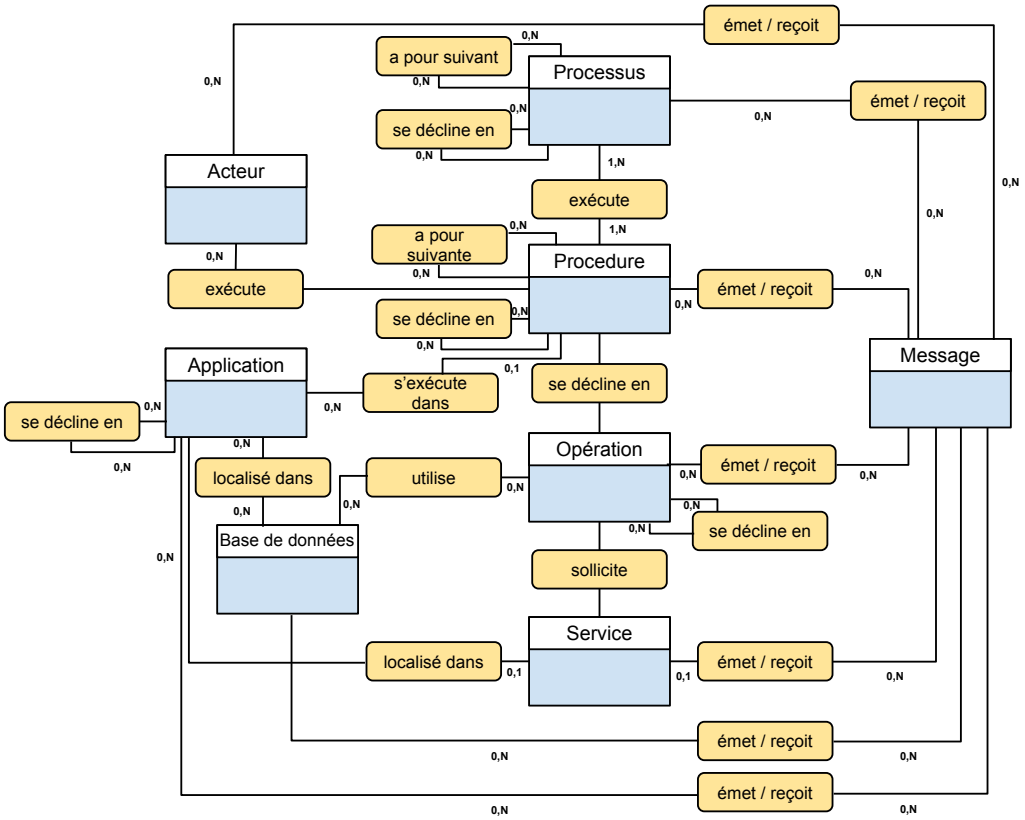


FIGURE 5.13 – Méta-modèle de SAMM

Nous avons alors décidé de réaliser une traduction manuelle d'une partie des diagrammes vers notre modèle métier BPM (cf. table 5.1). Les règles de traduction ont été déterminées à partir du méta-modèle MEGA spécifique défini à la figure 5.13.

Concept MEGA	Concept BPM	Description
Acteur	Acteur / Participant	Élément actif chargé d'une ou plusieurs activités dans le processus
Message	Transition	Lien orienté entre deux activités
Processus	Processus	Ensemble d'activités liées ayant même finalité
Procédure	Activité	Ensemble de tâches décrivant une vue d'ensemble d'une activité.
Opération	Tâche	Plus petit élément de décomposition d'une activité

TABLE 5.1 – Règles de traduction des concepts processus métier

Pour l'expérimentation, nous avons réduit le périmètre de traduction à un ensemble de diagrammes représentatifs correspondant à une partie de l'activité de SAMM appelée *Adresse Bancaire*. En effet, le référentiel d'entreprise MEGA contient 360 diagrammes, le travail complet de traduction serait beaucoup trop long et dénué d'intérêt au vu de nos objectifs de validation de l'approche.

L'expérimentation 5.4 a permis de vérifier la généricité de notre méta-modèle BPM. Tous les concepts du méta-modèle spécifique à SAMM ont bien été couverts par des concepts équivalents de BPM durant la traduction.

L'expérimentation suivante est l'extraction du référentiel MEGA du cas d'étude SAMI pour obtenir les modèles conformes aux méta-modèles BPM, Fun et App. Le cas d'étude SAMI n'a aucun code source disponible, cependant le référentiel d'entreprise MEGA contient les trois points de vue : processus métier, fonctionnel et applicatif. Dans le cas SAMI, il n'y a donc pas d'étape d'abstraction, mais uniquement une étape de concrétisation. L'alignement est également déjà réalisé dans le référentiel, il sera traduit durant la transformation. Nous abordons l'intérêt de refaire un alignement dans la section 5.3.3.

EXPÉRIMENTATION 5.5 – Extraction des modèles AE de SAMI depuis MEGA

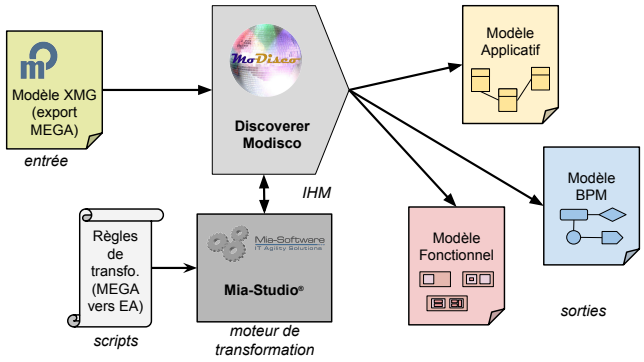


FIGURE 5.14 – Schéma du processus S.2 automatique

Le fichier source du référentiel est un export MEGA au format XML appelé XMG. Nous avons extrait les différents concepts pour construire les modèles processus, fonctionnel et applicatif à l'aide d'une transformation écrite avec Mia-Transformation illustrée en figure 5.15.

La transformation se déroule en deux étapes :

1. La première étape est la traduction des instances de concepts MEGA vers les instances de concepts de nos méta-modèles BPM (processus, activités), Fun (blocs, fonctionnalité) et App (services, composants applicatifs) ;
2. La seconde étape est la constitution des liens entre les nouvelles instances de concepts créés à partir de la traduction des liens entre les instances de concepts MEGA d'origine.

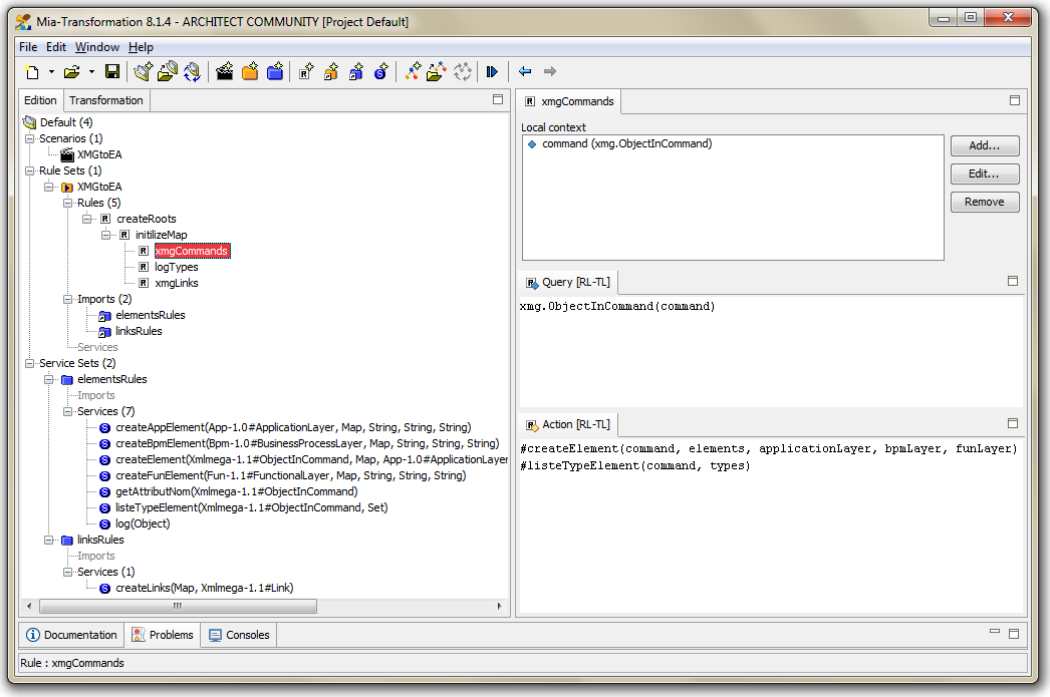


FIGURE 5.15 – Transformation de modèle XMG avec le logiciel Mia-Transformation

Tous les concepts de nos méta-modèles ne sont pas présents dans les sources MEGA, le tableau 5.2 dresse la liste des traductions réalisées.

Méta-Modèle AE	Concept AE	Concept MEGA
App	ApplicationComponent Service Interface Function DataObject	Application, Composant Service
BPM	Actor Task Role Event Condition Transition Process Activity DataObject	Acteur Tâche Rôle Evénement, Evénement (UML) Condition , Branchement Enchaînement, Message Processus Activité fonctionnelle
Fun	Functionality Block DataObject	Fonctionnalité

TABLE 5.2 – Règles de traduction des concepts processus métier

Certains concepts du MEGA n'ont pas été traduits soit par manque d'équivalence dans nos méta-modèles (Échange, Message, Synchronisation, Interface Homme-Machine, Attribut...), soit par leur nature technique propre à MEGA pour la représentation diagramme (Diagramme, Adresse prédéfinie, Ligne de vie, Référence externe...) ou utilisés comme documentation (Note, Mot-clé).

L'expérimentation 5.5 a permis de mettre en œuvre une extraction par transformation des informations depuis un référentiel d'entreprise vers nos modèles processus, fonctionnel et applicatif. La transformation des concepts n'est pas "1 pour 1", il y a une perte

d'informations en provenance du référentiel. La construction de la transformation se réalise par itérations en ajoutant au fur et à mesure les concepts manquants. Le temps pris par cette opération dépendant du nombre de concepts, l'architecte doit juger des conséquences de cette perte, en cas d'insuffisance de l'exhaustivité, il serait nécessaire d'étendre nos méta-modèles AE. Un dernier problème, dans le référentiel les concepts sont mélangés, il peut être difficile de reconnaître le point de vue d'appartenance.

L'expérimentation suivante est la modélisation fonctionnelle du cas SAMUT.

EXPÉRIMENTATION 5.6 – Modélisation fonctionnelle de SAMUT

Comme évoqué en introduction de chapitre, ce cas d'étude n'a initialement aucun modèle métier. Après audit des responsables métier et utilisateur, un architecte a modélisé et identifié des blocs fonctionnels en figure 5.16 .

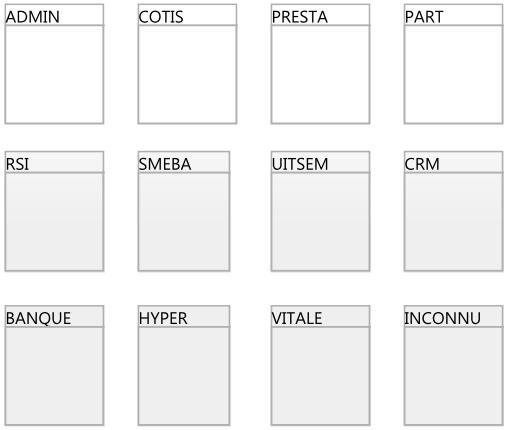


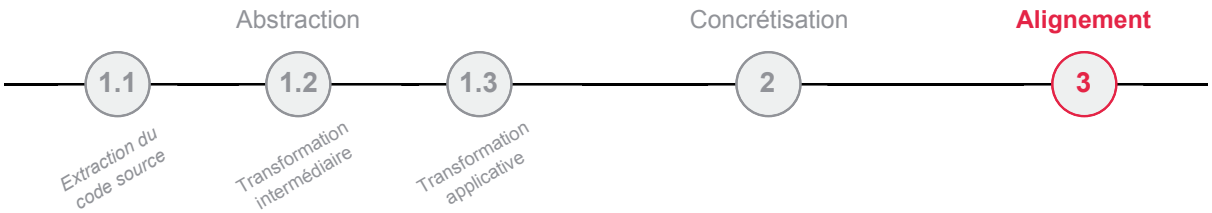
FIGURE 5.16 – Modèle fonctionnel SAMUT

Le patrimoine de SAMUT est donc composé de 12 blocs fonctionnels.

L'expérimentation 5.6 a permis de tester l'exhaustivité des concepts de notre méta-modèle fonctionnel pour modéliser l'organisation d'une entreprise. Dans cette expérience la modélisation a été simpliste, seuls les concepts de blocs et de fonctionnalités ont été utilisés.

L'étape suivante consiste à aligner les modèles processus et fonctionnel obtenus à cette phase de concrétisation avec les modèles applicatifs.

5.3.3 Étape d'alignement par tissage



La dernière étape est la mise en œuvre effective de l'alignement à l'aide de la méthode par facettes selon la définition de l'alignement détaillée dans la section 4. L'alignement est réalisé de deux façons différentes selon le cas rencontré :

1. **Manuellement** Les différents modèles métiers et applicatifs existent mais ne sont pas mis en relation. L'architecte, à partir du recueil des connaissances du SI, soit par un audit des utilisateurs, soit par la documentation existante, reconstitue les liens entre les différentes instances des modèles. Notre éditeur présenté à la section 4 l'assiste pour réaliser le tissage entre les instances ;
2. **Automatiquement** L'information d'un ancien alignement existe, par exemple il est possible d'extraire les liens depuis un référentiel d'entreprise. Les raisons d'une reprise d'un alignement peuvent être : l'abandon du référentiel existant au profit d'un nouvel atelier de modélisation, ou la nécessité d'ajout de nouveaux types de lien.

Notre méthode d'alignement par tissage dépasse le simple travail de cartographie et procure différents avantages.

Interopérabilité

Un référentiel d'entreprise n'établit généralement pas de lien avec le code source, alors que notre alignement de modèles peut être réalisé à partir de sources variées : code source, documentation, modèle et référentiel d'entreprise.

De plus, le support existant contenant les informations d'alignement peut être difficilement exploitable. Par exemple dans le cas SAMUT, les informations sont mémorisées dans une feuille de tableur. La transformation de ces informations vers notre modèle de tissage par facettes permet d'obtenir un alignement avec la navigation entre les modèles.

Extensibilité

L'alignement existant peut présenter des manques dans les liens entre concepts. Un réalignement par facettes permet de réviser et d'étendre la définition, comme par exemple ajouter un nouveau point de vue à l'alignement existant. Il est également aussi possible d'enrichir les concepts eux-mêmes. Les facettes permettent d'ajouter de nouveaux attributs et de réaliser de nouvelles classifications.

Exploitabilité

L'alignement n'est pas une simple traçabilité statique, les liens entre concepts peuvent être analysés à l'aide de notre outil d'écriture de requêtes OCL (cf. section 4.3.5). L'analyse par requêtes permet de calculer des indicateurs que nous définissons dans le chapitre 6. Les indicateurs permettent d'assister l'architecte dans la prise de décision des actions à effectuer sur le SI.

Dans la suite, nous présentons deux expérimentations correspondant aux deux scénarios spécifiques :

1. à partir du cas d'étude **SAMM**, l'alignement est réalisé par assistance à l'aide de notre éditeur entre un modèle applicatif et processus ;
2. à partir du cas d'étude **SAMUT**, l'alignement est réalisé de façon automatique à partir d'informations provenant d'un tableur.

La première expérimentation d'alignement est le tissage manuel du cas d'étude SAMM entre le modèle applicatif issu du processus abstraction (cf. expérimentation 5.3), et le modèle BPM issu du processus de concrétisation (cf. expérimentation 5.4).

EXPÉRIMENTATION 5.7 – Tissage du cas SAMM assisté à l'aide de l'éditeur

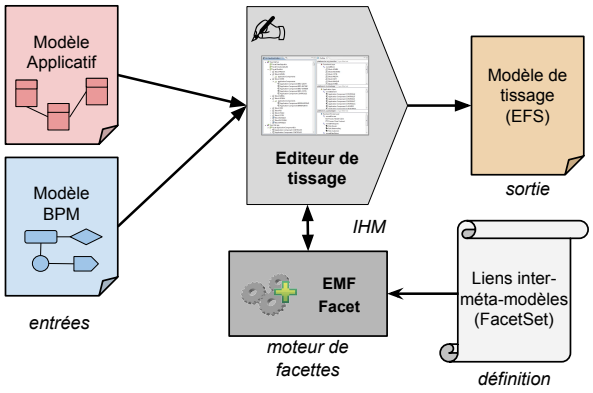


FIGURE 5.17 – Schéma du processus d'alignement

Dans le cas SAMM, nous avons chargé les modèles processus et applicatifs avec notre éditeur de tissage et la définition de l'alignement par facettes. Pour établir des liens entre les tâches provenant du modèle métier et les services provenant du modèle applicatif, nous recherchons les correspondances à l'aide de la fonction de filtre par nom d'instance. Ensuite, l'éditeur permet de **glisser** des éléments filtrés provenant de la partie de droite pour les **déposer** vers la partie de gauche, le résultat est la construction du lien. L'action de dépôt vérifie si le type de concept de l'instance est autorisé à être aligné avec l'instance ciblée.

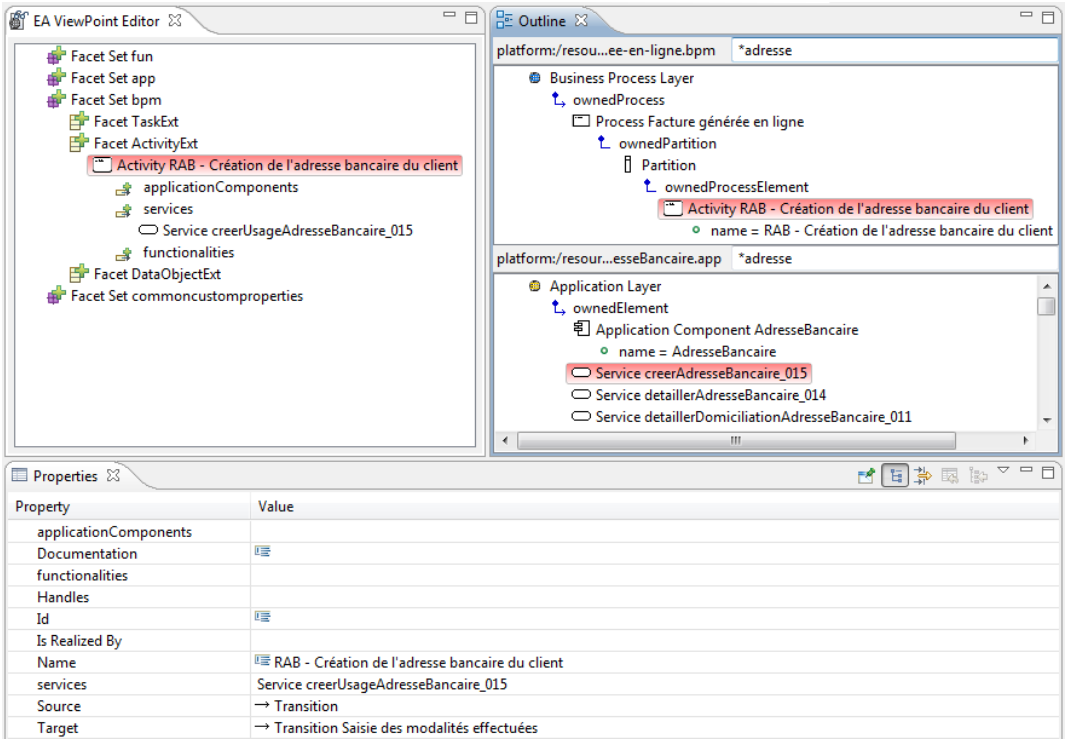


FIGURE 5.18 – Capture d'écran de l'alignement du cas SAMM avec notre éditeur

La figure 5.18 illustre comme exemple le résultat du tissage de l'activité "*Création de l'adresse bancaire du client*" alignée avec le service "*creerAdresseBancaire_015*". Cette unique fenêtre permet de regrouper les différents modèles de travail et le résultat de l'alignement.

L'expérimentation 5.7 a permis de tester notre méthode d'alignement à l'aide de facettes avec l'assistance de notre éditeur entre des modèles applicatif et processus métier.

La seconde expérimentation est l'alignement automatique des modèles fonctionnel et applicatif du cas d'étude SAMUT.

EXPÉRIMENTATION 5.8 – Tissage automatique par transformation

Dans le cas SAMUT, nous avons réalisé le tissage entre les blocs et les composants applicatifs. La correspondance entre les blocs et les composants applicatifs étaient initialement enregistrée dans une feuille de calcul de tableur à l'aide de deux colonnes. Puis, les requêtes OCL ont permis d'isoler des composants qui étaient orphelins (qui n'étaient rangés dans aucun bloc) pour affiner le tissage.

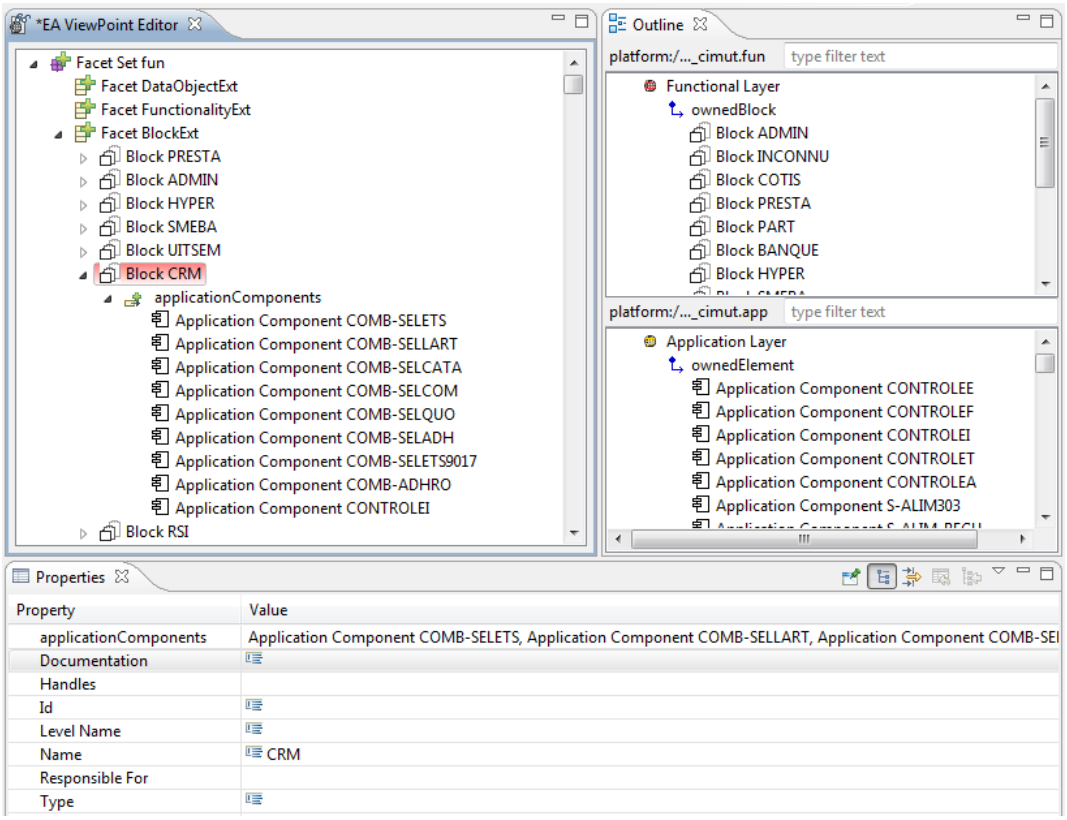


FIGURE 5.19 – Capture d'écran de l'alignement du cas SAMUT avec notre éditeur

La figure 5.19 illustre comme exemple le résultat du tissage du bloc "*CRM*" aligné avec 9 composants applicatifs ("COMB-SELETS", "COMB-SELLART"...).

L'expérimentation 5.8 a permis de tester l'interprétation d'informations d'alignement contenues dans un modèle tiers, et de parcourir l'alignement entre des modèles fonctionnel et applicatif à l'aide de notre éditeur.

Les différentes étapes de notre processus ont été détaillées et illustrées à l'aide d'expérimentations tirées de cas réels d'entreprise. Dans la section suivante, nous en faisons le bilan.

5.4 Le bilan des expérimentations

Le tableau 5.3 résume le nombre de concepts obtenus après extraction des données vers chaque modèle (Fun, BPM, App). Une cellule est vide si le cas d'étude n'a pas de donnée correspondante en entrée. La dernière colonne décrit la réalisation de l'étape d'alignement : manuelle avec notre éditeur de tissage 4.28, ou automatique à l'aide d'une transformation.

TABLE 5.3 – Comparaison entre les études de cas

Case Study	Fun		BPM		App					Weaving
	Block	Fun	Process	Activity	Comp.	Func.	Service	Interf.	DataObj.	
SAMM			360		1 002	4 808	2 334	502	4 203	Manual
SAMI	18	131	167	268	625		11 894			Auto
SAMUT	12				1 045				669	Manual

SAMM L'expérimentation 5.1 nous a permis de tester notre processus d'abstraction à partir d'un code source Java de taille importante. Le chargement des modèles obtenus par rétro-ingénierie de l'expérimentation 5.2 a été une mise à l'épreuve pour nos outils : allocation de mémoire vive, temps de traitement processeur. Le travail de transformation vers le modèle applicatif dans l'expérimentation 5.3 a été facilité par la présence d'une nomenclature des concepts qui par là même montre de bonnes pratiques de codage, même si des exceptions subsistent. Néanmoins, nous regrettons de ne pas avoir eu accès à la source originale du référentiel d'entreprise MEGA pour réaliser un alignement par tissage complet dans l'expérimentation 5.7.

SAMI L'expérimentation 5.5 révèle que notre méthode permet de réinterpréter des modèles et leur alignement depuis un référentiel d'entreprise existant par transformation. Le but de reprendre un existant est d'accroître les possibilités de documentation des points de vue à l'aide des facettes et moderniser le SI. Par exemple, lorsque des services applicatifs sont trop nombreux, il est possible d'introduire une nouvelle classification et obtenir une nouvelle granularité entre les services.

SAMUT La méthode utilisée dans les expérimentations 5.6 et 5.8 a permis d'assister l'architecte pour créer un tissage de façon efficace, car la correspondance entre les blocs et les composants applicatifs étaient initialement enregistrée dans une feuille de calcul de tableur. Puis, les requêtes OCL ont permis d'isoler des composants qui étaient orphelins (non rangés dans un bloc) pour affiner le tissage.

Les différents aspects détaillés dans le tableau 5.3 (absence ou présence de code source, référentiel, couche métier : fonctionnel ou processus, nombre d'éléments) de chaque cas d'étude ont permis de prouver la souplesse de notre démarche qui à chaque étape peut s'adapter. Néanmoins, chaque cas d'étude avec une source d'une typologie nouvelle implique de personnaliser le processus à son besoin : écrire ou adapter une transformation. Nos outils sont paramétriques, évolutifs et adaptables.

La transformation d'un langage de programmation vers KDM est réutilisable pour tous les cas aux versions près du langage de programmation. Par exemple, un code source écrit en Java 7 n'est pas entièrement interprétable par un extracteur écrit pour Java 6 et nécessite quelques modifications. De même, le langage Cobol présente une variété de versions (60, ..., 85, 2002, 2014) et de spécificités selon les éditeurs (Micro Focus, Fujitsu, IBM...).

La transformation qui nécessite le plus d'investissement d'écriture est la transformation de KDM vers notre modèle App dépendante des patrons de conception, des cadriciels et de l'architecture de développement adoptée. Cet effort doit s'inscrire dans une perspective de réutilisation de la transformation. La panoplie d'outils est un compromis entre **généricité** pour traiter un grand nombre de situations et **interopérabilité** pour traiter les spécificités de chaque situation.

5.5 Le bilan du processus d'IDM

Le succès des expérimentations d'entreprises a montré l'intérêt du processus d'alignement par ingénierie des modèles, notamment par l'obtention des modèles par transformations successives et la composition des modèles résultants à l'aide de facettes. Nous établissons un relevé des principaux avantages et qualités du processus proposé.

Généricité Les différents méta-modèles et les traductions proposées en section 3 composent un socle compatible avec les différents concepts rencontrés dans les expérimentations. Cette généricité a permis d'utiliser sans aucune adaptation notre définition de l'alignement et son implémentation par FacetSet. Cependant, l'utilisation de notre définition et son extension sont ouvertes. L'architecte peut choisir une norme de notation plutôt que nos méta-modèles, par exemple BPMN2, UML, ou Archimate. Dans ce cas, il sera nécessaire de fournir ou d'adapter la définition de l'alignement et le FacetSet aux concepts équivalents. L'écriture d'un FacetSet est rapide et souple, ce qui est un avantage important comparé aux autres techniques de composition nécessitant la modification d'un méta-modèle (cf. section 4.2.6). Notre méthode propose l'utilisation de trois modèles, dans certaines expérimentations nous n'en avions que deux de disponibles. L'architecte peut tisser autant de modèles nécessaires par extension de la définition de l'alignement et du FacetSet (cf. section 4.1.4).

Interopérabilité La méthode supporte des types d'entrée différents. L'architecture d'entreprise de SI patrimoniaux nécessite d'être compatible avec des informations très variées : complète/incomplète, à jour/obsolète, structurée/textuelle. Le niveau de maturité est également très différent d'une entreprise à une autre, les utilisateurs documentent le SI avec : une suite bureautique (Word, Excel, Powerpoint), des outils de modélisation (Rational Rose, StarUML, SysML, PowerDesigner), des référentiels d'entreprise (MEGA, Aris et CaseWise). Ainsi, nous avons utilisé ou créé des analyseurs et des transformations différents selon le format en entrée : XML/XMI, fichiers code source, base de donnée, fichier tableur. Si une analyse syntaxique n'est pas possible, l'utilisateur doit saisir les informations manuellement à l'aide de nos éditeurs de modèles.

Automatisation À l'aide de notre méthode lorsque le support est un modèle, nous pouvons enchaîner des transformations de façon automatique. Dans le cas contraire, le scénario ne peut être couvert à 100% par automatisation. Néanmoins, la place de l'utilisateur est indispensable durant l'alignement, les informations partiellement manquantes ou erronées doivent être corrigées ou complétées de façon manuelle.

Performance et passage à l'échelle La cartographie d'entreprise contient un volume d'informations important disséminé dans plusieurs modèles. Mettre en relation les différentes informations dans un tel volume sans assistance est une tâche difficile

pour l'utilisateur. Nos outils sont capables de charger des modèles de grande taille pour les visualiser et les manipuler. L'interface utilisateur fournit une fonctionnalité de recherche pour trouver les concepts à mettre en relation.

Ces différentes techniques issues de l'IDM ont été implémentées à l'aide d'outils. La prochaine section détaille les différents outils utilisés par chaque étape du processus.

5.6 Le bilan des outils supports du processus

Nous avons utilisé différents outils pour mettre en œuvre les différentes expérimentations du processus : Eclipse Modisco, Mia-Transformation et Eclipse EMF Facet. Nous détaillons outil par outil, ce que nous avons réutilisé, et ce que nous avons développé.

Transformations avec Modisco

Existant Nous avons réutilisé la transformation "texte vers modèle" de l'expérimentation 5.1 qui permet de construire un modèle Java à partir d'un code source Java dans un projet Eclipse.

Développé Nous avons développé l'interface de lancement, par menu contextuel, des transformations Mia-Transformation suivantes :
- **Java vers KDM** pour l'expérimentation 5.2 ;
- **KDM vers Java** pour l'expérimentation 5.3 ;
- **XMG MEGA vers BPM, Fun et App** pour l'expérimentation 5.5.

Transformations avec Mia-Transformation

Existant Néant.

Développé Nous avons écrit les transformations "modèle vers modèle" suivantes :
- **Java vers KDM** de l'expérimentation 5.2 ;
- **KDM vers Java** de l'expérimentation 5.3 ;
- **XMG MEGA vers BPM, Fun et App** de l'expérimentation 5.5.

Alignement avec EMF Facet

Existant Nous avons utilisé le moteur d'extension de méta-modèle par facettes du projet open source Eclipse EMF Facet.

Développé Nous avons réalisé des modifications sur EMF Facet (cf. section 4.3.2) et créé l'éditeur de tissage (cf. section 4.3.4). Nous les avons utilisés pour les expérimentations 5.7 et 5.8.

Nous avons utilisé les projets open source existants Eclipse Modisco et Eclipse EMF Facet comme socle d'outillage. À partir de leurs interfaces de programmation applicative (**API**), nous avons développé des interfaces utilisateurs et amélioré leur fonctionnement pour nos besoins spécifiques (persistance du résultat d'alignement), tandis que nous avons utilisé Mia-Transformation comme éditeur pour écrire les scripts de transformations et comme moteur d'exécution de nos transformations.

Il existe des travaux connexes utilisant les mêmes outils issus de la plateforme open source Eclipse. Dans la section suivante, nous passons en revue ces différents travaux, et nous nous positionnons par rapport à eux.

5.7 Des travaux connexes d'architecture d'entreprise exploitant l'IDM

Pour élaborer notre processus basé sur les techniques d'ingénierie des modèles, nous avons réalisé un état de l'art spécifique. Tout d'abord, nous avons fait des recherches sur les langages et techniques de modélisation pour élaborer les modèles. Puis, nous avons étudié les techniques de composition de modèles et d'alignement pour l'architecture d'entreprise. Enfin nous avons exploré les solutions pour construire les modèles et vérifier les concepts de l'alignement.

Langages et techniques de modélisation

L'architecture d'entreprise est un domaine où l'utilisation de l'ingénierie des modèles est devenue incontournable [54]. Cette pratique a été facilitée par l'émergence de standards et normes : *Meta-Object Facility* **(MOF)** a permis de créer un langage commun pour la définition des méta-modèles ; *XML Metadata Interchange* **(XMI)** basé sur la représentation en arborescence XML, a permis de créer un langage commun pour la persistance des modèles. MOF et XMI permettent l'interopérabilité des modèles de leur création à leur modification : modélisation, extension, transformation.

Vara et Marcos [101] proposent une revue des cadriciels basés sur l'ingénierie des modèles. Ils constatent que deux courants existent pour la modélisation : l'utilisation de langage dédié **(DSL)** ou d'UML étendu par profils. La technique de DSL est plus souple, les méta-modèles produits sont plus simples qu'avec les profils UML. Les profils UML requièrent une spécification plus complexe et peuvent provoquer de potentiels problèmes avec la persistance XMI.

Le cadriciel Eclipse EMF est majoritairement utilisé grâce à tout l'environnement d'outillage qui gravite autour de cette technologie [18] :

- GMF, Graphiti et Sirius (tant à remplacer les deux premiers outils) pour la création de **diagrammes** adaptés à un DSL : design, icône, couleur, palette d'outils ;
- ATL, QVT, Xpand, MOFscript et Acceleo comme langage et outils pour réaliser des **transformations** ;
- Xtext pour créer une **grammaire dédiée** à un DSL utilisable comme modèle ;
- ...

Plusieurs entreprises développent des outils d'IDM autour du cadriciel Eclipse EMF dont notamment Obeo et Mia-Software, tandis que des éditeurs proposent des suites logicielles commerciales pour cartographier un SI à l'aide des méthodologies d'architecture d'entreprise. Ces logiciels sont CaseWise, Aris, Modelio, Sparx Enterprise, etc.

Chaque référentiel fournit des méta-modèles standards qui définissent les concepts génériques pour chaque point de vue. Pour naviguer entre les différents points de vue, une relation par lien de traçabilité est possible. Ici est la différence avec notre technique d'alignement, puisque les liens de traçabilité sont génériques et ne possèdent aucune sémantique. L'analyse de l'alignement est donc limitée, mais les dépendances peuvent être analysées. Par exemple, Modelio propose un diagramme d'analyse d'impact pour identifier les conséquences de modifications sur le SI. MEGA fournit également une analyse d'impact automatique pour chaque objet cartographié. Et pour comparer différentes versions d'un point de vue, Aris offre une comparaison entre modèles "source" et "cible" à l'aide d'un éditeur de requêtes, le résultat de l'analyse d'impact est généré dans un tableur.

Composition de modèles

Une référence sur le sujet de la composition de modèles est la thèse de Clavreul [30]. Il explore les différentes façons de composer les modèles de deux manières différentes : modélisation et vérification. Il détermine une classification des techniques de transformation et composition de modèles selon deux catégories : les correspondances et les interprétations. Il définit une correspondance comme «*la similarité entre deux modèles ou plus à partir d'un ensemble de règles d'alignement entre des «patterns» d'éléments de modèles.*». Cette définition présente des similitudes avec notre approche d'alignement entre concepts de modèle. Afin d'unifier les définitions, il propose un langage de modélisation et une méthodologie pour la construction de compositions de modèles (ModMap).

Atkinson *et al.* [8] font un constat identique sur les besoins d'étendre un méta-modèle selon deux scénarios : l'ajout de propriétés pour son amélioration, l'extension de propriétés vers un autre méta-modèle. Après avoir testé la technique d'annotation, ils proposent une modélisation multi-niveau basée sur leur approche de tissage *Orthogonal Classification Architecture* (OCA). Ils ont également choisi EMF pour étendre les méta-modèles de représentation des systèmes d'information, tout comme EMF Facet, cette méthode d'extension de méta-modèle est non intrusive et s'applique à l'exécution. Leur outil open-source Melanee[5] doit implémenter leur approche dans une prochaine version.

Alignement assisté du SI

Wieringa *et al.* [107] proposent une méthode pour aligner l'architecture applicative au contexte métier. Quelques bonnes pratiques peuvent être reprises pour aider l'architecte dans le processus de cartographie, mais aucun outil n'est proposé.

Chen *et al.* [26] proposent par leur méthode *Semantic Enterprise Architecture Management* (SEAM), un alignement des modèles d'architecture d'entreprise dans un référentiel centralisé. Ils proposent d'utiliser des technologies sémantiques pour intégrer de façon hétérogène les différentes sources pour représenter le SI, particulièrement les modèles conformes aux notations TOGAF, BPMN2 et SOA. L'apport de ce travail est l'approche automatique pour prendre en compte les éventuels conflits de modèles lors de l'ajout d'information. Cependant, aucun outil n'est proposé et il s'agit d'une approche de fusion qui comme nous l'avons vu pose le problème de perte de synchronisation et donc de mise à jour des modèles d'origine.

Mettre en relation différents points de vue est une difficulté lorsqu'on utilise uniquement des modèles normés. Une solution possible est celle d'utiliser un modèle intermédiaire pour enregistrer les liens de relation. Par exemple, Meertens *et al.* [72] décrivent des relations de concepts entre Archimate et la représentation métier *Business Model Ontology* (BMO) de haut niveau du système d'information. Leur but est d'étendre Archimate pour combler des manques conceptuels. Nous partageons des similitudes avec le travail de Meertens *et al.* : ils proposent une définition des liens et expérimentent un cas tiré du secteur de l'assurance. Cependant, leur étude ne considère que deux modèles et aucune technique n'est proposée. Proposer une implémentation à l'aide de EMF Facet en écrivant un FacetSet à partir de leur définition pourrait être une contribution à leur travail.

D. Castro *et al.* [34] décrivent une approche "*top-down*" qui vise à générer une architecture web orientée service à partir de modèles métier à l'aide de Eclipse EMF. Ils détaillent de façon similaire à notre approche les règles de transformation et les concepts

[5] http://www.melanee.org

qui relient les différentes couches. Cependant, nous ne partageons pas le même but. Notre point de départ est un patrimoine applicatif quelconque. Notre méthode doit répondre à l'homogénéité des architectures techniques existantes dans un SI et des différentes représentations métiers possibles de ce SI. Elvesaeter *et al.* [42] proposent une méthode pour aligner les modèles métier BPMN et une architecture orientée service (SOA). La méthode consiste à réaliser des transformations à l'aide d'ATL vers le modèle SoaML. Les démarches de D. Castro et Elvesaeter se préoccupent uniquement des architectures SOA, cette structure homogène par la notion de services est de fait plus facilement traçable entre les couches.

Jouault *et al.* [58] font des constats similaires à nos travaux : un SI est hétérogène et composé de modèles utilisant différents DSL évoluant au cours des choix technologiques. Ils proposent une approche combinant deux techniques : le tissage de modèles et la gestion globale de modèles (le "*megamodeling*"). Leur méthode est implémentée à l'aide d'outils faisant également appel au cadriciel EMF : AMW et AM3[6]. Cependant, les deux projets semblent ne plus évoluer et être abandonnés depuis plusieurs années, les dernières versions datent de 2010 pour AMW, et 2011 pour AM3. Ce que proposait AMW est couvert par les améliorations que nous avons apportées à EMF Facet.

Clark *et al.* proposent une méthode d'alignement simplifiée basée sur le langage d'exécution LEAP. Comparé à Archimate, les concepts de LEAP incluent les concepts d'Archimate ; LEAP est plus généraliste et proche d'un langage de description d'architecture (ADL) bien qu'il n'en soit pas un. L'alignement est basé sur une relation de raffinement entre les éléments du modèle. Un outil est fourni pour éditer les modèles et analyser la conformité de raffinement par simulation : les modèles logiques et physiques doivent produire les mêmes résultats. Nous partageons le point de vue pratique de la méthode LEAP, néanmoins son inconvénient porte sur la partie métier, plutôt négligée, en se concentrant principalement sur la partie informatisée.

Wegmann *et al.* proposent la méthode *Systemic Enterprise Architecture Methods* (SEAM)[7] et l'outil SeamCAD pour l'alignement métier et applicatif [102, 103]. Les modèles multi-niveaux sont définis en utilisant le langage de spécification *Alloy* pour établir une définition formelle de l'alignement et la vérifier. Des similitudes existent avec les travaux de Clark qui se basent sur des concepts de comportements compatibles. Ainsi, les modèles des différents points de vue supportent des concepts comparables.

LEAP et SEAM utilisent tous deux des modèles et outils spécifiques, alors que nous utilisons des techniques standards. Aucune solution n'est proposée pour réaliser un alignement manuel, ni même de rétro-ingénierie pour les systèmes existants.

Mise à part la mise en œuvre technique de l'alignement, il s'agit également d'un problème d'intégration de SI à la suite d'acquisition d'entreprises concurrentes. Anaya et Ortiz [4] illustrent cette intégration en concrétisant les relations d'impact et de causalité, mais ils ne délivrent pas de solution pour la mise en pratique. Al Mosawi [76] propose une architecture d'*intégration d'application d'entreprise* (IAE) basée sur cinq types de modèles dissociant différentes préoccupations du domaine informatique : modèle technologique, modèle de transaction et service, modèle de service applicatif, modèle intra-application, et modèle inter-application. Cette séparation va bien plus loin que notre unique modèle applicatif, l'intégration d'applications nécessite d'introduire des notions spécifiques pour modéliser les interactions entre applications.

[6] *Atlanmod MegaModel Management*

[7] Ne pas confondre avec les travaux de Chen *et al.* utilisant le même acronyme SEAM

Construction et vérification de l'alignement

Le pré-requis à l'élaboration de l'alignement est d'obtenir les modèles à tisser. Lors de l'expérimentation des cas d'études, nous avons réalisé différentes transformations. L'écriture des scripts de transformation des concepts sources (référentiel MEGA, notations standards : BPMN, UML...) vers des concepts cibles de nos méta-modèles d'entreprise (BPM, Fun, App) est spécifique et prend du temps pour chaque cas d'étude. Le travail de Limyr [65] propose une technique d'assistance à la transformation *Semaphore* à l'aide d'un éditeur visuel pour réaliser les correspondances entre concepts sources et cibles, ainsi que les attributs, entre les différents méta-modèles hétérogènes. La correspondance permet de générer le script de transformation qui permet un gain de temps dans l'écriture des scénarios de transformations.

L'architecte a besoin d'aide pour construire l'alignement des modèles du SI, l'approche à l'aide de facettes permet de définir et enregistrer un modèle d'alignement. Pour aller plus loin avec le tissage manuel, nous avons élaboré des stratégies de base sur une simple comparaison des noms et des types. Nous avons appliqué cette stratégie quand les modèles ont respecté des règles d'écriture : conventions de nom pour les concepts, séparation entre donnée et traitement. Cependant, des outils plus performants sont nécessaires, par exemple le cadriciel *EMFCompare* permet de comparer les instances de modèles différents, cela peut être utile pour retrouver des concepts à travers des modèles de différents points de vue. Une solution envisageable est d'utiliser un algorithme basé sur les ontologies pour calculer un facteur de similitude qui combine les noms de concept, les types, les attributs, mais aussi les relations. Par exemple, à partir de notre définition de l'alignement illustré par la figure 4.6, si les concepts respectifs du lien <*BPM Activity - App Service*> manipulent des données, alors un lien <*BPM DataObject - App DataObject*> est éligible. Antunes *et al.* [6] ont expérimenté cette approche dans le contexte IAE. Des ontologies web structurées[8] (OWL) sont extraites depuis les différents modèles de point de vue Archimate, ces ontologies permettent d'analyser et de comparer les modèles à l'aide de requêtes. Travailler sur les ontologies permet de filtrer les détails inutiles pour se focaliser sur les informations pertinentes. Nous suivons d'ailleurs le même principe avec l'abstraction du code source vers le modèle applicatif.

[8] *Web Ontology Language*

Conclusion

Nous avons proposé une approche pragmatique au problème d'alignement des points de vue métier et applicatif s'insérant dans la démarche d'urbanisation des architectures d'entreprise. Nous avons proposé un processus complet par rapprochement successif en plusieurs étapes d'abstraction et de concrétisation jusqu'à l'alignement final. Notre approche utilise des techniques d'ingénierie des modèles (modélisation, transformation, tissage), et est instrumentée dans le cadre d'Eclipse EMF. Le processus a été expérimenté sur trois cas réels d'entreprises, permettant d'éprouver la viabilité de notre approche et sa compatibilité avec des supports hétérogènes (modèles, code source, référentiel d'entreprise, documentation).

Notre proposition est outillée de bout en bout. Nous avons utilisé les outils qui existent, et nous avons développé les outils qui manquaient, de façon à avoir une chaîne complète pour l'alignement à base de modèles.

Une fois l'alignement réalisé, le prochain objectif est d'analyser la valeur de l'alignement. Le but est d'assister l'architecte dans les décisions de transformations à appliquer sur le système d'information.

6

Assister l'évolution du SI par le résultat de l'alignement

Sommaire

Introduction

Dans les chapitres précédents, nous avons défini l'alignement du SI à partir de méta-modèles qui détaillent les points de vue processus métier, fonctionnel et applicatif. L'alignement a été implémenté à l'aide d'outillage faisant appel à l'ingénierie dirigée par les modèles, puis mis à l'épreuve par des expérimentations tirées de cas réels d'entreprises. Nous souhaitons à présent **exploiter le résultat de l'alignement** et réaliser des analyses dans le but d'**assister les décideurs lors de l'évolution du SI**.

Notre première approche d'exploitation du résultat de l'alignement était orientée par les métriques, elle visait à quantifier le taux de l'alignement par un indicateur global (un pourcentage) entre 0 et 1. Pour répondre à la question suivante : le SI est-il mal aligné (0), moyennement aligné (0.5) ou bien aligné (1) ? Nous nous sommes rendus compte que cette recherche de résultat était trop complexe, car le calcul est adhérent à la population statistique : le nombre d'éléments à aligner dans chaque point de vue. Autrement dit, il est difficile d'en apprécier la valeur globale qui masque des disparités de situation. Avoir des indicateurs plus locaux est plus pertinent à la fois pour analyser quantitativement et qualitativement la situation pour détecter les points d'amélioration. Ainsi, nous avons changé d'approche pour proposer plusieurs taux d'alignement notamment par couple de point de vue, et nous dissocions deux types d'instances de concept clairement distincts dans leur structure : les traitements et les données.

Les véritables questions sont d'identifier quels éléments d'un point de vue sont alignés ; et surtout quels éléments d'un point de vue ne sont pas alignés. Le résultat de l'indicateur est donc un nombre et une liste d'éléments plutôt qu'un simple score qui ne permet en aucun cas de **détecter les "désalignements"** (éléments incorrectement alignés ou non alignés) pour identifier les problèmes du SI. À partir de ces résultats, l'architecte va pouvoir procéder soit à un réalignement si la cartographie n'est ni complète ni à jour, soit à des actions sur le SI comme la suppression d'un composant applicatif qui n'est plus utilisé par le métier ; l'ajout de traitements techniques ; la restructuration de processus métier ; ou l'application de règles d'urbanisme.

La détection des éléments actifs ou passifs du SI n'est pas le seul indicateur que nous pouvons obtenir de l'exploitation de l'alignement. L'architecte d'entreprise souhaite mesurer l'impact d'un élément sur son voisinage : si l'élément est modifié ou supprimé, quelles en seraient les conséquences ? Nous souhaitons répondre à ces questions en permettant de dénombrer les **couplages** et de **visualiser les dépendances**.

Un certain nombre d'indicateurs considérés comme les plus pertinents sont rassemblés dans un tableau de bord en section 6.2. Nous proposons une extension à notre définition de l'alignement en section 6.1.2 pour réaliser l'analyse des éléments à aligner. Puis, nous détaillons les requêtes nécessaires au calcul des indicateurs du tableau de bord en section 6.3. Enfin, nous dédions la section 6.4 à la visualisation des dépendances à l'aide de matrices de dépendances.

L'importance pour les décideurs est d'obtenir une photo du SI à l'aide d'indicateurs utiles au pilotage.

6.1 Les éléments des modèles pour le calcul des indicateurs d'alignement

Nous considérons que les éléments des modèles exploités durant la phase d'alignement suivent un cycle de vie : à aligner, aligné ou non aligné. D'autre part, certains éléments ne sont pas alignables et doivent être mis à part du champ d'étude (cf. section 6.1.2). Nous définissons un nouveau type d'élément "*nonalignable*", dont nous donnons une solution d'implémentation à l'aide de EMF Facet.

6.1.1 Typologie des éléments

Nous définissons les différents types pour ranger les éléments composant l'alignement entre les modèles :

- **Éléments hors définition** ce sont tous les éléments du modèle qui ne font pas partie de la définition de l'alignement ;
- **Éléments à aligner** ce sont tous les éléments du modèle qui font partie de la définition de l'alignement : concepts de traitements et de données.
  - **Éléments non alignables** ce sont les éléments à aligner que l'architecte souhaite masquer, cf. section 6.1.2 ;
  - **Éléments alignables** ce sont les éléments à aligner, sans les éléments non alignables.
    - **Éléments alignés** ce sont tous les éléments alignables du modèle qui ont été alignés ;
    - **Éléments non alignés** ce sont tous les éléments alignables du modèle qui ne sont pas alignés.

Nous donnons un exemple d'alignement de deux modèles (App et BPM) en figure 6.1 pour illustrer les différentes catégories d'éléments de l'alignement.

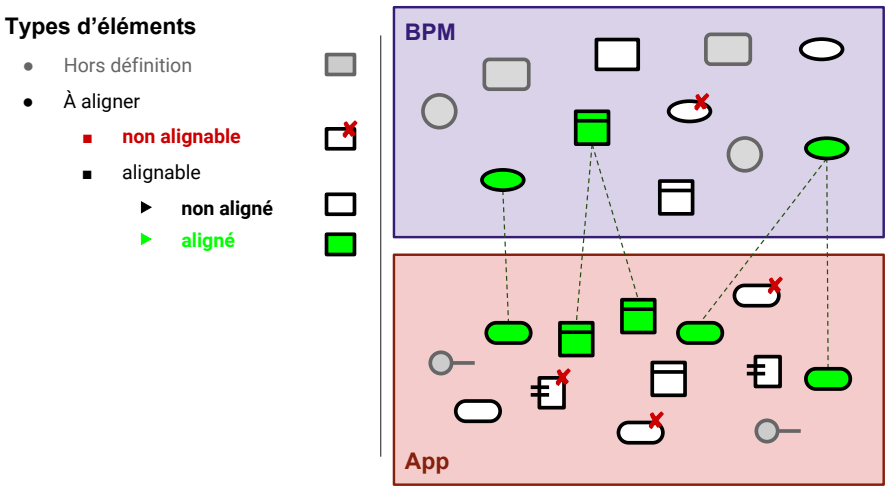


FIGURE 6.1 – Exemple d'alignement entre deux modèles BPM et App

Les éléments grisés sont liés aux éléments de chaque modèle par la définition de leur méta-modèle respectif et ne font pas partie de la définition de l'alignement, tandis que tous les autres éléments font partie de la définition et sont à aligner. Les associations sont toutes bi-directionnelles et peuvent être n-aires, par exemple sur la figure 6.1 un objet

de donnée du modèle BPM est associé à deux objets de données du modèle App. Les associations bi-directionnelles peuvent être décomposées en deux associations orientées : BPM vers App, et App vers BPM. Cette décomposition permet d'effectuer les requêtes par navigation en partant du concept d'un point de vue pour atteindre le ou les concepts d'un autre point de vue.

Les différents types d'éléments sont utilisés pour calculer les différents indicateurs du tableau de bord. Nous présentons les indicateurs composant le tableau de bord sous la forme d'une maquette dans la section 6.2.

6.1.2 Identification des éléments non alignables

Notre définition de l'alignement (cf. section 4.1) est insuffisante pour réaliser une analyse dans le but de suivre l'évolution du SI. En effet, certains éléments des modèles à aligner ont une raison de ne pas être alignés.

Nouveauté : l'élément est nouveau et n'a pas d'élément similaire dans un autre modèle. Par exemple, l'ajout d'une nouvelle activité non encore implémentée dans les applications.

Déprécié : l'élément n'est plus utilisé par le SI, mais encore présent. Par exemple, un composant applicatif a été remplacé par un autre est en attente d'être supprimé de la configuration.

Supprimé : l'élément n'existe plus dans le SI, mais nous souhaitons en garder une trace pour maintenir un historique des versions.

Pour éviter de biaiser la lecture des indicateurs par ces éléments particuliers, nous souhaitons les masquer dans les calculs des indicateurs. Nous souhaitons donc ajouter un marqueur sur les éléments à indiquer comme non alignables. Pour mettre en œuvre cette variante, nous proposons une nouvelle facette qui ajoute au concept de traitement et de donnée deux attributs :

notAlignable il s'agit d'un attribut de type booléen, la valeur *true* (vrai) signifie que l'élément n'est pas alignable.

state il s'agit d'un attribut de type *string* (texte), la valeur est libre, elle permet de mettre un motif par l'architecte pour expliquer pourquoi l'élément est non alignable. Ce motif peut être un de ceux que nous proposons précédemment : nouveauté, déprécié, supprimé.

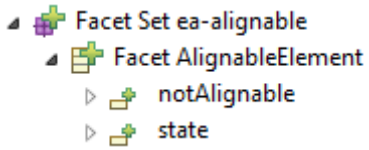


FIGURE 6.2 – Nouveau FacetSet avec une Facet pour ajouter les deux attributs

À l'aide de EMF Facet, nous créons une nouvelle FacetSet "*AlignableElement*" en figure 6.2 contenant une Facet avec les deux attributs : *alignable* et *state*.

Puis, nous modifions notre implémentation de la définition de l'alignement réalisée avec EMF Facet (cf. section 4.3.3). Pour rappel, chaque Facet de notre définition correspond à l'extension des concepts à aligner (traitements et données) : composant applicatif,

service, objet de donnée, fonctionnalité, etc. À chacune de ces Facets, en figure 6.3 nous étendons la Facet "*AlignableElement*" pour ajouter à chaque concept à aligner les deux nouveaux attributs.

Les nouveaux attributs apparaissent dans les modèles et des valeurs peuvent être affectées. Par exemple, le composant applicatif "S-ADM-SAIDIV01" en figure 6.4.

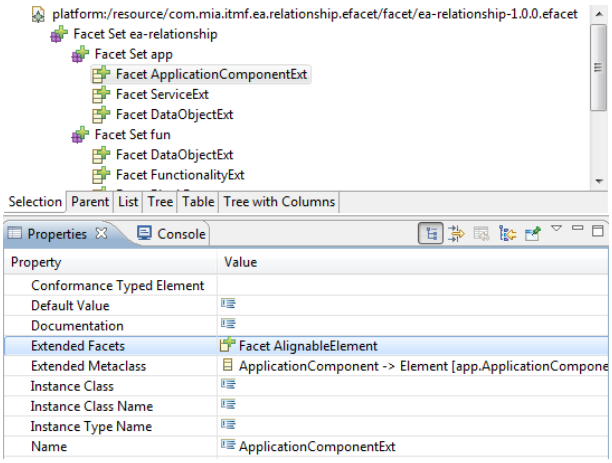


FIGURE 6.3 – Extension au FacetSet définissant l'alignement

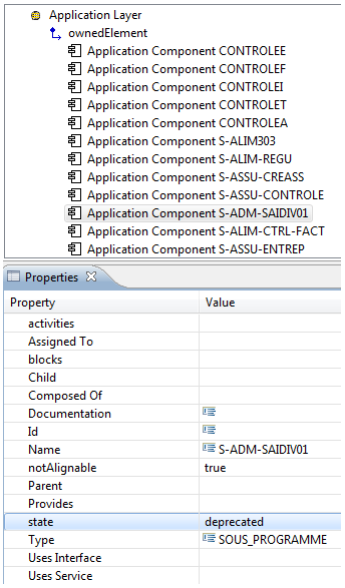


FIGURE 6.4 – Exemple de composant applicatif non alignable

Cette nouvelle facette nous permet d'ajouter un indicateur utile à l'analyse de l'alignement dont l'objectif est de spécifier qu'un élément est non alignable. La prochaine section spécifie des requêtes pour calculer les indicateurs du tableau de bord. Les requêtes utilisent le nouveau marqueur pour filtrer les éléments non alignables.

6.2 Le tableau de bord de l'alignement

Nous avons conçu un tableau de bord de l'alignement destiné à l'architecte d'entreprise. Le tableau de bord est composé d'indicateurs et de diagrammes pour surveiller l'évolution de l'alignement et aider l'architecte à comprendre l'état du SI. Noter qu'on peut imaginer des tableaux de bord spécifiques à chaque acteur.

6.2.1 Maquette

Nous présentons le tableau de bord à l'aide d'une maquette non fonctionnelle, cependant toutes les requêtes qui sont nécessaires pour implémenter ce tableau de bord sont définies en section 6.3.

La maquette est composée de différents diagrammes : secteurs, barres, barres empilées et courbes. Le diagramme en secteurs permet de visualiser des proportions de couleurs différentes entre chaque classe d'un ensemble. Le diagramme en barres permet de comparer différentes classes par la hauteur de chaque barre. Le diagramme en barres empilées représente une dimension supplémentaire pour chaque classe en additionnant la valeur de cette dimension. Le diagramme en courbes permet de visualiser l'évolution de plusieurs indicateurs en fonction du temps.

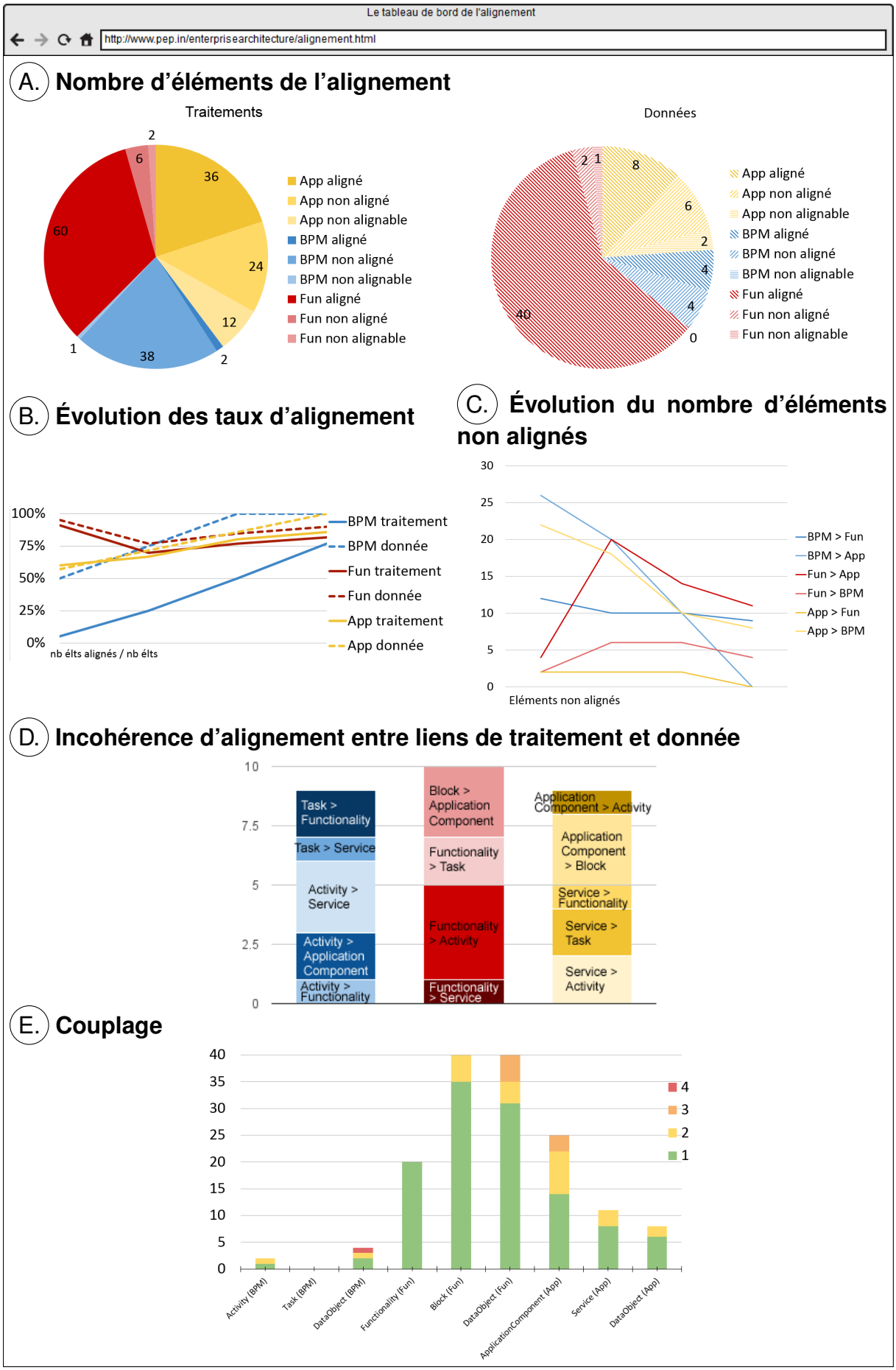


FIGURE 6.5 – Maquette du tableau de bord de l'alignement

Nous détaillons les indicateurs pour chaque diagramme de la maquette en figure 6.5 :

(A.) **Nombre d'éléments de l'alignement** Pour chaque point de vue (processus, fonctionnel et applicatif), on représente dans des secteurs différents du diagramme circulaire : le nombre d'éléments alignés, le nombre d'éléments non alignés et le nombre d'éléments non alignables. L'ensemble de ces trois catégories représente l'ensemble des éléments correspondant à la définition de l'alignement. Les éléments des diagrammes sont classés en deux parties distinctes : traitements et données. En effet, traitements et données ne sont ni comparables ni "unifiables". Ce diagramme permet de visualiser la proportion des éléments composant l'alignement des trois points de vue. Par exemple sur la maquette, 36 traitements de App sont alignés (orange foncé), 1 traitement de BPM est non alignable, ou encore 2 objets de données de Fun sont non alignés. Les calculs sont détaillés en section 6.3.1.

(B.) **Évolution des taux d'alignement** Le taux d'alignement est le nombre d'éléments alignés divisé par le nombre d'éléments alignables. Chaque classe du diagramme correspond aux éléments de traitement ou de donnée, des trois points de vue. Ce diagramme permet de connaître l'état des travaux d'alignement. Plus la valeur se rapproche de 1, plus l'alignement est complet. Par exemple sur la maquette, le taux d'alignement des traitements de BPM augmente à chaque itération (courbe pleine en bleu) et passe d'un taux proche de 0 à 75 %, ce qui signifie que trois quarts des éléments de traitements de BPM sont alignés. Les calculs sont détaillés en section 6.3.2.

(C.) **Évolution du nombre d'éléments non alignés** Le nombre d'éléments non alignés est un calcul plus complexe dont les requêtes sont détaillées en section 6.3.3. Le calcul est réalisé par couple de point de vue et permet de savoir combien d'éléments d'un point de vue ne sont pas alignés avec un autre. Par exemple sur la maquette, le nombre d'éléments de App non alignés avec BPM passe de 22 à 8 (courbe jaune clair).

(D.) **Incohérence d'alignement** Ce diagramme couvre tous les liens de traitements de la définition de l'alignement et vérifie la cohérence entre les liens de traitements et ceux de données. Pour chaque instance d'un concept de traitement T manipulant des données D et alignée avec une instance de concept de traitement T'. Si la donnée D est également alignée à une donnée D', il est vérifié que D' est manipulée par le traitement symétrique T' aligné à T. Par exemple sur la maquette, deux alignements entre des activités de BPM avec des composants applicatifs de App sont incohérents (empilement de barres bleues). Les calculs sont détaillés en section 6.3.4.

(E.) **Couplage** Tous les concepts de traitements ou de données de la définition de l'alignement sont représentés. Ce calcul permet de visualiser pour chaque concept le nombre d'instances par valeur de couplage : à combien d'instances cibles l'instance source est-elle alignée ? Par exemple sur la maquette 6.5, une activité de BPM est couplée avec une autre instance (en vert), et une activité est couplée avec deux autres instances (en orange). Plus la valeur du couplage est importante plus la complexité est importante. Ainsi, il sera plus difficile d'appliquer des modifications au SI, l'impact de modifier un élément étant plus important. Les requêtes sont détaillées en section 6.3.5, et pour visualiser plus finement le couplage, nous proposons l'analyse par matrice de dépendance (DSM) en section 6.4.

Plusieurs de ces indicateurs peuvent avoir deux présentations : instantanée ou temporelle. La présentation instantanée permet de se rendre compte de l'état de l'alignement à un instant t, tandis que la présentation temporelle permet de suivre l'évolution du processus d'alignement et les éléments qui le composent. Nous détaillons la signification de ces indicateurs en fonction de la présentation instantanée ou temporelle dans la section suivante.

6.2.2 Variante des indicateurs en fonction du temps

Dans le cas où le résultat d'alignement est enregistré à chaque itération d'évolution du SI, il est possible de présenter les classes correspondantes aux diagrammes en secteurs ou en histogramme à l'aide de diagrammes en courbes avec une dimension temporelle. Un bouton pourrait être ajouté au tableau de bord (cf. figure 6.5) pour basculer de la présentation instantanée à temporelle ou vice-versa.

(A.) Évolution du nombre d'éléments de l'alignement

Dans la maquette en figure 6.5, les deux diagrammes en secteurs représentent une analyse instantanée du nombre d'éléments de l'alignement. Nous proposons une version de l'historique en fonction du temps sur la figure 6.6.

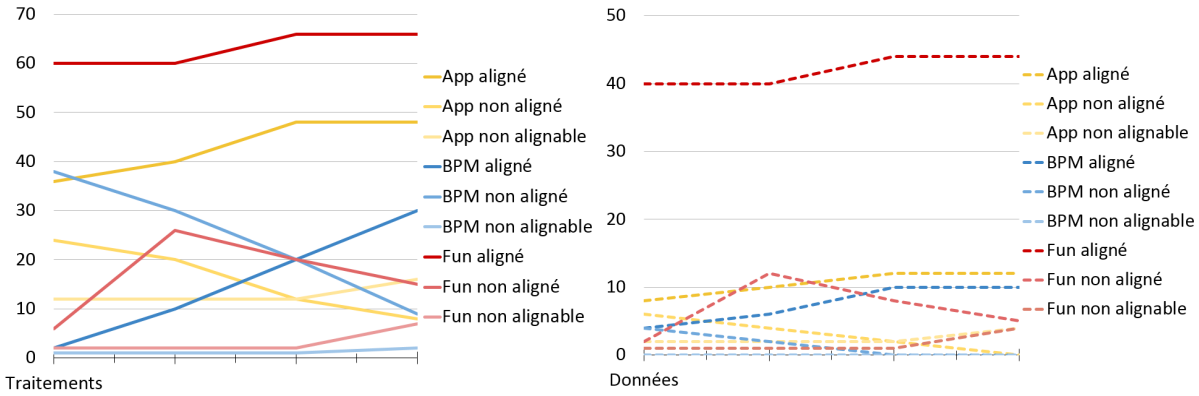


FIGURE 6.6 – Nombre d'éléments : traitements et données

La représentation temporelle permet d'interpréter l'évolution des éléments des modèles et de l'alignement. Si les modèles possèdent le même nombre d'éléments entre deux itérations, lors de l'alignement de nouveaux éléments, la courbe d'éléments alignés croît proportionnellement à la diminution de la courbe d'éléments non alignés. Si des éléments nouveaux sont ajoutés dans les modèles entre deux itérations sans nouvel alignement, la courbe d'éléments non alignés croît, tandis que la courbe d'éléments alignés reste constante. Enfin, la courbe d'éléments non alignables varie en fonction de l'affectation de l'indicateur non alignable (cf. section 6.1.2). Par exemple sur la figure, la courbe d'éléments de traitements non alignables de BPM ne varie presque pas, ainsi la courbe de BPM d'éléments de traitements alignés augmente et la courbe d'éléments de traitements non alignés diminue proportionnellement. Autre exemple, les courbes d'éléments de données non alignables et alignés de Fun ne varient pas lors des trois premières itérations, or la courbe d'éléments de données non alignés de Fun augmente, cela signifie que le nombre d'éléments alignables (et à aligner) dans le modèle a augmenté.

(B.) Taux d'alignement

Dans la maquette, le diagramme en courbe représente une analyse temporelle du taux d'alignement pour chaque point de vue. Une augmentation signifie que l'alignement devient plus complet. Tandis qu'un taux d'alignement à la baisse peut signifier l'ajout de nouveaux éléments à aligner.

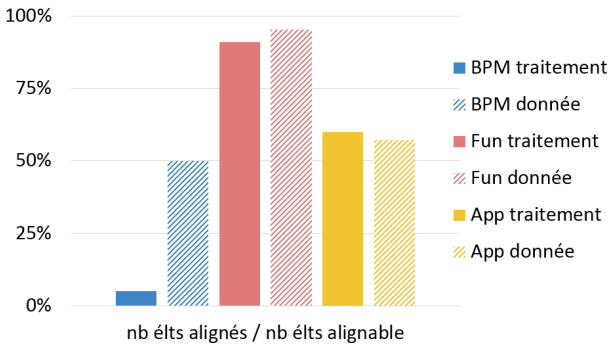


FIGURE 6.7 – Taux d'alignement

Nous proposons la version instantanée en figure 6.7. Cette version permet de comparer plus facilement sur quels points de vue le travail d'alignement a été effectué. Par exemple sur la figure, le taux d'éléments de traitements et de données de Fun avoisine 100%.

(C.) Nombre d'éléments non alignés

Dans la maquette, le diagramme en courbe représente une analyse temporelle du nombre d'éléments non alignés.

Une évolution à la hausse du nombre d'éléments non alignés permet d'interpréter soit une augmentation du nombre d'éléments alignables dans les modèles, soit un désalignement d'éléments. Pour vérifier si le nombre d'éléments a augmenté, il est possible de regarder le diagramme correspondant aux nombres d'éléments de l'alignement.

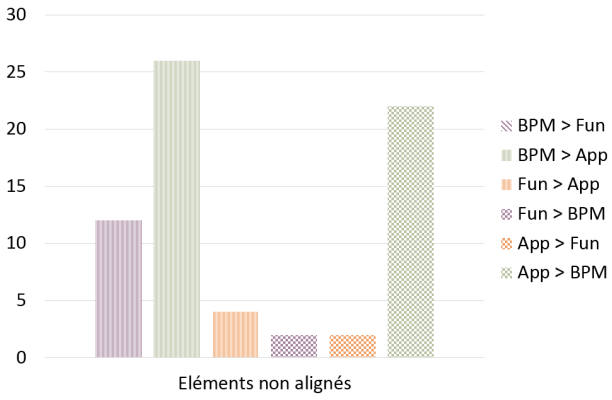


FIGURE 6.8 – Nombre d'éléments non alignés entre deux points de vue

Une évolution à la baisse correspond soit à la diminution du nombre d'éléments alignables (suppression ou ajout de l'indicateur non alignable à des éléments, cf. section 6.1.2) ; soit à l'augmentation du nombre d'éléments alignés (vérifiable avec le diagramme de taux d'alignement).

Nous proposons une version instantanée en figure 6.8. Cette version permet de comparer plus facilement entre quels points de vue les alignements ne sont pas effectués. Par exemple sur la figure, deux éléments de App sont non alignés avec Fun, et quatre éléments de Fun sont non alignés avec App.

Après avoir présenté les différents diagrammes de contrôle de l'alignement, nous développons dans les sections suivantes les différentes techniques permettant d'implanter le tableau de bord. La section 6.1.2 détaille le mécanisme pour marquer un élément comme non alignable, puis la section 6.3 détaille les différentes requêtes OCL pour réaliser les calculs.

6.3 Une analyse de l'alignement par requêtes OCL

Cette section a pour but de tirer bénéfice de l'alignement à l'aide de l'outil EMF Facet, et de l'outillage de requête que nous avons développé en section 4.3.5. L'alignement à l'aide de la technique par facettes permet une navigation à travers les modèles via les références entre instances de concepts. Ainsi, il est possible de passer d'une instance de concept d'un modèle vers une autre grâce au lien d'alignement. Notre console OCL permet d'écrire des requêtes qui exploitent cette navigation. Grâce au langage de requête OCL, nous pouvons obtenir des résultats selon les critères que l'on souhaite à l'aide d'opérations de projection, de sélection et de jointure.

Dans la suite, nous proposons des requêtes classées dans différentes catégories pour assister l'architecte dans l'interprétation de l'alignement des différents points de vue : processus métier, fonctionnel et applicatif. Le résultat des requêtes permet à l'architecte de décider des évolutions à appliquer au SI. Les catégories de requêtes sont les suivantes :

- **Alignement / Non alignement** : permet d'analyser quels sont les éléments de chaque point de vue qui sont ou non alignés ;
- **Cohérence / incohérence** : permet de vérifier si des concepts alignés par des liens par les traitements sont également alignés par des liens par données cohérents ou incohérents ;
- **Couplage** : permet de dénombrer le nombre d'instances de concepts alignées entre elles.

6.3.1 Nombre d'éléments de l'alignement

Un premier jeu de requêtes ci-dessous permet de calculer le nombre d'éléments de chaque instance de méta-modèle (BPM, Fun et App) selon différents aspects des concepts de type **traitement** et **donnée** contenus dans les modèles.

TABLE 6.1 – Guide des requêtes de calcul du nombre d'éléments de l'alignement

		N° requête - listing		
		BPM	Fun	App
Alignés	Traitements	6.1	6.7	6.13
	Données	6.2	6.8	6.14
Non alignables	Traitements	6.3	6.9	6.15
	Données	6.4	6.10	6.16
Non alignés	Traitements	6.5	6.11	6.17
	Données	6.6	6.12	6.18

alignés Le nombre d'instances de concepts alignées se calcule en dénombrant les références non vides pointant vers d'autres instances de concepts de même type ;

non alignables Le nombre d'instances de concepts non alignables se calcule en dénombrant les instances de concepts avec l'attribut marqueur "*notAlignable*" de valeur **vrai**.

non alignés Le nombre d'instances de concepts non alignées se calcule en dénombrant le nombre de références vides auquel on soustrait le nombre d'instances de mêmes concepts non alignables.

BPM

Les requêtes suivantes concernent les instances de concepts de traitements et données du méta-modèle BPM.

Traitements alignés La requête 6.1 dénombre les références non vides des activités et tâches du modèle BPM vers les composants applicatifs et services du modèle App et les fonctionnalités du modèle Fun.

Listing 6.1 – BPM traitements alignés

```
BpmProcessAlignedSize: context BusinessProcessLayer
bpm::Activity.allInstances()->select(applicationComponents->notEmpty() or
    services->notEmpty() or functionalities->notEmpty())->size() +
bpm::Task.allInstances()->select(services->notEmpty())->size()
```

Données alignées La requête 6.2 dénombre les références non vides des objets de données du modèle BPM vers les objets de données des modèles App et Fun.

Listing 6.2 – BPM données alignées

```
BpmDataAlignedSize: context BusinessProcessLayer
bpm::DataObject.allInstances()->select(dataObjectsFun->notEmpty() or
    dataObjectsApp->notEmpty())->size()
```

Traitements non alignables La requête 6.3 dénombre les activités et tâches du modèle BPM dont l'attribut "*notAlignable*" est de valeur vrai.

Listing 6.3 – BPM traitements non alignables

```
BpmProcessNotAlignableSize: context BusinessProcessLayer
bpm::Activity.allInstances()->select(notAlignable)->size() +
bpm::Task.allInstances()->select(notAlignable)->size()
```

Données non alignables La requête 6.4 dénombre les objets de données du modèle BPM dont l'attribut "*notAlignable*" est de valeur vrai.

Listing 6.4 – BPM données non alignables

```
BpmDataNotAlignableSize: context BusinessProcessLayer:
bpm::DataObject.allInstances()->select(notAlignable)->size()
```

Traitements non alignés La requête 6.5 dénombre les références vides des activités et tâches du modèle BPM vers les composants applicatifs et services du modèle App et les fonctionnalités du modèle Fun, auquel on soustrait le résultat de *BpmProcessNotAlignableSize*.

Listing 6.5 – BPM traitements non alignés

```
BpmProcessNotAlignedSize: context BusinessProcessLayer
bpm::Activity.allInstances()->select(applicationComponents->isEmpty() and
    services->isEmpty() and functionalities->isEmpty())->size() +
bpm::Task.allInstances()->select(services->isEmpty())->size()
- BpmProcessNotAlignableSize
```

Données non alignées La requête 6.6 dénombre les références vides des objets de données du modèle BPM vers les objets de données des modèles App et Fun, auquel on soustrait le résultat de la requête *BpmDataNotAlignableSize*.

Listing 6.6 – BPM données non alignées

```
BpmDataNotAlignedSize: context BusinessProcessLayer
bpm::DataObject.allInstances()->select(dataObjectsFun->isEmpty() and
    dataObjectsApp->isEmpty())->size()
- BpmDataNotAlignableSize
```

Fun

Les requêtes suivantes concernent les instances de concepts de traitements et données du méta-modèle Fun.

Traitements alignés La requête 6.7 dénombre les références non vides des fonctionnalités et blocs du modèle Fun vers les composants applicatifs et services du modèle App et les activités du modèle BPM.

Listing 6.7 – Fun traitements alignés

```
FunProcessAlignedSize: context FunctionalLayer
fun::Functionality.allInstances()->select(activities->notEmpty() or
   services->notEmpty())->size() +
fun::Block.allInstances()->select(applicationComponents->notEmpty())->size()
```

Données alignées La requête 6.8 dénombre les références non vides des objets de données du modèle Fun vers les objets de données des modèles App et BPM.

Listing 6.8 – Fun données alignées

```
FunDataAlignedSize: context FunctionalLayer
fun::DataObject.allInstances()->select(dataObjectsBpm->notEmpty() or
   dataObjectsApp->notEmpty())->size()
```

Traitements non alignables La requête 6.9 dénombre les fonctionnalités et blocs du modèle Fun dont l'attribut "*notAlignable*" est de valeur vrai.

Listing 6.9 – Fun traitements non alignables

```
FunProcessNotAlignableSize: context FunctionalLayer
fun::Functionality.allInstances()->select(notAlignable)->size() +
fun::Block.allInstances()->select(notAlignable)->size()
```

Données non alignables La requête 6.10 dénombre les objets de données du modèle Fun dont l'attribut "*notAlignable*" est de valeur vrai.

Listing 6.10 – Fun données non alignables

```
FunDataNotAlignableSize: context FunctionalLayer
fun::DataObject.allInstances()->select(notAlignable)->size()
```

Traitements non alignés La requête 6.11 dénombre les références vides des fonctionnalités et blocs du modèle Fun vers les composants applicatifs et services du modèle App et les activités du modèle BPM, auquel on soustrait le résultat de la requête *FunProcessNotAlignableSize*.

Listing 6.11 – Fun traitements non alignés

```
FunProcessNotAlignedSize: context FunctionalLayer
fun::Functionality.allInstances()->select(activities->isEmpty() and
   services->isEmpty())->size() +
fun::Block.allInstances()->select(applicationComponents->isEmpty())->size()
- FunProcessNotAlignableSize
```

Données non alignées La requête 6.12 dénombre les références vides des objets de données du modèle Fun vers les objets de données des modèles App et BPM, auquel on soustrait le résultat de la requête *FunDataNotAlignableSize*.

Listing 6.12 – Fun données non alignées

```
FunDataNotAlignedSize: context FunctionalLayer
fun::DataObject.allInstances()->select(dataObjectsBpm->isEmpty() and
   dataObjectsApp->isEmpty())->size()
- FunDataNotAlignableSize
```

App

Les requêtes suivantes concernent les instances de concepts de traitements et données du méta-modèle App.

Traitements alignés La requête 6.13 dénombre les références non vides des composants applicatifs et service du modèle App vers les activités et tâches du modèle BPM, et les fonctionnalités et blocs du modèle Fun.

Listing 6.13 – App traitements alignés

```
AppProcessAlignedSize: context ApplicationLayer
app::ApplicationComponent.allInstances()->select(activities->notEmpty() or
 blocks->notEmpty())->size() + app::Service.allInstances()->select(tasks->
 notEmpty() or activities->notEmpty() or functionalities->notEmpty())->size()
```

Données alignées La requête 6.14 dénombre les références non vides des objets de données du modèle App vers les objets de données des modèles Fun et BPM.

Listing 6.14 – App données alignées

```
AppDataAlignedSize: context ApplicationLayer
app::DataObject.allInstances()->select(dataObjectsBpm->notEmpty() or
    dataObjectsFun->notEmpty())->size()
```

Traitements non alignables La requête 6.15 dénombre les composants applicatifs et services du modèle App dont l'attribut "*notAlignable*" est de valeur vrai.

Listing 6.15 – App traitements non alignables

```
AppProcessNotAlignableSize: context ApplicationLayer
app::ApplicationComponent.allInstances()->select(notAlignable)->size()
   + app::Service.allInstances()->select(notAlignable)->size()
```

Données non alignables La requête 6.16 dénombre les objets de données du modèle App dont l'attribut "*notAlignable*" est de valeur vrai.

Listing 6.16 – App données non alignables

```
AppDataNotAlignableSize: context ApplicationLayer
app::DataObject.allInstances()->select(notAlignable)->size()
```

Traitements non alignés La requête 6.17 dénombre les références vides des composants applicatifs et services du modèle App vers les activités, tâches, fonctionnalités et blocs des modèles BPM et Fun, auquel on soustrait le résultat de la requête *AppProcessNotAlignableSize*.

Listing 6.17 – App traitements non alignés

```
AppProcessNotAlignedSize: context ApplicationLayer
app::ApplicationComponent.allInstances()->select(activities->isEmpty() and
    blocks->isEmpty())->size() + app::Service.allInstances()->select(tasks->
    isEmpty() and activities->isEmpty() and functionalities->isEmpty())->
    size() - AppProcessNotAlignableSize
```

Données non alignées La requête 6.18 dénombre les références vides des objets de données du modèle App vers les objets de données des modèles Fun et BPM, auquel on soustrait le résultat de la requête *AppDataNotAlignableSize*.

Listing 6.18 – App données non alignées

```
AppDataNotAlignedSize: context ApplicationLayer
app::DataObject.allInstances()->select(dataObjectsBpm->isEmpty() and
    dataObjectsFun->isEmpty())->size() - AppDataNotAlignableSize
```

Ce lot de requêtes permet de calculer le nombre d'éléments qui composent l'alignement selon le point de vue et le type de lien : traitement ou donnée. La prochaine section utilise le résultat de ces requêtes pour calculer un taux d'alignement.

6.3.2 Taux d'alignement

Le taux d'alignement permet de connaître l'avancement des travaux d'alignement. Une valeur de taux à 0, signifie qu'aucun alignement n'a été réalisé. Une valeur de taux à 1, signifie que l'alignement est complet.

TABLE 6.2 – Guide des requêtes de calcul du taux d'alignement

	N° requête - listing		
	BPM	Fun	App
Traitements	6.19	6.21	6.23
Données	6.20	6.22	6.24

Les requêtes suivantes utilisent le résultat des requêtes de calcul du nombre d'éléments de l'alignement de la section 6.3.1.

BPM

Traitements La requête 6.19 calcule le nombre d'instances de concepts de traitement alignées sur le nombre d'instances de concepts de traitement alignées et non alignées de BPM.

Listing 6.19 – Taux d'alignement des traitements de BPM

```
BPMProcessAlignedRate: context BusinessProcessLayer
BpmProcessAlignedSize / (BpmProcessAlignedSize + BpmProcessNotAlignedSize)
```

Données La requête 6.20 calcule le nombre d'instances de concepts de donnée alignées sur le nombre d'instances de concepts de donnée alignées et non alignées de BPM.

Listing 6.20 – Taux d'alignement des données de BPM

```
BPMDataAlignedRate: context BusinessProcessLayer
BpmDataAlignedSize / (BpmDataAlignedSize + BpmDataNotAlignedSize)
```

Fun

Traitements La requête 6.21 calcule le nombre d'instances de concepts de traitement alignées sur le nombre d'instances de concepts de traitement alignées et non alignées de Fun.

Listing 6.21 – Taux d'alignement des traitements de Fun

```
FunProcessAlignedRate: context FunctionalLayer
FunProcessAlignedSize / (FunProcessAlignedSize + FunProcessNotAlignedSize)
```

Données La requête 6.22 calcule le nombre d'instances de concepts de donnée alignées sur le nombre d'instances de concepts de donnée alignées et non alignées de Fun.

Listing 6.22 – Taux d'alignement des données de Fun

```
FunDataAlignedRate: context FunctionalLayer
FunDataAlignedSize / (FunDataAlignedSize + FunDataNotAlignedSize)
```

App

Traitements La requête 6.23 calcule le nombre d'instances de concepts de traitement alignées sur le nombre d'instances de concepts de traitement alignées et non alignées de App.

Listing 6.23 – Taux d'alignement des traitements de App

```
AppProcessAlignedRate: context ApplicationLayer
AppProcessAlignedSize / (AppProcessAlignedSize + AppProcessNotAlignedSize)
```

Données La requête 6.24 calcule le nombre d'instances de concepts de donnée alignées sur le nombre d'instances de concepts de donnée alignées et non alignées de App.

Listing 6.24 – Taux d'alignement des données de App

```
AppDataAlignedRate: context ApplicationLayer
AppDataAlignedSize / (AppDataAlignedSize + AppDataNotAlignedSize)
```

Afin d'améliorer le taux d'alignement, il est nécessaire d'identifier quels sont les éléments non alignés. Nous proposons des requêtes pour effectuer cette analyse dans la section suivante.

6.3.3 Liste des éléments non alignés

Pour détecter les éléments non alignés de chaque point de vue, nous définissons une requête pour chaque couple de points de vue : processus métier, fonctionnel et applicatif. Chaque requête liste les instances des concepts qui ne sont pas alignées vis-à-vis d'un autre point de vue. Dans le tableau 6.3 les requêtes sont symétriques deux à deux, puisque chaque lien d'alignement entre les points de vue est bi-directionnel. Ainsi 6.25 est symétrique avec 6.26, 6.27 avec 6.28, et 6.29 avec 6.30.

TABLE 6.3 – Guide des requêtes de liste d'éléments non alignés

	N° requête - listing		
↱	BPM	Fun	App
BPM		6.25	6.27
Fun	6.26		6.29
App	6.28	6.30	

Les instances de BPM non alignées avec les instances de Fun

La requête 6.25 répond aux questions suivantes :

- Toutes les activités sont-elles réalisées par une ou plusieurs fonctionnalités ?
- Tous les objets de données sont-ils représentés par un ou plusieurs objets de données fonctionnels ?

La requête 6.25 liste toutes les instances du méta-modèle BPM qui ne sont pas alignées avec les instances du méta-modèle Fun :

- les activités et les tâches non alignées avec une ou plusieurs fonctionnalités ;
- les objets de données non alignés avec un ou plusieurs objets de données.

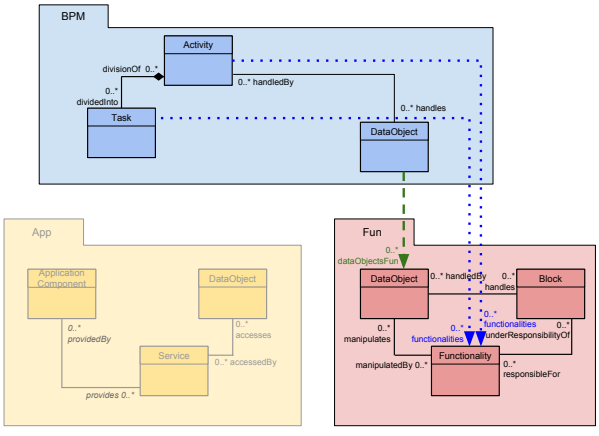


FIGURE 6.9 – Concepts de BPM alignés avec les concepts de Fun

Listing 6.25 – BPM ⇒ Fun

```
BPMFunNotAlignedList: context BusinessProcessLayer
bpm::Activity.allInstances()->select(not notAlignable and functionalities->
    isEmpty())->collect(oclAsType(ecore::EObject))->union(bpm::Task.
    allInstances()->select(not notAlignable and functionalities->isEmpty())
    ->collect(oclAsType(ecore::EObject)))->union(bpm::DataObject.
    allInstances()->select(not notAlignable and dataObjectsFun->isEmpty())->
    collect(oclAsType(ecore::EObject)))
```

Les instances de Fun non alignées avec les instances de BPM

La requête 6.26 répond aux questions suivantes :
- Toutes les fonctionnalités réalisent-elles une ou plusieurs activités ou tâches ?
- Tous les objets de données sont-ils représentés par un objet de donnée processus ?

La requête 6.26 liste toutes les instances du méta-modèle Fun qui ne sont pas alignées avec les instances du méta-modèle BPM :
- les fonctionnalités non alignées avec une ou plusieurs activités, et avec une ou plusieurs tâches ;
- les objets de données non alignés avec un ou plusieurs objets de données.

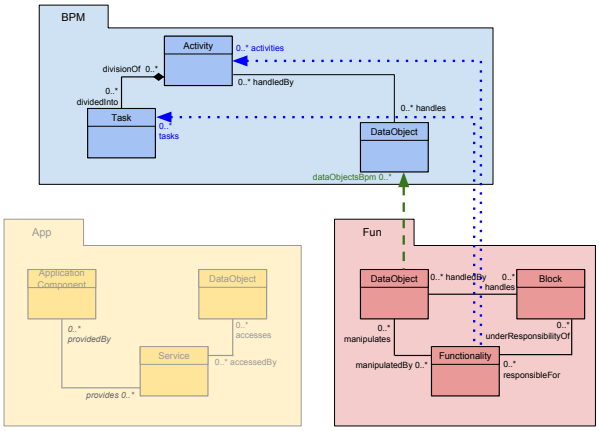


FIGURE 6.10 – Concepts de Fun alignés avec les concepts de BPM

Listing 6.26 – Fun ⇒ BPM

```
FunBPMNotAlignedList: context FunctionalLayer
fun::Functionality.allInstances()->select(not notAlignable and
    activities->isEmpty() and tasks->isEmpty())->collect(oclAsType(ecore::
    EObject))->union(fun::DataObject.allInstances()->select(notAlignable =
    false  and dataObjectsApp->isEmpty())->collect(oclAsType(ecore::EObject)))
```

Les instances de BPM non alignées avec les instances de App

La requête 6.27 répond aux questions suivantes :
- Toutes les activités sont-elles implémentées par un ou plusieurs services ou composants applicatifs ?
- Toutes les tâches sont-elles implémentées par un ou plusieurs services ?
- Tous les objets de données sont-ils implémentés par un ou plusieurs objets de donnée applicatif ?

La requête 6.27 liste toutes les instances du méta-modèle BPM qui ne sont pas alignées avec les instances du méta-modèle App :
- les activités non alignées avec un ou plusieurs services ou composants applicatifs ;
- les tâches non alignées avec un ou plusieurs services ;
- les objets de données non alignés avec un ou plusieurs objets de données.

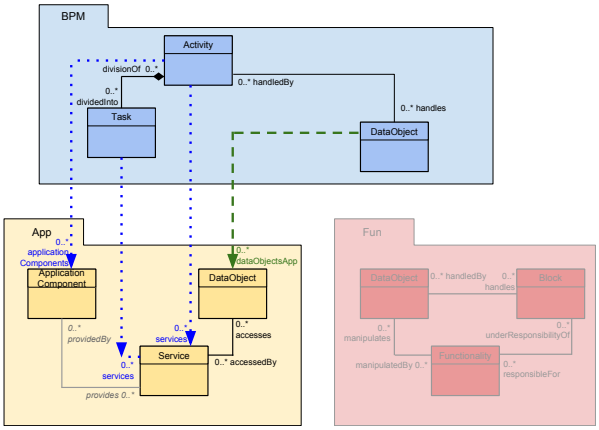


FIGURE 6.11 – Concepts de BPM alignés avec les concepts de App

Listing 6.27 – BPM ⇒ App

```
BPMAppNotAlignedList: context BusinessProcessLayer
bpm::Activity.allInstances()->select(not notAlignable and services->isEmpty
    () and applicationComponents->isEmpty())->collect(oclAsType(ecore::
    EObject))->union(bpm::Task.allInstances()->select(not notAlignable and
    services->isEmpty())->collect(oclAsType(ecore::EObject)))->union(bpm::
    DataObject.allInstances()->select(not notAlignable  and dataObjectsApp->
    isEmpty())->collect(oclAsType(ecore::EObject)))
```

Les instances de App non alignées avec les instances de BPM

La requête 6.28 répond aux questions suivantes :

- Tous les composants applicatifs implémentent-ils une ou plusieurs activités ?
- Tous les services implémentent-ils une ou plusieurs tâches ou activités ?
- Tous les objets de données implémentent-ils un ou plusieurs objets de données processus ?

La requête 6.28 liste toutes les instances du méta-modèle Fun qui ne sont pas alignées avec les instances du méta-modèle BPM :

- les composants applicatifs non alignés avec des activités ;
- les services non alignés avec une ou plusieurs tâches, et avec une ou plusieurs activités ;
- les objets de données non alignés avec un ou plusieurs objets de données.

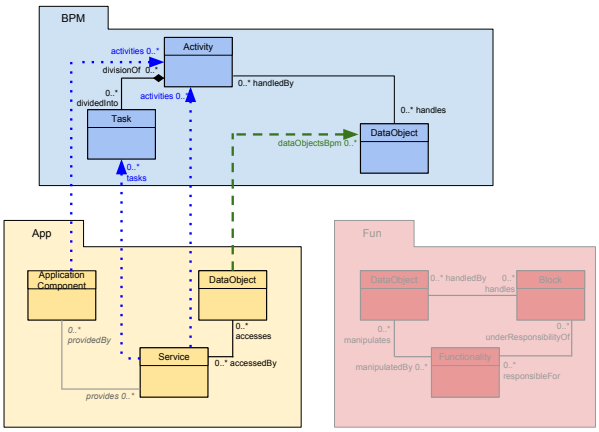


FIGURE 6.12 – Concepts de App alignés avec les concepts de BPM

Listing 6.28 – App ⇒ BPM

```
AppBPMNotAlignedList: context ApplicationLayer
app::ApplicationComponent.allInstances()->select(not notAlignable and
    activities->isEmpty())->collect(oclAsType(ecore::EObject))->union(app::
    Service.allInstances()->select(not notAlignable and tasks->isEmpty() and
     activities->isEmpty())->collect(oclAsType(ecore::EObject)))->union(app
    ::DataObject.allInstances()->select(not notAlignable and dataObjectsBpm
    ->isEmpty())->collect(oclAsType(ecore::EObject)))
```

Les instances de Fun non alignées avec les instances de App

La requête 6.29 répond aux questions suivantes :

- Tous les blocs sont-ils implémentés par un ou plusieurs composants applicatifs ?
- Toutes les fonctionnalités sont-elles implémentées par un ou plusieurs services ?
- Tous les objets de données sont-ils implémentés par un ou plusieurs objets de données applicatifs ?

La requête 6.29 liste toutes les instances du méta-modèle Fun qui ne sont pas alignées avec les instances du méta-modèle App :

- les fonctionnalités non alignées avec un ou plusieurs services ;
- les blocs non alignés avec un ou plusieurs composants applicatifs ;
- les objets de données non alignés avec un ou plusieurs objets de données.

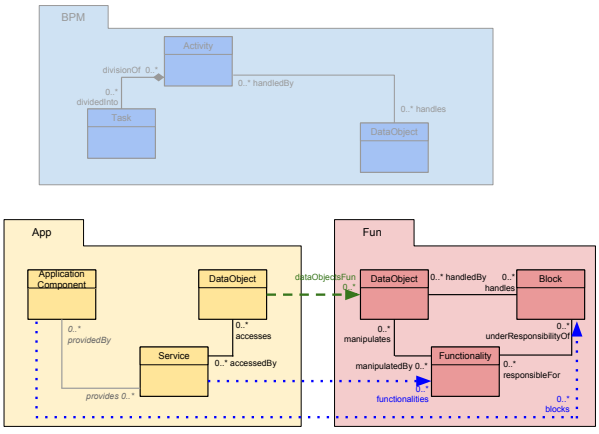


FIGURE 6.13 – Concepts de Fun alignés avec les concepts de App

Listing 6.29 – Fun ⇒ App

```
FunAppNotAlignedList: context FunctionalLayer
fun::Functionality.allInstances()->select(not notAlignable and services->
    isEmpty())->collect(oclAsType(ecore::EObject))->union(fun::DataObject.
    allInstances()->select(not notAlignable and dataObjectsApp->isEmpty())->
    collect(oclAsType(ecore::EObject)))->union(fun::Block.allInstances()->
    select(not notAlignable and applicationComponents->isEmpty())->collect(
    oclAsType(ecore::EObject)))
```

Les instances de App non alignées avec les instances de Fun

La requête 6.30 répond aux questions suivantes :

- Tous les composants applicatifs implémentent-ils un ou plusieurs blocs ?
- Tous les services implémentent-ils une ou plusieurs fonctionnalités ?
- Tous les objets de données implémentent-ils un ou plusieurs objets de données fonctionnels ?

La requête 6.30 liste toutes les instances du méta-modèle Fun qui ne sont pas alignées avec les instances du méta-modèle App :

- les composants applicatifs non alignés avec un ou plusieurs blocs ;
- les services non alignés avec une ou plusieurs fonctionnalités ;
- les objets de données non alignés avec un ou plusieurs objets de données.

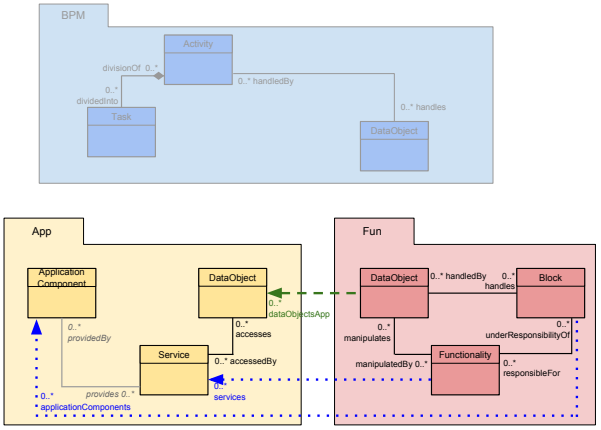


FIGURE 6.14 – Concepts de App alignés avec les concepts de Fun

Listing 6.30 – App ⇒ Fun

```
AppFunNotAlignedList: context ApplicationLayer
app::ApplicationComponent.allInstances()->select(not notAlignable and
    blocks->isEmpty())->collect(oclAsType(ecore::EObject))->union(app::
    Service.allInstances()->select(not notAlignable and functionalities->
    isEmpty())->collect(oclAsType(ecore::EObject)))->union(app::DataObject.
    allInstances()->select(not notAlignable and dataObjectsFun->isEmpty())->
    collect(oclAsType(ecore::EObject)))
```

Liste des éléments alignés

Il est possible également de s'intéresser à lister les éléments qui sont alignés pour chaque couple de points de vue. Les requêtes sont quasiment identiques aux requêtes précédentes pour lister les instances non alignées. La seule différence dans l'écriture des requêtes en OCL est qu'au lieu de tester si les références entre instances de concept sont vides à l'aide de la fonction `isEmpty()`, il faut désormais tester si les références sont non vides à l'aide de la fonction `notEmpty()`. Ainsi pour chaque requête précédente, il faut uniquement remplacer `isEmpty()`, par `notEmpty()`, c'est la raison pour laquelle nous ne redonnons pas le listing des requêtes.

Nous avons élaboré des requêtes pour calculer le nombre d'éléments alignés, puis d'autres requêtes pour lister les éléments non alignés. La prochaine section est dédiée à lister les éléments dont l'alignement est incohérent entre les relations de traitements et de données.

6.3.4 Incohérence d'alignement entre liens de traitement et donnée

Notre définition de l'alignement (cf. section 4.1) sépare les liens de traitement des liens de données. Dans chaque point de vue les traitements peuvent utiliser des données, ainsi cette utilisation doit être similaire dans chaque élément des points de vue alignés. Par exemple, les fonctionnalités du modèle fonctionnel, manipulant des objets de données, sont alignées avec des activités du modèle processus, manipulant également des objets de données. Si l'alignement est cohérent, les objets de données similaires ont aussi un lien d'alignement entre eux. Cependant, toutes les instances de concept de type traitement ne manipulent pas nécessairement un objet de donnée. Dans ce cas, le concept de traitement est écarté de la vérification de la cohérence d'alignement.

Nous donnons la requête de vérification de cohérence par concepts de traitement alignés entre eux. Cette requête retourne la liste des concepts avec une incohérence d'alignement.

BPM

Les premières requêtes sont récapitulées sur la figure 6.15, elles sont orientées à partir du point de vue processus métier (BPM) vers les points de vue fonctionnel (Fun) et applicatif (App). Nous testons d'abord la cohérence de l'alignement des activités du point de vue processus avec les fonctionnalités du point de vue fonctionnel, et les composants applicatifs et les services du point de vue applicatif. Puis, nous testons la cohérence de l'alignement des tâches du point de vue processus avec les services du point de vue applicatif et les fonctionnalités du point de vue fonctionnel.

activités
- 6.31 alignées aux fonctionnalités
- 6.32 alignées aux composants applicatifs
- 6.33 alignées aux services

tâches
- 6.34 alignées aux services
- 6.35 alignées aux fonctionnalités

FIGURE 6.15 – Guide des requêtes d'incohérence de l'alignement à partir de BPM

Les activités avec un alignement incohérent

▷ **Fonctionnalités** : La requête 6.31 liste les activités ayant un alignement incohérent avec des fonctionnalités. La navigation de la requête est illustrée par la figure 6.16.

Listing 6.31 – Activités ⇒ Fonctionnalités

```
ActivityFunctionalityNonConsistent:
context BusinessProcessLayer
bpm::Activity.allInstances()->select
    (functionalities->notEmpty())->
    symmetricDifference(
bpm::Activity.allInstances()->select
    (ac | ac.functionalities.
    manipulates.dataObjectsBpm.
    handledBy->includes(ac)))
```

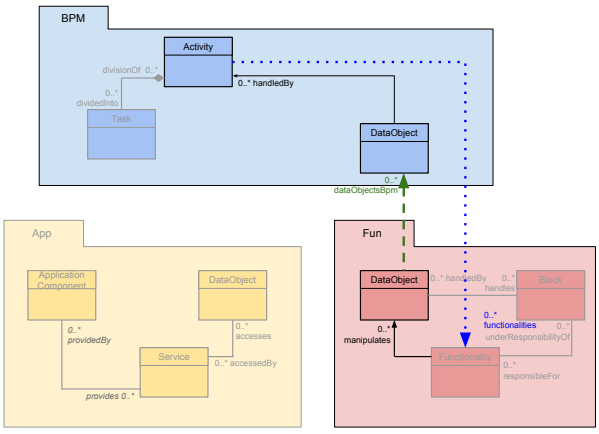


FIGURE 6.16 – Cohérence entre activités et fonctionnalités

▷ **Composants applicatifs** : La requête 6.32 liste les activités ayant un alignement incohérent avec des composants applicatifs. La navigation de la requête est illustrée par la figure 6.17.

Listing 6.32 – Activités ⇒ Composants applicatifs

```
ActivityAppComponentNonConsistent:
context BusinessProcessLayer
bpm::Activity.allInstances()->select
    (applicationComponents->notEmpty
    ())->symmetricDifference(
bpm::Activity.allInstances()->select
    (ac | ac.applicationComponents.
    provides.accesses.dataObjectsBpm.
    handledBy->includes(ac)))
```

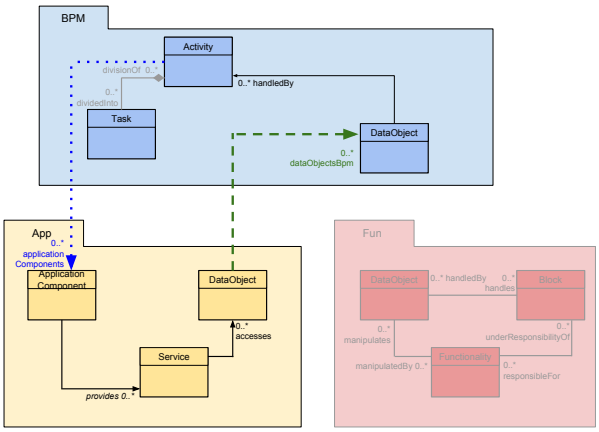


FIGURE 6.17 – Cohérence entre activités et composants applicatifs

▷ **Services** : La requête 6.33 liste les activités ayant un alignement incohérent avec des services. La navigation de la requête est illustrée par la figure 6.18.

Listing 6.33 – Activités ⇒ Services

```
ActivityServiceNonConsistent:
context BusinessProcessLayer
bpm::Activity.allInstances()->select
    (services->notEmpty())->
    symmetricDifference( bpm::
    Activity.allInstances()->select(
    ac | ac.services.accesses.
    dataObjectsBpm.handledBy->
    includes(ac)))
```

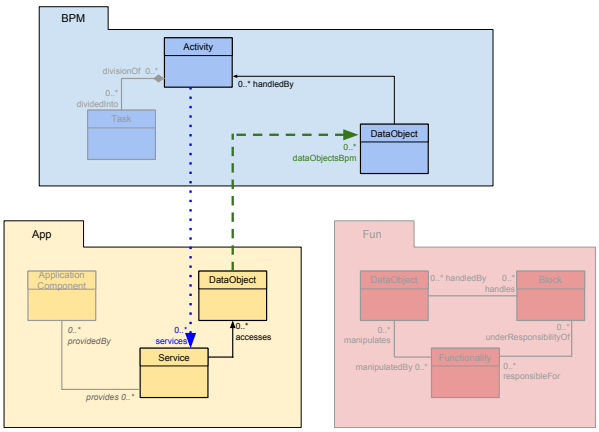


FIGURE 6.18 – Cohérence entre activités et services

Les tâches avec un alignement incohérent

▷ **Services** : La requête 6.34 liste les tâches ayant un alignement incohérent avec des services. La navigation de la requête est illustrée par la figure 6.19.

Listing 6.34 – Tâches ⇒ Services

```
TaskServiceNonConsistent:
context BusinessProcessLayer
bpm::Task.allInstances()->select(
    services->notEmpty())->
    symmetricDifference(
bpm::Task.allInstances()->select(tsk
     | tsk.services.accesses.
    dataObjectsBpm.handledBy->
    includes(tsk)))
```

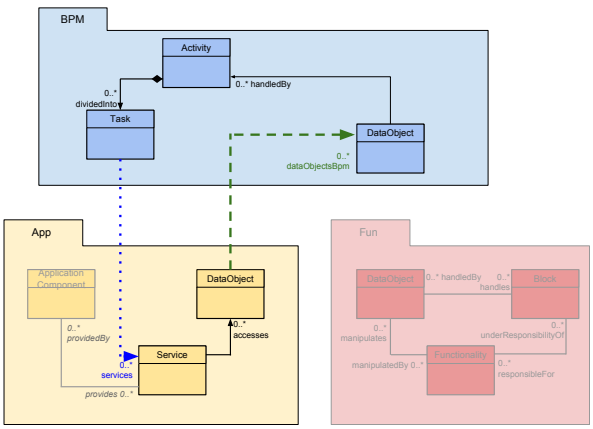


FIGURE 6.19 – Cohérence entre tâches et services

▷ **Fonctionnalités** : La requête 6.35 liste les tâches ayant un alignement incohérent avec des fonctionnalités. La navigation de la requête est illustrée par la figure 6.20.

Listing 6.35 – Tâches ⇒ Fonctionnalités

```
TaskFonctionalityNonConsistent:
context BusinessProcessLayer
bpm::Task.allInstances()->select(
    functionalities->notEmpty())->
    symmetricDifference(
bpm::Task.allInstances()->select(tsk
     | tsk.functionalities.
    manipulates.dataObjectsBpm.
    handledBy.dividedInto->includes(
    tsk)))
```

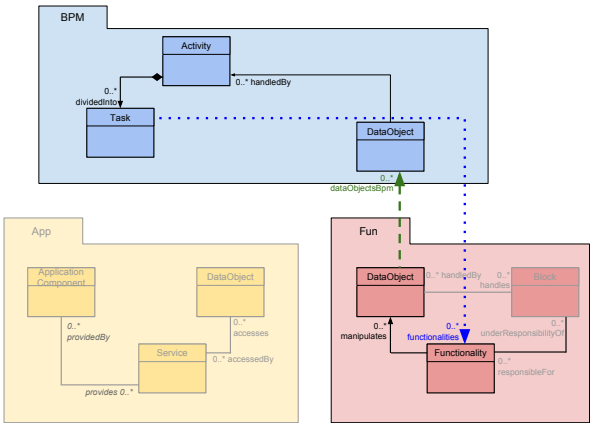


FIGURE 6.20 – Cohérence entre tâches et fonctionnalités

Fun

Les requêtes sont récapitulées sur la figure 6.21, elles sont orientées à partir du point de vue fonctionnel (Fun) vers les points de vue applicatif (App) et processus (BPM). Nous testons d'abord la cohérence de l'alignement des fonctionnalités du point de vue fonctionnel avec les services du point de vue applicatif, et les activités et les tâches du point de vue processus. Puis, nous testons la cohérence de l'alignement des blocs du point de vue fonctionnel avec les composants applicatifs du point de vue applicatif.

fonctionnalités
- 6.36 alignées aux services
- 6.37 alignées aux activités
- 6.38 alignées aux tâches

blocs
- 6.39 alignés aux composants applicatifs

FIGURE 6.21 – Guide des requêtes d'incohérence de l'alignement à partir de Fun

Les fonctionnalités avec un alignement incohérent

▷ **Services** : La requête 6.36 liste les fonctionnalités avec un alignement incohérent avec des services. La navigation de la requête est illustrée par la figure 6.22.

Listing 6.36 – Fonctionnalités ⇒ Services

```
FunctionalityServiceNonConsistent:
context FunctionnalLayer
fun::Functionality.allInstances()->
    select(services->notEmpty())->
    symmetricDifference(
fun::Functionality.allInstances()->
    select(fun | fun.services.
    accesses.dataObjectsFun.
    manipulatedBy->includes(fun)))
```

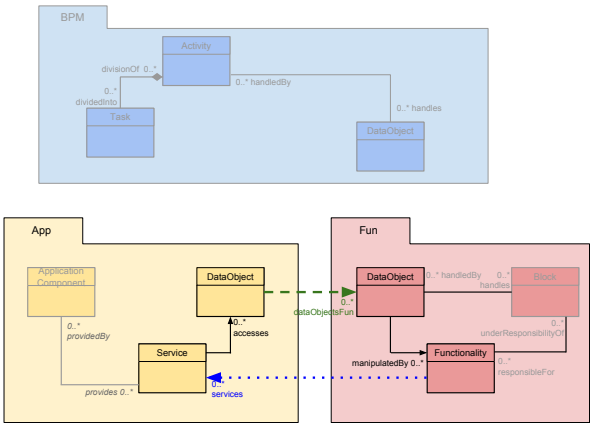


FIGURE 6.22 – Cohérence entre fonctionnalités et services

▷ **Activités** : La requête 6.37 liste les fonctionnalités ayant un alignement incohérent avec des activités. La navigation de la requête est illustrée par la figure 6.23

Listing 6.37 – Fonctionnalités ⇒ Activités

```
FunctionalityActivityNonConsistent:
context FunctionnalLayer
fun::Functionality.allInstances()->
    select(activities->notEmpty())->
    symmetricDifference(
fun::Functionality.allInstances()->
    select(fun | fun.activities.
    handles.dataObjectsFun.
    manipulatedBy->includes(fun)))
```

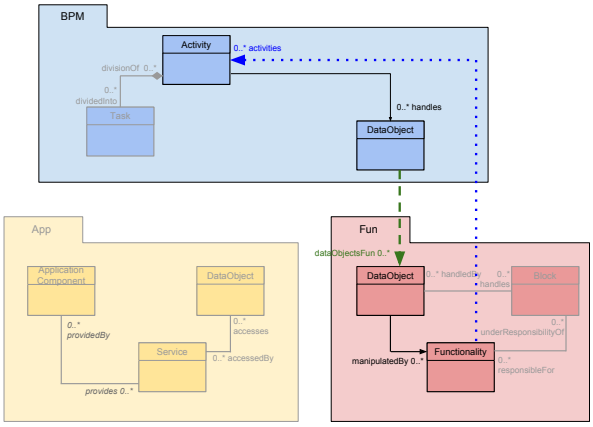


FIGURE 6.23 – Cohérence entre fonctionnalités et activités

▷ **Tâches** : La requête 6.38 liste les fonctionnalités ayant un alignement incohérent avec des tâches. La navigation de la requête est illustrée par la figure 6.24.

Listing 6.38 – Fonctionnalités ⇒ Tâches

```
FunctionalityTaskNonConsistent:
context FunctionnalLayer
fun::Functionality.allInstances()->
    select(tasks->notEmpty())->
    symmetricDifference(
fun::Functionality.allInstances()->
    select(fun | fun.tasks.divisionOf
    .handles.dataObjectsFun.
    manipulatedBy->includes(fun)))
```

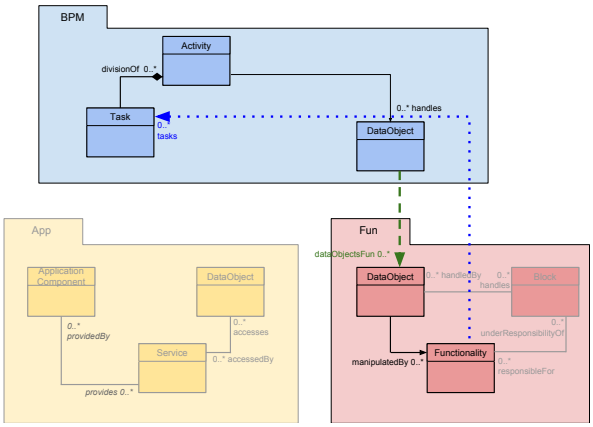


FIGURE 6.24 – Cohérence entre fonctionnalités et tâches

Les blocs avec un alignement incohérent

▷ **Composants applicatifs** : La requête 6.39 liste les blocs ayant un alignement incohérent avec des composants applicatifs. La navigation de la requête est illustrée par la figure 6.25.

Listing 6.39 – Blocs ⇒ Composants applicatifs

```
BlockAppComponentNonConsistent:
context FunctionnalLayer
fun::Block.allInstances()->select(
    applicationComponents->notEmpty()
    )->symmetricDifference(
fun::Block.allInstances()->select(
    blk | blk.applicationComponents.
    provides.accesses.dataObjectsFun.
    handles->includes(blk)))
```

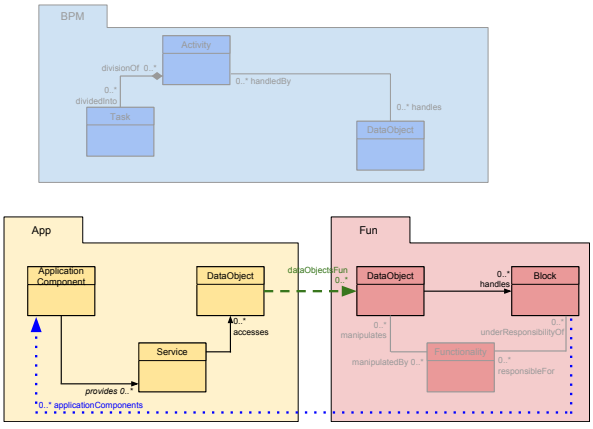


FIGURE 6.25 – Cohérence entre blocs et composants applicatifs

App

Les requêtes sont récapitulées sur la figure 6.26, elles sont orientées à partir du point de vue applicatif (App) vers les points de vue processus (BPM) et fonctionnel (Fun). Nous testons d'abord la cohérence de l'alignement des services du point de vue applicatif avec les activités et les tâches du point de vue processus et les fonctionnalités du point de vue fonctionnel. Puis, nous testons la cohérence de l'alignement des composants applicatifs du point de vue applicatif avec les activités du point de vue processus, et les blocs du point de vue fonctionnel.

services
- 6.40 alignés aux activités
- 6.41 alignés aux tâches
- 6.42 alignés aux fonctionnalités

composants applicatifs
- 6.43 alignés aux activités
- 6.44 alignés aux blocs

FIGURE 6.26 – Guide des requêtes d'incohérence de l'alignement à partir de App

Les services avec un alignement incohérent

▷ **Activités** : La requête 6.40 liste les services ayant un alignement incohérent avec des activités. La navigation de la requête est illustrée par la figure 6.27.

Listing 6.40 – Services ⇒ Activités

```
ServiceActivityNonConsistent:
context ApplicationLayer
app::Service.allInstances()->select(
    activities->notEmpty())->
    symmetricDifference(
app::Service.allInstances()->select(
    svc | svc.activities.handles.
    dataObjectsApp.accessedBy->
    includes(svc)))
```

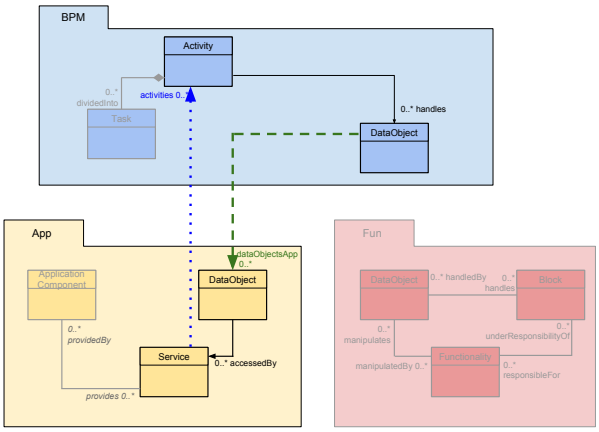


FIGURE 6.27 – Cohérence entre services et activités

▷ **Tâches** : La requête 6.41 liste les services ayant un alignement incohérent avec des tâches. La navigation de la requête est illustrée par la figure 6.28.

Listing 6.41 – Services ⇒ Tâches

```
ServiceTaskNonConsistent:
context ApplicationLayer
app::Service.allInstances()->select(
    tasks->notEmpty())->
    symmetricDifference(
app::Service.allInstances()->select(
    svc | svc.tasks.divisionOf.
    handles.dataObjectsApp.accessedBy
    ->includes(svc)))
```

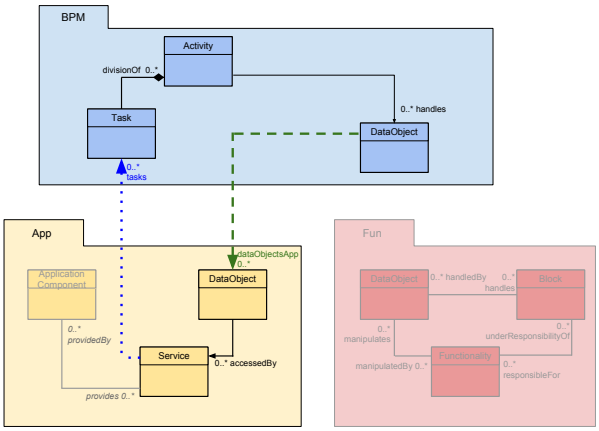


FIGURE 6.28 – Cohérence entre services et tâches

▷ **Fonctionnalités** : La requête 6.42 liste les services ayant un alignement incohérent avec des fonctionnalités. La navigation de la requête est illustrée par la figure 6.29.

Listing 6.42 – Services ⇒ Fonctionnalités

```
ServiceFunctionalityNonConsistent:
context ApplicationLayer
app::Service.allInstances()->select(
    functionalities->notEmpty())->
    symmetricDifference(
app::Service.allInstances()->select(
    svc | svc.functionalities.
    manipulates.dataObjectsApp.
    accessedBy->includes(svc)))
```

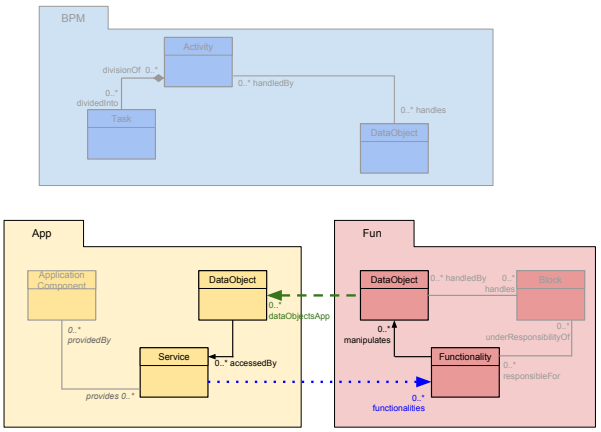


FIGURE 6.29 – Cohérence entre services et fonctionnalités

Les composants applicatifs avec un alignement incohérent

▷ **Activités** : La requête 6.43 suivante liste les composants applicatifs avec un alignement incohérent avec des activités. La navigation de la requête est illustrée par la figure 6.30.

Listing 6.43 – Composants applicatifs ⇒ Activités

```
AppComponentActivityNonConsistent:
context ApplicationLayer
app::ApplicationComponent.
    allInstances()->select(activities
    ->notEmpty())->
    symmetricDifference(
app::ApplicationComponent.
    allInstances()->select(cpn | cpn.
    activities.handles.dataObjectsApp
    .accessedBy.providedBy->includes(
    cpn)))
```

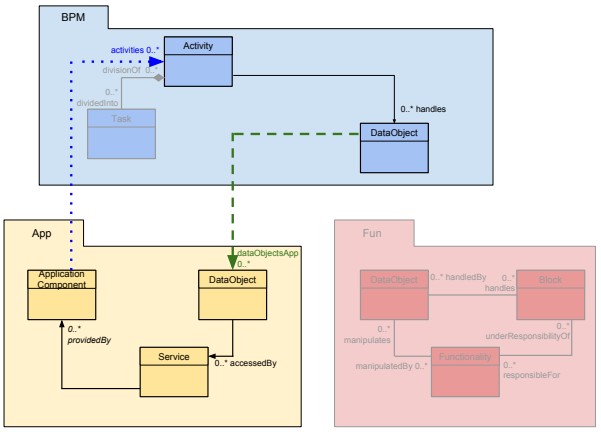


FIGURE 6.30 – Cohérence entre composants applicatifs et activités

▷ **Blocs** : La requête 6.44 liste les composants applicatifs ayant un alignement incohérent avec des blocs. La navigation de la requête est illustrée par la figure 6.31.

Listing 6.44 – Composants applicatifs ⇒ Blocs

```
AppComponentBlockNonConsistent:
context ApplicationLayer
app::ApplicationComponent.
    allInstances()->select(blocks->
    notEmpty())->symmetricDifference(
app::ApplicationComponent.
    allInstances()->select(cpn | cpn.
    blocks.handledBy.dataObjectsApp.
    accessedBy.providedBy->includes(
    cpn)))
```

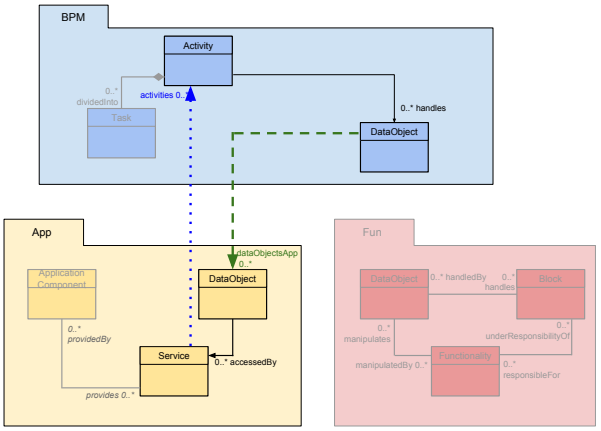


FIGURE 6.31 – Cohérence entre composants applicatifs et blocs

Nous avons vérifié la cohérence structurelle de l'alignement entre les liens de traitement et de donnée de concepts similaires. Dans la section suivante, nous mesurons le couplage entre instances de concepts.

6.3.5 Couplage

Le couplage est le nombre de liens à partir d'une instance de concepts vers d'autres instances de concepts. Un couplage fort révèle une complexité importante en termes de dépendance, et entraîne des difficultés d'évolution.

TABLE 6.4 – Guide des différentes requêtes de liste de couplage

	N° requête - listing	
BPM	Activité	6.45
	Tâche	6.46
	Objet de donnée	6.47
Fun	Fonctionnalité	6.48
	Bloc	6.49
	Objet de donnée	6.50
App	Composant applicatif	6.51
	Service	6.52
	Objet de donnée	6.53

Chaque requête de calcul du couplage retourne une liste de valeurs du nombre de liens pour chaque élément. Exemple : 0, 0, 1, 2, 1, 3.

Un calcul de post-traitement est nécessaire pour compter le nombre de couples de même valeur. Exemple : {0 ;2}, {1 ;2}, {2 ;1}, {3 ;1}... qui se lit 2 éléments avec 0 lien, 2 éléments avec 1 lien, 1 élément avec 2 liens, 1 élément avec 3 liens, etc.

L'ensemble des requêtes est récapitulé par le tableau 6.4.

BPM

Nous testons d'abord les couplages des activités, tâches et objets de données du point de vue processus.

Activité La requête 6.45 calcule le nombre de couples entre les activités de BPM, les composants applicatifs et services de App, et les fonctionnalités de Fun.

Listing 6.45 – Couplage des activités

```
BpmActivityDependencies:
context BusinessProcessLayer:
bpm::Activity.allInstances()->collect(applicationComponents->size())->union
    (bpm::Activity.allInstances()->collect(services->size()))->union(bpm::
    Activity.allInstances()->collect(functionalities->size()))
```

Tâche La requête 6.46 calcule le nombre de couples entre les tâches de BPM et les services de App.

Listing 6.46 – Couplage des tâches

```
BpmTaskDependencies:
context BusinessProcessLayer:
bpm::Task.allInstances()->collect(services->size())
```

Objet de donnée La requête 6.47 calcule le nombre de couples entre les objets de données de BPM, les objets de données de Fun et App.

Listing 6.47 – Couplage des objets de données de BPM

```
BpmDataObjectDependencies:
context BusinessProcessLayer:
bpm::DataObject.allInstances()->collect(dataObjectsFun->size())->union(bpm
    ::DataObject.allInstances()->collect(dataObjectsApp->size()))
```

Fun

Nous testons ensuite les couplages des fonctionnalités, blocs et objets de données du point de vue processus.

Fonctionnalité La requête 6.48 calcule le nombre de couples entre les fonctionnalités de Fun, les activités de BPM et les services de App.

Listing 6.48 – Couplage des fonctionnalités

```
FunFunctionalityDependencies:
context FunctionalLayer:
fun::Functionality.allInstances()->collect(activities->size())->union(fun::
    Functionality.allInstances()->collect(services->size()))
```

Bloc La requête 6.49 calcule le nombre de couples entre les blocs de Fun et les composants applicatifs de App.

Listing 6.49 – Couplage des blocs

```
FunBlockDependencies:
context FunctionalLayer:
fun::Block.allInstances()->collect(applicationComponents->size())
```

Objet de donnée La requête 6.50 calcule le nombre de couples entre les objets de données de Fun et les objets de données de BPM et App.

Listing 6.50 – Couplage des objets de données de Fun

```
FunDataObjectDependencies:
context FunctionalLayer:
fun::DataObject.allInstances()->collect(dataObjectsApp->size())->union(fun
    ::DataObject.allInstances()->collect(dataObjectsBpm->size()))
```

App

Nous testons enfin les couplages des composants applicatifs, services et objets de données du point de vue processus.

Composant applicatif La requête 6.51 calcule le nombre de couples entre les composants applicatifs de App, les activités de BPM, et les blocs de Fun.

Listing 6.51 – Couplage des composants applicatifs

```
AppApplicationComponentDependencies:
context ApplicationLayer:
app::ApplicationComponent.allInstances()->collect(activities->size())->
    union(app::ApplicationComponent.allInstances()->collect(blocks->size()))
```

Service La requête 6.52 calcule le nombre de couples entre les services de App et les activités et tâches de BPM.

Listing 6.52 – Couplage des services

```
AppServiceDependencies:
context ApplicationLayer:
app::Service.allInstances()->collect(activities->size())->union(app::
    Service.allInstances()->collect(tasks->size()))
```

Objet de donnée La requête 6.53 calcule le nombre de couples entre les objets de données de App et les objets de données de Fun et BPM.

Listing 6.53 – Couplage des objets de données de App

```
AppDataObjectDependencies:
context ApplicationLayer:
app::DataObject.allInstances()->collect(dataObjectsBpm->size())->union(app
    ::DataObject.allInstances()->collect(dataObjectsFun->size()))
```

Le calcul de la valeur des couplages permet d'indiquer si l'alignement possède beaucoup de couplages forts ; néanmoins cet indicateur est insuffisant pour visualiser les couplages. Dans la section suivante, nous proposons une méthode de visualisation matricielle des couplages par analyse des dépendances.

6.4 La visualisation des dépendances

L'alignement est composé d'éléments de différents modèles reliés par une relation. Ces éléments et relations forment un graphe : les éléments sont les nœuds (ou sommets) et les relations sont les arcs.

Nous souhaitons obtenir une vue d'ensemble lisible afin de détecter les dépendances entre éléments.

Problème

La représentation sous forme de graphe a l'inconvénient d'être difficilement lisible et navigable lorsque le nombre de nœuds et arcs devient important (plusieurs dizaines et plus). Les deux représentations les plus communes pour visualiser un graphe sont : la vue arborescente en figure 6.32, et le tracé de graphe en figure 6.34.

La vue arborescente est construite en ingénierie des modèles de la façon suivante : un nœud de l'arbre (élément possédant des fils) est caractérisé par un élément contenant un lien d'agrégation ou composition vers d'autres éléments. Un modèle sans agrégation (contenant uniquement des associations) est donc à plat, les éléments étant tous racines. Par exemple sur la figure 6.32, l'arbre contient trois nœuds "*Business Process Layer*", "*Process Handle Claim*" et "*Process Close Contract*".

Le tracé de graphe est un diagramme composé de boîtes et segments représentant les nœuds et les arcs. Le placement des éléments graphiques est réalisé par un "algorithme de placement"[1]. Cependant, les algorithmes génériques sont très limités et ne font pas de reconnaissance de groupements.

[1] *Layout algorithm* en anglais

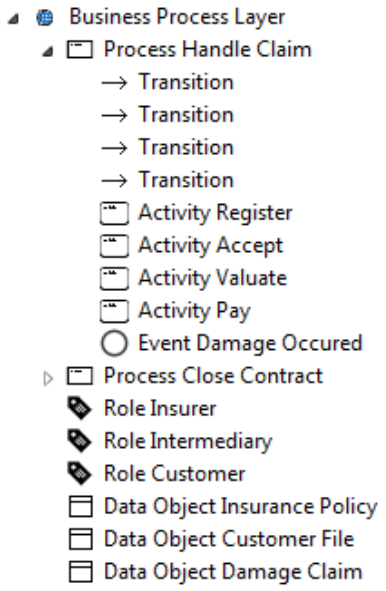


FIGURE 6.32 – Vue arborescente

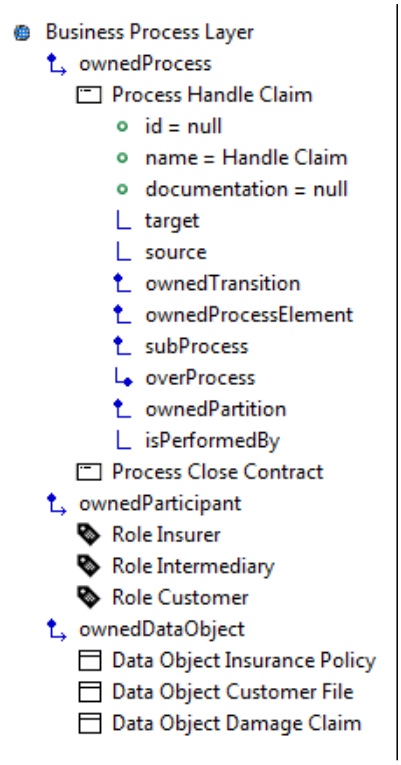


FIGURE 6.33 – Vue arborescente avec affichage des références et attributs

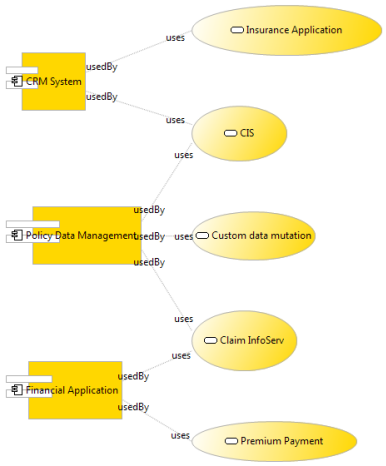


FIGURE 6.34 – Tracé de graphe

La vue arborescente est un mode de visualisation limité : les nœuds permettent uniquement de représenter des liens de compositions. Il existe un autre type de vue arborescente en figure 6.33 qui affiche en plus les relations (et donc tous les arcs du graphe correspondant) mais qui complexifie d'autant plus l'affichage.

Le tracé de graphe a aussi des problèmes de lisibilité lorsque le diagramme est composé de trop nombreux nœuds et que plusieurs arcs entrent et sortent de chaque nœud.

L'architecte d'entreprise manipule une cartographie du SI représentée par de volumineux modèles. Ainsi, nous avons d'une part pour objectif d'être en capacité de charger et afficher des modèles représentant jusqu'à 10 000 nœuds. D'autre part les modèles peuvent être de différents points de vue, il faut donc gérer cette appartenance multi-domaines.

Solution

Une solution pour afficher un graphe est la matrice d'adjacence, encore appelée **DSM** pour : *Design Structure Matrix*, *Dependency Structure Matrix*, *Dependency Structure Method*, ou *Dependency Source Matrix*. Nous choisissons la définition ***Dependency Structure Matrix*** qui caractérise davantage notre besoin de représenter les dépendances sous forme de matrice. La matrice d'adjacence est une structure mathématique utilisée en théorie des graphes, alors que DSM est un domaine d'étude plus large utilisé en sciences de l'information avec des outils pour exploiter les matrices d'adjacence. Dans la section suivante, nous allons présenter en détail les DSM.

6.4.1 DSM : matrice de dépendances

La matrice de dépendances permet d'afficher tout type de concept de graphe aussi bien issu des sciences de gestion comme les activités d'un planning (GANTT/PERT) ; ou des sciences informatiques comme des composants informatiques [43, 64]. La matrice de dépendances permet de représenter des graphes orientés (arcs directionnels sortants et entrants) ou non (arcs bi-directionnels).

Pour introduire la notion de DSM, nous utilisons l'exemple en figure 6.35, il s'agit d'un petit graphe composé de 7 nœuds et de 10 arcs non orientés. Par la représentation qui a été choisie, nous voyons deux regroupements, appelés communément en théorie des graphes *cluster*. Le premier regroupement est composé des nœuds 1, 2, 3, 4, tous inter-connectés entre eux ; le second regroupement est composé des nœuds 5, 6, 7. Les deux regroupements sont reliés par un seul arc entre les nœuds 2 et 5. L'interprétation révèle que les nœuds de chaque regroupement sont fortement dépendants entre eux.

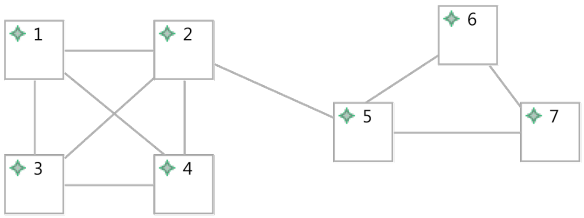


FIGURE 6.35 – Exemple de graphe

La figure étant assez simple et ses composantes fortement connexes, l'identification de ces deux groupes est facilitée, néanmoins avec davantage de nœuds la tâche d'identification de tels groupements deviendrait de plus en plus difficile.

À partir de la lecture de la figure, nous allons réaliser la matrice d'adjacence. Tout d'abord, il faut constituer la liste d'adjacence en figure 6.36 en parcourant les nœuds du graphe à partir des arcs. Par exemple, le nœud **1** est relié aux nœuds 2, 3, 4 ; le nœud **2** est relié aux nœuds 1, 3, 4, 5 ; etc.

1 : 2, 3, 4
2 : 1, 3, 4, 5
3 : 1, 2, 4
4 : 1, 2, 3
5 : 2, 6, 7
6 : 5, 7
7 : 5, 6

FIGURE 6.36 – Liste d'adjacence du graphe d'exemple

Nous remplissons ensuite la matrice du tableau 6.5 à l'aide de la liste. Le graphe étant non orienté, la matrice est symétrique.

TABLE 6.5 – Matrice d'adjacence du graphe d'exemple

	1	2	3	4	5	6	7
1		1	1	1			
2	1		1	1	1		
3	1	1		1			
4	1	1	1				
5		1				1	1
6					1		1
7					1	1	

TABLE 6.6 – Matrice d'adjacence et les deux regroupements

	1	2	3	4	5	6	7
1		1	1	1			
2	1		1	1	1		
3	1	1		1			
4	1	1	1				
5		1				1	1
6					1		1
7					1	1	

Sur la matrice 6.5 nous distinguons deux regroupements, les mêmes que sur la représentation graphique, nous les avons coloriés dans la matrice 6.6.

Cependant, l'apparition de regroupement dans la matrice dépend de l'ordre de parcours des nœuds du graphe. Ainsi, on remarquera que la matrice d'adjacence du tableau 6.7 produite avec un ordre de parcours différent, est totalement différente et ne permet pas la détection de regroupements.

TABLE 6.7 – Matrice d'adjacence avec un nouvel ordre de lecture du graphe

	5	2	3	7	1	6	4
5		1		1		1	
2	1		1		1		1
3		1			1		1
7	1					1	
1		1	1				1
6	1			1			
4		1	1		1		

La lisibilité des regroupements du graphe sur la matrice étant dépendante de l'ordre de lecture des nœuds, nous allons étudier dans la section suivante un algorithme permettant de détecter les regroupements afin d'appliquer l'ordre des nœuds optimal sur la matrice.

6.4.2 Algorithme de clustering : MCL

Il existe plusieurs algorithmes de détection de regroupements [108], néanmoins ils ne s'appliquent pas à tous les types de données constituant le graphe. Un comparatif est dressé dans le tableau 6.8.

TABLE 6.8 – Comparaison des algorithmes d'analyse de regroupement

Type	Application	Méthode d'analyse			
		Partitioning	Tearing	Banding	Clustering
Tâches et activités	Gestion de projet, réduction de temps	✓	✓	✓	✗
Aide à la décision multi-critères	Dresser la liste des critères et leurs prédécesseurs	✗	✓	✓	✗
Management	Organisation, gestion des ressources	✗	✗	✗	✓
Composant	Architecture système, modélisation et ingénierie	✗	✗	✗	✓

La méthode d'analyse qui correspond à notre application est le **clustering**. Pour ce type d'analyse, il existe également plusieurs algorithmes [89]. Nous avons réalisé des recherches et des expérimentations pour trouver le plus adapté avec un résultat optimal. Tous sont basés sur le modèle mathématique de la marche aléatoire[2]. Le principe est qu'il y a davantage d'arcs à l'intérieur d'un cluster qu'entre les clusters. Ainsi, en naviguant d'un nœud à l'autre de façon aléatoire, les probabilités sont supérieures de rester dans un arc du cluster que de sortir du cluster. Les chaînes de Markov permettent de calculer la marche aléatoire d'un graphe.

[2] *Random Walks* en anglais

Dans les nombreux algorithmes de clustering existants [15], la plupart ne sont pas utilisables dans notre contexte car ils nécessitent de connaître le nombre de regroupements à l'avance. Seul un algorithme se distingue : le *Markov Clustering* (**MCL**). Il permet en particulier de trouver les regroupements sans fixer un nombre au préalable, et le calcul est déterministe, le résultat est identique à chaque exécution.

Note Pour aller plus loin, Van Dongen [100] propose des travaux très détaillés sur l'algorithme MCL ainsi qu'une implémentation.

Normalisation par marche aléatoire

La marche aléatoire dans MCL à l'aide de la chaîne de Markov est l'étape de normalisation de l'algorithme. Sur la table 6.9, on calcule à partir de la matrice d'adjacence la probabilité du marcheur de passer d'un nœud à l'autre. Par exemple, à partir du nœud 1, la probabilité du marcheur de passer aux nœuds 2, 3 ou 4 est de 33% ; et la probabilité du marcheur de passer aux nœuds 5, 6, ou 7 est de 0%.

TABLE 6.9 – Normalisation de la matrice d'adjacence

	1	2	3	4	5	6	7
1		1	1	1			
2	1		1	1	1		
3	1	1		1			
4	1	1	1				
5		1				1	1
6					1		1
7					1	1	

▶

	1	2	3	4	5	6	7
1		.25	.33	.33			
2	.33		.33	.33	.33		
3	.33	.25		.33			
4	.33	.25	.33				
5		.25				.5	.5
6					.33		.5
7					.33	.5	

Déroulement de l'algorithme MCL

PRÉTRAITEMENT

Ajout de boucles à chaque nœud, permet d'augmenter la probabilité de rester au sein d'un même cluster lors de la marche aléatoire ;

Normalisation recalcule les probabilités de la matrice.

RÉPÉTER

Expansion (*expand*) ou exponentation, calcule la matrice à la puissance e ;

Dilatation (*inflate*) :

produit matriciel de Hadamard élève à la puissance h chaque valeur de la matrice ;

réduction (*pruning*) met à zéro toutes les valeurs en dessous d'un seuil p proche de zéro ;

normalisation recalcule les probabilités de chaque colonne. La somme des valeurs de chaque colonne est de 1.

JUSQU'À [Les itérations s'arrêtent lorsque les valeurs de la matrice M^i nouvellement calculée sont identiques aux valeurs de la matrice M^{i-1} calculée à l'itération i précédente.]

L'algorithme terminé, les regroupements apparaissent dans la matrice résultat : chaque ligne représente un regroupement de nœuds par les cellules non vides, il suffit de collecter les numéros de colonne correspondants à ces cellules. Enfin, pour réordonner les nœuds de la matrice, il faut appliquer l'ordre des nœuds de chaque regroupement précédemment collecté, les clusters apparaissent sur la diagonale.

Paramètres de MCL et influences

- **Seuil de réduction** (p) : 0,0001 ; 0,001 ; 0,01 ; ... plus le seuil est élevé, plus le nombre d'itérations est réduit, l'exécution est plus rapide, mais la précision est inférieure : les clusters deviennent plus nombreux, mais de petite taille (composés de moins de nœuds). Le seuil de réduction satisfaisant durant nos expérimentations était de 0,001.
- **Puissance de Hadamard** (h) : plus la valeur de la puissance est élevée, moins le nombre d'itérations est important, mais la précision est également inférieure. Le problème est que le résultat d'une puissance élevée se rapproche trop rapidement du seuil de réduction. La valeur de la puissance satisfaisante constatée après expérimentation est 2.
- **Puissance de la matrice** (e) : ce paramètre impacte fortement le temps d'exécution de l'algorithme. Il s'agit de l'opération la plus coûteuse, la complexité de l'algorithme de multiplication naïf est $O(n^3)$. Par défaut on multiplie la matrice M par elle-même qu'une seule fois, soit M^2, cependant nous avons testé jusqu'à M^5. Dans ce cas, le résultat possède moins de clusters, ceux-ci sont de plus grande taille. Par exemple pour une matrice 526x526, avec M^5, MCL produit 88 clusters, contre 155 clusters avec M^2. L'observation de chacun de ces deux résultats, nous a permis de déterminer que la multiplication M^2 permet un regroupement des dépendances plus précis.

Il existe d'autres algorithmes concurrents à MCL comme *Restricted Neighbourhood Search Clustering* (**RNSC**). Dhara [36] a réalisé un comparatif des deux algorithmes. Le gros avantage de RNSC est qu'il est très rapide, mais l'inconvénient réside dans le fait qu'il faut connaître à l'avance le nombre de clusters. RNSC n'est donc pas utilisable dans notre cas. Cependant, nous avons testé RNSC en entrant le nombre de clusters trouvés avec MCL, le résultat était très proche, voire identique selon les matrices testées. Cette expérimentation nous a conforté dans la qualité des résultats trouvés avec MCL.

Dans la section suivante, nous présentons l'éditeur que nous avons implémenté pour afficher une matrice d'un modèle de nos points de vue (BPM, Fun et App), puis notre implémentation de l'algorithme MCL pour afficher les clusters.

6.4.3 Conception d'un éditeur de matrice interactif

Notre but est de réaliser un éditeur de matrice de dépendances pour Eclipse. Le composant natif standard pour créer des tableaux et arborescences est *Jface TreeViewer*. Nous avons fait une première implémentation avec ce composant d'interface graphique que nous avons vite abandonné pour des problèmes de passage à l'échelle.

Le *TreeViewer* présente un gros défaut : le composant est entièrement calculé, mêmes les parties non visibles à l'écran de l'utilisateur. Cependant, l'objectif est d'être compatible avec des matrices d'une taille allant jusqu'à 10 000 nœuds, soit 100 000 000 de cellules (10 000 x 10 000). Nous ne souhaitons pas afficher l'intégralité à l'écran, mais pouvoir focaliser sur des nœuds qui nous intéressent pour analyser leurs dépendances. Nous avons trouvé un composant qui calcule uniquement la partie de matrice à afficher à l'écran, il s'agit du composant Nebula NatTable[3].

Nous avons donc développé notre éditeur de matrice avec NatTable, une illustration est donnée en figure 6.37 dans laquelle nous avons ouvert le graphe d'exemple fil rouge

[3] https://eclipse.org/nattable/

de la figure 6.35. L'éditeur est générique et permet d'ouvrir tout modèle EMF quel que soit le méta-modèle. Le calcul des adjacences entre nœuds est basé sur les références entre les objets du modèle EMF.

	1	2	3	4	5	6	7	8
1 Graph	X							
2 Node 1		X	1	1	1			
3 Node 2		1	X	1	1	1		
4 Node 3		1	1	X	1			
5 Node 4		1	1	1	X			
6 Node 5			1			X	1	1
7 Node 6						1	X	1
8 Node 7						1	1	X

FIGURE 6.37 – L'exemple fil rouge de graphe ouvert avec notre éditeur

Les fonctionnalités d'interface

Pour faciliter l'utilisation de notre éditeur, nous avons enrichi son interface utilisateur de plusieurs fonctionnalités :

- **Vue arborescente** Lorsqu'une référence entre objets est une composition, le nœud est pliable / dépliable (cf. figure 6.38).
- **Zoom** Nous avons implémenté une fonction de zoom pour afficher plus d'éléments à l'écran (cf. figure 6.39).
- **Filtre** Nous avons créé une fonction de filtre disponible dans la vue propriété pour masquer les concepts de méta-modèle que nous ne souhaitons pas analyser (cf. figure 6.40).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	19	27	28	29	30	31	32
1 Business Pr...	X																					
2 Process Ha...		X																				
3 Transition			X				1				1											
4 Transition				X			1	1														
5 Transition					X			1	1													
6 Transition						X			1	1												
7 Activity Re...			1	1			X														1	1
8 Activity Ac...				1	1			X												1	1	1
9 Activity Val...					1	1			X													1
10 Activity Pay						1				X											1	
11 Event Da...			1								X											
12 Process te...												X										
13 Event eve...													X									
14 Condition ...														X								
15 Process s...															X							
19 Process C...																X						
27 Role Insurer																	X					
28 Role Inter...																		X				
29 Role Cust...																			X			
30 Data Obje...								1												X		
31 Data Obje...							1	1		1											X	
32 Data Obje...							1	1	1													X

FIGURE 6.38 – Capture d'écran d'un modèle processus avec arborescence sur les nœuds de type processus

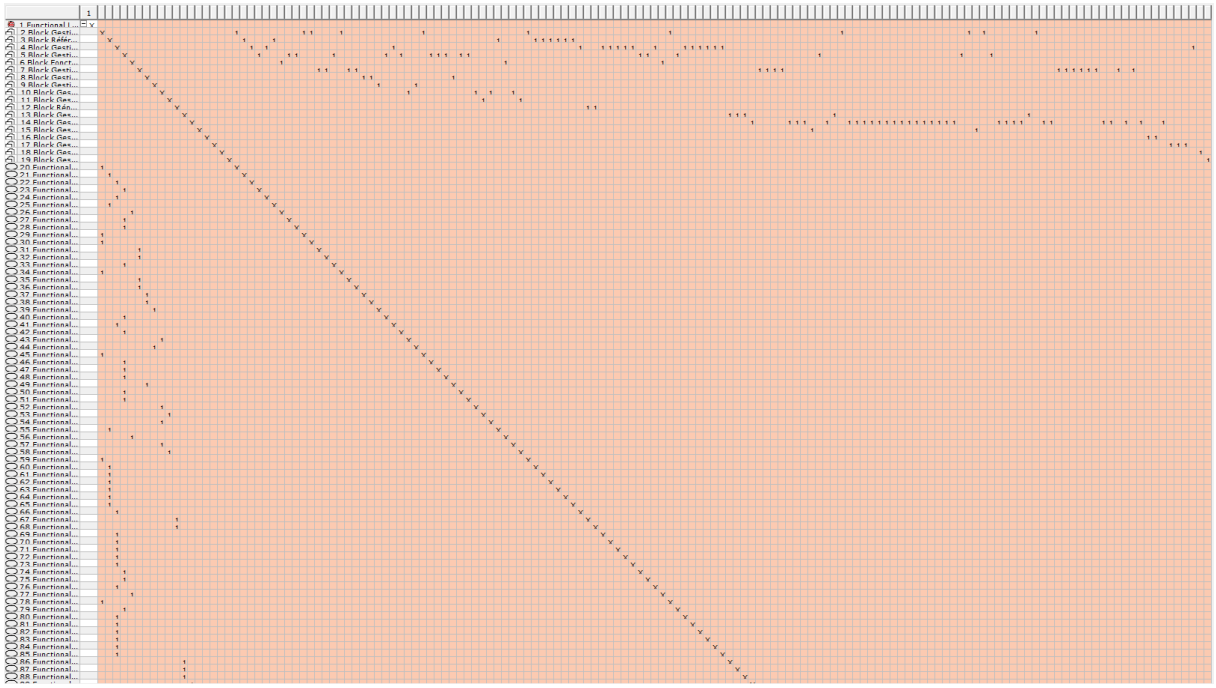


FIGURE 6.39 – Capture d'écran du modèle fonctionnel du cas SAMI au zoom 50%

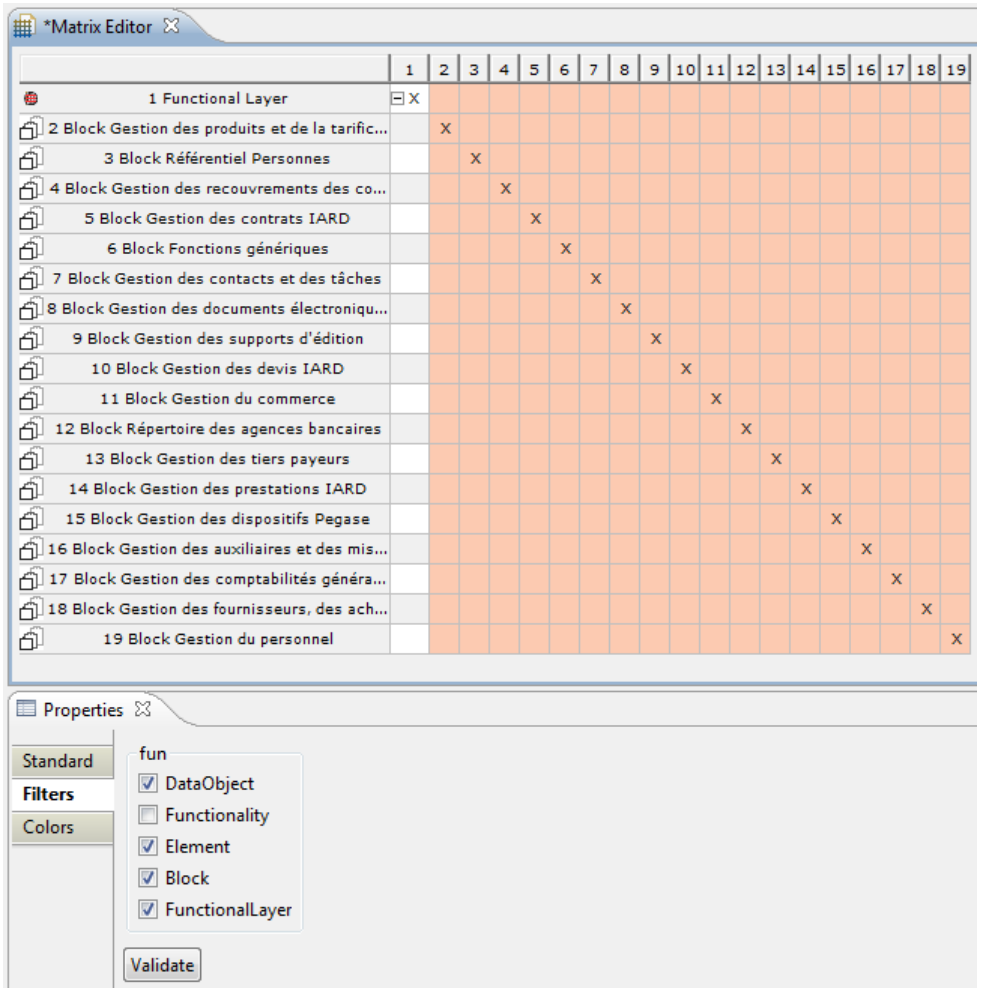


FIGURE 6.40 – Capture d'écran du modèle fonctionnel SAMI avec les fonctionnalités filtrées

Clustering MCL

Nous avons affiné l'implémentation de l'algorithme MCL jusqu'à avoir des performances permettant le passage à l'échelle et traiter dans un temps acceptable un graphe volumineux ayant jusqu'à 10 000 sommets.

Tout d'abord, nous avons implémenté entièrement en Java l'algorithme MCL et notamment l'étape la plus coûteuse de multiplication. Sur la matrice cible à 10 000 sommets, après une dizaine d'heures, le calcul ne finissait toujours pas. Nous avons alors opté pour externaliser la multiplication de matrices par une librairie dédiée écrite en Fortran : openBLAS. Les performances sont au rendez-vous : sur une matrice de petite taille ayant moins de 1 000 sommets, l'exécution dure quelques secondes. Pour une matrice constituée de 12 000 nœuds, la mémoire occupe 800 Mo (en précision flottante simple) et l'exécution dure 40 minutes.

La matrice de la figure 6.41 est la même que celle de la figure 6.39, une fois appliqué le clustering MCL. Les différents clusters apparaissent en surbrillance ce qui facilite la lecture des dépendances entre les nœuds.

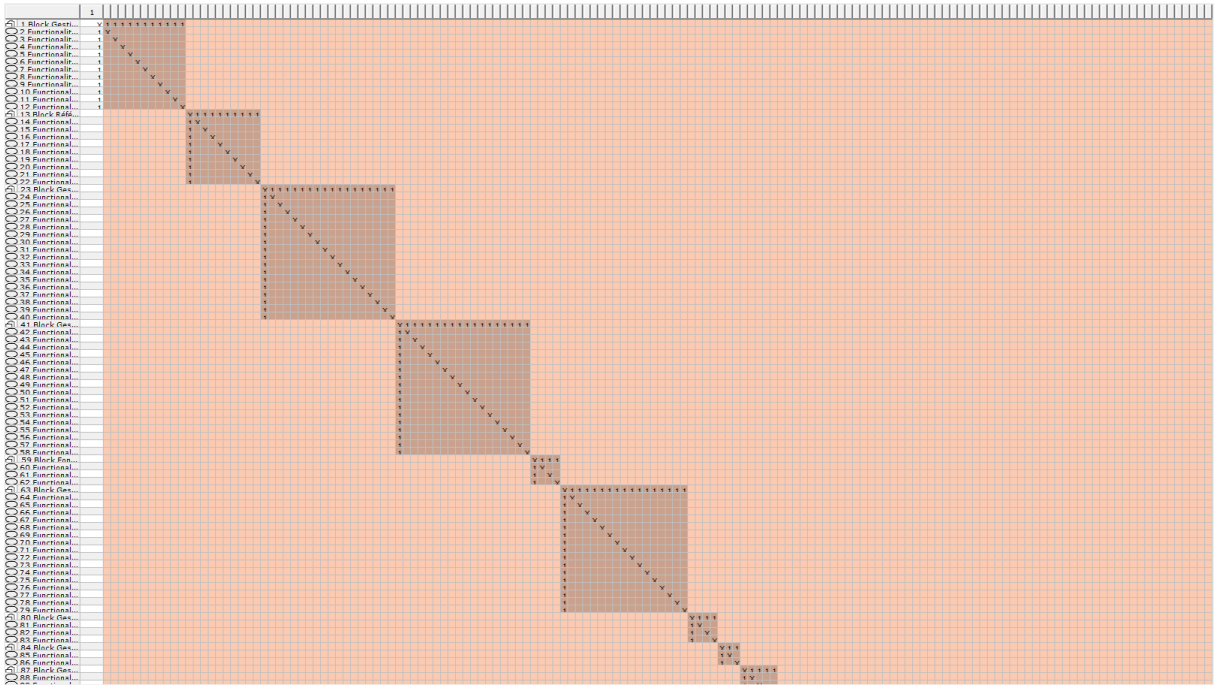


FIGURE 6.41 – Capture d'écran du modèle fonctionnel du cas SAMI après clustering

Pour améliorer les performances, d'autres pistes sont envisageables comme la multiplication de matrices creuses [109] (qui possèdent des cellules vides), mais ces pistes n'ont pour le moment pas été explorées.

Notre matrice permet d'afficher un modèle (un point de vue) et les dépendances des éléments de ce modèle. Cependant, notre but est de pouvoir afficher le résultat de l'alignement entre plusieurs points de vue. Nous présentons les améliorations pour intégrer plusieurs modèles dans notre éditeur dans la section suivante.

6.4.4 Matrice multi-domaines

Nous souhaitons afficher plusieurs modèles (BPM, Fun, App) au sein d'une même matrice afin d'afficher les dépendances résultant de l'alignement des modèles. Dans la littérature

cette matrice s'appelle *Multiple-Domain Matrix* (**MDM**) [12], c'est une extension du DSM.

La première version n'est pas conçue pour charger plusieurs modèles, et encore moins pour supporter des éléments provenant de méta-modèles différents. Nous avons donc apporté différentes améliorations dans notre éditeur :

Support multi-domaines Nous avons ajouté un item dans le menu contextuel de notre éditeur pour charger plusieurs modèles sur une même matrice. Chaque modèle possède une couleur différente et paramétrable pour faciliter la lecture de la matrice.

Support de EMF Facet Le résultat de l'alignement est enregistré dans un fichier de sérialisation appelé **EFS**, nous avons ajouté le support de EMF Facet à la matrice ainsi que le chargement de fichier EFS.

Ces deux nouvelles fonctions permettent déjà de visualiser les dépendances entre éléments de l'alignement. Par exemple, sur la figure 6.42 la matrice permet de visualiser le résultat d'alignement entre des blocs du point de vue fonctionnel et des composants applicatifs du point de vue applicatif tiré du cas d'étude SAMI. Nous avons limité le périmètre de SAMI à 6 blocs et 24 fonctionnalités du point de vue fonctionnel en rouge, et 62 services et 1 composant applicatif du point de vue applicatif en jaune. Au lieu de respectivement 18 blocs, 131 fonctionnalités, 11 894 services et 624 composants applicatifs de la totalité des points de vue (cf. table 5.3). Un trop grand nombre d'éléments à étudier engendre des difficultés de lecture et d'affichage de la matrice en totalité.

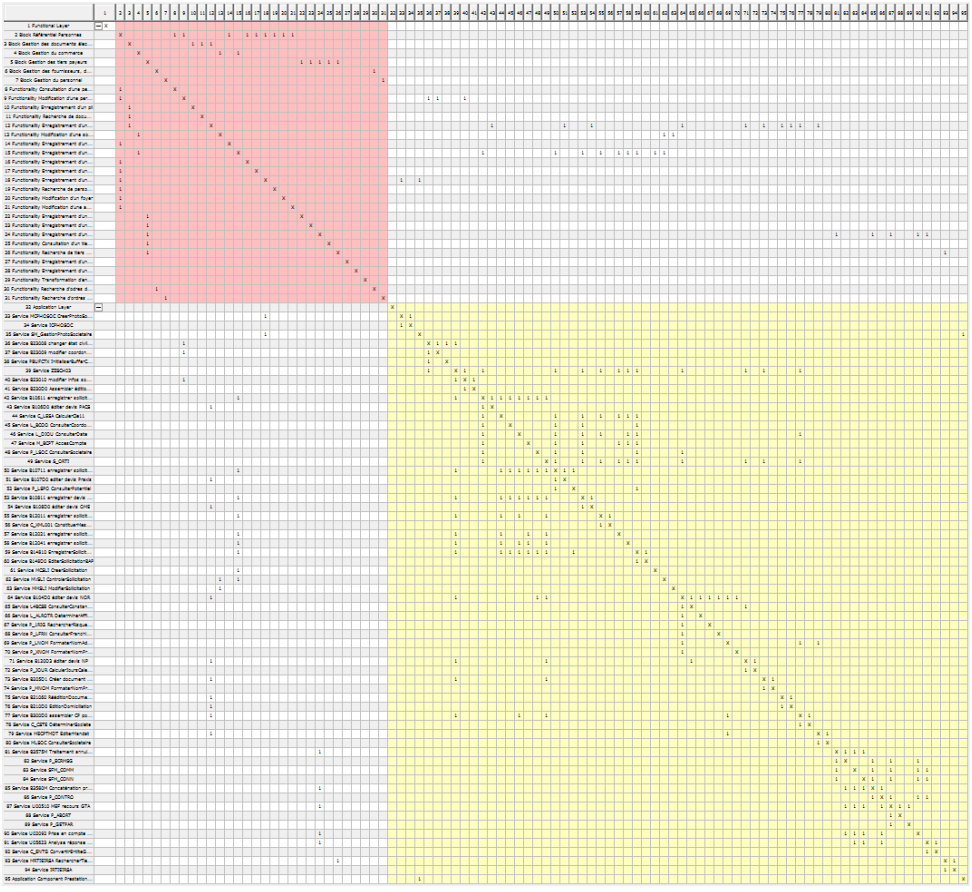


FIGURE 6.42 – Capture d'écran de l'alignement entre les points de vue fonctionnel et applicatif de SAMI

Néanmoins, un problème se pose pour réaliser le clustering multi-domaines. Nous considérons que la matrice multi-domaines est composée de sous-matrices au nombre de

d^2 où d est le nombre de domaines différents. Par exemple en figure 6.43 la matrice est chargée avec trois domaines correspondants aux trois points de vue de notre définition de l'alignement (BPM, Fun et App).

	Fun	BPM	App
Fun	Fun	Fun / BPM	Fun / App
BPM	BPM / Fun	BPM	BPM / App
App	App / Fun	App / BPM	App

FIGURE 6.43 – Matrice multi-domaines avec les points de vue Fun, BPM et App

La matrice est composée de neuf sous-matrices, dont on définit deux types de sous-matrices :

Sous-matrice de domaine Il s'agit des trois matrices : BPM, Fun, et App.

Sous-matrice inter-domaines Il s'agit des six matrices correspondantes aux couples de domaines : Fun / BPM, Fun / App, BPM / Fun, etc.

Le clustering ne peut s'effectuer que sur une matrice adjacente : mêmes nœuds en colonne et en ligne et donc même taille en ligne et en colonne. Le clustering de la matrice a une influence sur l'ordre des nœuds et donc des différentes matrices de domaine qui partagent la même ligne et colonne. Or les matrices inter-domaines ne sont pas nécessairement carrées et ne sont pas adjacentes. Le clustering inter-domaines semble à première vue impossible, et les travaux autour de ce sujet dans l'état de l'art ne proposent pas de solution. Il est mathématiquement impossible d'effectuer le clustering sur des matrices non adjacentes.

Dans la suite, nous proposons trois types de clustering pour analyser les dépendances entre domaines : global, domaine, inter-domaines.

Clustering global

Le clustering gobal s'effectue sur toute la matrice, comme illustré par la figure 6.44 les dépendances des sous-matrices de domaine et inter-domaines sont prises en compte dans le calcul du clustering.

	Fun	BPM	App
Fun	Fun	Fun / BPM	Fun / App
BPM	BPM / Fun	BPM	BPM / App
App	App / Fun	App / BPM	App

FIGURE 6.44 – Ensemble des dépendances analysées par le clustering global

Le clustering ne fait pas de distinction entre sous-matrices de domaine ou inter-domaines, et considère la matrice dans son ensemble.

EXPÉRIMENTATION 6.1 – Clustering global sur le cas SAMI

Le clustering s'effectue sur les points de vue fonctionnel et applicatif du cas SAMI, ainsi que le résultat de l'alignement par facettes. Avant de "zoomer" sur une partie détaillée en figure 6.46, voici un aperçu général en figure 6.45.

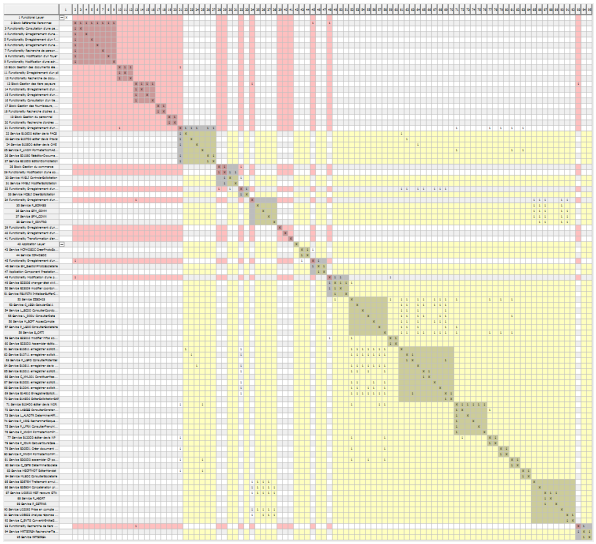


FIGURE 6.45 – Clustering global sur le cas SAMI

Le résultat est composé de 27 clusters dont 4 boucles de nœuds sur la diagonale, le plus grand est composé de 10x10 nœuds. Les clusters mélangeant du rouge (point de vue fonctionnel) et du jaune (point de vue applicatif) signifient que les points de vue sont en dépendance. On remarque sur la figure 6.45 que seulement 6 clusters ont des dépendances inter-domaines.

En agrandissant le résultat en figure 6.46, on analyse les dépendances à l'aide des clusters en surbrillance. Par exemple au niveau des nœuds 28 à 31, les services "*MVSLI ControlerSollicitation*" et "*MMSLI ModifierSollicitation*" dépendent de la fonctionnalité "*Modification d'une sollicitation*" elle-même dépendante du bloc "*Gestion du commerce*".

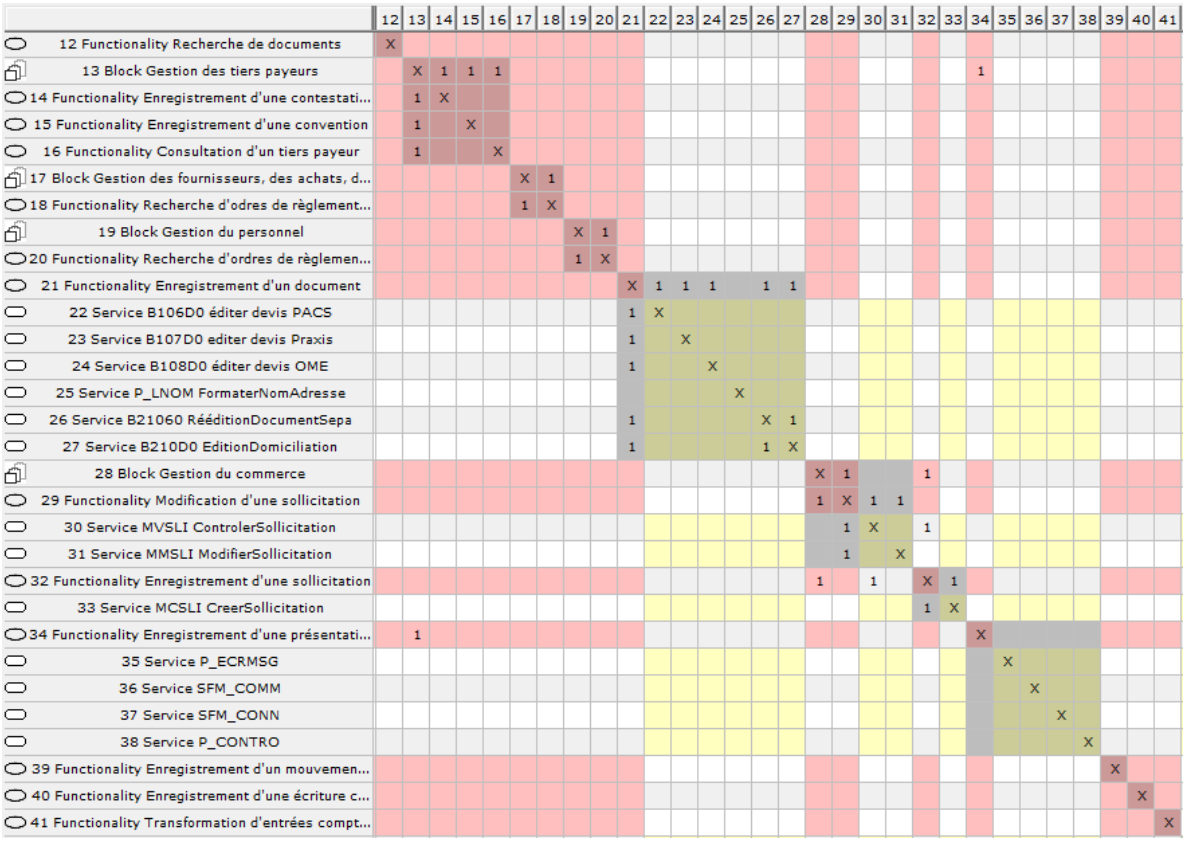


FIGURE 6.46 – Agrandissement du résultat de clustering global sur le cas SAMI

Le clustering global permet d'identifier toutes les dépendances, qu'elles soient des relations de modèle ou des relations inter-modèles par les facettes. Dans la suite, nous allons réaliser un clustering intra-domaines pour visualiser uniquement les dépendances des relations de modèle.

Clustering intra-domaines

Le clustering intra-domaines s'effectue uniquement sur les dépendances de chaque sous-matrice de domaine. Comme illustré par la figure 6.49, les dépendances inter-domaines ne sont pas prises en compte.

EXPÉRIMENTATION 6.2 – Clustering intra-domaines sur le cas SAMI

Le clustering s'effectue sur les points de vue fonctionnel et applicatif du cas SAMI, le résultat de l'alignement par facettes est affiché, mais non utilisé. Nous donnons un aperçu du clustering sur la figure 6.47 avant de "zoomer" sur une partie en figure 6.48

Le résultat sur la figure 6.47 est composé de 21 clusters dont 7 boucles de nœuds sur la diagonale, le plus grand est composé de 11x11 nœuds. Tous les clusters sont indépendants, ils sont soit rouges (point de vue fonctionnel), soit jaunes (point de vue applicatif).

En agrandissant le résultat en figure 6.48, on analyse les dépendances à l'aide des clusters en surbrillance. Par exemple au niveau des nœuds 19 à 24, le bloc "*Gestion des tiers payeurs*" a en dépendance 5 fonctionnalités.

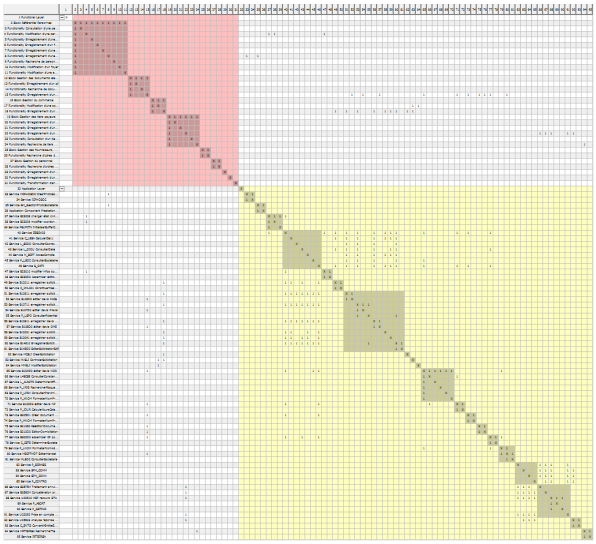


FIGURE 6.47 – Clustering intra-domaines sur le cas SAMI

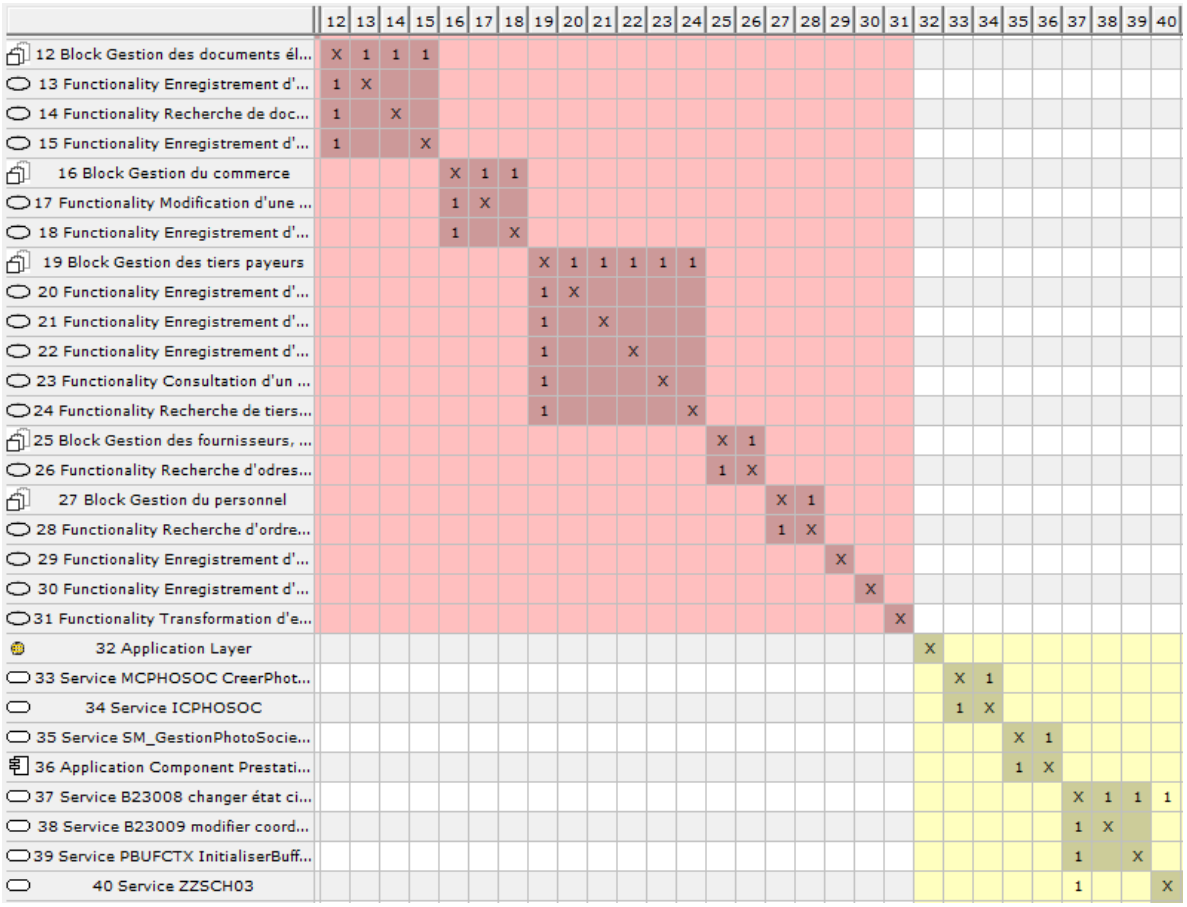


FIGURE 6.48 – Agrandissement du résultat de clustering intra-domaines sur le cas SAMI

Le clustering s'exécute sur chaque sous-matrice de domaine : si une matrice est composée de n domaines n_D, le clustering est réalisé n fois.

Le clustering intra-domaines permet d'identifier toutes les dépendances de relations des modèles. Dans la suite, nous allons réaliser un clustering inter-domaines pour visualiser uniquement les dépendances entre les modèles réalisées par l'alignement par facettes.

	Fun	BPM	App
Fun		Fun / BPM	Fun / App
BPM	BPM / Fun		BPM / App
App	App / Fun	App / BPM	

FIGURE 6.49 – Ensemble des dépendances analysées par le clustering intra-domaines

	Fun	BPM	App
Fun	Fun		
BPM		BPM	
App			App

FIGURE 6.50 – Ensemble des dépendances analysées par le clustering inter-domaines

Clustering inter-domaines

Comme illustré par la figure 6.50, le clustering inter-domaines prend en compte uniquement les dépendances entre les domaines, c'est la différence entre les clustering global et de domaines.

Pour réaliser le clustering inter-domaines nous appliquons un *masque* pour prendre en compte uniquement les dépendances inter-domaines. Puis, nous appliquons le résultat du clustering sur la matrice initiale.

EXPÉRIMENTATION 6.3 – Clustering inter-domaines sur le cas SAMI

Le clustering s'effectue sur le résultat de l'alignement par facettes entre les points de vue fonctionnel et applicatif du cas SAMI. Nous donnons un aperçu du clustering sur la figure 6.51 avant de "zoomer" sur une partie en figure 6.52.

Le résultat en figure 6.51 est composé de 64 clusters, seulement 7 clusters étant représentatifs, les autres étant des boucles de nœuds sur la diagonale. Le plus grand cluster est composé de 11x11 nœuds. Les 7 clusters mélangent du rouge (point de vue fonctionnel) et du jaune (point de vue applicatif) car les points de vue sont en dépendance.

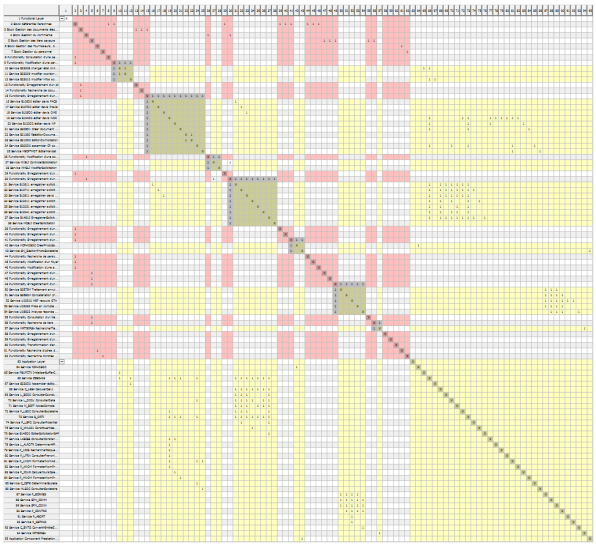


FIGURE 6.51 – Clustering inter-domaines sur le cas SAMI

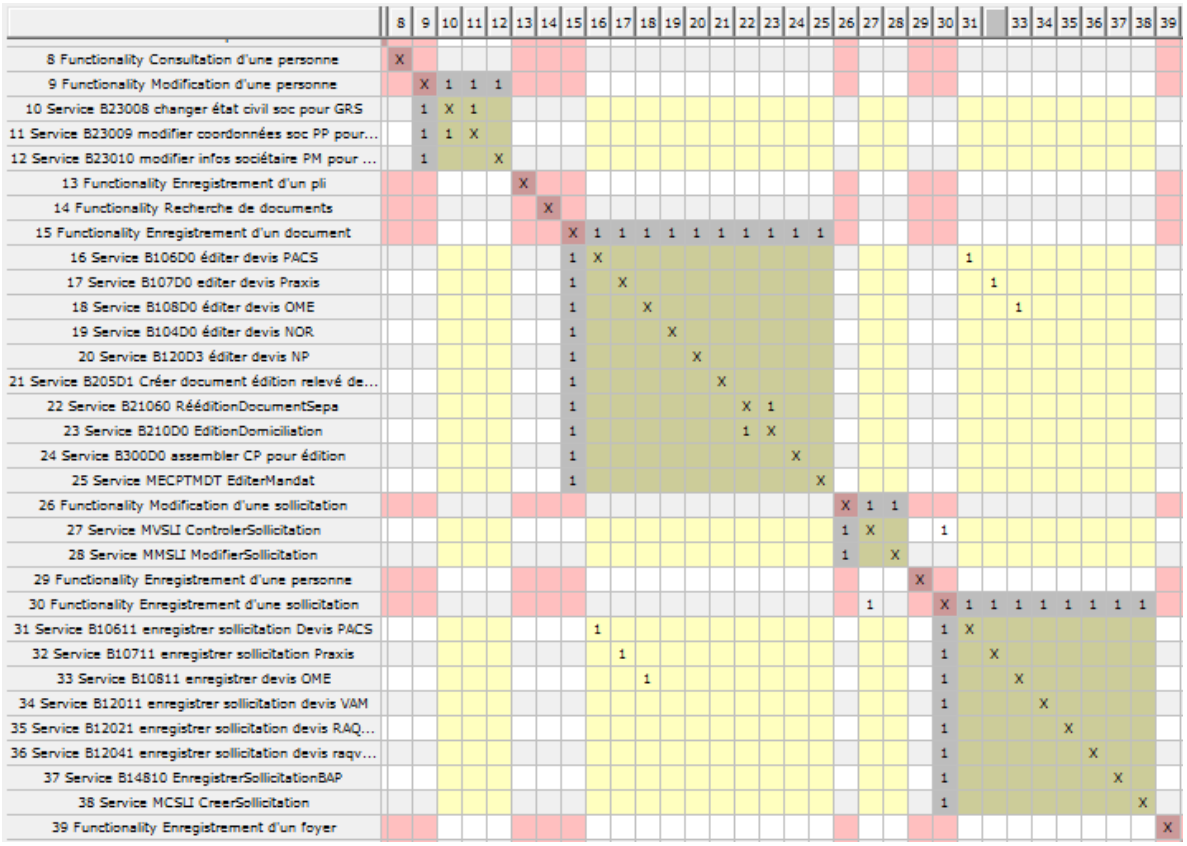


FIGURE 6.52 – Agrandissement du résultat de clustering inter-domaines sur le cas SAMI

Le clustering inter-domaines permet d'identifier toutes les dépendances de relations entre les modèles à l'aide des facettes.

Dans la section suivante, nous dressons un bilan des trois types de clustering et de leur niveau de préoccupation.

Bilan des expérimentations de clustering

TABLE 6.10 – Bilan par type de clustering sur le cas SAMI

Type	nb clusters	nb boucles	diff	taille max
Global	27	4	23	10x10
Intra-domaines	21	7	14	11x11
Inter-domaines	64	57	7	11x11

Les trois types de clustering (global, intra-domaines et inter-domaines) permettent un point de vue différent sur les dépendances intra et inter domaines. Le clustering global est le plus complexe puisqu'il mélange clusters intra et inter-domaines ; le clustering inter-domaines permet de visualiser facilement les dépendances de l'alignement. Ainsi, selon l'analyse souhaitée l'architecte d'entreprise aura le choix entre les différents types de clustering.

Le cas SAMI a été réduit à 95 nœuds ; la table 6.10 permet de comparer le résultat des différents types de clustering sur SAMI. On remarque que l'ordre de grandeur de la taille des clusters est similaire. La colonne *diff* est la différence entre le nombre de

clusters et de boucles, il s'agit donc du nombre de clusters *utiles*. La somme du nombre de clusters utiles résultant des clusterings intra-domaine et inter-domaines est du même ordre de grandeur que le nombre de clusters utiles du clustering global. Les deux types de clustering intra-domaines et inter-domaines sont complémentaires.

6.5 Une interprétation des indicateurs pour l'aide à la décision

Les indicateurs que nous avons conçus aident l'architecte à comprendre l'état du SI pour décider des actions de transformation à réaliser. Mais ceci ne constitue pas un **scénario solution**. L'interprétation du résultat des indicateurs est nécessaire par l'architecte pour élaborer le ou les scénarios solutions.

Affichage du résultat des indicateurs dans les modèles

Une première amélioration serait d'inclure le résultat de nos indicateurs directement dans les interfaces graphiques au plus proche des instances concernées pour visualiser nos modèles, nous donnons quelques exemples envisagés :

- **élément de l'alignement** : afficher une étiquette spécifique sur l'icône de l'instance en fonction de son état : non aligné (pas d'icône) ; non alignable ✗ ; aligné ✓.
- **taux d'alignement** : insérer une vue spécifique dans Eclipse qui donne une statistique instantanée du taux d'alignement sur les modèles ouverts ou une sélection d'instances ;
- **incohérence** : afficher une étiquette ⚠ "*attention*" sur les instances en incohérence entre les liens de donnée et de traitement ;
- **couplage** : afficher en étiquette sur les instances la valeur de son couplage avec d'autres instances par l'alignement (①, ②, ③, ④...) ;

Aide à l'alignement et analyse d'impact

Nous distinguons les indicateurs présentés dans le tableau de bord en section 6.2 selon deux types :

Assistance à l'élaboration du processus d'alignement : *nombre d'éléments de l'alignement*, *taux d'alignement*, *nombre d'éléments non alignés*, et *incohérence d'alignement* ;

Impact : *couplage*, et *matrice de dépendances*.

Les indicateurs d'assistance à l'élaboration de l'alignement sont principalement dédiés à assister l'architecte à conduire l'alignement : connaître les éléments alignés, non alignés, mal alignés ; et connaître l'état d'avancement de l'alignement par le taux et le nombre d'éléments. Ils répondent aux problématiques initiales de la thèse de modélisation du SI afin de réaliser une **traçabilité** et permettre une **navigation** entre les différents points de vue.

Les indicateurs de décision répondent à la problématique initiale suivante de la thèse : **évaluer l'impact** des changements selon les points de vue étudiés et **choisir un scénario** de changement.

L'architecte calcule cette analyse d'impact pour **décider** à l'aide de nos outils. Prenons l'exemple concret du cas SAMI expérimentation 6.3, l'analyse des dépendances à l'aide de la matrice extraite en figure 6.53, nous permet de connaître quels composants applicatifs implémentent la fonctionnalité "*Enregistrement d'un document*".

20 Functionality Enregistrement d'un document	X	1	1	1	1	1	1	1	1	1	1
21 Service B106D0 éditer devis PACS	1	X									
22 Service B107D0 editer devis Praxis	1		X								
23 Service B108D0 éditer devis OME	1			X							
24 Service B104D0 éditer devis NOR	1				X						
25 Service B120D3 éditer devis NP	1					X					
26 Service B205D1 Créer document édition r...	1						X				
27 Service B21060 RééditionDocumentSepa	1							X	1		
28 Service B210D0 EditionDomiciliation	1							1	X		
29 Service B300D0 assembler CP pour édition	1									X	
30 Service MECPTMDT EditerMandat	1										X

FIGURE 6.53 – Extrait du clustering de SAMI mettant en évidence les dépendances sur la fonctionnalité *Enregistrement d'un document*

Le scénario de transformation de SAMI faisant intervenir différents acteurs de l'entreprise pourrait être le suivant :

1. Demande par un responsable métier ou par un utilisateur d'une modification de la fonctionnalité "*Enregistrement d'un document*") ;
2. Analyse par l'architecte d'entreprise de l'alignement à l'aide de la matrice de dépendance. Détection du fait que la fonctionnalité 15 ("*Enregistrement d'un document*") dépend de dix services : 16, 17, 18, 18, 20 ("*Bxxxxx Éditer devis xxx*"), 21 ("*B205D1 Créer document...*"), 22 ("*B21060 Réédition Document Sepa*"), 23 ("*B210D0 Édition domiciliation*"), 24 ("*B300D0 Assembler CP pour édition*"), 25 ("*MECPTMDT Éditer mandat*") ;
3. Demande de l'architecte au responsable métier de spécifications sur les modifications à effectuer ;
4. Transmission des spécifications à un développeur pour analyser dans le code source des 10 services les méthodes à modifier ;
5. Remontée de l'analyse à l'architecte, il décide ou non de faire effectuer les modifications dans le code source au développeur ;
6. Modification effectuée, mise à jour des cartographies (numéro de composants, suppression ou ajout de nouveaux services, etc.).

Notre outil a permis de mettre en évidence par l'analyse d'impact que plusieurs services du point de vue applicatif sont dépendants d'une fonctionnalité du point de vue fonctionnel. L'architecte a pu décider et mettre en œuvre les actions de transformation. Mais il joue aussi le rôle de garant de la mise à jour de l'information dans les cartographies.

Conclusion

À l'issue de la réalisation de l'alignement par les modèles (cf. section 5.3.3), nous souhaitions exploiter l'alignement pour avoir une meilleure compréhension des points de vue du SI et réaliser sa transformation.

Le suivi de l'alignement est permis à l'aide d'un tableau de bord (cf. section 6.2) composé de différents indicateurs : nombre d'éléments qui composent l'alignement, taux d'alignement, nombre d'éléments non alignés, incohérence et couplage. Nous avons étendu notre définition de l'alignement en ajoutant un marqueur optionnel pour identifier les éléments non alignables (cf. section 6.1.2). Les indicateurs du tableau de bord sont calculés à l'aide de requêtes OCL exécutables et conformes à notre définition de l'alignement (cf. section 4) que nous détaillons en section 6.3.

Au-delà du suivi de l'alignement, nous pouvons le visualiser. Nous avons proposé un éditeur de matrices de dépendances que nous avons implémenté en section 6.4. En fonction du nombre d'éléments qui composent les différents points de vue du SI, les matrices peuvent avoir de grandes tailles, la lecture des dépendances est complexe. Nous avons proposé plusieurs types de clustering à l'aide de l'algorithme MCL pour regrouper les dépendances autour de la diagonale de la matrice. Enfin, nous avons proposé par l'exemple sur le cas SAMI un scénario de transformation du SI par interprétation de l'alignement à l'aide d'une analyse des dépendances.

7

Conclusion

Sommaire

7.1 Bilan

Différents **problèmes** touchent le sujet d'architecture du système d'information de l'entreprise et plus précisément l'alignement entre les visions métier et informatique, pour chacun d'entre eux nous avons proposé des **solutions** outillées.

Le premier problème est d'obtenir une image à jour et complète du SI, mais aussi de classer et organiser les différents types d'informations selon les préoccupations et les acteurs de l'entreprise. Nous avons identifié plusieurs points de vue du SI, néanmoins nous avons retenu uniquement trois points de vue à la frontière entre domaines métier et informatique. Nous avons défini leurs concepts à l'aide de méta-modèles représentant chaque point de vue : processus métier (BPM), fonctionnel (Fun), et applicatif (App).

Le deuxième problème est de réaliser la traçabilité descendante et ascendante entre les différents concepts des points de vue. Nous avons défini un alignement par les traitements et par les données à partir des concepts de nos méta-modèles. L'alignement se réalise techniquement à l'aide d'un tissage non intrusif par facettes qui consiste à créer des liens d'association virtuels entre les instances de concept. Nous avons développé un éditeur de tissage pour construire et exploiter l'alignement. À l'aide de nos outils, il est désormais possible de naviguer d'un point de vue à l'autre à travers les liens : par une interface graphique, et aussi par un éditeur de requêtes OCL.

Le troisième problème est de collecter les informations de chaque point de vue pour obtenir les modèles à aligner. Nous avons proposé un processus de rétro-ingénierie complet par étapes successives. En fonction du point de vue, la méthode est différente : pour obtenir le modèle applicatif plusieurs transformations de modèles sont réalisées à partir du code source pour abstraire des concepts applicatifs des concepts techniques ; pour obtenir les modèles métiers (processus et fonctionnels) une extraction et une traduction depuis un référentiel ou des diagrammes existants sont réalisées. Lorsqu'il n'y a aucune information, il faut faire appel aux connaissances des différentes parties prenantes de l'entreprise : analyste métier, sachant fonctionnel, décideur, architecte, développeur, etc. Des éditeurs de diagramme sont proposés dans nos outils pour réaliser la cartographie correspondant à chaque point de vue.

Le quatrième et dernier problème est d'évaluer l'état du SI et de décider des actions de transformation à mener. Nous proposons des outils d'analyse à travers des indicateurs affichés dans un tableau de bord. Deux types d'indicateurs sont proposés : des indicateurs d'élaboration du processus d'alignement, et des indicateurs d'impact. Les indicateurs d'alignement permettent de connaître l'état des éléments qui composent l'alignement, le taux d'avancement, point de vue par point de vue et par catégorie (traitement et donnée), ainsi que les incohérences. Les indicateurs d'impact permettent de réaliser une analyse d'impact du SI des points de vue métier à informatique, et vice-versa. Chaque indicateur est défini par une requête permettant le calcul d'un nombre ou liste d'éléments. Pour une analyse d'impact affinée, nous avons créé un éditeur sous forme de matrice de dépendances avec un algorithme de calcul de regroupements.

Nos différents outils et méthodes assistent l'architecte pour réaliser l'analyse d'impact à partir de l'alignement et pour élaborer un scénario de changement du SI. L'analyse du SI et sa transformation permettent d'obtenir un SI flexible répondant aux exigences de développement d'une entreprise : optimiser les activités productives, faire face à la concurrence en "*juste à temps*", obtenir un retour sur investissement, prendre en compte le cadre légal, et s'adapter aux nouvelles technologies.

7.2 Contributions

Les travaux de la thèse, outre l'élaboration d'une démarche complète basée sur des modèles pour l'alignement, se sont concrétisés par l'élaboration de prototypes logiciels qui sont maintenant expérimentés par l'entreprise Sodifrance. Les travaux ont également fait l'objet de diffusion de résultats via des publications décrivant les différentes méthodes et présentant les prototypes.

7.2.1 Publications

Durant les trois années de thèse, les travaux ont été publiés au fur et à mesure de leur avancement dans plusieurs ateliers et conférences ; nous avons soumis un article à une revue. Nous donnons le détail de chaque intervention et faisons un récapitulatif dans le tableau 7.1.

JDOC *Journée de DOCtorants de l'école doctorale* STIM permettant la rencontre des doctorants par une présentation des travaux avec un poster.

CIEL / GdR GPL Organisées durant la même semaine, la *Conférence en IngénieriE du Logiciel* et les journées du *Groupement de Recherche Génie de la Programmation et du Logiciel*.

ICEIS 17ème édition de la conférence internationale sur les systèmes d'informations d'entreprise (*International Conference on Enterprise Information Systems*).

RIMEL Journée de travail du groupe de *Rétro-Ingénierie, Maintenance & Evolution de Logiciels* du GdR GPL dans le cadre de la conférence Evolille.

CAiSE 28ème édition de la conférence internationale sur l'ingénierie des systèmes d'information avancés (*International Conference on Advanced Information Systems Engineering*).

HCRES Journée d'évaluation du *Laboratoire Informatique de Nantes Atlantique* (LINA) par le *Haut Conseil de l'évaluation de la Recherche et de l'Enseignement Supérieur*.

CSIMQ Journal trimestriel de publications à comité de lecture sur les systèmes complexes d'informatique et de modélisation (*Complex Systems Informatics and Modeling Quarterly*).

TABLE 7.1 – Récapitulatif des publications et communications

Date	Évnmnt	Titre	Article	Poster	Démo	Lieu
2014	**JDOC**	Architecture d'entreprise : alignement des points de vue métier et applicatif par transformation et tissage de modèles	x	x		Nantes, France
	CIEL	Alignement de modèles métiers et applicatifs : une approche pragmatique par transformations de modèle	x			Paris, France
	GdR GPL	Enterprise Architecture : Can Business Models be Aligned with IT ?		x	x	
2015	**ICEIS**	A Method for Business-IT Alignment of Legacy Systems	x	x		Barcelone, France
	RIMEL	Alignement de modèles pour l'évolution de patrimoines applicatifs	x			Lille, France
2016	**CAiSE**	A Facet-based Model Mapping Method for EA Alignment and Evolution	x	x	x	Ljubljana, Slovénie
	HCRES	Enterprise Architecture : Can Business Models be Aligned with IT ?		x	x	Nantes, France
	CSIMQ	An Improved Model Facet Method to Support EA Alignment and Evolution	x			Riga, Lettonie

La figure 7.1 illustre le poster présenté durant la conférence CAiSE, il montre notre méthode utilisant les facettes : implémentation de définition de l'alignement à partir des trois méta-modèles (BPM, Fun et App) à l'aide de EMF Facet, et les outils développés.

CAISE FORUM 2016 - Ljubljana, Slovenia

A Facet-based Model Mapping Method for EA Alignment and Evolution

Jonathan Pepin[1,2], Pascal André[1], Christian Attiogbé[1] and Erwan Breton[2]

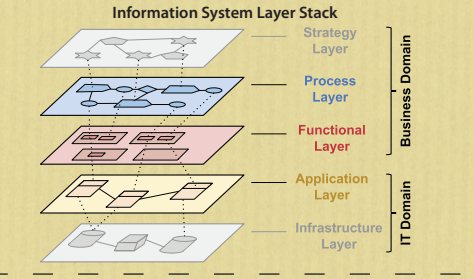
[1] AeLoS Team - LINA CNRS UMR 6241 - University of Nantes
{firstname.lastname}@univ-nantes.fr

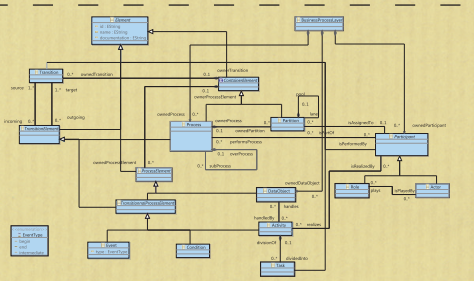
[2] Mia-Software Nantes
ebreton@sodifrance.fr

Information System Challenges

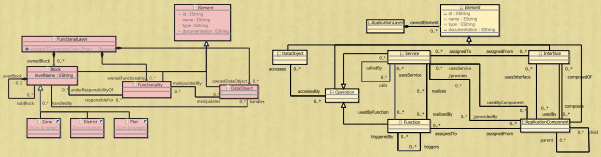
- Enterprise Architecture Layer's alignment & evolution
- Layering navigation
- Tool assistance
- Metrics oriented IS modernization

Three Meta-Models





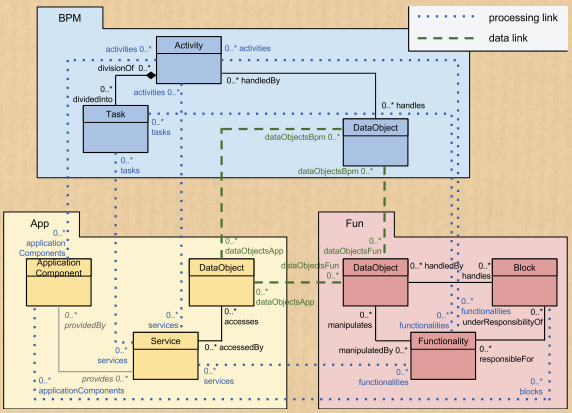
BPM : business processes (*actors, activities, tasks and data*)



Fun : functional organization (*blocks and data stores*)

App : software architecture (*components, services and data*)

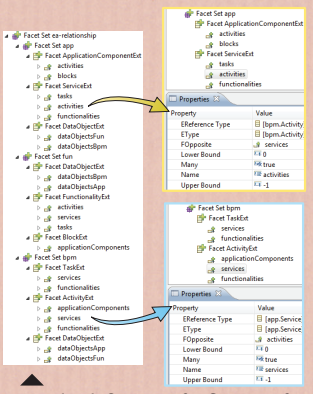
Alignment



- Cross links between model elements
- Separation of concerns : data and processes

Facet Definition

EMF Facet is open-source project to extend a meta-model (MM) without intrusion. It provide a mechanism to add virtually attributes and references to and between existing concepts.



Extending EMF Facet:
Originally Facet have limitations for weaving: virtual features calculated only by query, no manual assignment. We contribute to the project by modifying the MM and engine:
- manual values of FacetReference and FacetAttribute
- add reflexive reference fOpposite
- improve serialization mechanism

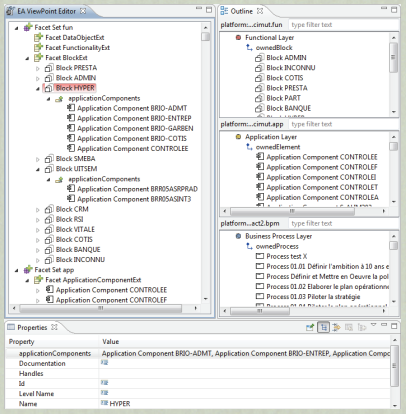
Example: definition of reflexive references *activities* and *services* between **Activity** from BPM and **Service** from App

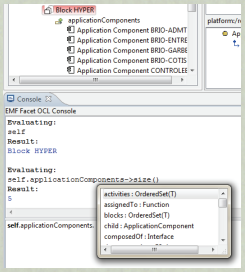
Tooling

Our developed Eclipse Plugin weaver assistant:

Right, models to weave, tree-like browser and quick search of concepts

Left, the result, creation of specific link by drag & drop using Facet definition





Contribution to EMF Facet project to extend OCL query engine:
- navigation in new Facet features
- auto-complete to assist query writing
- calculation and list from concepts

Results

- Comparison of model composition techniques according quality criteria
- Improvement of the selected approach (EMF Facet)
- Application to EA challenges and feasability provided by experimentations in real case studies

Perspectives

- Improving the assistance by a misalignment analyser with help from our OCL console
- Improving feedback: tracking and visualization with DSM

FIGURE 7.1 – Poster présenté à la conférence CAiSE 2016

7.2.2 Prototypes développés

Tout au long de cette thèse, l'élaboration de la méthode d'alignement s'est déroulée avec la création de nouveaux outils et l'amélioration de socles existants. De nombreux éditeurs avec interfaces graphiques ont été présentés, tous ont été développés sur la plateforme Eclipse. Chaque composant développé est un livrable sous forme d'un plug-in.

Les éditeurs graphiques développés

- **diag** : les trois éditeurs de diagrammes graphiques permettant de modéliser les points de vue conformes aux méta-modèles BPM, Fun et App ;
- **weav1** : l'éditeur reposant sur notre méta-modèle de tissage permettant la création de liens d'alignement par glisser-déposer, et filtre par fonction de recherche ;
- **weav2** : l'éditeur de tissage par facettes permettant la création de liens d'alignement par glisser-déposer ou par l'édition de propriétés, et filtre par fonction de recherche ;
- **ocl** : la console OCL compatible avec les extensions définies à l'aide de Facet. La console permet de construire des requêtes avec assistance par auto-complétion, puis de les exécuter et afficher le résultat ;
- **prpr** : la vue propriété permet d'éditer les valeurs des références et attributs de facette ;
- **dsm** : l'éditeur de matrice de dépendance (DSM) multi-domaines, permet d'afficher les dépendances entre modèles alignés en exploitant le résultat de l'alignement par facettes.

Les algorithmes développés

- **trfJ** : réécriture d'une transformation de code Java vers un modèle KDM ;
- **trfM** : écriture de transformations pour extraire les informations en provenance d'un référentiel Mega vers BPM, Fun et App ;
- **fcet** : modifications de EMF Facet pour supporter l'affectation de valeurs aux extensions de référence et d'attributs, et pour persister ces valeurs ;
- **mcl** : implémentation d'un algorithme de clustering MCL pour calculer le regroupement de dépendances dans la matrice DSM.

Choix des technologies

Le tableau 7.2 présente un récapitulatif de chaque développement. Comme évoqué précédemment, les éditeurs reposent sur la plateforme Eclipse. Nous avons utilisé des cadriciels supplémentaires qui offrent un environnement de développement enrichi, nous ne présentons pas à nouveau EMF Facet (cf. section 4.3.1), ni Mia-Transformation (cf. note 5.3.1).

EMF permet de créer des méta-modèles spécifiques et de générer le code correspondant en Java. La sphère EMF est très active et possède un panel large et complet d'outils disponibles pour exploiter les instances de méta-modèles conçus : les modèles.

GMF permet de créer des éditeurs graphiques de type diagramme pour créer des modèles conformes à une spécification de méta-modèle. GMF repose sur GEF pour gérer les objets du diagramme et Draw 2D pour le rendu graphique. GMF tend à disparaître

au profit du projet open source Sirius développé par Obeo, même si Sirius repose en partie sur le moteur GMF. Cependant, ce projet n'existait pas au début de la thèse.

Nebula permet de créer des éditeurs SWT plus performants que l'API standard. Nebula est adapté au traitement de volumes de données importants.

openBLAS est une bibliothèque open source implémentant des opérations d'algèbre linéaire optimisées et de hautes performances publiées en 1979. Les opérations de BLAS permettent l'addition de vecteurs, des produits scalaires ou des multiplications de matrices. C'est cette dernière opération que nous utilisons dans notre implémentation de l'algorithme MCL.

TABLE 7.2 – Récapitulatif des développements : socles techniques et licences

Dév.	socles	licence	publié	article
diag	Eclipse : EMF + GMF	propriétaire	✓	ICEIS
weav1	Eclipse EMF	propriétaire	✓	CIEL, ICEIS
weav2	Eclipse EMF Facet	propriétaire	✓	RIMEL, CAiSE, CSIMQ
ocl	Eclipse : EMF + EMF Facet	open source	✓	RIMEL, CAiSE, CSIMQ
prpr	Eclipse : EMF + EMF Facet	open source	✓	CAiSE, CSIMQ
dsm	Eclipse : EMF + Nebula	propriétaire	~	CAiSE, CSIMQ
trfJ	Mia-Transformation	propriétaire	✗	
trfM	Mia-Transformation	propriétaire	✗	
fcet	Eclipse EMF Facet	open source	✓	CAiSE, CSIMQ
mcl	Java + openBLAS	propriétaire	~	CAiSE, CSIMQ

✓ : publié en détail, ~ : évoqué en publication, ✗ : non publié

Les développements améliorant le cadriciel EMF Facet ont été soumis comme contributions au projet Eclipse, le code est open source. Le reste des développements est la propriété de Sodifrance. La plupart des développements ont illustré nos publications soit de façon détaillée, soit de manière plus succincte ; notamment pour le sujet autour de la matrice de dépendance DSM et l'algorithme DSM qui ont été abordés dans une publication, mais il manque un article spécifiquement autour de ce sujet.

7.3 Perspectives

Ce travail de thèse n'a pas pu aborder tous les points pour assister l'évolution du SI par son alignement, ni résoudre totalement les problèmes soulevés. Nous donnons quelques pistes à approfondir, mais aussi de nouvelles préoccupations qui découlent de nos travaux. Les perspectives sont à la fois scientifiques et industrielles.

7.3.1 Perspectives scientifiques

L'étude de l'état de l'art de l'architecture d'entreprise et plus précisément de l'alignement, nous a ouvert des pistes d'exploration. Cependant, nous avons dû nous contenter d'un

périmètre de problématiques à résoudre (cf. section 7.1). Ainsi, des améliorations sont envisagées, notamment sur l'extension de la cartographie et l'extension de l'alignement par la prise en compte de davantage de visions.

Vision stratégique par le modèle des affaires

Nous avons fait le choix de nous limiter aux trois points de vue au cœur du problème d'alignement entre domaine métier et informatique (BPM, APP, Fun). Il existe cependant un point de vue stratégique de plus haut niveau que les deux points de vues métier (processus et fonctionnel). Au cours de nos recherches, nous avons rencontré plusieurs notations stratégiques orientées affaires [55].

BMO *Business Model Ontologie* se traduit en ontologie de modèle d'affaires. Proposé par Osterwalder en 2004 [81], l'objectif de BMO est de modéliser le modèle d'affaires et sa logique de création de valeur pour prendre des décisions sur l'activité productive de l'entreprise. BMO est constitué de quatre piliers : produits et services, infrastructure et réseau partenaire, capital relationnel, et aspects financiers. Ainsi, le BMO représente les segments de marché cible, les clients et fournisseurs, la livraison et les moyens financiers et les flux de revenus durables. Les travaux de Osterwalder sont très connus par l'utilisation de son *Business Model Canvas* par les start-ups et illustré en figure 7.2.

E3Value Proposé par Gordijin en 2001 pour représenter la création, transfert et consommation de valeurs entre les réseaux d'acteurs économiques. Le principal objectif de cette modélisation est de répondre aux questions suivantes : qui offre ? à qui ? qu'est-ce qui est attendu en retour ?

REA Apparu en 1982 par McCarthy [1], *Resource Events Agents* est basé sur trois aspects permettant l'échange de : ressources économiques (produits, services, travail) ; évènements économiques (échanges entre acteurs, transferts et transformations) à la hausse et à la baisse ; agents économiques.

Nous donnons un comparatif des concepts des trois notations dans le tableau 7.3.

TABLE 7.3 – Comparaison des concepts des différents modèles d'affaires

	BMO	e^3value	REA
Offer Related	Value Proposition	Value In, Value Port	Economic Event
Customer Related	Target Customer, Distribution Channel, Relationship Mechanism	Actor	Partner
Network Related	Value Configuration, Resource / Core Capability, Partnership Agreement, Actor	Value Activity, Value Exchange, Value Object, Actor	Economic Resource, Duality
Finance Related	Revenue Stream, Cost Account	Profitability calculation	Economic Commitment, Economic Contract, Agreement

Le *Business Model Canvas* d'Alexander Osterwalder et d'Yves Pigneur [82] (cf. figure 7.2) est très populaire en réunion de travail pour modéliser une stratégie d'affaires. Il serait intéressant de traduire le canevas à l'aide de la notation BMO et l'introduire comme nouveau point de vue dans notre alignement.

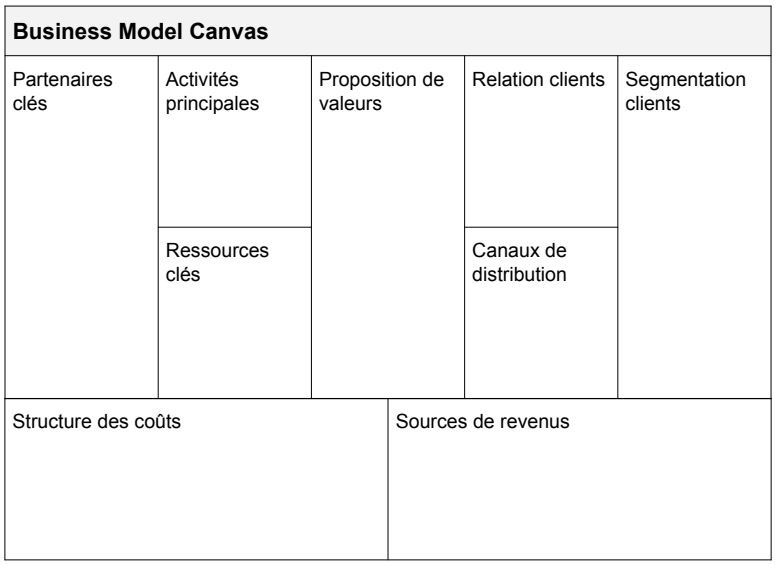


FIGURE 7.2 – *Business Model Canvas* simplifié [82]

Vision par les données

Au cours de la thèse, nous avons abordé le terme **référentiel d'entreprise** [13]. Nous avons exploité les informations contenues dans le référentiel du logiciel Mega Entreprise. Les référentiels sont utilisés au sein de grands groupes (assurance, banque, grande distribution, télécommunication, énergie) dans le cadre d'une cellule dédiée avec une véritable direction, où la tenue à jour de la cartographie du SI est un véritable défi : nombreuses filiales, situation géographique internationale, SI différents issus de fusions & acquisitions...

Les informations que nous avons extraites pour constituer les points de vue (processus, fonctionnel et applicatif) ne sont pas les seules présentes dans le référentiel d'entreprise. Plus largement, elles présentent des *données de références* qui ont une place distincte dans le SI. À contrario, dans les points de vue orientés traitement que nous avons proposés, où les données sont modélisées en même temps que les traitements. Les données de référence sont disséminées dans de multiples bases de données, voire répliquées avec une sémantique différente à différents endroits du SI dont les traductions sont rendues possibles grâce au *dictionnaire de données*.

Ce suivi et contrôle des données nécessiterait d'apporter un nouveau point de vue spécifique orienté données, à aligner avec notre méthodologie.

Visions technologiques

Nous avons fait le choix de n'intégrer que le point de vue applicatif dans notre méthode d'alignement ; néanmoins, il existe d'autres points de vue informatiques. Il serait possible de faire une extension, nous pensons notamment à deux points de vue technologiques décrits ci-après :

Déploiement ce point de vue a pour but de décrire comment sont déployées les applications au niveau logiciel. La vue applicative décrit les composants, fonctions ou objets de donnée, mais pas leur nature technologique. Par exemple, un ou plusieurs composants écrits en Java peuvent être encapsulés dans un *jar*, *war* ou *ear*[1] et

[1] Archive contenant des classes compilées et des fichiers de configuration

déployés dans un serveur d'application J2EE (Glassfish, JBoss, Apache Tomcat...) ; un ou plusieurs objets de données sont stockés dans une table de base de données (MySQL, Postgre, Oracle, Neo4j, MongoDB...).

Infrastructure ce point de vue a pour but de décrire le matériel où sont installés tous les logiciels, ainsi que le réseau et les interfaces pour les échanges d'informations. Ce point de vue modélise la situation géographique du matériel (pays > ville > site > bâtiment > salle > baie > unité) et peut donc couvrir plusieurs sites distants.

Certaines méthodes de modélisation d'architecture d'entreprise réunissent les deux descriptions, c'est le cas d'Archimate® dont nous présentons le méta-modèle du point de vue technologique[2] en version 3.0 sur la figure 7.3.

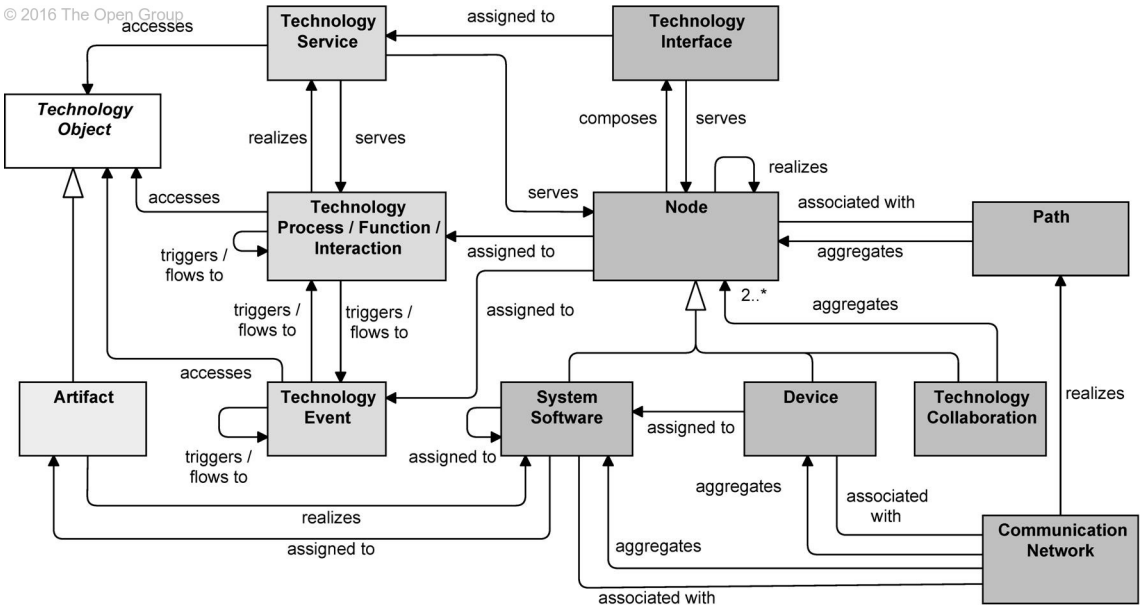


FIGURE 7.3 – Le méta-modèle du point de vue technologique de Archimate® 3.0 [94]

Traçabilité de l'alignement

Dans la section 5.3.1, nous obtenons le modèle applicatif à partir du code source par rétro-ingénierie. Ce processus est composé d'étapes de transformations successives avec un modèle en entrée et un modèle en sortie. Les modèles sont indépendants entre eux, et ne se référencent pas. Or la transformation est la traduction d'un élément source vers un élément cible conforme à un méta-modèle. Il serait possible d'enregistrer ce **lien de transformation** à l'aide d'une ***trace*** [3] pour garder un **historique** des transformations réalisées, et naviguer d'un modèle à un autre. Nous imaginons implémenter cette traçabilité à l'aide de EMF Facet. Chaque instance des modèles aurait deux nouvelles références : objet de transformation source, objet de transformation cible. En reprenant le scénario de transformation du cas SAMM, un objet "*ClassUnit*" du modèle KDM aurait comme *objet de transformation source* une "*ClassDeclaration*" (représentation de la classe Java) du modèle Java, et comme *objet de transformation cible* un composant applicatif "*ApplicationComponent*". Ainsi, il serait possible de **naviguer** du modèle Java jusqu'au modèle applicatif. Les liens de traçabilité de transformation peuvent être construits sans

[2] http://pubs.opengroup.org/architecture/archimate3-doc/chap10.html

étape supplémentaire au moment de l'exécution des scripts de transformation ; l'impact est faible sur le temps d'exécution.

Alignement par les usages

Notre méthode procède à un alignement **statique** de la cartographie du SI. Ce travail est réalisé principalement par les architectes qui auditent régulièrement les **besoins** des utilisateurs finaux. Or une cartographie ne reflète pas toujours la réalité. Les applications sont composées d'innombrables fonctionnalités, il est possible que certaines soient toujours maintenues alors qu'elles sont faiblement ou plus du tout utilisées. Pour vérifier l'utilisation des fonctionnalités, la solution serait d'injecter un **espion** [80] au niveau des traitements : appel de services, cinématique des IHM ; et aussi au niveau des données : requêtes en base de données. L'espion permettrait de conserver un **journal** d'historique[3] et un calcul permettrait d'ajouter un score d'utilisation aux composants de la cartographie applicative. Une analyse des cartographies par les **usages** permettrait de remonter au niveau des modèles métiers, les fonctions et les activités qui sont dépréciées par les utilisateurs. Deux scénarios sont envisageables :

- la fonctionnalité n'est plus utilisée car inutile à l'activité de l'entreprise, elle peut être supprimée et engendrer une économie de maintenance ;
- la fonctionnalité est utile à l'entreprise mais méconnue des utilisateurs, elle permet de développer ou d'accélérer l'activité de l'entreprise. Une formation doit être donnée pour obtenir un retour sur investissement.

Cette nouvelle dimension est un alignement **dynamique**, la cartographie pourrait être mise à jour périodiquement avec les données du Journal : jour, semaine, mois...

Réusinage de modèle par personnalisation

Le choix de la technologie EMF Facet pour réaliser l'alignement apporte également d'autres fonctionnalités comme la "Customisation" présentée dans la section 4.3.1. Cette fonctionnalité a du potentiel pour résoudre des problèmes de réusinage[4] de modèles, nous proposons deux scénarios :

- **Nommage des concepts** : en pratique, chaque entreprise a un vocabulaire différent et spécifique pour nommer les mêmes concepts. Dans le but de réaliser la cartographie, une entreprise peut choisir un bon méta-modèle mais qui a un lexique différent. Pour éviter toute mauvaise compréhension et éviter de changer les conventions existantes de l'entreprise, il est préférable de renommer les concepts impactés.
- **Classification** : avec l'augmentation du volume d'instances de concept dans un modèle, il est nécessaire de préciser une nouvelle classification des éléments par catégories. En termes de modélisation, nous voulons créer de nouvelles sous-classes par héritage. EMF Facet permet de créer ces nouvelles classes calculées à partir d'instances existantes par une requête, et d'afficher ces instances avec un nouveau label et une icône spécifique.

[3] *Logs* en anglais
[4] *Refactoring* en anglais

Ces deux scénarios pourraient être mis en œuvre par transformations de modèles statiques, mais l'avantage de EMF Facet permet d'éviter le problème d'intégrité du modèle entre deux versions différentes. Le réusinage par personnalisation est dynamique, il ne détruit pas le modèle original et il est possible de créer différents points de vue applicables à la demande.

Par exemple, supposons un modèle applicatif contenant beaucoup de composants "*ApplicationComponent*". Tous les composants ont le même nom de label et la même icône, mais ils ont un attribut "*type*" différent auquel on trouve deux valeurs différentes : **Application** ou **Component**. Ainsi, nous souhaitons réaliser une nouvelle classification pour distinguer les deux.

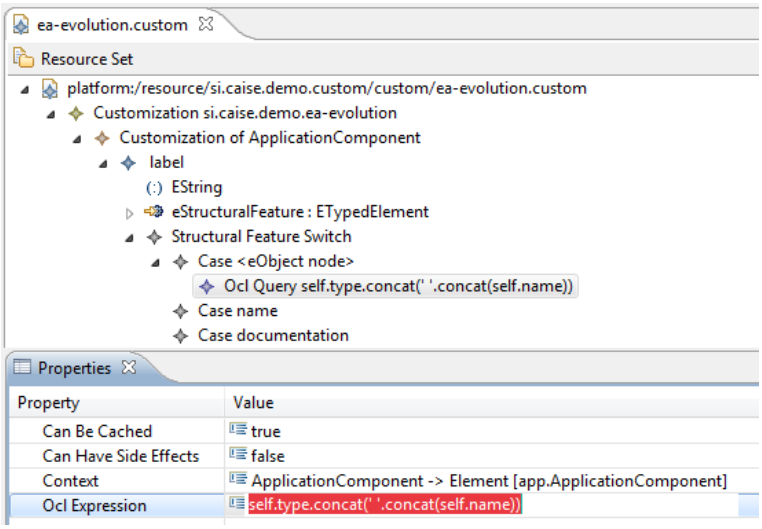


FIGURE 7.4 – Exemple de FacetSet implémentant le réusinage de classification

La figure 7.4 illustre comment implémenter à l'aide de EMF Facet une personnalisation. Ici, une requête OCL permet de calculer le nouveau nom de label à partir de l'attribut "*type*" et le nom original : type + ' ' + name. Une autre requête, celle-ci en Java, retourne la nouvelle icône pour les composants applicatifs avec comme valeur d'attribut "*type*" : application. Le résultat est présenté sur la figure 7.5.

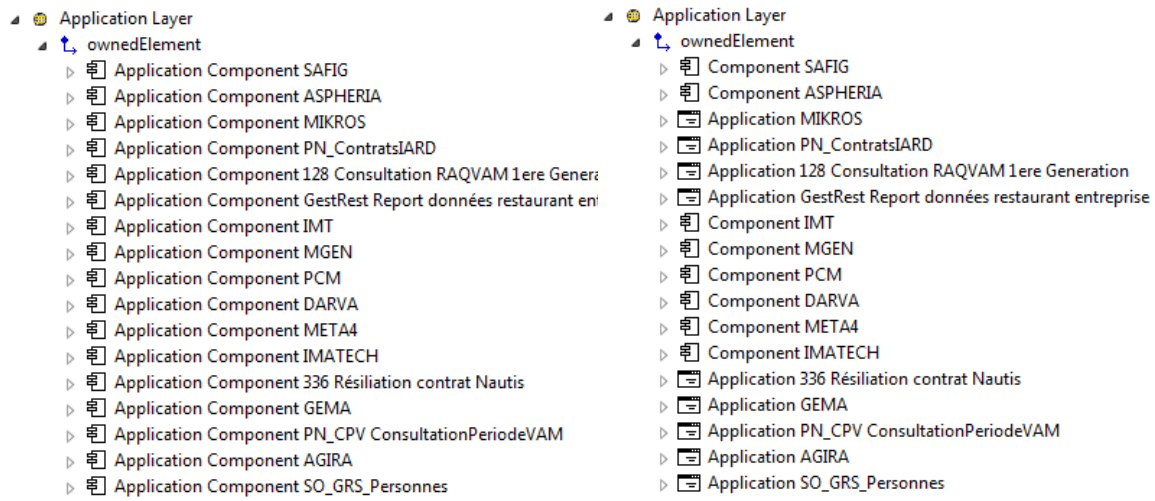


FIGURE 7.5 – Résultat du réusinage

Cette démonstration de faisabilité du réusinage par personnalisation de concepts a été réalisée lors de la conférence CAiSE.

7.3.2 Perspectives industrielles

Tous les outils ont été développés uniquement dans le but d'expérimenter la méthodologie de la thèse. Les développements désignés comme propriétaires sont donc des prototypes non commercialisés par Sodifrance.

La filiale de Sodifrance appelée Mia-Software®(logo en figure 7.6) est spécialisée dans l'édition de logiciels de modernisation de patrimoine d'entreprise. Elle intervient également directement auprès du client en prestation pour étudier et assister à la mise en place des solutions Mia-Software. Une des offres les plus importantes est l'outil de cartographie applicative Mia-Discovery.

Chaque client ayant des technologies et architectures différentes, Mia-Discovery a besoin d'un "chargeur" spécifique par technologie et des adaptations selon le scénario du client. Ces dernières années, les clients souhaitent connecter la cartographie applicative avec des modèles d'urbanisation. Actuellement, Mia-Discovery permet de cartographier le patrimoine applicatif, mais n'a pas de notion orientée architecture d'entreprise. Une première version rudimentaire propose la création de "lots" pour élaborer une pseudo cartographie fonctionnelle, un lot s'apparente à un bloc fonctionnel.

Ainsi, Mia-Software a la volonté de répondre aux souhaits de ses clients en proposant de nouvelles fonctionnalités répondant à des problématiques d'architecture d'entreprise. La feuille de route[5] du logiciel Mia-Discovery prévoit déjà d'intégrer des fonctionnalités issues de ces travaux de thèse.

Le premier objectif à court terme est de proposer un point de vue orienté fonctionnel similaire au méta-modèle de la thèse, avec alimentation par source externe (précisément un chargeur depuis un référentiel MEGA) et alignement avec le reste du patrimoine. En fonction des demandes et retours clients utilisateurs, nous ajouterons progressivement le point de vue processus métier, les indicateurs par requêtes, et la matrice de visualisation des dépendances.

Dans une perspective à plus long terme, la notion d'évolution du SI est un thème à approfondir. Nous avons déjà introduit la temporalité dans le "tableau de bord de l'alignement" en suivant l'évolution du nombre d'éléments alignés entre plusieurs itérations de l'alignement. Pour poursuivre ce sujet, il serait intéressant d'élaborer une méthodologie et des outils pour comparer plusieurs alignements de modèles, notamment par analyse d'écart.

FIGURE 7.6 – Logo de la filiale Mia-Software®

En marge des outils, cette thèse a permis à Sodifrance de monter davantage en compétences sur les notions d'architecture d'entreprise. Avec le concours des architectes seniors, la société pourra répondre plus amplement aux problématiques de modernisation du SI dans sa globalité, et potentiellement d'élargir son offre.

[5] *Roadmap* en anglais

Bibliographie

[1] Mohannad Al Jallad. *REA Business Modeling Language : Toward a REA based Domain Specific Visual Language*. 2012. 195

[2] Simon Allier, Houari Sahraoui, Salah Sadou, and Stéphane Vaucher. Restructuring Object-Oriented Applications into Component-Oriented Applications by Using Consistency with Execution Traces. volume 6092, pages 216–231, June 2010. 128

[3] Bastien Amar, Jean-Rémy Falleri, Marianne Huchard, Clementine Nebut, and Hervé Leblanc. Un Framework de traçabilité pour des transformations à caractère impératif. volume RNTI-L-1, pages 139–152. Cépadues Edition, March 2008. 197

[4] Victor Anaya and Angel Ortiz. How enterprise architectures can support integration. pages 25–30. ACM, November 2005. 142

[5] Pascal André, Nicolas Anquetil, Gilles Ardourel, Jean-Claude Royer, Petr Hnetynka, Tomás Poch, Dragos Petrascu, and Vladiela Petrascu. JavaCompExt - Extracting Architectural Elements from Java Source Code. pages 317–318. IEEE, October 2009. 128

[6] Gonçalo Antunes, Marzieh Bakhshandeh, Rudolf Mayer, José Borbinha, and Artur Caetano. Using Ontologies for Enterprise Architecture Integration and Analysis. *Complex Systems Informatics and Modeling Quarterly*, 0(1) :1–23, March 2014. 143

[7] Florence Arson. La cartographie, outil de pilotage de l'évolution des systèmes d'information. 2005. 26

[8] Colin Atkinson, Ralph Gerbig, and Mathias Fritzsche. A multi-level approach to modeling language extension in the Enterprise Systems Domain. *Information Systems*, 54 :289 – 307, 2015. 141

[9] Lerina Aversano, Carmine Grasso, and Maria Tortorella. A Literature Review of Business/IT Alignment Strategies. In José Cordeiro, Leszek A. Maciaszek, and Joaquim Filipe, editors, *Enterprise Information Systems*, number 141 in Lecture Notes in Business Information Processing, pages 471–488. Springer Berlin Heidelberg, January 2013. 47

[10] David Avison, Jill Jones, Philip Powell, and David Wilson. Using and validating the strategic alignment model. *The Journal of Strategic Information Systems*, 13(3) :223–246, September 2004. 49

[11] Elizabeth White Baker and Fred Niederman. Integrating the {IS} functions after mergers and acquisitions : Analyzing business-IT alignment. *The Journal of Strategic Information Systems*, 23(2) :112 – 127, 2014. 49

[12] J. Bartolomei, M. Cokus, J. Dahlgren, R. de Neufville, D. Maldonado, and J. Wilds. Analysis and applications of design structure matrix, domain mapping matrix, and engineering system matrix frameworks. 2010. 180

[13] Joël Bizingre, Joseph Paumier, and Pascal Rivière. *Les référentiels du système d'information : Données de référence et architectures d'entreprise*. Dunod, July 2013. 92, 196

[14] Vivien Bression, Florence Dietsch, and Stéphane Rouhier. Referentiels de la DSI - Etat de l'art - Usages et bonnes pratiques, 2009. 30

[15] Sylvain Brohée and Jacques van Helden. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics*, 7(1) :488, November 2006. 175

[16] Hugo Brunelière and Grégoire Dupé. Virtual EMF - Transparent Composition, Weaving and Linking of Models. In *EclipseCon Europe 2011*, November 2011. 99

[17] Janice M. Burn and Colonel Szeto. A comparison of the views of business and IT management on success factors for strategic alignment. *Information & Management*, 37(4) :197–216, 2000. 49

[18] Jean Bézivin. Model Driven Engineering : An Emerging Technical Space. In *Proceedings of the 2005 International Conference on Generative and Transformational Techniques in Software Engineering*, GTTSE'05, pages 36–64, Berlin, Heidelberg, 2006. Springer-Verlag. 140

[19] Jordi Cabot and Martin Gogolla. Object Constraint Language (OCL) : A Definitive Guide. In *Proceedings of the 12th International Conference on Formal Methods for the Design of Computer, Communication, and Software Systems : Formal Methods for Model-driven Engineering*, SFM'12, pages 58–90, Berlin, Heidelberg, 2012. Springer-Verlag. 113

[20] Jérôme Capirossi. *Architecture d'entreprise*. Hermes Science Publications, Paris, 2011. 34

[21] Yves Caseau. *Urbanisation, SOA et BPM - 4ème édition - Le point de vue du DSI*. Dunod, 4e édition edition, August 2011. 14, 24, 34, 38, 48

[22] Thierry Chamfrault and Claude Durand. *Les services agiles et la gouvernance des SI - Gouvernance et cycle de vie*. Dunod, Paris, September 2011. 30, 32

[23] Sylvain Chardigny. *Extraction d'une architecture logicielle à base de composants depuis un système orienté objet. Une aproche par exploration*. phdthesis, Université de Nantes, October 2009. 128

[24] Suwatana Charoensuk, Winai Wongsurawat, and Do Ba Khang. Business-IT Alignment : A practical research approach. *The Journal of High Technology Management Research*, 25(2) :132–147, 2014. 47

[25] Hong-Mei Chen, Rick Kazman, and Aditya Garg. BITAM - An engineering-principled method for managing misalignments between business and IT architectures. *Science of Computer Programming*, 57(1) :5–26, July 2005. 47

[26] Willy Chen, Claudia Hess, Melanie Langermeier, Janno Stuelpnagel, and Philipp Diefenthaler. Semantic Enterprise Architecture Management :. pages 318–325. SciTePress - Science and and Technology Publications, 2013. 141

[27] Elliot J. Chikofsky and James H. Cross II. Reverse Engineering and Design Recovery : A Taxonomy. *IEEE Softw.*, 7(1) :13–17, January 1990. 120

[28] CIGREF. Gouvernance du système d'information, 2002. 30

[29] Bruno Claudepierre. *Conceptualisation de la Gouvernance des Systèmes d'Information : Structure et Démarche pour la Construction des Systèmes d'Information de Gouvernance*. phdthesis, Université Panthéon-Sorbonne - Paris I, December 2010. 32, 33, 34

[30] Mickaël Clavreul. *Model and Metamodel Composition : Separation of Mapping and Interpretation for Unifying Existing Model Composition Techniques*. PhD thesis, Université Rennes 1, December 2011. 95, 141

[31] John C. Coates. The goals and promise of the Sarbanes-Oxley Act. *The Journal of Economic Perspectives*, 21(1) :91–116, 2007. 29

[32] Benoît Combemale. Ingénierie Dirigée par les Modèles (IDM) – État de l'art, 2008. 117

[33] Llanos Cuenca, Andrés Boza, Angel Ortiz, and Jos J. M. Trienekens. Business-IT Alignment and Service Oriented Architecture - A Proposal of a Service-Oriented Strategic Alignment Model :. pages 490–495. SCITEPRESS - Science and and Technology Publications, 2014. 49

[34] Valeria De Castro, Esperanza Marcos, and Juan Manuel Vara. Applying CIM-to-PIM Model Transformations for the Service-oriented Development of Information Systems. *Inf. Softw. Technol.*, 53(1) :87–105, January 2011. 141

[35] Philippe Desfray and Gilbert Raymond. *TOGAF en pratique - Modèles d'architecture d'entreprise*. Dunod, May 2012. 25, 38, 41, 42, 59, 60, 62, 69, 70

[36] Mousumi Dhara and K. K. Shukla. Comparative Performance Analysis Of Rnsc And Mcl Algorithms On Power-Law Distribution. *Advanced Computing : An International Journal (ACIJ)*, 2012. 176

[37] Marcos Didonet, Del Fabro, Jean Bézivin, and Patrick Valduriez. Weaving Models with the Eclipse AMW plugin. In *In Eclipse Modeling Symposium, Eclipse Summit Europe*, 2006. 99

[38] Nicolas Dieudonné. Les étapes de la mise en place d'un outil de cartographie dans une démarche d'urbanisation au sein d'une grande entreprise : le cas AREVA. 2007. 47

[39] Dominique Dionisi. *L'essentiel sur Merise*. Eyrolles, Paris, 2e éd. edition, October 1998. 57, 88

[40] DISIC and Government of France. Cadre commun d'Architecture d'Entreprise applicable au système d'information de l'Etat et à sa transformation, 2012. 42, 43, 44, 45, 58, 76, 89

[41] Pascal Eck, Van, Henk Blanken, and Roel Wieringa. Project GRAAL : Towards Operational Architecture Alignment. *International Journal of Cooperative Information Systems*, January 2012. 50

[42] Brian Elves\a eter, Dima Panfilenko, Sven Jacobi, and Christian Hahn. Aligning Business and IT Models in Service-oriented Architectures Using BPMN and SoaML. In *Proceedings of the First International Workshop on Model-Driven Interoperability*, MDI '10, pages 61–68, New York, NY, USA, 2010. ACM. 142

[43] Steven D. Eppinger and Tyson R. Browning. *Design Structure Matrix Methods and Applications*. MIT Press, Cambridge, Mass, 2012. 173

[44] Anne Etien. *L'ingénierie de l'alignement : Concepts, Modèles et Processus.
La méthode ACEM pour la correction et l'évolution d'un système d'information aux processus d'entreprise*. phdthesis, Université Panthéon-Sorbonne - Paris I, March 2006. 48, 93

[45] Henri Fayol. *Administration industrielle et générale. Le texte fondateur du management*. Dunod, Paris, 2e éd. edition, 1999. 68

[46] Christiane Feral-Schuhl and Sophie Soubelet-Caroit. Sécurité financière : obligations pour les entreprises françaises. *Archimag*, (191) :52–53, 2006. 29

[47] Hector Florez, Mario Sánchez, Jorge Villalobos, and Germán Vega. Coevolution assistance for enterprise architecture models. pages 27–32. ACM, October 2012. 48

[48] Jean-Louis Foucard. *La Boîte à outils du Pilote des Systèmes d'Information - 2e éd.* Dunod, Paris, 2e édition edition, 2014. 29

[49] Archon Fung, Mary Graham, and David Weil. *Full disclosure : The perils and promise of transparency*. Cambridge University Press, 2007. 29

[50] Matthias Galster. Dependencies, Traceability and Consistency in Software Architecture : Towards a View-based Perspective. In *Proceedings of the 5th European Conference on Software Architecture : Companion Volume*, ECSA '11, pages 1 :1–1 :4, New York, NY, USA, 2011. ACM. 98

[51] Brice Govin, Nicolas Anquetil, Anne Etien, Stéphane Ducasse, and Arnaud Monegier. Récuperation de Composants à partir d'un système patrimonial. Lille, December 2015. 128

[52] J. C. Henderson and N. Venkatraman. Strategic Alignment : Leveraging Information Technology for Transforming Organizations. *IBM Syst. J.*, 32(1) :4–16, January 1993. 49

[53] Paul Hermans. The Zachman Framework for Architecture Revisited - On conceiving the informational enterprise, 2015. 39

[54] Knut Hinkelmann, Aurona Gerber, Dimitris Karagiannis, Barbara Thoenssen, Alta van der Merwe, and Robert Woitsch. A new paradigm for the continuous alignment of business and IT : Combining enterprise architecture modelling and enterprise ontology. *Computers in Industry*, pages –, 2015. 140

[55] Tharaka Ilayperuma. *Reference Ontology for Business Models - Towards Interoperability between Business Modelling Ontologies*. PhD thesis, Stockholm, 2007. 195

[56] Gérard Jean. *Urbanisation du business et des SI*. Management et informatique. Hermes Science Publications, September 2000. 48

[57] Frédéric Jouault, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev. ATL - A model transformation tool. *Science of Computer Programming*, 72 :31–39, 2008. 125

[58] Frédéric Jouault, Bert Vanhooff, Hugo Bruneliere, Guillaume Doux, Yolande Berbers, and Jean Bezivin. Inter-DSL Coordination Support by Combining Megamodeling and Model Weaving. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 2011–2018, New York, NY, USA, 2010. ACM. 142

[59] Jeong-Dong Kim, Jiseong Son, and Doo-Kwon Baik. CA5w1honto - Ontological Context-Aware Model Based on 5w1h. *International Journal of Distributed Sensor Networks*, 2012, March 2012. 39

[60] R. Koschke. Atomic architectural component recovery for program understanding and evolution. pages 478–481. IEEE, 2002. 128

[61] Brahim Lahna, Ounsa Roudiès, and Jean-Pierre Giraudin. Approches par points de vue pour l'ingénierie des Systèmes d'information. 2 ème partie, July 2009. 26

[62] Marc M. Lankhorst. *Enterprise Architecture at Work - Modelling, Communication and Analysis (3. ed.)*. The Enterprise Engineering Series. Springer, 2013. 14, 25, 50

[63] James Lapalme, Aurona Gerber, Alta Van der Merwe, John Zachman, Marne De Vries, and Knut Hinkelmann. Exploring the future of enterprise architecture : A Zachman perspective. *Computers in Industry*, pages –, 2015. 40

[64] Jannik Laval, Alexandre Bergel, Stéphane Ducasse, and Romain Piers. Matrice de dépendances enrichie. March 2009. 173

[65] Andreas Limyr, Tor Neple, Arne-Jørgen Berre, and Brian Elves\a eter. Semaphore – a Model-based Semantic Mapping Framework. In *Proceedings of the 2006 International Conference on Business Process Management Workshops*, BPM'06, pages 275–284, Berlin, Heidelberg, 2006. Springer-Verlag. 143

[66] Christophe Longépé. Démarche de conception d'une cible urbanisée et du plan de convergence. 2002. 38

[67] Christophe Longépé. *Le projet d'urbanisation du SI : Cas concret d'architecture d'entreprise*. Dunod, 4e édition edition, May 2009. 24, 26, 36, 37, 38, 40, 59, 61, 69, 70, 74

[68] Jerry Luftman. Assessing Business-IT Alignment Maturity. *Communications of the Association for Information Systems*, 4(1) :14, 2000. 47

[69] Florian Mantz, Gabriele Taentzer, Yngve Lamo, and Uwe Wolter. Co-evolving meta-models and their instance models : A formal approach based on graph transformation. *Science of Computer Programming*, 104 :2 – 43, 2015. Special Issue on Graph Transformation and Visual Modeling Techniques (GT-VMT 2013). 48

[70] Raphaël Marvie. *Séparation des préoccupations et méta-modélisation pour environnements de manipulation d'architectures logicielles à base de composants*. phdthesis, Université des Sciences et Technologie de Lille - Lille I, December 2002. 88

[71] Dirk Matthes. *Enterprise Architecture Frameworks Kompendium*. Xpert.press. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. 34

[72] L. O. Meertens, M. E. Iacob, L. J. M. Nieuwenhuis, M. J. van Sinderen, H. Jonkers, and D. Quartel. Mapping the Business Model Canvas to ArchiMate. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 1694–1701, New York, NY, USA, 2012. ACM. 141

[73] Pierre Merlin and Françoise Choay. *Dictionnaire de l'urbanisme et de l'aménagement*. PRESSES UNIVERSITAIRES DE FRANCE - PUF, 4e édition edition, 2015. 26, 36

[74] Daniel Minoli. *Enterprise Architecture A to Z : Frameworks, Business Process Modeling, SOA, and Infrastructure Technology*. Auerbach Publications, Boca Raton, 2008. 39

[75] Chantal Morley, Marie Bia-Figueiredo, and Yves Gillette. *Processus métiers et S.I. - Gouvernance, management, modélisation - 3e édition*. Dunod, 3e édition edition, February 2011. 57, 59, 60, 61, 62, 65

[76] Adra Al Mosawi, Liping Zhao, and Linda A. Macaulay. A Model Driven Architecture for Enterprise Application Integration. In *ResearchGate*, volume 8, January 2006. 142

[77] Yukio Namba and Junichi Iijima. "EII Meta-model" on integration framework for viable enterprise systems — City planning metaphor based on structural similarity. *J. Syst. Sci. Syst. Eng.*, 12(1) :111–126, March 2003. 37

[78] Dominique Nanci and Bernard Espinasse. *Ingénierie des systèmes d'information : Merise ; 2eme génération, 4e édition*. Vuibert, Paris, 2001. 57, 58, 88

[79] Kestutis Normantas, Sergejus Sosunovas, and Olegas Vasilecas. An Overview of the Knowledge Discovery Meta-model. In *Proceedings of the 13th International Conference on Computer Systems and Technologies*, CompSysTech '12, pages 52–57, New York, NY, USA, 2012. ACM. 124

[80] Takayuki Omori, Shinpei Hayashi, and Katsuhisa Maruyama. A Survey on Methods of Recording Fine-grained Operations on Integrated Development Environments and their Applications. *Computer Software*, 32(1) :1_60–1_80, 2015. 198

[81] Alexander Osterwalder. The Business Model Ontology – A Proposition in a Design Science Approach. *ResearchGate*, January 2004. 195

[82] Alexander Osterwalder and Yves Pigneur. *Business Model Nouvelle Génération : Un guide pour visionnaires, révolutionnaires et challengers*. PEARSON EDUCATION, Paris, September 2011. 195, 196

[83] Padideh Parchami. Enterprise Architecture & Alignment. September 2011. 48, 50

[84] Renaud Phelizon and Stéphane Rouhier. Gouvernance du systeme d'information problematiques et demarches, September 2002. 29

[85] Jacques Printz. *Architecture Logicielle : Concevoir des Applications Simples, Sûres et Adaptables*. Dunod, 3e édition edition, June 2012. 25

[86] Akshay Rasiwasia. *Meta Modeling for Business Model Design : Designing a Meta model for E3 value model based on MOF*. PhD thesis, Royal Institute of Technology, 2012. 86

[87] Robert Reix. *Systèmes d'information et management des organisations*. Vuibert, Paris, 2011. 24

[88] J. Saat, U. Franke, R. Lagerstrom, and Mathias Ekstedt. Enterprise Architecture Meta Models for IT/Business Alignment Situations. In *Enterprise Distributed Object Computing Conference (EDOC), 2010 14th IEEE International*, pages 14–23, 2010. 89

[89] Satu Elisa Schaeffer. Graph clustering. *Computer science review*, 1(1) :27–64, 2007. 174

[90] Abderrahmane Seriai, Salah Sadou, and Houari Sahraoui. Enactment of Components Extracted from an Object-Oriented Application. pages 234 – 249, August 2014. 128

[91] Jacqueline Sidi, Martine Otter, Laurent Hanaud, and Jean-Pierre Corniou. *Guide des certifications SI : Comparatif, analyse et tendances ITIL, CobiT, ISO 27001, eSCM*. Dunod, Paris, 2e édition edition, 2009. 31

[92] The Open Group. *TOGAF Version 9.1*. Van Haren Publishing, 10th new edition edition, 2011. 89

[93] The Open Group. *ArchiMate 2.1 Specification*. Van Haren Publishing, 2013. 79

[94] The Open Group. *ArchiMate 3.0 Specification*. Van Haren Publishing, 2016. 89, 197

[95] Laure-Hélène Thevenet. *Proposition d'une modélisation conceptuelle d'alignement stratégique : La méthode INSTAL*. phdthesis, Université Panthéon-Sorbonne - Paris I, December 2009. 50

[96] Laure-Hélène Thevenet, Camille Salinesi, Anne Etien, Ines Gam, and Menel Lassoued. Experimenting a Modeling Approach for Designing Organization's Strategies in the Context of Strategic Alignment. page 1, December 2006. 49

[97] Leendert van der Torre, Marc M. Lankhorst, Hugo ter Doest, Jan T. P. Campschroer, and Farhad Arbab. Landscape Maps for Enterprise Architectures. In Eric Dubois and Klaus Pohl, editors, *Advanced Information Systems Engineering*, number 4001

in Lecture Notes in Computer Science, pages 351–366. Springer Berlin Heidelberg, January 2006. 26

[98] Azmat Ullah and Richard Lai. A Systematic Review of Business and Information Technology Alignment. *ACM Trans. Manage. Inf. Syst.*, 4(1) :4 :1–4 :30, April 2013. 47

[99] Club URBA-EA. *Urbanisme des SI et gouvernance : Bonnes pratiques de l'architecture d'entreprise*. InfoPro. Management des systèmes d'information. Dunod, 1 edition, 2010. 25, 34

[100] Stijn van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, May 2000. 175

[101] Juan Manuel Vara and Esperanza Marcos. A Framework for Model-driven Development of Information Systems : Technical Decisions and Lessons Learned. *J. Syst. Softw.*, 85(10) :2368–2384, October 2012. 140

[102] A. Wegmann, G. Regev, I. Rychkova, L. S. Le, and P. Julia. Business and IT Alignment with SEAM for Enterprise Architecture. In *Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International*, pages 111–111, October 2007. 50, 142

[103] Alain Wegmann, Gil Regev, and Bertrand Loison. Business and IT Alignment with SEAM. *1st International Workshop on Requirements Engineering for Business Need, and IT Alignment (REBNITA2005)*, pages 74–84, 2005. 142

[104] Mathias Weske. *Business Process Management : Concepts, Languages, Architectures*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007. 27

[105] Randii R. Wessen and David Porter. Market-based approaches for controlling space mission costs : the Cassini Resource Exchange. *Journal of Reducing Space Mission Cost*, 1(1) :9–25, March 1998. 27

[106] Roel Wieringa. A survey of structured and object-oriented software specification methods and techniques. *ACM Computing Surveys*, 30(4) :459–527, December 1998. 25

[107] Roel J. Wieringa, H. M. Blanken, M. M. Fokkinga, and P. W. P. J. Grefen. Aligning Application Architecture to the Business Context. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Johann Eder, and Michele Missikoff, editors, *Advanced Information Systems Engineering*, volume 2681, pages 209–225. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. 141

[108] Ali A. Yassine. An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix (DSM) Method. *ResearchGate*, (9), January 2004. 174

[109] Raphael Yuster and Uri Zwick. Fast sparse matrix multiplication. *ACM Transactions on Algorithms (TALG)*, 1(1) :2–13, July 2005. 179

Liste des tableaux

Liste des figures

Liste des définitions

Liste des exemples

Liste des expérimentations

UNIVERSITE BRETAGNE LOIRE

UNIVERSITÉ DE NANTES

Thèse de Doctorat

Jonathan PEPIN

Architecture d'entreprise : alignement des cartographies métiers et applicatives du système d'information

Enterprise Architecture
Business-IT alignment

Résumé

L'architecture d'entreprise se préoccupe de la maintenance d'un système d'information (SI) qui nécessite une mise en phase répétée avec le pilotage de l'entreprise. En pratique, les points de vue métiers et applicatifs s'éloignent à mesure qu'évoluent la stratégie de l'entreprise et le patrimoine applicatif. Ainsi, il est difficile pour les architectes de mesurer l'impact de l'évolution des processus métiers vis-à-vis des technologies utilisées. Le risque d'incohérence devient grand et préjudiciable dans le temps.
Nous considérons chaque point de vue du SI comme un modèle ; pour associer les points de vue nous proposons une définition de l'alignement au niveau méta-modèle. L'alignement est par conséquent mis en pratique à l'aide d'outils d'ingénierie des modèles.
Il n'est pas imaginable d'aligner directement le code source aux modèles métiers existants. Nous avons donc défini un alignement entre modèles métiers et modèles applicatifs. Ces derniers sont construits par rétro-ingénierie du code.
L'analyse du résultat de l'alignement permet de générer des indicateurs de cohérence et de complétude qui aident l'architecte dans son processus d'analyse et de décision des évolutions du système d'information.
L'approche présentée est expérimentée sur plusieurs cas concrets de taille significative issus d'entreprises du secteur de l'assurance, et ouvre des perspectives intéressantes de valorisation.

Mots clés
Architecture d'entreprise, système d'information, alignement, évolution.

Abstract

Enterprise software architects take great care of information system maintenance, which requires regular adjustments in order to stay in sync with management needs.
In practice, business and application viewpoints diverge as the enterprise strategy and the legacy applications evolve.
Thus, it is difficult for architects to evaluate the impact of process evolution with respect to chosen technologies. Risk of inconsistency becomes more important and damaging over time.
We consider each IS viewpoint as a model; to aggregate these, we propose to map the models with an alignment definition based on meta-models. This alignment is therefore supported by model engineering tools.
As existing business models and source code cannot be aligned, business models is aligned with application models. These are built by performing a reverse engineering of the code.
Futhermore, analysing the alignment results enables one to measure consistency and completeness indicators. They help the architect in the process of making changes to the information system.
The presented approach is illustrated with various large scale french insurance companies use cases. It reveals valuable improvement perspectives.

Key Words
Enterprise Architecture, Information System, Alignment, Evolution.