Alignement de modèles métiers et applicatifs : une approche pragmatique par transformations de modèles

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

Résumé

La maintenance du système d'informations nécessite de le mettre en phase 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 la maintenance des applications informatiques. Par conséquent, il est difficile pour les architectes logiciels de mesurer l'impact de l'évolution de celles-ci vis-à-vis des processus métiers ou des technologies utilisées.

Nous proposons une vision pragmatique de rapprochement des deux points de vue, via la rétro-ingénierie du code applicatif et des modèles métiers existants. L'idée est de transformer progressivement de part et d'autre les modèles afin de rapprocher les points de vue. Le résultat de ce processus de transformations permet de tisser les modèles obtenus dans le but d'obtenir un alignement ou de détecter des incohérences. L'approche présentée est mise en œuvre par des transformations de modèles et expérimentée sur un cas concret de taille significative.

Mots-clés : Systèmes d'information - Architecture d'entreprise - Rétro-ingénierie - Alignement - Ingénierie des modèles

1 Introduction

Compte-tenu de l'aspect prégnant de l'informatique dans la gestion des entreprises, l'évolution de leurs systèmes d'information (SI) est devenue un enjeu majeur. En effet les cycles de décisions (réorganisation, compétitivité, législation) sont plus courts et nécessitent une forte réactivité du SI. Le SI est aussi assujetti aux progrès technologiques continus tant matériels que logiciels mais le cycle de renouvellement du parc applicatif, souvent hétérogène, est bien plus lent du fait de contraintes budgétaires ou organisationnelles. Maintenir le patrimoine applicatif en phase avec l'évolution des métiers (et des technologies) implique des coûts importants. L'architecture d'entreprise [1] (ou urbanisation en version française) est un domaine qui contribue à proposer des méthodes pour maîtriser les systèmes de l'entreprise et leur évolution. Elle fait apparaître différentes visions du système d'information, visions qui ne sont pas toujours compatibles ou pas toujours compréhensibles selon le point de vue. Il existe souvent un fossé entre le point de vue métier et le point de vue informatique. Ce fossé s'accroit au fil des opérations de maintenances lors desquelles la vision architecturale du système informatique se détache de la vision opérationnelle, le niveau dans lequel s'effectue la réelle évolution. L'alignement entre la stratégie d'entreprise (*Business*) et l'informatique d'entreprise (*Information Technology* - IT) est un enjeu important. L'alignement est un problème complexe du fait de l'éloignement "culturel" des acteurs et des rôles (management, métier, informatique) et les résultats pratiques sont insatisfaisants [5, 2]. D'un côte le pilotage

1. *Enterprise architecture*

raisonne à un niveau stratégique avec des modèles informels et hétérogènes (documents, présentations, feuilles de calcul). De l'autre côté, il est difficile de présenter à la DSI une cartographie réaliste et compréhensible du patrimoine fait de multiples programmes et supports physiques. Pour mettre en œuvre l'urbanisation il faut rapprocher les points de vue afin d'en mesurer l'alignement, qui en pratique reste assez général voire approximatif.

Les visions stratégique et opérationnelle (ou technique) ne peuvent être alignées directement, du fait du nombre de concepts techniques à intégrer et de la distance sémantique entre les niveaux. Notre motivation est d'aligner les deux points de vue pour déterminer les effets croisés de leur évolution. Le travail présenté ici cible deux objectifs pour répondre au défi de l'alignement entre les visions métier et informatique. Le premier objectif est de rapprocher les points de vue ; l'idée sous-jacente est que chacun "fasse le pas vers l'autre". La vision stratégique doit se décliner en vision métier via des processus et un découpage fonctionnel. La vision opérationnelle doit masquer les détails de la mise-en-œuvre et présenter une architecture de plus en plus abstraite. Pour atteindre cet objectif, un processus de rétro-ingénierie peut être mis en place, à base de transformations de modèles. Une fois que des modèles "plus compatibles" sont mis en évidence, le second objectif est de déterminer une technique d'alignement outillée pour établir les correspondances entre les concepts dans ces modèles.

Nous proposons ici une approche opérationnelle du rapprochement et de l'alignement des visions métier et applicatives. Nous considérons que chaque point de vue est représenté par un ensemble de modèles. Pour rapprocher les points de vue, nous proposons un processus de rétro-ingénierie qui permette de gagner en abstraction du côté applicatif. Nous définissons ainsi les caractéristiques nécessaires dans les modèles intermédiaires, en nous inspirant des travaux sur l'urbanisation [24, 20, 14] et les architectures logicielles [18]. Ensuite des transformations de modèles permettent de définir le processus d'abstraction des applications en architectures à composants et services. Enfin nous proposons une solution non-intrusive d'alignement pour tisser les modèles métiers et applicatifs. Nous avons choisi un mécanisme de contraintes défini dans le métamodèle pour maintenir les liens plutôt qu'un croisement des modèles [10] ou une séparation des préoccupations [11]. L'ensemble est outillé et une expérimentation sur un cas concret a été menée. L'approche se veut générique du point de vue des modèles et langages utilisés.

L'article est organisé comme suit. La section 2 situe le problème de l'alignement des points de vue d'un système d'information et présente notre vision du problème. Nous proposons dans la section 3 une solution pragmatique pour aligner une modélisation métier et les applications associées. Cette approche est mise en œuvre par un processus de transformations de modèles réalisant une rétro-ingénierie et un tissage non-intrusif. Nous les détaillons dans la section 4. La section 5 relate une expérimentation de l'approche sur un cas concret d'une compagnie d'assurance avec d'un côté des modèles métier MEGA et de l'autre un code source volumineux en java et JSP. Nous discutons des travaux connexes dans la section 6. La section 7 conclut l'article et trace des perspectives.

2 Aligner les points de vue d'un système d'information

Le système d'information est un *ensemble organisé de ressources : matériel, logiciel, personnel, données, procédures… permettant d'acquérir, de traiter, de stocker des informations (sous forme de donnée, textes, images, sons, etc.) dans et entre des organisations*[19]. Les systèmes d'information sont un élément majeur dans le fonctionnement des entreprises car ils constituent le lien entre les différentes parties de l'entreprise, formant les niveaux stratégiques, décisionnels et opérationnels. Le système d'information a des ramifications dans toute l'entreprise. Plus l'entreprise est grande plus il est difficile de maîtriser le lien entre la gestion stratégique et la gestion opérationnelle. L'architecture d'entreprise vise à modéliser le fonctionnement de l'entreprise pour en contrôler l'évolution [4]. L'urbanisme cible plus spécifiquement le système d'information et la démarche d'urbanisation vise à en améliorer le fonction-

nement en le rationalisant dans un "cercle vertueux de transformation et d'amélioration continue" [24]. Pour Y. Caseau [20], l'urbanisation est en premier lieu une démarche technique utilisant des principes simples (décomposition, découplage, intermédiation) pour répondre à des objectifs de flexibilité, de mutualisation, de maintenabilité, etc. Ces thématiques sont très actives depuis plus d'une décennie et mobilisent actuellement la communauté du logiciel.

Le système d'information est perçu par ses acteurs selon deux aspects complémentaires : l'aspect gestion ou métier (*business*) et l'aspect informatique et technique (*Information Technology IT*). Cette dualité *business/IT* est la caractéristique majeure des systèmes d'information d'entreprise. Elle implique des visions différentes, des méthodes et techniques différentes qu'il faut rapprocher. De plus pour chaque aspect, en fonction des acteurs concernés, plusieurs points de vue se confrontent (des niveaux de préoccupation selon [24]). Par exemple dans l'entreprise on distingue entre autres la vue stratégique (pilotage, gouvernance), la vue métier (organisationnelle, fonctionnelle) et la vue opérationnelle. 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 [7].

Les modèles d'architectures présentent souvent ces points de vue en couche d'abstraction où les niveaux bas sont les plus concrets et opérationnels. Par exemple le cadre de la figure 1, inspiré de Longépé [24] propose cinq couches.

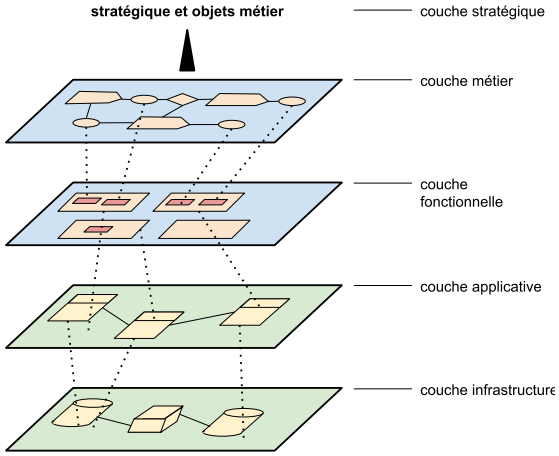

FIGURE 1 – Couches du système d'information et alignement

Stratégique : Le niveau stratégique donne le positionnement de l'entreprise (marché, organisation, produits) et ses objectifs métier.

Métier : La couche métier prend en charge l'organisation à travers des processus et des activités.

Fonctionnelle : La couche fonctionnelle organise hiérarchiquement les fonctionnalités des activités de la couche métier par exemple en termes de blocs fonctionnels (ex : zone, îlot, quartier).

Applicative : La couche applicative modélise les applications, composants logiciels et services implantant des fonctionnalités, ainsi que leur relations.

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).

L'alignement *business/IT* est un instrument d'efficacité pour les architectures d'entreprise et l'urbanisation [23, 2]. En pratique le but est surtout de détecter des incohérences d'alignement. A travers la

vision usuelle de la figure 1, l'alignement peut être interprété comme une ligne de traçabilité traversant les couches. L'alignement définit alors les concepts à relier entre les couches. Mais le problème est plus complexe [2]. Au sens large, l'alignement couvre différents aspects et d'une méthode à l'autre, on privilégie l'aspect gestion ou bien l'aspect informatique. La méthode GRAAL par exemple distingue trois dimensions : sociale (l'entreprise), physique (infrastructure) et symbolique (logiciel) [14]. L'alignement proposé se fait alors deux à deux. Dans le travail présenté ici, nous réduisons le problème à l'alignement social-symbolique en considérant les deux hypothèses suivantes :

- L'alignement entre l'organisation et l'infrastructure perd de l'importance compte-tenu des architectures réparties et du *cloud*. Il n'est plus nécessaire de rapprocher les personnes des serveurs.
- Le déploiement est souvent décrit finement et représente explicitement l'alignement entre les programmes (bas niveau des applications) et l'infrastructure.

Dans l'idéal, l'alignement se situe entre la stratégie et les programmes, qui représentent les applications à bas niveau (*cf.* figure 2). Mais la stratégie n'ayant pas de modèle, le niveau purement stratégique ne sera pas considéré ici. La nature des liens d'alignement varie selon les travaux. Dans les cadres TO-

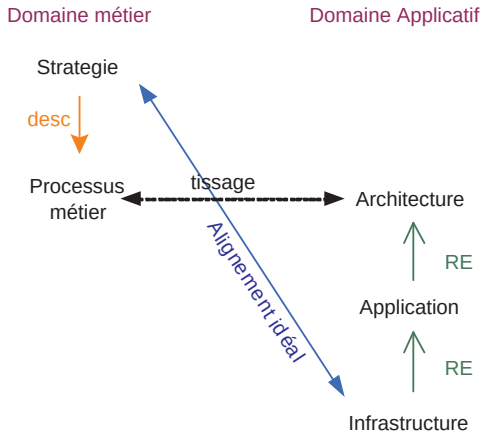

FIGURE 2 – Rapprochement et alignement

GAF [12] ou ArchiMate [22], le lien est défini entre les services métiers et les services applicatifs. Dans d'autres travaux [9, 21] le lien est établi entre activités métiers (ou tâches) au sens BPM [3] et des fonctionnalités d'applications. Pour fixer nos choix en vue de proposer une assistance à l'alignement, deux points sont à considérer :

- Les propositions d'assistance à l'alignement doivent être génériques et adaptables à différents cadres, différentes notations.
- L'alignement n'est pas purement vertical mais ressemble plus à une connexion entre deux mondes qu'il faut rapprocher. On concrétise la stratégie d'entreprise en processus métier et on abstrait le patrimoine applicatif en fonctionnalités et blocs, ce que montre la figure 2.

L'alignement devient plus simple si on dispose d'une notion de service des deux côtés, mais elle ne s'applique pas naturellement sur le patrimoine ancien. La section suivante présente le processus de transformation qui supporte notre méthode d'alignement.

2. Un alignement parfait n'est jamais atteint [15] car il y a trop de facteurs de changements tant technologiques que légaux ou encore concurrentiel.
3. Business Process Management

3 Un processus de transformation pour rapprocher les modèles

Nous décrivons les étapes du processus esquissé dans la section 2.

3.1 Abstraction du code

Afin de réaliser un modèle applicatif à partir du code source, trois étapes sont nécessaires :

E1 La 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.

E2 La transformation vers un modèle pivot pour s'abstraire des spécificités du langage de programmation.

E3 La transformation vers le méta-modèle applicatif comportant les concepts retenus.

Ces étapes peuvent se référer à l'ingénierie dirigée par les modèles (IDM) : la première étape permet d'obtenir le PSM (*Platfom Specific Model*), il s'agit du modèle correspondant à la plateforme du code source ; la seconde étape s'abstrait de l'architecture logicielle et constitue le modèle PIM (*Platform Independent Model*). La définition de ces modèles est habituellement utilisée dans un contexte MDA "*top-down*" ; on part du CIM (*Computation Independent Model*) qui est un modèle métier indépendant de l'informatisation vers le PSM dans une logique de génération de code. Alors que notre démarche d'abstraction est inverse, on part du PSM pour un tissage avec le CIM. La figure 3 illustre les deux dernières transformations.

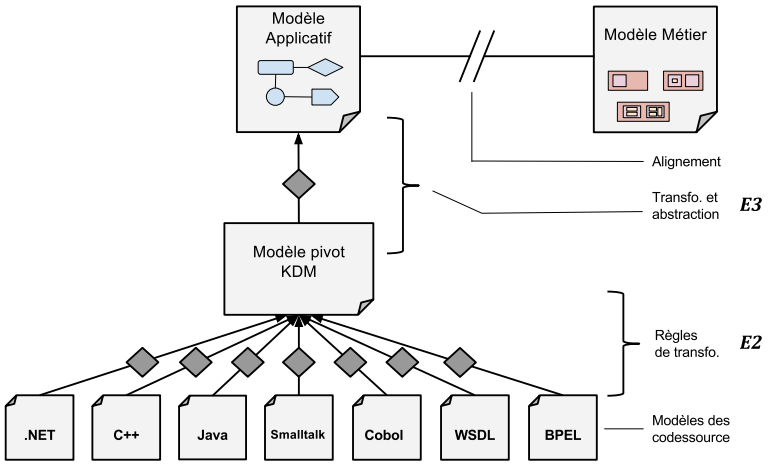

FIGURE 3 – Les étapes de transformation

L'étape *E1* est dépendante du langage et de l'architecture employée, or les systèmes d'information sont hétérogènes et composés de technologies diverses évoluant rapidement dans le temps. Ainsi, il sera nécessaire de réaliser un analyseur syntaxique (*parser*) différent pour chaque technologie rencontrée, tout en essayant de capitaliser entre les technologies similaires. Les langages les plus communément rencontrés en entreprise sont Java, C++, .Net, Smalltalk, Cobol...

C'est également la raison pour laquelle la seconde étape (*E2*) de notre méthode est une transformation vers un modèle intermédiaire. Ce dernier est défini par un méta-modèle unique capable de recevoir les concepts des différentes technologies issues de la première étape *E1*. Pour cela nous avons choisi le méta-modèle KDM (*Knowledge Discovery Metamodel*) qui est un standard spécifié par l'OMG utilisé

dans de nombreux outils informatiques. KDM inclut de nombreuses couches pour stocker les différents aspects des langages de programmations communs [17]. En ce qui nous concerne, nous n'utilisons qu'une partie du méta-modèle KDM, notamment le paquetage `Code`.

Enfin, l'étape *E3* consiste en l'abstraction la plus forte pour se séparer de la logique de code programme vers une logique applicative. Ainsi, nous avons défini un méta-modèle à partir de la proposition issue de travaux antérieurs [1] mais aussi d'ArchiMate [22] définissant la notion d'application constituée de composants, services, interfaces, fonctions et objets de données. La transformation doit ainsi détecter les aspects contenus dans le modèle KDM, bien que KDM soit un modèle pivot des spécificités peuvent subsister. En effet, la rétro-ingénierie d'un code source complet ne se préoccupe pas de l'utilité des éléments contenus dans le code source, s'il s'agit d'éléments liés à l'aspect métier, utilitaire ou simplement technique. Ainsi, c'est à cette étape de transformation que l'on va tamiser les différents éléments de l'application et ne capter que les concepts utiles pour peupler le modèle applicatif.

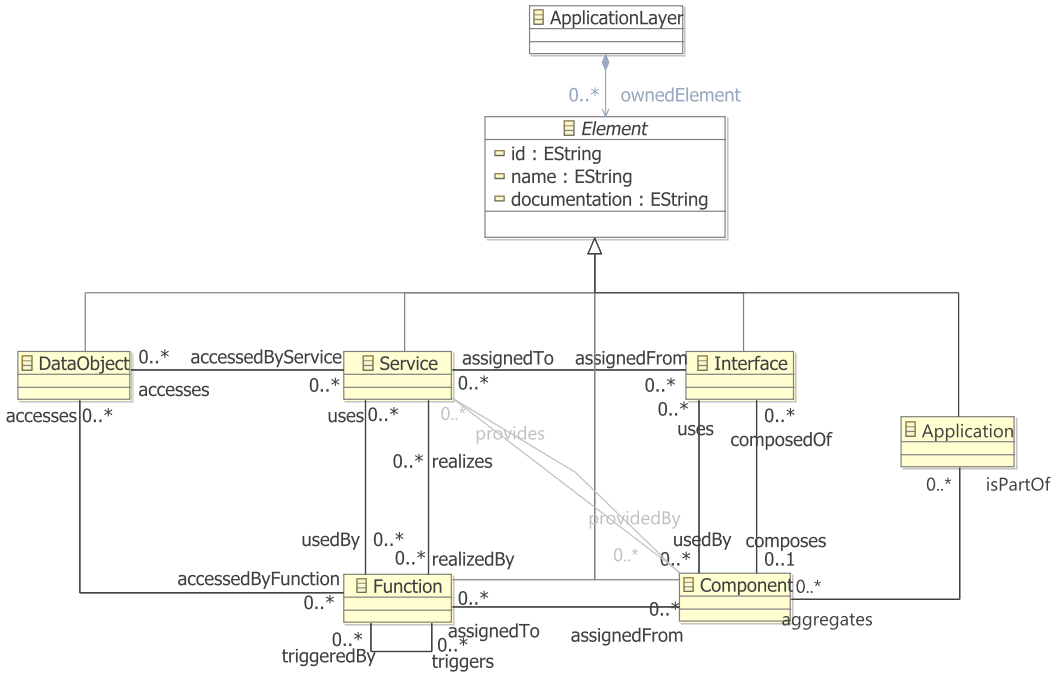

FIGURE 4 – Le métamodèle applicatif

3.2 Concrétisation de la stratégie métier

Les modèles métiers sont issus de la modélisation du fonctionnement de l'entreprise. Cette modélisation identifie les processus composés d'activités organisées de façon chronologique et représentées par des transitions [16]. Chaque processus peut être décomposé en sous-processus. Le niveau de granularité peut aussi décomposer les activités en tâches qui correspondent au degré le plus fin d'une cartographie processus. L'ensemble du SI comporte donc de nombreux processus qui sont contenus dans plusieurs modèles contenant un ou plusieurs processus. Il existe de nombreux langages de modélisation tels que Merise, UML, BPMN… Le nom des concepts, le niveau de granularité et la représentation peuvent différer, mais les concepts clefs sont faciles à transcrire d'un modèle à l'autre. On retrouve toujours

les concepts essentiels suivants : rôle/acteur, processus, activité, tâche, condition, transition, événement. La modélisation des processus métier peut également être représentée dans un référentiel plus vaste où sont également cartographiés des éléments des couches applicative et infrastructure tels que application, base de données, bus, message, etc. Nous verrons dans la section 5 comment les processus métier sont modélisés dans un référentiel MEGA.

3.3 Tissage et mise en perspective

Les modèles applicatif et métier sont liés par des concepts précis que nous avons déterminés après l'étude des différentes démarches d'architecture d'entreprise. Nous nous sommes notamment inspirés des travaux de la DISIC [4] et du document de référence [9]. La Figure 5 en propose un extrait qui représente des liens entre les différentes couches du SI.

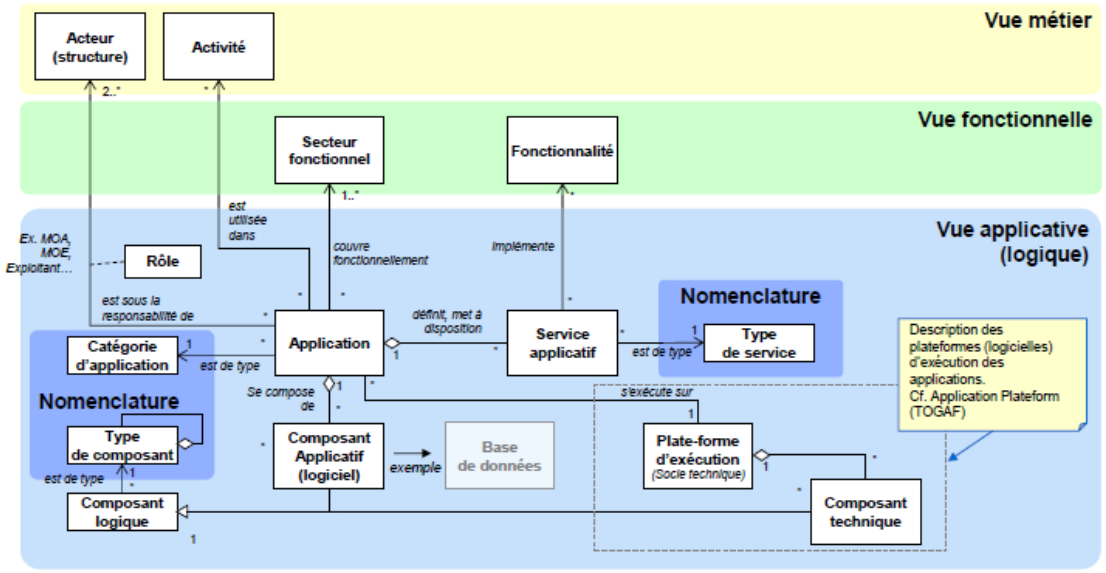

FIGURE 5 – Le modèle applicatif et ses liens selon le DISIC

Par exemple, nous avons établi des liens entre les activités du métier et les applications, entre les tâches du métier et les services applicatifs. La réalité ne se résume pas à ces cas. La mise en correspondance dépend du niveau de granularité choisi, de la portée de l'alignement (SI total ou partiel, une ou plusieurs applications), du type des éléments (structure ou comportement), des objectifs de l'alignement…

Cette définition des liens, nous permet d'établir des règles de tissage entre les différents modèles.

4. Direction interministérielle des systèmes d'information et de communication

4 Mise en œuvre du processus de rapprochement

Notre démarche d'urbanisation est accompagnée d'une solution outillée afin d'automatiser chaque étape du processus de rétro-ingénierie et d'assister le concepteur durant l'étape de tissage.

4.1 Les outils de transformation

Les étapes de transformation employant l'ingénierie des modèles, nous avons choisi d'utiliser la technologie Eclipse EMF qui permet de définir ses propres méta-modèles et de développer des extensions (*plugins*) les manipulant. Nous nous sommes également basés sur un projet *open source* de modernisation des systèmes d'information nommé Eclipse Modisco, ce projet initié par AtlanMod [5] est aujourd'hui porté par l'entreprise Mia-Software [6]. Modisco inclut un mécanisme de découverte pour réaliser l'analyse d'un code source et la transformation de modèle EMF. Modisco inclut également des méta-modèles technologiques pour Java, KDM et SMM.

Vers un modèle du source Java Pour nos expérimentations, relatées en section 5, nous avons choisi le langage Java comme support du code source des programmes. Pour l'étape de rétro-ingénierie *E1*, nous utilisons l'analyseur Java Discovery intégré à Modisco. Il s'appuie sur l'extension JDT (Java Development Toolkit) d'Eclipse pour créer un graphe syntaxique des éléments du langage contenu dans les fichiers de code source. Cette découverte produit un modèle du code Java en sortie.

Vers un modèle KDM La seconde étape *E2* est le passage du modèle de la plateforme source au modèle intermédiaire et indépendant des langages de programmation. En l'occurrence il s'agit du passage de Java vers KDM dans notre mise-en-œuvre. Modisco incorpore une transformation Java vers KDM en ATL [7]. Des premiers tests ont été élaborés avec cette transformation, mais au fur et à mesure que les modèles d'expérimentation en entrée étaient de plus en plus volumineux (plusieurs centaines de mégaoctets), la transformation s'est révélée inexploitable car elle ne se terminait pas malgré plusieurs dizaines d'heures de traitement. Or dans le domaine de l'architecture d'entreprise il est nécessaire de pouvoir traiter des modèles volumineux, c'es-à-dire qui englobent tout ou partie d'un SI. Nous avons tenté de détecter l'erreur et de mettre à jour le moteur d'exécution vers une version plus récente. Mais faute de maitriser l'outil ATL, nous avons décidé d'écrire notre propre transformation avec un autre outil. Nous avons utilisé l'éditeur avancé du logiciel Mia-Transformation pour écrire ces transformations de Java vers KDM en nous inspirant, évidemment, des règles écrites en ATL dans Modisco avec quelques modifications lorsque des améliorations utiles ont été détectées.

Vers un modèle applicatif Pour la troisième étape de transformation et d'abstraction du modèle KDM vers le modèle Applicatif (étape *E3*), nous avons codé le méta-modèle Applicatif dans un fichier `.ecore` à l'aide d'EMF, puis nous avons développé la transformation avec l'outil Mia-Transformation. Dans le métamodèle EMF applicatif nous avons ajouté des règles OCL de validation. Le listing 1 en illustre un échantillon. Les règles de transformations sont de deux types. Un premier jeu structurel correspond aux langages de programmation orienté objet : manipulation de classes, interfaces, méthodes, paquetages... Un second jeu sémantique permet de détecter les composants et les objets de données du programme. Cette analyse est la plus complexe et variable d'un jeu de test à l'autre et doit donc être redéfini au cas par cas.

Après ces trois étapes de transformation, on obtient le modèle applicatif stocké dans un fichier XMI, la norme d'échange de modèles de l'OMG.

5. `http://www.emn.fr/z-info/atlanmod/`
6. `http://www.mia-software.com`, Deux auteurs de ce papier font partie de cette entreprise.
7. ATL est un langage et un moteur de transformation *open source* initié par Atlanmod

Listing 1 – Règles OCL

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

4.2 L'assistant de tissage des modèles

La technologie de tissage permet de créer un modèle indépendant qui référence plusieurs modèles issus de méta-modèles différents et ainsi créer des liens entre les éléments sources et cibles des différents modèles [11]. Cette technologie est non intrusive puisque les modèles référencés ne sont pas modifiés par le tissage. Des outils ont été développés pour Eclipse EMF, notamment : AMW[8] [8] et Virtual EMF[3]. 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 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 [9]. Ainsi, nous avons créé notre propre éditeur inspiré à partir de ces travaux en y incluant des améliorations et nos propres fonctionnalités.

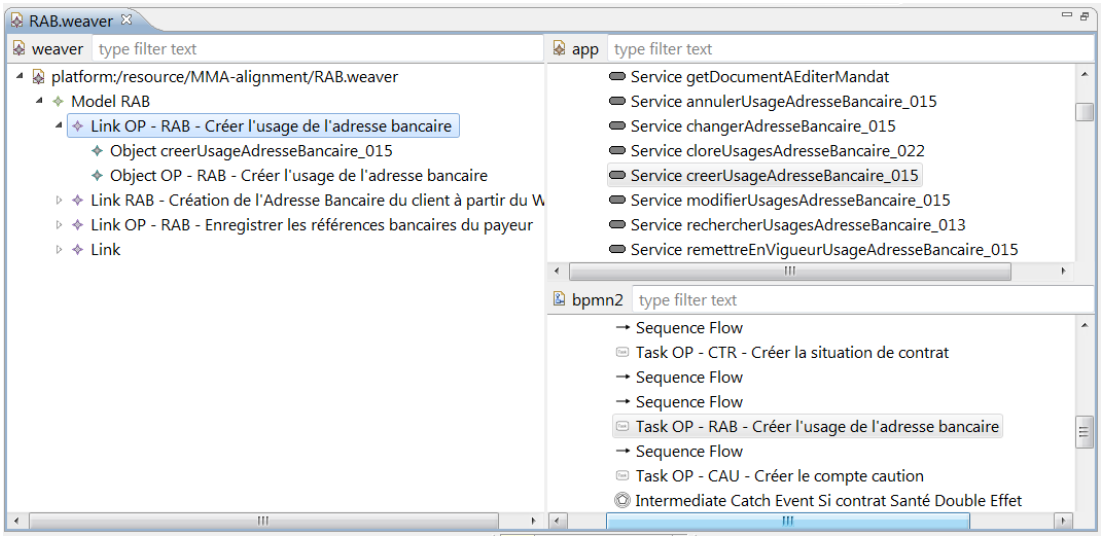

FIGURE 6 – Capture d'écran de notre plugin de tissage

8. Atlas Model Weaver
9. Testé sur Eclipse Juno 4.2

L'éditeur présente une interface en deux parties (*cf.* figure 6). A gauche l'arborescence contient les liens tissés. A droite, les différents modèles chargés et regroupés par méta-modèles. Ainsi, il est possible de tisser autant de modèles et de méta-modèles différents que souhaité. Pour aider à la réalisation du tissage, l'interface dispose d'un champ de recherche qui permet de trouver rapidement les éléments des modèles par leur nom de concept. C'est une fonctionnalité essentielle pour l'architecture d'entreprise, car les volumes des modèles peuvent devenir très importants (*cf.* section 5). La technologie de tissage nous posait des difficultés car les liens créés n'ont pas de signification, ils peuvent contenir des éléments de n'importe quel méta-modèle. Or dans notre démarche nous avons défini des liens entre les méta-modèles applicatif et métier (*cf.* section 3.3), ainsi nous voulons réaliser des restrictions pour ne pouvoir tisser que des concepts de type particulier : par exemple, Service depuis le modèle applicatif et Activité depuis le modèle métier. Pour palier à ce problème, nous avons créé un mécanisme de contrainte en réalisant un tissage au niveau méta-modèle. En effet, il suffit d'ouvrir ce même éditeur avec les méta-modèles à tisser et de réaliser le lien avec les types des concepts souhaités. En reprenant notre exemple, il faut charger les méta-modèles applicatif et métier, puis créer un lien appellé `ServiceVersActivité` contenant le type `Service` et le type `Activité`. Lors du tissage au niveau modèle, le nouveau type de lien sera disponible et limitera le tissage à ces types d'éléments. Le tissage étant réalisé de façon manuelle à l'aide de cet éditeur, cette solution de contrainte permet d'éviter les erreurs.

Dans la section suivante, nous relatons une expérimentation à partir d'un cas d'étude proposé par une société d'assurance mutuelle française. L'expérimentation a été réalisée à l'aide de nos outils.

5 Expérimentation

Le cas d'étude est composé *i)* d'un code source complet écrit en Java avec 33 400 classes, environ 3 400 000 lignes de codes et 800 fichiers JSP et *ii)* d'un référentiel métier sous la forme d'un portail HTML exporté depuis le logiciel MEGA Enterprise Architecture [10]. Le référentiel contient 360 diagrammes de processus métier couvrant la totalité du SI. Le cas d'étude est significatif pour notre problématique de modernisation des systèmes d'informations, le volume de l'application est conséquent et représente un véritable challenge pour traiter ce volume d'informations. De plus le code source fourni provient d'un logiciel actuellement en production et déployé sur le SI de la société d'assurance mutuelle. L'expérimentation se déroule en deux parties : la première consiste à enchaîner les transformations outillées définies dans la section précédente ; la deuxième consiste à réaliser le tissage entre le modèle applicatif obtenu à l'étape précédente et le modèle métier provenant du MEGA.

5.1 La chaîne de transformation automatisée

La première étape de la chaîne de transformation est la découverte avec l'outil Eclipse (*cf.* section 4.1). Cette étape de rétro-ingénierie malgré la taille du code source est très rapide : environ 10 minutes. Alors que le fichier XMI généré en sortie a une taille de 1,4 Go.

La deuxième étape est la transformation du modèle Java vers le modèle KDM avec l'outil Mia-Transformation (*cf.* section 4.1). Les premières tentatives ont échoué fautes de mémoire vive suffisante. Nous avons dû utiliser une machine pourvu de 10 Go de mémoire vive et d'un processeur avec 4 cœurs. Après environ 2 heures de traitement nous obtenons avec succès un modèle KDM, sa taille est de 780 Mo. Comparé au modèle Java, la différence de taille est due au concept de langage qui est moins détaillé dans le paquetage code du métamodèle KDM que dans le métamodèle Java, ainsi toutes les spécificités du langage Java ne sont pas traduites en détail. Néanmoins, les concepts essentiels (classes, interfaces, méthodes, routines, variables, commentaires, etc) sont présents et sont correctement retranscrits.

10. `www.mega.com/fr/solution/business-architecture`

La dernière étape est la transformation du modèle KDM vers le modèle applicatif (*cf.* section 4.1). 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. Enfin, nous avons validé le modèle obtenu avec les règles OCL incorporées dans le méta-modèle (Listing 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.

5.2 Tissage avec le modèle métier

Le référentiel MEGA fourni par la société d'assurance mutuelle 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 pas de modèle XMI à charger ou de rétro-ingénierie possible. Nous avons alors décidé de réaliser une traduction manuelle d'une partie des diagrammes vers le langage de modélisation métier BPMN2. Nous avons choisi BPMN 2.0 [11] car il s'agit d'une norme de l'OMG qui est utilisée dans de nombreux projets. Pour modéliser nos processus métier, nous avons employé le *plugin* Eclipse BPMN2 modeler. Cet outil de modélisation a pour avantage d'avoir un méta-modèle interne qui suit rigoureusement les spécifications de l'OMG.

MEGA	**BPMN2**	**Description**
Acteur	Pool ; Couloir	Élément actif chargé d'une ou plusieurs activités dans le processus
Message	Lien de séquence	Lien orienté entre deux activités
Processus	Processus	Ensemble d'activités liées ayant même finalité
Procédure	Sous-processus	Ensemble de tâche permettant d'avoir une vue d'ensemble d'une activité.
Opération	Tâche	Plus petit élément de décomposition d'une activité

TABLE 1 – Règles de traductions des concepts processus métier

Nous avons réduit le périmètre de traduction à une partie de l'activité de la société d'assurance mutuelle appelée *Adresse Bancaire*, car le référentiel 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.

Nous avons chargé les modèles BPMN2 et les modèles applicatifs dans notre éditeur de tissage (*cf.* section 4.2). Nous avons établit des liens entre les tâches provenant du modèle métier et les services provenant du modèle applicatif. Pour cela nous nous sommes servis de la fonction de recherche par nom pour trouver les correspondances. Dans ce cas d'étude, la similarité des noms facilite l'établissement des correspondances. Le travail a été facilité par la présence d'une nomenclature des concepts qui se retrouve à différents niveaux et par là même montre de bonnes pratiques de codage, même si des exceptions subsistent.

11. Business Process Model and Notation

6 Travaux connexes

Nos travaux traitent principalement deux questions : comment réaliser l'alignement du système d'information ? Comment y parvenir à l'aide de l'ingénierie des modèles ? Nous avons repris trois exemples parmi les travaux étudiés qui abordent ces deux problématiques.

6.1 La méthode d'alignement GRAAL

La problématique d'alignement des systèmes d'information et notamment des points de vue métier et informatique est étudiée depuis de nombreuses années et les recherches se poursuivent faute de solution ultime et aboutie. La principale raison est que le problème évolue au fil des innovations techniques et des nouvelles organisations des entreprises. De plus aucune entreprise ne ressemble à une autre, chaque méthode d'architecture d'entreprise doit-être appropriée, modifiée, personnalisée. Une méthode d'alignement que nous avons étudiée est la méthode GRAAL [12] [14] dont la première version a été publié en 1996 et éprouvé depuis. GRAAL fait référence à la quête du graal pour les entreprises, la méthode qui permettra l'alignement de l'architecture. La méthode est composée de 4 dimensions7, les 3 premières correspondent à des points de vue différents qui permettent l'observation d'un système d'information : *les aspect du système*, *le niveau d'agrégation du système*, et *le cycle de vie du système* ; la quatrième dimension *le niveau de raffinement* concerne le niveau de détail de la description du système.

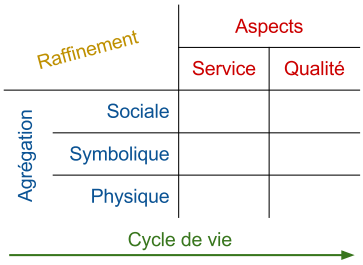

FIGURE 7 – Les 4 dimensions de GRAAL

La première dimension s'occupe des propriétés extérieures observables du système classées en deux catégories : les services et les qualités offerts par le système et attendus par l'utilisateur [13]. La deuxième dimension hiérarchise le système en 3 types : le monde physique (ordinateurs, réseaux...), le monde sociale (processus métier, normes, lois, argents...) et le monde symbolique (logiciels et documents). La notion de temps est abordé par la troisième dimension qui s'occupe des différentes phases dans la vie d'un système d'information : conception, versions successives, fréquences de mise à jour, maintenance, utilisation...

C'est la deuxième dimension qui nous importe le plus ; c'est elle qui va permettre la réalisation de l'alignement. La méthode GRAAL indique que chaque monde (physique, social, symbolique) doit-être aligné aux deux autres. Plus précisément les deux points de vue qui nous intéressent dans le présent article sont les points de vue métier et applicatif. GRAAL énonce le principe d'alignement suivant : *Pour aligner les logiciels (le symbolique) aux processus métier (le social), nous devons aligner les services offerts par le logiciel aux services offerts par les processus* [14]. Retenons donc qu'aligner c'est toujours se ramener à comparer des éléments comparables.

12. Guidelines Regarding Architecture Alignment

6.2 Les différentes techniques de mise en relation des modèles

Plusieurs méthodes sont possibles pour relier des modèles entre eux. Nous avons utilisé le tissage car il a l'avantage d'être non intrusif, néanmoins il existe d'autres technologies que nous avons étudiées.

L'extension de méta-modèles Une méthode possible est d'étendre chaque méta-modèle à mettre en relation pour inclure les concepts opposés à relier. En effet, il est tout à fait possible de réaliser une référence vers le concept d'un autre méta-modèle, notamment un lien d'association, cette méthode à été testé avec EMF. Dans notre cas d'alignement des modèles métiers et applicatif, cela consisterai à modéliser une association dans le méta-modèle métier vers le concept de **Service applicatif** provenant du méta-modèle applicatif, et dans le méta-modèle applicatif de modéliser une association vers le concept de **Tâche** provenant du méta-modèle métier. Néanmoins chaque cas d'étude n'utilisera pas systématiquement le même méta-modèle métier, ainsi il faudra réaliser cette modification pour chaque cas. De plus si un méta-modèle normé est utilisé, sa modification le rend non conforme : par exemple, BPMN.

Définition d'un unique méta-modèle globale Une autre méthode est de créer un méta-modèle global qui inclut à la fois les concepts métier et applicatif, ainsi que les liens entre les concepts des deux couches initiales. Cette méthode a comme désavantage de devoir créer des "super méta-modèles" pour chaque nouveau cas rencontré.

Utilisation de *Facet* Une autre technologie existe pour réaliser une mise en relation non intrusive par l'utilisation du projet *open-source* : Eclipse EMF Facet [13]. Ce projet permet d'étendre un modèle EMF par le mécanisme de facette qui ajoute virtuellement des attributs, des références et des opérations à un concept préexistant d'un méta-modèle. La facette fonctionne grâce à l'implantion de requêtes qui peuvent être écrites en Java, en OCL, ou en Javascript. L'inconvénient d'EMF Facet est que les requêtes doivent être exécutées au chargement du modèle ce qui peut avoir un impact négatif sur les performances.

L'étude de différentes méthodes, nous a permis de choisir la méthode la plus souple : le tissage. Elle permet de ne pas redéfinir de nouveau méta-modèle pour chaque cas d'étude. Le méta-modèle de tissage étant lui générique et défini une fois pour toute ; il peut contenir des liens de concepts en provenance de n'importe quel méta-modèle.

6.3 L'IDM pour l'architecture d'entreprise

L'utilisation de l'ingénierie des modèles appliquée à l'architecture d'entreprise et la transformation de modèles à base de technologie Eclipse EMF n'est pas une nouvelle approche. D. Castro et al.[6] décrivent une approche "*top-down*" qui vise à générer une architecture web orientée service à partir de modèles métier. Ils détaillent de façon similaire à notre approche les règles de transformation et les concepts qui relient les différentes couches.

Cependant, nous ne partageons pas le même but. Notre point de départ est le patrimoine applicatif existant, quel qu'il soit. Notre méthode doit répondre à l'homogénéité des architectures techniques existantes dans un SI et des différentes représentations métiers possibles de ce SI. La démarche D. Castro est uniquement hétérogène à une architecture orientée service, or une structure homogène est plus facilement traçable entre les couches.

Notre méthode ne se prétend pas être universelle avec tous les langages de programmation et toutes les notations de modélisation métiers, une adaptation plus ou moins importante sera nécessaire pour chaque cas rencontré.

13. http ://www.eclipse.org/modeling/emft/facet/

7 Conclusion et perspectives

Nous avons proposé une approche pragmatique au problème d'alignement des points de vue métier et informatique dans l'urbanisation des architectures d'entreprise. Notre solution est basée sur une proposition de modèles intermédiaires génériques, un rapprochement des points de vue et un tissage des concepts. Le rapprochement est rendu possible par une abstraction progressive du code en architecture applicative à base de composants et services. Concrètement cette abstraction est une rétro-ingénierie basée sur un processus de trois transformations de modèles. L'alignement effectif est implanté par un assistant de tissage de modèles incluant un mécanisme de contrainte des liens entre méta-modèles.

L'ensemble de notre méthode est automatisé dans le cadre d'Eclipse EMF et a été expérimenté sur un cas réel d'entreprise. Ce cas d'étude nous a permis d'éprouver notre approche sur une application issue de la réalité d'un SI d'une société d'assurance mutuelle. Cette application a un source code Java de taille importante, ce qui a été un défi pour que les modèles obtenus par rétro-ingénierie puissent être chargé par nos outils. De bonnes pratiques de codage, notamment pour la nomenclature des noms ont permis de conserver une certaine traçabilité entre les concepts métiers et ceux de l'application. Cette transformation sera nécessairement adaptée pour chaque cas d'étude, et trouver un algorithme générique pour détecter les composants d'architecture reste un problème complexe de génie logiciel [1]. Nous regrettons ne pas avoir accès au source code du référentiel MEGA pour réaliser un tissage plus complet. Ce qui nous aurait ensuite permis d'envisager une analyse et des mesures d'alignement.

Notre approche permet à ce stade un alignement des modèles métier et applicatif ; cependant aucune analyse, aucune mesure ne permet d'indiquer la qualité de l'alignement. Par conséquent la suite des travaux est de définir des règles pour réaliser différents types d'analyse : couverture, découplage, et dépendance. L'analyse peut être dirigée par les deux points de vue : métier ou applicatif. Par exemple, une analyse permettrait à partir d'une activité d'un modèle processus de connaître les différents composants applicatifs qui sont adhérents. L'analyse peut également être réalisée par les données pour déterminer où sont disséminé dans le SI des objets métier ; ou par les traitements afin d'identifier si le flot d'exécution est conforme à l'enchaînement des tâches d'un processus. Toutes les règles pourraient être exécutées en lot à la demande pour effectuer une analyse qualité régulière sur le SI. La définition de violation par un niveau de seuil permettrait d'obtenir des indicateurs et un ordre de priorité pour procéder à des actions de correction.

Une autre perspective est l'enrichissement du tissage avec plus de concepts pour mieux prendre en compte les points d'évolution du système d'information, pas seulement la structure [21]. En particulier, nous souhaitons pouvoir représenter le couplage entre parties de modèles. Dans cette lignée, nous devons mettre à l'épreuve nos méta-modèles vis-à-vis de pratiques (non automatisées) d'urbanisation et d'alignement.

Références

[1] Nicolas Anquetil, Jean-Claude Royer, Pascal André, Gilles Ardourel, Petr Hnetynka, Tomas Poch, Dragos Petrascu, and Vladiela Petrascu. JavaCompExt : extracting architectural elements from java source code. In *Proceedings of WCRE 2009*, pages 317–318. IEEE, 2009.

[2] L. Aversano, C. Grasso, and M. 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 LNBIP, pages 471–488. Springer, January 2013.

[3] Hugo Brunelière and Grégoire Dupé. Virtual EMF - transparent composition, weaving and linking of models. In *EclipseCon Europe 2011*, November 2011.

[4] J. Capirossi. *Architecture d'entreprise*. Collection Management et informatique. Hermes, 2011.

[5] Jae Choi, Derek L. Nazareth, and Hemant K. Jain. The impact of SOA implementation on IT-Business alignment : A system dynamics approach. *ACM Trans. Manage. Inf. Syst.*, 4(1) :3 :1–3 :22, April 2013.

[6] V. De Castro, E. Marcos, and Juan M. Vara. Applying CIM-to-PIM model transformations for the service-oriented development of information systems. *Inf. Softw. Technol.*, 53(1) :87–105, January 2011.

[7] Philippe Desfray and Gilbert Raymond. *TOGAF en pratique : Modèles d'architecture d'entreprise*. Management des systèmes d'information. Dunod, 1 edition, 2012.

[8] 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.

[9] DISIC and Government of France. Cadre commun d'Architecture d'Entreprise applicable au système d'information de l'Etat et à sa transformation, 2012.

[10] Marius Duedahl, Jostein Andersen, and Maung K. Sein. When models cross the border : Adapting IT competencies of business managers. In *Proceedings of the 2005 ACM SIGMIS CPR Conference*, SIGMIS CPR '05, page 40–48, New York, NY, USA, 2005. ACM.

[11] 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, page 1 :1–1 :4, New York, USA, 2011. ACM.

[12] The Open Group. *TOGAF Version 9.1*. van Haren Publishing, 10th edition, 2011.

[13] Brahim Lahna, Ounsa Roudiès, and Jean-Pierre Giraudin. Approches par points de vue pour l'ingénierie des systèmes d'information. *e-TI - la revue électronique des technologies d'information*, Numéro 5, 2009.

[14] Marc M. Lankhorst. *Enterprise Architecture at Work - Modelling, Communication and Analysis (3. ed.)*. The Enterprise Engineering Series. Springer, 2013.

[15] Jerry Luftman, Raymond Papp, and Tom Brier. Enablers and inhibitors of business-IT alignment. *Commun. AIS*, 1(3es), March 1999.

[16] Chantal Morley. *Management d'un projet Système d'Information - Principes, techniques, mise en oeuvre et out : Principes, techniques, mise en oeuvre et outils*. Management des systèmes d'information. Dunod, 7 edition, 2012.

[17] Kestutis Normantas, Sergejus Sosunovas, and Olegas Vasilecas. An overview of the knowledge discovery meta-model. In *Proc. of the 13th International Conference on Computer Systems and Technologies*, CompSysTech '12, page 52–57, NY, USA, 2012. ACM.

[18] Jacques Printz. *Architecture logicielle - Concevoir des applications simples, sûres et adaptables*. Etudes et développement. Dunod, 3 edition, 2012.

[19] Robert Reix. *Systèmes d'information et management des organisations*. Vuibert, Paris, 2011.

[20] Gérard Roucairol and Yves Caseau. *Urbanisation, SOA et BPM : Le point de vue du DSI*. InfoPro, Management des systèmes d'information. Dunod, 4 edition, 2011.

[21] 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.

[22] The Open Group. *Archimate 2.1 Specification.* Van Haren Pub, 2013.

[23] 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.

[24] 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.