De la modélisation à l'automatisation des prises de décisions opérationnelles avec une démarche d'Architecture d'Entreprise

Thierry Biard

université PARIS-SACLAY

NNT : 2017SACLC072

Thèse de doctorat

CentraleSupélec

De la modélisation à l'automatisation des prises de décisions opérationnelles avec une démarche d'Architecture d'Entreprise

Thèse de doctorat de l'Université Paris-Saclay
préparée à CentraleSupélec

École doctorale n° 573 Interfaces
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Gif-sur-Yvette, le 27 novembre 2017, par

Thierry Biard

Composition du Jury :

Eric Bonjour Professeur des Universités, ENSGSI Nancy	Président
David Chen Professeur des Universités, Université Bordeaux – IMS	Rapporteur
Hervé Pingaud Professeur des Universités, INU Champollion Albi	Rapporteur
Bernard Yannou Professeur des Universités, CentraleSupélec – LGI	Examinateur
Jean-Claude Bocquet Professeur émérite, CentraleSupélec – LGI	Directeur de thèse
Michel Bigand Maître de Conférences HDR, Centrale Lille	Examinateur
Jean-Pierre Bourey Professeur des Universités, Centrale Lille	Examinateur

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

ADF	Activity-Decision Flow
ADM	Architecture Development Method
BMM	Business Motivation Model
BPMN	Business Process Model and Notation
BPSS	Business Process Specification Schema
BRE	Business Rules Engine
BRMS	Business Rules Management System
CIMOSA	Open System Architecture for Computer Integrated Manufacturing
CMMN	Case Management Model and Notation
DMN	Decision Model and Notation
DRD	Decision Requirements Diagram
DRL	Drools Rule Language
DSL	Domain Specific Language
EA	Enterprise Architecture
EAM	Enterprise Architecture Management
EDOC	Electronic DOCument
EPC	Event-driven Process Chain
ERP	Enterprise Resource Planning
FEEL	Friendly Enough Expression Language
GERAM	Generalized Enterprise Reference Architecture and Methodology
GRAI	Graphe à Résultats et Activités Inter-reliés
IAF	Integrated Architecture Framework
IDEF	Integration DEFinition
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
JDK	Java Development Kit

MDA	Model Driven Architecture
MOF	Meta-Object Facility
OWL	Web Ontology Language
PIM	Platform Independent Model
PoC	Proof of Concept
PSM	Platform Specific Model
RDF	Resource Description Framework
RIF	Rule Interchange Format
RSE	Responsabilité Sociale et Environnementale
RuleML	Rule Markup/Modeling Language
SBVR	Semantics of Business Vocabulary and Business Rules
SGBD	Systèmes de Gestion de Bases de Données
SOA	Service Oriented Architecture
SoC	Separation of Concerns
SQL	Structured Query Language
TDM	The Decision Model
TOGAF	The Open Group Architecture Framework
TSE	Topologie du Système Entreprise
UEML	Unified Enterprise Modelling Language
UML	Unified Modeling Language
XML	eXtensible Markup Language
XOR	eXclusive OR
XSD	XML Schema Definition

1 Introduction Générale

Après la description du contexte de la transformation de l'Entreprise, avec l'aide de l'Architecture et des modèles, la problématique pose la question de la modélisation, voire de l'automatisation, des prises de décision opérationnelles.

Après un état de l'art des solutions existantes, qui se révèlent limitées, une proposition de solution est présentée, puis sa validation est effectuée. La conclusion synthétise notre contribution, puis évoque quelques perspectives intéressantes.

Voyons plus en détail la structure de ce mémoire de thèse :

Chapitre 2 Contexte : L'Entreprise, qu'elle soit privée ou publique, est en pleine mutation. Elle doit relever de nouveaux défis, comme l'agilité afin de s'adapter rapidement aux changements imposés par son environnement, mais aussi répondre aux besoins de ses clients de plus en plus exigeants, tout en profitant des opportunités offertes par les nouvelles technologies. Pour arriver à ses fins, elle doit mener avec rigueur un projet de transformation.

L'Architecture d'Entreprise, en tant que discipline, avec un cadre de représentation, comme celui de la méthode Praxeme par exemple, permet de maitriser sa complexité intrinsèque avant de commencer son projet de transformation de l'Entreprise. Encore faut-il que ce cadre soit rempli de modèles pertinents pour chaque besoin de représentation.

Les modèles utilisant un langage standard permettent assurément une meilleure communication entre les parties prenantes. Ils sont aussi plus facilement réutilisables entre les unités organisationnelles de l'Entreprise (voire au sein d'un groupe d'entreprises) ayant des activités similaires et concourent à une meilleure synergie des ressources nécessaires.

Concernant la démarche globale d'Architecture d'Entreprise, c'est la méthode Praxeme qui est utilisée. Cette méthode permet de représenter l'Entreprise dans sa globalité selon 7 aspects différents. Chaque aspect représente une portion de la réalité, isolée pour en faciliter l'étude, en respectant sa logique interne ; l'aspect se trouve du côté de l'objet étudié.

L'Entreprise globale, représentée par ses 7 aspects, est un champ d'investigation bien trop large pour une thèse. Il est néanmoins intéressant d'étudier ses principaux aspects. Il ne s'agira pas de survoler ces aspects superficiellement, mais de « prélever » un échantillon représentatif de chaque aspect, de la même façon que l'on effectue un carottage en géologie.

Chapitre 3 Problématique : La représentation des processus métier, qui tient désormais une place prépondérante dans l'Entreprise, peut d'appuyer sur un langage standard et international : BPMN (Business Process Model and Notation). Ce langage propose un ensemble d'éléments graphiques et différents diagrammes, tel que le diagramme de collaboration, qui permet de capturer toute la complexité des processus, qu'ils soient de niveau métier ou de niveau technique pour leur mise en œuvre dans des outils d'exécution.

Mais la modélisation des processus ne saurait constituer à elle seule la représentation du savoir-faire de l'Entreprise. La modélisation des prises de décisions opérationnelles, incluant la formalisation des règles métier, fait partie de ce savoir-faire également et répond aux besoins croissants de transparence et de partage des connaissances des collaborateurs de l'Entreprise.

Lorsqu'il s'agit de représenter ces prises de décision opérationnelles, selon des règles métier, nous démontrons que le diagramme de collaboration BPMN a atteint ses limites. Ceci nous amène à poser la question de recherche suivante :

Comment modéliser les prises de décisions opérationnelles, voire les automatiser, tout en garantissant la cohérence avec les autres modèles de l'Entreprise, notamment ceux qui représentent les processus métier, avec une démarche globale d'Architecture ?

Chapitre 4 Etat de l'Art : L'état de l'art présente donc les méthodes et cadres d'Architecture d'Entreprise, les langages pour la modélisation des processus métier et enfin les langages existants pour la modélisation des prises de décisions opérationnelles. Ces langages ne répondent que partiellement à cette question de recherche. Nous terminerons cet état de l'art en introduisant le nouveau langage DMN (Decision Model and Notation) définie par l'OMG (Object Management Group), sur laquelle s'appuiera notre contribution.

Chapitre 5 Proposition de Solution : Ce nouveau langage DMN, qui a été proposée pour répondre à cette question précise de modélisation des prises de décisions opérationnelles et des règles métier, en complément de BPMN pour la modélisation des processus métier, est développée dans ce chapitre. Les éléments graphiques pour composer les diagrammes et les tables de décision pour formaliser les règles métier sont détaillés.

Chapitre 6 Validation de la Solution : Le langage DMN est mis en pratique sur un cas réel. Après la modélisation en langage DMN, l'automatisation des prises de décisions opérationnelles, en utilisant un moteur de règles, est mise en œuvre. Cette possibilité technique intéressante, qui permet de s'affranchir de plusieurs inconvénients des prises de décisions humaines, est également validée.

Chapitre 7 Conclusion et Perspectives : Une synthèse de notre contribution est présentée, puis quelques pistes sont ouvertes pour continuer l'exploration de la modélisation puis de l'automatisation des prises de décisions opérationnelles.

Carte conceptuelle : La carte conceptuelle ci-dessous positionne les principaux concepts abordés dans ce mémoire de thèse (sauf les indicateurs pour mesurer la performance de l'Entreprise et leur organisation dans le tableau de bord).

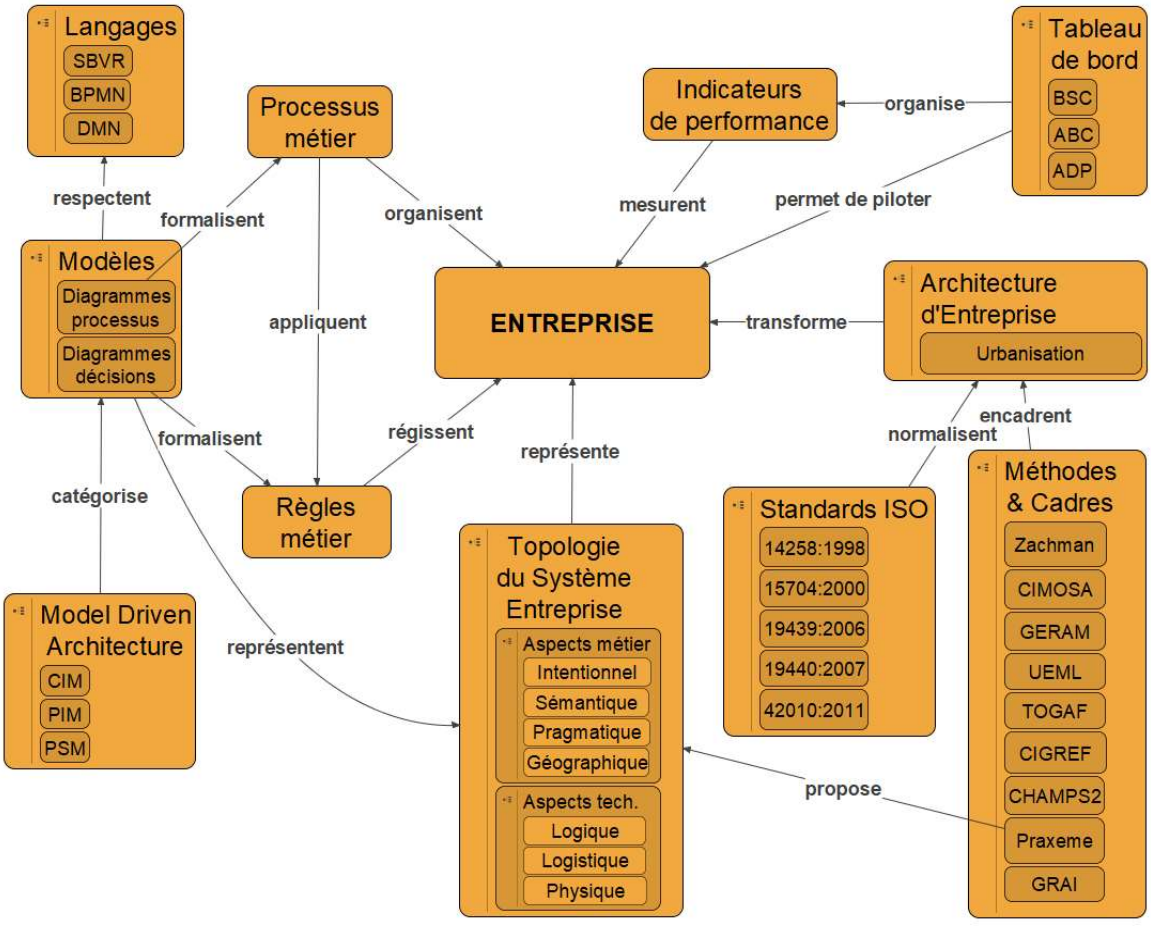


Figure 1 : Carte conceptuelle de la thèse

2 Contexte

2.1 Transformation de l'Entreprise

2.1.1 Trajectoire de transformation de l'Entreprise

Dans un contexte économique difficile et notamment pour s'adapter à l'ère numérique, l'Entreprise doit se transformer. Elle doit être agile pour pouvoir se transformer constamment et rapidement. L'Entreprise est considérée comme un système sociotechnique complexe.

La transformation de l'Entreprise consiste à aller du système existant tel qu'il est « As-is » vers le futur système « To-be », en effectuant un ensemble d'activités qui la modifient et lui permettent d'atteindre les objectifs définis par sa stratégie, tout en suivant une trajectoire maitrisée, car prédéfinie. La Figure 2 symbolise cette trajectoire simplifiée :

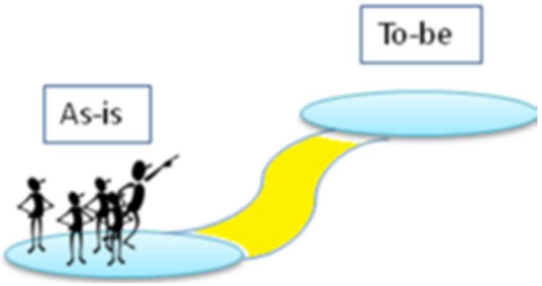


Figure 2 : Trajectoire simplifiée de transformation d'entreprise

2.1.2 Cycle du projet de transformation de l'Entreprise

Pour gagner différents enjeux détaillés ci-après, il convient de mener un (ou plusieurs) projet de transformation. Afin de représenter l'Entreprise, telle qu'elle est « As-is » et telle qu'elle voudrait être « To-be ». Afin de comprendre, voire de simplifier, son fonctionnement complexe, il est recommandé d'élaborer des modèles. Ces modèles sont des représentations partielles de la réalité, mais doivent être bien adaptés au contexte.

Avant de commencer un projet de transformation, il est recommandé [Mamoghli, 2013], d'imaginer des états intermédiaires du système, représentés également par des modèles, afin d'explorer le champ des possibles. On peut alors désirer le système idéal « As-wished », puis le réaliser grâce à des développements sur-mesure, ou choisir un progiciel qui répond au mieux aux besoins exprimés, voire faire des concessions, si ce n'était pas le cas. A l'inverse, il arrive qu'un progiciel dispose de fonctionnalités qui n'ont pas été demandées, mais qui se révèlerait utiles « Might-be ». La Figure 3 représente ces 4 états du système par autant de modèles.

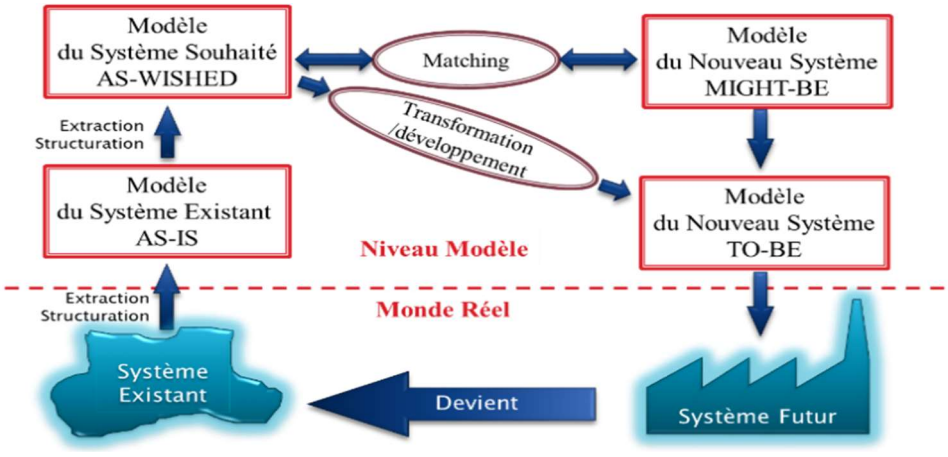


Figure 3 : Cycle du projet de transformation de l'Entreprise

La transformation peut, comme l'amélioration, être considérée comme un processus continu. Ainsi, le système futur réalisé devient le système existant et l'on peut commencer une nouvelle itération de transformation. C'est une forme de cercle vertueux et il faut profiter de chaque itération pour affiner les modèles, qui deviennent plus précis et qui représentent mieux la réalité.

2.2 Architecture d'Entreprise

2.2.1 Définition de l'Architecture d'Entreprise

L'Architecture d'Entreprise est une discipline (ensemble de règles de conduite imposées) qui permet de concevoir l'Entreprise dans ses principaux aspects, du métier à la technique. L'Architecture d'Entreprise est un des moyens pour mener à bien son projet de transformation (en complément de la conduite du changement et du pilotage).

L'Entreprise doit s'appuyer sur **l'Architecture d'Entreprise (composée essentiellement d'une méthode, d'un cadre de représentation et de modèles)** pour maitriser sa complexité, puis gouverner sa transformation. Il faut également utiliser des langages ou des notations – si possible standardisés – et des outils, pour élaborer les modèles.

2.2.2 Enjeux de l'Architecture d'Entreprise

L'Architecture d'Entreprise permet de faire face à différents enjeux, aussi variés qu'ambitieux, notamment :

- aligner son Système d'Information sur sa stratégie,
- maitriser la complexité et réduire la complication,
- changer les processus métier pour respecter la règlementation qui évolue,
- augmenter l'efficience et diminuer les coûts,
- lancer un programme de transformation numérique,
- réussir une fusion-acquisition de sociétés pour agrandir leur périmètre,
- réduire au contraire le périmètre des activités, voire les effectifs,
- maitriser les évolutions technologiques,
    - profiter des nouvelles technologies
    - gérer l'obsolescence des anciennes technologies,
- communiquer efficacement le projet de transformation entre les différents partis.

L'architecture de l'Entreprise doit avant tout être alignée sur sa stratégie. Mais, la mise en place de l'Architecture d'Entreprise peut, voire doit, faire partie de la stratégie de l'Entreprise (« Enterprise Architecture as Strategy » [J. W. Ross et al., 2006]) et constitue elle-même un enjeu majeur.

2.2.3 Différents types d'alignements

L'Architecture d'Entreprise peut être décomposée en amont en Architecture Métier (son offre de produits et de services, son organisation, ses processus, ses règles, ses décisions, etc.) et en aval en Architecture Technique (un ensemble cohérent de logiciel et de matériel, pour ne citer que les deux composantes qui concernent son Système d'Information ; de la logistique de façon plus générale).

L'Architecture Technique met à disposition de l'Entreprise les ressources nécessaires lui permettant d'exercer les activités définies par l'Architecture Métier. Idéalement, l'Architecture Technique doit donc être parfaitement alignée sur l'Architecture Métier, elle-même préalablement alignée sur la Stratégie de l'Entreprise, mécanisme connu sous le nom d'Alignement Stratégique [Henderson et Venkatraman, 1992].

2.2.4 Architecture d'Entreprise et Urbanisation

L'Architecture d'Entreprise est une discipline qui a commencé à partir des années 1990 aux Etats-Unis d'Amérique, puis s'est développée à travers le monde, les pays anglo-saxons d'abord, puis les autres.

A la même période, en France, la discipline similaire était l'Urbanisation, une métaphore de l'aménagement des villes appliquée aux Systèmes d'Information. Bien que le terme Architecture d'Entreprise ait tendance désormais à supplanter le terme Urbanisation, des différences existent entre ces deux disciplines, mais elles restent très proches [Longépé, 2009].

Ces deux disciples partagent notamment le même objectif d'accroitre l'agilité du Système d'Information [CIGREF, 2003]. Le Club Urba-SI (SI comme Systèmes d'Information) créé en 2000, qui regroupe des urbanistes et des architectes de grandes entreprises françaises, a été renommé en Club Urba-EA (EA comme Enterprise Architecture) en 2006, afin de suivre cette tendance globale.

2.2.5 Besoin d'une méthode

Une méthode doit répondre à la question « Comment faire ? » en indiquant notamment des actions ordonnées à exécuter et des règles à respecter pour atteindre un but. Elle doit y répondre de manière suffisamment précise, par étapes successives, afin que la façon de faire soit identique à chaque utilisation et garantisse le même résultat.

Une méthode peut s'appuyer sur une ou plusieurs théories. Elle constitue alors la mise en pratique de cette théorie. Une méthode doit être documentée, voire si possible outillée si la façon de faire est complexe. Elle peut être décomposée en processus, en procédures et en procédés ou modes opératoires. Une méthode permet également d'apprendre une science ou une technique, de transmettre la connaissance, voire de s'imposer une discipline, comme celle de l'Architecture d'Entreprise.

Dans notre cadre d'étude, une méthode doit notamment appliquer des principes et des règles d'architecture sur son cadre de représentation et des principes et des règles de modélisation, mais aussi de transformation, sur ses différents modèles.

Une méthode d'Architecture d'Entreprise doit permettre de prendre et de justifier des décisions. Nous présenterons plus loin les différents types de décisions. Les décisions d'architecture, celles qui sont prises en appliquant la méthode, sont généralement de type tactique. Procéder avec méthode reste le meilleur moyen de prendre les bonnes décisions.

2.2.6 Besoin d'un cadre de représentation

L'Architecture d'Entreprise s'appuie sur un cadre de représentation (*framework*) qui est constitué d'un socle de référence (idéalement, un métamodèle, c.-à-d. un modèle générique pour les modèles spécifiques de l'Entreprise), sur lequel une méthode plus ou moins aboutie peut être élaborée. Un cadre de représentation permet d'organiser la perception de l'Entreprise.

Selon les bonnes pratiques, un cadre idéal devrait respecter certains principes comme :

- la focalisation sur l'Entreprise,
- la prise en compte de toute sa réalité (approche holistique),
- l'unicité de tout élément à l'intérieur du cadre
  (et son corollaire, le bannissement de la redondance).

2.2.7 Besoin de modèles

2.2.7.1 Modélisation d'Entreprise

Modéliser l'Entreprise dans sa globalité, afin d'expliciter sa structure et son fonctionnement complexe, c'est un des buts de l'Architecture d'Entreprise. Un modèle peut servir également à analyser son comportement, voire estimer les performances de l'Entreprise via la simulation. Enfin, un modèle peut servir d'aide à la décision, lors de son exploitation. Les modèles permettent aussi de proposer des points de vue différents, mais cohérents, de l'Entreprise, adaptés à chacun, qu'il soit spécialiste métier, informaticien ou autre. Ils fournissent une représentation consensuelle et partagée de l'Entreprise.

Les modèles ont rarement une représentation graphique uniquement : ils sont généralement représentés par des diagrammes (notation graphique) et du texte. Certains langages, comme BPMN, séparent bien le modèle (les objets à représenter) de sa représentation graphique (les symboles qui représentent ces objets et leur disposition sur un écran ou sur un page). Ainsi, modifier la disposition des symboles de la représentation graphique ne modifie le modèle.

Afin de représenter et comprendre l'Entreprise sous tous ses aspects, les besoins de représentation par des modèles sont multiples. On voudrait représenter depuis longtemps sa mission, ses objectifs, sa stratégie, ses activités sous forme de processus métier, son vocabulaire, ses règles métier voire ses contraintes, l'organisation de ses ressources, humaines notamment, son offre de produits et de services et leurs cycles de vie, la structure de ses données, les flux physiques et les flux d'information (liste non limitative). La représentation des prises de décision est un besoin plus récent, qui est exprimé depuis une dizaine d'années seulement, et qui sera développée dans la suite de ce mémoire.

2.2.7.2 Utilité, utilisabilité et utilisation d'un modèle

Tout le monde s'accorde à dire qu'un modèle est utile : ce postulat de départ ne sera pas démontré. Il faut toutefois que l'objectif du modèle fasse partie des questions préliminaires à la modélisation, afin de justifier son utilité.

Il faut également qu'un modèle soit si nécessaire utilisable. Par exemple, un modèle représentant un processus métier, avec un fort niveau de détail, peut devenir exécutable. C'est le meilleur moyen pour s'assurer que le modèle représente bien la réalité (ou bien éventuellement que la réalité soit alignée sur le modèle).

Enfin, à chaque utilisation d'un modèle dans le monde réel, une instance est créée (l'instance est l'action de créer un « objet » à partir d'un modèle). Idéalement, on devrait pouvoir compter le nombre d'instances d'un modèle, c.-à-d. mesurer son utilisation réelle, pour piloter efficacement les opérations. Par exemple, il sera intéressant de compter combien d'instances il y a eu du modèle de processus métier « Traitement d'une commande », car ce comptage indiquera le nombre de commandes traitées.

2.2.7.3 Principes de modélisation

Toute démarche d'Architecture s'appuie sur la modélisation de l'Entreprise et ses principes :

- la cohérence des modèles,
- la mise en relation des éléments les uns avec les autres,
- l'aide à la transformation d'un modèle vers un autre,
- l'existence d'un métamodèle de référence.

2.2.7.4 Pyramide de modélisation

2.2.7.4.1 (Bon) Modèle

Un bon modèle doit représenter au mieux une partie du monde réel. Une seule classe d'objets dans un modèle peut représenter des millions d'objets (alias des instances) dans le monde réel. De manière plus précise, [Brackett, 1990] nous indique les avantages d'un bon modèle :

(1) réduit le niveau de complexité et doit être compris du premier coup,

(2) est plus économique à construire et à modifier qu'un système réel,

(3) facilite la description des aspects complexes d'un système réel.

Certains auteurs préfèrent parler de modèles pertinents (*relevant*). Nous avons toutefois vu des modèles qui semblaient pertinents (vus du modélisateur), mais qui étaient loin d'être compris par tous. D'autres auteurs encore parlent de *truthful models*. Les traductions en français de *« truthful »* sont multiples, mais la plupart de ces traductions correspondent bien aux qualités d'un « bon » modèle : véridique, honnête, vrai & fidèle. Il y a quand même de bons modèles qui simplifient délibérément la réalité, afin de faciliter sa compréhension.

Voici une liste de 7 critères pour tenter de qualifier un bon modèle [Mader et al., 2007] :

- Il a un objet de modélisation clairement spécifié,
- Il a un objectif clairement spécifié,
- Il est simple,
- Il est véridique,
- Il est traçable,
- Il est extensible et réutilisable,
- Il est conçu pour l'interopérabilité et le partage.

Tandis que les 4 premiers critères génériques concernent tous les domaines métier, les 3 derniers critères répondent précisément à des besoins dans le domaine de l'Architecture d'Entreprise, qui nous intéresse ici :

Traçabilité : chaque élément du modèle doit correspondre à un aspect précis de l'objet de modélisation et cette relation bidirectionnelle doit être clairement documentée. Si la traçabilité est assurée, cela permet d'envisager notamment la transformation d'un modèle vers un autre.

Extensibilité et réutilisabilité : il y a rarement une seule version d'un modèle. Celui-ci doit pouvoir évoluer. C'est même indispensable quand un modèle sert de base à l'automatisation d'une activité et qu'un changement s'avère nécessaire. Quant à la réutilisabilité, il s'agit de capitaliser les modèles existants, rangés dans le cadre de représentation de l'Entreprise.

Interopérabilité et partage : il est recommandé qu'un modèle réalisé respecte un formalisme standard, afin de faciliter sa compréhension et son partage. L'étymologie de « standard » nous apprend que ce mot vient de « étendard », qui sert de point de ralliement, et cette métaphore semble adaptée pour un bon modèle. Le choix d'un formalisme standard permet généralement d'échanger les modèles entre des outils différents, grâce à des fonctionnalités d'export puis d'import : c'est un cas concret d'interopérabilité.

2.2.7.4.2 Métamodèle

Pour respecter rigoureusement ce formalisme standard, ce modèle (le plus souvent représenté par un diagramme) doit être conforme à un métamodèle (rappel : modèle générique pour les modèles spécifiques). Ainsi un diagramme de collaboration pour représenter un processus métier doit être conforme au métamodèle BPMN (Business Process Model and Notation) et un diagramme des exigences pour représenter une prise de décision doit être conforme au métamodèle DMN (Decision Model and Notation), comme représenté sur la Figure 4.

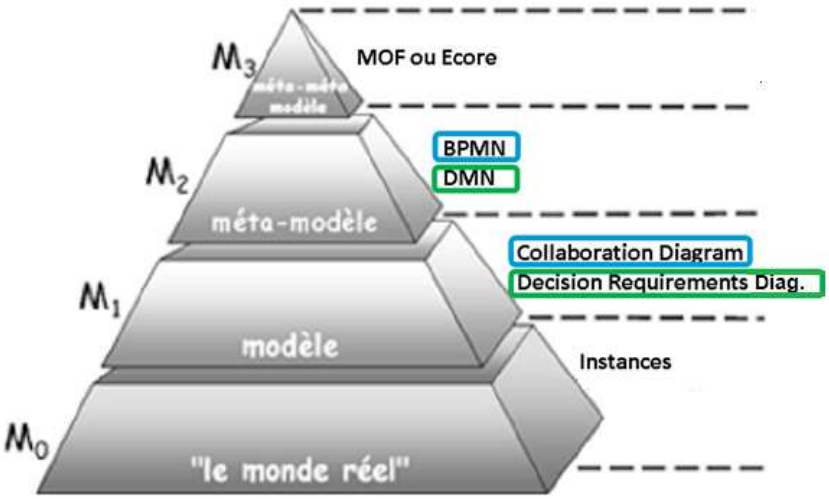


Figure 4 : Pyramide de modélisation (OMG)

Tous les langages et notations standards présentés dans les chapitres suivants sont définis par des métamodèles. En théorie, il est indispensable de formaliser le métamodèle d'un langage avant de l'utiliser. Dans la pratique, les langages de modélisation sont souvent définis par un métamodèle incomplet dans leur première version. On peut citer par exemple la version 1.0 du langage BPMN dont le métamodèle ne contenait qu'une syntaxe abstraite décrite en langage naturel et qui fut donc sujette à mauvaise interprétation.

Dans la mesure où les développeurs des outils de modélisation ont respecté et implanté le métamodèle, ce sont ces outils qui vérifient la conformité des modèles avec leur métamodèle, soit au fur et à mesure de leur élaboration (il est alors impossible de dessiner un modèle non conforme), soit sur demande de l'utilisateur (ou bien lors de l'enregistrement), ce qui produit alors un rapport d'erreurs. Certains outils vérifient également le respect de bonnes pratiques ; si elles ne sont pas respectées, cela produit un rapport d'avertissements.

2.2.7.4.3 Métamétamodèle

Enfin, afin de formaliser la représentation des métamodèles, il existe un métamétamodèle. Idéalement, faudrait-il qu'il y en ait qu'un seul, pour formaliser tous les métamodèles ? En pratique et peut-être pour éviter tout dogmatisme, nous en connaissons au moins deux :

(1) MOF (Meta-Object Facility), défini par l'OMG,

(2) Ecore, défini au sein de la communauté Eclipse (cf. chapitre 6.4.1.2).

Certains se demanderont sans doute comment est formalisé le métamétamodèle. Pas besoin d'un niveau supplémentaire de représentation : le métamétamodèle est capable de se décrire lui-même, grâce à une fonctionnalité particulière dite de métacircularité. Ainsi, dans le monde OMG, le MOF est décrit en UML [Budinsky, 2007] tandis que, dans le monde Eclipse, le métamétamodèle est décrit en Ecore.

Le modèle, le métamodèle et le métamétamodèle constituent la pyramide de modélisation représentée sur la Figure 4 ci-dessus (Niveau M3 : métamétamodèle). La taille de chaque niveau de cette pyramide est proportionnelle aux nombres d'éléments concernés :

- deux métamétamodèles au niveau M3,
- des dizaines de métamodèles au niveau M2,
- des milliers de modèles au niveau M1,
- des millions d'instances dans le monde réel au niveau M0.

3 Problématique

3.1 Modélisation des processus métier

3.1.1 Approche processus généralisée

Poussée par les exigences en matière de qualité, comme celles définies dans le standard ISO 9001 (International Organization for Standardization a publié la première version en 1987 ; la dernière version date de 2015), puis de la méthode Six Sigma (Motorola en 1986), l'approche processus connait un succès grandissant depuis les années 1990, au point de devenir quasiment incontournable dans les entreprises, surtout les grandes bien sûr, mais aussi celles de taille intermédiaire.

A la même période, les évolutions technologiques, dont la révolution numérique, ont créé de nouveaux besoins, puis ont permis d'y répondre, par exemple la communication multicanal côté Client, la dématérialisation des documents (projets zéro papier, dits de transformation numérique désormais) côté Entreprise.

Et puis, le périmètre d'analyse pour l'amélioration de la performance s'est étendu en dehors de l'Entreprise, côté Clients (expérience utilisateur, parcours client), mais aussi côté Fournisseurs (logistique collaborative avec la Supply Chain), conduisant à partager les modèles de processus métier entre ces différents acteurs.

Plus récemment, l'entrée en vigueur de nouvelles lois, la nécessité de maitriser les risques, la Responsabilité Sociale et Environnementale (RSE) des entreprises ont constitué autant de raisons de modéliser les processus métier. Nous préférons ces exemples concrets à l'objectif idéal, mais trop générique de « viser l'excellence opérationnelle ».

Finalement, l'approche processus peut être une stratégie pour piloter l'entreprise en assurant la cohérence globale des différents processus métier.

3.1.2 Représentation d'un processus métier

La représentation usuelle d'un processus métier se fait sous la forme d'un enchainement d'activités (tâches élémentaires ou sous-processus) depuis un événement déclencheur (le début), jusqu'à un événement interrupteur (la fin). Tandis que le début d'un processus est souvent unique et bien identifié, les fins sont souvent multiples et doivent être correctement gérées. « Les événements interrupteurs sont les miroirs de la performance de l'entreprise » selon [Chelli, 2003].

Cet enchainement des activités qui compose un processus métier se fait dans un ordre prédéterminé, souvent en série, parfois en parallèle. La parallélisation des activités est le moyen le plus efficace de réduire la durée d'exécution d'un processus. Chaque activité peut être assignée à un rôle (comportement précis nécessitant les compétences adéquates, associé à un poste de travail par exemple). Il est recommandé de montrer les interactions avec les autres acteurs (les personnes qui prennent une part importante à une activité) internes (autres services dans la même entreprise) ou externes (clients, fournisseurs, Administration, etc.).

3.2 Modélisation des prises de décisions

3.2.1 Prises de décisions à différents niveaux

A tous les niveaux de l'entreprise, il faut prendre des décisions. On distingue généralement trois niveaux dans l'entreprise : le niveau stratégique, le niveau tactique et le niveau opérationnel. C'est l'accomplissement d'activités à chaque niveau qui conduit à prendre des décisions.

Et réciproquement, les décisions prises conduisent à effectuer ensuite d'autres activités. Ce sont donc les décisions qui permettent d'effectuer les activités différentes selon le contexte. Le Tableau 1 indique essentiellement les activités qui concernent la gestion des processus métier (listes à chaque niveau non exhaustives).

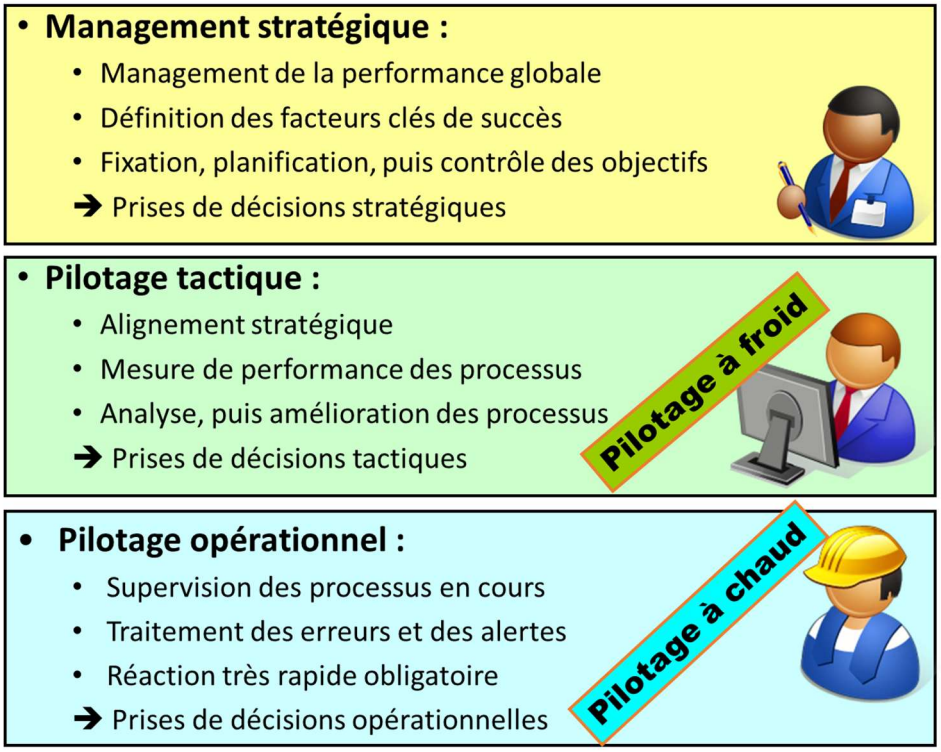


Tableau 1 : Prises de décisions à tous les niveaux de l'entreprise

Des mécanismes d'alerte proactifs, selon l'analyse des tendances, permettent désormais de prendre des décisions opérationnelles conduisant à des actions correctives, avant même que les erreurs ne surviennent. Il s'agit là d'un progrès certain, car ces erreurs sont souvent signalées par les acteurs externes participant aux processus métier, notamment les clients… Ces corrections de trajectoire dynamiques constituent un véritable pilotage à chaud des activités composant les processus métier.

L'automatisation des prises de décisions opérationnelles est une solution pour réagir très rapidement (plus rapidement que les prises de décisions humaines), mais la rapidité n'est pas son seul avantage, comme ce sera précisé plus loin.

3.2.2 Principaux Types de Décisions

Une décision est le fait d'un acteur (ou d'un ensemble d'acteurs) qui effectue un choix, si possible après réflexion, entre plusieurs solutions possibles pour affronter une situation difficile, résoudre un problème délicat ou répondre à une question complexe. Ces différentes situations, qui correspondent aux différents niveaux de l'Entreprise, permettent de distinguer trois grands types de décisions : les décisions stratégiques, les décisions tactiques et les décisions opérationnelles.

Le Tableau 2 présente les principales différences entre les décisions stratégiques-tactiques et les décisions opérationnelles. Nous ne séparons pas les décisions stratégiques des décisions tactiques, car nous allons en fait nous intéresser par la suite aux décisions opérationnelles.

Type de décision	Stratégique-Tactique	Opérationnelle
Environnement	Incertain	Prédéfini
Portée	Globale	Locale
Impact	Fort	Faible mais cumulatif
Terme	Long-moyen	Court
Fréquence	Petite-Moyenne	Grande
Période	Années-Mois	Secondes (Temps Réel)
Orienté processus	Faible	Fort
Prise de décision	Humaine avec Système d'Aide à la Décision	Automatisable
Exemple de décision	Recrutement de vacataires	Recevabilité d'un vacataire

Tableau 2 : Principaux types de décisions

Tandis que les décisions stratégiques et tactiques servent le management stratégique et le pilotage tactique (pilotage « à froid »), les décisions opérationnelles servent le pilotage opérationnel (pilotage « à chaud »). Dans un environnement prédéfini, il est possible d'associer les décisions opérationnelles aux processus métier et cette possibilité sera développée dans la suite de ce mémoire.

3.2.3 Impacts des décisions stratégiques-tactiques et opérationnelles

Une décision opérationnelle a généralement un impact faible à court terme. Mais la fréquence de ce type de décisions est grande et leurs impacts peuvent se cumuler. L'impact total de nombreuses décisions opérationnelles peut finalement être aussi important qu'une décision stratégique, même si leurs types sont différents. Cela est illustré par la Figure 5.

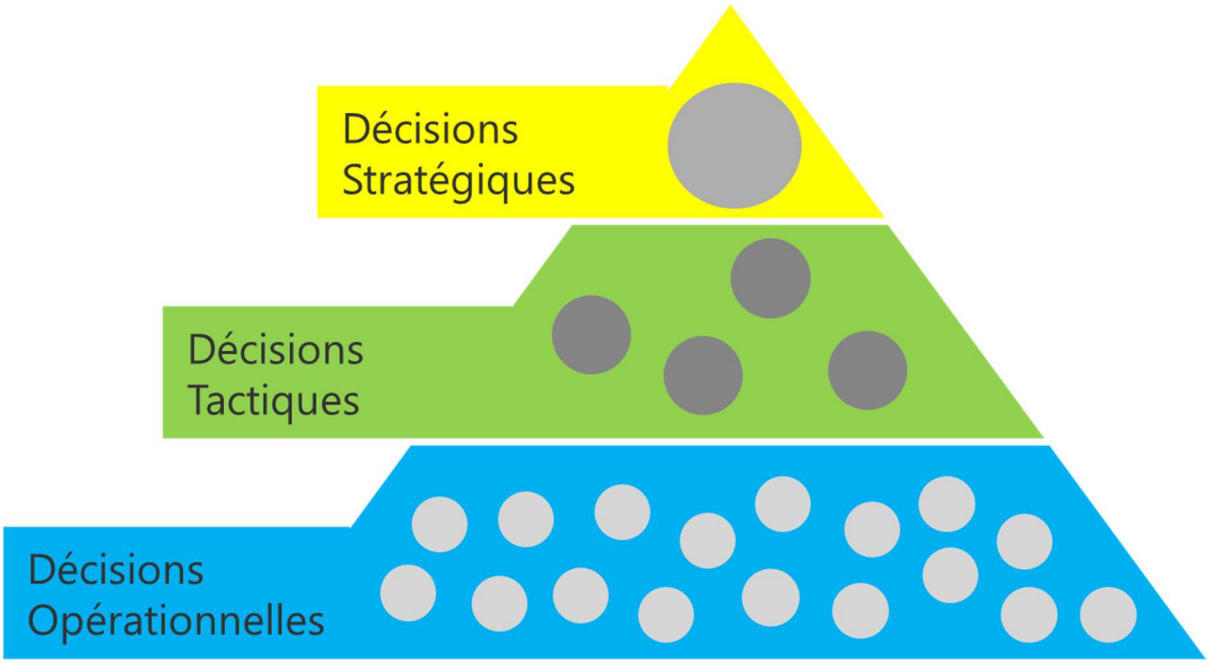


Figure 5 : Impact des différents types de décisions

3.2.4 Prises de décisions humaines versus automatisées

Les prises de décisions humaines comportent quelques inconvénients : règles métier appliquées floues, décisions parfois inconsistantes (en fonction du décideur, du moment et du lieu notamment, les décisions prises à partir de données d'entrée identiques seront pourtant différentes) et enfin ces décisions humaines sont souvent prises lentement. La logique de décision est souvent dans la tête des décideurs uniquement (pas de documentation). Elles sont, par contre, très faciles (voire parfois trop faciles) à modifier.

Les prises de décisions automatisées permettent d'améliorer certains points, qui deviennent même des avantages : les décisions sont toujours consistantes (elles sont identiques à partir de données d'entrées identiques, dans un environnement prédéfini) et elles peuvent être prises très rapidement (en quelques secondes).

Les prises de décisions automatisées ne sont toutefois pas exemptes d'inconvénients, surtout si elles n'ont pas été modélisées préalablement : elles ne correspondent pas forcément aux exigences (selon l'interprétation de ces exigences par les développeurs), elles manquent de visibilité (une fois qu'elles sont codées « en dur », voire cachées, dans les applications) et enfin elles sont plus difficiles à modifier (il faut lancer un mini-projet informatique à chaque demande de changement du métier), ce qui constitue un frein à l'agilité de l'entreprise.

Le Tableau 3 compare les avantages et les inconvénients des décisions humaines et des décisions automatisées (sans modélisation préalable).

Décisions humaines	Décisions automatisées
☒ Règles métier appliquées quelquefois floues,	☒ Ne correspondent pas forcément aux exigences,
☒ Parfois inconsistantes,	☑ Toujours consistantes,
☒ Prises lentement par moments,	☑ Prises très rapidement,
☒ Très rarement documentées,	☒ Manquent de visibilité,
☑ Faciles à modifier.	☒ Difficiles à modifier.

Tableau 3 : Décisions humaines versus décisions automatisées (sans modélisation)

Nous verrons plus loin qu'une modélisation préalable et adéquate des prises de décisions opérationnelles limite grandement les inconvénients aussi bien des décisions humaines que des décisions automatisées, alors qu'elles sont pourtant différentes. En fait, la modélisation d'une prise de décision humaine est souvent le point de départ pour réussir son automatisation.

4 Etat de l'Art

4.1 Méthodes et cadres de représentation pour l'Architecture d'Entreprise

4.1.1 Périmètre de l'état de l'art

Au moins 80 cadres d'architecture ont été recensés à travers le monde [Pragmatic EA, 2013] (Category Framework:Architecture). Plus d'une vingtaine sont représentés sur la Figure 14 plus loin. Certains ne comptent peut-être qu'un seul utilisateur, leur concepteur uniquement, tandis que d'autres comptent plusieurs dizaines de milliers d'utilisateurs (par exemple, The Open Group qui publie le cadre TOGAF® indique en temps réel le nombre de personnes certifiées : 72.454 au 06/10/2017).

La sélection d'une dizaine de méthodes et cadres de représentation seulement, présentés dans un ordre plus ou moins chronologique, conduit à un état de l'art partiel (et forcément partial), mais que nous trouvons représentatif. Le nombre d'utilisateurs, difficile à mesurer, bien qu'évoqué dans le paragraphe préliminaire, n'est pas le critère principal de sélection. Des raisons historiques ont rendu certaines méthodes et certains cadres emblématiques. Ils figurent normalement dans cet état de l'art.

La diffusion libre et la standardisation, bien que ces deux notions ne soient pas toujours compatibles, font partie de nos critères de sélection. Il nous faut reconnaitre que le pays d'origine d'une méthode ou d'un cadre a une influence importante. Ainsi, les méthodes et cadres d'origine française sont bien représentés.

Bien entendu, l'étude des ouvrages spécialisés sur ce sujet nous a permis de dresser un panorama certes non exhaustif, mais cependant fidèle des méthodes et des cadres.

Deux approches radicalement différentes se dégagent des méthodes et cadres : ceux focalisés sur l'Entreprise et ceux focalisés sur l'Architecture. On peut supposer que les seconds pourraient s'appliquer à des systèmes autres que celui du système Entreprise.

4.1.2 Zachman Framework

Le cadre de John Zachman [Zachman, 2011], qui a le grand mérite d'avoir été le premier (1987), pose le problème de l'Architecture d'Entreprise avec une série de 6 questions : Quoi ? Comment ? Où ? Qui ? Quand ? Pourquoi ? Puis il faut y répondre selon 6 perspectives différentes. Le cadre de Zachman se définit lui-même comme une ontologie d'Entreprise, ce qui correspond bien à la définition suivante : « Ensemble structuré de concepts, organisés dans un graphe et liés par des relations sémantiques et logiques, destiné à modéliser un ensemble de connaissances dans un domaine donné ».

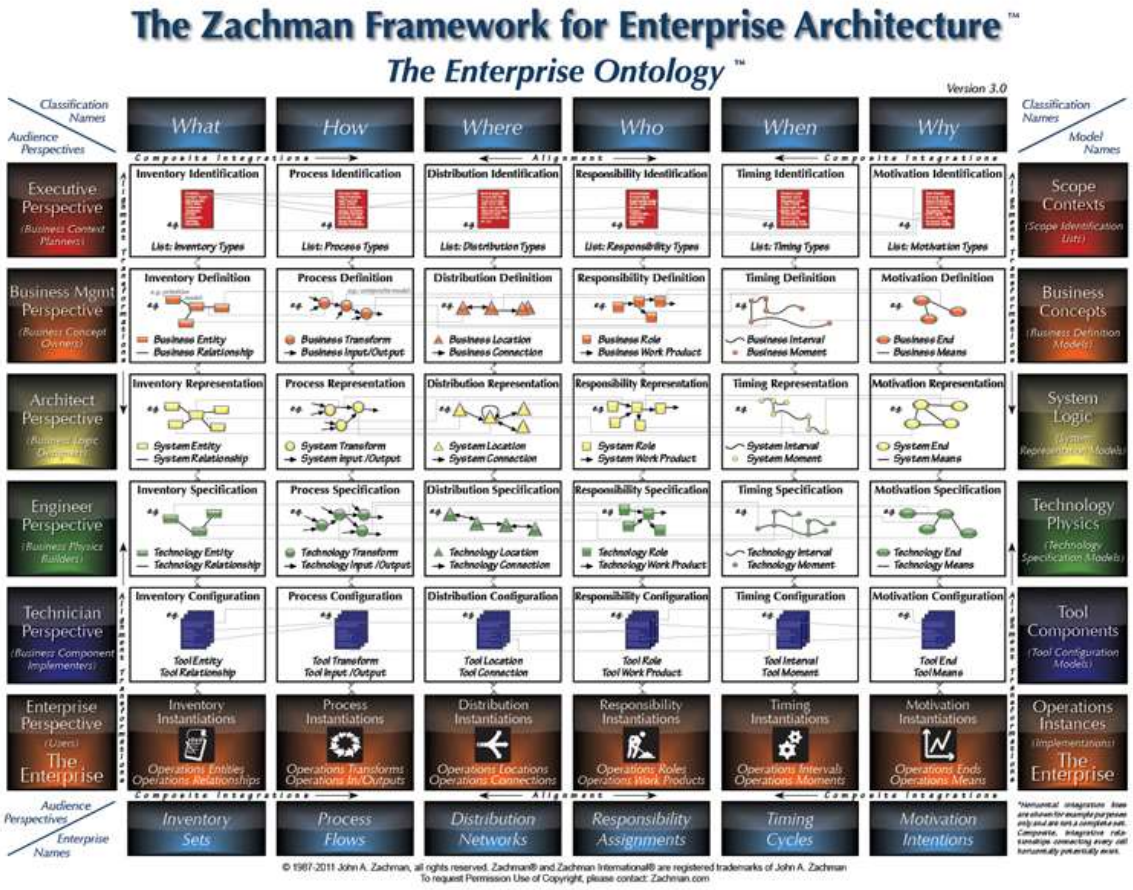


Figure 6 : The Zachman Framework for Enterprise Architecture

Sa représentation matricielle donne l'illusion d'une approche cartésienne et rassurante. Mais finalement, il ne propose donc pas moins de 36 représentations différentes (version 2011) de l'Entreprise, ce qui semble un nombre trop élevé. Avec ses 36 représentations, il se révèle inadapté à la réalité du terrain et lourd à mettre en œuvre.

En outre, tandis que la séparation des données (Quoi) et des fonctions (Comment) est adaptée au monde de l'Entreprise, elle n'est pas adaptée à la Programmation Orientée Objet, pour le développement informatique des applications avec un langage de programmation tel que Java (Attribut et Méthodes sont les termes utilisés).

Le cadre de Zachman est uniquement… un cadre. Ce n'est ni une méthode ni un processus.

4.1.3 CIMOSA

Open System Architecture for Computer Integrated Manufacturing
[ESPRIT Consortium AMICE, 1993]

Projet européen (9 pays différents) regroupant une trentaine d'organisations du monde industriel et du monde académique, CIMOSA est basée sur le concept de cycle de vie du système et propose une méthodologie et un langage de modélisation.

N.B. A l'époque de CIMOSA, on ne parlait pas encore d'Entreprise Architecture (EA), mais d'Enterprise Integration et d'Enterprise Modelling (EM).

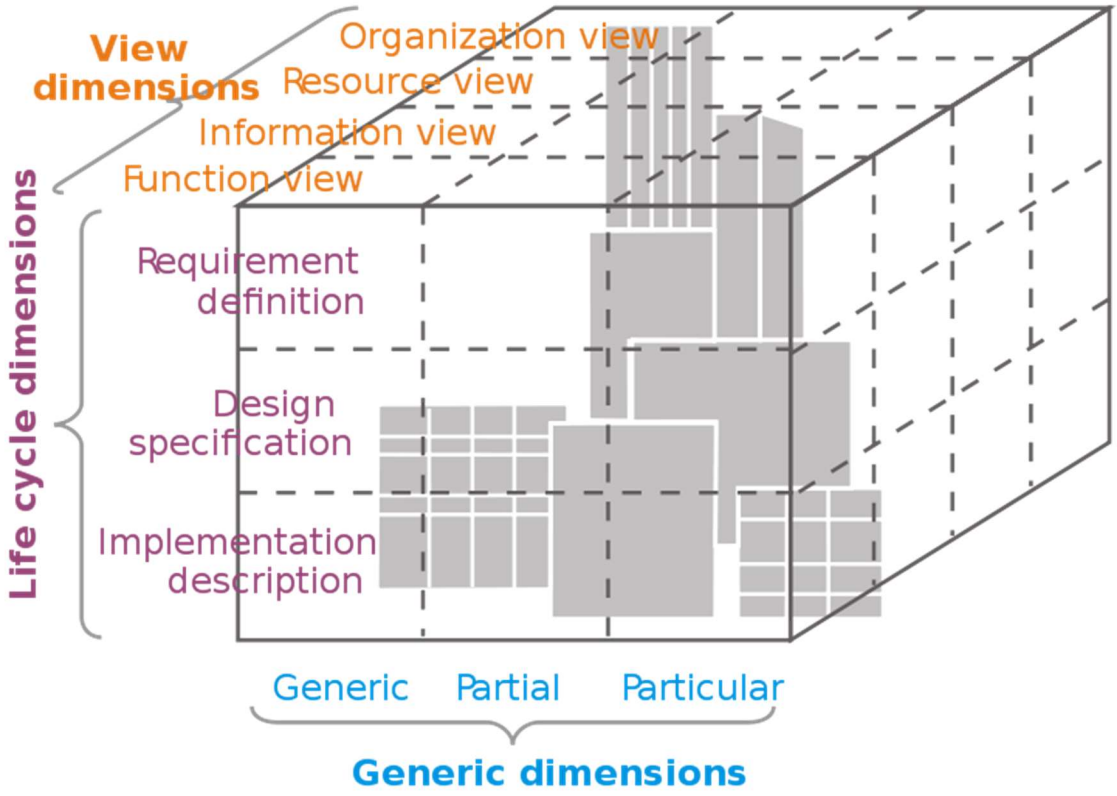


Figure 7 : Cube CIMOSA

La notion de Vue fait son apparition (versus Perspective dans le cadre Zachman, Point de vue dans la cadre TOGAF et Aspect dans la méthode Praxeme). Le découpage ou plutôt la séparation dans cette dimension Vue ne seront pas reprises telles quelles dans les méthodes et cadres suivants. Par contre, le principe de la séparation (des préoccupations en fait) lui-même sera repris dans la plupart de ces méthodes et cadres.

4.1.4 GERAM

Generalized Enterprise Reference Architecture and Methodology [Bernus et Nemes, 1996]

GERAM développe les concepts proposés précédemment par CIMOSA. Ainsi, le cycle de vie du système y est plus détaillé. La GERA (la partie Architecture uniquement de GERAM) définit une structure tridimensionnelle (Figure 8) avec des 7 phases de cycle de vie, 3 niveaux de généricité (caractère de ce qui est générique) et 4 vues de modèles.

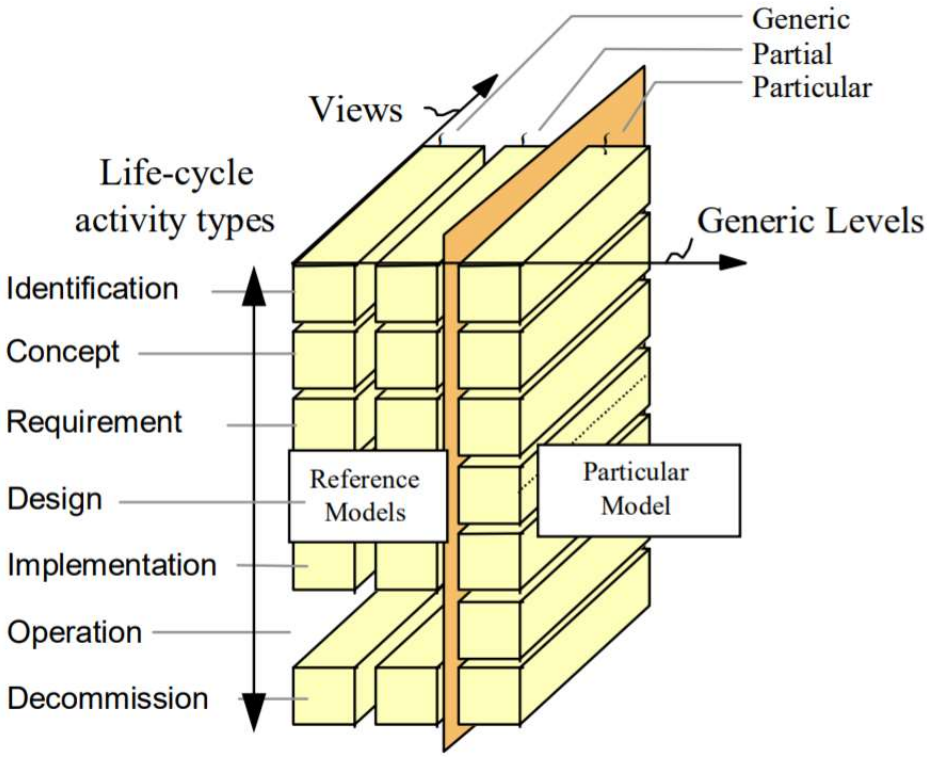


Figure 8 : Structure tridimensionnelle de la GERA

La couverture de GERAM est complète [Lillehagen et Krogstie, 2010], avec de l'Architecture et de la Méthodologie donc, mais aussi des langages, des modèles et des outils (Figure 9).

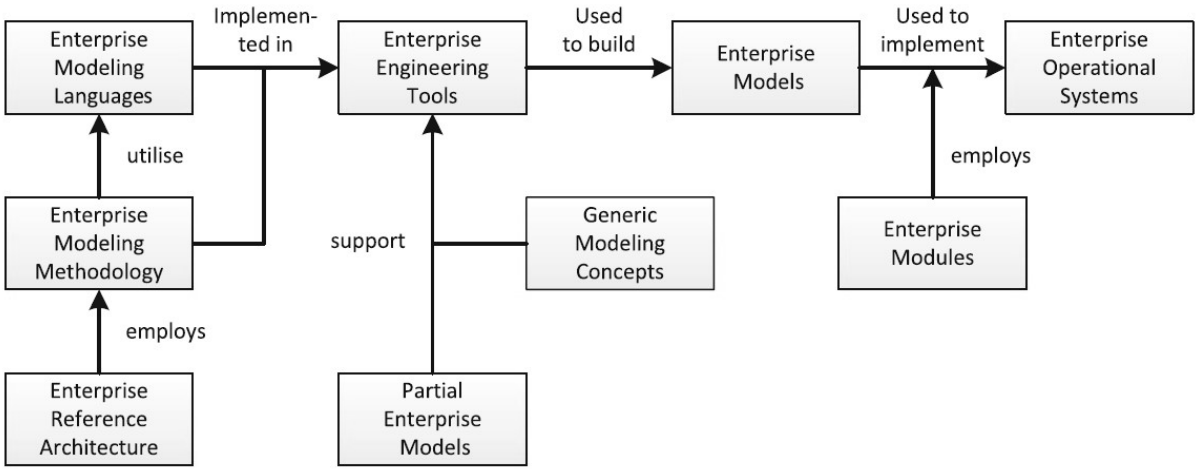


Figure 9 : Aperçu des composants de GERAM

4.1.5 Standards ISO pour l'architecture et la modélisation

4.1.5.1 ISO 14258-15704-19439-19440

Plusieurs groupes de travail (CIMOSA, GERAM et GRAI présentée plus loin, dans le chapitre Modélisation des décisions) ont apporté leurs contributions aux standards de l'ISO, notamment dans les domaines « Industrial automation systems » et « Enterprise integration ». Il est bien difficile de mesurer l'influence de ces standards ISO dans l'industrie, notamment chez les éditeurs d'outils de gestion de l'Architecture d'Entreprise (EAM – Enterprise Architecture Management) et de modélisation, car ils ne sont jamais cités.

La numérotation des standards ISO n'est pas intuitive (sauf deux numéros de standards qui se suivent et qui forment une paire) et une carte comme celle de la Figure 10 est nécessaire pour s'y retrouver :

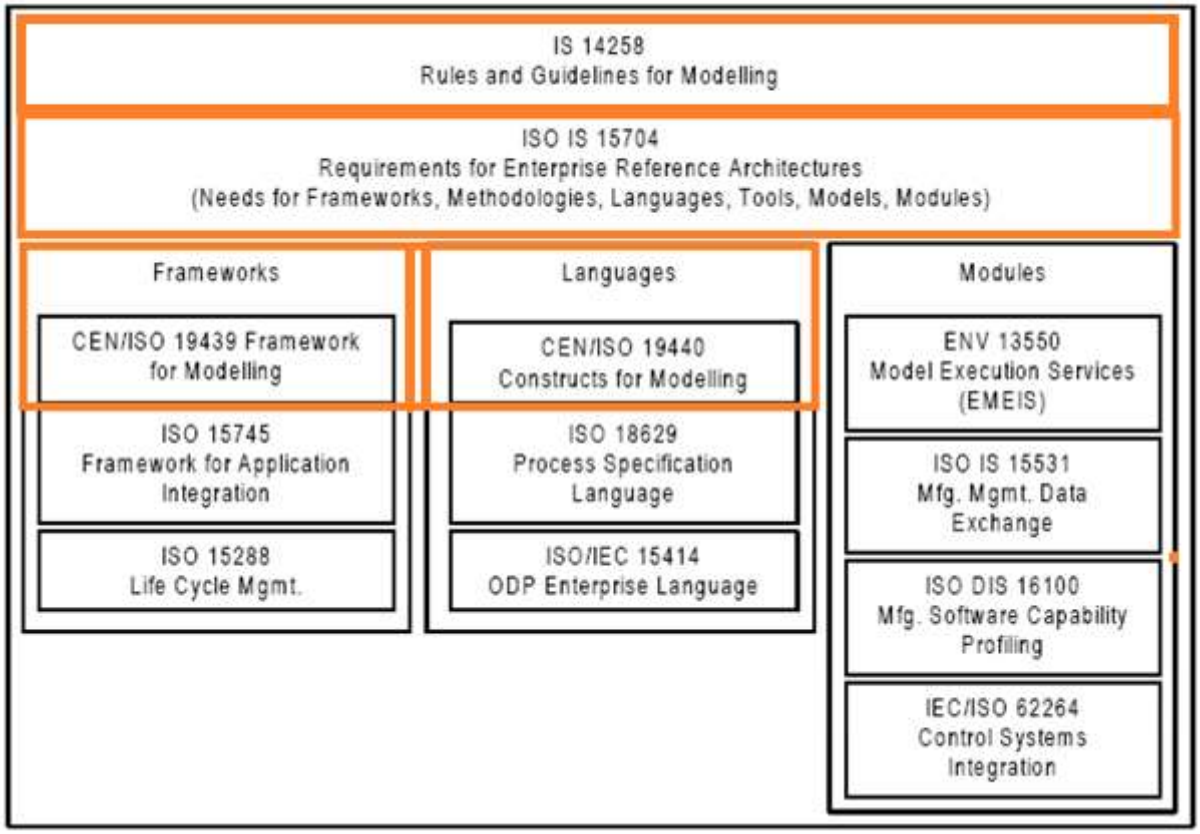


Figure 10 : Carte des standards ISO concernant Architecture d'Entreprise et Modélisation

N.B. Les noms des standards ISO ont légèrement évolué avec le temps, avec l'ajout du mot « enterprise » notamment.

ISO 14258:1998 Concepts and rules for enterprise models (15 pages)
qui pose les bases pour guider et contraindre les standards ISO suivants sur le même sujet.

ISO 15704:2000 Requirements for enterprise-reference architectures and methodologies (43 pages) qui est directement dérivé de GERAM.

ISO 19439:2006 Framework for enterprise modelling (34 pages)
qui répond aux exigences (*requirements*) spécifiées dans le standard ISO 15704 précédent, en proposant le cadre du GERA. La Figure 11 montre toutefois mieux la troisième dimension avec ses 4 vues par fonction, information, ressource et organisation.

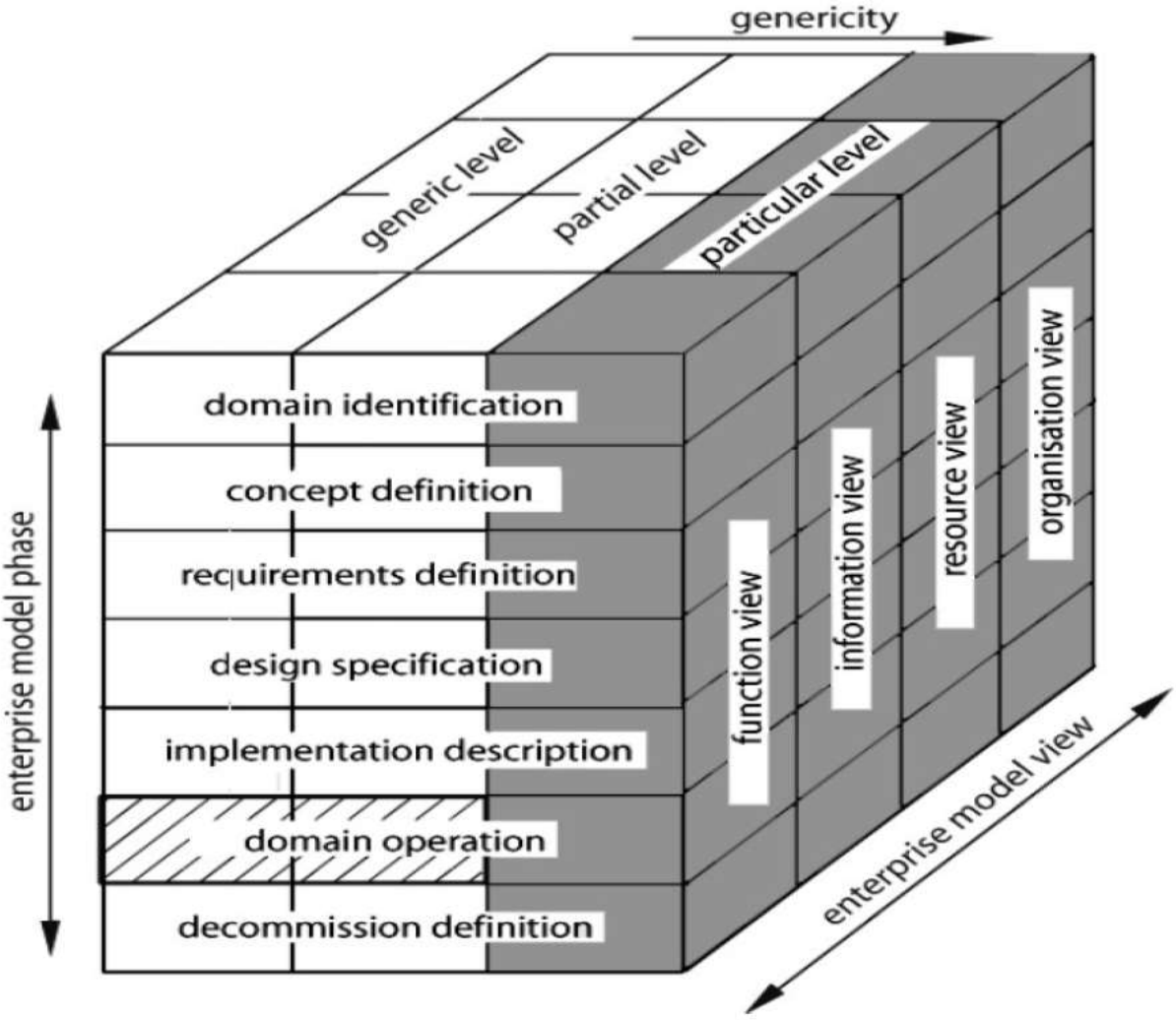


Figure 11 : Structure tridimensionnelle du standard ISO 19439

ISO 19440:2007 Constructs for enterprise modelling (111 pages)
qui définit et décrit les concepts (*constructs*) nécessaires à la modélisation d'entreprise, tout en supportant la structure définie dans le standard ISO 19439 précédent. La liste de ces concepts contient ceux d'UEML (décrit chapitre suivant), plus quelques autres comme *Domain* et *Capability*.

N.B. Chaque numéro de standard ISO est suivi de l'année de sa dernière version. On remarque ainsi que l'élaboration de ces différents standards a été faite dans un ordre chronologique de haut en bas (*top-down*), qui respecte la structure arborescente de la Figure 10.

A noter qu'il s'agit de documents conceptuels assez concis (quelques dizaines de pages seulement), sauf le dernier concernant le standard ISO 19440:2007, qui décrit une douzaine de concepts qui constituent les principaux objets de l'Architecture d'Entreprise.

4.1.5.2 ISO 42010

Il ne s'agit ici de détailler tous les standards ISO concernant l'architecture d'entreprise et la modélisation, car ils sont nombreux (et accessoirement payants, ce qui limite leur diffusion). Mais un autre standard dans le domaine du logiciel est venu enrichir le fonds de l'organisation ISO et mérite sa place dans cet état de l'art :

ISO 42010:2011 Systems and software engineering -- Architecture description

Ce standard concerne la création, l'analyse et la maintenance de l'architecture des systèmes, grâce à l'utilisation de descriptions. En plus d'établir un modèle conceptuel de l'architecture, il spécifie les besoins en points de vue, cadres d'architecture et langages de description.

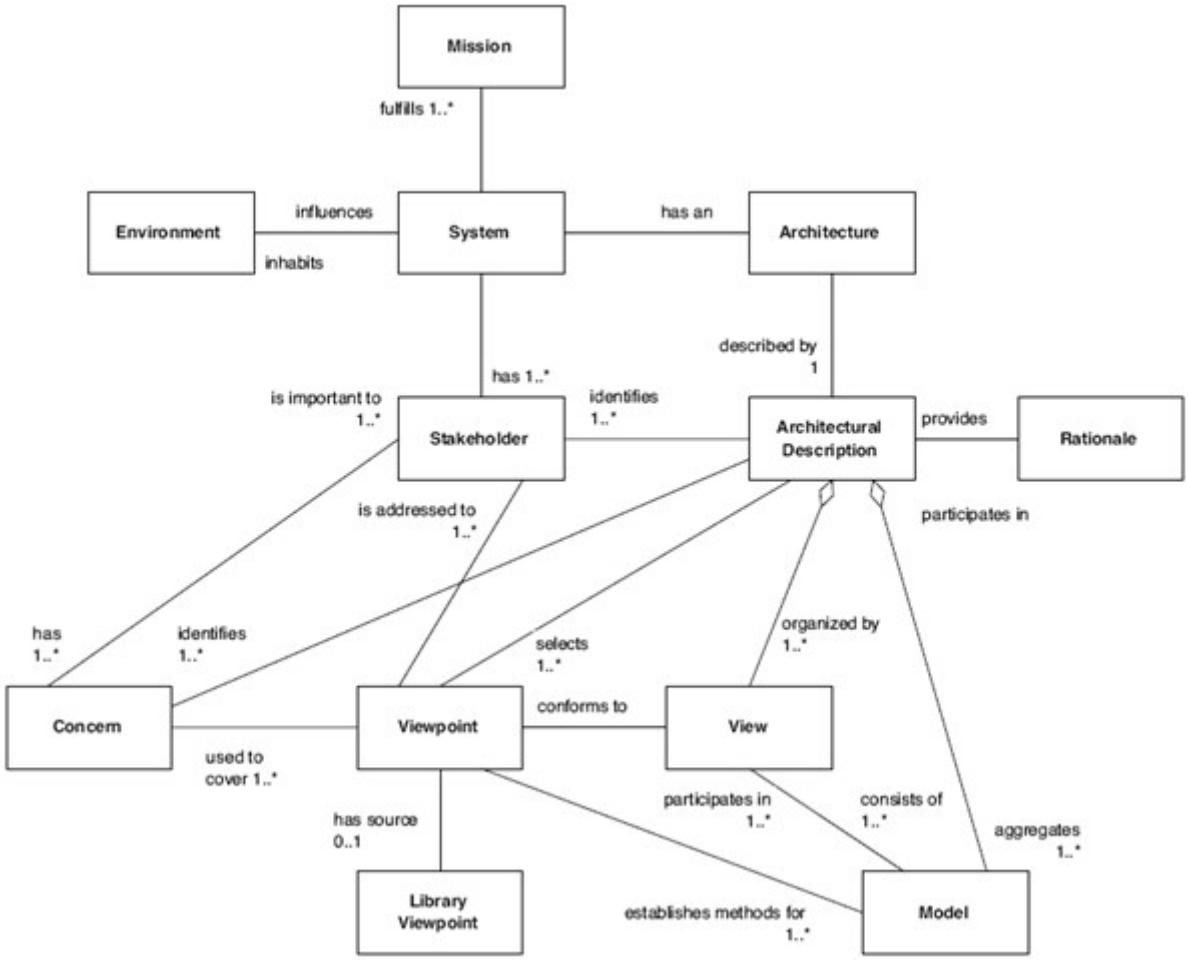


Figure 12 : Cadre conceptuel de description d'architecture ISO 42010

Le diagramme de classes en UML en Figure 12 ne représente pas ici les objets métier de l'entreprise, mais les objets d'architecture, comme la description architecturale (au milieu, à gauche) qui entre autres agrège des modèles et des vues (en bas, à droite).

Le standard ISO 42010 n'impose aucun formalisme pour la représentation des modèles : c'est à l'architecte de sélectionner le format adéquat de ses modèles en fonction du contexte, selon les préoccupations (*Concern*) des parties prenantes (*Stakeholder*) notamment.

4.1.6 UEML

Unified Enterprise Modelling Language [Vernadat, 2001]

Comme sa signification l'indique, le but d'UEML est de proposer un langage de modélisation d'entreprise unifié et tenter de mettre un terme à la situation qualifiée par son auteur de « Tour de Babel » (sic) : la multiplicité des langages de modélisation d'entreprise, qu'il faut apprendre pour comprendre et autant d'outils différents.

UEML prend en compte les 4 principes suivants :

- Langage défini précisément (cf. chapitre Langage versus Notation plus loin),
- Séparation des processus et des ressources,
- Séparation de la fonction et du comportement de l'Entreprise,
- Séparation des ressources et des unités organisationnelles.

Ces différents principes de séparation sont mis en application dans la Figure 13, qui représente les concepts au cœur d'UEML avec leurs relations et qui constitue finalement le métamodèle.

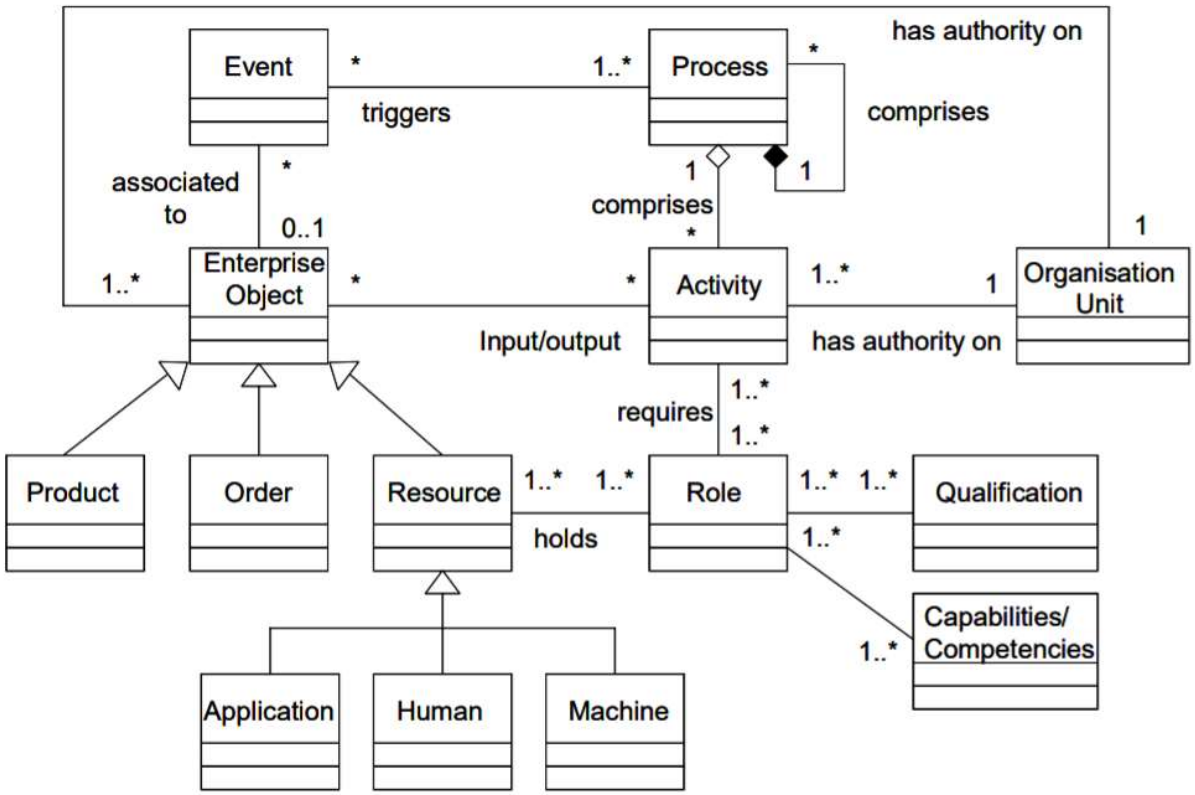


Figure 13 : Relations entre les concepts au cœur d'UEML

On voit clairement sur cette Figure 13 les concepts qui serviront de base pour la modélisation des processus métier notamment : Process, Activity, Role, Event.

Tandis que cette initiative de langage unifié est louable, il semble qu'elle n'est pas rencontrée le succès auprès des utilisateurs métier et des éditeurs d'outils, ces deux types d'acteurs étant bien entendu dépendants l'un de l'autre : peu d'utilisation sans outil (et réciproquement).

Des recommandations et des règles pour incorporer d'autres langages de modélisation dans UEML ont été proposées récemment [Harzallah et al., 2012].

4.1.7 TOGAF®

Le cadre TOGAF® - *The Open Group Architecture Framework* [The Open Group, 2011], est inspiré entre autres de l'IAF – Integrated Architecture Framework – développé par la société de service informatique Capgemini de 1996 à 2006 (ce cadre IAF propriétaire n'est pas détaillé dans ce mémoire). The Open Group est un consortium international qui regroupe plus de 500 organisations et qui permet la réalisation des objectifs métier grâce à l'utilisation de standards pour les systèmes d'information.

La Figure 14 représente chronologiquement les influences des cadres d'architecture les uns sur les autres pendant 25 ans. L'adhésion de nombreuses entreprises au cadre TOGAF® a fait que la plupart des cadres concurrents sont tombés en désuétude.

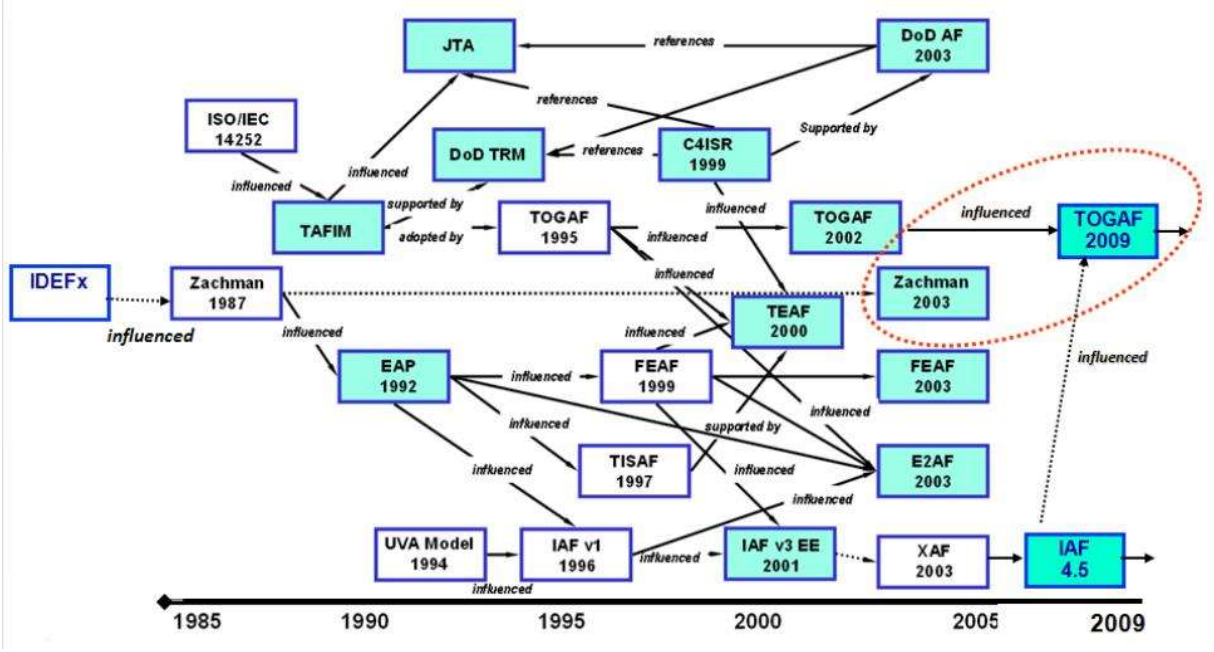


Figure 14 : Histoire et influences des cadres d'Architecture d'Entreprise

Le cadre TOGAF®, dont il faut constater la diffusion récente dans les grandes entreprises, d'abord anglo-saxonnes puis françaises, est surtout constitué d'une longue liste d'activités à réaliser par les architectes, en respectant certains principes. Cette liste est de type taxonomique : elle a pour but de classifier les différents éléments du domaine de l'architecture d'entreprise. Si un cadre de contenu détaillé est fourni, TOGAF® n'est pas d'une véritable méthode d'Architecture d'Entreprise.

Il faudrait en effet que TOGAF® se focalise plus sur l'Entreprise et propose des points de vue différents de celui des architectes. Néanmoins, TOGAF® propose quatre domaines différents d'architecture : l'architecture métier, l'architecture de données, l'architecture applicative et l'architecture technique (architecture technologique littéralement). L'architecture de données, et l'architecture applicative permettent de séparer l'architecture métier et l'architecture technique, en se présentant sous forme de services.

Ce cadre est en outre avare en recommandations concernant la modélisation, qui ne semble pas être sa préoccupation principale (sans doute pour laisser la place au langage de modélisation ArchiMate, qui sera présenté plus loin, publié par le même organisme The Open Group). En outre, TOGAF® s'intéresse encore moins à la transformation des modèles. Cette lacune est toutefois comblée par un outil comme Modelio [Desfray et Raymond, 2014].

Paradoxalement, sans doute grâce à sa forme harmonieuse et rassurante, la *Figure 15* emblématique de TOGAF® est l'ADM (Architecture Development Method), qui est en fait le processus d'Architecture en mode projet, découpé en 9 phases, dont une phase préliminaire, qu'il « suffit » d'exécuter dans l'ordre chronologique :

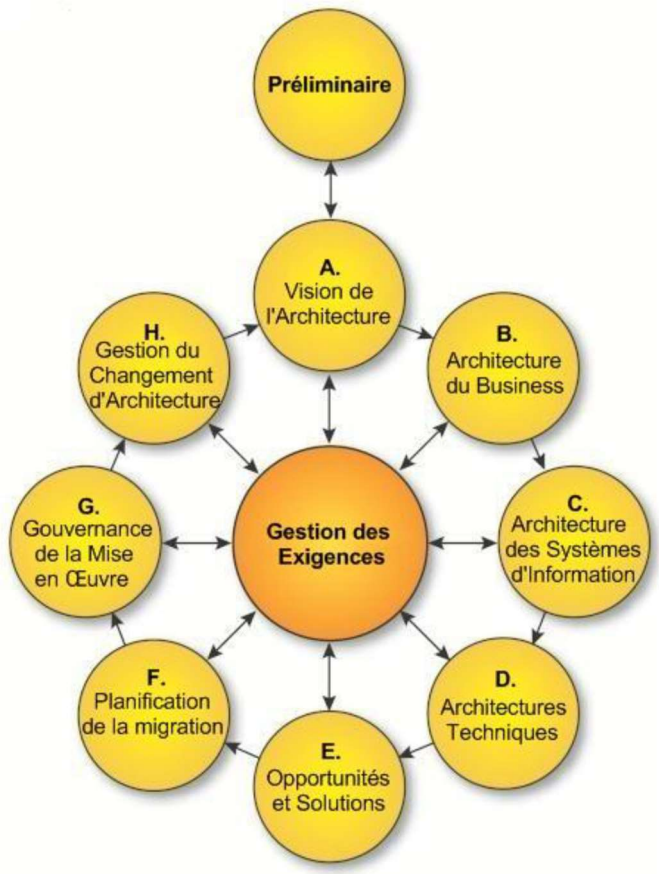


Figure 15 : Architecture Development Method (TOGAF®)

Mais le vrai cadre d'architecture de TOGAF®, destiné à recevoir les différents éléments qui représentent l'Entreprise, est l'Architecture Content Framework représenté sur la Figure 16. Il s'agit en fait d'un métamodèle, qui définit la structure de « rangement » qui va recevoir ces différents éléments d'architecture. Il va permettre de retrouver, puis de ranger, ces éléments lors de l'exécution des phases de l'ADM.

Les modèles, représentés le plus souvent par des diagrammes, sont des éléments d'architecture (des artéfacts selon TOGAF®). Ces modèles rentreront dans la composition de livrables plus complets, comme le document de définition de l'architecture.

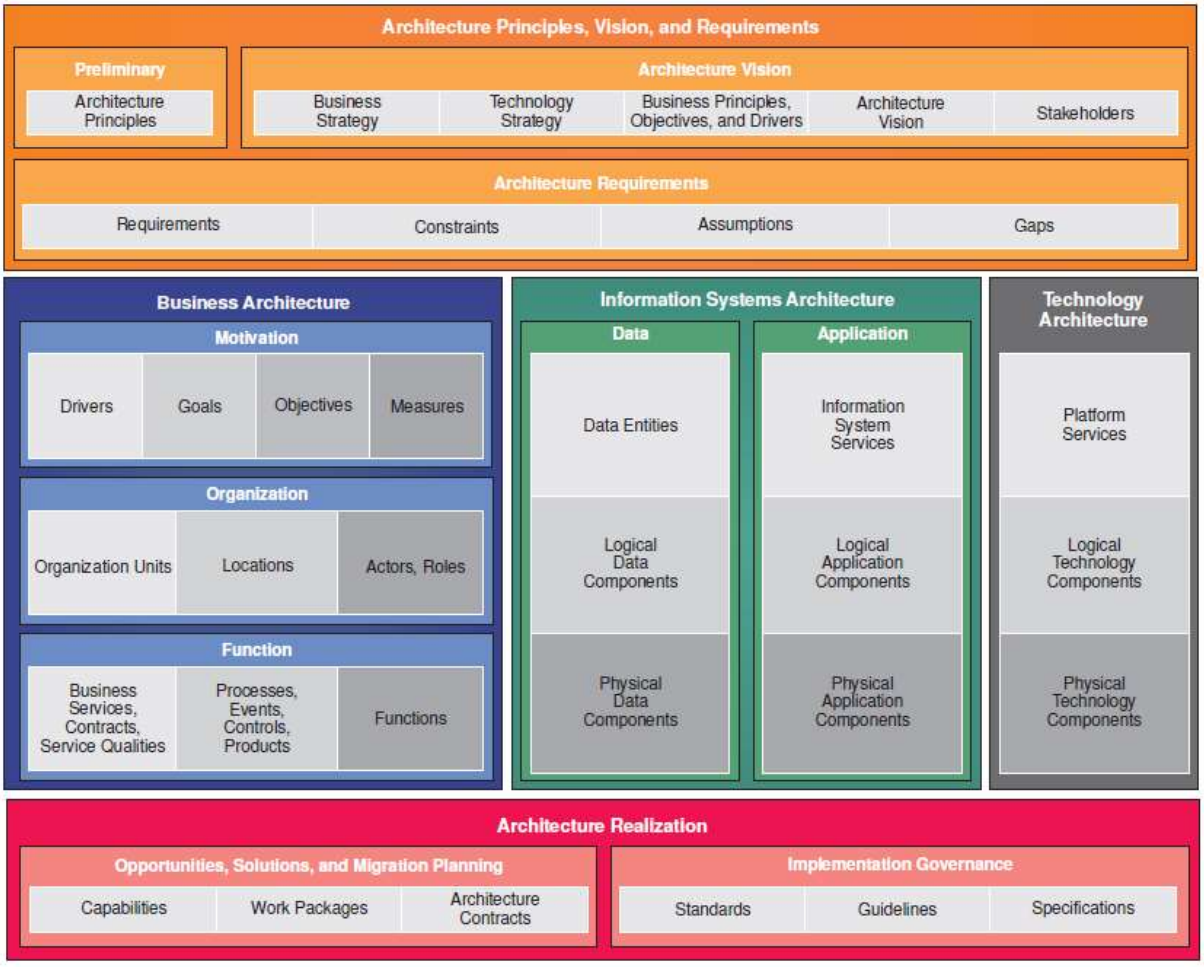


Figure 16 : TOGAF® Architecture Content Framework

Entre la représentation dynamique du processus (ADM) et la représentation statique du cadre d'architecture (Architecture Content Framework), la Figure 17 propose une représentation mixte dynamique statique, où des flèches symbolisent les relations entre les cases. Cette figure montre également en haut à droite quels modèles de référence et quels standards externes sont utilisés dans l'Entreprise et potentiellement réutilisables. En bas dans le bloc Gouvernance Log (de l'architecture), on évalue la capacité d'architecture, grâce à ses ressources, et l'on conserve une trace des décisions prises par le comité d'architecture (Architecture Board) à droite.

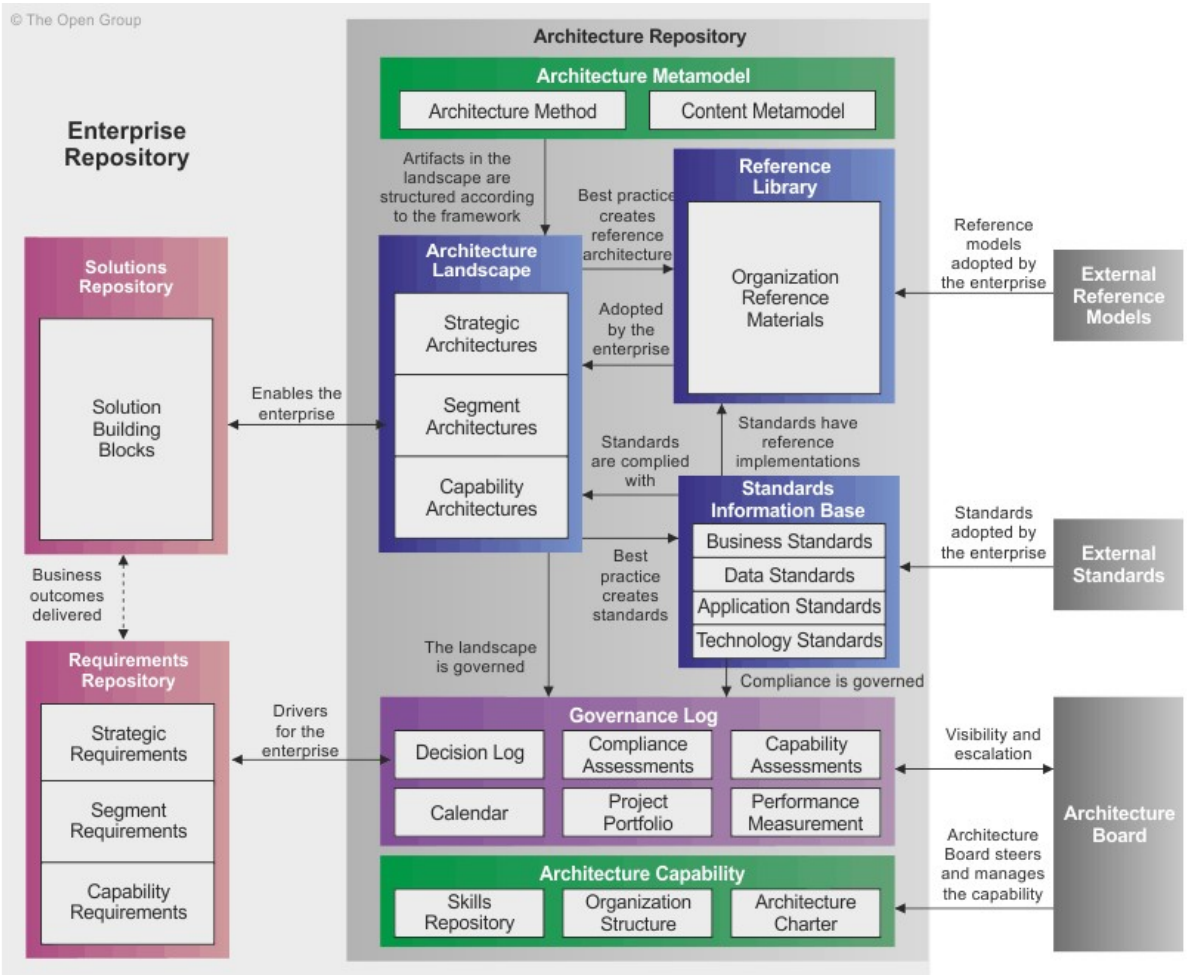


Figure 17 : Vue d'ensemble du référentiel d'architecture TOGAF®

N.B. La cadre TOGAF® couvre un périmètre très large et la documentation de sa dernière version 9.1 fait près de 700 pages. Il ne s'agissait pas dans ce chapitre d'être exhaustif ne quelques pages, mais de présenter uniquement les principaux éléments du cadre TOGAF®, surtout ceux en rapport avec les modèles qui seront présentés dans les chapitres suivants.

4.1.8 CIGREF

Pour le CIGREF (Club Informatique des GRandes Entreprises Françaises, qui regroupe une centaine d'entreprises), le Système d'Information peut être structuré en quatre strates qui sont déclinées en autant de types d'architecture, couvrant l'ensemble de l'Entreprise : Métier, Fonctionnel, Application et Technique, comme représenté sur la Figure 18 [CIGREF, 2003].

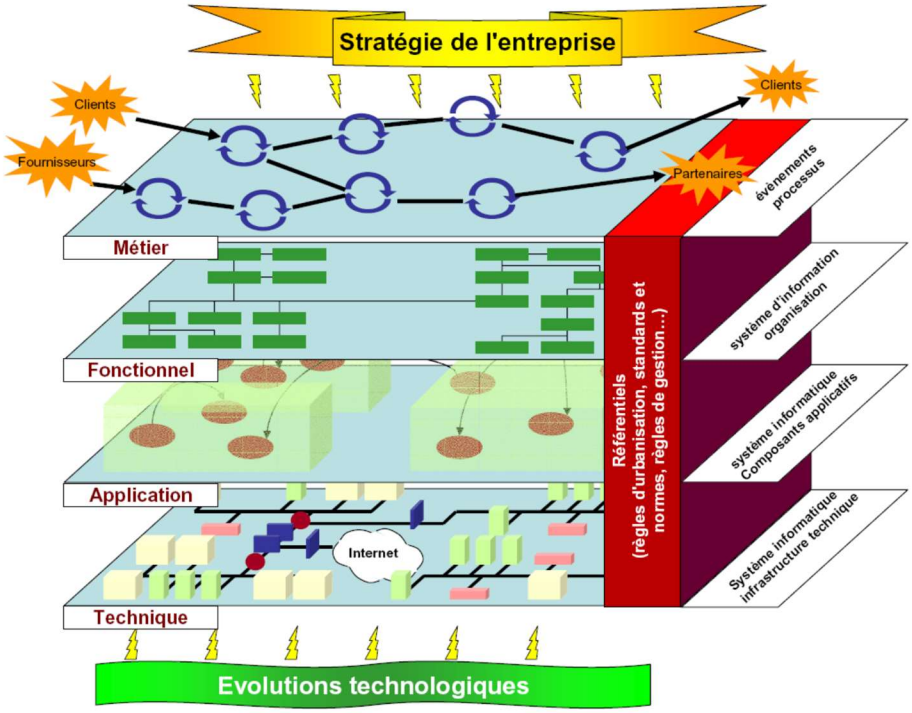


Figure 18 : Les 4 strates du Système d'Information selon le CIGREF

Ce modèle de haut niveau du CIGREF s'intéresse à l'impact des décisions stratégiques sur strate métier, qui regroupe les processus (démarche top-down), mais il n'est pas assez détaillé pour s'intéresser aux prises de décisions opérationnelles.

Ce modèle du CIGREF a servi de base à la DINSIC (Direction Interministérielle du Numérique et du Système d'Information et de Communication) pour établir, en 2012, le Cadre Commun d'Urbanisation du Système d'Information de l'Etat (français), dans lequel la stratégie est devenue une cinquième strate, appelée désormais « Vue », à part entière (et la strate Architecture Technique est devenue la Vue Infrastructure) [DGME, 2012] :

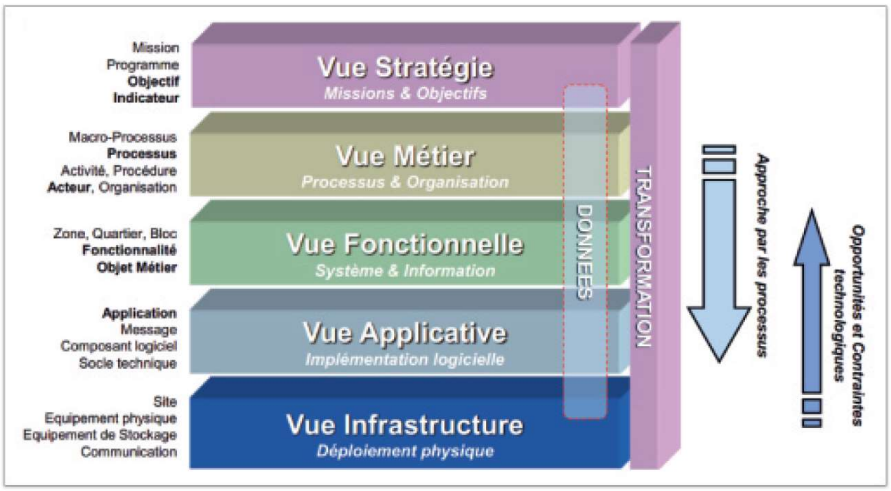


Figure 19 : Cadre Commun d'Urbanisation du Système d'Information de l'Etat

4.1.9 CHAMPS2

CHAMPS2 (CHAnge Management for the Public Sector) est une méthode de conduite du changement du métier. Elle aide à définir les besoins stratégiques (selon la vision des Produits et Services), avant de proposer un parcours de transformation sur mesure en 8 phases afin d'obtenir les résultats attendus. Chaque phase se décompose en étapes détaillées. Cette méthode fut développée pour le Conseil municipal de Birmingham (Royaume-Uni) [Birmingham City Council, 2006], mais est désormais accessible gratuitement à tous, y compris pour le secteur privé.

Figure 20 : Les 8 phases de la méthode de conduite de changement CHAMPS2

CHAMPS2, qui a une approche holistique du changement et met en évidence l'analyse d'impact, se base essentiellement sur l'alignement des processus métier, de la structure de l'organisation et de la technologie pour atteindre son objectif. CHAMPS2 aidera par exemple ses utilisateurs à concevoir les nouveaux processus métier. Cette méthode définit la bonne orientation, fournit les techniques et les outils, pour réduire les coûts et surtout d'atténuer les risques liés aux changements et permet de prendre les décisions adéquates au bon moment. Elle promeut également l'innovation.

On retrouve dans la méthode CHAMPS2 certaines caractéristiques de l'Architecture d'Entreprise. On pourrait dire que la méthode CHAMPS2 est un sous-ensemble de l'Architecture d'Entreprise spécialisé dans la transformation et la conduite du changement (tandis que l'Architecture d'Entreprise s'occupe également des « affaires courantes », en plus de la transformation).

Le déploiement de cette méthode britannique CHAMPS2 ne semble toutefois pas avoir traversé la Manche et n'est plus très active depuis 2012.

4.1.10 Praxeme

4.1.10.1 Présentation de la méthode publique

Mettant en pratique le concept de l'IDM - Ingénierie Dirigée par les Modèles - [Jézéquel et al., 2012] popularisée par la MDA – Model Driven Architecture [OMG, 2014], la méthode Praxeme [Praxeme Institute, 2006], d'origine française, propose une alternative émergente.

Selon son étymologie, Praxeme signifie « le sens de l'action ». Praxeme, tout en tenant compte des évolutions technologiques, s'inscrit depuis 2003 dans l'héritage de la méthode de référence Merise [Tardieu et al., 2000].

La méthode Praxeme est supposée complète, rigoureuse et très bien adaptée à l'Architecture Orientée Services – SOA [Caseau, 2011], bien qu'elle puisse être utilisée pour d'autres styles d'Architecture. Elle a été utilisée avec succès pour la refonte du système d'information d'un assureur [Bonnet et al., 2007], l'urbanisation de stations de contrôle de drones et la modélisation de systèmes de transport, dont la signalisation.

Praxeme est une méthode publique et ouverte, dont le développement, la promotion et la formation sont assurés, depuis 2006, par une association à but non lucratif, reconnue d'utilité publique, le Praxeme Institute, qui met son fonds documentaire librement à la disposition du public.

La période du financement public d'une méthode semble révolue. Toutefois, la Direction Générale de la Modernisation de l'État recommandait en 2009 la méthode Praxeme dans la version 1.0 de son Référentiel Général d'Interopérabilité [DGME, 2009].

Il ne s'agit pas de concurrencer les cadres de représentation précédents : la méthode Praxeme peut compléter le cadre TOGAF® notamment, pour lui donner plus de consistance, lorsqu'il s'agit de remplir ce cadre de référence (et TOFAF® n'est pas contre son utilisation conjointe avec d'autres cadres ou méthodes).

La méthode Praxeme propose une nouvelle approche de l'Architecture d'Entreprise, tout en s'appuyant sur des standards reconnus. Elle propose un cadre de représentation qui accueille déjà des diagrammes UML (Unified Modeling Language), mais également des diagrammes BPMN (Business Process Model and Notation) pour la modélisation des processus métier.

4.1.10.2 Topologie du Système Entreprise

4.1.10.2.1 Les 7 Aspects de Représentation de l'Entreprise

La méthode Praxeme s'intéresse à l'Entreprise, considérée comme un système complexe, dans sa globalité (approche holistique). La méthode Praxeme, qui considère l'Entreprise comme un tout, est basée sur sa représentation selon 7 aspects. Certains aspects peuvent être représentés par des modèles au standard UML. Une représentation structurée de ses 7 aspects est donnée dans la Topologie du Système Entreprise, fondement de la méthode Praxeme.

Les aspects intentionnel, sémantique, pragmatique et géographique constituent l'architecture métier. Les aspects logique, logistique et physique constituent l'architecture technique. En fait, l'aspect logique a une place à part, car il permet de découpler l'architecture métier de l'architecture technique (de la rendre indépendante d'un changement de technologie par exemple). Ce découplage a du sens dans l'Architecture Orientée Services également.

Le regroupement des différents aspects de la Topologie du Système Entreprise de Praxeme sont similaires aux « niveaux d'abstraction » de Merise (conceptuel, logique, physique) et conformes au principe de séparation des préoccupations préconisé dans la MDA – *Model Driven Architecture*.

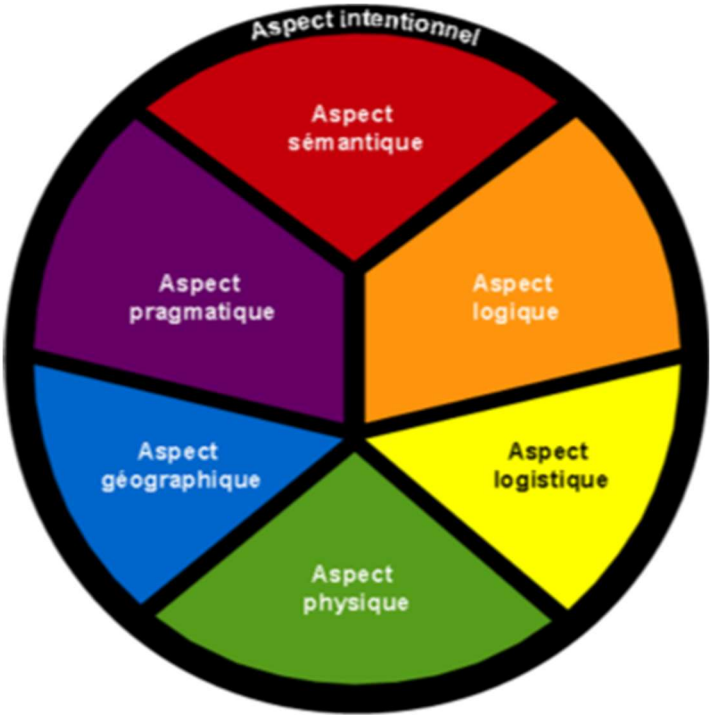


Figure 21 : Topologie du Système Entreprise Praxeme avec ses 7 aspects

Ces 7 aspects de la méthode sont conformes au métamodèle Praxeme et semblent bien articulés les uns avec les autres. Des règles précises de transformation (dérivation, selon le thésaurus Praxeme) entre ses différents aspects apportent à la méthode Praxeme de la rigueur dans la traçabilité.

Dans le cadre de ce travail de recherche doctoral, la démarche d'architecture d'entreprise, effectuée pour modéliser puis automatiser les prises de décisions opérationnelles, a été de parcourir les différents aspects – traverser les différentes couches – de la Topologie du Système Entreprise de la méthode Praxeme, sans pouvoir les explorer entièrement, à la manière d'un carottage.

Figure 22 : Principe du carottage à travers les aspects de la TSE Praxeme

4.1.10.2.2 Les 4 Aspects de l'Architecture Métier

Les aspects Intentionnel, Sémantique, Pragmatique et Géographique constituent l'Architecture Métier de l'Entreprise. Tout commence par l'omniprésent aspect Intentionnel. Puis, les aspects Sémantique et Pragmatique servent de base aux autres aspects et permettent de modéliser des systèmes sociotechniques complexes.

L'aspect Intentionnel formalisera la finalité, les intentions, la stratégie et les objectifs de l'Entreprise, voire son éthique et ses valeurs. Les indicateurs clefs sont également regroupés dans cet aspect ; il est alors possible de les organiser en un ensemble cohérent, appelé Arbre de Performance® [Garibian, 1985].

L'aspect Sémantique est basé sur la modélisation des objets métier. Il convient souvent d'extraire la sémantique des processus métier, décrits dans l'aspect Pragmatique. Le paradoxe de la modélisation des objets métier, souvent appelés Fondamentaux ou Cœur du métier, est le suivant : c'est le travail le plus difficile à faire, mais c'est aussi le travail le plus durable, car indépendant de l'organisation.

L'aspect Pragmatique est basé sur la modélisation des processus métier et l'organisation des acteurs nécessaires pour leur exécution.

La séparation des aspects Sémantique et Pragmatique est la spécificité essentielle de la méthode Praxeme. Cette séparation originelle, qui peut se révéler complexe à effectuer, est un gage de flexibilité pour faciliter les transformations à venir. Il est évident que ces deux aspects sont indépendants de toutes technologies - dont les cycles de vie sont souvent bien plus courts - qui seront utilisées pour les mettre en œuvre.

Dans ces deux aspects Sémantique et Pragmatique, figurent les définitions des Règles Métier qui formalisent les différentes contraintes, exigences, mais aussi la connaissance et le savoir-faire de l'Entreprise. Si nécessaire, l'organisation pourra être modifiée, les processus métier reconfigurés et l'Entreprise transformée.

La méthode Praxeme profite des diagrammes UML pour représenter l'aspect Pragmatique et supporter l'aspect Sémantique de l'Entreprise. Un diagramme de classes UML ne représente que les relations entre les concepts : ce sont surtout les noms des classes, leurs attributs et leurs associations qui portent la sémantique.

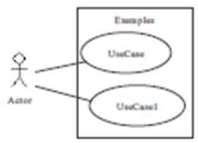
Aspect	Portée locale (programme, solution spécifique)		
	Qui	Pourquoi	Comment
Sémantique	Utilisateur	Exprimer la connaissance, les représentations	diagramme de classes  diagramme d'états-transitions 
Pragmatique	Utilisateur	Montrer comment mener l'activité	diagramme des cas d'utilisation 

Figure 23 : L'apport d'UML pour représenter les aspects Sémantique et Pragmatique

L'aspect Géographique est positionné du côté de l'Architecture métier, car il dépend bien souvent de l'histoire, la culture et l'organisation de l'Entreprise. Il s'agit d'envisager toutes les options de déploiement, puis de définir la localisation des activités (les sites), selon la mobilité du personnel et les moyens de communication.

4.1.10.2.3 Les 3 Aspects de l'Architecture Technique

Les aspects Logique, Logistique, et Physique constituent l'Architecture Technique de l'Entreprise.

L'aspect Logique a le rôle « pivot » le plus important : il est l'intermédiaire entre l'Architecture Métier (ses aspects Sémantique et Pragmatique) et l'Architecture Technique ; il assure le découplage entre la solution conçue et les technologies utilisées pour la mettre en œuvre (déployées dans les aspects Logistique - logiciel & matériel informatique - et Physique), ce qui garantit sa pérennité. La discipline de l'urbanisation devra être appliquée dans l'aspect Logique, pour structurer le Système d'Information et le préparer à l'Architecture Orientée Services (SOA).

Les aspects Sémantique et Pragmatique doivent être transformés en aspect Logique selon des règles précises, en respectant la conformité avec le métamodèle de Praxeme. L'aspect Logique dépend donc des aspects Sémantique et Pragmatique. La Figure 24 représente le principe de transformation entre ces 3 aspects et les dépendances entre tous les aspects.

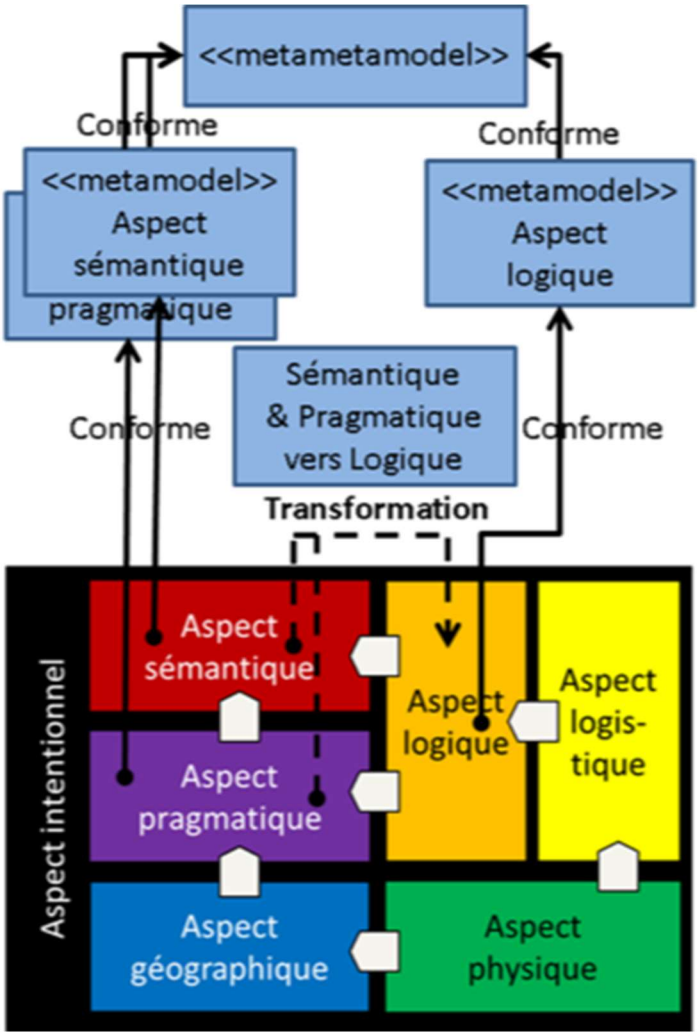


Figure 24 : Transformation vers l'aspect Logique et dépendances entre les aspects Praxeme

L'aspect Logistique décrit les solutions techniques logicielles et matérielles (technologies, composants, équipements). Grâce au découplage apporté par l'aspect Logique, les choix techniques de l'aspect Logistique peuvent être substituables.

L'aspect Physique décrit avec précision la réalité du système défini dans l'aspect Logistique puis déployé (instanciation) selon l'aspect Géographique (localisation).

4.2 Modélisation d'Entreprise

4.2.1 Différents Besoins de Modélisation d'Entreprise

Idéalement, chaque aspect de l'Architecture d'Entreprise devrait être représenté par un modèle, de préférence graphique. C'est le langage UML, un standard de l'OMG (Object Management Group), qui a apporté la plus forte contribution à la modélisation d'entreprise.

Parmi la quinzaine de diagrammes proposés par la version 2.5 d'UML, citons les plus utilisés pour l'architecture métier, selon la méthode Praxeme : le diagramme de Cas d'Utilisation pour l'aspect Pragmatique, le diagramme de Classes et de diagramme d'Etats-Transitions pour l'aspect Sémantique. D'autres diagrammes UML sont également utilisés pour l'architecture technique, notamment pour le génie logiciel et l'infrastructure.

Il convient de choisir le niveau d'analyse adéquat du Système Entreprise. Le Tableau 4, inspiré de [Brandenburg et Wojtyna, 2009], propose plusieurs niveaux d'analyse. Dans cette thèse, nous nous intéressons surtout à la modélisation des processus métier et, après avoir séparé les préoccupations, de la modélisation des prises de décisions opérationnelles et des règles métier.

Contrairement à ce que l'on pourrait croire, il convient de ne pas se focaliser sur le Système Entreprise tout seul (approche traditionnelle à éviter), mais prendre en compte son environnement. Ainsi, il est clairement établi que la modélisation des processus métier apporte une plus grande valeur ajoutée si l'on représente les interactions entre l'Entreprise et les acteurs externes (« Ce qui est dehors »), comme les Clients, les Fournisseurs et l'Administration notamment.

	Ce qui est dehors = ENVIRONNEMENT	**Objet de l'analyse = SYSTEME**	**Ce qui est dedans = SOUS-SYSTEME**
Niveau 1	Les clients, les fournisseurs	L'entreprise	L'usine
Niveau 2	L'entreprise	L'usine	L'atelier
Niveau 3	L'usine	L'atelier	Le poste de travail
Niveau 4	L'atelier	Le poste de travail	Les outils de l'opérateur

Tableau 4 : Différents niveaux d'analyse pour la modélisation

4.2.2 Modélisation des processus métier

4.2.2.1 Représentation des macro-processus

Il est important d'avoir une vision globale des principaux processus métier qui sont appliqués dans une entreprise, ce que l'on appelle généralement la cartographie des processus. Chaque processus n'est pas détaillé : il est vu comme un seul élément au niveau d'abstraction macro. Bien qu'il n'y ait pas de notations standards pour représenter ces macro-processus, ceux-ci sont souvent représentés sous forme de chevrons plus ou moins imbriqués (selon leurs interactions) dans un ordre chronologique.

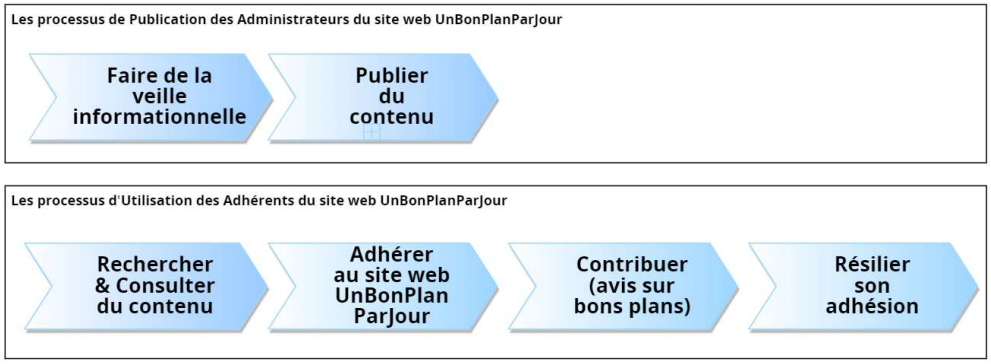


Figure 25 : Cartographie des processus (représentation informelle des macro-processus)

L'éditeur Signavio confirme que leur représentation ne s'appuie pas sur une spécification formelle ou des règles de syntaxe et suggère d'en profiter pour être créatif ! Mais de rappeler aussitôt que l'objectif premier de cette cartographie est la lisibilité et que les acteurs doivent retrouver rapidement « leurs » processus, dans lesquels ils sont personnellement impliqués.

Cette relative liberté dans la représentation des macro-processus ne doit pas faire oublier que leur cartographie doit permettre de vérifier leur découpage et leurs interactions.

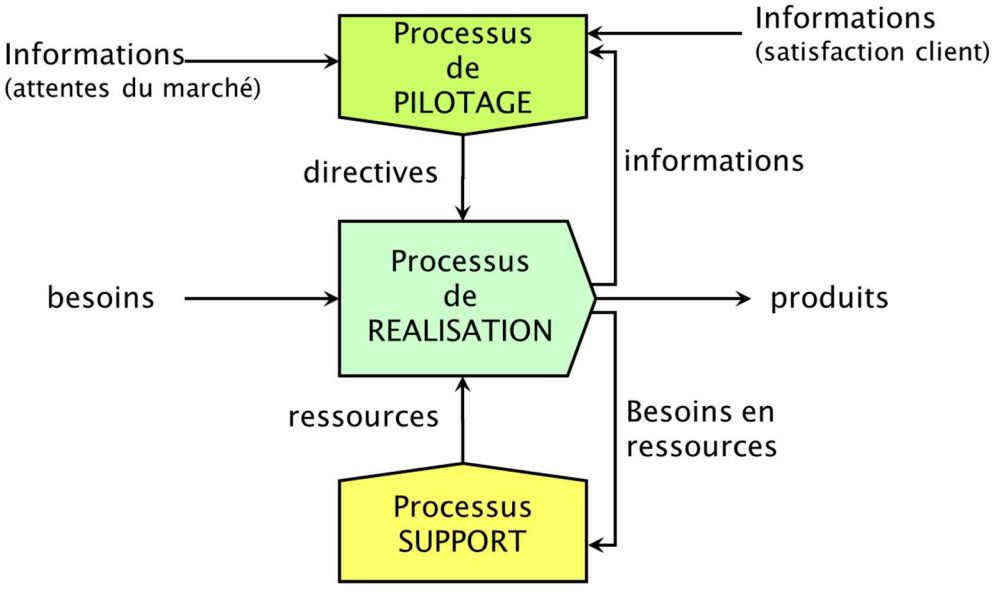


Figure 26 : Interactions entre les 3 types de macro-processus

Cette représentation est sans doute inspirée par la chaine de valeur de Porter [Porter, 2003]. L'usage est de représenter les processus de réalisation (appelés parfois processus opérationnels), qui sont au cœur du métier de l'entreprise, au milieu des processus de support en bas et des processus de pilotage en haut, comme représenté sur la Figure 26.

4.2.2.2 Diagramme d'activité UML

Avant le succès de BPMN (chapitre suivant), c'était essentiellement le diagramme d'Activités UML qui était utilisé pour modéliser les processus métier. La Figure 27 montre que la principale différence visible avec le diagramme de collaboration BPMN qui va suivre est l'élément graphique qui représente les branchements :

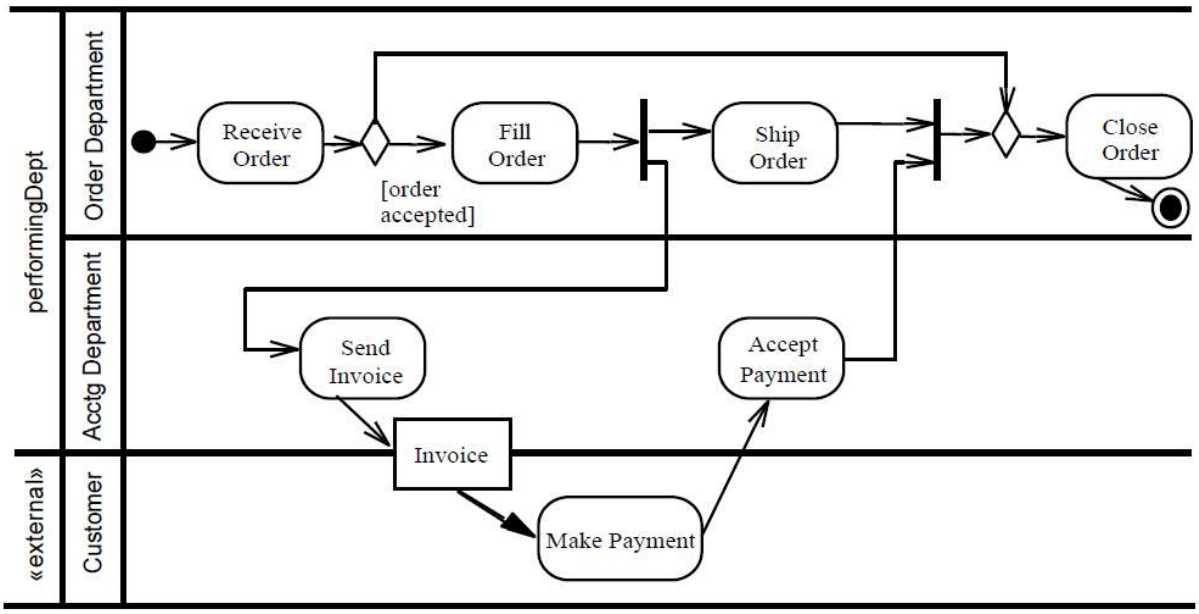


Figure 27 : Exemple de diagramme d'activité UML

Les types de branchements disponibles justement sont moins nombreux dans le langage UML que dans d'autres langages. Ainsi, le branchement de type « OU inclusif », permettant de sélectionner une ou plusieurs branches pour représenter des options par exemple, n'existe pas dans UML.

Par ailleurs, le niveau de granularité des diagrammes d'activité UML est parfois très fin et utile pour les informaticiens, mais avec certaines opérations et certains concepts trop détaillés pour répondre aux préoccupations des analystes métier.

4.2.2.3 BPMN (Business Process Model and Notation)

4.2.2.3.1 Diagramme de collaboration

La version 1.0 de la spécification BPMN (Business Process Model and Notation) a été publiée en 2003. Plusieurs versions se sont succédé ; la dernière version 2.0.2 [OMG, 2013], en plus d'être un standard de l'OMG, a également été publiée comme norme ISO/IEC 19510 en 2013.

BPMN version 2.0.2 propose 3 diagrammes pour représenter les processus métier. Le diagramme de Collaboration BPMN est celui qui a rencontré le plus de succès. Il a largement pris le dessus sur le diagramme d'activité UML (chapitre précédent), grâce à des possibilités de représentation plus larges des processus (plus de choix de branchements par exemple).

Les deux autres diagrammes de Conversation et de Chorégraphie sont peu utilisés (et ne sont pas supportés par la plupart des outils de modélisation BPMN).

4.2.2.3.2 Limitation pour la représentation des prises de décisions

Le diagramme de collaboration BPMN sert à représenter des séquences d'activités et les interactions entre les différents acteurs, représentées sous la forme de messages. Le diagramme de collaboration est supposé être suffisamment simple pour être compris par tous, y compris les parties prenantes du métier.

Pour emprunter des chemins différents selon la valeur de certains critères, c'est l'élément Branchement (*Gateway*) qui est utilisé, le plus souvent de type OU exclusif (XOR). Dès qu'il y a beaucoup de critères, les branchements en cascade nuisent à la clarté de la représentation (qui ressemble alors à un arbre de décision), comme nous le montrerons dans l'exemple qui suit.

De plus, cette représentation traditionnelle des décisions par des branchements au cœur même des processus métier constitue un couplage fort entre ces deux types d'éléments de natures différentes. L'ajout d'un critère de décision supplémentaire peut avoir un fort impact sur le modèle de processus et sur son implémentation technique, si son exécution est automatisée.

Une représentation alternative serait d'utiliser un branchement complexe, qui doit alors obligatoirement être annotée par un texte explicatif, qui est verbeux par définition et va à l'encontre du souhait initial d'obtenir une représentation graphique.

4.2.2.3.3 Exemple de Modélisation BPMN sans DMN

Au sein d'un processus plus global de recrutement dans un établissement public d'enseignement supérieur, il s'agit dans cet exemple de déterminer si un vacataire est recevable ou pas, avant de le recruter en continuant le processus. Voici l'extrait du diagramme BPMN de collaboration qui nous concerne :

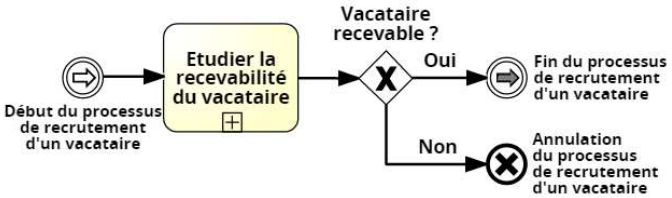


Figure 28 : Extrait du processus de recrutement (BPMN) avec un sous-processus replié

Les Articles 2 & 3 du « Décret n°87-889 du 29 octobre 1987 relatif aux conditions de recrutement et d'emploi de vacataires pour l'enseignement supérieur » indiquent précisément les critères de recevabilité d'un vacataire. Ces critères sont surlignés dans l'extrait ci-après :

Décret n°87-889 du 29 octobre 1987 relatif aux conditions de recrutement et d'emploi de vacataires pour l'enseignement supérieur.

Article 2 :

Les chargés d'enseignement vacataires sont des personnalités choisies en raison de leur compétence dans les domaines scientifique, culturel ou professionnel, qui exercent une activité professionnelle principale consistant :

-soit en la direction d'une entreprise ;

-soit en une activité salariée d'au moins neuf cents heures de travail par an ;

-soit en une activité non salariée à condition d'être assujetties à la contribution économique territoriale ou de justifier qu'elles ont retiré de l'exercice de leur profession des moyens d'existence réguliers.

Article 3 :

Les agents temporaires vacataires doivent être inscrits en vue de la préparation d'un diplôme du troisième cycle de l'enseignement supérieur.

Les personnes, âgés de moins de soixante-sept ans, bénéficiant d'une pension de retraite, d'une allocation de préretraite ou d'un congé de fin d'activité, à la condition d'avoir exercé au moment de la cessation de leurs fonctions une activité professionnelle principale extérieure à l'établissement, peuvent être recrutées en qualité d'agents temporaires vacataires dans les disciplines dont la liste est fixée par arrêté du ministre chargé de l'enseignement supérieur et lorsqu'elles n'assurent que des vacations occasionnelles dans toutes les disciplines.

Figure 29 : Articles 2 & 3 du Décret n°87-889 du 29 octobre 1987

Voici le sous-processus déplié « Etudier la recevabilité du vacataire », selon le langage BPMN, qui représente scrupuleusement ces différents critères de recevabilité. A noter que l'Article 5 de ce décret fixe un nombre maximum d'heures de vacations, dont la vérification n'est pas représentée ci-dessous.

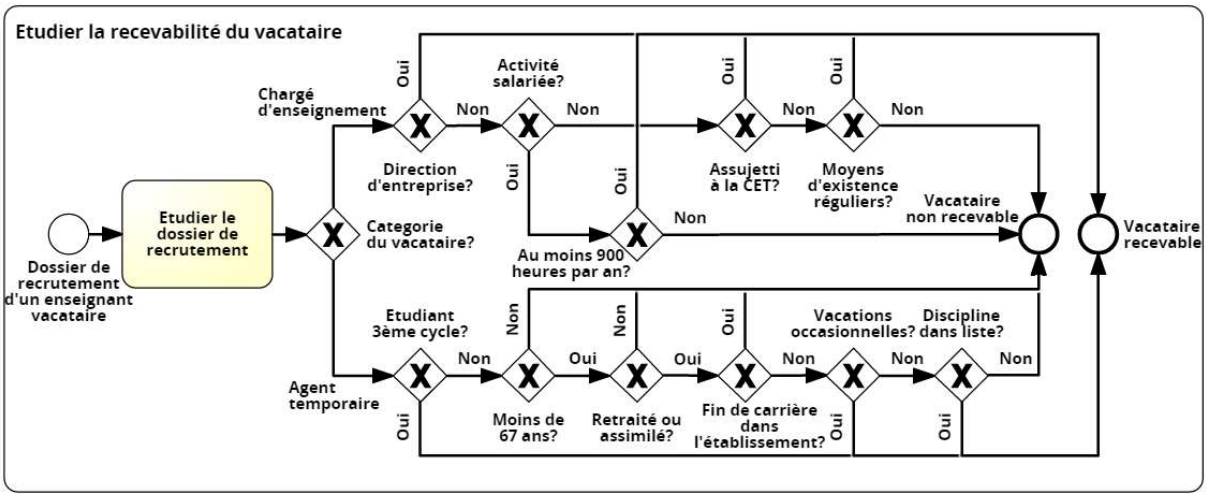


Figure 30 : Sous-processus déplié "Etudier la recevabilité du vacataire" (BPMN)

4.2.2.4 Autres notations pour représenter les processus métier

4.2.2.4.1 EPC (Event-driven Process Chain)

En plus du diagramme d'activité UML précédemment cité, EPC (Event-driven Process Chain) fait partie des notations qui ont officiellement examiné lors de la conception de BPMN, avec d'autres que nous n'avons jamais vues, utilisées en France : « Examples of other notations or methodologies that were reviewed are UML Activity Diagram, UML EDOC Business Processes, IDEF, ebXML BPSS, Activity-Decision Flow (ADF) Diagram, RosettaNet, LOVeM, and Event-Process Chains (EPCs) » [OMG, 2013].

Nous citons cette notation EPC (chaine de processus événementielle en français) pour deux raisons : elle est utilisée en France par le ministère de la Culture par exemple, car ses utilisateurs trouvent qu'elle est plus simple que BPMN ; elle est bien supportée par plusieurs outils de modélisation d'origine allemande (EPC fut inventée en 1992 par le Pr Scheer à l'Université de la Sarre, puis proposée dans son outil ARIS publié par Software AG), mais aussi dans l'outil Visio publié par Microsoft.

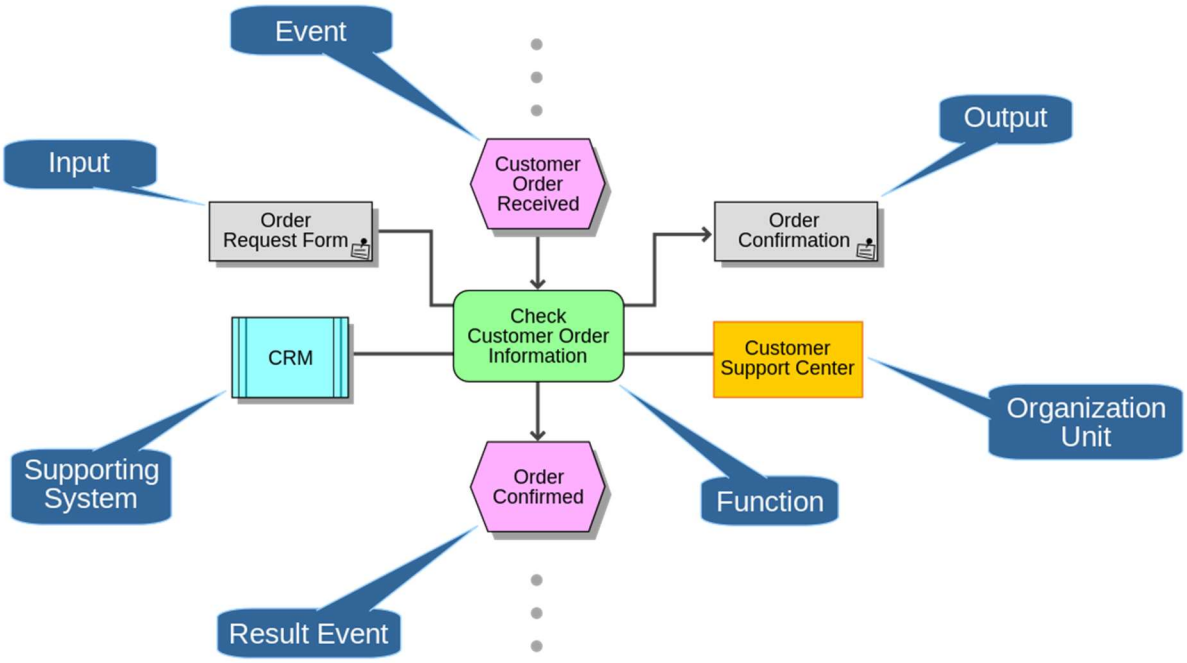


Figure 31 : Principaux éléments graphiques de l'Event-driven Process Chain (EPC)

La notation EPC a notamment été utilisée pour la modélisation des processus métier lors de la mise en place de solutions ERP (Enterprise Resource Planning), comme celle du leader du marché SAP R/3, ce qui a contribué certainement à sa diffusion.

Toutefois, EPC n'est pas un standard publié et ne semble plus mis à jour, et bien qu'il soit encore utilisé par un nombre significatif d'anciens utilisateurs regroupés en communauté [ARIS Community, 2009], il ne constitue pas un choix évident pour un nouveau projet de modélisation de processus, sans historique existant.

4.2.2.4.2 ArchiMate

ArchiMate est la notation proposée par The Open Group [The Open Group, 2017] en complément de son cadre d'architecture TOGAF®. ArchiMate fait toutefois référence à, voire s'inspire de BMM (Business Motivation Model), publiée par l'OMG, ce qui semble démontrer l'absence de cloisonnement entre ces deux organismes, grands pourvoyeurs de standards.

ArhiMate est un langage pour décrire toutes les architectures de l'Entreprise (métier, applicative, technique, …). Un processus métier est représenté dans ArchiMate avec une granularité importante. Il pourra ensuite être représenté de façon plus détaillée avec un langage spécialisée dans les processus métier, comme BPMN.

La Figure 32 représente le processus métier d'une réclamation. Le processus lui-même est dans le bloc Claims Administration, considéré comme une fonction métier (*Business function*), tandis que les participants (deux rôles et un groupe) sont représentés en dessous.

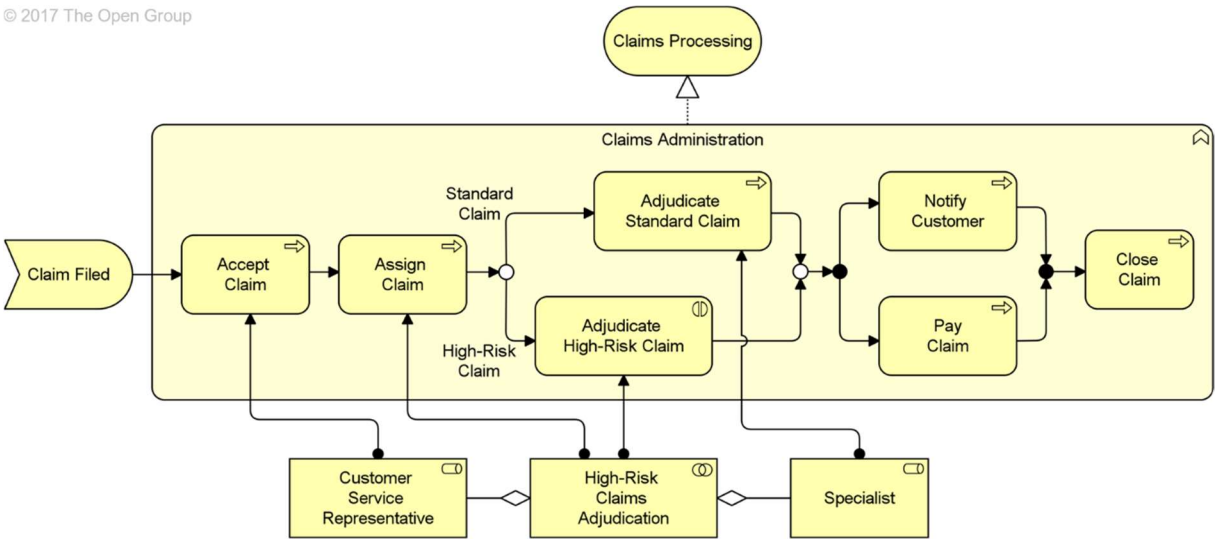


Figure 32 : Exemple de processus métier représenté selon ArchiMate 3

Notez les petits symboles en haut à droite de chaque rectangle qui permettent de distinguer les fonctions métier (qui regroupent des activités selon les compétences, les connaissances et les ressources nécessaires) des processus métier (qui décrivent un séquencement d'activités, comme en BPMN) et les rôles individuels des groupes collaboratifs (ces groupes étant une agrégation de plusieurs rôles individuels). Les significations de ces quelques symboles sont indiquées sur la Figure 33.

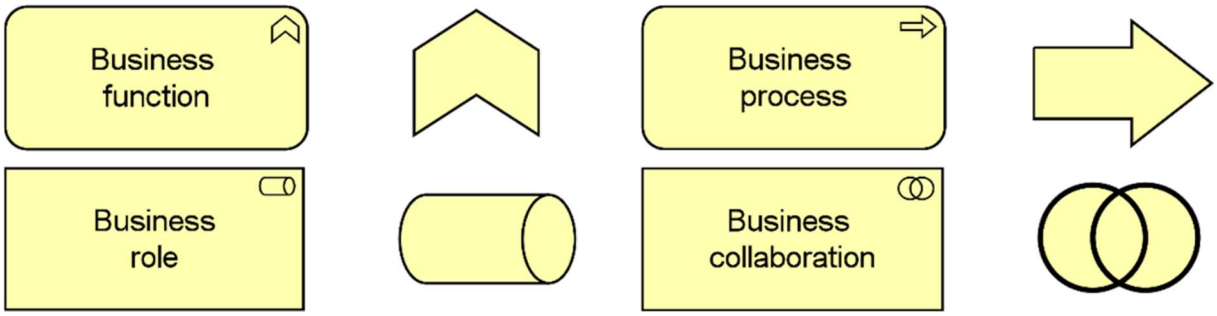


Figure 33 : Quelques symboles graphiques d'ArchiMate 3 avec leurs significations

4.2.3 Modélisation des décisions

4.2.3.1 Décisions et Règles métier dans les Cadres de Représentation

Les cadres et méthodes d'Architecture d'Entreprise ne s'intéressent principalement qu'aux décisions tactiques pour sa mise en œuvre. Les décisions stratégiques sont prises en amont : ce sont celles sur lesquelles il faut aligner l'Architecture d'Entreprise, grâce aux décisions tactiques. Les décisions tactiques d'Architecture, qui ont un impact structurant sur l'organisation notamment, devront être justifiées et tracées : on doit pouvoir savoir pourquoi et quand on a pris telle décision.

Par contre, les décisions opérationnelles n'y figurent pas, car elles apparaissent à un niveau de détail trop précis. Quant aux règles, il s'agit de règles fondamentales d'architecture, voire de principes généraux, mais pas des règles métier.

4.2.3.2 Décisions et Règles métier dans la modélisation d'entreprise

Dans le contexte plus précis de la modélisation d'entreprise, il apparait donc que les langages et notations standards utilisés pour la représentation des processus métier (diagramme d'activité UML et diagramme de collaboration BPMN) se révèlent peu adaptées pour la représentation des prises de décisions opérationnelles et des règles.

Rappelons la contribution principale du monde académique avec les grilles de décision de la méthode GRAI, avant de présenter les contributions principales du monde industriel (ces deux mondes nous semblent relativement cloisonnés l'un de l'autre).

Quelques notations intéressantes à l'initiative d'éditeurs spécialisés sont apparues ces dernières années, mais elles sont propriétaires. La notation propriétaire TDM (The Decision Model) est toutefois considérée comme l'ancêtre de la notation standard DMN (Decision Model and Notation).

4.2.3.3 GRAI

4.2.3.3.1 GRAI, une méthode globale

La méthode GRAI (Graphe à Résultats et Activités Inter-reliés, mais cette signification détaillée semble peu usitée) est le fruit du travail de chercheurs de l'Université de Bordeaux commencé à la fin des années 1980. Elle est basée notamment sur les théories de la décision [Simon, 1983] et de la systémique [Le Moigne, 1994].

La méthode GRAI est basée sur la modélisation d'entreprise dans le but de concevoir ou de transformer les systèmes de production (de produits ou de services). Il s'agit bien d'une méthode, utile à tout projet de modélisation pour la conception, car elle propose une démarche structurée, avec des étapes prédéfinies à suivre.

La méthode GRAI vise globalement l'amélioration de la performance. Elle est basée sur un modèle de référence avec tous les concepts génériques et leurs relations, permettant de modéliser tout système de production, grâce à des langages de modélisation graphiques (grilles, réseaux, diagrammes de classes et actigrammes servant à représenter le système physique).

Cependant, contrairement à des langages comme UML ou BPMN pour lesquels les métamodèles sont complètement définis dans les documents de spécifications, les formalismes utilisés dans la méthode GRAI n'ont fait l'objet que d'une définition partielle de leur métamodèle dans les différentes publications et travaux de recherche, comme [Roque, 2005].

L'originalité principale de la méthode GRAI est de proposer une démarche et un formalisme pour aborder la modélisation de la partie décisionnelle [Doumeingts et al., 1998], pour la conduite du système physique de l'Entreprise représenté sur la Figure 34. Notre champ d'investigation étant la prise de décision, nous nous focaliserons donc sur la modélisation de la partie décisionnelle de la méthode GRAI.

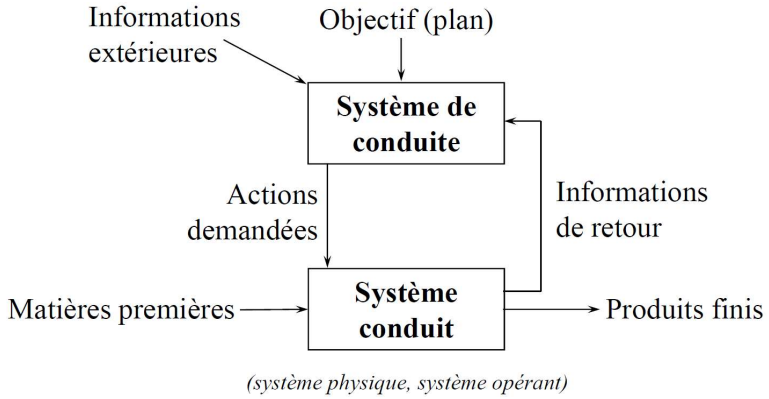


Figure 34 : La conduite d'un système selon la méthode GRAI

4.2.3.3.2 Partie décisionnelle de la méthode GRAI

Les décisions à prendre avec l'aide du système de conduite, qui concernent la gestion des produits (et des services), la gestion des ressources et la planification des deux ensembles, comme indiqué sur la Figure 35, ne sont pas uniquement des décisions opérationnelles, comme on pourrait le supposer, mais également des décisions stratégiques et tactiques.

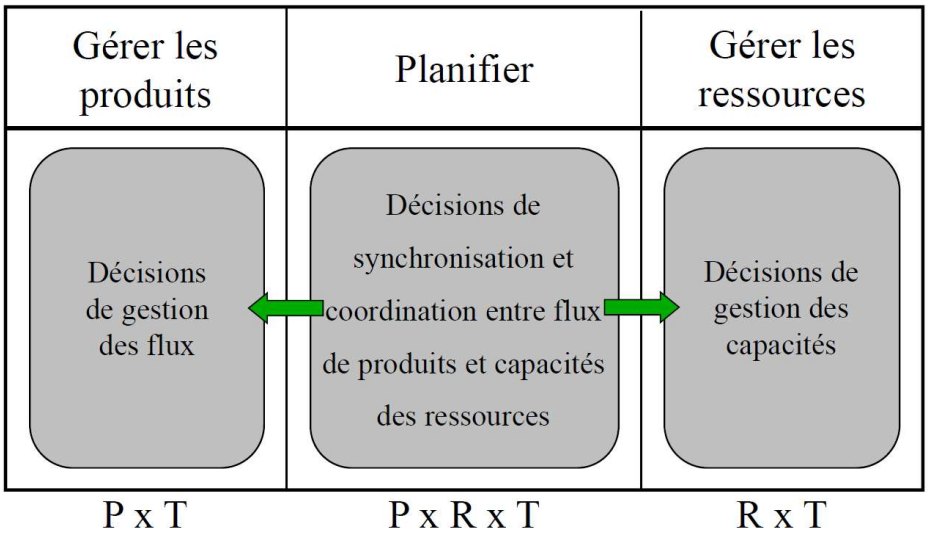


Figure 35 : Les trois fonctions élémentaires de conduite selon la méthode GRAI

La méthode GRAI se focalisant donc sur la partie décisionnelle, cette partie est très détaillée. Ainsi, en plus de distinguer classiquement les décisions stratégiques, tactiques et opérationnelles, la méthode GRAI distingue les décisions nominales, prises de façon périodique des prises de décisions non nominales, prises uniquement lorsqu'un événement survient.

4.2.3.3.3 Grille GRAI

La grille GRAI est le premier formalisme de modélisation de la méthode GRAI. N.B. Une grille ne sert qu'à la représentation des décisions nominales (périodiques). Les fonctions de l'Entreprise sont les colonnes de cette grille ; une première colonne « Informations externes » et une dernière colonne « Informations externes » sont ajoutées ; les niveaux temporels (couples horizons-périodes) sont les lignes.

Les niveaux, numérotés, sont classés par ordre décroissant : les couples horizons-périodes à long terme en haut, pour les décisions stratégiques, les couples à moyen terme au milieu, pour les décisions tactiques et les couples à court terme en bas, pour les décisions opérationnelles. Dans l'exemple reproduit sur le Tableau 5 ci-dessous, la période TR signifie Temps Réel :

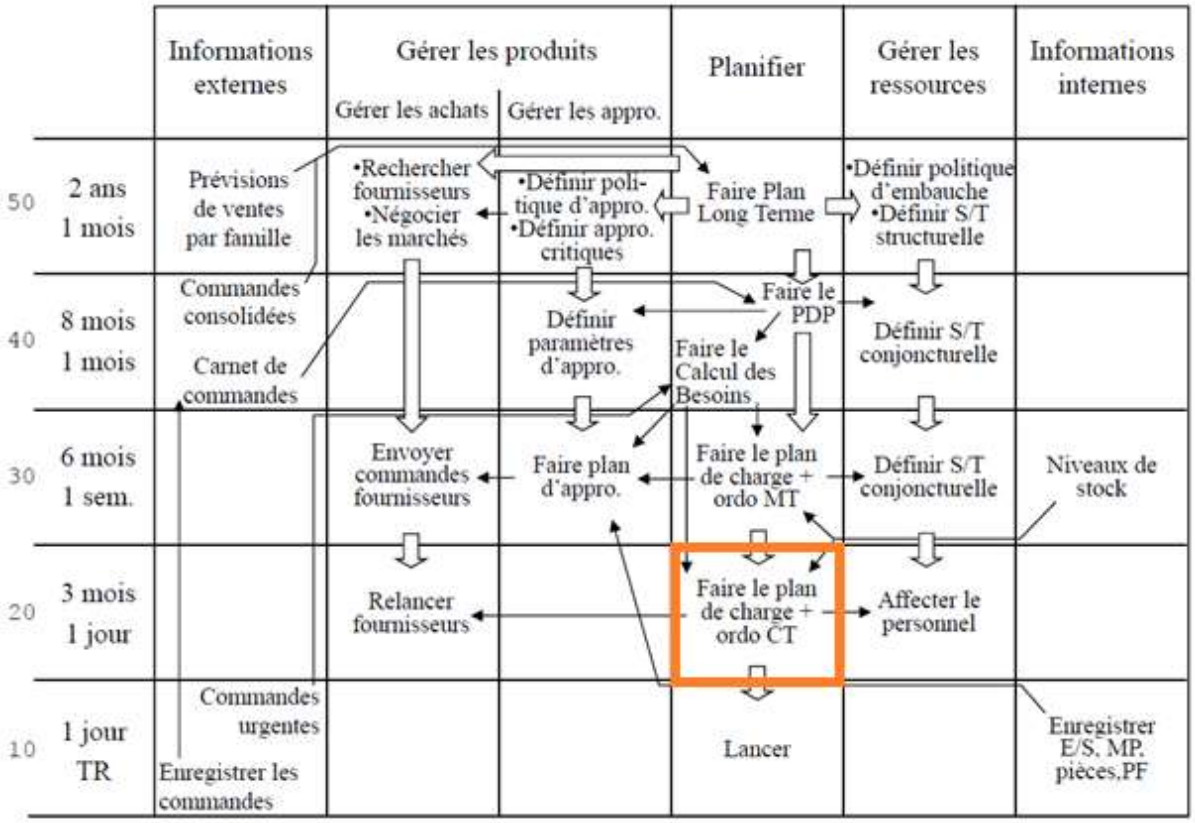


Tableau 5 : Grille de conduite GRAI (Exemple)

Toute cellule remplie de la grille, représentant un couple fonction-niveau, constitue un centre de décision. Des flèches épaisses représentent les cadres de décisions (objectifs, variables, contraintes sur ces variables et critères de chaque décision), tandis que des flèches fines, optionnelles, représentent les flux d'information.

A noter une définition peu courante, mais néanmoins pertinente, des Indicateurs de Performance : ils sont considérés comme les résultats des décisions passées.

4.2.3.3.4 Réseau GRAI

Chaque centre de décision est associé à un réseau GRAI qui représente son fonctionnement. Le centre de décision PL/20 (Colonne Planifier / Niveau 20) surligné dans le Tableau 5, sera détaillé page suivante. Il y a une représentation graphique horizontale pour les activités d'exécution et une représentation graphique verticale pour les activités de décision, dont la Figure 36 montre les principes.

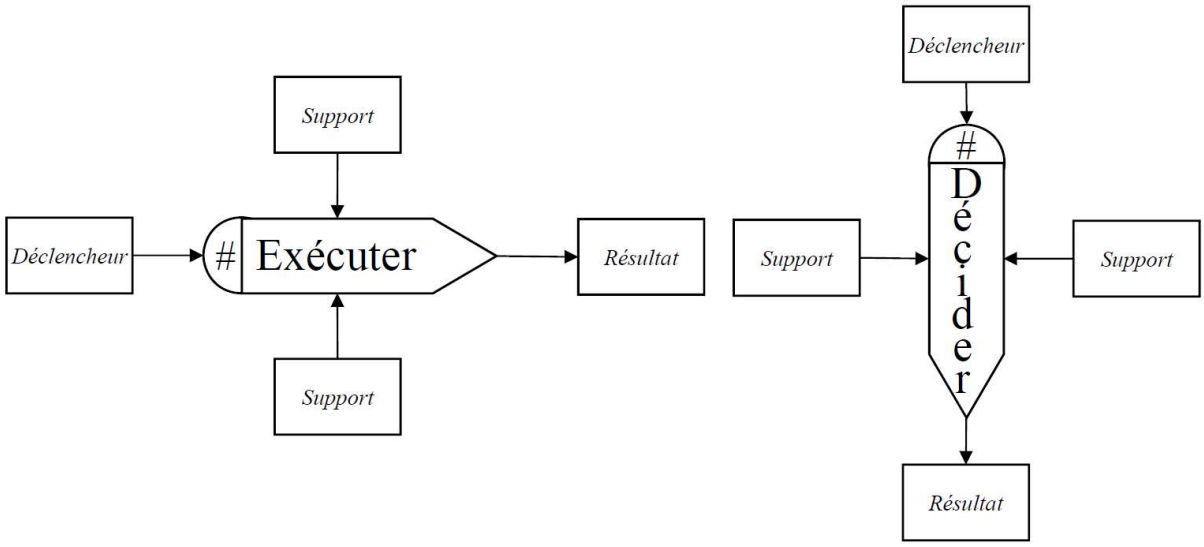


Figure 36 : Réseau GRAI pour Activité d'Exécution et Activité de Décision (Principes)

Les activités d'exécution et de décision utilisent les mêmes entités, représentées sous forme de rectangles : le déclencheur déclenche l'activité et le résultat est produit par l'activité. Les entités de support sont nécessaires au déroulement de l'activité ; elles sont obligatoires pour les activités de décision.

Il existe 7 natures différentes d'entités : Information, Critère, Indicateur de performance, Objectif, Règle, Ressource et Variable de Décision (VD). Leur nature est indiquée en haut de chaque rectangle (sauf Information, qui est la nature d'une entité par défaut). Cette description volontairement succincte ne vise qu'à comprendre l'exemple qui suit.

La Figure 37 propose un vrai exemple de réseau GRAI qui combine deux activités d'exécution (horizontales) puis une activité de décision (verticale). Ce réseau GRAI est associé à la cellule PL/20 (Colonne Planification / Niveau 20) de la grille GRAI précédente (Tableau 5).

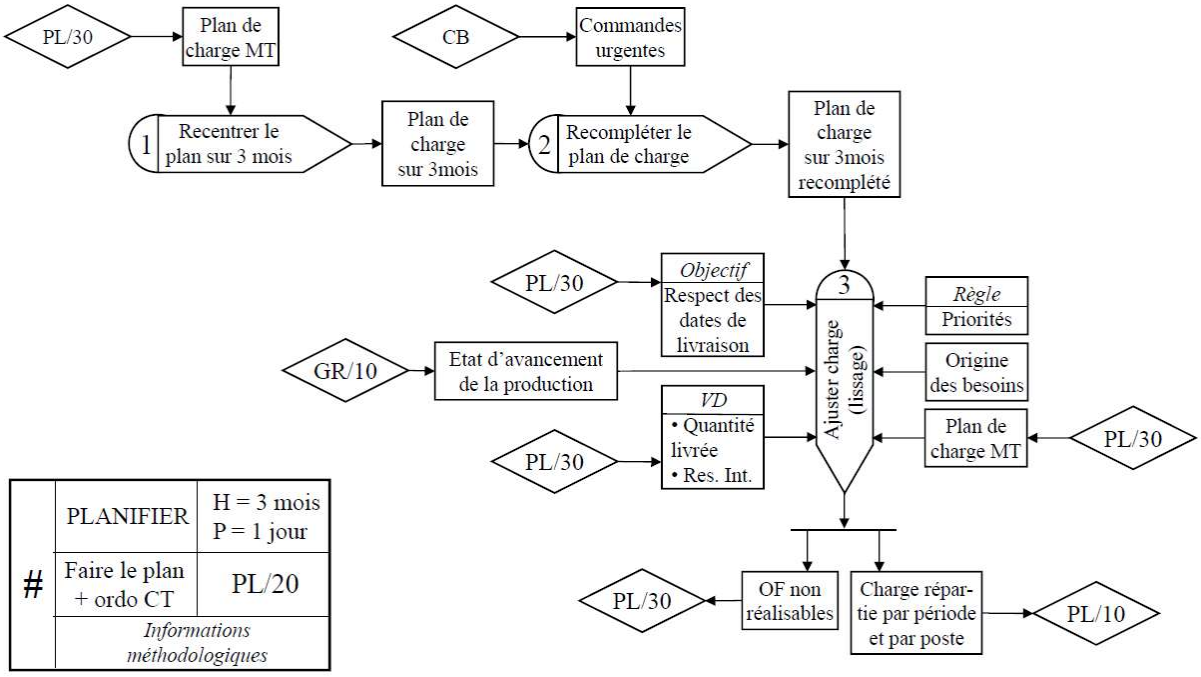


Figure 37 : Réseau GRAI pour Activité d'Exécution et Activité de Décision (Exemple)

Le symbole en forme de losange, dit opérateur de renvoi, représente une relation avec un autre centre de décision, en tant qu'origine ou en tant que destination (R/A : Réseau/Activité), tandis que le symbole de droite sur la Figure 38 représente l'opérateur logique OU (divergent) :

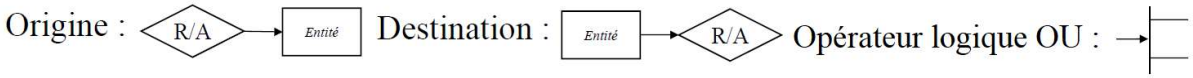


Figure 38 : Quelques symboles supplémentaires du réseau GRAI

Contrairement à d'autres langages ou notation pour la représentation des prises de décisions, l'activité de décision (verticale) du réseau GRAI n'a pas besoin d'un environnement (ou contexte) prédéfini. Ainsi, une décision peut être différente, alors que les entités d'entrée sont identiques (Exemple : choix d'une solution en contexte incertain).

Cette possibilité est sans doute utile pour représenter la réalité des décisions humaines, mais elle nous semble peu adaptée pour pouvoir prendre des décisions automatisées. L'activité d'exécution (horizontale) est quant à elle déterministe (des entités d'entrée identiques donnent toujours le même résultat en sortie).

4.2.3.4 TDM (The Decision Model)

La première initiative marquante d'une entreprise privée (KPI) pour la modélisation des décisions avec une orientation métier fut TDM (The Decision Model) [Von Halle et Goldberg, 2009]. Les règles métier sont formalisées dans des tables de décision. La décision est symbolisée par un octogone, traditionnellement représenté en haut du diagramme, comme dans la Figure 39. Remarquer les deux traits plein et pointillé dans chaque « rectangle » vert, qui encadrent leurs dépendances. TDM a eu une grande influence sur le langage suivant DMN.

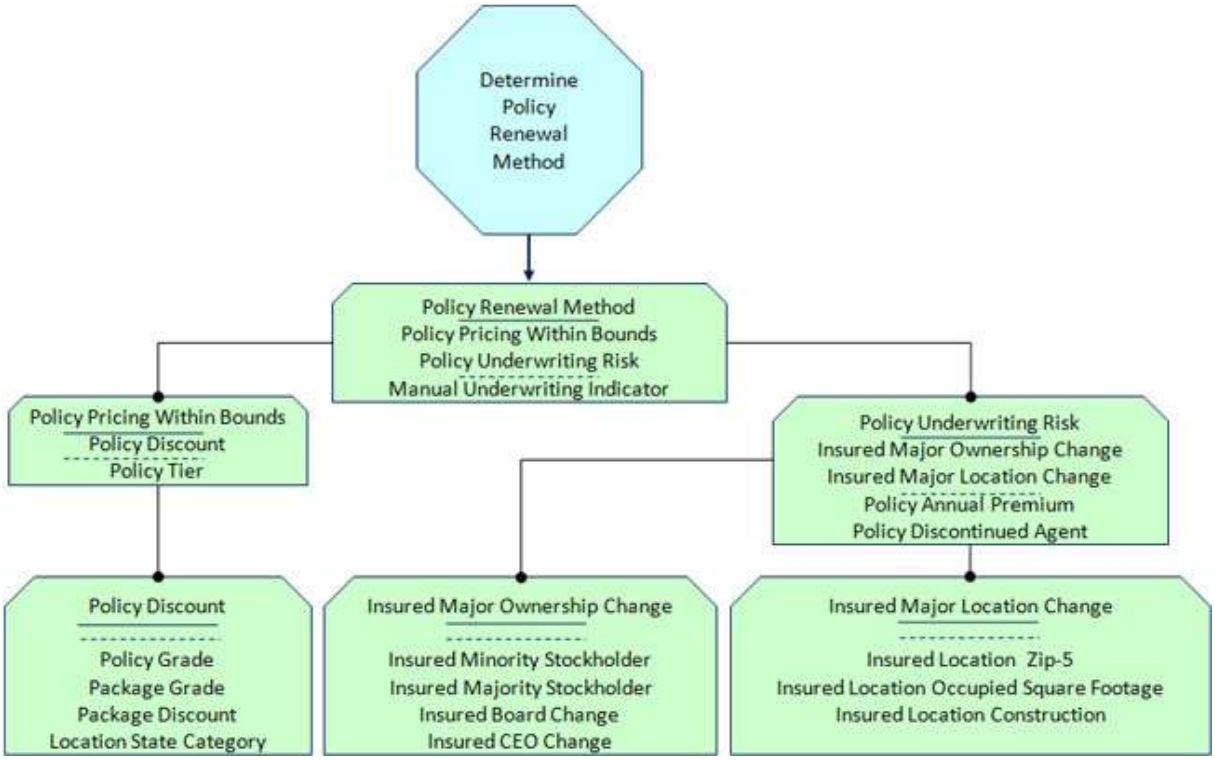


Figure 39 : The Decision Model (TDM) example (KPI)

4.2.3.5 DMN (Decision Model and Notation)

Depuis 2014, un nouveau langage nommé DMN (Decision Model and Notation) est venu compléter le bouquet de standards proposés par l'OMG (Object Management Group) dans le cadre de la modélisation métier (Business Modeling).

Ce langage DMN permet de modéliser les prises de décision et les règles métier. Son objectif principal est de fournir une représentation de la prise de décision compréhensible par toutes les parties prenantes (acteurs métier qui gèrent et pilotent les décisions, analystes métiers qui spécifient les exigences relatives aux prises de décisions, développeurs responsables de l'automatisation de tout ou partie de la prise de décision).

Nous n'en dirons pas plus dans ce chapitre, puisque le langage DMN est l'objet principal de ce mémoire et sera détaillé dans les chapitres suivants.

4.2.3.6 Autres langages et notations pour représenter les règles métier

Nous n'avons pas trouvé d'autres langages ou notations standards pour la formalisation des prises de décision. Par contre, il y en a quelques autres pour la formalisation des règles métier et notamment la représentation de leur sémantique :

- RuleML (Rule Markup Language, mais aussi Rule Modeling Language) est supporté par l'association sans but lucratif éponyme. Cette association est dévouée à la cause de la sémantique des règles métier et à leur interopérabilité. Elle a participé aux travaux sur SBVR (OMG) et sur RIF (W3C ; lire ci-dessous). Un traducteur expérimental est disponible pour traduire le langage DMN en langage RuleML,

- RDF, OWL & RIF proposés par le W3C (World Wide Web Consortium) :
RDF : Resource Description Framework,
OWL : Web Ontology Language,
RIF : Rule Interchange Format.
Ces standards sont essentiellement destinés au web sémantique. Ils n'ont aucun lien connu avec le langage DMN et ne sont pas présentés ici.

5 Proposition de Solution

5.1 Séparation des préoccupations

La Séparation des Préoccupations (*Separation of Concerns*) [Dijkstra, 1974] est un bon principe provenant de l'informatique. Il date donc de 1974, mais s'agissant d'un principe indépendant de toute technologie, son application est toujours préconisée.

Dans le contexte qui nous intéresse, la Séparation des Préoccupations est notamment appliquée par l'Architecture Dirigée par les Modèles (*MDA* : *Model-Driven Architecture).* Les préoccupations seront séparées, afin de réduire sa complexité. Chaque préoccupation sera traitée par un aspect du système, aussi indépendant que possible des autres aspects.

Lorsqu'on applique ce principe à notre contexte, il s'agira alors d'extraire les prises de décision des modèles de processus métier, afin de modéliser leur structure séparément. Les règles métier elles-mêmes seront également modélisées à part. C'est l'application des règles métier qui sert généralement à prendre des décisions.

L'avantage de la séparation de ces modèles faiblement couplés est leur relative indépendance facilitant les changements. Chaque modèle, autonome et consistant, peut évoluer séparément : c'est un gage d'agilité, voire de résilience.

Pour comprendre l'un des modèles, il n'y a pas besoin de connaitre l'autre. Aussi, chaque modèle peut s'adresser à un public différent, comme celui des Analystes métier ou celui des Informaticiens, qui ont chacun des préoccupations différentes.

En appliquant le principe de séparation des préoccupations, ce mémoire vise à démontrer la pertinence (ou pas) du nouveau langage standard DMN (Decision Model and Notation) pour modéliser ces prises de décisions et ces règles métier différemment.

5.2 Evolution de l'Architecture des Applications Métier

Nous observons l'influence de la Séparation des Préoccupations sur l'évolution de l'architecture des applications métier depuis plusieurs décennies. Dans la Figure 40 ci-dessous, le symbole des Règles représente aussi les prises de décision :

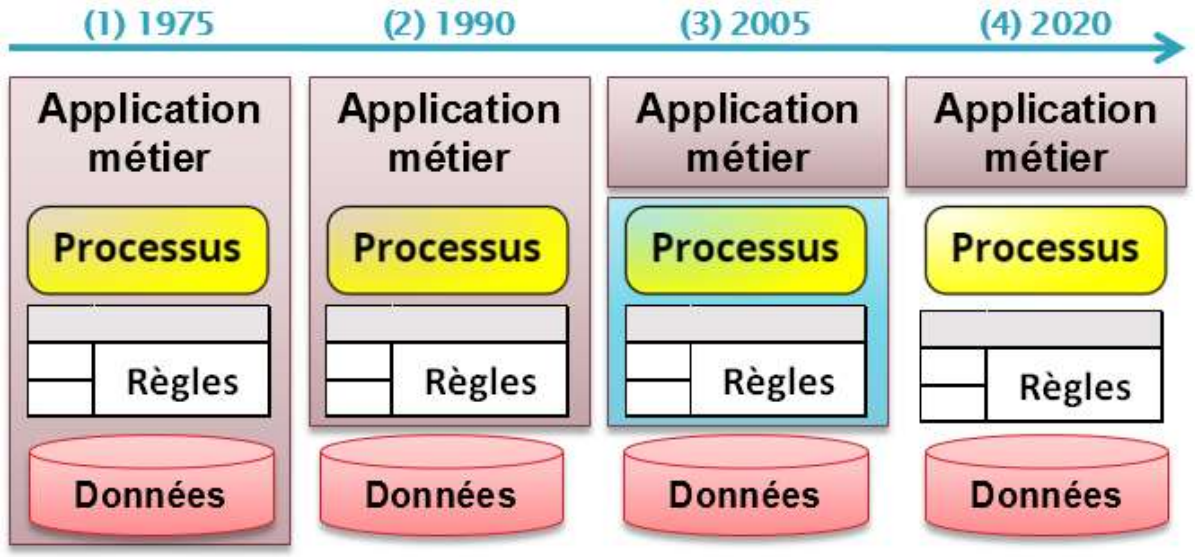


Figure 40 : Evolution de l'architecture des applications métier

Nous distinguons 4 grandes étapes dans l'évolution des applications métier (les dates sont estimées par l'auteur) :

(1) Les premières applications "monolithiques",

(2) L'externalisation massive des données dans des SGBD (Systèmes de Gestion de Bases de Données). C'est le langage SQL qui a permis aux applications d'accéder aux SGBD,

(3) L'extraction des processus et des règles métier, dans des suites BPM (Business Process Management), mais les règles restant alors imbriquées dans les processus,

(4) L'extraction des décisions et règles métier des processus.

C'est l'Architecture Orientée Services (SOA) qui a permis aux processus d'orchestrer plusieurs applications métier et ensuite d'invoquer les moteurs de règles.

Les moteurs de règles existent depuis une quinzaine d'années déjà et il est donc possible d'automatiser les prises de décisions dans les processus métier [Fish, 2012]. Par contre, la modélisation standardisée des prises de décisions (objet principal de ce mémoire) est récente et sera développée dans les chapitres suivants.

5.3 Bouquet de standards de l'OMG

5.3.1 Cartographie des principaux standards

L'OMG (Object Management Group) publie des dizaines de spécifications de standards, dont le plus connu est UML (Unified Modeling Language) [OMG, 2015b]. La Figure 41 met en évidence, grâce à des bandeaux de couleur, les standards qui nous intéressent ici, ceux du domaine Business Modeling.

Nous positionnerons le langage DMN par rapport aux autres spécifications standards, qui sont des relations avec lui (formalisées dans son métamodèle ; extrait commenté plus loin).

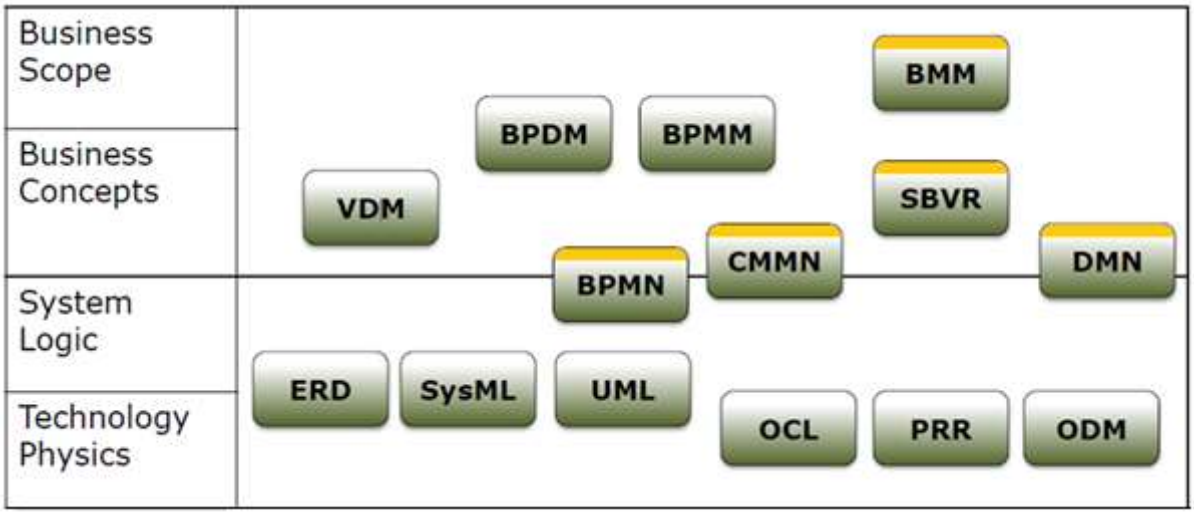


Figure 41 : Bouquet de standards de l'OMG, classés selon [Pitschke, 2014]

5.3.2 Langage versus Notation

Bien que les sigles de 3 standards BPMN, CMMN et DMN qui nous intéressent se terminent par « N » comme Notation, il s'agit en fait de langages. Car il faut prendre en compte le « M » comme (Meta) Model, qui précède le « N ».

« Model and Notation » correspondent bien à un langage, constitué de 4 composants principaux définis par un métamodèle [Clark et al., 2015], décrit dans une spécification détaillée :

- Sa syntaxe abstraite (les concepts et leurs relations),
  par exemple, le concept d'Activité BPMN se spécialise en Tâche ou Sous-Processus,

- Ses règles de construction (qui s'appliquent aux concepts),
  par exemple, si un processus BPMN possède un événement de début,
  il doit avoir au moins un événement de fin,

- Sa syntaxe concrète (notation graphique et/ou textuelle pour représenter les concepts),
  par exemple, une Activité BPMN (Tâche ou Sous-Processus) est représentée
  par un rectangle à coins arrondis,

- Sa sémantique (pour décrire la signification des concepts),
  par exemple, une Activité BPMN est un travail qui est exécuté
  dans le cadre d'un processus métier.

Qualifier un Langage de Notation serait donc réducteur. Un métamodèle décrit dans une spécification détaillée est fourni avec chaque langage standard proposé par l'OMG.

5.3.3 BMM

BMM (Business Motivation Model) [OMG, 2015a] pour définir les objectifs d'une entreprise, peu connue, car peu utilisée directement, mais qui sert de référence à d'autres standards. BMM fait référence au cadre de Zachman pour répondre aux questions Why, Who, How. Une décision représentée en DMN peut supporter un objectif (*Objective*) représenté en BMM.

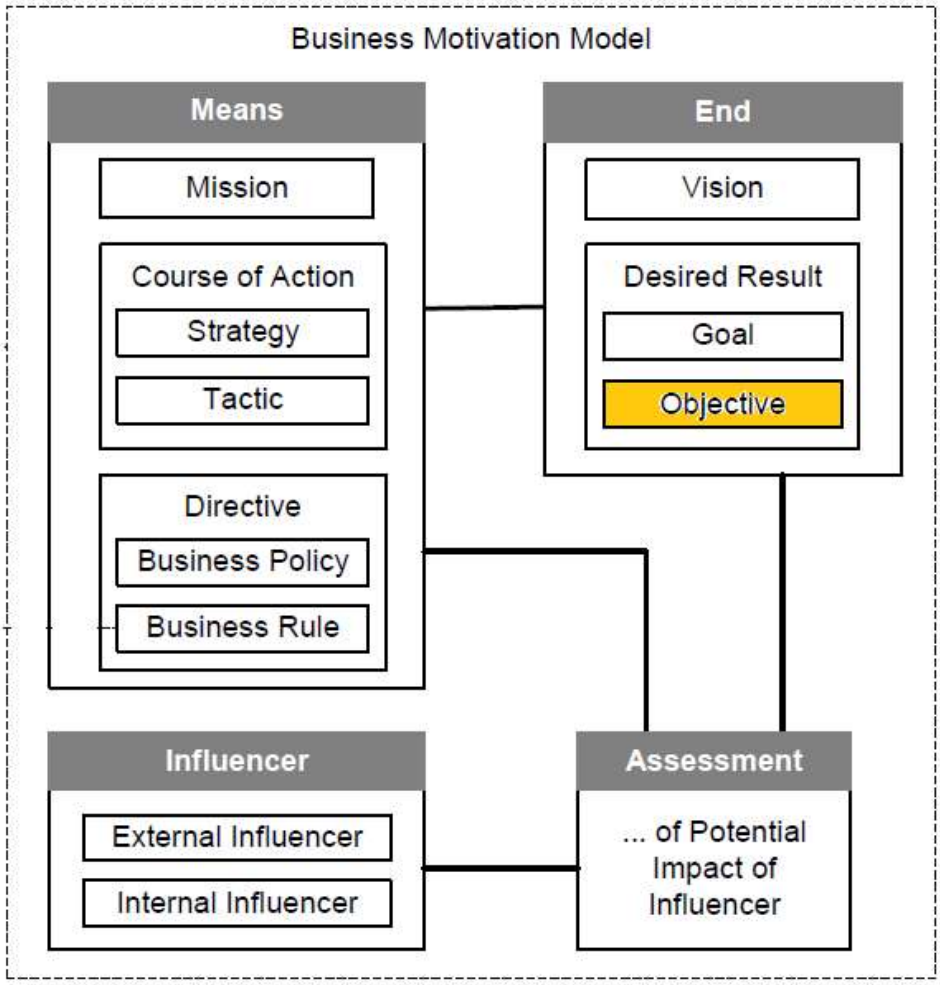


Figure 42 : Business Motivation Model (OMG)

La Figure 42 est en fait un métamodèle, car BMM n'a pas de représentation graphique : ce n'est donc pas une notation. Sans doute cela a-t-il limité son utilisation directe.

5.3.4 SBVR

SBVR (Semantics of Business Vocabulary and Business Rules) [OMG, 2017], pour définir le vocabulaire métier, selon un formalisme particulier, que DMN peut utiliser, c'est une simple recommandation [Linehan et de Sainte Marie, 2011]. Contrairement aux autres langages du bouquet de standards de l'OMG, SBVR n'a pas de notation graphique. Est-ce là l'explication de son succès dans les entreprises ? Exemple :

Figure 43 : Exemple d'une règle métier exprimée selon SBVR

5.3.5 BPMN

BPMN (Business Process Model and Notation) [OMG, 2013], pour modéliser les processus métier ; le langage DMN peut s'utiliser seul ou en complément d'un processus métier représenté en BPMN ; leur complémentarité sera démontrée plus loin. BPMN est prédictif : un processus est décrit en une séquence prédéfinie d'activités, qui représente la routine en quelque sorte. Les processus réalistes représentent toutefois les aléas et autres perturbations : il faut souvent prendre des décisions, représentées par des branchements (*gateways*), avant d'emprunter des chemins alternatifs.

Le langage BPMN complète est assez riche, comptant une centaine d'éléments graphiques différents (dont de nombreuses variantes). La Figure 44, extraite du poster BPMN proposé par un ensemble d'éditeurs berlinois [BPM Offensive Berlin, 2015], représente un processus métier imaginaire, mais qui utilise presque tous ces éléments.

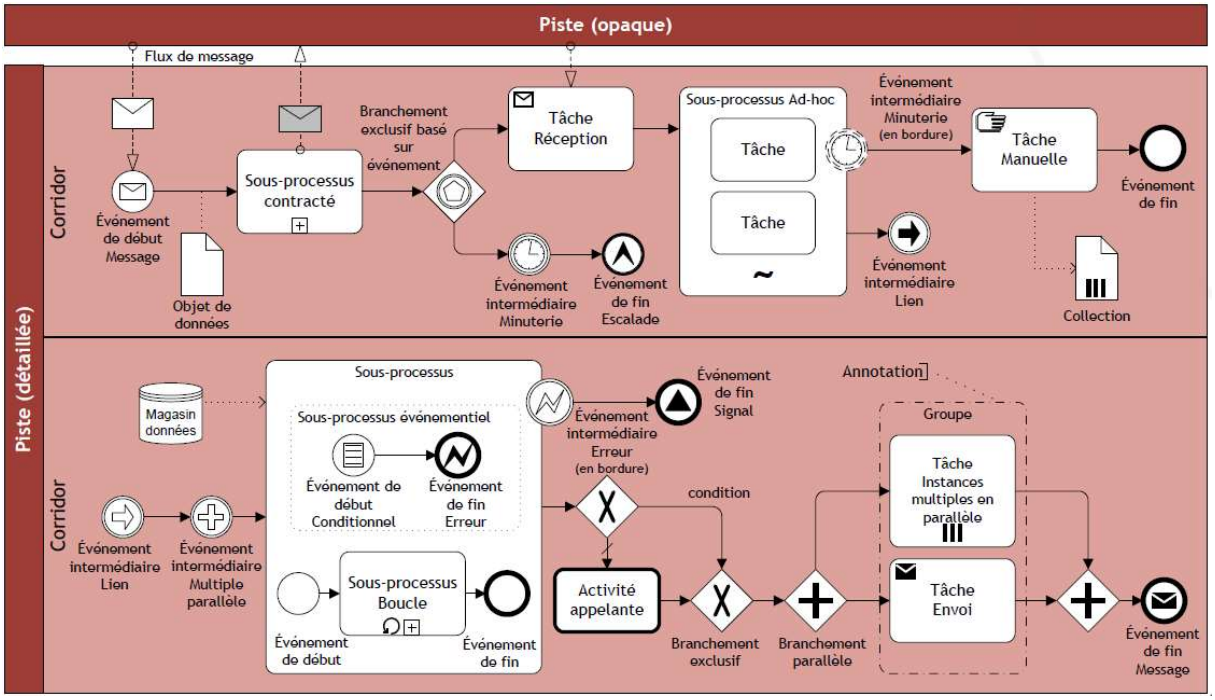


Figure 44 : Diagramme de collaboration BPMN (processus métier imaginaire)

5.3.6 CMMN

CMMN (Case Management Model and Notation) [OMG, 2016a] est une alternative à BPMN, quand les activités qui composent un processus métier peuvent s'exécuter dans un ordre indéterminé. Ce langage est proche du sous-processus BPMN de type "Ad Hoc".

CMMN n'a aucune relation directe avec le langage DMN, mais fait partie du même domaine Business Modeling des langages standards de l'OMG. CMMN étant non prédictif, il semble effectivement bien difficile de décider « à l'avance » des chemins alternatifs à emprunter.

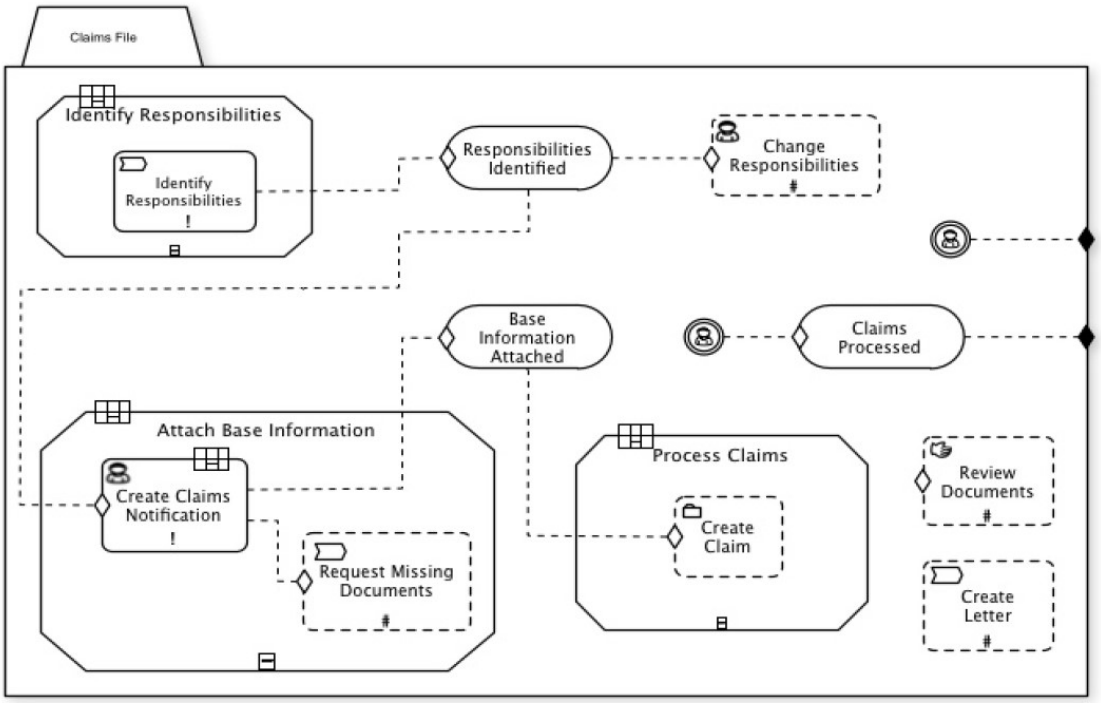


Figure 45 : Claims Management Example in CMMN (OMG)

5.3.7 DMN

Le langage DMN (Decision Model and Notation) [OMG, 2016b] est un standard relativement récent de l'OMG. La première version 1.0 bêta de la spécification DMN a été publiée en février 2014, mais présentait plusieurs lacunes dans son métamodèle (par exemple : des espaces incongrus dans les noms de classes, voire pas de nom du tout), rendant son utilisation difficile. Nous avons néanmoins réussi à développer sur cette base notre propre outil de modélisation DMN [Biard et al., 2015].

La version 1.0 finale, qui n'était pas encore stable, a été publiée en septembre 2015. C'est la version 1.1, publiée en juin 2016, qui a permis enfin son utilisation réelle et qui sert de base à la plupart des outils disponibles sur le marché aujourd'hui.

La durée de cette thèse nous a permis de suivre dans le temps l'évolution de ce langage standard DMN, qui reste encore méconnu.

L'OMG ne s'est pas contenté de rédiger une spécification de 182 pages décrivant en détail son métamodèle : un schéma sous la forme d'une fichier XSD, essentiellement à l'attention des éditeurs de logiciel, est également fourni. Chaque standard de l'OMG est fourni avec un métamodèle, sous la forme de diagramme de classes UML.

L'extrait du métamodèle (diagramme de classes UML) concernant la Décision sur la Figure 46 formalise, les relations éventuelles des modèles DMN avec les modèles BMM et BPMN :

- La relation à droite indique qu'une Décision peut soutenir un Objectif (*Objective*) préalablement défini dans un modèle BMM (Business Motivation Model). Cette relation est peu utilisée, car il existe rarement un modèle BMM en amont.

- Les deux relations à gauche indiquent qu'une Décision peut être utilisée par une Tâche et/ou par un Processus définis dans un modèle BPMN (Business Process Model and Notation) version 2.0. La relation entre une Décision et une Tâche est très utilisée ; nous démontrerons plus loin la complémentarité entre DMN et BPMN.

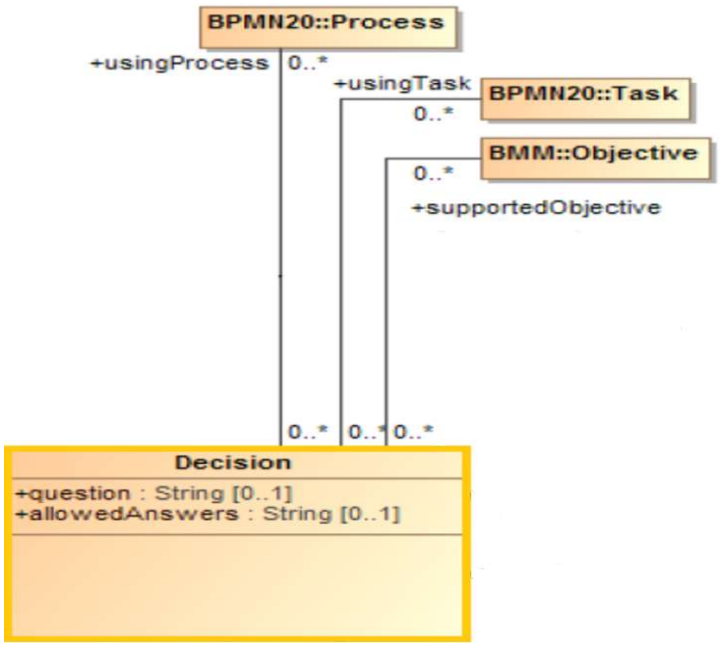


Figure 46 : Relation d'une Décision DMN avec BMM et BPMN

5.3.8 BPMN + CMMN + DMN

L'OMG affirme que le trio de langages BPMN + CMMN + DMN (nommé *triple crown*) permet de répondre à tous les besoins d'amélioration des processus. Le Tableau 6 présente les différences entre des trois langages, qui ne sont pas concurrents, mais complémentaires.

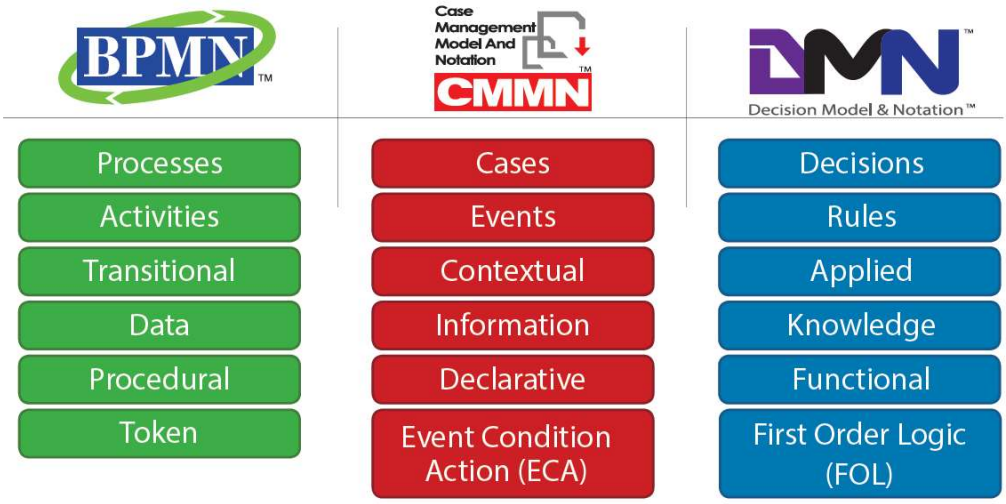


Tableau 6 : Différences en termes de notions de base et de sémantique (OMG)

N.B. La représentation des prises de décisions selon le langage DMN peut être qualifiée de fonctionnelle. Par contre, par la représentation des règles métier par des tables de décision selon le langage DMN est déclarative, comme cela sera démontré plus loin.

Pour le coup, avec ces 3 langages différents, il devient légitime de se demander quel est le langage adéquat à utiliser en fonction du besoin de représentation. Le Tableau 7 donne quelques éléments de réponse sous la forme de bonnes pratiques pour choisir le langage adéquat. La formule "BPMN + Gateways = DMN" sera largement illustrée par la suite.

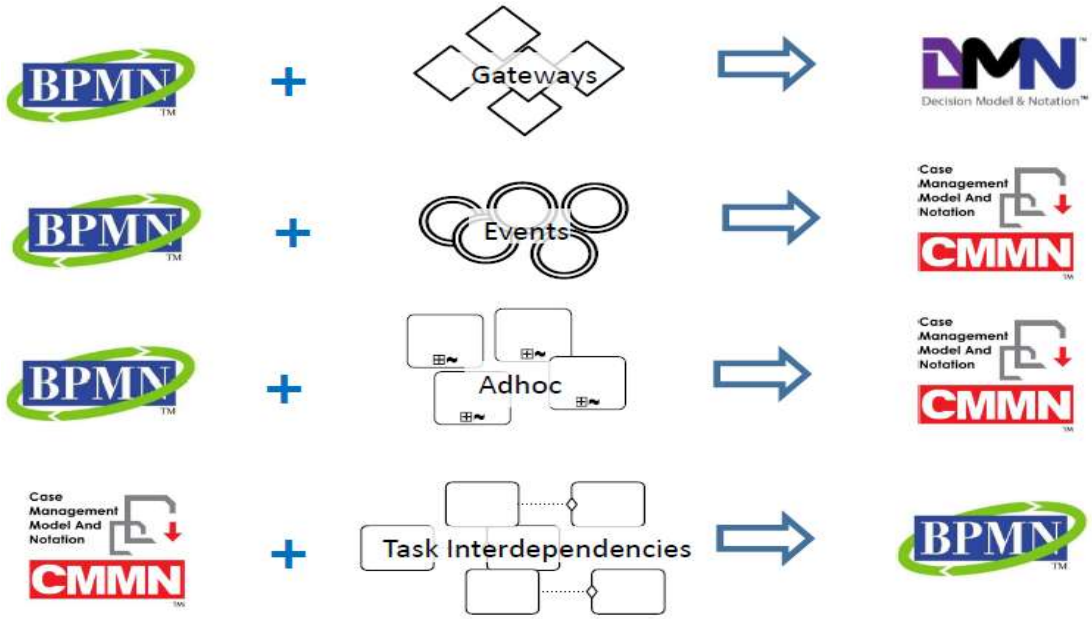


Tableau 7 : Bonnes pratiques d'usage de BPMN, CMMN ou DMN

Certains éditeurs comme [Trisotech, 2017] proposent une suite complète et cohérente d'outils permettant de modéliser selon ces 3 langages standards.

5.4 Modélisation des décisions avec DMN

5.4.1 DRD (Decision Requirements Diagram)

5.4.1.1 Quatre éléments graphiques principaux

La figure emblématique du langage DMN est le *Decision Requirements Diagram* (DRD), afin de représenter les exigences d'une prise de décision. Ce diagramme contient quatre éléments graphiques principaux. Les deux premiers éléments sont obligatoires ; les deux derniers facultatifs :

– *Input Data* (Donnée d'entrée nécessaire pour prendre une décision ; au moins une, généralement plusieurs ; doit être typée : nombre, texte, date, booléen, etc.) ;

– *Decision* (Décision ; au moins une ; la Décision applique une logique sur les Données d'entrée ; la Décision prise est en fait la donnée de sortie ; une Décision peut être constituée de plusieurs sous-décisions ; elles ont alors une relation de dépendance) ;

– *Knowledge Source* (on ajoute souvent sur le diagramme au moins une Source de Connaissance : un expert ou un texte de référence qui fait autorité dans le métier concerné) ;

– *Business Knowledge Model* (on ajoute plus rarement un Modèle de connaissance métier, contenant une logique de décision, qui peut être réutilisée dans plusieurs diagrammes ; un peu l'équivalent de la *Call Activity* de BPMN).

Le Tableau 8 indique dans la colonne de droite « Notation » les formes des éléments du Decision Requirements Diagram (DRD). A part le Business Knowledge Model avec ses deux coins coupés, mais dont l'utilisation est optionnelle et assez rare, les 3 autres éléments peuvent être représentés avec la plupart des applications ayant des fonctionnalités de dessin (Microsoft Excel par exemple). L'usage d'un modeleur spécialisé DMN n'est donc pas obligatoire, mais simplement recommandé (pour vérifier la conformité avec le métamodèle DMN).

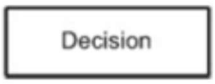
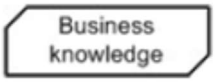
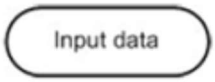
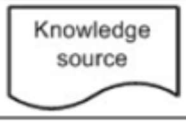
Component		Description	Notation
Elements	Decision	A decision denotes the act of determining an output from a number of inputs, using decision logic which may reference one or more business knowledge models.	
	Business Knowledge Model	A business knowledge model denotes a function encapsulating business knowledge, e.g. as business rules, a decision table, or an analytic model.	
	Input Data	An input data element denotes information used as an input by one or more decisions. When enclosed within a knowledge model, it denotes the parameters to the knowledge model.	
	Knowledge Source	A knowledge source denotes an authority for a business knowledge model or decision.	

Tableau 8 : Les 4 éléments graphiques du diagramme DRD de DMN (OMG)

5.4.1.2 Trois types de liens différents

Trois types de liens différents, les *Requirements*, permettent de représenter les relations entre ces quatre éléments graphiques principaux :

– *Information Requirement*, trait plein terminé par une pointe, qui relie :

- soit une Donnée d'entrée vers une Décision ;

- soit une Décision secondaire vers une Décision principale ;

– *Authority Requirement*, trait pointillé terminé par un point, qui symbolise l'invocation d'une Source de Connaissance ;

– *Knowledge Requirement*, trait pointillé terminé par une pointe, qui symbolise l'invocation d'un Modèle de connaissance métier.

Les types de liens adéquats (*Requirements*) pour relier les quatre éléments principaux, présentés dans le chapitre précédent, sont automatiquement sélectionnés par les outils de modélisation DMN, conformément au métamodèle, fourni par l'OMG. C'est pour cette raison que nous n'avons pas traduit leurs noms en français. Une quinzaine d'outils supportent déjà DMN [OpenRules, 2016].

5.4.1.3 Exemple de diagramme DRD

Tous ces éléments graphiques de DMN présentés ci-dessus (les quatre éléments principaux et les trois liens différents) sont utilisés dans le diagramme DRD de la *Figure 47* :

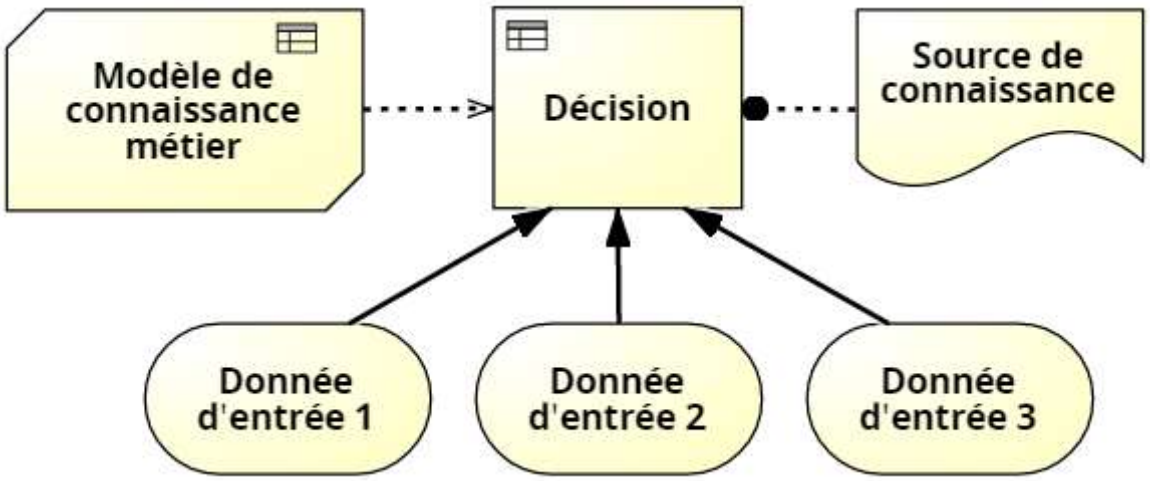


Figure 47 : Tous les composants graphiques de DMN dans un diagramme DRD

Une donnée d'entrée peut être partagée entre plusieurs sous-décisions, voire directement par la décision finale. Il est possible, comme dans un diagramme BPMN, d'ajouter des annotations, afin de commenter un diagramme DRD (aucun impact direct sur l'automatisation, bien sûr). Un crochet gauche symbolise l'artéfact d'annotation, relié à l'élément concerné par un lien de type association (trait pointillé sans extrémités particulières).

Les éléments graphiques du Decision Requirements Diagram peuvent avoir des couleurs différentes, afin de distinguer plus facilement les sous-décisions par exemple. Ces différentes options (annotations, liens d'association, couleurs) permettent d'effectuer une représentation raffinée, comme celle de la Figure 48 ci-après.

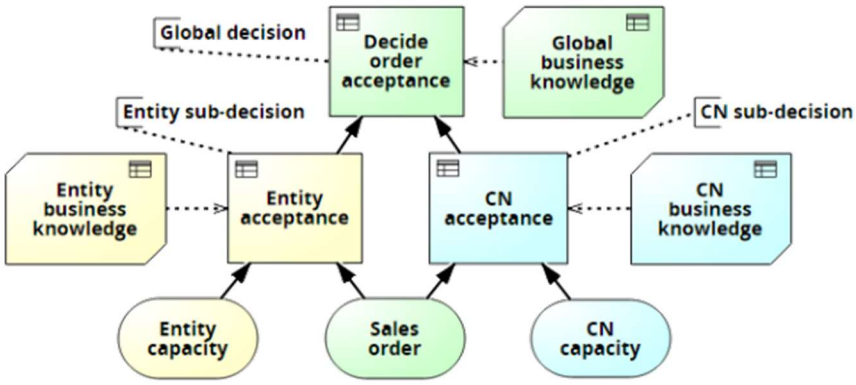


Figure 48 : Diagramme DRD avec donnée d'entrée partagée, annotations et couleurs

Graphiquement, il n'y a donc que 9 éléments de représentation dans DMN, dont seulement 3 qui sont obligatoires (Donnée d'Entrée, Décision et *Information Requirement* pour relier les deux premiers). En comparaison, BPMN propose environ une centaine d'éléments graphiques. Sous une apparente simplicité, le Decision Requirements Diagram peut toutefois devenir complexe (si la prise de décision est elle-même complexe). La Figure 49 ci-dessous montre un tel diagramme complexe (exemple officiel dans la spécification DMN de l'OMG).

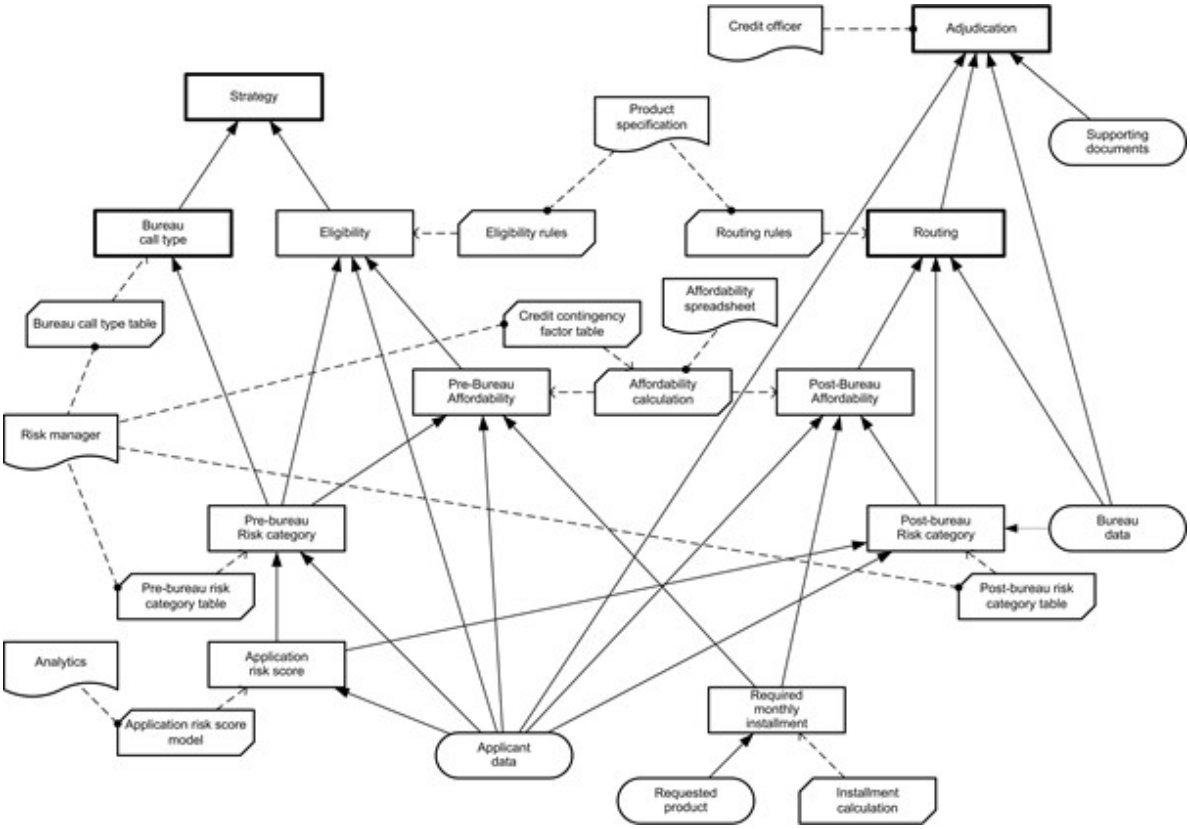


Figure 49 : Exemple de diagramme DMN complexe

Ce diagramme DMN représente les décisions à prendre lors de la mise en place d'un prêt bancaire. Il représente plusieurs décisions principales, comme Strategy et Routing, qui partagent des données d'entrée identiques, celles concernant le demandeur Applicant, ainsi que des décisions secondaires. Il est possible, voire conseillé, de séparer ces décisions principales dans des diagrammes distincts (et de répéter les données d'entrée sur chaque diagramme, forme de redondance, mais ne nuisant pas à la lisibilité de l'ensemble).

5.4.1.4 Extrait du métamodèle DMN

Les liens-relations entre les quatre éléments principaux sont explicites dans le métamodèle DMN (extrait en Figure 50), c'est-à-dire qu'ils sont représentés par des classes, ce qui n'est pas très courant (les liens-relations sont généralement implicites) et ce qui nous a posé problème lors de la réalisation de notre propre outil de modélisation DMN.

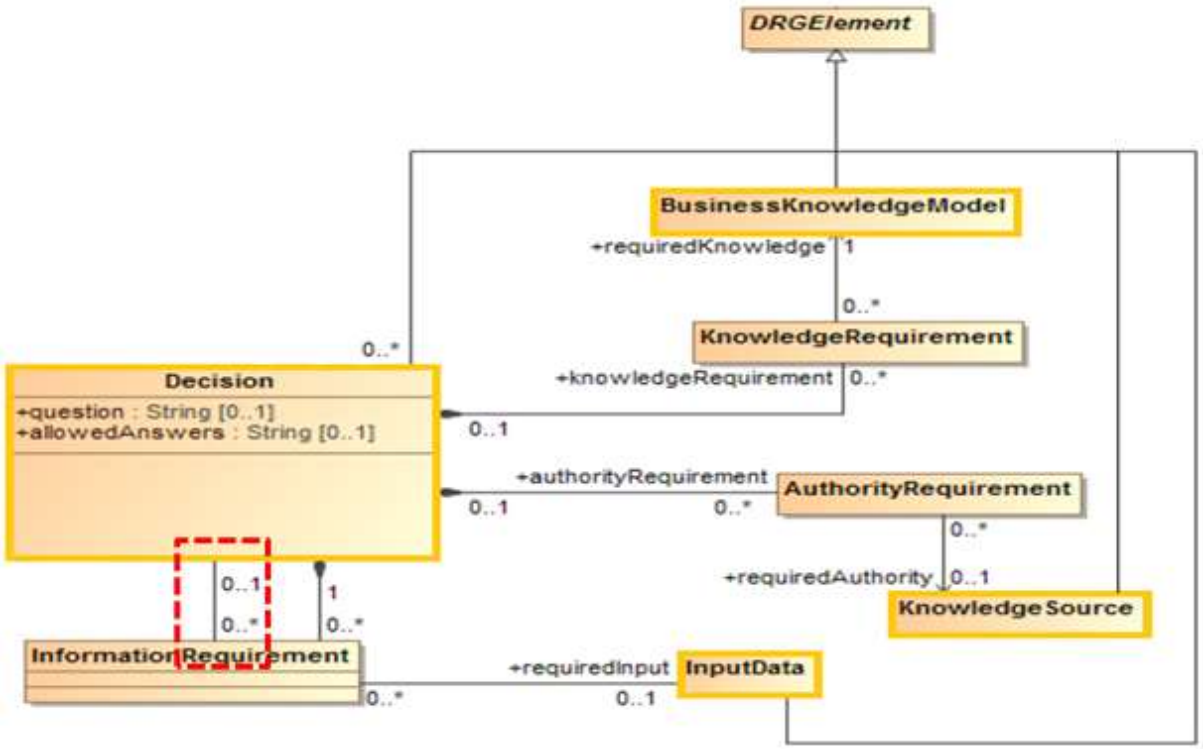


Figure 50 : Relations entre les 7 éléments d'un diagramme DRD

Les quatre classes représentant les quatre éléments graphiques principaux sont entourées d'un trait orange épais. Les trois classes représentant les trois types de liens différents sont entourées d'un trait noir fin.

La seconde relation entre Decision et InformationRequirement entourée d'un trait pointillé rouge formalise la possibilité qu'une décision (principale) puisse elle-même être composée de plusieurs décisions (secondaires), via des liens de type InformationRequirement, forme de récursivité indirecte.

5.4.2 Modélisation des Règles Métier

5.4.2.1 Différentes Façons d'Exprimer la Logique de Décision

Il existe trois façons dans DMN pour exprimer la logique d'une décision (Comment une décision – la donnée en sortie – est-elle prise en fonction des données en entrée ?) :

(1) Une expression littérale libre : un simple texte descriptif pour définir les règles métier, difficilement automatisable, mais qui laisse une marge au décideur pour son interprétation,

(2) Une expression littérale formalisée, qui utilise un langage de programmation générique comme Java, ou mieux encore, le DSL (*Domain Specific Language*) FEEL (*Friendly Enough Expression Language*), le langage déclaratif proposé dans la spécification DMN qui permet de formaliser la logique de décision sous la forme de code informatique, indépendamment de toute plate-forme [Silver, 2016]. Exemple :

```
if Vacataire.Age < 67 then true else false
```

(3) Une table de décision, qui est la façon la plus formelle et la plus courante, et qui est détaillée ci-dessous. Une table de décision est une forme particulière d'expression « en boîte » (*boxed expression*), dont l'usage doit être privilégié dans DMN.

Ces trois types d'expression de la logique de décision utilisent généralement les mêmes opérateurs de comparaison usuels sur les variables numériques (Exemple : « < » inférieur à). Optionnellement, on peut fixer pour chaque variable numérique un minimum et un maximum, c.-à-d. une plage de valeurs possibles (aussi appelé contrainte), voire préciser l'unité de mesure.

A noter que la représentation des exigences d'une prise de décision avec le diagramme DRD et l'expression de la logique de décision (selon l'une des trois façons précédentes) peuvent être effectuées par des personnes différentes, ayant les compétences adéquates. Et chacune de ces deux parties de DMN pourra évoluer de manière relativement indépendante de l'autre.

5.4.2.2 Représentation des Tables de Décision

La notation graphique - relativement simple - du DRD ne permet pas de représenter la logique de décision elle-même. Celle-ci est généralement représentée, surtout lorsque les critères sont relativement nombreux, par une table de décision [Vanthienen et Dries, 1994].

Le formalisme choisi dans la spécification DMN pour représenter les tables de décision est directement issu des travaux du CODASYL, qui datent de quelques décennies [CODASYL, 1982]. La formalisation des règles métier est une spécialité à part entière [R. G. Ross, 2013].

Dans les tables de décision dites horizontales (représentation recommandée), les règles métier sont listées sous forme de lignes. Les données d'entrée, qui sont donc les critères de décision, sont listées sous forme de colonnes, ainsi que la donnée en sortie (résultat de la décision).

Il doit y avoir une séparation claire entre les conditions (données en entrée) et les actions (décisions en sortie) [Vanthienen et Dries, 1994]. On peut préciser les types de données (nombre, booléen, etc.) et leurs domaines de valeur respectifs (min., max.).

Décision				
U	**Entrées**			**Sortie**
	Donnée d'entrée 1	**Donnée d'entrée 2**	**Donnée d'entrée 3**	**Décision**
1	Valeur 11	Valeur 21	Valeur 31	Valeur A
2	Valeur 12	Valeur 22	Valeur 32	Valeur B
3	Valeur 13	Valeur 23	Valeur 33	Valeur C

Tableau 9 : Table de décision générique (principe)

5.4.2.3 Deux Propriétés Cruciales des Tables de Décision

5.4.2.3.1 Complétude et Cohérence

Bien que cela ne soit pas précisé dans le standard DMN [Silver, 2016], il a deux propriétés cruciales que doivent avoir les bonnes tables de décision : la complétude et la cohérence. Cette exigence est ancienne [Suwa et al., 1982], mais toujours pertinente pour répondre à un reproche récurrent fait aux approches déclaratives (cf. 6.4.2.3 Programmation déclarative).

Tandis qu'il est possible de formaliser les tables de décision avec un simple tableur, les outils spécialisés dans la modélisation des décisions offrent une aide précieuse pour vérifier automatiquement leur complétude et leur cohérence, qui sont deux gages de qualité.

5.4.2.3.2 Complétude d'une Table de Décision

La vérification de la complétude permet d'ajouter des règles implicites afin de couvrir toutes les combinaisons possibles des valeurs des données d'entrée. Il s'agit généralement des règles conduisant à une sortie de type « faux », les textes de spécifications se contentant généralement de décrire les règles (explicites, donc) qui conduisent à une sortie de type « vrai ». Le modélisateur devra souvent prendre quelques initiatives de simplification, afin de limiter le nombre de combinaisons et donc de règles possibles, tout en maintenant l'exhaustivité.

5.4.2.3.3 Cohérence d'une Table de Décision

La vérification de la cohérence détecte un chevauchement éventuel entre deux règles ou plus, où des valeurs identiques en entrée pourraient conduire à des valeurs différentes en sortie. La cohérence est obligatoire pour les tables de décision avec une « politique de succès » (*hit policy*) de type Unique : une et une seule règle doit s'appliquer aux données d'entrée.

5.4.2.4 Politiques de Succès des Tables de Décision

Une politique de succès (*hit policy*) doit être définie pour chaque table de décision

Il est recommandé d'utiliser des tables de décision avec une politique de succès de type **U**nique, où toutes les règles doivent être disjointes. Il existe toutefois 6 autres types de politique de succès :

- simples (Unique, **A**ny, **P**riority ou **F**irst)

ou

- multiples (**O**utput order, **R**ule order ou **C**ollect),

qui ne seront ni détaillés, ni même traduits, afin de conserver leurs lettres initiales, indiquées en haut à gauche de chaque table de décision. Notons toutefois que les politiques de succès **F**irst et **R**ule order, qui dépendent de l'ordre des règles, doivent être évitées, afin de converser le style de langage déclaratif des tables de décision.

5.4.3 Autres éléments de DMN

5.4.3.1 Modèle interchangeable DMN 1.1 XML

Bien que cet élément soit en dernière position, il a toute son importance. En effet, tous les éléments précédents de DMN (diagramme, tables de décision, voire langage FEEL) peuvent être modélisés (on dit aussi « sérialisé ») dans un modèle DMN 1.1 au format XML. Ce modèle DMN 1.1 XML n'est pas seulement décrit dans la spécification publiée par l'OMG, il est également formalisé sous la forme d'un schéma XSD, fourni par l'OMG également, ce qui permet de valider les messages XML générés.

La spécification et le schéma permettent d'échanger les modèles DMN 1.1 XML entre deux outils de modélisation concurrents (caractéristique d'interchangeabilité ; mais la mise en page des diagrammes est souvent modifiée), mais aussi entre un outil de modélisation et un moteur de règles (caractéristique d'interopérabilité ; cette possibilité particulièrement intéressante sera détaillée puis utilisée plus loin).

5.4.4 Complémentarité de DMN avec BPMN

5.4.4.1 Relation forte, mais implicite

Le langage DMN peut s'utiliser seul ou bien en complément du langage BPMN. L'interface entre DMN et BPMN semble particulièrement intéressante [Debevoise et Taylor, 2014]. Les processus métier et les prises de décision peuvent désormais être modélisés séparément, avec leurs propres langages, tout en ayant une relation forte, mais implicite.

Chaque Décision peut être utilisée par un processus ou une tâche, comme indiqué sur l'extrait du métamodèle (Figure 46). En pratique, une tâche concrète de type « Business Rule », comme celle représentée sur la Figure 51, est utilisée pour relier implicitement un diagramme DMN avec un diagramme BPMN.

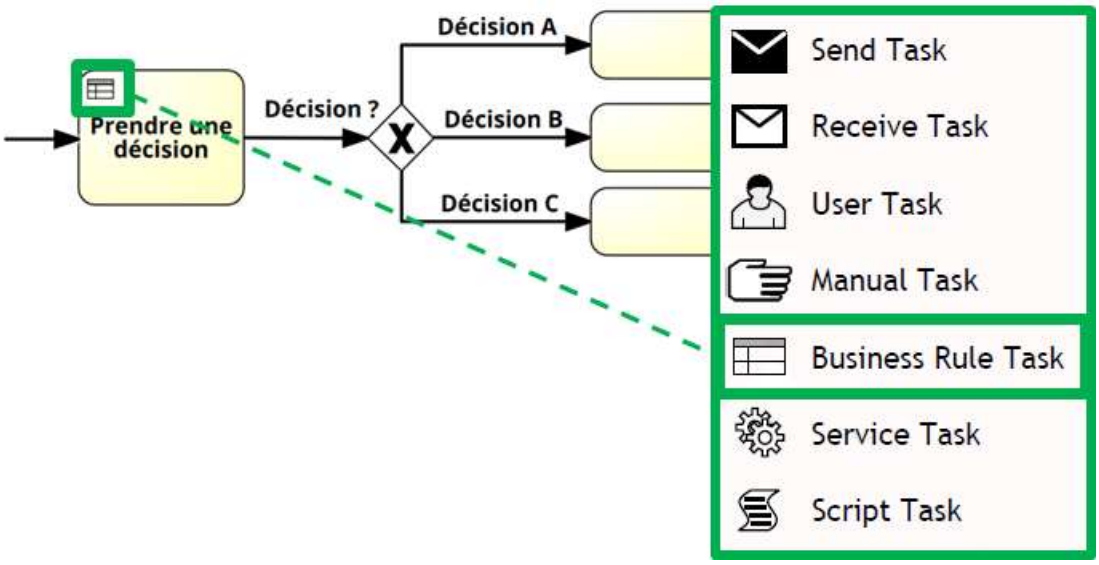


Figure 51 : Tâche de type Business Rule pour relier les diagrammes DMN et BPMN

5.4.4.2 Tâche Business Rules avec branchement

Nous avons démontré dans un précédent article que le langage BPMN se révélait souvent mal adapté pour représenter des prises de décisions complexes [Biard et al., 2015].

Dans un diagramme BPMN, il suffit alors de remplacer une multitude de branchements en cascade par une seule tâche de type Business Rules, symbolisée par une petite table de décision en haut à gauche, pour faire le lien avec un diagramme DMN. La bonne pratique est de nommer à l'identique la tâche du diagramme BPMN et la décision principale du diagramme DMN.

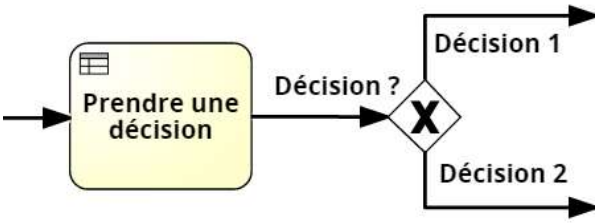


Figure 52 : Tâche de type Business Rules, suivie d'un branchement OU exclusif (BPMN)

5.4.4.3 Tâche Business Rules sans branchement

Généralement, une tâche de type Business Rules est suivie par un branchement du type OU exclusif, comme dans l'extrait de modèle ci-dessus, mais pas toujours. En effet, le langage DMN peut servir à représenter un calcul complexe, comme un pourcentage de remise en fonction de la fidélité d'un client ou le taux d'un emprunt en fonction des risques présentés par l'emprunteur : dans ce cas, la tâche de type Business Rules retourne le résultat du calcul uniquement, qui sera transmis à la tâche suivante directement (pas de branchement), comme l'exemple de la Figure 53.

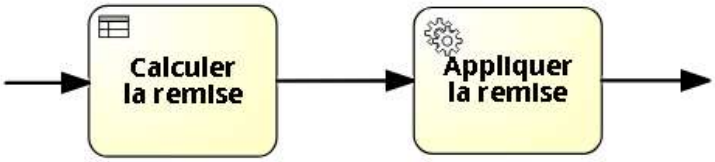


Figure 53 : Tâche de type Business Rules pour représenter un calcul complexe

N.B. Dans ce cas particulier de calcul à effectuer, une politique de succès multiple de type Collect est souvent mieux adaptée qu'une politique de succès simple de type Unique, car elle va permettre par exemple de cumuler les remises.

Les verbes d'action, suivis de compléments d'objets directs, les plus courants pour nommer les tâches de type Business Rules sont Déterminer, Sélectionner, Choisir, Décider, Vérifier, Calculer, Evaluer, Noter.

5.4.5 Contrôle Qualité d'un modèle DMN

Le diagramme DRD sera validé selon le métamodèle DMN. La complétude et la cohérence et des tables de décisions seront vérifiées. Certains outils de modélisation DMN proposent des fonctionnalités de simulation (différente de l'automatisation), qui pourront traiter des jeux de test, afin de contrôler que l'on obtienne bien les résultats attendus.

Comme tous les modèles, on pourra s'assurer que d'autres modélisateurs ont la même compréhension de la représentation proposée. La dernière étape n'est pas des plus faciles : le modèle validé, vérifié et finalement testé devra idéalement être approuvé par le responsable de la décision concernée.

5.4.6 Positionnement de DMN dans la méthode Praxeme

5.4.6.1 Rappel des aspects métier de la méthode Praxeme

Nous avons vu que précédemment la Topologie du Système Entreprise de la méthode Praxeme propose 7 aspects différents. Les 4 aspects amont, que l'on peut qualifier d'architecture métier, sont ceux qui peuvent recevoir les modèles DMN. En fait, l'aspect Géographique n'est pas directement concerné (on peut toutefois imaginer des décisions de géolocalisation) ; il reste donc 3 aspects pour recevoir les éléments des modèles DMN : Intentionnel, Sémantique et Pragmatique.

Lors de la lecture des deux chapitres suivants, il pourrait paraitre étrange au premier abord de vouloir positionner les Diagrammes DRD et les Règles métier dans des aspects différents : il s'agit encore une fois de l'application du principe de Séparation des Préoccupations. Les Diagrammes DRD et les Règles métier peuvent être réalisés par des personnes aux compétences différentes. Et surtout, les Règles métier peuvent être modifiées sans toucher aux Diagrammes DRD (surtout si l'on ne modifie pas les données d'entrée).

5.4.6.2 Positionnement des Diagrammes DRD

Le positionnement le plus naturel des diagrammes DMN, représentant la structure des prises de décision, serait au plus près des diagrammes BPMN, c.-à-d. dans l'aspect Pragmatique qui présente les processus métier. Par contre, cette approche semble trop simpliste pour positionner les tables de décision, car elle ne permettrait de représenter qu'une partie des règles métier.

Comme il est possible également d'utiliser DMN seul, on pourrait le positionner dans l'aspect Sémantique également. Par contre, DMN qui s'appuie sur des Objets Métier, ne semble pas avoir sa place directement dans l'aspect Intentionnel. Ce positionnement reste à affiner.

5.4.6.3 Traitement et positionnement des Règles métier

La chaine de traitement des règles métier peut être décomposée en plusieurs étapes : collecte des règles métier utilisées dans l'Entreprise et exprimées en langage naturel ; analyse ; qualification (selon leur légitimité et le niveau de contrainte qu'elles imposent) ; reformulation et enfin typage, pour les positionner dans les aspects adéquats de la Topologie du Système Entreprise, avant de les formaliser.

Les règles dites de gestion agissant sur les objets métier seront positionnées sur l'aspect Sémantique, tandis que les règles agissant sur les organisations seront positionnées sur l'aspect Pragmatique. Le respect de la règlementation introduit des exigences, type de règles particulières, obligatoires et contraignantes, à formaliser également.

La couverture complète du domaine métier par ces différentes règles devra être approchée, tout en évitant les recouvrements, et en garantissant la cohérence d'ensemble. La technique de vérification de l'exhaustivité des cas d'utilisation UML, basée sur l'étude du cycle de vie des objets, pourrait être transposée aux règles métier.

6 Validation de la Solution

6.1 PoC (Proof of Concept)

L'aspect de ce langage DMN, qui ne comporte que quatre éléments graphiques principaux, semble trop simple de prime abord pour permettre l'automatisation des prises de décisions qui ont été modélisées. Le but de cette Proof of Concept est de démontrer qu'au contraire, il est possible d'automatiser les prises de décisions qui ont été modélisées avec le langage DMN.

A noter que notre Proof of Concept utilisait une nouvelle version béta du moteur de règles, avec documentation provisoire et incomplète, qu'il s'agissait d'alimenter avec un modèle conçu avec l'outil d'un autre éditeur. Malgré un format d'interchange standardisé, quelques ajustements furent nécessaires de part et d'autre… Cette Proof of Concept a donc eu une portée globale.

6.2 Etude de cas

6.2.1 DRD (Decisions Requirements Diagram)

6.2.1.1 Relation implicite avec le diagramme BPMN

Dans l'étude de cas qui illustre ce mémoire, il s'agit donc, dans un établissement public d'enseignement supérieur, de décider de la recevabilité (ou non) d'un vacataire lors du processus de recrutement, selon 11 critères différents.

Le sous-processus modélisé sur la Figure 28 précédente en BPMN exclusivement est ici remplacé par une tâche de type règle métier :

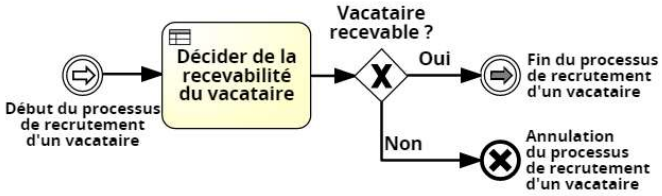


Figure 54 : Extrait du processus de recrutement avec une tâche Règle métier

Avec un outil de modélisation BPMN supportant également DMN, comme Signavio Process Manager [Signavio, 2017a], il suffit alors de cliquer sur la minuscule table de décision pour lancer l'outil de modélisation DMN et afficher le modèle DMN associé (*Figure 55* suivante).

Une recommandation à suivre est que la décision principale dans un diagramme DMN porte le même « nom » (généralement un verbe d'action, suivi d'un complément d'objet direct) que la tâche qui l'appelle dans un diagramme BPMN, car la relation entre les deux diagrammes DMN et BPMN est implicite (pas de lien graphique entre les deux).

6.2.1.2 Représentation en notation DMN retenue

Le DRD représenté par la *Figure 55* reste encore simple. Le choix a été fait de représenter chaque critère comme une donnée d'entrée. Cela permet de voir au premier coup d'œil que les critères sont complètement différents, selon qu'il s'agit d'un poste de chargé d'enseignement ou d'un poste d'agent temporaire. Dans ce mode de représentation, chaque donnée d'entrée dans le diagramme DMN correspond à une question associée à un branchement OU exclusif dans le diagramme BPMN (Figure 30).

Un autre choix de modélisation aurait pu être de représenter uniquement le vacataire comme seule donnée d'entrée, les 11 critères étant alors considérés comme ses attributs.

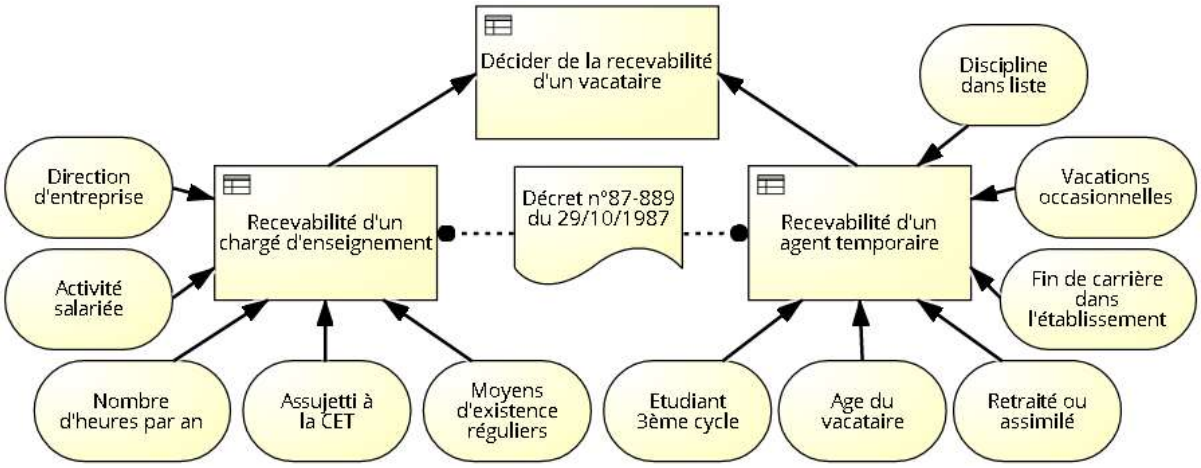


Figure 55 : Diagramme (DRD) "Décider de la recevabilité d'un vacataire"

On voit sur cet exemple que la décision principale « Décider de la recevabilité d'un vacataire », représentée par convention tout en haut du DRD, est constituée de deux décisions secondaires « Recevabilité d'un chargé d'enseignement » OU « Recevabilité d'un agent temporaire ».

Le Décret n°87-889 est considéré comme Source de Connaissance pour chacune des décisions secondaires. Les règles métier, qu'il convient d'appliquer pour décider de la recevabilité d'un vacataire, proviennent de ce document de référence.

Par contre, cette étude de cas n'utilise pas de Modèle de connaissance métier, car la logique de décision est ici très spécifique et ne pourrait pas être réutilisée ailleurs. Un Modèle de connaissance métier s'utile comme une fonction, que l'on appelle avec des paramètres spécifiques à chaque contexte. Son objectif premier est la réutilisation.

6.2.1.3 Alternatives de représentation possibles

Comme souvent, la forme de représentation du modèle dépend des choix effectués par le modélisateur. Aussi, différentes variantes de représentation restent possibles.

Il aurait été possible techniquement de se passer de ces deux sous-décisions et d'associer les 11 données d'entrées directement à la décision principale, mais la table de décision aurait été plus compliquée et bien moins lisible.

L'utilisation des sous-décisions est un bon moyen pour représenter la complexité de la logique de décision à modéliser. Dès qu'il y a plus de 5 données d'entrée (environ), il convient de se demander si des sous-décisions seraient utiles pour faciliter la compréhension du modèle.

S'il s'agit de réutiliser une logique de décision dans plusieurs modèles, l'usage de l'élément graphique Business Knowledge Model, qui est prévu pour cela, nous semble préférable à celui d'une sous-décision réutilisée plusieurs fois (forme de redondance).

Tandis que la décision principale DMN est nommée comme la tâche BPMN de type Business Rule qui l'attend, afin de faciliter la relation implicite entre les deux diagrammes (bonne pratique évoquée précédemment), le nommage des sous-décisions est plus ouvert. Ainsi, le nom d'une sous-décision peut être directement le nom de sa donnée de sortie, ce qui nous semble assez clair et lisible. Exemples : Eligibilité, Applicabilité, Score, Classement (ou Recevabilité comme dans notre étude de cas).

Une autre alternative aurait été de représenter seulement le candidat vacataire comme unique donnée d'entrée, les différents critères détaillés devenant alors des attributs de cette donnée principale (Exemple : Vacataire.Age ; en pratique, il est préférable d'utiliser la date de naissance – invariable – comme attribut). Cette représentation n'aura pas fait ressortir le fait que les critères de recevabilité des chargés d'enseignement et des agents temporaires sont complètement différents (Etait-ce vraiment la volonté du législateur ?).

Des données d'entrée représentées par des objets élémentaires de différents types (textes, nombres, booléens, etc.) qui partagent le même contexte peuvent être regroupées pour composer un seul objet de type complexe. Exemple : l'objet complexe Adresse est composé d'objets élémentaires comme Numéro (nombre), Rue (texte), CodePostal (nombre) et Ville (texte). A noter que DMN propose également le type Enumération, ce qui permet de limiter les données d'entrée à un ensemble fini de valeurs. Ainsi le CodePostal et la Ville pourraient être du type Enumération, afin de n'autoriser que des codes postaux et des villes qui existent bien.

Ces deux sous-décisions et leurs tables de décision sont ici alignées sur les deux articles distincts du décret de référence qui fixent respectivement les critères de recevabilité des chargés d'enseignement et des agents temporaires. D'ailleurs, au lieu de représenter une seule Knowledge Source faisant par deux fois référence au décret en entier, nous aurions pu représenter deux Knowledge Sources distinctes faisant chacune référence à l'article concerné.

6.2.2 Tables de décision

La logique de chaque décision et sous-décision est représentée par une table de décision. Les règles métier sont listées sous forme de lignes, dans le même ordre que les articles du décret. Les données en entrée sont listées sous forme de colonnes, dans le même ordre que celui des articles du décret également. Comme nous l'avons expliqué précédemment (DMN est un langage de style déclaratif), cet ordre est arbitraire, mais facilite grandement la vérification de la bonne interprétation des règles métier originelles, souvent exprimées par du texte.

Nommer les données d'entrée avec les mêmes termes que ceux employés dans la Source de Connaissance, sous réserve que ceux-ci soient déjà utilisés dans l'Entreprise ou au moins bien compris, garantit leur uniformité et leur traçabilité à l'intérieur des règles métier. Dans le cas contraire, un glossaire des termes utilisés avec leurs synonymes s'avérera très utile.

Les valeurs des entrées dans une table de décision DMN correspondent aux valeurs sur les branches en sortie de branchements d'un diagramme BPMN.

La logique pour décider de la recevabilité d'un chargé d'enseignement est représentée dans le Tableau 10, en fonction de 5 critères en entrée (4 booléens et 1 nombre) :

Recevabilité d'un chargé d'enseignement						
U	**Entrées**					**Sortie**
	Direction entreprise	**Activité salariée**	**Nombre d'heures par an**	**Assujetti à la CET**	**Moyens existence réguliers**	**Recevabilité d'un chargé enseignement**
	Booléen	Booléen	Nombre	Booléen	Booléen	Booléen
1	= vrai	-	-	-	-	vrai
2	= faux	= vrai	≥ 900	-	-	vrai
3	= faux	= vrai	< 900	-	-	faux
4	= faux	= faux	-	= vrai	-	vrai
5	= faux	= faux	-	= faux	= vrai	vrai
6	= faux	= faux	-	= faux	= faux	faux

Tableau 10: Table de décision "Recevabilité d'un chargé d'enseignement"

La logique pour décider de la recevabilité d'un agent temporaire est représentée par une table de décision similaire, avec des critères en entrée différents, bien sûr. Le *Tableau 11* contient 6 critères en entrée (5 booléens et 1 nombre) :

Recevabilité d'un agent temporaire							
U	**Entrées**						**Sortie**
	Etudiant 3e cycle	**Age du vacataire**	**Retraité ou assimilé**	**Fin de carrière établis.**	**Vacations occasion.**	**Discipline dans liste**	**Recevabilité d'un agent temporaire**
	Booléen	Nombre	Booléen	Booléen	Booléen	Booléen	Booléen
1	= vrai	-	-	-	-	-	vrai
2	= faux	< 67	= vrai	= faux	= faux	= vrai	vrai
3	= faux	< 67	= vrai	= faux	= faux	= faux	faux
4	= faux	< 67	= vrai	= faux	= vrai	-	vrai
5	= faux	< 67	= vrai	= vrai	-	-	faux
6	= faux	< 67	= faux	-	-	-	faux
7	= faux	≥ 67	-	-	-	-	faux

Tableau 11: Table de décision "Recevabilité d'un agent temporaire"

Il convient de préciser que le tiret « - » comme valeur d'entrée signifie qu'elle est « sans importance » et évite de représenter toutes les combinaisons possibles (*any*) dans une table de décision. Toute démarche de simplification est toutefois discutable ; par exemple, un étudiant de 3ème cycle, mais âgé de plus de 67 ans, est-il vraiment recevable ? Ce n'était sans doute pas la volonté du législateur, mais telle est l'interprétation fidèle du décret.

Les deux tables de décision présentées sur cette page sont cohérentes et complètes. Mais, comme tous les modèles, il faut s'assurer qu'elles représentent bien la réalité, c'est la préoccupation principale du modélisateur. Ou bien il faut aligner la réalité sur ces tables, c'est alors une décision tactique du manager. Car cet exercice de modélisation des décisions et de formalisation des règles dans des tables révèle souvent des différences entre théorie et pratique.

Quant à la logique de décision principale de recevabilité d'un vacataire, il s'agit d'une table plus simple représentant un OU logique des deux décisions secondaires précédentes. Le vacataire recevable peut être chargé d'enseignement ou agent temporaire, selon les cours.

Décider de la recevabilité d'un vacataire			
U	Entrées		Sortie
	Recevabilité d'un chargé d'enseignement	Recevabilité d'un agent temporaire	Décider de la recevabilité du vacataire
	Booléen	Booléen	Booléen
1	= faux	= faux	faux
2	= faux	= vrai	vrai
3	= vrai	= faux	vrai
4	= vrai	= vrai	vrai

Tableau 12: Table de décision "Décider de la recevabilité d'un vacataire"

On voit clairement dans le *Tableau 12* que les sorties des deux décisions secondaires deviennent les entrées de cette décision principale. Cette table de décision très simple aurait pu être remplacée par une ligne de code en langage FEEL ; c'eut été plus compact, mais moins cohérent avec les représentations par tables de décisions des deux décisions secondaires.

La lettre « U » (comme Unique) positionnée en haut et à gauche de ces trois tables de décision signifie qu'il ne doit pas avoir de recouvrement entre chacune des règles. Unique est la politique de succès (*hit policy*) qui est recommandée pour DMN : pour chaque jeu différent de données d'entrée, une seule règle doit s'appliquer.

6.3 Méthode pour comparer les solutions

6.3.1 MDA (Model Driven Architecture)

La Model Driven Architecture (MDA) ou l'Architecture Dirigée par les Modèles est un concept proposé par l'OMG également [OMG, 2014] pour concevoir un grand système, complexe par nature. Ce concept est indépendant de tout langage ou notation. La MDA applique le principe de Séparation des Préoccupations.

6.3.2 Trois modèles CIM, PIM & PSM

La MDA, grâce à ses trois modèles CIM, PIM et PSM à trois niveaux, va nous fournir la grille de lecture adéquate pour comprendre, voire comparer, puis mettre en œuvre trois solutions techniques différentes, que nous avons nommées spécifique et générique, pour automatiser les prises de décisions modélisées selon le langage DMN. Le *Tableau 13* rassemble ces 3 modèles :

Trois modèles MDA	Description
CIM (Computation Independent Model)	Modèle de représentation du métier, indépendant de toute considération informatique
PIM (Platform Independent Model)	Modèle de conception pour l'informatique, indépendant de la plate-forme d'exécution
PSM (Platform Specific Model)	Modèle de conception pour l'informatique, spécifique à la plate-forme d'exécution

Tableau 13 : Les 3 modèles MDA : CIM, PIM & PSM

6.3.3 Projection de DMN sur les modèles MDA

Si l'on projette les éléments principaux de DMN présentés précédemment sur les deux premiers modèles MDA, et que l'on complète le troisième niveau avec la plate-forme cible choisie Drools [Red Hat, 2017], le plus connu des BRMS (Business Rules Management System ; contient entre autres un moteur de règles), on obtient assez logiquement le *Tableau 14* ci-après.

Trois modèles MDA	Principaux éléments de DMN en théorie
CIM (Computation Independent Model)	DRD (Decision Requirements Diagram) + Tables de décision
PIM (Platform Independent Model)	DMN 1.1 XML (Modèle interchangeable) contenant éventuellement du langage FEEL
PSM (Platform Specific Model)	DRL (Drools Rule Language)

Tableau 14 : Projection des principaux éléments de DMN sur les modèles MDA

Il est bien entendu très intéressant d'utiliser le modèle CIM : c'est la valeur ajoutée de DMN. Nous verrons par la suite, lors de la réalisation du démonstrateur, que les solutions techniques retenues n'utilisent pas simultanément les modèles PIM « ET » PSM, mais plutôt les modèles PIM « OU » PSM.

6.4 Démonstrateur pour l'automatisation des décisions

6.4.1 Composants du démonstrateur

6.4.1.1 Deux outils principaux

Notre démonstrateur est constitué essentiellement de deux outils :

- Premier outil : le modeleur DMN (pour la modélisation),
- Deuxième outil : le moteur de règles (pour l'automatisation).

D'autres outils d'une importance moindre se sont révélés néanmoins indispensables.

Ces différents outils sont présentés ci-dessous. La présentation du modeleur DMN, qui fut d'abord construit sur mesure, puis remplacé par un outil standard, est plus détaillée.

6.4.1.2 Modeleur DMN sur mesure

Au début de cette thèse, en 2014, il n'y avait pas encore d'outils de modélisation DMN standards. Nous avons donc réalisé notre propre outil sur mesure dans l'environnement de développement intégré Eclipse, avec le module Sirius [Eclipse, 2013] qui permet de créer des éditeurs graphiques les domaines métier spécifiques. La représentation graphique était sommaire (les formes des éléments de la notation DMN étaient remplacées par des couleurs et l'indication explicite des types d'éléments), mais le résultat était néanmoins conforme au métamodèle DMN fourni par l'OMG.

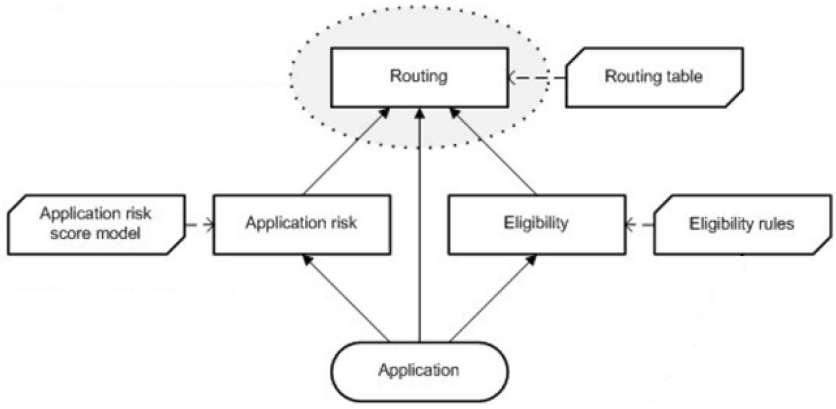


Figure 56 : Exemple de diagramme DMN, extrait de la spéc. V1.0 bêta de l'OMG

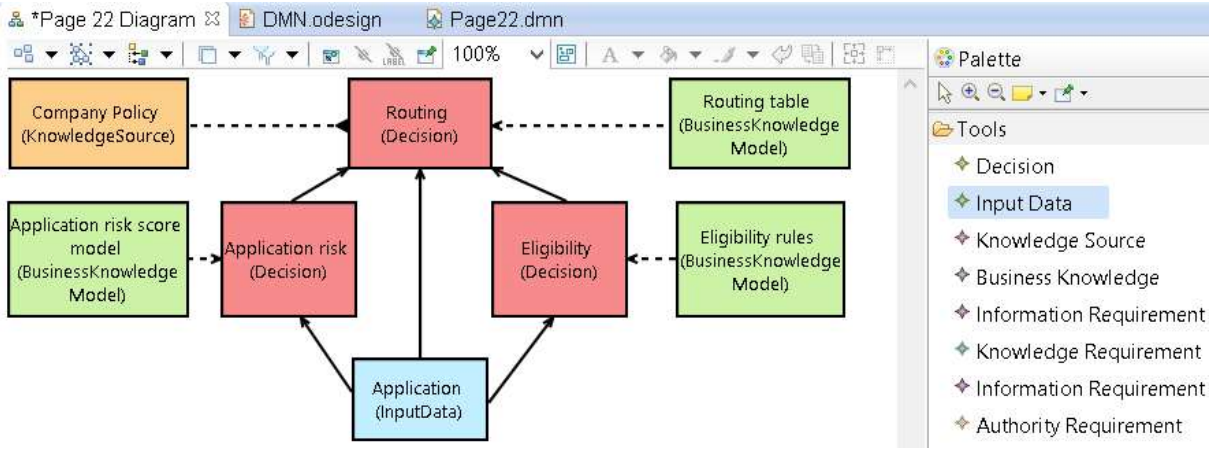


Figure 57 : Même exemple, réalisé avec notre modeleur DMN sur mesure (Sirius)

Les difficultés à surmonter avant d'arriver à ce résultat furent multiples. Ainsi, le métamodèle DMN fourni par l'OMG est conforme au métamétamodèle MOF (Meta-Object Facility), tandis que le métamodèle utilisé par le modeleur Sirius dans l'environnement Eclipse doit être conforme au métamétamodèle Ecore (Eclipse Modeling Framework). La conversion du métamodèle DMN version 1.0 Beta du format MOF vers le format Ecore a mis en évidence de nombreuses erreurs de syntaxe (par exemple, une classe sans nom !), qu'il a fallu corriger.

Les relations "Requirements" explicites (c.-à-d. matérialisées par des classes dans le métamodèle DMN, cf. *Figure 50*) entre les éléments graphiques principaux nous ont posé problème également (généralement, les relations entre éléments graphiques sont implicites).

Finalement, cette réalisation sur mesure a mis en évidence des lacunes structurelles dans métamodèle DMN version 1.0 bêta, qui subsistèrent dans la version 1.0 finale, mais qui furent enfin corrigées dans la version 1.1, qui sert de base à tous les modeleurs DMN standards.

Néanmoins, elle nous permit d'avancer jusqu'à la génération automatique de code FEEL à partir du diagramme DRD, afin de pouvoir envisager l'automatisation ultérieure des prises de décisions [Biard et al., 2015]. Le module Acceleo pour l'environnement Eclipse [Eclipse, 2006] fut nécessaire, afin de transformer le modèle DMN en texte, en l'occurrence du code FEEL, conforme à la spécification DMN. Ce module implémente le langage de transformation standard de l'OMG M2T (Model-To-Text ; [OMG, 2008]).

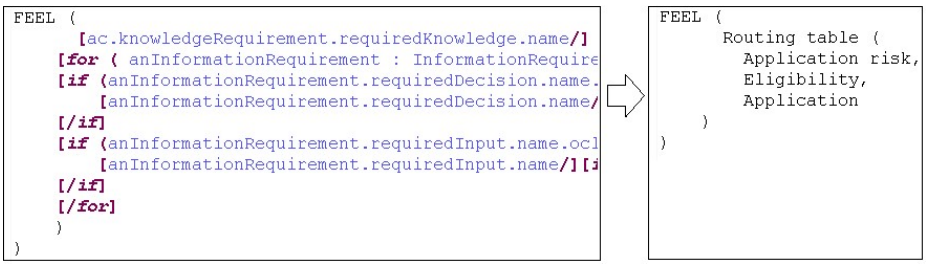


Figure 58 : Transformation Model-To-Text (FEEL) avec le module Acceleo

6.4.1.3 Modeleur DMN standard

Entre temps, les éditeurs spécialisés dans les outils de modélisation se sont intéressés également à DMN et, avec des équipes de développeurs, ont proposé des outils standards et plus puissants que notre outil sur mesure (par exemple, pour éditer des tables de décisions, mais surtout pour contrôler leur complétude et leur cohérence). Aussi nous a-t-il semblé plus raisonnable de continuer nos expérimentations avec un outil de modélisation standard.

Notre choix s'est porté rapidement sur :

- Signavio Process Manager version 11.1.0 [Signavio, 2017a] pour la modélisation en BPMN des processus métier et en DMN des prises de décision (diagrammes et tables de décision) ; cet outil fonctionne entièrement en ligne en mode SaaS et ne requiert donc qu'un navigateur web ; il est de bonne facture et gratuit pour le monde académique.

6.4.1.4 Moteur de règles

Notre choix d'un moteur de règles fut encore plus facile :

- Drools (version 7.0.0.Final) pour l'automatisation des prises de décisions ; il s'agit du BRMS (Business Rules Management System) de référence, qui existe en version gratuite (open source) ou payante (sous le nom de Red Hat JBoss BRMS), proposé seul ou intégré dans des logiciels plus complets ; Drools est utilisé ici sous la forme d'un plug-in pour Eclipse. Nous utiliserons uniquement le moteur de règles métier - Business Rules Engine (BRE) - de Drools.

La Figure 59 présente les trois solutions techniques qui seront mises en œuvre, puis comparées, dans les chapitres suivants :

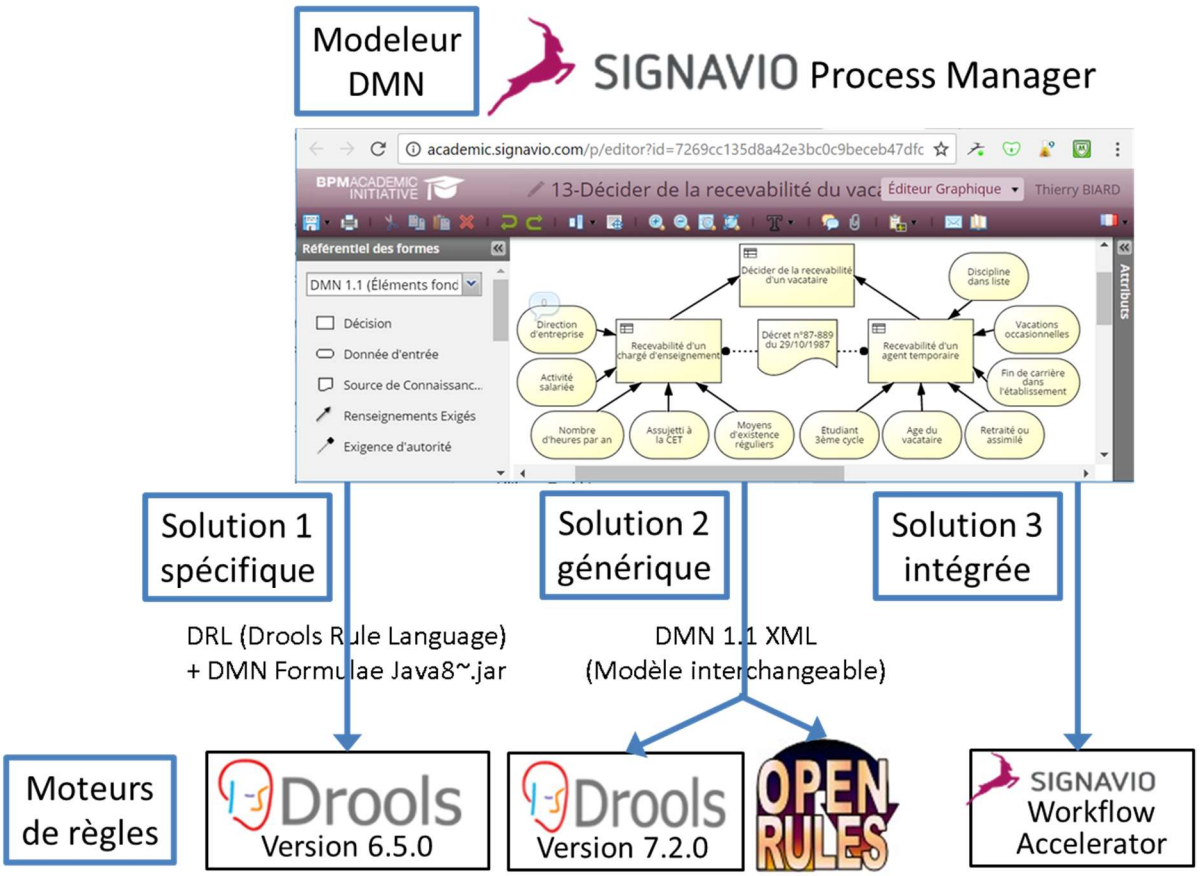


Figure 59 : 3 solutions pour l'automatisation des prises de décisions DMN

6.4.1.5 Autres outils

L'environnement de développement intégré Eclipse (version neon.3 ; Modeling Tools package) a donc été utilisé pour faire fonctionner Drools. Il convient d'installer préalablement le JDK (Java Development Kit). D'autres outils comme Git pour la gestion des versions et Maven pour la gestion des dépendances (sous la forme du plug-in intégré M2Eclipse) s'avèrent également utiles.

Et enfin, Drools fonctionnant en mode boîte noire (c.-à-d. sans aucun message) par défaut, l'ajout de la bibliothèque de fonctions de logging LOGBack (*core* plus *classic*) nous a paru indispensable pour mettre au point nos programmes, y compris ceux qui fonctionnent du premier coup, mais qui restent désespérément muets sans cette bibliothèque.

6.4.2 Paradigmes de programmation

6.4.2.1 Changement de paradigme de programmation

Un moteur de règles comme Drools ne propose pas moins qu'un changement de paradigme de programmation, la programmation déclarative [Van Roy et Haridi, 2004], à la place de la programmation impérative (ou séquentielle) utilisée par la plupart des langages informatiques traditionnels. Aussi est-il intéressant de s'y attarder pour bien comprendre la différence entre ces deux paradigmes.

Voyons un exemple pour chacun des deux types de paradigmes de programmation qui nous intéressent dans ce mémoire (N.B. Il existe d'autres types de paradigmes, la programmation événementielle par exemple).

6.4.2.2 Programmation impérative

```
for each objet in mesDonnees do
   if une première condition est détectée
      then une première action est faite
      else
         if une deuxième condition est détectée
            then une deuxième action est faite
            else une troisième action est faite
         endif
   endif
endfor
```

Algorithme 1 : Programmation impérative avec "if...then…else"

Dans la programmation impérative, les instructions « if…then…else » sont souvent imbriquées et rendent le développement, le test et les changements compliqués.

6.4.2.3 Programmation déclarative

```
rule "une"
   when une première condition est détectée
   then une première action est faite
end
```

```
rule "deux"
   when une deuxième condition est détectée
   then une deuxième action est faite
end
```

```
rule "trois"
   when une troisième condition est détectée
   then une troisième action est faite
end
```

Algorithme 2 : Programmation déclarative avec "when...then"

Avec la programmation déclarative, il n'y plus d'instructions « if…then…else » imbriquées, mais uniquement un « when…else » pour chaque règle métier, que l'on peut facilement ajouter, modifier ou supprimer en toute indépendance des autres règles.

Un autre avantage majeur est que l'ordre des règles métier n'est pas important (si la politique de succès est de type Unique). Même la boucle « for each » pour prendre en compte tous les objets concernés par les règles métier n'est plus utile en programmation déclarative.

6.4.3 Première solution spécifique : du CIM au PSM (sans PIM)

6.4.3.1 Principe de la solution spécifique

Depuis 2016, Signavio Process Manager permet de générer directement un fichier au format DRL (Drools Rule Language), c'est-à-dire du code informatique spécifique à une plate-forme (PSM), à partir d'un diagramme DMN et des tables de décision (CIM) qui lui sont associés. Cette première solution spécifique n'utilise pas le modèle PIM de niveau intermédiaire.

Le fichier DRL (800 lignes de code) utilise, en plus des bibliothèques standards de Java, une bibliothèque de fonctions spécifiques à Signavio, fournie sous la forme d'un fichier « DMN Formulae Java8-1.0-SNAPSHOT.jar ».

Trois modèles MDA	**Drools version 6 (solution 1)**
CIM (Computation Independent Model)	**DRD (Decision Requirements Diagram) + Tables de décision**
PIM (Platform Independent Model)	✘
PSM (Platform Specific Model)	**DRL (Drools Rule Language) + DMN Formulae Java8-1.0-SNAPSHOT.jar**

Tableau 15 : Projection des éléments de la première solution sur les modèles MDA

6.4.3.2 Déclaration simplifiée des données

Le challenge était de pouvoir utiliser les fichiers au format DRL tels qu'ils sont générés par le modeleur, sans aucune modification, afin qu'un changement de règle puisse être répercuté facilement. Seuls les caractères accentués de la langue française posent problème, car ils sont interdits dans la plupart des langages de programmation et en l'occurrence supprimés lors de la génération des fichiers au format DRL, rendant la compréhension des noms des éléments compliquée. Un diagramme et ses tables de décision modifiés, sans aucun caractère accentué, ont donc été nécessaires. Le langage DRL déclare ensuite les Données d'entrée de façon très simple, grâce à la directive *declare*.

```
1: declare Input
2:    directionEntreprise : Boolean
3:    activiteSalariee: Boolean
4:    nombreHeuresParAn: BigDecimal
5:    assujettiCet : Boolean
6:    moyensExistenceReguliers: Boolean
7: end
```

Algorithme 3 : Déclaration des Données d'entrée simplifiée avec le langage DRL

6.4.3.3 Déclaration des règles métier

Les vraies règles métier de notre étude de cas s'avèrent assez verbeuses et peu lisibles, comme c'est souvent le cas du code informatique généré automatiquement. L'exemple ci-après (Algorithme 4) a été simplifié (suppression de préfixes), afin que les lignes aient une longueur raisonnable dans ce mémoire.

```
1: rule "recevabiliteChargeEnseignement_rule_2"
2:   when
3:     eval(nullSafeEval(equals(getDirectionEntreprise(),false)))
4:     eval(nullSafeEval(equals(getActiviteSalariee(),true)))
5:     eval(nullSafeEval(greaterThanOrEqualTo(getNombreHeuresParAn(),
                                          BigDecimal.valueOf(900.0))))
6:   then
7:     RecevabiliteChargeEnseignement $recevabiliteChargeEnseignement
                                  = new RecevabiliteChargeEnseignement();
8:     $recevabiliteChargeEnseignement.
                                 setRecevabiliteChargeEnseignement(true);
9:     insert($recevabiliteChargeEnseignement);
10:end
```

Algorithme 4 : Exemple de code DRL simplifié pour la règle n°2 du Tableau 1

Dans la section *when*, les Données d'Entrée sont évaluées en lignes 3, 4 et 5. Dans la section *then*, si les critères de décision sont respectés, la Décision RecevabiliteChargeEnseignement devient vraie en ligne 8. A noter que cette Décision est insérée dans la base de faits en ligne 9, car il s'agit d'une sous-décision pour la Décision principale DeciderRecevabilitéVacataire et sa valeur sera utilisée ultérieurement dans une autre règle.

La Figure 60 met en évidence la correspondance entre la table de décision et le code DRL, cette fois-ci complet (c.-à-d. non simplifié) :

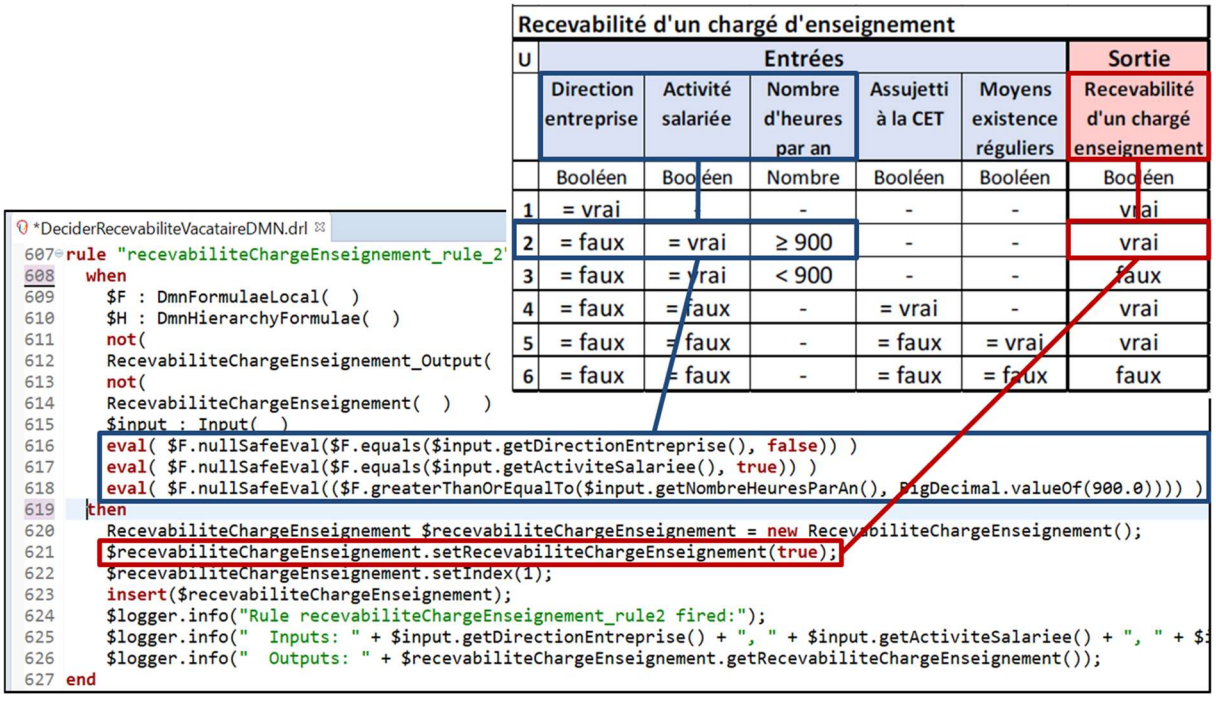


Figure 60 : Correspondance entre table de décision et code DRL

6.4.3.4 Application des règles métier

Afin d'appliquer toutes les règles métier générées automatiquement (ligne 09), il convient d'écrire un programme en langage Java, qui va d'abord créer (ligne 02) puis initialiser (lignes 03 à 07) un objet de type vacataire, puis l'insérer (ligne 08) parmi les faits de la session Drools. Cette partie du programme est relativement simple et lisible.

Ce n'est pas le cas du code qui suit (lignes 10 à 13), nécessaire à la récupération du résultat (la décision *output*). Ces quatre lignes de codes ont nécessité beaucoup d'essais, notamment à cause du typage spécifique des objets.

```
01: FactType vacataireType =
       kieBase.getFactType("com.signavio.droolsexport.mon_modele_dmn",
          "Input");
02: Object thierry = vacataireType.newInstance();
03: vacataireType.set(thierry, "directionEntreprise", true);
04: vacataireType.set(thierry, "activiteSalariee", false);
05: vacataireType.set(thierry, "nombreHeuresParAn",
       new BigDecimal("800"));
06: vacataireType.set(thierry, "assujettiCet", true);
07: vacataireType.set(thierry, "moyensExistenceReguliers", true);
08: kieSession.insert(thierry);
09: kieSession.fireAllRules();

10: FactType outputType =
       kieBase.getFactType("com.signavio.droolsexport.mon_modele_dmn",
          "DeciderRecevabiliteVacataire_Output");
11: Collection<?> outputObjects = kieSession.getObjects(new
       ClassObjectFilter(outputType.getFactClass()));
12: Object outputObject = outputObjects.iterator().next();
13: boolean output = (boolean) outputType.get(outputObject,
       "deciderRecevabiliteVacataire");
14: System.out.println("Recevabilité du vacataire = " + output);
```

Algorithme 5 : Programme Java d'application des règles, première solution (extrait)

Quant à l'affichage du résultat (la décision *output* en ligne 14), celui-ci s'avère peu spectaculaire dans notre étude de cas, puisqu'il s'agit d'une simple variable booléenne qui représente la recevabilité du vacataire lors de son recrutement : *true* ou *false* :

```
Recevabilité du vacataire = true
```

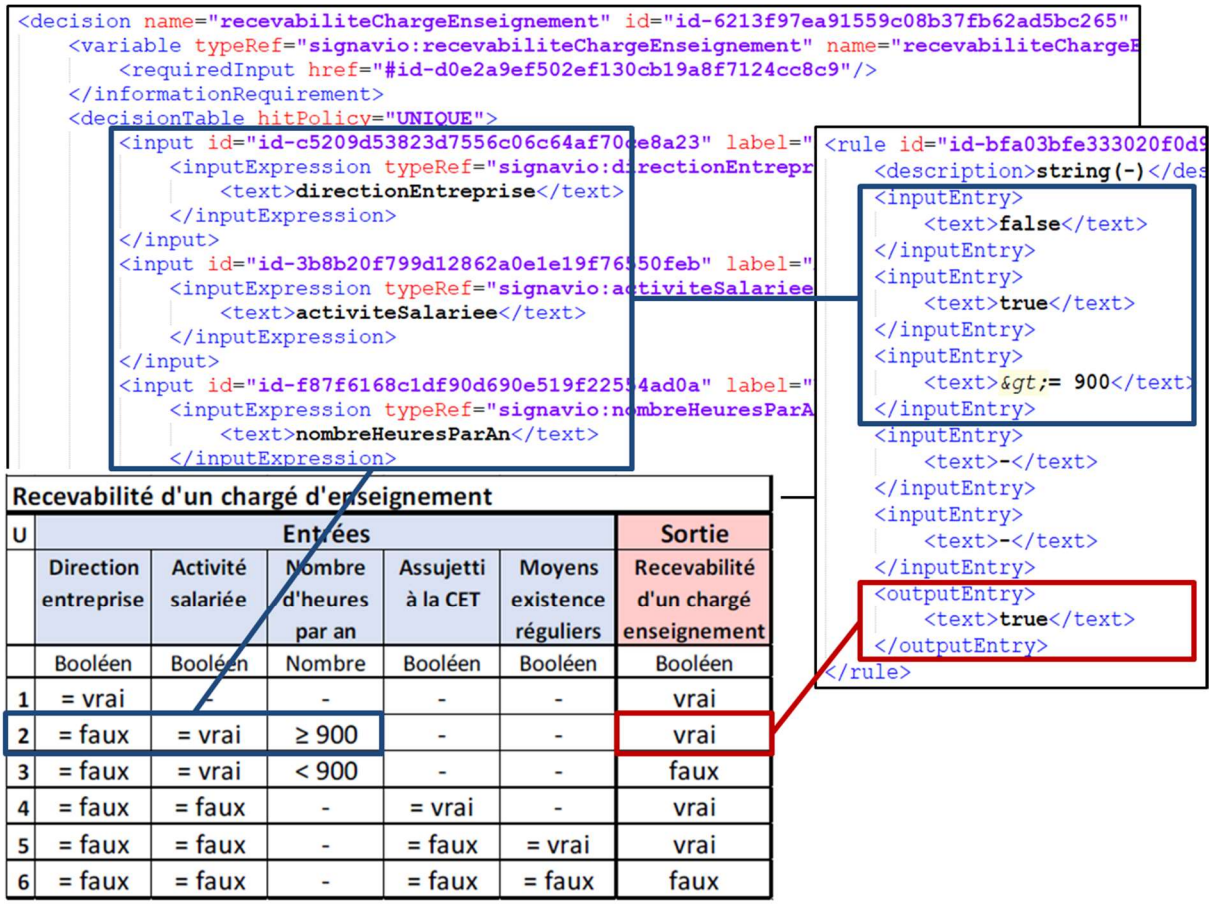
La valeur de cette variable booléenne sera bien entendu récupérée automatiquement par le système BPM (Business Process Management) pour déterminer le bon chemin à prendre, c.-à-d. pour suivre le flux de séquence adéquat, dans le diagramme représentant le processus métier.

6.4.4 Deuxième solution générique : du CIM au PIM (sans PSM)

6.4.4.1 Principe de la solution générique

Depuis janvier 2017, Signavio Process Manager permet également de générer un fichier XML au format DMN version 1.1 (appelé « DMN 1.1 XML »), toujours à partir d'un diagramme DMN et des tables de décision (CIM) qui lui sont associés. « DMN 1.1 XML » est un format d'interchange défini dans la spécification DMN. Un schéma XSD est fourni par l'OMG, permettant de valider le fichier XML généré. Il s'agit donc d'un PIM, car il est encore indépendant de toute plate-forme.

La Figure 61 met en évidence la correspondance entre la table de décision et le code XML généré automatiquement, à l'intention d'un programme informatique et non d'un être humain :



Recevabilité d'un chargé d'enseignement

U	Entrées					Sortie
	Direction entreprise	Activité salariée	Nombre d'heures par an	Assujetti à la CET	Moyens existence réguliers	Recevabilité d'un chargé enseignement
	Booléen	Booléen	Nombre	Booléen	Booléen	Booléen
1	= vrai	-	-	-	-	vrai
2	= faux	= vrai	≥ 900	-	-	vrai
3	= faux	= vrai	< 900	-	-	faux
4	= faux	= faux	-	= vrai	-	vrai
5	= faux	= faux	-	= faux	= vrai	vrai
6	= faux	= faux	-	= faux	= faux	faux

Figure 61 : Correspondance entre table de décision et code en langage XML

Le but originel de ce format d'interchange est de pouvoir échanger des modèles de prise de décision entre des outils de modélisation différents. Ce fichier XML dit sérialisé définit tous les éléments (y compris des métadonnées) nécessaires à la représentation du diagramme et des tables de décision. Ce format d'interchange est également capable de contenir du langage FEEL (qui lui n'est pas suffisant pour tout définir).

Mais la dernière version 7.0 du moteur de règles Drools est également capable d'interpréter directement le format « DMN 1.1 XML » pour automatiser les prises de décision : la génération de code spécifique à une plate-forme (PSM) devient alors superflue. Cette nouvelle approche est résumée dans le *Tableau 16*.

Trois modèles MDA	Drools version 7 (solution 2)
CIM (Computation Independent Model)	DRD (Decision Requirements Diagram) + Tables de décision
PIM (Platform Independent Model)	DMN 1.1 XML (Modèle interchangeable) contenant éventuellement du langage FEEL
PSM (Platform Specific Model)	✘

Tableau 16 : Projection des éléments de la deuxième solution sur les modèles MDA

Quelques autres outils spécialisés dans la modélisation et l'automatisation des prises de décisions sont également capables d'interpréter directement le format « DMN 1.1 XML », comme OpenRules [Feldman, 2017] représenté sur la Figure 59.

6.4.4.2 Application des règles métier

Le premier fichier « DMN 1.1 XML » généré par Signavio Process Manager version 10.11.0 (550 lignes de code eXtensible Markup Language) n'a pas fonctionné tel quel avec Drools version 7. Quelques ajustements ont été nécessaires, essentiellement l'ajout du namespace « signavio » comme préfixe des types de variables créées avec cet outil : `<variable typeRef="signavio:monType"` (tandis que les types de variables génériques sont déjà définis dans le namespace « feel » : `<typeRef>feel:boolean</typeRef>`). Le préfixe adéquat a ensuite été ajouté dans la version suivante 11.0.0 de l'outil.

Afin d'appliquer les règles métier générées automatiquement, il convient également d'écrire un petit programme en langage Java. Alors qu'il faut ajouter au préalable deux lignes de code pour charger le fichier « DMN 1.1 XML », la création et l'initialisation du contexte sont très similaires à celui d'un objet. Par contre, la méthode de récupération du résultat (toujours aussi peu spectaculaire) s'avère particulièrement concise, voire élégante, par rapport à celle de la première solution.

```
01: DMNRuntime runtime = DMNRuntimeUtil.createRuntime(
       "13-Decider-recevabilite-vacataire-DMN.dmn", this.getClass() );
02: DMNModel dmnModel = runtime.getModel(
       "http://www.signavio.com/dmn/1.1/diagram",
          "13-Decider-recevabilite-vacataire-DMN" );
03: DMNContext context = DMNFactory.newContext();
04: context.set( "directionEntreprise", true);
05: context.set( "activiteSalariee", false);
06: context.set( "nombreHeuresParAn", 800);
07: context.set( "assujettiCet", true);
08: context.set( "moyensExistenceReguliers", true);
09: DMNResult dmnResult = runtime.evaluateAll( dmnModel, context );
10: DMNContext result = dmnResult.getContext();
11: System.out.println( "Recevabilité du vacataire : " + result.get(
       "deciderRecevabiliteVacataire" ) );
```

Algorithme 6 : Programme Java d'application des règles, deuxième solution (extrait)

La Figure 62 montre l'exécution de ce programme et donc l'automatisation de la prise de décision opérationnelle (Résultat) dans l'environnement de développement Eclipse :

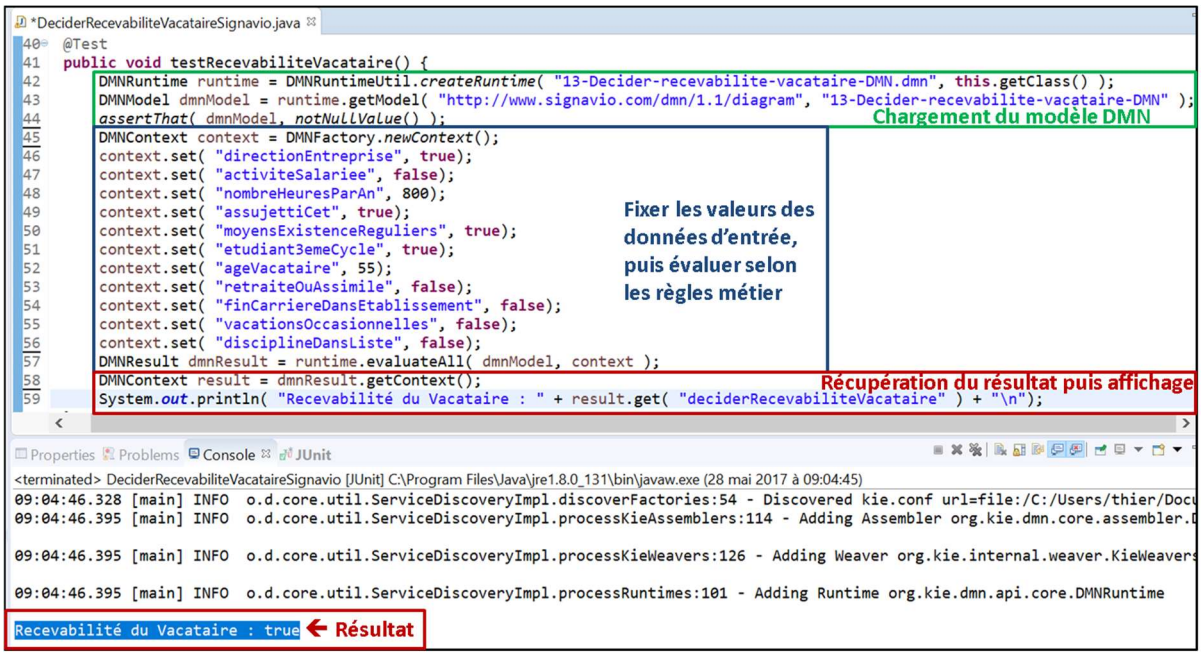


Figure 62 : Extrait du programme Java et exécution dans l'environnement Eclipse

6.4.5 Troisième solution intégrée : le CIM uniquement

Une troisième solution dite intégrée existe en utilisant l'outil Signavio Workflow Accelerator [Signavio, 2017b]. Les modèles de décision, appelés par les modèles de processus, sont alors transférés du Process Manager vers le Workflow Accelerator (déploiement simple et rapide) pour l'exécution des processus et l'automatisation des prises de décision. Les utilisateurs de cette plate-forme intégrée n'ont alors pas accès aux fichiers échangés et encore moins au code des programmes. Cette solution très pratique, mais propriétaire à l'éditeur Signavio, n'est pas détaillée dans cette thèse.

Trois modèles MDA	Signavio Workflow Accelerator (solution 3)
CIM (Computation Independent Model)	DRD (Decision Requirements Diagram) + Tables de décision
PIM (Platform Independent Model)	✘
PSM (Platform Specific Model)	✘

Tableau 17 : Projection des éléments de la troisième solution sur les modèles MDA

6.4.6 Comparaison des trois solutions

Ces trois solutions, réalisées à l'aide des mêmes outils gratuits (pour le monde académique au moins) ont été comparées à partir de la même étude de cas (Décider de la recevabilité d'un vacataire lors d'un processus de recrutement dans un établissement public d'enseignement supérieur). Sans surprise, ces trois solutions donnent, à partir du même modèle, le même résultat - une prise de décision identique, mais avec deux approches assez différentes.

Le *Tableau 18* compare ces trois solutions avec l'aide des 3 modèles de la MDA :

MDA	Solution 1 spécifique Drools	Solution 2 générique	Solution 3 intégrée Signavio
CIM	DRD (Decision Requirements Diagram) + Tables de décision	DRD (Decision Requirements Diagram) + Tables de décision	DRD (Decision Requirements Diagram) + Tables de décision
PIM	✘	DMN 1.1 XML (Modèle interchangeable)	✘
PSM	DRL (Drools Rule Language) + DMN Formulae Java8-1.0-SNAPSHOT.jar	✘	✘

Tableau 18 : Comparaison des trois solutions d'automatisation des prises de décision

Le critère principal de comparaison est celui de l'indépendance aux outils utilisés. Aussi, la deuxième solution générique a sans aucune hésitation notre préférence et c'est celle que nous recommandons. Elle est mieux alignée sur les modèles MDA que la première solution spécifique, car cette deuxième solution est indépendante de toute plate-forme. Elle est très intéressante si la plate-forme cible - le moteur de règles (BRMS) - est capable d'interpréter directement des fichiers au format « DMN 1.1 XML » (ce qui reste encore assez rare).

En fait, la première solution a permis de mettre en valeur la deuxième ! La première solution a également permis de découvrir le changement important de paradigme, la programmation déclarative, qui est également utilisée de manière implicite par la seconde solution.

La troisième solution intégrée est pratique, car entièrement hébergée chez l'éditeur Signavio. Si l'utilisateur a accès au modèle DMN (diagramme et tables de décision), il n'a accès ni aux fichiers échangés entre les modules ni au code des programmes utilisés pour l'automatisation. Cette solution n'a donc pas pu être étudiée complètement dans le cadre de cette thèse.

7 Conclusion et Perspectives

7.1 Conclusion

7.1.1 Approche complète, de bout en bout

La modélisation des prises de décisions opérationnelles a donc été effectuée grâce au nouveau langage standard DMN (Decision Model and Notation), avec un outil développé sur mesure, puis avec un outil standard. La réalisation d'un démonstrateur a permis de démontrer qu'il était possible d'automatiser ces prises de décisions, modélisées préalablement. Cette approche complète, de bout en bout, a été faite avec une démarche d'Architecte d'Entreprise.

7.1.2 Modélisation des prises de décisions adaptée

Après avoir présenté les principes de l'Architecture d'Entreprise, les besoins en modélisation et les limites du langage BPMN, bien adapté aux processus métier, pour la représentation des prises de décision opérationnelles, cette thèse a proposé une nouvelle approche.

En séparant les préoccupations du modélisateur (comme l'architecte l'a déjà fait pour les applications métier), les processus, les prises de décisions et les règles métier sont alors représentés par des modèles pertinents, cohérents entre eux, complémentaires, bien que faiblement couplés. La notation graphique de DMN sous forme de diagramme a été présentée. Quant au formalisme des tables de décision, il semble éprouvé et robuste.

L'objectif était de démontrer que DMN est un langage pertinent (ou pas) pour modéliser les prises de décisions opérationnelles, en complément du langage BPMN utilisé pour modéliser les processus métier. Deux exemples de modélisation en BPMN uniquement (sans DMN donc), puis avec DMN, ont permis de comparer l'approche traditionnelle avec cette nouvelle approche. Dans ce cas particulier, extrait de la réalité, le modèle DMN semble bien plus simple à comprendre et éventuellement à modifier que le modèle BPMN. La pertinence du langage DMN semble ainsi démontrée.

7.1.3 Automatisation des prises de décisions possible

La notation graphique simple de DMN ne constitue pas un inconvénient, bien au contraire : nous avons démontré, lors de cette thèse, que les prises de décision modélisées selon le langage DMN (diagramme et tables de décision) pouvaient être automatisées. Les trois modèles de la MDA (CIM, PIM et PSM) nous ont permis de porter un regard d'architecte sur les solutions techniques proposées, afin de les comprendre, puis de les mettre en œuvre. Selon l'approche moderne de l'architecture des applications métier, c'est un moteur de règles externe qui a été utilisé.

Cette mise en œuvre nécessite toutefois des compétences de haut niveau. Avec quelques dizaines lignes de code Java, judicieusement choisies puis validées par notre expérimentation, il est possible d'appliquer des règles métier complexes, qui ont été définies en amont par des analystes métier, afin d'automatiser les prises de décisions opérationnelles.

Plus que de présenter les nouvelles fonctions Java pour utiliser un modèle DMN, il s'agissait surtout de dégager une méthode, voire quelques bonnes pratiques, pour les mises en œuvre suivantes. Par exemple, si le jeu de données d'entrée (les critères de décision) est complet dès le départ, il est possible de modifier les règles métier dans le modèle (en l'occurrence, directement dans une table de décision), sans changer le code Java qui les applique, ce qui est parfait pour garantir l'agilité du système. Voilà donc une bonne pratique préliminaire : identifier, une bonne fois pour toutes, les données d'entrée nécessaires, minutieusement et exhaustivement.

7.1.4 De l'intérêt de s'appuyer sur un langage standard

Tandis que l'automatisation des prises de décisions opérationnelles existe depuis une quinzaine d'années [Taylor, 2007], le fait de pouvoir désormais s'appuyer en amont sur le langage standard DMN est un progrès indéniable. Après une courte formation, il devient possible d'expliquer en détail comment sont prises les décisions, selon des règles métier bien précises. Et de discuter dans un langage commun, dont une notation graphique accessible, entre spécialistes métier et informaticiens notamment.

C'est aussi une bonne manière pour documenter les règles métier mises en place pour répondre aux contraintes réglementaires, de plus en plus nombreuses (cette remarque vaut également pour la modélisation des processus métier). L'Entreprise peut alors organiser ses modèles de processus, de prises de décisions et de règles métier, dans le cadre de représentation de son architecture, accessible au plus grand nombre.

7.1.5 Méthode, langage et outils

Après que les processus métier ont été extraits des applications dans les années 2000, nous pouvons présager que la prochaine étape sera sans doute l'extraction des règles métier dans les années qui viennent (cf. 5.2 Evolution de l'Architecture des Applications Métier).

L'existence du langage standard DMN pour la modélisation des prises de décisions opérationnelles, et la possibilité récente d'automatiser ces prises de décisions constituent de sérieux atouts pour la réalisation de ce présage.

Quant aux outils disponibles, une vingtaine déjà, dont certains sont gratuits, permettent de modéliser les prises de décision opérationnelles. Tandis que la notation graphique est accessible facilement, ces outils permettent aussi de vérifier automatiquement la complétude et la cohérence des règles métier formalisées dans les tables de décision, ce qui s'avère bien pratique pour concevoir des modèles de qualité.

Mais ce langage standard DMN et les outils de modélisation, voire d'automatisation, ne seront vraiment efficaces que s'ils sont utilisés avec méthode.

Aussi, une ébauche de proposition pour enrichir la méthode Praxeme - rappelons qu'il s'agit d'une méthode ouverte - a été proposée. L'impact, sur l'architecture métier en général et sur l'aspect Logique de la méthode Praxeme en particulier, sera évalué, ainsi que le gain potentiel sur l'évolutivité grâce à la séparation des préoccupations (processus et décisions) et la factorisation des règles métier et donc la réduction du code.

Nous nous rapprocherons ainsi du triptyque idéal Méthode-Langage-Outil.

7.2 Perspectives

7.2.1 Solution indépendante des moteurs de règles

Les moteurs de règles *open source* (intégrés dans des systèmes complets BRMS), comme Drools et OpenRules ont déjà commencé à tirer parti du langage DMN en étant capable d'interpréter directement le format d'interchange DMN 1.1 XML. Cette solution constitue une avancée indéniable, bien plus pérenne que celle basée sur les formats spécifiques, qui a encore la préférence des moteurs de règles payants. En effet, les utilisateurs du langage DMN préfèreront utiliser une solution indépendante de toute plate-forme technique.

7.2.2 Evolutions attendues de DMN

Le langage standard DMN est très récent. La simplicité de son diagramme emblématique n'est qu'apparente et la représentation des prises de décisions opérationnelles peut devenir complexe (cf. Figure 49). La version actuelle de DMN est la version 1.1. Des améliorations sont déjà proposées pour la prochaine version [Taylor et Purchase, 2016]. En voici quelques exemples :

Pour le diagramme : cardinalité (avec le même symbole « ||| » multi-instance utilisé par BPMN)

Pour la table de décision : support des expressions en langage FEEL comme données d'entrée

Pour la logique de décision : représentation de la logique par un arbre de décision

7.2.3 Solveur de contraintes

Un moteur de règles peut compléter un solveur de contraintes : alors que le solveur explore les solutions possibles dans un environnement non prédéfini, un moteur de règles peut être utilisé en complément pour calculer les scores des solutions trouvées, afin de ne retenir que les mieux notées. Le calcul d'un score n'est pas vraiment modélisé : les contraintes dites douces sont indiquées directement dans un tableau (très similaire à une table de décision, avec un politique de succès multiple de type Collect, qui cumule les scores obtenus).

Un diagramme DMN préalable faciliterait sans doute la compréhension de la méthode de calcul du score, sans toutefois apporter beaucoup de valeur ajoutée. Il s'agit d'un cas d'application du chapitre « 5.4.4.3 Tâche Business Rules sans branchement », où le langage DMN est utilisé pour effectuer un calcul complexe. Mais d'autres contraintes dites dures sont codées directement dans un langage de programmation spécifique et s'avèrent peu lisibles et bien difficiles à représenter graphiquement. Peut-être que le langage déclaratif FEEL (Friendly Enough Expression Language) fourni avec DMN permettrait une représentation plus générique.

7.2.4 Formalisation des règles de l'Architecture d'Entreprise

Il fut envisagé dans un premier temps de choisir comme terrain d'investigation un « vrai » domaine métier, comme celui du Transport et de la Logistique, pour appliquer la théorie à la pratique (tandis que les exemples de prises de décisions opérationnelles dans les ouvrages spécialisés ont souvent choisi le domaine de la Banque et de l'Assurance).

L'idée est ensuite venue d'appliquer la modélisation, voire l'automatisation, des prises de décisions au domaine métier de l'Architecture d'Entreprise lui-même (forme de récursivité, voire propriété de métacircularité). Il s'agirait donc de formaliser puis d'appliquer des règles du métier d'architecte pour prendre les bonnes décisions. Tandis que cela semble bien difficile pour les règles de haut niveau, qui conduisent à prendre des décisions stratégiques ou tactiques, les règles de bas niveau, celles de la modélisation par exemple, qui conduisent à prendre des décisions opérationnelles, semblent plus « facilement » formalisables.

Idéalement, il faudrait notamment arriver à modéliser les règles de transformation des modèles, afin de pouvoir automatiser cette transformation. Et enfin, intégrer cette perspective dans une approche processus plus globale, visant à simplifier la démarche d'Architecture d'Entreprise pour le rendre accessible à un nombre plus important d'individus.

La principale perspective est donc de proposer un nouvel outil d'architecture qui permette à l'Entreprise de relever les défis de ses transformations à venir.

8 Bibliographie et autres Références

ARIS Community. (2009). Event-driven Process Chain (EPC). http://www.ariscommunity.com/event-driven-process-chain

Bernus, P., Nemes, L. (1996). A framework to define a generic enterprise reference architecture and methodology. *Computer Integrated Manufacturing Systems*, *9*(3), 179-191. https://doi.org/10.1016/S0951-5240(96)00001-8

Biard, T., Le Mauff, A., Bigand, M., Bourey, J.-P. (2015). Separation of Decision Modeling from Business Process Modeling Using New "Decision Model and Notation" (DMN) for Automating Operational Decision-Making. In L. M. Camarinha-Matos, F. Bénaben, & W. Picard (Éd.), *Risks and Resilience of Collaborative Networks* (Vol. 463, p. 489-496). Springer International Publishing.

Birmingham City Council. (2006). CHAMPS2 Business Change Method. http://www.champs2.info/

Bonnet, P., Detavernier, J.-M., Vauquier, D. (2007). *Le système d'information durable*. Hermès science.

BPM Offensive Berlin. (2015). BPMN Poster. http://www.bpmb.de/index.php/BPMNPoster

Brackett, J. W. (1990). Software requirements. *Carnegie-Mellon University Software Engineering Institute*.

Brandenburg, H., Wojtyna, J.-P. (2009). L'approche processus mode d'emploi. Eyrolles-Ed. d'organisation.

Budinsky, F. (Éd.). (2007). *Eclipse modeling framework: a developer's guide*. Addison-Wesley.

Caseau, Y. (2011). *Urbanisation, SOA et BPM*. Dunod.

Chelli, H. (2003). *Urbaniser l'entreprise et son système d'information ; Guide des entreprises agiles*. Vuibert.

CIGREF. (2003). *Accroître l'agilité du système d'information - Urbanisme : des concepts au projet*. http://www.cigref.fr/accroitre-lagilite-du-systeme-dinformation

Clark, T., Sammut, P., Willans, J. S. (2015). Applied Metamodelling: A Foundation for Language Driven Development (Third Edition). *CoRR*, *abs/1505.00149*.

CODASYL. (1982). A modern appraisal of decision tables. ACM.

Debevoise, T., Taylor, J. (2014). *The MicroGuide to Process and Decision Modeling in BPMN/DMN*. ACR.

Desfray, P., Raymond, G. (2014). *TOGAF en pratique*. Dunod.

DGME. (2009). Référentiel Général d'Interopérabilité. http://references.modernisation.gouv.fr/interoperabilite

DGME. (2012). Cadre Commun d'Urbanisation du Système d'Information de l'Etat. http://references.modernisation.gouv.fr/urbanisation-du-systeme-dinformation-de-letat

Dijkstra, E. W. (1974). On the role of scientific thought.

Doumeingts, G., Vallespir, B., Chen, D. (1998). Decisional Modelling using the GRAI Grid. In *International Handbook on Information Systems* (p. 313-337).

Eclipse. (2006). Acceleo. http://www.eclipse.org/acceleo/

Eclipse. (2013). Sirius. http://www.eclipse.org/sirius/

ESPRIT Consortium AMICE (Ed.). (1993). *CIMOSA: open system architecture for CIM* (2nd, rev. and extended ed.). Springer-Verlag.

Feldman, J. (2017). *DMN in Action with OpenRules*. Amazon.

Fish, A. (2012). *Knowledge automation: how to implement decision management in business processes*. Wiley.

Garibian, G. (1985). Arbre de Performance®. https://fr.slideshare.net/GeorgesGaribian

Harzallah, M., Berio, G., Opdahl, A. L. (2012). New perspectives in ontological analysis: Guidelines and rules for incorporating modelling languages into UEML. *Information Systems*, *37*(5), 484-507. https://doi.org/10.1016/j.is.2011.11.001

Henderson, J., Venkatraman, N. (1992). Strategic Alignment.

Jézéquel, J.-M., Combemale, B., Vojtisek, D. (2012). *Ingénierie dirigée par les modèles*. Ellipses.

Le Moigne, J. L. (1994). *La théorie du système général: théorie de la modélisation*. Presses universitaires de France.

Lillehagen, F. M., Krogstie, J. (2010). *Active knowledge modeling of enterprises*. Springer.

Linehan, M., de Sainte Marie, C. (2011). The Relationship of Decision Model and Notation (DMN) to SBVR and BPMN. http://www.brcommunity.com/b597.php

Longépé, C. (2009). *Le projet d'urbanisation du SI : cas concret d'architecture d'entreprise*. Dunod.

Mader, A. H., Wupper, H., Boon, M. (2007). *The Construction of Verification Models for Embedded Systems*. Centre for Telematics and Information Technology University of Twente.

Mamoghli, S. (2013, janvier). *Contribution to the alignment of off-the-shelf product based information systems : towards a model-driven engineering, based on risk identification* (Theses). Université de Strasbourg. https://tel.archives-ouvertes.fr/tel-00814495

OMG. (2008). Model To Text Transformation Language (M2T). http://www.omg.org/spec/MOFM2T/

OMG. (2013). Business Process Model and Notation (BPMN). http://www.omg.org/spec/BPMN/

OMG. (2014). Model Driven Architecture (MDA). http://www.omg.org/mda/specs.htm

OMG. (2015a). Business Motivation Model (BMM). http://www.omg.org/spec/BMM/

OMG. (2015b). Unified Modeling Language (UML). http://www.omg.org/spec/UML/

OMG. (2016a). Case Management Model and Notation (CMMN). http://www.omg.org/spec/CMMN/

OMG. (2016b). Decision Model and Notation (DMN). http://www.omg.org/spec/DMN/

OMG. (2017). Semantics of Business Vocabulary and Rules (SBVR). http://www.omg.org/spec/SBVR/

OpenRules. (2016). Decision Model and Notation (DMN) Supporting Tools. http://openjvm.jvmhost.net/DMNtools/

Pitschke, J. (2014). Mastering Business Modeling – Applying Business Architecture and Standards successfully. http://www.enterprise-design.eu/files/images/download-praesentaionen/BPMEurope2014_JPitschke.pdf

Porter, M. E. (2003). *L'avantage concurrentiel*. Dunod.

Pragmatic EA. (2013). Enterprise Frameworks. http://www.pragmaticea.com/frameworks.asp

Praxeme Institute. (2006). Initiative pour une méthode publique. http://www.praxeme.org/

Red Hat. (2017). Drools, Business Rules Management System. http://www.drools.org

Roque, M. (2005). *Contribution à la définition d'un langage générique de modélisation d'entreprise*. Université Bordeaux 1. http://grenet.drimm.u-bordeaux1.fr/pdf/2005/ROQUE_MATTHIEU_2005.pdf

Ross, J. W., Weill, P., Robertson, D. (2006). *Enterprise architecture as strategy: creating a foundation for business execution*. Harvard Business School Press.

Ross, R. G. (2013). *Business rule concepts: getting to the point of knowledge* (4th Ed). Business Rule Solutions.

Signavio. (2017a). Signavio Process Manager. https://www.signavio.com/products/process-manager/

Signavio. (2017b). Signavio Workflow Accelerator. https://www.signavio.com/fr/products/workflow-accelerator/

Silver, B. (2016). *DMN method and style*. Cody-Cassidy Press.

Simon, H. A. (1983). *Administration et processus de decision.* Economica.

Suwa, M., Scott, A. C., Shortliffe, E. H. (1982). An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System. *AI Magazine*, *3*, 16-21.

Tardieu, H., Rochfeld, A., Colletti, R. (2000). *La méthode Merise : principes et outils*. Ed. d'Organisation.

Taylor, J. (2007). *Smart (enough) systems: how to deliver competitive advantage by automating the decisions hidden in your business*. Prentice Hall.

Taylor, J., Purchase, J. (2016). *Real-World Decision Modeling with DMN*. Meghan-Kiffer Press.

The Open Group. (2011). TOGAF Version 9.1. http://www.opengroup.org/togaf/

The Open Group. (2017). ArchiMate® 3 Specification. http://pubs.opengroup.org/architecture/archimate3-doc/

Trisotech. (2017). Digital Enterprise Suite. http://www.trisotech.com/digital-enterprise-suite

Van Roy, P., Haridi, S. (2004). *Concepts, techniques, and models of computer programming*. MIT Press.

Vanthienen, J., Dries, E. (1994). Decision Tables: Refining the Concept and a Proposed Standard. *ResearchGate*.

Vernadat, F. (2001). UEML: Towards a Unified Enterprise Modelling Language. http://www1.utt.fr/mosim01/pdf/invite/ARTICLE-invite-vernadat.pdf

Von Halle, B., Goldberg, L. (2009). *The Decision Model*. Auerbach.

Zachman, J. (2011). About the Zachman Framework™. http://www.zachman.com/about-the-zachman-framework

université PARIS-SACLAY
ÉCOLE DOCTORALE
INTERFACES
Approches Interdisciplinaires : Fondements, Applications et Innovation

Titre : De la modélisation à l'automatisation des prises de décisions opérationnelles avec une démarche d'Architecture d'Entreprise

Mots clés : architecture d'entreprise ; prise de décision ; règle métier ; modélisation ; DMN ; Praxeme.

Résumé : Après avoir défini l'Architecture d'Entreprise, en tant que discipline, son contexte de transformation, puis ses principaux cadres et méthodes (la méthode Praxeme surtout), cette thèse décrit les besoins en modélisation, notamment pour représenter les processus métier et les prises de décision opérationnelles.
Après un état de l'art des langages et notations existants pour la modélisation des processus métier, des prises de décisions et des règles métier, tant dans le monde académique quand dans le monde industriel, les langages et notations standards sont présentés en détail.
Cette thèse démontre les limites des langages et notations pour modéliser les processus métier à représenter les prises de décisions opérationnelles.
Puis, elle évalue le nouveau langage DMN (Decision Model and Notation), proposée par l'OMG (Object Management Group) et sujet de recherche principal de cette thèse, afin de vérifier qu'elle constitue une solution alternative mieux adaptée, en appliquant le principe de séparation des préoccupations.
Le modèle DMN obtenu est composé d'un diagramme et de tables de décision. Les expérimentations avec un démonstrateur, mises en œuvre dans cette thèse, montrent qu'il est possible d'automatiser les prises de décisions opérationnelles ainsi modélisées. Plusieurs solutions techniques seront détaillées puis comparées, à la lumière de la MDA (Model Driven Architecture).
Enfin, plusieurs perspectives intéressantes de l'utilisation de DMN sont développées dans la conclusion.

Title: From modeling to automation of operational decision-making with an Enterprise Architecture approach

Keywords: Enterprise Architecture; decision-making; business rules; modeling; DMN; Praxeme.

Abstract: After defining the Enterprise Architecture, as a discipline, its context of transformation, and then its main frameworks and methods (the Praxeme method above all), this thesis describes the modeling needs, notably to represent the business processes and the operational decisions.
After a state of the art of existing languages and notations for the modeling of business processes, decision-making and business rules, both in the academic world and in the industrial world, the standard languages and notations are presented in detail.
This thesis demonstrates the limitations of the languages for modeling business processes to represent operational decision-making.
It then evaluates the new DMN language (Decision Model and Notation), proposed by the OMG (Object Management Group) and the main research subject of this thesis, in order to verify that it is a better adapted alternative solution, applying the separation of concerns principle.
The resulting DMN model is composed of a diagram and decision tables. The experiments with a demonstrator, implemented in this thesis, show that it is possible to automate the operational decision-making and modeled. Several technical solutions will be detailed and compared in light of the MDA (Model Driven Architecture).
Finally, several interesting perspectives of the DMN use are developed into the conclusion.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France