Approches pour la modélisation des systèmes complexes

Ana Roxin, Christophe Castaing

5

Approches pour la modélisation des systèmes complexes

Ana Roxin[1], Christophe Castaing[2]

[1] *Laboratoire Informatique de Bourgogne, Univ. Bourgogne Franche-Comté, Dijon, France*
[2] *Egis, Paris, France*

5.1. Introduction

Dans ce chapitre, nous présentons deux principales familles d'approches pour la modélisation de systèmes complexes. Notamment, nous étudions les approches à base de modèles de données et celles à base de modèles de connaissances. Pour chacune de ces deux familles d'approches, nous en résumons l'historique, les principes de modélisation sous-jacents et présentons les normes internationales qui les utilisent.

Toutefois, avant de rentrer dans les détails et afin de mieux comprendre les différences entre ces approches, nous commençons avec un bref aperçu du cadre de référence pour la gouvernance des données e.g. la série de normes ISO 8000. Publiée par l'ISO/TC 184/SC 4, cette série traite des principes de qualité des données et spécifie des caractéristiques permettant d'évaluer la qualité (ou le degré de conformité) des données d'une organisation. La qualité des données est en effet un facteur déterminant de la qualité des informations et, par conséquent, de la précision et de la fiabilité des connaissances pouvant être déduites de ces informations. En effet, les informations sont des données placées dans un contexte et les connaissances sont des informations accompagnées d'une expérience ou d'un savoir-faire. Selon l'ISO 8000-1 (ISO/DIS 8000-1, 2011), les données ont des attributs de provenance, précision et complétude. Elles respectent une syntaxe formelle, et utilisent des concepts définis dans des dictionnaires.

Ces éléments se retrouvent dans les différentes approches informatiques existant pour la structuration des données. Ces approches sont implémentées sous la forme de langages informatiques et peuvent être classées selon leur degré de formalité. En informatique, un tel langage est dit formel et comprend un alphabet (ensemble d'éléments), des règles pour déterminer si un élément appartient à l'alphabet du langage (grammaire) et si un ensemble composé d'éléments respecte ces règles (syntaxe). En informatique, les éléments d'un langage formel, en plus d'une syntaxe, ont aussi un sens ou une sémantique. Plus la sémantique est définie de manière explicite et logique, plus le langage est dit formel. Ceci est illustré par la **Erreur ! Source du renvoi introuvable.**. Les dictionnaires de données, au sens informatique, sont considérés informels, car ils contiennent seulement des termes avec leurs définitions. L'interprétation de ces termes (leur sémantique) relève de l'utilisateur et ne peut pas être faite par la machine. La même remarque peut être faite à propos des glossaires et des hiérarchies de termes. A l'opposé, les langages logiques, basés sur la logique du premier ordre (en anglais FOL ou First Order Logic), représentent des approches pour spécifier de manière explicite le sens d'énoncés, permettant ainsi leur interprétation par une machine (au travers l'exécution d'algorithmes appelés procédures de décision).

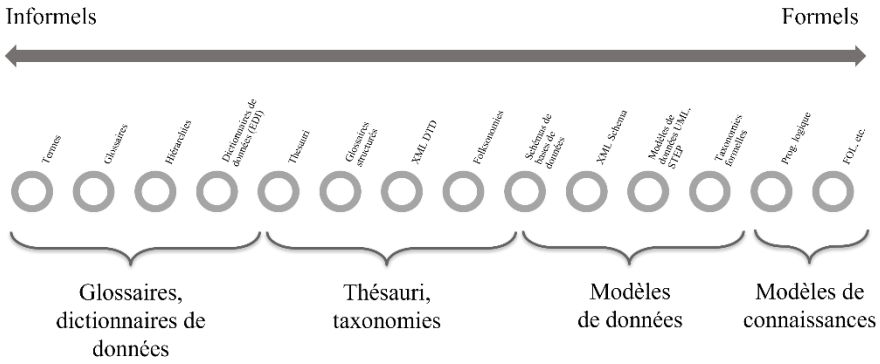

Figure 5.1. *Approches de structuration des données selon le degré de formalité des langages utilisés*

Dans un contexte de modélisation des informations de la construction (BIM), différents travaux de normalisation, menés au sein de l'ISO, apportent un cadre complémentaire des approches informatiques pour la conception de dictionnaires de données. Nous pouvons citer ici les normes suivantes:

- La norme ISO 12006-2:2015 (ISO 12006-2, 2015) est à destination d'organismes élaborant et publiant des systèmes et des tableaux de classification. L'ISO 12006-2:2015 spécifie des intitulés de tableaux de classifications en vue d'une harmonisation mutuelle;

- La norme ISO 23386:2020 (ISO 23386, 2020) précise des règles pour la définition et la gestion de propriétés utilisées dans la construction. Un ensemble d'attributs devant être utilisés dans la définition de telles propriétés est ainsi défini. Des processus BPMN (Business Process Modeling Notation) sont spécifiés pour l'échange numérique de propriétés (ou groupes de propriétés) ainsi décrit.

Toutefois, aucune de ces normes ne précise pas un type de modèle (au sens de la Figure 5.1) à utiliser pour structurer les données sous-jacentes. En effet, la Figure 5.1, identifie deux familles de modèles: les modèles de données et les modèles de connaissances. Les langages pour les modèles de données (e.g. UML ou Unified Modeling Language) sont plus formels que ceux permettant de définir un dictionnaire. En effet, les règles de composition d'éléments (syntaxe) sont plus contraintes. Toutefois, le sens des données n'est pas spécifié formellement: les modèles de données représentent (graphiquement) des connaissances (pour cela ils sont dits signifiants), mais un algorithme informatique ne peut pas raisonner sur ces connaissances – un utilisateur humain, expert du domaine, est nécessaire pour interpréter les connaissances ainsi représentées.

Les approches à base de langages logiques constituent la famille des modèles de connaissances. Leur degré de formalité est maximisé, et ils constituent des spécifications explicites et formelles de connaissances. Dans un tel contexte, il est donc possible d'utiliser un algorithme pour simuler un raisonnement humain sur ces connaissances. Qui plus est, un tel raisonnement peut être lui-même spécifié au travers de langages logiques, et toutes les déductions faites par l'algorithme peuvent être expliquées de manière logique (par exemple sous la forme d'arbres de décision).

Les sections suivantes détaillent les deux familles d'approches et présentent leurs avantages et inconvénients respectifs lorsqu'appliqués à l'étude et à l'analyse des systèmes complexes, tels que les projets de construction d'infrastructures ou encore le jumeau numérique d'un bâtiment.

5.2. Les approches à base de modèles d'objets

UML et les formalismes associés sont apparus dans le cadre plus global de ce qu'on appelle la conception orientée-objet. En effet, les modèles dits à base d'objets

(ou de classes) sont apparus au début des années '90, dans un contexte où les modèles de données du type {Entité / Relation} ainsi que les diagrammes de flux étaient déjà largement utilisés par les entreprises pour les exigences système (depuis les années '70). Afin de différencier les approches de modélisation existantes, une distinction est faite selon ce sur quoi porte la modélisation: la nature (ou le type) d'une activité (ou d'un domaine métier (modélisation du problème), ou un système qui l'implémente (modélisation de la solution).

C'est ainsi que lorsque le terme d'"analyse orientée-objet" est inventé par les auteurs de (Coad, et al., 1990), il désignait une approche pour intégrer des services et des messages (concepts issus de la programmation orientée-objet) dans des modèles {Entité/Relation}. L'idée initiale était d'améliorer la gestion de l'héritage faite par ces derniers. Pour Coad et Yourdon, un "objet" était une "une abstraction de quelque chose dans l'espace du problème, reflétant les capacités d'un système pour garder des informations à ce sujet, interagir avec ou les deux ; une encapsulation de valeurs d'attributs et de leurs services exclusifs" (Coad, et al., 1990). Selon cette définition, une analyse orientée-objet vise l'espace du problème (e.g. l'entreprise, l'organisation, le domaine métier considéré) et non pas l'espace des solutions (e.g. les ordinateurs, les langages et les programmes informatiques). Ainsi, initialement, une modélisation avec une approche orientée-objet visait l'espace du "problème" et non de la solution. Malheureusement, ce lien avec l'espace "problème" est devenu de moins en moins fort. En effet, lorsqu'UML est apparu, les aspects d'analyse de l'environnement hors informatique semblent avoir été totalement perdus.

Ceci s'est confirmé en 1999, avec la définition d'un objet formulée par (Rumbaugh, et al., 1999). Un objet est défini en tant qu'"instance d'une classe". Toujours selon les auteurs de (Rumbaugh, et al., 1999), une classe est un "descripteur pour un ensemble d'objets qui partagent les mêmes attributs, opérations, méthodes, relations et comportements". Avec ces définitions, qui font loi dans la communauté UML, tout est objet. Le lien avec l'implémentation (informatique) du système deviennent intrinsèques à toute modélisation UML. Une modélisation UML met ainsi l'accent sur l'orienté-objet plutôt que sur l'analyse d'un espace problème. Nous verrons dans la section 5.3 que les modèles à base de connaissances sont plus adaptés pour la modélisation de problèmes. Le langage UML est central dans les approches dites à base de modèles (MDA ou Model-Driven Architecture). Dans la section suivante, nous présentons ces approches MDA et leur structure en couches.

5.2.1. Les architectures à base de modèles et les normes associées

Spécifiées dans le cadre du Object Modeling Group (OMG), et reprenant les principes de l'ingénierie dirigée par les modèles (MDE ou Model-Driven Engineering), les architectures à base de modèles (MDA ou Model-Driven

Architecture) définissent des règles pour structurer la représentation de systèmes sous la forme de modèles. Les approches MDA (Miller, et al., 2003) considèrent trois niveaux d'abstraction à partir desquels un système peut être décrit. Dans l'approche MDA, un tel niveau est considéré comme "une technique d'abstraction permettant de se concentrer sur un ensemble particulier de problèmes au sein d'un système tout en supprimant tous les détails non pertinents. Un niveau d'abstraction peut être représenté via un ou plusieurs modèles" (Tuyen, 2006). Chaque niveau d'abstraction contient une représentation du système considérée, représentation appelée modèle de point de vue. L'approche MDA définit trois niveaux d'abstraction correspondant à trois types de modèles :

- Le modèle indépendant du calcul (CIM ou Computation Independent Model) concerne le contexte et les besoins (prérequis) du système, sans prendre en compte sa structure ou ses processus. Le langage UML (Booch, et al., 1998) permet de définir ce type de modèles (ISO/IEC 19505, 2012) ;
- Le modèle indépendant de la plate-forme (PIM ou Platform Independent Model) décrit les capacités opérationnelles d'un système (définies en tant qu'abstractions de la plate-forme), sans tenir compte de détails d'implémentation spécifiques à la plate-forme (ou à un ensemble de plateformes) ;
- Le modèle spécifique à une plate-forme (PSM ou Platform Specific Model) représente une traduction d'un modèle PIM par rapport à une plate-forme spécifique. Ces modèles PSM sont définis au travers de langages dits d'implémentation (e.g. Java, Python ou encore XML Schema). Des outils automatisés (e.g. des outils de transformation de modèles) sont utilisés pour assurer la traduction du modèle PIM en différents modèles PSM.

Les approches MDA reposent sur une architecture dite de méta-modèles, structurée en couches, illustrées dans la figure 5.2 et décrites ci-dessous.

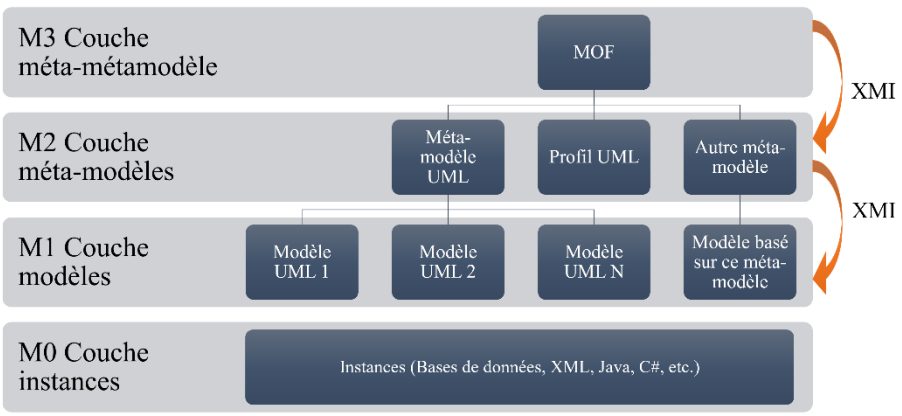

Figure 5.2. *Couches de l'architecture de méta-modèles MDA*

Les modèles mentionnés dans les couches ci-dessous peuvent être d'un des 3 types (CIM, PIM ou PSM) ci-dessus :

1. La couche méta-métamodèle (M3) est définie au travers de la norme MOF (Meta-Object Facility) (2014). Cette dernière définit un langage abstrait et un cadre pour la spécification, la construction et la gestion de métamodèles indépendamment d'une technologie. Le langage MOF permet de se représenter lui-même, il est dit autodescriptif. Il permet aussi de représenter tous les autres langages de l'approche MDA (par exemple l'UML).
2. La couche méta-modèle (M2) comprend les langages construits selon la norme MOF, notamment UML (Booch, et al., 1998) et XMI (XML Metadata Interchange) (XMI, 2014). Le langage UML permet de spécifier différents aspects d'un système (qu'il s'agisse d'une organisation ou d'un logiciel), notamment ses aspects statiques (e.g. diagrammes de classes, de composants), ses aspects dynamiques (e.g. comportement à l'exécution, processus métiers au travers de diagrammes d'activités ou de séquences), de même que les relations entre les utilisateurs et les fonctionnalités du système (e.g. diagrammes de cas d'usage). La norme XMI permet d'échanger des modèles représentés sous la forme de documents XML. La norme EML (Eclipse Modeling Framework) permet de représenter les modèles en tant que classes Java, tout en permettant d'effectuer des opérations sur ces modèles e.g. validation ou transformation (Budinsky, et al., 2003). Le langage SysML (OMG, 2002) est un autre exemple de langage du niveau métamodèle ou M2. Développé en tant que profil UML 2, SysML est un langage de représentation graphique, très utilisé en ingénierie système, dans le cadre de l'ISO/TC184/SC4 "Données industrielles".
3. La couche modèle (M1) contient des modèles faits par les utilisateurs, souvent sous la forme de classes
4. La couche instance (M0) contient les instances des modèles de la couche supérieure (M1). Ces instances correspondent généralement aux éléments du monde réel, modélisés au travers des modèles de la couche M1.

D'autres approches de la famille MDA sont pertinentes pour la modélisation de systèmes complexes. Parmi celles-ci, nous pouvons citer l'approche Model-Based System Engineering (MBSE) pour l'ingénierie système basée sur les modèles, développée dans le cadre de l'INCOSE (International Council on Systems Engineering). Cette approche supporte la spécification de systèmes, de leurs besoins, leur conception, analyse, vérification ou encore validation. Elle s'applique à l'ensemble des phases du cycle de vie du système considéré (Long, et al., 2011). En lien avec MBSE, nous pouvons citer l'approche MBRE (Model-Based Requirements

Engineering) qui concerne l'ingénierie et la gestion des exigences à base de modèles. L'approche MBRE est indépendante de tout outil informatique, et peut donc être implémentée à l'aide de n'importe quel outil ou combinaison d'outils (Holt, et al., 2012). L'approche MBRE repose sur l'utilisation d'éléments graphiques afin d'échanger des exigences plus précises et concises. Avec cette approche, les exigences sont considérées une à une, par rapport à leurs définitions, puis on leur associe un "sens" en les plaçant dans le "contexte" approprié". Une exigence placée dans un contexte (les acteurs considérés ou les niveaux hiérarchiques constituant le système) s'appelle un "cas d'utilisation".

5.2.2. Normes internationales utilisant ce type de modélisation

L'approche MDA est utilisée dans le domaine des Systèmes d'Information Géographique. Les normes associées reposent sur différents schémas, modèles et méta-modèles, dont la structuration en couches correspond à l'approche MDA. Selon l'approche MDA, le langage utilisé pour la définition de modèles conceptuels SIG est le langage UML, tel que normalisé par l'ISO 19505 (ISO/IEC 19505-2, 2012). Les modèles SIG sont structurés selon les quatre couches de l'approche MDA. Pour chacune de ces couches, différentes normes internationales ont été définies, listées dans le tableau 5.1 ci-dessous.

	Normes
M3 Couche Méta-méta-modèle	Meta-Object Facility (MOF) ISO/IEC 19508:2014
M2 Couche Méta-modèle	Méta-modèle UML ISO/IEC 19505 Profil UML core ISO 19103 Profils UML pour les schémas d'application SIG ISO 19103
M1 Couche Modèle	Schémas d'application: INSPIRE GML / OGC CityGML / OGC InfraGML / OGC IndoorGML
M0 Couche Instances	XML / XSD - XMI (ISO/IEC 19509) Java - JMI (Java Metadata Interchage)

Tableau 5.1. *Normes internationales utilisant une modélisation à base d'objets*

La directive européenne 2007/2/CE INSPIRE (Infrastructure for Spatial InfoRmation in the European community) reprend aussi les principes MDA pour spécifier des principes pour l'interopérabilité et l'accessibilité aux informations géographiques. Au sens d'INSPIRE, l'interopérabilité représente une compatibilité entre deux systèmes leur permettant d'échanger des informations afin que d'autres systèmes puissent les comprendre (Ansorge, et al., 2016). Dans le cadre d'INSPIRE, les fonctionnalités du système sont spécifiées sous la forme d'un modèle indépendant

de la plate-forme (PIM). Ce modèle PIM est une spécification de données commune à tous les systèmes compatibles INSPIRE et en cela il représente une résolution de l'interopérabilité conceptuelle entre systèmes. Le langage utilisé pour définir ce modèle PIM est le langage UML. Pour l'interopérabilité opérationnelle, la directive INSPIRE considère des traductions du modèle PIM en des modèles spécifiques à des plateformes (PSM). Ces traductions sont réalisées via des procédures automatiques, reposant sur des langages tels que Java, XML Schema ou encore Python (Ansorge, et al., 2016). La spécification du modèle conceptuel INSPIRE est disponible dans (INSPIRE, 2013).

La norme STEP (ISO 10303, 1994) définit le cadre de référence pour les projets de construction ou d'ingénierie d'infrastructures. La famille de normes ISO 10303 définit une approche pour la représentation et l'échange de données produits, basée principalement sur le langage SysML (OMG, 2002). Elle fournit un mécanisme neutre permettant de décrire les produits tout au long de leur cycle de vie. La norme ISO 16739 IFC (Industrial Foundation Classes) (ISO 16739, 2018) représente un équivalent pour les modèles de données bâtiment.

La famille de normes ISO 10303 comporte différentes parties correspondant à ce qu'on appelle des protocoles d'application (AP ou Application Protocol), chacun définissant les échanges d'informations tout au long du cycle de vie d'un système. est une réponse à un ensemble de cas d'utilisation et d'exigences métier, permettant l'interopérabilité des informations produit. Chaque AP comprend un objectif, un diagramme d'activités décrivant ce qu'un ingénieur doit faire pour atteindre l'objectif visé (défini en BPMN), ainsi qu'un modèle des exigences d'application spécifiant les besoins en information des activités de l'ingénieur (IER ou Information Exchange Requirement). Ces besoins et exigences sont ensuite mappées dans l'ensemble commun de ressources intégrées (ou IR pour "integrated ressources").

Dans un contexte BIM, le protocole STEP AP 225 décrit les exigences associées aux éléments d'un bâtiment ("building éléments"), et le protocole STEP AP 242 spécifie les besoins associés à l'ingénierie 3D à base de modèles ("Managed Model-Based 3D Engineering"). Selon ces protocoles, et toujours dans un contexte BIM, les besoins et exigences sont spécifiés en langage naturel, selon la méthodologie définie par la norme Information Delivery Manual ou IDM (ISO 29481-1, 2016). Selon la norme IDM, la syntaxe formelle à utiliser pour l'échange de données construction est celle précisée par l'Industry Foundation Classes ou IFC (ISO 16739-1, 2018) : il s'agit soit du format EXPRESS, tel que défini dans (ISO 10303-21, 2016) ou du format XML.

Toujours dans un contexte BIM, parmi ces protocoles d'application, nous pouvons aussi mentionner l'AP233 "Représentation des données de l'ingénierie système" dont les concepts correspondants au découpage fonctionnel (ISO/TS 10303-1216, 2008) et au découpage structurel (ISO/TS 10303-1216, 2008) ont été normalisés par le projet PLCS (Product LifeCycle Support) (Eckert, et al., 2005).

5.3. Les approches à base de modèles de connaissances

Comme précisé dans l'introduction de ce chapitre (et illustré dans la Figure 5.1), les modèles à base de connaissances sont des spécifications explicites et formelles de connaissances, employant des langages logiques, et pouvant être interprétées par des algorithmes. Comparées aux modèles de données vus précédemment, les modèles à base de connaissances permettent de spécifier des problèmes dans le contexte d'un domaine métier, sans aucun lien avec une implémentation informatique. On parle dans ce cas de modélisation sémantique. Les modèles à base de connaissances utilisent le concept de "classe" de manière différente par rapport aux modèles orientés objet :

- En modélisation sémantique, une "entité" n'est pas concernée par des opérations, des méthodes ou un comportement (contrairement à un "objet" dans le monde "orienté objet"). Ces éléments appartiennent au domaine de la "modélisation de processus".
- La classe d'une "entité" dans un modèle sémantique n'est pas simplement une classe de l'ensemble "d'entités discrètes avec des limites et identités bien définies" (comme c'est le cas pour UML). Une classe est ici limitée à ce que Richard Barker appelle des classes de "choses ou d'objets d'intérêt, réels ou imaginaires, à propos desquels des informations doivent être connues ou conservées[1]" (Baker, 1990). Nous pouvons résumer cela en disant qu'en modélisation sémantique, une "classe" regroupe les instances ayant la même interprétation (le même sens), qui partagent un ensemble de propriétés.

Selon la famille de langages logiques utilisés, différents types de modèles de connaissances peuvent être définis. Pour les caractériser, deux indicateurs importent: l'expressivité et la décidabilité du langage utilisé. Pour simplifier, nous pouvons définir l'expressivité d'un langage formel comme le nombre d'opérateurs ou de constructeurs pouvant être utilisé pour combiner des concepts afin d'en former des nouveaux (selon les règles de syntaxe du langage). La décidabilité peut être définie comme la capacité d'appliquer des algorithmes de déduction à des modèles décrits avec le langage logique considéré. Un algorithme de déduction doit pouvoir vérifier en un temps fini que tous les énoncés respectent à la fois la syntaxe du langage

[1] "classes of things or objects "of significance, whether real or imagined, about which information needs to be known or held.""

(correctitude du modèle) mais aussi la sémantique associée (la complétude du modèle). Plus un langage logique est expressif, moins il est décidable. En d'autres termes, plus un langage utilise d'opérateurs et de constructeurs logique, moins il est possible de vérifier en un temps fini que les modèles définis avec ce langage sont corrects et/ou complets. Dans un souci d'efficacité d'implémentation, il convient de trouver un langage permettant d'atteindre cet équilibre entre expressivité et décidabilité (Roxin, 2018). Il s'agit des langages de la logique de description (DL ou Description Logics) qui sont utilisés pour la spécification d'ontologies. Dans ce qui suit, nous présentons les approches à base d'ontologies, les éléments manipulées et les normes informatiques en lien.

5.3.1. Présentation de l'approche et des normes associées

Similairement aux modèles à base d'objets, une ontologie comprend des classes, des propriétés et des instances. Toutefois, une ontologie permet de définir des propriétés indépendamment des classes. Les propriétés sont utilisées pour spécifier les conditions qu'une instance doit remplir pour appartenir à une classe. Il s'agit soit de conditions nécessaires (l'appartenance d'une instance à une classe implique le respect de ces conditions par l'instance), soit des conditions nécessaires et suffisantes (l'appartenance d'une instance à une classe équivaut au respect de ces conditions par l'instance).

Une ontologie fait la distinction entre des connaissances assertionnelles (ou instances correspondant au niveau M0 de l'approche MDA) et des connaissances terminologiques (pouvant s'apparenter aux couches de M1 à M3 de l'approche MDA). Une ontologie contient obligatoirement des connaissances terminologiques (ou TBox). Si une ontologie contient aussi des connaissances assertionnelles (ou ABox), elle est appelée base de connaissances. Un algorithme appelé raisonneur peut être exécuté soit sur une ontologie, soit sur une base de connaissances. Les vérifications et les déductions associées à chaque type de connaissances sont illustrées dans la Figure 5.3. Un tel algorithme permet de déduire et de matérialiser (dans l'ontologie ou dans la base de connaissances) l'ensemble des connaissances non explicites initialement. Ce processus s'appelle l'inférence, et c'est pour cela que l'algorithme permettant de l'implémenter est appelé moteur d'inférence. Enfin lors d'une implémentation d'une approche reposant sur une telle base de connaissances, une interface graphique est souvent ajoutée afin de faciliter l'interaction avec la base de connaissances et le moteur d'inférence (voir Figure 5.3).

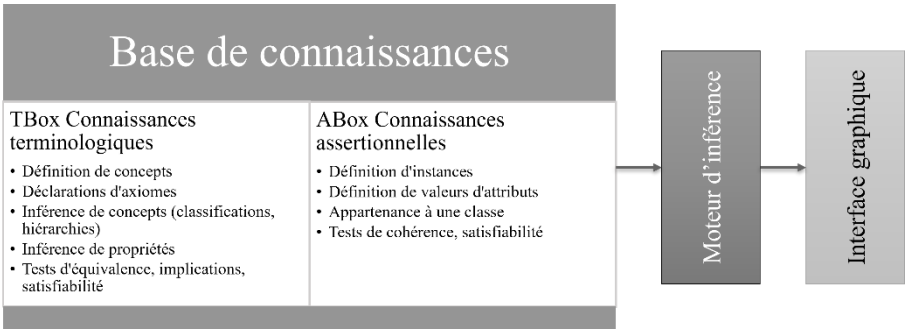

Figure 5.3. *Architecture générale des approches à base de modèles de connaissances.*

Les langages informatiques utilisés pour spécifier et implémenter des ontologies sont définis et normalisés par le W3C (World Wide Web Consortium). L'ensemble de ces langages et technologies constitue les technologies dites du Web sémantique pour lesquelles le W3C a défini une architecture en couches (voir Tableau 5.1). Nous la présentons et résumons les langages associés dans les paragraphes suivants.

	Normes
Couche modèle	RDF Schema, OWL, SKOS
Couche échange d'informations	RDF (Resource Description Format)
Couche identification et encodage	URI (Uniform Ressource Identifier), Unicode

Tableau 5.1. *Normes internationales utilisant une modélisation à base d'objets*

Pour mieux comprendre le Web sémantique, nous commençons par introduire les principes des "données liées" (ou Linked Data). L'approche dite des "données liées" représente un sous-ensemble de principes et de technologies du Web sémantique, visant le partage et la réutilisation des données à l'échelle du Web. En effet, il s'agit de reprendre les principes de l'architecture du Web actuel, et les étendent afin de décrire des connaissances (voir Tableau 5.2). Comme pour le Web classique, le Web sémantique repose sur l'utilisation d'URI en tant que mécanisme d'identification unique et globale des ressources, à l'échelle du Web. Le protocole http est lui aussi, comme pour le Web traditionnel, le mécanisme d'accès universel aux ressources. La différence réside au niveau de la description des ressources. Alors que le Web classique utilise HTML pour les pages Web, le Web sémantique repose sur le modèle RDF (Hayes, et al., 2014), fondé sur une structure en graphe et permettant d'utiliser des liens typés (dépassant ainsi les limites des liens <a href> utilisés en HTML). Un modèle RDF est un graphe orienté et étiqueté de ressources identifiées par le biais

d'URIs. Les ressources sont contenues dans des triplets respectant la forme <Sujet Prédicat Objet>. Par exemple le triplet <Paris est_la_capitale_de France> relie le sujet "Paris" à l'objet "France" par la propriété "est la capitale de" (Roxin, 2018). Chacune de ces éléments (le sujet, le prédicat et l'objet) sont identifiés par une URI http à l'échelle du Web. Dans ce contexte (Web), deux éléments sont considérés identiques par un raisonneur si et seulement si leurs identifiants (donc les chaînes de caractères composants leurs URIs) sont identiques (Roxin, 2018).

	Web traditionnel	Web sémantique
Description	HTML	RDF
Accès	http (HyperText Transfer Protocol)	
Identification	URI (Uniform Ressource Identifier)	

Tableau 5.2. *Les principes du Web classique comparés aux principes des données liées.*

C'est suivant les principes d'identification et accès ci-dessus que tous les éléments de RDF, des langages de description d'ontologies et de toute ontologie définie avec ces langages sont publiés sur le Web et identifiés avec des URI http. Afin d'éviter à avoir à écrire des chaines de caractères trop longues, il est possible d'utiliser le schéma d'abréviation d'URI appelé *qname*. Dans sa forme la plus simple, une URI exprimée selon ce schéma comprend 2 parties : un nom de domaine et un identifiant, séparées par ":". Ainsi, le préfixe "rdf" correspond au nom de domaine où est publié le modèle RDF, c'est-à-dire à la chaîne "http://www.w3.org/1999/02/22-rdf-syntax-ns#". Ainsi rdf:Resource représente une abréviation de l'URI http complète permettant d'identifier le concept Resource tel que défini en RDF (http://www.w3.org/1999/02/22-rdf-syntax-ns#Resource).

Les langages de description d'ontologies viennent en complément de ces principes, et impliquent tous l'utilisation du modèle RDF. En effet, RDF est le modèle générique pour la description de ressources, notamment à travers son concept de plus haut niveau rdf:Resource. En RDF, tout est ressource : RDF ne permet pas de faire la différence entre une instance et une classe. Cette fonctionnalité abstraite de RDF lui permet d'être le candidat idéal pour l'intégration de données hétérogènes. Cependant, lorsqu'il s'agit de décrire des connaissances dans un domaine métier, des termes complémentaires sont nécessaires. Ils sont spécifiés au travers des langages RDF Schema et OWL (Web Ontology Language).

RDF Schema (Brickley, et al., 2014) introduit un vocabulaire au-dessus de RDF, avec différents termes permettent de caractériser davantage les énoncés formés avec RDF. Pour identifier des termes de RDF Schema, le préfixe rdfs est utilisé et il

correspond au nom de domaine "http://www.w3.org/2000/01/rdf-schema#". Avec RDF Schema il est possible de définir des classes (rdfs:Class), des sous-classes (rdfs:subClassOf), des sous-propriétés (rdfs:subPropertyOf) ainsi que des ensembles de départ (rdfs:domain) et d'arrivée (rdfs:range) pour les propriétés. La figure ci-dessous (Figure 5.4) illustre les nouveaux termes apportés par RDF Schema par rapport à RDF. Ainsi une instance peut appartenir à une classe (via la propriété rdf:type). Une classe et une propriété demeurent sous-classes du concept abstrait rdf:Resource.

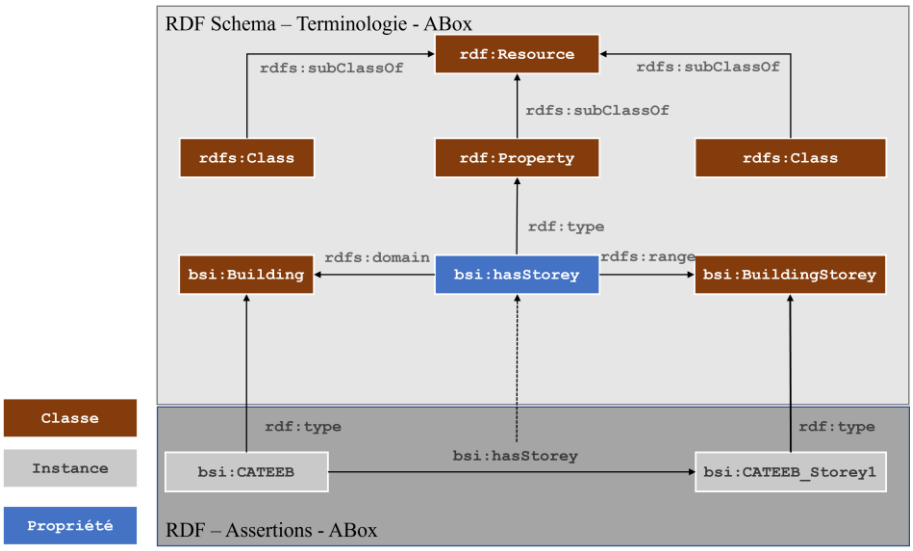

Figure 5.4. *Illustration des nouveaux concepts apportés par le langage RDFS par rapport à RDF (Roxin, 2018)*

Malgré ses apports, RDF Schema a des limites qui n'en font pas un candidat adapté pour la modélisation de connaissances complexes. Supposons le modèle décrit ci-dessous (voir Figure 5.5), à savoir un bâtiment a des étages qui peuvent être de deux types: soit "sous-sol" (bsi:Basement), soit "étage" (bsi:Floor). Pour spécifier la classe d'un bâtiment sans sous-sol (bsi:BuildingWithoutBasement), il faudrait pouvoir spécifier une restriction sur la propriété bsi:hasStorey (ne doit pas contenir d'étage de type sous-sol). RDF Schema ne permet pas d'ajouter ce type de contrainte. C'est la famille de langages OWL qui permet de combler ces manques.

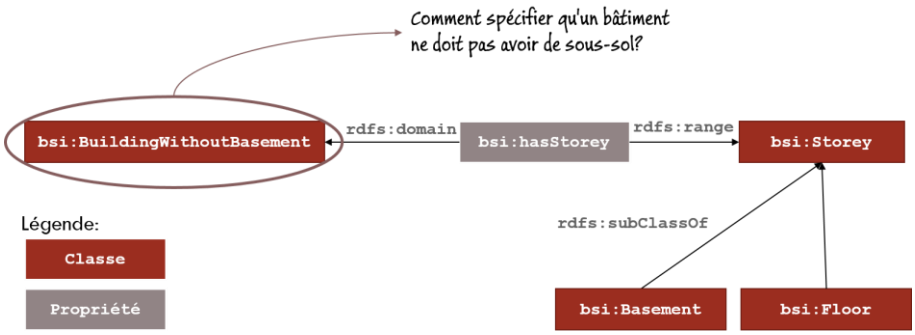

Figure 5.5. *Illustration des limites de RDFS (Roxin, 2018)*

Les langages OWL (Bechhofer, et al., 2004) utilisent des langages de la famille DL pour axiomatiser des connaissances d'un domaine métier donné. Selon les constructeurs DL implémentés différents profils OWL sont spécifiés par le W3C. Leur étude sort du cadre du présent chapitre. Pour identifier des termes de OWL, le préfixe owl est utilisé; il correspond au nom de domaine "http://www.w3.org/2002/07/owl#". La famille de langages OWL définit deux types de propriétés: a) les propriétés dites d'objet (owl:ObjectProperty), et b) les propriétés dites de type de données (owl:DatatypeProperty). Ces dernières sont utilisées dans des triplets ayant comme objet une valeur ou un type de données, alors que les propriétés d'objet sont utilisées dans des triplets ayant des classes ou des instances de classes comme objet. OWL fournit aussi des propriétés permettant de spécifier des "liens de synonymie" entre classes, propriétés et instances (respectivement owl:equivalentClass, owl:equivalentProperty et owl:sameAs). OWL permet aussi de définir des restrictions sur le propriétés, par rapports à leurs valeurs (owl:allValuesFrom, owl:someValuesfrom, owl:hasValue) comme par rapport à leurs cardinalités (owl:minCardinality, owl:maxCardinality, owl:cardinality).

Pour requêter les modèles de connaissances spécifiés avec RDF Schema ou OWL, le W3C a spécifié le langage SPARQL (SPARQL Protocol and RDF Query Language) (Harris, et al., 2013). SPARQL comporte à la fois un protocole d'adressage de requêtes via http et la spécification d'un langage permettant de composer de telles requêtes. SPARQL représente l'équivalent de SQL (Structured Query Language) pour les modèles de connaissances.

5.3.2. Discussion

Les technologies sémantiques fournissent des moyens génériques et flexibles favorisant la découverte et l'intégration de données à partir de sources de données

réparties sur le Web. Lorsque comparées aux approches de modélisation à base d'objets (e.g. API, bases de données relationnelles), les approches à base de connaissances (modélisation sémantique) présentent différents avantages, résumés ci-dessous.

Traditionnellement, la modélisation des orienté objet démarre avec la définition d'une structure (schéma) qui est ensuite instancié par l'utilisateur avec des données réelles (issues du Web, d'applications, de bases de données, etc.). Avec les approches sémantiques il est toujours possible de fonctionner ainsi, mais il est surtout possible d'inverser ce processus. Il est en effet possible de partir directement des données (indépendamment de leur taille ou niveau d'hétérogénéité) et soit manuellement, soit automatiquement, les affecter à des classes (soit définies au préalable soit générées à partir des données elles-mêmes si couplage avec des approches machine learning).

Le modèle de données RDF est un langage abstrait pouvant être utilisé pour définir des instances de données, des structures de données ainsi que les relations entre elles. Dans d'autres approches, deux langages auraient été nécessaires pour cela. Par exemple, dans l'approche STEP (ISO 10303, 1994), on utilise le langage EXPRESS pour les structures de données et le langage SPFF (STEP Physical File Format) pour les instances de données (ISO 10303-21, 2016). RDF permet de combiner des déclarations à plusieurs niveaux à travers son concept de plus haut niveau (rdf:Resource) qui peut être défini à tout méta niveau.

Dans les approches MDA (y compris pour les technologies STEP e.g. EXPRESS/SPFF/SDAI), on définit un méta-concept de base auquel sont reliés les autres concepts. Par exemple, en EXPRESS, le méta-concept de base est le concept Entity qui implémente des attributs et des contraintes. La définition d'un attribut se fait toujours dans le contexte d'un concept Entity. Or, ce n'est pas le cas avec les approches sémantiques, où les classes et les propriétés d'une ontologie sont indépendantes entre elles. Il est dès lors possible de spécifier des propriétés et contraintes en dehors de toute classe et de les associer à différentes classes.

En modélisation sémantique, l'hypothèse de départ stipule que l'absence d'information dans une ontologie n'est jamais interprétée comme négative (Open World Assumption). Ceci est appelée modélisation en monde ouvert et s'oppose à la modélisation en monde clos, qui fait l'hypothèse contraire (Closed World Assumption) et qui caractérise les approches orientées objet. En monde ouvert, si on spécifie qu'un bâtiment contient un étage, puis qu'on exécute la requête "combien d'étages a le bâtiment", le raisonneur ne pourra pas répondre. En effet, le bâtiment peut contenir d'autres étages, et cette connaissance n'a seulement pas été spécifiée dans l'ontologie. Si nous souhaitons avoir une réponse à la requête formulée, il faudrait spécifier que le bâtiment ne contient qu'un seul étage. Le fonctionnement en monde ouvert est une conséquence directe de l'idée à la base du Web "tout le monde peut tout

dire sur tout", mais aussi lié au fait que la connaissance humaine est par défaut incomplète.

Les ontologies définies en utilisant les langages standards du Web sémantique sont conçues afin d'être spécifiés, étendues et maintenues, de manière distribuée et incrémentale ou progressive. Il s'agit en effet de conséquences directes de l'hypothèse du monde ouvert. Lors du démarrage du processus de modélisation de l'ontologie, celle-ci est vide, donc à priori tout est possible. C'est lors du processus (itératif) de modélisation, que des contraintes sont ajoutées ce qui rend l'ontologie plus restrictive. Lorsque l'on travaille en monde ouvert, il s'agit de spécifier ce qui n'est pas possible, interdit ou exclu. Cela correspond à comment nous, êtres humains, fonctionnons dans le monde réel : nous avons l'habitude de gérer les informations incomplètes. Contrairement aux systèmes fonctionnant en monde ouvert, dans un système en monde clos on spécifie de manière explicite ce qui est possible. Dans un tel système, tout ce qui ne peut pas être déduit (ou prouvé VRAI) à partir des connaissances spécifiées sera considéré comme faux. C'est notamment le cas avec les approches orientées-objets.

5.3.3. Normes internationales utilisant ce type de modélisation

La norme ISO/DIS 21597-1:2020 Information Container for Data Drop (ICDD) spécifie une ontologie pour un conteneur permettant de regrouper des documents et des parties de documents (ISO 21597-1, 2020). La première partie de cette norme s'accompagne d'un ensemble de méthodes exploitant les ontologies pour relier des données par ailleurs disjointes au sein de ces documents. La deuxième partie étend les types de liens pouvant être spécifiés entre documents (ISO 21597-2, 2020).

Cette norme a été conçue afin de répondre au besoin, dans l'industrie de la construction, d'une approche uniforme pour l'organisation de l'information lors des échanges de données, en fournissant des outils pour créer des liens sémantiques entre des concepts présents dans des documents distincts.

En effet, dans le secteur de la construction, les livraisons d'informations sont souvent constituées d'une combinaison de plans, de modèles d'information (représentant des éléments bâtis ou naturels de l'environnement physique), de documents textes, de feuilles de calcul, etc. La possibilité de spécifier des relations entre des éléments d'information présents dans des documents différents peut contribuer à augmenter la valeur d'un tel échange de données. La composition d'un tel conteneur découle à la fois des exigences du processus (par exemple la livraison d'informations conformes à l'exécution) et de l'objectif fonctionnel spécifique (par exemple l'exécution d'un devis quantitatif ou la communication d'aspects relatifs aux modèles 3D).

Toutefois les parties 1 et 2 de cette norme se heurtent à de nombreuses limitations, principalement dues au fait que les ontologies définies pour le conteneur de document et les liens entre documents ont été construites à partir de modèles UML. Ces ontologies sont donc inutilement complexes et des implémentations en industrie ont clairement mis en évidence leurs limitations (notamment en termes de raisonnement). En l'état leur utilisation est peu utile encore moins intuitive. Toutefois, la partie 2 de la norme précise qu'il est possible de modifier, étendre ou encore adapter les ontologies du conteneur et des liens. Cela représente une possibilité de non seulement utiliser correctement les langages de modélisation d'ontologie (e.g. OWL) afin d'augmenter l'efficacité d'implémentation mais aussi d'ajouter de fonctionnalités supplémentaires afin d'adapter le conteneur selon un objectif donné. La simplification des ontologies sous-jacentes pourra faciliter le développement de logiciels innovants tout en restant conformes à la norme.

5.4. Les approches hybrides

Plusieurs chercheurs ont investigué l'adaptation des approches MDA en approches de modélisation à base d'ontologies. Ceci a donné naissance à la branche de recherche MDD ou *Model-driven Development*. (Djurić, et al., 2005) (Djurić, et al., 2005) définissent une architecture d'ontologies basée sur l'approche MDA. Cette architecture inclut la spécification d'un métamodèle d'ontologies et d'un profil UML pour les ontologies. Une transformation de l'ontologie UML en OWL a été implémentée. L'approche générale est illustrée dans la figure ci-dessous (voir la Figure 5.6).

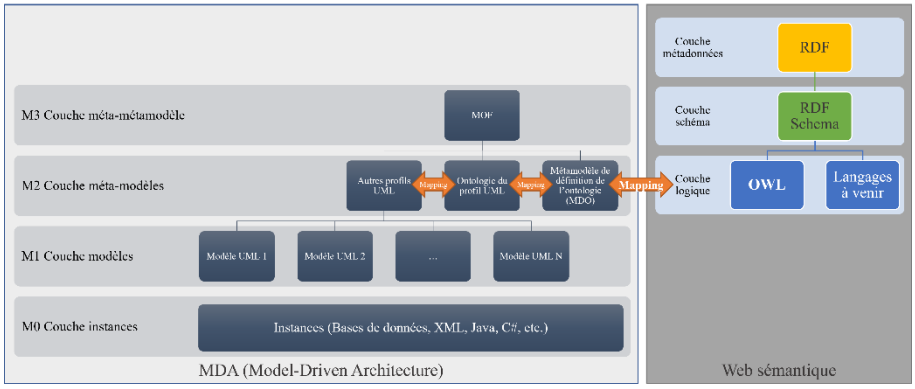

Figure 5.6. Illustration des correspondances potentielles entre MDA et OWL - adapté d'après (Djurić, et al., 2005)

ISO 19150-2 :2015 définit un ensemble de règles pour la traduction de modélisations UML vers OWL (ISO 19150-2, 2015). Toutefois, les différentes implémentations de ces traductions ont prouvé leurs limites. Les modèles OWL ainsi générés ne permettent pas une implémentation efficace et démontrent des performances faibles d'un point de vue raisonnement. Ceci est principalement dû aux différences conceptuelles entre la modélisation UML et la modélisation OWL, telles que résumées dans les paragraphes précédents. Nous pouvons ajouter ici, qu'en UML une classe est toujours l'union de ses sous-classes, alors que ceci n'est pas vrai en OWL. Or, lorsque des ontologies OWL sont générées à partir de représentations UML, les classes OWL sont définies en tant qu'union de leurs sous-classes (owl:unionOf). Ce type d'axiome impacte négativement les performances des raisonneurs, sans mentionner que cela va à l'encontre des principes de conception OWL. Malheureusement ce type d'approche (traduction automatique de UML vers OWL) est de plus en plus appliqué.

Toutefois, il existe des approches qui appliquent correctement des transformations horizontales entre ces deux approches de modélisation. Nous pouvons citer l'approche spécifiée dans le cadre de la directive INSPIRE (Ansorge, et al., 2016). Le langage OWL peut être considéré pour la spécification de métamodèles et schémas conceptuels. UML peut être utilisé pour représenter ces mêmes éléments. Par contre un alignement entre deux schémas conceptuels est à définir de OWL vers UML et non l'inverse.

Enfin, l'architecture STEP (ISO 10303, 1994) prévoit une traduction des concepts SysML en OWL afin de pouvoir exprimer le modèle conceptuel STEP en OWL. Les ontologies sont aussi prévues pour la définition des dictionnaires et autres ressources intégrées dans un projet BIM. Une étude détaillée de ces approches sort du cadre de ce chapitre.

5.5. Conclusion

Malgré la multitude d'approches pour simuler une intelligence humaine, les ordinateurs rencontrent des difficultés à gérer les connaissances comme le font les humains. Ceci est principalement dû au fait que les connaissances sont des informations interprétées par rapport à un contexte et une expérience. Dans des situations différentes, deux informations seront interprétées différemment par un humain. Pour qu'un ordinateur puisse effectuer une interprétation similaire, il est nécessaire de lui spécifier un modèle explicite dans un langage formel.

Dans ce chapitre, nous avons présenté deux principales approches pour la modélisation des systèmes complexes: les approches à base de modèles objet et les

approches à base de modèles de connaissances. Issues de courants différents de l'informatique, ces deux approches permettent de répondre à des problématiques différentes, problématiques qu'il est important de bien définir afin de pouvoir utiliser l'approche adaptée. Les approches à base de modèles objets ont un lien fort avec l'implémentation informatique du système considéré. Elles sont dites semi-formelles, car elles sont une représentation (graphique) d'une implémentation (informatique) d'un système complexe. Les connaissances (c'est-à-dire l'interprétation à associer aux informations modélisées) ne sont pas modélisées de manière explicite, elles sont sous-entendues. Pour une interprétation correcte d'un tel modèle objet un expert humain est nécessaire. Ce n'est pas le cas avec les modèles à base de connaissances: ceux-ci reposent sur l'utilisation de langages logiques formels pour spécifier de manière explicite des connaissances. Les modèles à base de connaissances sont définis sans aucun lien avec une quelconque implémentation informatique. Un ordinateur peut interpréter ces modèles (de manière similaire à un utilisateur humain) et des connaissances implicites peuvent être inférées, en utilisant des algorithmes de raisonnement.

Du fait des différences entre ces deux approches, si des transformations sont à considérer de l'une vers l'autre, il est critique de les appliquer dans le bon sens, c'est-à-dire des modèles formels (à base de connaissances) vers les modèles semi-formels (à base d'objets). Ces transformations devraient toutefois être utilisées avec parcimonie, et par rapport à un cas d'utilisation ou une problématique bien définie. Les deux approches de modélisation correspondent à des conceptions totalement différentes, un modèle OWL n'est donc pas qu'une traduction dans un autre format de fichier d'un modèle UML et vice versa. Ce n'est qu'en prenant conscience de ces différences et en utilisant l'approche de modélisation adapté à la problématique visée qu'une utilisation efficace des modèles résultants pourra être faite.

Car même si les modèles à base de connaissances permettent de spécifier l'interprétation d'informations dans un certain contexte, il faut encore pouvoir définir des liens entre les interprétations à avoir dans différents contextes. Dans le domaine du jumeau numérique, ceci est d'autant plus crucial qu'il faut pouvoir gérer différentes abstractions (ou vues) de la connaissance de l'ouvrage (ou du bâtiment) à différents moments de son cycle de vie. Il est nécessaire de pouvoir basculer de l'une à l'autre et assurer un continuum d'interprétation entre acteurs, pour chaque abstraction considérée. Ceci n'est possible que si des liens sémantiques (comme des liens de synonymie par exemple) sont définis entre les éléments des modèles de connaissance sous-jacents. La construction des modèles objet associés assurera l'implémentation informatique adaptée de ce continuum d'interprétation.

Bibliographie

Ansorge, Christian, et al. 2016. The INSPIRE Model-Driven Approach. *INSPIRE Data Specification Extensions.* [En ligne] 2016. http://inspire-extensions.wetransform.to/inspire-mda.html.

Baker, Richard. 1990. *CASE*Method: Entity Relationship Modeling.* Wokingham, England : Addison-Wesley, 1990.

Bechhofer, Sean, et al. 2004. OWL Web Ontology Language Reference. [En ligne] 10 Février 2004. https://www.w3.org/TR/owl-ref/.

Booch, G., Rumbaugh, J. et Jacobson, I. 1998. *The Unified Modeling Language User Guide.* Massachusetts : Addison-Wesley, 1998.

Brickley, D. et Guha, R.V. 2014. RDF Schema 1.1 . [En ligne] 25 Février 2014. https://www.w3.org/TR/rdf-schema/.

Budinsky, Frank, et al. 2003. *Eclipse Modeling Framework.* s.l. : Addison-Wesley, 2003.

Coad, Peter et Yourdon, Edward. 1990. *Object-Oriented Analysis.* Englewood Cliffs, NJ : Yourdon Press, 1990.

Djurić, D, et al. 2005. A UML Profile for OWL Ontologies. *Model Driven Architecture (MDAFA), Lecture Notes In Computer Science.* Springer, Berlin, Heidelberg., 2005, Vol. 3599.

Djurić, D., Gašević, D. et Devedžić, V. 2005. Ontology Modeling and MDA. *Journal of Object Technology.* 2005, Vol. 4, 1, pp. pp. 109-128.

Eckert, R, Mansel, W. et Specht, G. 2005. Model Transfert among CASE Tools in System Engineering. *Systems Engineering.* Wiley Periodicals, 2005, Vol. 8, 1.

Harris, Steve, Seaborne, Andy et Prud'hommeaux, Eric. 2013. SPARQL 1.1 Query Language. [En ligne] 21 Mars 2013. https://www.w3.org/TR/sparql11-query/.

Hayes, P. J. et Patel-Schneider, P. F. 2014. *RDF 1.1 Semantics.* [En ligne] 25 Février 2014. https://www.w3.org/TR/rdf11-mt/.

Holt, Jon, et al. 2012. Model-based requirements engineering for system of systems. *Proceedings of the 7th International Conference on System of Systems Engineering (SoSE).* 2012, doi:10.1109/SYSoSE.2012.6384145.

INSPIRE. 2013. INSPIRE Generic Conceptual Model. *D2.5: Generic Conceptual Model.* [En ligne] 5 Avril 2013.

https://inspire.ec.europa.eu/documents/Data_Specifications/D2.5_v3.4rc3.pdf. D2.5_v3.4rc3.

ISO 10303. 1994. Systèmes d'automatisation industrielle et intégration — Représentation et échange de données de produits. 1994.

ISO 10303-21. 2016. Systèmes d'automatisation industrielle et intégration — Représentation et échange de données de produits — Partie 21: Méthodes de mise en application: Encodage en texte clair des fichiers d'échange. s.l. : ISO/TC 184/SC 4 Données industrielles, 2016.

ISO 16739. 2018. Classes IFC pour le partage des données dans le secteur de la construction et de la gestion de patrimoine. 2018.

ISO 16739-1. 2018. Classes IFC pour le partage des données dans le secteur de la construction et de la gestion de patrimoine — Partie 1: Schéma de données. s.l. : ISO/TC 59/SC 13 Organisation et numérisation des informations relatives aux bâtiments et ouvrages de génie civil, y compris modélisation des informations de la construction (BIM), 2018.

ISO 19150-2. 2015. Information géographique — Ontologie — Partie 2: Règles pour le développement d'ontologies dans le langage d'ontologie Web (OWL). s.l. : ISO/TC 211 Information géographique/Géomatique, 2015.

ISO 21597-1. 2020. Conteneur d'informations pour la livraison de documents liés — Spécification d'échange — Partie 1: Conteneur. 2020.

ISO 21597-2. 2020. Conteneur d'informations pour la livraison de documents liés — Spécification d'échange — Partie 2: Types de liens. 2020.

ISO 29481-1. 2016. Modèles des informations de la construction — Protocole d’échange d’informations — Partie 1: Méthodologie et format. s.l. : ISO/TC 59/SC 13 Organisation et numérisation des informations relatives aux bâtiments et ouvrages de génie civil, y compris modélisation des informations de la construction (BIM), 2016.

ISO/DIS 8000-1. 2011. Qualité des données — Partie 1: Aperçu. 2011.

ISO/IEC 19505. 2012. Technologies de l'information — Langage de modélisation unifié OMG (OMG UML). 2012.

ISO/IEC 19505-2. 2012. *Technologies de l'information — Langage de modélisation unifié OMG (OMG UML) — Partie 2: Superstructure.* 2012.

ISO/TS 10303-1214. 2004. Systèmes d'automatisation industrielle et intégration — Représentation et échange de données de produits — Partie 1214: Module d'application: Décomposition de système. 2004.

ISO/TS 10303-1216. 2008. Systèmes d'automatisation industrielle et intégration — Représentation et échange de données de produits — Partie 1216: Module d'application: Décomposition en éléments fonctionnels. 2008.

Long, David et Scott, Zane. 2011. A Primer for Model-Based System Engineering. *Vitech.* [En ligne] 2011. http://www.ccose.org/media/upload/MBSE_Primer_2ndEdition_full_Vitech_2011.10.pdf.

Miller, J. et Mukerji, J. 2003. MDA Guide Version 1.0. *OMG Document: omg/2003-05-01.* [En ligne] Mai 2003. http://www.omg.org/mda/mda_files/MDA_Guide_Version1-0.pdf.

2014. MOF Specification v2.4.2. *OMG Document.* [En ligne] Avril 2014. https://www.omg.org/spec/MOF/2.4.2.

OMG. 2002. Systems Modeling Language (SysML), Version 1.2. [En ligne] 6 Octobre 2002. http://www.sysml.org/docs/specs/OMGSysML-v1.2-10-06-02.pdf.

Roxin, Ana. 2018. *Raisonnement et connaissances – à la recherche de l'équilibre entre expressivité et efficacité.* Dijon : Habilitation à diriger des recherches, Université de Bourgogne Franche-Comté, 2018. https://hal.archives-ouvertes.fr/tel-01940781.

Rumbaugh, James, Booch, Grady et Jacobson, Ivar. 1999. *The Unified Modeling Language Reference Manual.* s.l. : Addison-Wesley, 1999. 9780201309980.

Tuyen, Frank. 2006. The Fast Guide to Model Driven Architecture, The Basics of Model Driven Architecture (MDA). *Cephas Consulting Corp.* [En ligne] Janvier 2006. https://www.omg.org/mda/mda_files/Cephas_MDA_Fast.

XMI, OMG. 2014. XML Metadata Interchange Version 1.1. *Object Management Group, Needham, MA, USA.* [En ligne] December 2014. http://www.omg.org/spec/XML/.