Contribution à une modélisation ontologique des informations tout au long du cycle de vie du produit

Virginie Fortineau

2014-ENAM-0049

École doctorale n° 432 : Science des Métiers de l'ingénieur

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École Nationale Supérieure d'Arts et Métiers

Spécialité " Génie industriel "

présentée et soutenue publiquement par

Virginie FORTINEAU

le 18 novembre 2013

Contribution à une modélisation ontologique des informations tout au long du cycle de vie du produit.

Directeur de thèse : **Samir LAMOURI**
Co-encadrement de la thèse : **Thomas PAVIOT**

Jury

M. Damien TRENTESAUX, Professeur, Université de Valenciennes et du Hainaut-Cambrésis	Président
M. Alain BERNARD, Professeur, Ecole Centrale de Nantes	Rapporteur
M. Bernard GRABOT, Professeur, Ecole Nationale d'Ingénieurs de Tarbes	Rapporteur
M. Dimitris KIRITSIS, Professeur, Ecole Polytechnique Fédérale de Lausanne	Rapporteur
M. Michel GOURGAND, Professeur, Université de Clermont-Ferrand	Examinateur
M. Thomas PAVIOT, Docteur, Arts et Métiers Paristech	Examinateur
M. Samir LAMOURI, Professeur, Arts et Métiers Paristech	Examinateur
M. Jean-Louis GOBLET, Ingénieur, EDF	Invité

T H È S E

Arts et Métiers ParisTech - Centre de Paris

REMERCIEMENTS

Je tiens à remercier dans un premier temps le jury scientifique, qui me fait l'honneur d'examiner ces travaux de recherche : Alain Bernard, avec lequel j'ai eu l'occasion d'échanger à plusieurs reprises, Bernard Grabot, Michel Grougand, Damien Trentesaux et Dimitris Kiritsis. J'ai apprécié les échanges que j'ai pu avoir avec Dimitris lors de sa venue en France et je le remercie pour le temps qu'il m'a accordé.

Je remercie également Jean-Louis Goblet de me faire l'honneur de participer à ce jury. L'aboutissement de ce travail doit beaucoup à la relation de confiance qu'il a su instaurer pendant le projet EDF/ARTS ainsi qu'à la qualité scientifique et intellectuelle des échanges qu'ils nous a permis d'avoir. Je souhaite également adresser un remerciement particulier à Françoise Boutin pour la confiance qu'elle a placé en nous et qui n'a jamais failli, tout au long de ce que nous appelons entre nous « le projet EDF » ; de même, j'ai une pensée pour l'ensemble des membres de l'équipe EDF, qui sont des partenaires de travail agréables et enrichissants.

Je tiens ensuite à remercier Samir Lamouri et Thomas Paviot, qui ont formé un duo idéal tout au long de cette thèse. Samir m'a probablement plus appris encore sur l'histoire de France et la littérature que sur la science elle-même, et c'est pour ces trois années côte à côte, dans leur entièreté, que je tiens à le remercier. Nous avons affronté ensemble de nombreuses batailles, et des moments exceptionnels, qui ont contribué à rendre cette période si riche ; parce qu'aussi, nous partageons les mêmes valeurs, et la même curiosité pour ce qui fait la vie, au-delà de la sphère professionnelle. Thomas est pour moi un exemple, dont j'espère à l'avenir m'imprégner. Je sais la chance qui fut la mienne de l'avoir à mes côtés durant ces trois années. Il a su allier rigueur intellectuelle et scientifique, droiture et exigence tout en me laissant « m'exprimer » pleinement tout au long de ces

années de recherche. C'est un équilibre que je sais difficile à trouver, et rare.

Mes pensées vont ensuite à tous mes collègues de cette merveilleuse école que sont les Arts et Métiers. Je ne peux tous les citer ici, mais je pense particulièrement aux membres de l'équipe « LOGIL ». Aux « anciens » qui nous ont laissé un patrimoine, et particulièrement à Robert Canonne, qui fut le premier à m'accorder une chance. A ceux qui ont pris la relève et qui font du travail au quotidien un plaisir : Simon, Ronan, Andrée-Anne, Niccolo et tous les collègues de province, et ceux qui viendront bientôt compléter la fine équipe.

Je tiens particulièrement à remercier Roch Bertucat et Xenia Fiorentini, pour tout ce qu'ils m'ont apporté scientifiquement, et pour l'accueil chaleureux qu'ils m'ont réservé à Rome l'hiver dernier.

J'ai une pensée pour mes élèves, qui sont une source d'inspiration, de remise en question et de progrès intarissable.

Enfin, je remercie mes parents, pour l'éducation qu'ils m'ont donné, pour avoir toujours soutenu mes ambitions et surtout pour m'avoir appris ce qui est le plus important pour l'épanouissement d'un enfant : que l'échec n'est pas une fin en soi mais un droit, et qu'il faut seulement s'efforcer à le dépasser.

TABLE DES MATIÈRES

OBJECTIFS ET APPORTS DE LA THÈSE

Dès le début des années 1980, la bonne gestion des informations s'est révélée un enjeu stratégique pour les entreprises. Jacques Laurent, dans la préface du livre « La gestion de l'information dans l'entreprise » [David and Sutter, 1985] identifie les rôles de l'information dans l'entreprise comme suit : c'est à la fois un outil de création, un outil de travail, un instrument de formation professionnelle, un facteur d'échange entre les intervenants et un patrimoine de « savoir-faire » et de matière grise. Ainsi, la maîtrise de la chaîne d'information est gage de valeur-ajoutée pour l'entreprise, puisqu'elle permet de mieux échanger avec les partenaires, de réutiliser la connaissance métier et enfin de structurer le travail des métiers.

L'émergence des technologies de l'information et de la communication (TIC) au début des années 1990, dont les possibilités et la diffusion n'ont cessé de croître depuis, a changé la donne. Le web, la robustesse des bases de données, le développement des outils applicatifs et la réduction des coûts de stockage ont conduit à l'augmentation quasi-exponentielle des quantités d'informations numériques au sein de l'entreprise.

Une conséquence de ces mutations est que divers outils et méthodes émergent pour permettre d'exploiter, maintenir, stocker ou encore échanger les informations dans les entreprises industrielles : maquette numérique, Système de Gestion des Données Techniques (SGDT), web services, conception intégrée, plate-forme collaborative, etc. Cependant, pris indépendamment les uns des autres, ces outils et méthodes ont un impact très limité sur les verrous et les problématiques liées à l'information. La fédération de l'ensemble de ces outils dans une démarche cohérente est le seul moyen d'en tirer une valeur ajoutée notable. Nous considérons que cette démarche cohérente doit consister en une approche cycle de vie, du type Product Lifecycle Management (PLM) [Terzi et al., 2010] ou, plus

généralement, closed-loop Lifecycle Management [Kiritsis, 2011].

Dans sa thèse, Thomas Paviot [Paviot, 2010] a montré que les enjeux d'une approche cycle de vie étaient aux niveaux tactiques, stratégiques et opérationnels et a identifié comme verrou majeur à l'élaboration d'une démarche PLM la capacité à faire interopérer les systèmes. L'interopérabilité, qui est la capacité de deux systèmes ou plus à collaborer, doit alors se réaliser tant d'un point de vue sémantique, que technique et organisationnel [EIF, 2004]. Dans ces travaux, qui se veulent la continuité des travaux de doctorat de Thomas Paviot, nous nous focalisons sur l'aspect sémantique du problème et postulons que réaliser l'interopérabilité ne suffit pas : encore faut-il que les modèles soient porteurs de sens, c'est-à-dire d'une richesse sémantique suffisante. Ils doivent permettre de transmettre une information source de connaissance.

Pour procurer des modèles riches en sémantique et interopérables, les paradigmes de modélisation doivent évoluer. Depuis quelques années, les ontologies permettent d'obtenir des transformations importantes dans les modélisations de produits grâce à leur possibilités d'expressivité et de raisonnement. Dans cette thèse, nous explorons la possibilité d'une modélisation fondée sur les ontologies, et proposons un cadre de modélisation pour l'ensemble du cycle de vie. La validation des propositions de cette thèse s'effectue par leur application sur un cas d'étude industriel, issu du domaine de l'ingénierie nucléaire. Bien qu'il s'agisse là d'un domaine industriel très particulier, nous proposons un cadre générique de modélisation et une méthodologie reproductible dans d'autres types d'industries. L'industrie nucléaire concentre beaucoup des problématiques actuelles liées à l'information : grande quantité de données, multiplicité des métiers, normes et exigences importantes, longue durée de vie des produits (plus d'un siècle entre la conception et le démantèlement d'une centrale), ensemble du cycle de vie couvert, multiples acteurs. De plus, la taille et les enjeux de l'industrie du nucléaire l'obligent à développer une vision à long terme et lui permettent de peser sur les orientations des éditeurs de logiciels comme des communautés industrielle et scientifique.

STRUCTURE DE LA THÈSE

Outre le présent chapitre, qui positionne les travaux de recherche, la thèse est divisée en cinq parties, délimitées comme suit :

Le chapitre 1 présente le contexte de la recherche. Il positionne tout d'abord le rôle de la modélisation dans la fonction information, définit ensuite les notions de donnée, d'information, de connaissance et de sémantique, et décrit les étapes du cycle de vie d'un produit, ainsi que les approches de gestion du cycle de vie que sont le Product Lifecycle Management et le Closed Loop Lifecycle Management. Il explore ensuite les modèles sémantiques actuels, qu'ils soient *ad hoc* ou standards. Ces modèles s'inscrivent bien souvent dans une démarche unificatrice, dont les limites sont explicitées. D'autres paradigmes de modélisation sont alors à envisager. Nous nous intéressons plus particulièrement aux ontologies d'inférence, et posons ainsi la problématique de recherche : **en quoi les ontologies peuvent permettre de lever les limites des modèles unifiés actuels, en terme de richesse sémantique, d'échange d'information, et de pérennité du flux sémantique tout au long du cycle de vie du produit ?**

Le chapitre 2 présente ensuite les caractéristiques des ontologies, et plus particulièrement celles des langages logiques sur lesquels elles reposent : l'expressivité et le raisonnement. Nous montrons à travers un état de l'art et des études complémentaires que ces deux caractéristiques permettent d'apporter des solutions à trois niveaux : proposer une plus grande richesse sémantique, permettre un échange dynamique des informations et envisager le partage d'une sémantique commune. Enfin, une revue de la littérature est réalisée sur les méthodologies existantes pour la conception de modèles à base d'ontologies.

Le chapitre 3 présente la proposition générale de cette thèse, qui est de deux ordres.

Tout d'abord, un cadre général de modélisation des informations numériques liées au produit est proposé. Il repose sur trois niveaux de modélisation : un niveau méta-modèle pour le partage des données, un niveau de domaine (spécifique à un domaine donné) pour la capitalisation et l'exploitation d'informations sources de connaissance et un niveau d'exécution/application pour leur mise en œuvre. Si la décomposition en plusieurs niveaux des modèles ontologiques existe dans la littérature [Ishak, 2010], la particularité de notre proposition est que chaque niveau de modélisation joue un rôle différent. Le meta-modèle est le modèle de représentation des informations partagées qui, seul, dialogue avec les autres modèles. Les modèles de domaine permettent la gestion et l'intégration des informations, et enfin, les modèles applicatifs mettent en œuvre l'information partagée. L'ontologie de domaine n'est alors plus une étape vers la spécialisation du modèle, sorte de charnière entre la méta-ontologie et les applications, mais un moyen d'apporter de l'information source de connaissance vers le méta-modèle, qui joue alors le rôle de pivot. Une spécification de chacun des niveaux de modélisation est ensuite réalisée. Dans un second temps, ce chapitre présente une méthodologie générale de mise en place du cadre de modélisation, qui aborde la question de la formalisation des informations sous forme de modèles ontologiques.

Dans le chapitre 4 la méthodologie et la formalisation ontologique des informations sont appliquées à un cas d'étude industriel : l'expression et l'exécution de règles métier pour les centrales nucléaires. Dans cet exemple, chaque étape de la méthodologie est illustrée, depuis l'analyse du besoin sémantique et sa conceptualisation, jusqu'à la formalisation sous forme d'ontologies. Cette étude permet de révéler certaines limites des ontologies, qui sont alors discutées, avant de pouvoir appréhender la question de l'implémentation et de valider l'approche par la réalisation d'un démonstrateur.

Le chapitre 5 traite de la validation du cadre de modélisation proposé. Cette validation, passe par la réalisation d'un démonstrateur, qui doit être en mesure de « jouer » un scénario industriel : l'étude d'une ligne de tuyauterie, composant de la centrale. Tout d'abord, les choix d'implémentation sont exposés, ainsi que les raisons de ces choix. De plus, le choix d'une implémentation fondée sur Java oblige à repenser la formalisation des modèles de règles, dont la nouvelle représentation UML est commentée. Ce travail de re-formalisation permet de réaliser une comparaison entre une modélisation UML et une modélisation OWL, et par là-même, de mettre en lumière les apports spécifiques des ontologies pour la modélisation des informations.

Enfin, le chapitre 6 conclut ces travaux de thèse et propose des perspectives de recherche. Ces perspectives se situent à deux niveaux : elles traitent dans un premier temps du cas d'étude industriel, et plus particulièrement, de la question de la modélisation des règles ; puis, des perspectives sur la démarche de modélisation et la modélisation on-

tologique sont exposées, discutant notamment de la levée possible des verrous actuels concernant la modélisation par les ontologies.

CHAPITRE 1

LA MODÉLISATION DES INFORMATIONS AU LONG DU CYCLE DE VIE

Résumé du chapitre

Dans ce chapitre sont délimités la problématique scientifique ainsi que le périmètre des travaux de recherche. Dans la section 1.1 est d'abord définie la notion de modèle, par l'analyse de la place de la modélisation dans la gestion de l'information en général. Puis, les différences entre donnée, information et connaissance sont rappelées, avant de définir la notion de sémantique, qui est le cœur de ces travaux de recherche.

Nous décrivons ensuite, dans la section 1.2, le cycle de vie du produit et les systèmes d'information industriels, avant d'aborder les approches de modélisation qui considèrent l'ensemble du cycle de vie telles que le Product Lifecycle Management (PLM) et le Closed-Loop Lifecycle Management. Ces approches, puisqu'elles impliquent une interaction entre les acteurs, les systèmes et les produits, posent la question de l'interopérabilité, et plus particulièrement l'interopérabilité sémantique. Nous discutons alors des modèles sémantiques existants, principalement d'unification, et notamment des standards, dont nous présentons également les limites.

Afin de lever les verrous des approches unificatrices, la littérature propose des modèles fondés sur les ontologies, comme le montre la section 1.3, qui sont à la base des approches dites « fédératives ». Alors, la problématique de recherche, décrite dans la section 1.4, peut se formuler ainsi : **les ontologies permettent-elles, en tant que nouveau paradigme de modélisation, de dépasser les limites des approches actuelles d'unification, concernant la modélisation des informations au long du cycle de vie du produit ?**

1.1 La modélisation sémantique des informations

1.1.1 La place de la modélisation dans la gestion des informations

La gestion des informations numériques est un processus complexe, qui fait intervenir plusieurs fonctions. Il est important pour la bonne compréhension des travaux de cette thèse d'expliquer clairement la place de la modélisation dans le processus global de gestion des informations. D'après la norme ISO 14258, un modèle est *« une représentation de quelque chose d'autre, exprimée sous forme mathématiques, symbolique ou de mots »*. La figure 1.1 présente un schéma simplifié du processus de gestion des informations. Dans ce processus, les informations sont détenues en premier lieu par l'utilisateur métier, qui les introduit dans un système d'exploitation possédant une Interface Homme-Machine (IHM). Ce système s'appuie sur un modèle de données, qui organise les informations numériques selon les actions de l'utilisateur. Ensuite, les données sont stockées, mises-à-jour et maintenues dans un système de gestion des données, en général des bases de données. On voit alors que le modèle est une étape intermédaire, qui a pour objectif de « représenter » et donc exprimer l'information.

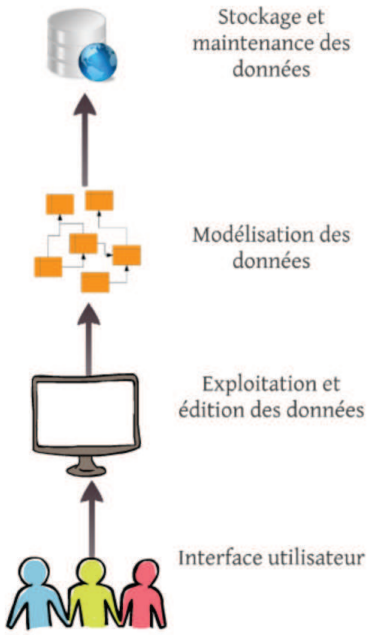


Figure 1.1 – La place de la modélisation dans le processus de gestion de l'information

Ainsi, la gestion des informations a des implications nombreuses, tant sémantiques,

techniques, qu'organisationnelles. Si ces différents points s'influencent nécessairement les uns les autres, il est ambitieux de prétendre les traiter tous à la fois. C'est pourquoi, la présente thèse se focalise uniquement sur la problématique de modélisation des données, considérant les autres enjeux comme des contraintes externes. Les aspects exploitation, stockage et maintenance des données et des informations ne sont donc pas ou peu considérées, tout comme le processus de captation de l'information auprès de l'expert métier (gestion des connaissances).

Parmi l'ensemble des étapes d'un processus de gestion des données, nous considérons que l'étape de modélisation est une étape support : elle est au service de l'exploitation et de la maintenance des données. En effet, un modèle doit à la fois procurer toute la sémantique au système d'exploitation, et assurer la pérennité des données stockées, en leur apportant un sens et en en assurant l'organisation. Par conséquent, l'étape de modélisation est une étape cruciale, qui doit être envisagée en amont des autres : c'est le nœud du processus.

1.1.2 La donnée, l'information et la connaissance

Dans le langage usuel, et même dans la littérature scientifique, les termes « donnée » et « information » sont quelquefois utilisés à même escient. Pourtant, ils ne sont pas synonymes. Nous rappelons ici les différences entre « donnée », « information » et « connaissance », en nous appuyant sur la définition de [Tsuchiya, 1993] :

Définition de l'information [Tsuchiya, 1993]

« Quand on donne un sens à une donnée à travers un cadre d'interprétation, elle devient une information ; et quand une information est lue et comprise à travers un cadre d'interprétation, elle devient de la connaissance. »

Comme le montre la figure 1.2, l'information est une donnée qui a été structurée, organisée selon un modèle prédéfini. En ce sens, l'emploi du terme « information » pour désigner le contenu des systèmes d'information industriels est correct, puisque que les données y sont structurées via les modèles propres de ces systèmes. Elles ont donc une valeur d'information, si tant est que l'on comprenne le modèle en question : par exemple, des informations de CAO auront un sens pour le logiciel de CAO qui les a produites, ou tout autre outil utilisant le même modèle.

La « connaissance », elle, nécessite de replacer l'information dans un contexte afin de l'interpréter vis-à-vis d'un domaine particulier. Une connaissance n'est donc pas intègre

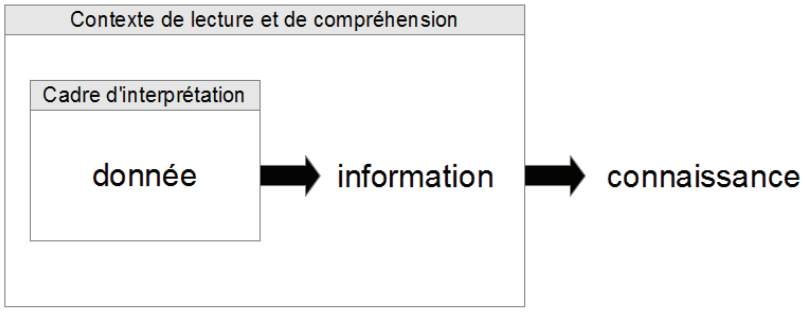


FIGURE 1.2 – Les différences entre donnée, information et connaissance

dans le sens où, hors contexte, elle n'a plus d'existence. De plus, une même information peut générer des connaissances différentes en fonction du domaine dans lequel elle est replacée. Modéliser la connaissance implique donc d'être capable de modéliser le domaine d'application de cette connaissance, et de maintenir vivant le langage employé afin d'éviter toute perte de sens. Ce problème d'unicité de la connaissance, déjà soulevé par [Louis-Sidney, 2011], qui précise dans sa thèse que « la connaissance est la rencontre d'une donnée et d'un individu », pose même le problème « du statut ontologique de la connaissance », c'est-à-dire de son existence propre, puisque la connaissance n'est alors plus une substance objective. Alors, seule la « commensurabilité » des schémas interprétatifs -en d'autres termes la présence d'un minimum de sens commun entre les cadres d'interprétation- peut garantir l'existence d'une connaissance entre plusieurs systèmes.

Exemple des différences entre donnée, information et connaissance

Nous pouvons illustrer les trois définitions précédentes par un exemple simple. 33681423761 est une donnée : c'est une suite de chiffres. Présentée comme suit : +33 6 81 42 37 61, il conviendra à tout lecteur français qu'il s'agit d'un numéro de téléphone. La donnée a été structurée et lue par un système applicatif qui en comprend la structure (le cerveau du lecteur), elle est devenue une information.

Pour devenir de la connaissance, l'information doit être replacée dans un contexte : ici, celui des numéros de téléphones internationaux. La présence de +33 indique à quiconque connaît le domaine des numéros de téléphone qu'il s'agit d'un numéro français. Ensuite, la présence du 6 indique à toute personne connaissant le domaine des numéros de téléphones français qu'il s'agit d'un numéro de téléphone mobile.

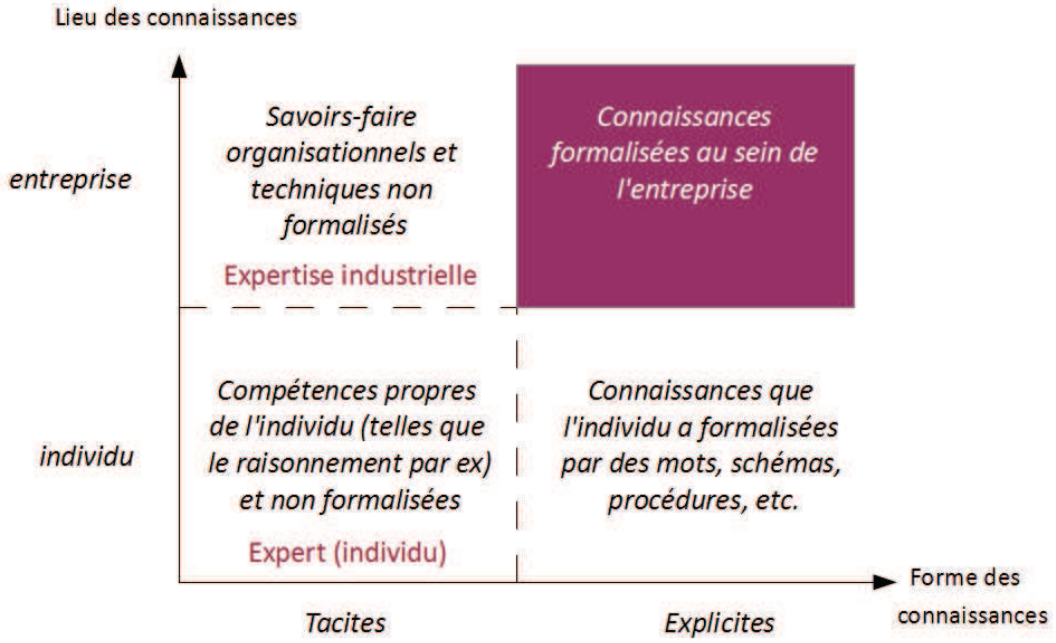


FIGURE 1.3 – Les quatre états statiques de la connaissance

Les aspects statiques et dynamiques de la connaissance

La modélisation des connaissances est donc, en quelques sortes, une prolongation de la modélisation des informations. Il est alors pertinent de s'intéresser à la nature de la connaissance ainsi qu'à sa formalisation dans le but d'en ressortir les particularités vis à vis de l'information. C'est dans ce cadre uniquement que cette section discute des aspects statiques comme dynamiques de la connaissance.

La connaissance peut être classée selon deux critères : sa forme et sa localisation. Les connaissances peuvent en effet être divisées en connaissances tacites (non formalisées) ou en connaissances explicites (formalisées) [Barcikowsky, 2006]. Seules les connaissances explicites peuvent être exploitées par un système d'information, les connaissances tacites étant du domaine de l'expertise (personnelle ou d'entreprise) [Ammar-Khdoja, 2007] : il s'agit de l'ensemble des connaissances que l'on met en œuvre sans réellement s'en rendre compte. De plus, les connaissances peuvent être d'ordre privé (elles sont personnelles), ou bien diffusées dans l'entreprise. La combinaison de ces deux typologies de connaissances produit quatre états statiques possibles de la connaissance, résumés sur la figure 1.3.

Tout l'enjeu de la connaissance est qu'elle puisse être diffusée. Cela n'est possible que quand elle est et formalisée et mise à disposition des acteurs de l'entreprise (état 4, en

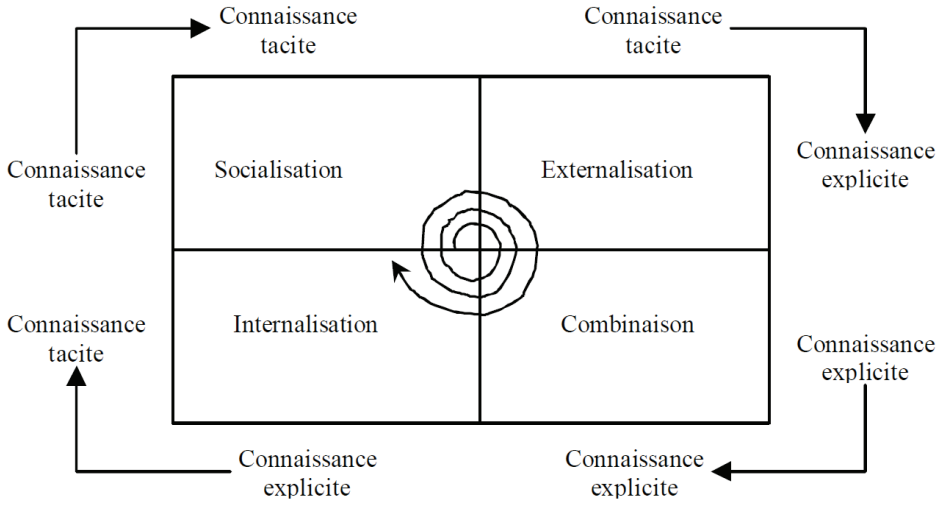


FIGURE 1.4 – Le cycle vertueux de la connaissance [Nonaka and Takeuchi, 1995]

violet sur la figure 1.3). Le processus de transformation des connaissances, généralement appelé « transfert de connaissances » permet d'y parvenir. Ce transfert se fait au travers de différentes actions qui ont été modélisées par [Nonaka and Takeuchi, 1995] sous la forme du *cycle vertueux de la connaissance*. Il peut se réaliser de façon directe, entre individus (transfert de type maître-apprenti) : c'est la *socialisation*. Mais alors la connaissance n'est pas capitalisée. Pour réaliser cette capitalisation, une démarche en trois étapes doit être envisagée (voir figure 1.4) : il faut tout d'abord formaliser les connaissances tacites, afin qu'elles deviennent explicites (c'est l'*externalisation* des connaissances). Ce n'est qu'à cette condition qu'elles pourront passer du domaine de l'individu à celui de l'entreprise, être diffusées (*combinaison*), et finalement, ré-appropriées par d'autres individus (*internalisation*).

Ces transferts successifs de connaissances se heurtent à des difficultés d'ordres cognitif et humain. La difficulté cognitive concerne essentiellement les connaissances « tacites ». Difficiles à structurer et à expliciter, elles sont logiquement difficiles à transférer. Le nœud de cette difficulté est la différence notoire entre le « dire » et le « faire » [Wilson and Schooler, 1991], que l'on peut résumer par la phrase de Polanyi « nous en savons plus que nous ne pouvons le dire » [Polanyi, 1966]. La difficulté humaine, elle, vient du refus potentiel de partager la connaissance : l'expertise étant mise en valeur dans nos modèles d'entreprise, elle peut devenir chasse gardée de l'expert. Malgré ces freins, selon [Bisseret, 1995], les connaissances expertes sont toujours exploitables, car elles sont en réalité organisées en schémas fonctionnels, donc modélisables : l'activité et la procédure y jouent un rôle prépondérant.

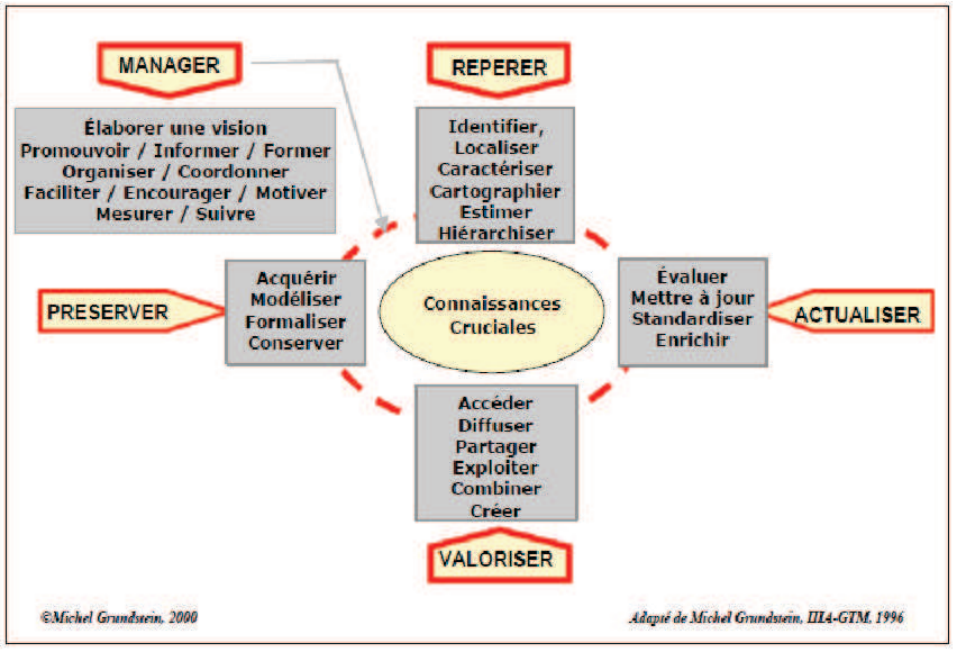


FIGURE 1.5 – Méthodologie de capitalisation de la connaissance [Grundstein, 2000]

Pour réussir le transfert de connaissances et ainsi lever ces verrous, [Grundstein, 2000] propose une méthodologie en quatre phases, illustrée sur la figure 1.5 : il s'agit dans un premier temps de *repérer* la connaissance, c'est-à-dire d'identifier les experts et de cartographier les expertises ; il faut ensuite *préserver* cette connaissance identifiée, notamment en engageant sa formalisation. C'est à cette condition qu'elle peut être *valorisée*, c'est-à-dire partagée et diffusée. Ces différentes étapes nécessitent une structure de gestion des connaissances, qui veillera en dernier lieu à les *actualiser*.

De l'information à la connaissance

Les notions présentées précédemment montrent que passer de l'information à la connaissance n'est pas chose aisée. En effet, si l'information possède un caractère intrinsèque, la connaissance, elle, est multiple et complexe. Partager de la connaissance nécessite au préalable de partager des schémas interprétatifs.

Puisque ces schémas cognitifs sont variés et difficilement formalisables, et que gérer la connaissance nécessite une démarche de préservation et de diffusion, il est difficile de l'intégrer dans un système d'information. On préfèrera souvent formaliser une information *source* de connaissance, c'est-à-dire une information riche en sémantique [Paquet, 2006],

reposant sur des concepts partagés, et dont l'humain, ensuite, pourra faire une connaissance par l'intermédiaire de ses propres schémas interprétatifs. Ces schémas lui seront enseignés parallèlement, afin qu'ils correspondent (soient commensurables) aux autres acteurs de son périmètre de collaboration. La section suivante définit ce que nous entendons par « sémantique » et « information riche en sémantique ».

1.1.3 La sémantique

La sémantique *versus* la syntaxe

Le terme « sémantique » est issu du grec $\sigma\eta\mu\alpha\iota\nu\omega$[1] : « signifier ». Il est souvent opposé au terme « syntaxe », notamment quand il s'agit du domaine de la programmation informatique. La syntaxe est alors la forme du langage de programmation, quand la sémantique en est le fond. Michel Bréal, grand artisan de la sémantique, l'appelle dès son premier *essai de sémantique* « la science des signifiés » [Bréal, 1897]. Jusqu'alors, la syntaxe (ou l'art de la grammatique) était censée expliquer tout langage d'après une position logique. L'introduction de la sémantique permet de se pencher sur le sens des mots, et non plus uniquement sur leurs articulations réciproques. Michel Bréal fait admettre avec la création de la sémantique que si un terme a plusieurs sens, ce n'est pas la faute d'un langage familier, dont il faudrait ignorer l'existence, mais bien parce qu'en tout mot se trouvent deux éléments distincts : les *objets* du langage et leur *forme*, ou, pour employer les termes de la linguistique, le *signifié* et le *référent*. Le référent « roue », par exemple, possède plusieurs signifiés : c'est à la fois un élément de voiture et une figure de gymnastique. Traiter de la sémantique, c'est donc traiter du sens de l'information.

Donner du sens à l'information

Du point de vue de l'information numérique, s'intéresser à la sémantique signifie alors s'intéresser au sens que l'on donne aux données lorsqu'on les structure suivant les modèles d'information. La représentation de l'information numérique est bien souvent fondée sur la logique et la comparaison de termes, c'est-à-dire de référents. Or, le traitement de la sémantique de l'information nécessite d'aller plus loin, et d'établir des liens sémantiques fondés sur le sens, le signifié. Cette dichotomie entre l'objet et son sens relève de la *topique*, opposée dès Aristote à la logique : une information riche d'éléments logiques ne sera pas toujours riche de sémantique, donc de sens pour le métier. De plus, l'analyse des convergences sémantiques entre les points de vue métier ne peut pas se

1. Prononcer « sémaïno »

faire uniquement sur la base d'une comparaison logique des termes ou d'un partage des catégories (au sens d'Aristote : « groupe de termes ») : encore faut-il que chaque catégorie ait la même signification pour chaque acteur. C'est l'enjeu de la modélisation sémantique des informations.

Conclusion de la section 1.1

Cette section définit les frontières de la thèse, afin d'en dessiner le périmètre. La première de ces frontières est celle du modèle : en effet, les travaux de cette thèse se limitent aux problématiques de modélisation des données. Une fois représentées dans un modèle, les données produisent de l'information, qui peut alors être échangée, capitalisée et exploitée. Surtout, si la sémantique du modèle le permet, l'information ainsi formalisée sera source de connaissances pour les utilisateurs. La section suivante délimitent les frontières du terrain que nous envisageons, c'est-à-dire, l'espace et le temps. L'espace, est celui des informations liées à un produit, et le temps, celui du cycle de vie du produit.

1.2 Le cycle de vie et l'interopérabilité des informations

1.2.1 Le cycle de vie du produit et les systèmes d'information

Les informations que nous considérons dans ce travail de recherche sont les informations liées au produit industriel, tout au long de son cycle de vie. [Stark, 2011] décompose le cycle de vie du produit en cinq étapes majeures, sous forme de cinq verbes d'action : imaginer / définir / réaliser / utiliser et maintenir / retraiter et disposer. Ces cinq verbes correspondent aux états communément définis du produit : idée, conception, fabrication, usage et maintenance, démantèlement et recyclage ; et aux acronymes anglais *BOL* (pour Beginning of Life, le BOL regroupant l'idée, la conception et la fabrication), *MOL* (pour Middle of Life, ou utilisation et maintenance) et *EOL* (pour End of Life, soit le démantèlement et le recyclage). Comme le montre la figure 1.6, à l'échelle macroscopique, le cycle de vie du produit semble alors plutôt linéaire.

Les interfaces métiers

On peut se demander pourquoi le cycle de vie est regroupé en cinq étapes principales, et particulièrement celles-ci. Ces cinq étapes délimitent en réalité des interfaces physiques qui sont le lieu de transformations majeures du produit, et par conséquent, elles délimitent des points de vue métiers très différents. Le *métier* peut alors être défini comme :

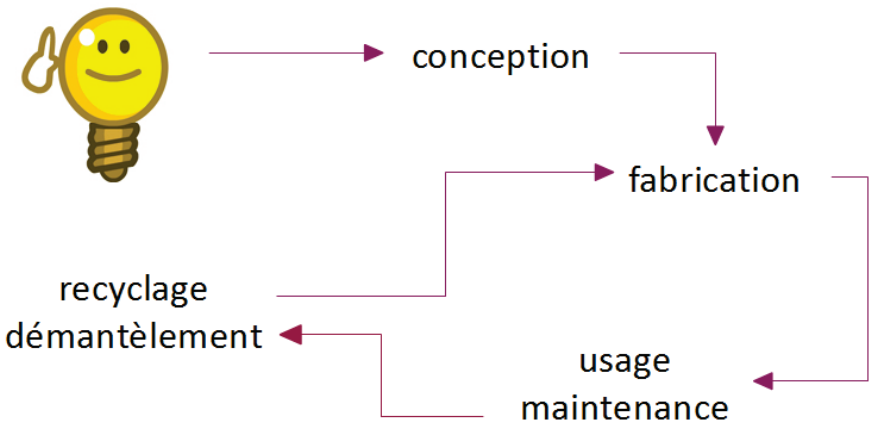


FIGURE 1.6 – Vue macroscopique du cycle de vie du produit

Définition du métier [Paviot, 2010]

« l'ensemble des savoirs et savoirs-faire mobilisés par un opérateur pour mener à bien les activités d'un domaine identifié »

Les interfaces entre les métiers impliqués dans le cycle de vie du produit sont les suivantes [Stark, 2011] :

- de l'idée à la conception du produit, il y a un changement d'acteurs : du marketing et/ou de la création vers les concepteurs. Mais surtout, il y a un changement de point de vue : on passe d'une description des besoins à celle de fonctions ;
- l'interface entre conception et fabrication est le passage du virtuel au réel. D'un produit unique et idéal, défini selon ses fonctionnalités, il faut passer à une somme de pièces, toutes différentes, bien réelles cette fois, qui, assemblées, donneront des produits eux aussi réels et variés, mais répondant aux exigences du cahier des charges. Le virtuel se transforme alors en réel, le « fonctionnel » en « conforme » ;
- entre le BOL et le MOL, c'est-à-dire de la livraison du produit conforme à son usage et sa maintenance, une seule différence, cependant majeure : l'apparition de la variable temps. Le produit s'use, se transforme, est modifié. Il passe de mains en mains et sort généralement du périmètre de l'entreprise qui l'a fabriqué et/ou conçu. L'introduction du paramètre temps à cette étape a des conséquences importantes sur les paradigmes de modélisation de l'information liée au produit ;
- enfin, à l'heure du démantèlement, le produit perd son intégrité : il devient une

somme de morceaux, regroupés non plus suivant des groupes fonctionnels, comme en conception, mais suivant des critères de recyclage : type de matériau, toxicité, biodégradabilité. Comment regrouper ses morceaux ? Comment en connaître la composition ? Tels sont les enjeux que posent aujourd'hui cette étape du cycle de vie.

Le flux d'information et les systèmes d'information

On comprend alors, même à l'échelle macroscopique, que ces points de vue métier divergeants induisent des difficultés de compréhension entre les différents acteurs, et donc, découpent le flux d'information. Le flux d'information peut être de différents types : verbal, écrit, ou numérique. Nous assimilerons par la suite le flux d'information à sa composante numérique, qui est contenue dans les systèmes d'information (SI). Nous entendons par système d'information « l'infrastructure de gestion des informations numériques ». Plus précisément, le Committee on National Security Systems[2] définit le système d'information comme suit :

Définition du système d'information (CNSS)

« un ensemble de ressources organisées pour la collecte, le stockage, le traitement, la maintenance, l'utilisation, le partage, la dissémination, la mise à disposition, l'affichage ou la transmission d'informations »

Le nombre et la diversité des systèmes d'information de l'industrie vont croissant, si bien que les SI de l'industrie peuvent être considérés comme un système de systèmes, où l'information est *distribuée* - selon la définition de [Zimmermann, 2008]. Alors, l'ensemble des SI forment un système complexe [Paviot, 2010], la complexité étant à entendre ici en termes de diversité et non de quantité.

Illustration de la diversité des systèmes d'information

Le tableau ??, qui n'est pas exhaustif, illustre cependant la diversité de systèmes d'information que l'on peut rencontrer tout au long du cycle de vie d'un produit classique, de type manufacturier.

Pendant la phase de conception, l'information liée au produit peut se situer différents types de SI : les SI dits d'authoring, qui regroupent l'ensemble des SI de conception assis-

2. le CNSS a édité le National Information Assurance Glossary, glossaire qui définit les termes liés à la sécurité de l'information, dans le but de fournir un vocabulaire commun.

tée par ordinateur (CAO), de simulation, etc ; les outils de gestion des données techniques, tels que les Product Data Management systems (PDM) par exemple ; ou les outils de pilotage des activités de conception, par exemple MS Project ou PTC Windchill Project-Link.

Phase du cycle de vie	Type d'outils
conception	outil d'authoring (CFAO, simulation, bureautique) gestion des données techniques (PDM) pilotage et collaboration (MS Project, plate-forme)
production	ERP MES APS
transport distribution	TMS DRP APS
maintenance	GMAO
fin de vie	

TABLE 1.1: La diversité des systèmes d'information : un exemple

Pendant la phase de production, il existe principalement trois types d'outils : les Enterprise Resource Planning (ERP) permettent de gérer les fournisseurs, les ressources, les clients et la fabrication. Les ERP constituent « l'arête dorsale » de la plupart des systèmes d'information de production aujourd'hui [Grabot et al., 2008] ; les Manufacturing Execution System (MES) contribuent à gérer et à optimiser le processus de fabrication du produit [Barkmeyer et al., 1999] ; et enfin les Advanced Planning System (APS) permettent de synchroniser les flux de la supply chain [Genin et al., 2005].

Le produit fabriqué est pris en charge par la logistique et entre alors dans le milieu de vie (MOL). Plusieurs types de systèmes d'information permettent de piloter et d'optimiser les activités logistiques. On peut citer les Transport Management Systems (TMS) pour le transport, ou les Distribution Resource Planning (DRP) pour la gestion des entrepôts.

Pendant son utilisation, le produit peut être maintenu en conditions opérationnelles ou réparé, et éventuellement, les informations de maintenance seront sauvegardées dans des SI comme les systèmes de Gestion de la Maintenance Assistée par Ordinateur (GMAO) en général ad hoc. De nouvelles initiatives apparaissent aujourd'hui cherchant à distribuer des fonctionnalités classiquement réservées aux SI directement sur/dans les produits. On parle de SCP (Systèmes Contrôlés par le Produit), de HMS (Holonic Manufacturing Systems), etc. Ces initiatives tendent à rendre actifs les produits en mouvement dans la chaîne logistique, à leur faire capter, porter et distribuer des informations à des fins de réactivité, entre autre.

1.2.2 Les approches cycle de vie dans l'industrie

La maîtrise du système complexe de systèmes que représente l'ensemble des SI industriels est un enjeu majeur pour la compétitivité industrielle. Pour dépasser cette complexité, des approches considérant le produit sur l'ensemble de son cycle de vie ont émergé, notamment le Product Lifecycle Management (PLM), ou, en français, gestion du cycle de vie du produit.

Le product lifecycle management : concept et définitions

Plusieurs définitions du PLM co-existent dans la littérature. Nous retenons ici les définitions sur lesquelles un consensus existe [3], en premier lieu, celle de l'International Journal on Product Lifecycle Management (IJPLM), selon lequel :

Définition du PLM (IJPLM)

« Le PLM est une méthodologie, dont l'enjeu stratégique principal est de permettre la gestion et l'utilisation effectives du capital intellectuel de l'entreprise. »

La définition de l'IJPLM est intéressante car elle introduit la notion de capital intellectuel. Ce terme renvoie à la connaissance et non à l'information, ce qui laisse supposer que le PLM doit évoluer d'une gestion de l'information vers celle de la connaissance. Nous retenons également de la définition de l'IJPLM que le PLM doit être considéré comme une « méthodologie », et non un simple outil, excluant donc la possibilité de définir « un » PLM. Le terme « approche » est souvent employé par d'autres définitions, notamment celle de CIMData, cabinet de conseil dans le domaine du PLM :

Définition du PLM (CIMData)

« Une approche stratégique qui met en œuvre un ensemble cohérent de pratiques permettant de supporter la création collaborative ainsi que l'organisation, la diffusion et l'utilisation des informations relatives à la définition du produit au travers de l'entreprise étendue, de la conception à la fin de vie, et d'intégrer les hommes, les processus, les systèmes d'organisation et d'information. »

---

3. Cette considération tient compte du nombre de citations dans la littérature scientifique

Si « approche » est souvent utilisé à même escient que « méthodologie », c'est à tort : les deux termes ne sont pas synonymes. Le choix du terme « méthodologie » par l'IJPLM est fort, puisqu'une méthodologie va plus loin qu'une simple approche : selon [Zellner, 2011], qui s'appuit sur les travaux de [Braun et al., 2005] et [Winter and Schelp, 2006], une méthodologie doit comporter :

- un modèle procédural : une séquence d'activités à réaliser ;
- un ensemble de techniques : comment générer les livrables en support des activités ;
- des rôles : l'identification des acteurs et de leurs responsabilités ;
- un modèle informationnel de la méthodologie : un modèle qui décrit les éléments de la méthodologie et leurs relations.

Le travail de [Cantamessa et al., 2012] montre qu'il n'existe pas réellement de méthodologie d'implantation du PLM, ni d'évaluation de ses effets. La partie « méthodologie » aujourd'hui n'est donc pas réalisée par le PLM. Le terme « approche », lui, signifie selon le Larousse : « une manière d'aborder un sujet, un problème ».

La définition de CIMData introduit la notion de collaboration, et délimite le PLM dans le temps : de la conception à la fin de vie ; et dans l'espace : il concerne les hommes, les processus, les systèmes d'information et les organisations. Selon CIMData, le PLM n'implique donc pas seulement l'information numérique. Si nous partageons ce point de vue, comme [Terzi et al., 2010], nous considérons que l'information numérique et les outils qui la modèlent sont tout de même le cœur du PLM, et qu'ils influencent ou intègrent ensuite les hommes et les organisations, grâce à un point de vue processus :

Définition du PLM [Terzi et al., 2010]

« Le PLM est une approche intégrée qui utilise les technologies de l'information pour permettre la gestion collaborative des données numériques relatives au produit, au cours de toutes les phases de son cycle de vie. Ainsi le PLM implique :

- ***un point de vue stratégique, selon lequel le produit est considéré comme le seul créateur de valeur pour l'entreprise ;***
- ***la mise en œuvre d'une approche collaborative permettant la mise en commun de toutes les compétences de l'entreprise, distribuées parmi différents acteurs ;***
- ***l'adoption d'un grand nombre de solutions informatiques et d'outils. »***

Il existe d'autres définitions du PLM, confrontées par Thomas Paviot dans sa thèse [Paviot, 2010]. Il y souligne que les communautés scientifique, industrielle et de l'édition informatique ne s'accordent pas complètement sur la notion de PLM, et que, par conséquent, aucune définition précise n'a jamais émergé. Pour contourner ces divergences de

définitions, il est possible de définir le PLM selon ses objectifs, et non ce qu'il est. Les objectifs peuvent être à différents termes : stratégique (long terme), tactique (moyen terme) ou opérationnel (court terme). A long terme, l'objectif du PLM est alors le suivant :

Objectif du PLM [Paviot, 2010]

« L'objectif du Product Lifecycle Management est la maîtrise de la complexité qui caractérise le développement et le suivi des produits. Cet objectif s'inscrit dans une stratégie d'entreprise visant à réduire les coûts, les délais et à augmenter la qualité des produits. »

L'objectif est donc de maîtriser la complexité du système de systèmes de l'entreprise. Mais cette définition replace le PLM dans un contexte d'entreprise, remettant la fonction information à sa place : celle d'une fonction support au produit, dont l'évolution est dirigée par le triptyque stratégique qualité/coûts/délais.

Les applications du PLM

A l'heure actuelle, les applications du PLM proposées par les éditeurs sont éloignées des définitions conceptuelles du PLM, énoncées précédemment. Les outils PLM se centrent aujourd'hui essentiellement sur la phase de conception détaillée [Segonds, 2012], ainsi que sur les liens avec la phase de fabrication. Ainsi, [Garetti et al., 2005] illustrent le périmètre d'un outil PLM par le schéma 1.7. Bien que datant de 2005, ce schéma est toujours d'actualité, puisque les tentatives d'intégration de la maintenance dans les outils PLM restent marginales.

La focalisation des outils PLM sur les phases de conception et de fabrication est à la fois historique et économique. Les premiers travaux sur le PLM ont été le résultat des évolutions des outils de conception, et notamment de la CAO, du PDM, et plus récemment de la maquette numérique. De plus, si pendant la phase de BOL, les informations sont assez bien identifiées, stockées et échangées via des systèmes d'information, au niveau du MOL et de l'EOL, le flux d'information est plus dispersé et moins bien contrôlé : les systèmes d'information se font plus rares. Certains produits portent en eux-mêmes une partie de l'information (cas des produits dits « intelligents »), mais bien souvent, pendant cette phase, l'information est entièrement perdue [Jun et al., 2007]. Un autre facteur important est que les efforts d'investissement ont longtemps été portés sur la phase de conception, car, si celle-ci ne représente en moyenne que 20% du cycle de vie du produit,

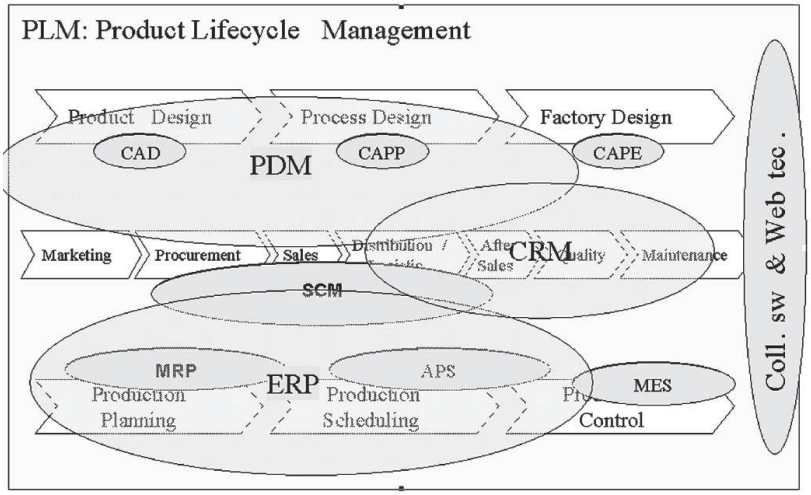


FIGURE 1.7 – Le périmètre du PLM selon [Garetti et al., 2005]

elle détermine plus de 80% des coûts relatifs au produit sur l'ensemble de son cycle de vie. Peu à peu, la problématique d'interface entre la conception et la fabrication a abouti à une intégration relative de la phase de fabrication, notamment par de nombreux travaux sur l'interface PDM/ERP ou PDM/MES/ERP. Ainsi, les outils PLM entendent couvrir aujourd'hui l'ensemble de l'entreprise étendue, mais pas encore l'ensemble du cycle de vie.

Le Closed-loop lifecycle management

Le PLM est une approche qui permet de considérer l'ensemble des activités du cycle de vie en faisant inter-agir les différents acteurs et systèmes d'information. Le PLM est cependant une approche très centrée sur le produit, et qui se concentre pour le moment essentiellement sur le BOL. Ainsi, si cette approche apporte une visibilité et un pilotage des informations liées au produit, elle ne considère cependant pas assez les agents, ni la réutilisation nécessaire de l'information : le PLM ne ferme pas les boucles de l'information, empêchant une réutilisation en amont (par exemple en conception) des informations de l'aval (par exemple de la maintenance) [Kiritsis, 2011]. C'est ce bouclage de l'information que le Closed-loop Lifecycle Management (LM) entend permettre.

Le concept de Closed-loop LM apparaît sous le terme de « Closed-loop PLM » en 2007 [Jun et al., 2007], marquant bien son lien avec l'approche PLM. Ce concept s'appuie sur les nouvelles technologies de traçage et de suivi des produits (RFID, puces, etc.) qui permettent d'envisager de suivre chaque produit pendant le MOL et le EOL, mais aussi, et ce de manière virtuelle, dans le BOL, et d'en extraire les informations nécessaires pour

sa maintenance, son éventuel recyclage, mais aussi pour améliorer la conception et la fabrication de futurs produits. L'information ne doit donc plus être transmise et gérée de phases en phases comme dans les approches PLM : comme le montre la figure 1.2.1, ces nouvelles technologies impliquent de fermer les boucles de l'information tout au long du cycle de vie.

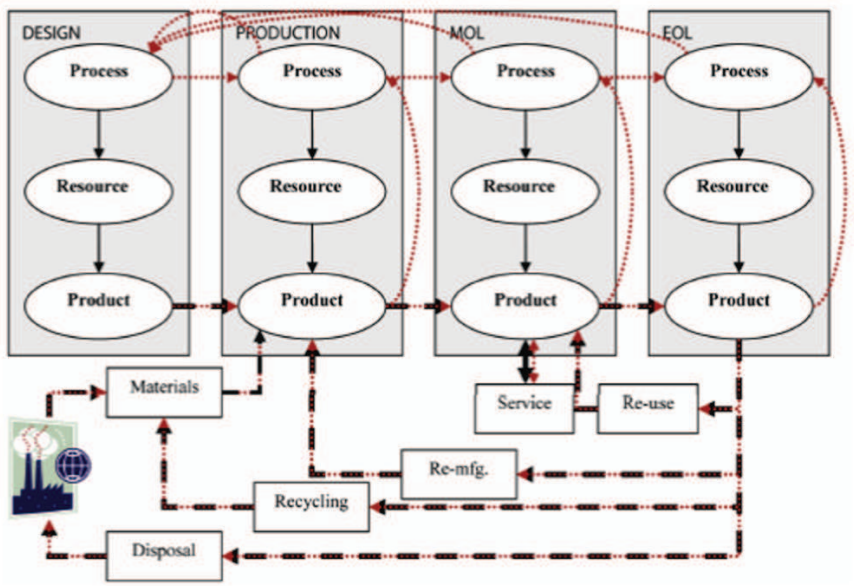


FIGURE 1.8 – Fermer les boucles d'information du PLM (issu de [Kiritsis, 2011])

Selon [Jun et al., 2007], le Closed-loop LM a pour objectifs :

Les objectifs du Closed-loop LM [Jun et al., 2007]

- ***de procurer aux concepteurs les informations issues du MOL et de l'EOL, comme les modes d'utilisation, les conditions de retrait, le recyclage, ce, afin de permettre d'améliorer la conception des produits ;***
- ***de procurer des informations en temps réel du terrain vers l'ingénierie pendant la phase de fabrication ;***
- ***d'assister les agents de maintenance dans leur tâche grâce à une actualisation permanente des informations sur le statut des produits ;***
- ***de permettre aux recycleurs et aux nouveaux utilisateurs d'avoir des informations précises sur les matériels qui arrivent via les filières de recyclage ou de démantèlement.***

Les problématiques soulevées sont donc assez différentes de celles du PLM. En effet, plus que le pilotage et la gestion de l'information que propose le PLM, il faut permettre l'analyse de l'information et la prise de décisions. Il faut également partager une certaine sémantique entre les phases et entre les acteurs, afin de parvenir à un bouclage sémantique des informations. Enfin, il faut garantir des informations pertinentes et actualisées en temps réel. Cependant, ces approches impliquent de faire lire, comprendre ou interpréter la même information à plusieurs acteurs ou SI. C'est pourquoi, comme le soulignent [Noël and Roucoules, 2008], le premier levier d'une démarche cycle de vie est de garantir l'interopérabilité des systèmes d'information composant le système complexe de systèmes.

1.2.3 L'interopérabilité sémantique

Définitions de l'interopérabilité et de l'interopérabilité sémantique

[Wegner, 1996] définit l'interopérabilité comme :

Définition de l'interopérabilité [Wegner, 1996]

« l'aptitude de deux systèmes (ou plus) à communiquer, coopérer et échanger des données et services, et ce malgré les différences dans les langages, les implémentations et les environnements d'exécution ou les modèles d'abstraction. »

D'après [EIF, 2004], il existe trois niveaux d'interopérabilité : le niveau technique ; le niveau sémantique ; et le niveau organisationnel. Si l'on fait l'analogie avec une conversation téléphonique, le niveau sémantique correspond au fait que les deux interlocuteurs parlent chacun dans un langage qui leur permet de se comprendre mutuellement ; le niveau technique doit garantir l'infrastructure nécessaire à ce que la conversation ait lieu, comme la présence d'un réseau, la compatibilité des protocoles, le matériel, etc. ; enfin, le niveau organisationnel concerne les acteurs : tous deux doivent être disponibles au même moment, et alertés que la conversation doit avoir lieu. Un système n'est alors par définition interopérable que lorsqu'il respecte simultanément ces trois niveaux d'interopérabilité. Comme expliqué précédemment, nous nous focalisons dans ce travail sur l'aspect sémantique du problème.

[Özacar et al., 2011] définit l'interopérabilité sémantique comme :

Définition de l'interopérabilité sémantique [Özacar et al., 2011]

« la capacité à interpréter automatiquement l'information échangée avec justesse et sens, dans le but de produire des résultats utiles aux yeux des utilisateurs finaux du système en question »

En d'autres termes, l'interopérabilité sémantique est la capacité du réseau de systèmes d'information à garantir de manière dynamique la pérennité du flux sémantique entre tous les acteurs du réseau, ce flux étant par ailleurs constitué d'informations riches de sens.

Les approches d'interopérabilité sémantique

Plusieurs approches existent pour tenter de résoudre les problématiques d'interopérabilité sémantique. Selon la norme [ISO-14258-1998, 1998] on peut classer ces approches en trois types : l'intégration, l'unification et la fédération.

L'intégration correspond au partage d'un modèle de données commun : si tous les acteurs modélisent l'information suivant le même modèle, alors, il n'y a plus de problème d'échange d'information entre systèmes. Cette approche présente rapidement des limites, que relève [Hoffmann, 2008] : parce qu'elle recherche un consensus entre les différents acteurs du réseau, l'intégration ne peut présenter qu'un niveau de compatibilité limité avec chacun des acteurs. En effet, il n'est pas possible de proposer un modèle qui s'adapte parfaitement à chacun des acteurs puisque ce modèle doit convenir à tous. D'ailleurs, selon [Panetto, 2006], l'intégration est souvent considérée comme allant plus loin que l'interopérabilité, en forçant une certaine dépendance fonctionnelle des applications.

L'unification consiste à établir des correspondances sémantiques entre les acteurs du réseau, plus précisément, il s'agit d'établir les liens sémantiques directement entre les informations des systèmes du réseau. Pour établir ces liens, il est nécessaire de pouvoir traduire la sémantique d'un système à un autre. Il faut alors trouver les correspondances (mapping) entre les sémantiques, comme un linguiste établirait un dictionnaire entre deux langues différentes. Pour cela, un modèle pivot central est utilisé comme médiateur [Tursi et al., 2009, Galeta et al., 2006, Paviot et al., 2011]. Le choix du modèle pivot d'unification peut procéder de deux approches :

- la définition d'un modèle *ad hoc* spécifique au besoin ;
- le choix d'un modèle standard, c'est-à-dire normalisé et ouvert.

Dans la section suivante, nous discutons plus particulièrement des modèles pivot standards existant.

1.2.4 Les modèles standards d'unification

Les modèles standards pour l'interopérabilité

[Lee and Jeong, 2006], comme [Chen and Vernadat, 2004], soulignent l'importance des standards pour l'établissement d'un modèle d'unification des données, comme par exemple la norme STEP (Standard for the Exchange of Product model data) [Pratt et al., 2005], initiée par l'International Standard Organisation (ISO), et connue également sous le nom de ISO 10303. Dans cette étude, nous considérons uniquement les standards pour la modélisation du cycle de vie du produit, en fait, les standards du PLM. Selon [Rachuri et al., 2008] ces standards peuvent être de quatre types, ou « niveaux de définition » :

- Niveau 0 : il s'agit des standards pour les langages d'implémentation ;
- Niveau 1 : il s'agit des standards pour modéliser les informations, par exemple, le langage EXPRESS pour modéliser les informations de la norme STEP ;
- Niveau 2 : il s'agit des standards qui définissent le contenu et la sémantique des modèles d'unification, par exemple, la norme STEP pour la modélisation du produit ;
- Niveau 3 : il s'agit des standards pour l'architecture de travail.

Si l'on ne considère que la sémantique, seuls les niveaux 1 et 2 sont impactés : ils déterminent à la fois l'expressivité des langages utilisés et le contenu sémantique des modèles [Paviot, 2010].

La cartographie des standards du PLM

La figure 1.2.4, issue de [Rachuri et al., 2008], propose une cartographie des standards du PLM. Ils sont représentées en fonction des étapes du cycle de vie qu'ils concernent (axe des abscisses) et d'une classification produit/process/ressource (axes des ordonnées). Cette figure inclue :

- la norme STEP qui propose une modélisation produit pour les phases de conception et de fabrication. Le protocole d'application 239 de cette norme, appelé également PLCS (pour Product Life Cycle Support) permet de donner une dimension « service » à cette norme, en incluant une modélisation de l'ensemble de la supply chain et de la maintenance du produit. Une analyse poussée de STEP a été réalisée par [Paviot, 2010] ;

- Le standard SCOR (Supply Chain Operations Reference) qui est un modèle pour la spécification des processus et des services de la supply chain ;
- PSL (Process Specification Language) qui est un standard d'activités pour la phase de fabrication ;
- SysML (Systems Modeling Langage), langage de modélisation pour l'ingénierie système ;
- L'EDI est un standard pour l'échange sécurisé des informations entre les entreprises ;
- ebXML (Electronic Business using eXtensible Markup Language) est un standard dédié au e-commerce.

A cette représentation qui date de 2008, il convient d'apporter les modifications suivantes :

- La norme ISO 15926 est un standard pour l'échange et l'intégration des données techniques, qui s'adresse plus particulièrement au domaine de l'énergie. Construite pour résoudre des problématiques d'exploitation et de maintenance, elle concerne cependant l'ensemble du cycle de vie.
- Le champ de la norme STEP doit être étendu vers les phases amont du cycle de vie, et sa modularité doit être augmentée : STEP est un standard plus générique et extensible que la représentation de [Rachuri et al., 2008] ne le montre.

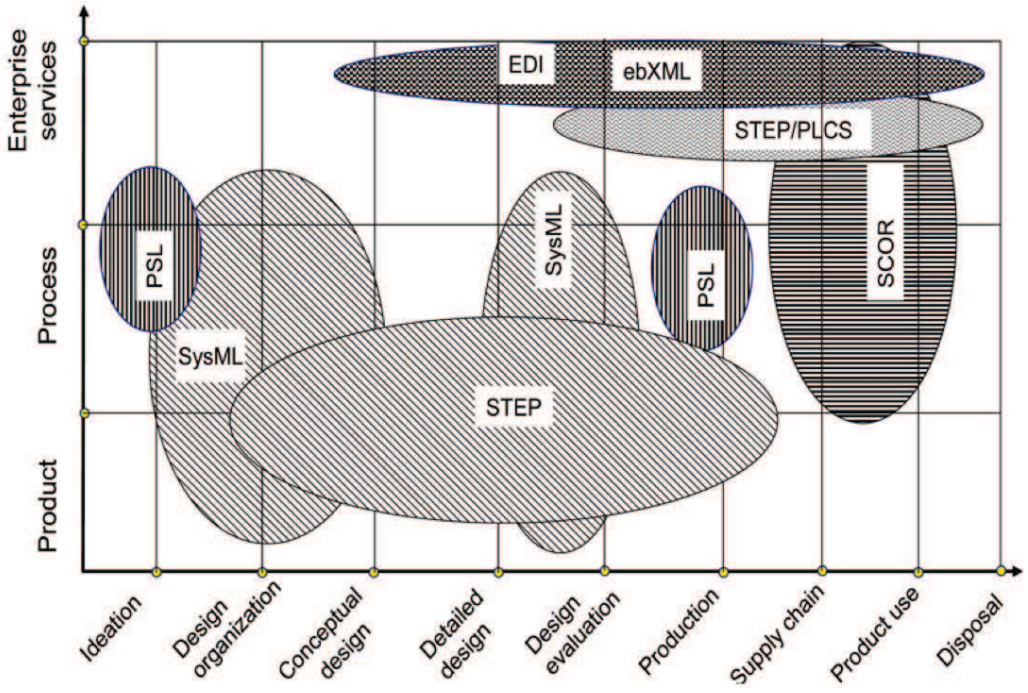


FIGURE 1.9 – Cartographie des standards du PLM [Rachuri et al., 2008]

1.2.5 La fédération : combattre la rigidité des standards

Le troisième paradigme d'interopérabilité sémantique existant est la fédération. Il repose sur l'idée que chaque « métier » (ou système) doit pouvoir conserver son propre modèle d'information pour en garantir le sens et la flexibilité. Alors, pour échanger les informations entre les systèmes, la fédération propose d'établir des connexions automatiques entre modèles, fondées sur la logique. Nul besoin de se conformer à un modèle pivot pré-établi, puisque les échanges peuvent se faire directement et dynamiquement. Pour y parvenir, les ontologies sont vues aujourd'hui comme des outils potentiellement capables de réaliser cet échange dynamique, car les modèles ontologiques sont fondés sur la logique et peuvent raisonner. De plus, les travaux académiques ont permis de réaliser des mapping logiques et directs entre ontologies : on appelle cela l'alignement d'ontologies. Seulement, comme le précise [Hoffmann, 2008] (et nous y reviendrons par la suite), les résultats dans ce domaine ne sont pas encore satisfaisants, si bien qu'il n'y a pas à l'heure actuelle de réalisation avérée de l'interopérabilité fédérative.

Conclusion de la section 1.2

Nous avons déjà évoqué que ce travail porte sur la modélisation sémantique des informations. Dans cette section, nous avons délimité le terrain : les informations considérées sont contenues dans les systèmes d'information dont la multiplicité conduit à considérer leurs interactions mutuelles selon un point de vue cycle de vie global. L'un des enjeux majeurs d'une telle démarche est de maîtriser l'interopérabilité entre systèmes, tant aux niveaux technique, sémantique, qu'organisationnel. En matière d'interopérabilité sémantique, il existe aujourd'hui trois approches possibles : l'intégration, l'unification et la fédération. Le développement des modèles standards a permis de dépasser les limites de l'intégration, trop contraignante, en permettant de réaliser l'unification. Récemment, avec l'augmentation des quantités de données et le développement du web, de nouveaux paradigmes de modélisation ont vu le jour, fondés sur l'utilisation d'ontologies [Zhang and Yin, 2008]. Pour plusieurs auteurs, les ontologies présentent de nouvelles capacités sémantiques qui permettraient d'atteindre la fédération, approche où les modèles seraient alignés dynamiquement, de par la logique qu'ils portent en eux. Alors, « ontologies » et « approche fédérative » sont souvent associées, et leur utilisation dans l'industrie va croissante. La section suivante définit la notion d'ontologie et montre, par une étude quantitative, l'utilisation qui est faite des ontologies dans des applications « cycle de vie ».

1.3 L'émergence des modèles à base d'ontologies

1.3.1 Les ontologies : définition et langages

Les ontologies : définition

Les ontologies sont des outils de modélisation, développés dans le cadre du web sémantique. [Berners-Lee et al., 2001] donnent une définition du web sémantique selon les objectifs qu'il poursuit :

Définition du web sémantique [Berners-Lee et al., 2001]

« le web sémantique n'est pas un nouveau web, mais l'extension de l'actuel, dans lequel les informations auraient un sens bien défini, permettant aux ordinateurs et aux utilisateurs de mieux travailler ensemble. Les premières étapes de tissage du web sémantique au travers de la structure actuelle du web sont déjà en cours. Dans un futur proche, ces développements vont ouvrir la voie à de nouvelles fonctionnalités, puisque les machines vont être bien plus capables de traiter et de « comprendre » les données, que, pour le moment, elles ne font qu'afficher. »

Afin d'échanger des informations au sein du web sémantique, il est nécessaire que tous les participants s'accordent sur l'ensemble des éléments qui existent dans leur domaine et sur les relations entre ces éléments [Noy, 2003]. Cet accord peut être spécifié de façon formelle par une ontologie. La définition la plus largement reconnue [4] de l'ontologie est celle donnée par [Gruber, 1995] :

Définition de l'ontologie [Gruber, 1995]

« Une ontologie est une représentation explicite d'une conceptualisation. »

Cette courte définition est riche d'informations. Tout d'abord, l'ontologie est une « représentation » : cela signifie qu'il s'agit d'un outil de modélisation. Le terme « explicite » indique que l'ontologie est capable de modéliser l'ensemble des informations d'un domaine donné, quel que soit le niveau d'abstraction ou le type d'application. Dans le cas de la modélisation d'un produit, une ontologie doit permettre de représenter les informations liées au produit, mais aussi les *meta-informations*, corollaires au produit, mais de

4. en témoigne le nombre de citations dans la littérature

l'ordre du processus, de l'organisation, ou encore des ressources ou des règles. Enfin, le terme « conceptualisation » nous indique que le paradigme de modélisation de l'ontologie est de type *concept-propriété* (ou *classe-relation*). Les entités d'une ontologie, que l'on appelle *instances*, sont donc regroupées par concepts, qui interagissent potentiellement entre eux au moyen de relations. De plus, la conceptualisation renvoie à la définition d'une sémantique propre au domaine : selon [Lee et al., 2009], une conceptualisation est « une extraction du vocabulaire d'un domaine et est une vue abstraite et simplifiée du monde que l'on souhaite représenter ». [Borst and Akkermans, 1997] complètent la définition de [Gruber, 1995] en 1997 :

Définition de l'ontologie [Borst and Akkermans, 1997]

« Une ontologie est une spécification formelle d'une conceptualisation partagée. »

L'ajout du terme « formelle » implique que l'ontologie doit être compréhensible par une machine et l'ajout du terme « partagée » implique que plusieurs utilisateurs peuvent utiliser la même ontologie pour décrire le même domaine. Avec [Borst and Akkermans, 1997], l'ontologie prend une dimension collaborative, et son implémentation est envisagée.

Enfin, plus récemment, [Noy and McGuiness, 2001] définissent l'ontologie comme :

Définition de l'ontologie [Noy and McGuiness, 2001]

« Une description formelle et explicite de concepts dans un domaine de discours (les classes ou concepts), de propriétés décrivant les différents attributs et caractéristiques de chaque concept (les rôles ou propriétés) et de restrictions. »

[Noy, 2003] remplace en réalité le terme « conceptualisation », par sa définition informatique : il s'agit d'une modélisation de type classe-relation. Il rapporte également que pour pouvoir être explicite, cette conceptualisation nécessite une richesse sémantique, réalisée au travers de restrictions sur les classes et relations. Ces restrictions sont exprimées à partir d'axiomes fondamentaux du langage ontologique, si bien que [Maedche, 2002] définit finalement mathématiquement l'ontologie, par un 5-uplet :

Définition mathématique de l'ontologie [Maedche, 2002]

« $O := \{C, R, H, ref, A^0\}$ ***où*** C ***représente l'ensemble de concepts (ou classes),***

disjoint de R qui représente l'ensemble des relations (ou propriétés). H représente la hiérarchie (ou taxonomie) entre les concepts, tandis que ref est une fonction qui associe les relations non-taxonomiques aux différents concepts. Enfin A^0 est un ensemble d'axiomes exprimés dans un langage logique. »

On peut illustrer le principe d'une ontologie en prenant comme exemple la modélisation des relations entre les membres d'une famille. Imaginons une famille composée de cinq personnes dont le point central est Claire : Claire a une mère Sylvie, un frère Pierre, un fils Simon et une fille Sophie. Une ontologie[5] permet d'exprimer les liens familiaux ainsi que les rôles respectifs de chacun.

Exemple de modélisation par une ontologie

Définition de l'ontologie « famille »

La définition exhaustive de l'ontologie famille se trouve en annexe 7.1. Les classes (C) principales sont : **humain**, **homme** et **femme**. **Homme** et **femme** sont des sous-classes d'**humain** (H). Elles sont définies comme disjointes, grâce à un axiome ($A°$) : aucune instance ne peut appartenir à la fois à la classe femme et homme. Les liens familiaux sont exprimés par des propriétés, restreintes à leur domaine de valeur (ref). Par exemple, la propriété *a_parent* est restreinte de **enfant** vers **parent** : le domaine d'application (*range*) de la propriété *a_parent* est `parent` alors que son domaine de définition (*domain*) est enfant. Sur cette propriété, on peut imposer une restriction de cardinalité ≤ 2 pour exprimer le fait que toute instance ne peut avoir que deux parents au maximum. La figure 1.10 présente les classes et leurs liens respectifs dans l'ontologie finale.

Les autres classes visibles sur la figure 1.10 - **fils**, **frere**, **sœur**, **mere**, etc. - peuvent être définies avec des axiomes OWL, comme suit :

fille $\equiv$ **femme** that *a_parent* some **humain**

Cet axiome définit une fille comme étant une femme qui a au moins un parent. Les autres axiomes de l'ontologie sont visibles dans l'annexe 7.1.

---

5. cette ontologie utilise l'expressivité du langage OWL2. Les détails de la modélisation sont donnés en annexe.

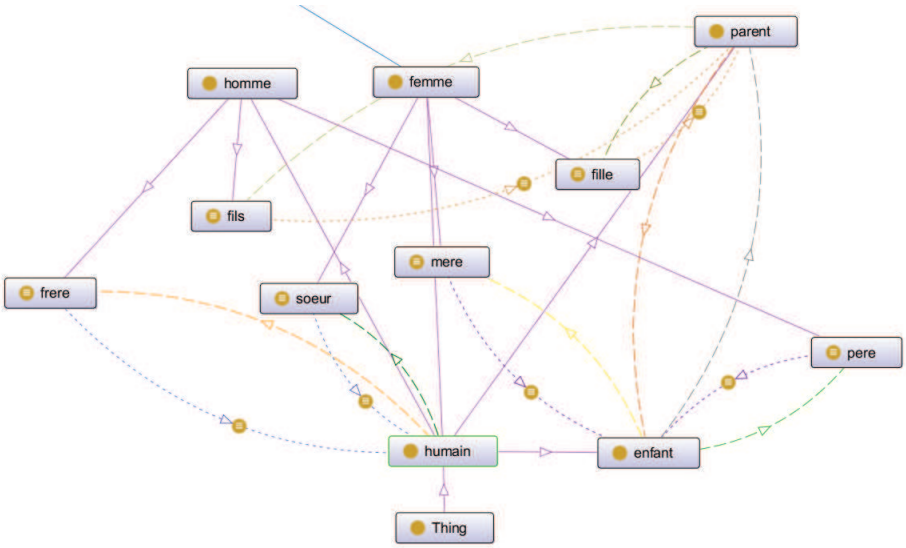


FIGURE 1.10 – Taxonomie de l'ontologie « famille »

Les ontologies d'inférence

Il est possible de classer les ontologies suivant deux types principaux : les ontologies de stockage et les ontologies d'inférence [Fankam, 2009]. Cette classification est de type informatique, car elle se fonde uniquement sur le type de données que les ontologies permettent de manipuler. D'autres classifications d'ontologies existent, s'appuyant sur le niveau d'abstraction ou le type d'application, que nous évoquerons plus tard.

Les ontologies de stockage ne traitent que des données canoniques, c'est-à-dire qu'une instance ne peut se trouver que dans une et une seule classe à un moment donné. Leur but est de permettre une description claire et structurée du domaine métier. Elles sont par conséquent particulièrement adaptées à des problématiques de type « catalogue ». Les langages standards associés à ces ontologies sont le Resource Description Format [6] (RDF) et son extension RDF Schema (RDFS) ainsi que PartLIBraries (PLIB), qui est le standard ISO 13584.

Les ontologies d'inférence permettent de traiter des données non-canoniques, c'est-à-dire qu'une instance peut se trouver dans plusieurs classes en même temps. L'intérêt alors de ces ontologies est de pouvoir déduire des appartenances d'instances à des classes grâce aux axiomes et restrictions de l'ontologie. On appelle cela « inférer des informations ». L'inférence, qui est obtenue grâce à des moteurs d'inférences (également

6. http ://www.w3.org/RDF/

appelés raisonneurs), est selon [Krima et al., 2009] :

Définition de l'inférence (Krima)

« la capacité de faire des déductions sur les instances à partir des classes, propriétés et axiomes explicitement définis dans l'ontologie. »

Les ontologies d'inférence s'expriment avec les languages de type Description Logics (DLs) [Baader et al., 2005], et plus particulièrement les langages Web Ontology Language (OWL)[7].

1.3.2 Les applications des ontologies pour l'approche cycle de vie

Depuis la standardisation du langage OWL, l'emploi d'ontologies va croissant, particulièrement pour les applications de gestion du cycle de vie du produit. La figure 1.11 montre les résultats obtenus sur le site Science direct pour les requêtes *ontology + industry* et *ontology + lifecycle*. L'indicateur utilisé est le nombre de publications par année[8]. Pour ces deux requêtes, on remarque une progression constante année après année du nombre de résultats, progression qui est sensiblement la même dans les deux cas : une multiplication par 5 entre 2004 et 2012. (Sur google scholar, toutes années confondues, ces requêtes montent à 159 000 résultats pour *ontology + industry* et 33 100 pour *ontology + lifecycle*). Ces deux requêtes traitent en priorité des sujets suivants :

	Ontology + industry	ontology + lifecycle
sujets principaux	web services	software
	semantic web	web services
	software	semantic web
	information systems	information systems
	knowledge management	business process

Table 1.2: Principaux sujets traités dans la littérature (source : science direct)

Il est intéressant de noter, au-delà du fait que les sujets sont connexes dans les deux requêtes, que les deux sujets les plus traités -hors outils informatiques- sont la gestion des connaissances et la modélisation des processus métier.

7. Il existe d'autres langages d'inférence, mais dont l'utilisation reste marginale, car seuls les languages OWL sont standardisés.

8. Les résultats ont été obtenus au mois de mars 2013. Les valeurs pour 2013 sont des projections : le triple de la valeur en mars 2013.

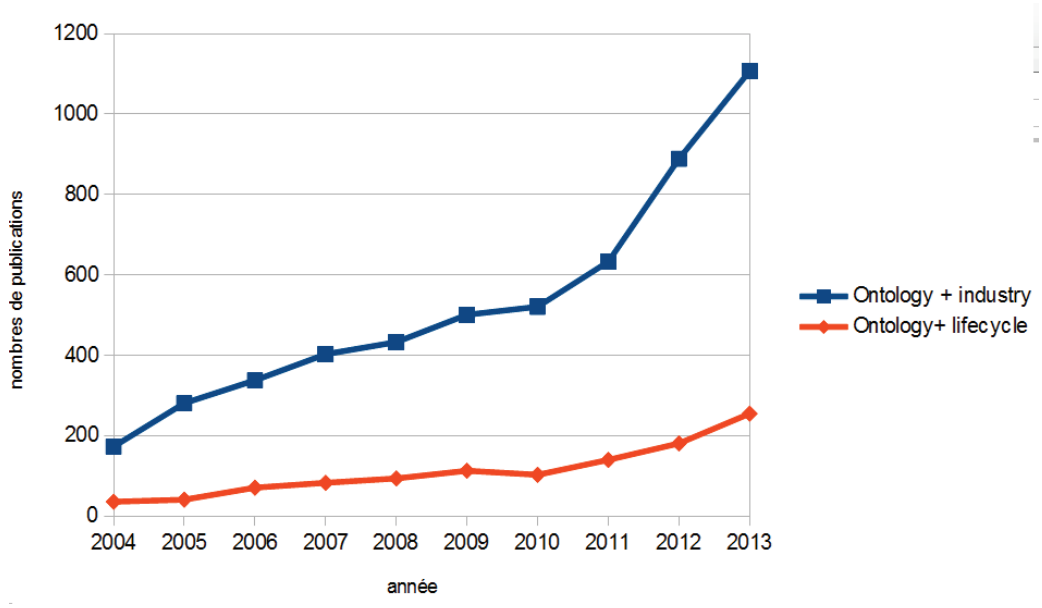


FIGURE 1.11 – Résultats de recherche bibliographique sur Science direct

Nous avons ensuite affiné la recherche, en nous limitant aux travaux utilisant les ontologies d'inférence pour des applications industrielles (les états de l'art par exemple, sont exclus de ce recensement). Les industries concernées sont les industries manufacturières, la construction et l'énergie. Cela exclut, par exemple, les travaux -nombreux- liés à la modélisation de l'environnement dans le cade de l'industrie agro-alimentaire. Nous n'avons également sélectionné que les travaux se centrant sur le produit ou les processus industriels. Dans les tableaux 1.3, 1.4 et 1.5, les travaux de la littérature sont reportés suivant l'année de publication (en abscisse), depuis 2004 (année de standardisation du langage OWL) jusqu'à 2012. La dimension verticale représente le cycle de vie du produit. Les travaux sont ainsi classés en fonction des étapes du cycle de vie qu'ils concernent : conception, fabrication, maintenance/usage et fin de vie. Enfin, sur la ligne « autre » sont reportés les travaux qui n'adressent pas une phase du cycle de vie en particulier, soit parce qu'ils en concernent plusieurs, parce qu'ils proposent des méthodologies génériques ou parce qu'ils réalisent une évaluation des potentialités des ontologies, sans se focaliser sur une activité en particulier. Les travaux en italique sont issus d'actes de conférences ou de rapports techniques, tandis que les autres sont des articles de journaux scientifiques.

Année	2004	2005	2006
conception		*[Beetz et al., 2005]*	[Kim et al., 2006]
			[Nanda et al., 2006]
fabrication		*[Lee et al., 2005]*	
MOL	*[Preuveneers et al., 2004]*		
EOL	[Price and Bodington, 2004]		
Autre	*[Baysal et al., 2004]*	*[Dori and Shpitalni, 2005]*	*[Mun et al., 2006]*
	[Cruz et al., 2004]		
	[Cao et al., 2004]		
Total	1(*4*)	0(*3*)	2(*1*)

Table 1.3: Etat de l'art des ontologies d'inférence dans l'industrie, 2004-2006

Année	2007	2008	2009
conception	*[Ahmed et al., 2007]*	[Chang et al., 2008]	[Catalano et al., 2009]
	[Pandit and Zhu, 2007]	[Yang et al., 2008]	[Lee et al., 2009]
	[Fiorentini et al., 2007]	[Zhang and Yin, 2008]	*[Krima et al., 2009]*
		[Brandt et al., 2008]	[Han and Park, 2009]
			[Morbach et al., 2009]
fabrication	[Lin and Harding, 2007]	[Gimenez et al., 2008]	[Tursi et al., 2009]
	[Barkmeyer, 2007]	[Blomqvist and Ohgren, 2008]	
MOL		[Mun et al., 2008]	
EOL			
Autre		[Zhao and Liu, 2008]	[Chen et al., 2009]
		[Fiorentini et al., 2008]	
		[Kim et al., 2008]	
Total	2(*3*)	8(*2*)	6(*1*)

Table 1.4: Etat de l'art des ontologies d'inférence dans l'industrie, 2007-2009

Année	2010	2011	2012
conception	*[Jeong et al., 2010]*	*[Wicaksono et al., 2011]*	[Demoly et al., 2012]
	[Fiorentini et al., 2010]	[Afacan and Demirkan, 2011]	[Cai et al., 2012]
	[Kossman et al., 2010]	[Dong et al., 2011]	[Pardo et al., 2012]
	[Ramos-Garcia, 2010]	[Lim et al., 2011]	[Sun et al., 2012]
	[El-Gohary and El-Diraby, 2010]		[Lee and Jeong, 2012]
			[Barbau et al., 2012]
fabrication	*[Lu et al., 2010]*	[Raza et al., 2011]	*[Kiritsis et al., 2012]*
	[Yan et al., 2010]	[Mikos et al., 2011]	*[Pintzos et al., 2012]*
	[Alsafi and Vyatkin, 2010]	[Vegetti et al., 2011]	*[Oh and Shin, 2012]*
			[Panetto et al., 2012]
			[Giovannini et al., 2012]
			[Landherr and Constantinescu, 2012]
MOL	[Matsokis and Kiritsis, 2010]	[Kim et al., 2011]	
	[Sorensen et al., 2010]	[Mun and Ramani, 2011]	
	[D'Elia et al., 2010]	[Matsokis and Kiritsis, 2011]	
		[Rosselló-Busquet et al., 2011]	
		[Koukias et al., 2012]	
EOL	*[Merdan et al., 2010]*		
Autre	*[Lee et al., 2010]*	[Wang and Yu, 2011]	*[Krima et al., 2012]*
		[Özacar et al., 2011]	[Pardo et al., 2012]
		[Albani and Dietz, 2011]	
Total	7(*6*)	12(*2*)	10(*4*)

Table 1.5: Etat de l'art des ontologies d'inférence dans l'industrie, 2010-2012

Dans ces tableaux, on retrouve à nouveau la croissance annuelle du nombre de publications sur le sujet. Mais il est surtout intéressant de constater que les travaux se concentrent sur le début de vie, et plus particulièrement sur la phase de conception. Historiquement, la recherche sur le PLM s'est en effet focalisée sur la conception : suite à l'émergence des outils de CAO/DAO, les approches PDM ont été développées afin de capitaliser les informations de la conception. Ensuite, la fabrication a peu à peu été intégrée : c'est le début véritable du PLM. Les solutions PLM actuelles sont dans les faits essentiellement des solutions de gestion documentaire pour l'entreprise étendue (soit conception et fabrication). Peu d'entre elles intègrent la maintenance. L'importance des travaux sur la conception dans les démarches PLM explique alors en grande partie cette phase concentre également l'essentiel des travaux sur les modèles ontologiques.

Cependant, la figure 1.5 permet également de remarquer une diversification avec le temps des activités envisagées, avec une intégration progressive de la maintenance notamment. Cette extension au milieu de vie est essentiellement portée par le développement du standard ISO 15926 et par le projet européen PROMISE [Matsokis and Kiritsis, 2010]. La lecture du tableau 1.5 laisse enfin penser que les applications sur la fin de vie sont marginales. Cela s'explique par le fait que la plupart des applications des ontologies pour cette phase tentent de modéliser l'environnement du produit ou les impacts écologiques des activités, et pas directement les produits ni les processus de fin de vie en eux-mêmes.

Conclusion de la section 1.3

L'intérêt de la communauté scientifique pour les modèles à base d'ontologies est certain, en témoignent la quantité de publications sur le sujet. Le paradigme de modélisation qu'apportent les ontologies a déjà permis de repenser des modèles de conception et de fabrication, phases sur lesquelles se concentrent les travaux à l'heure actuelle. Cependant, les possibilités des ontologies semblent très larges puisque les contributions de la littérature portent sur des domaines et des phases de plus en plus variés. La problématique de ces travaux de recherche, décrite dans la section suivante, est alors de mieux comprendre l'apport réel des ontologies pour la modélisation sémantique des modèles produit.

1.4 La problématique et la démarche de recherche

1.4.1 Dépasser les limites de l'unification, et celles de la fédération

Nous l'avons évoqué précédemment, l'unification est la seule approche d'interopérabilité qui soit réellement réalisée à l'échelle industrielle, et qui permette d'améliorer l'échange d'informations entre les systèmes tout au long du cycle de vie. Cependant, cette approche fondée sur les standards a aussi ses limites. Outre le fait que le cycle de vie n'est pas entièrement couvert à l'heure actuelle, les limites sémantiques de cette approche sont les suivantes :

- des problématiques liées à **la faible richesse sémantique des modèles existants**. En effet, Les approches d'unification actuelles n'intègrent pas l'ensemble des informations nécessaires au produit. La norme STEP par exemple couvre aujourd'hui les champs de la CFAO et du PDM, mais exclut les informations non-geométriques, comme la fonction, par exemple [Barbau et al., 2012] ;
- une **dimension statique du standard d'unification**, frein à un échange dynamique des informations mais aussi à une extensibilité des modèles. Comme le montre

[Hoffmann, 2008], l'unification nécessite une connaissance a priori des modèles de chaque système. De plus, il n'existe souvent pas un unique modèle de plus haut niveau pour le réseau, ce qui implique que le modèle pivot choisi doit être mis à jour régulièrement en fonction des modifications du réseau. Cela confère un aspect statique à l'unification, qui n'est pas en adéquation avec la dynamique intrinsèque au système d'information PLM [Paviot et al., 2011]. De plus, l'établissement des transformations sémantiques entre les acteurs du système et le modèle unificateur, sur lesquelles repose la méthode d'unification, sont souvent établies manuellement et donc, peu évolutives.

– **l'absence de partage d'une sémantique commune** qui limite la compréhension mutuelle des informations contenues dans les modèles. Ce problème devient d'autant plus prégnant que le modèle a une sémantique riche : il faut alors garantir l'interprétation de l'information qu'il contient.

Dans ce contexte, il est nécessaire d'évoluer vers un nouveau paradigme de modélisation. Mais il n'est pas certain que ce nouveau paradigme soit la fédération, au sens d'un seul alignement dynamique des modèles. En effet, cette vision de la fédération nous semble sémantiquement limitée : peut-on réellement prétendre automatiser le sens des concepts par de la simple logique ? Les ontologies ne permettraient-elles pas d'aller plus loin que la fédération ? D'ailleurs, l'ISO 14258 précise que « *les approches d'unification comme les approches fédératives ont besoin d'être supportées par des standards [...]. Pour supporter l'unification, la construction des modèles doit être standardisée, tandis que pour réaliser la fédération, les interfaces sont les éléments à standardiser.* »

Il semble en effet que ne peuvent résulter des approches d'alignement que des rapprochements au mieux syntaxiques, alors que la sémantique, nous l'avons vu précédemment, est plus subtile et complexe que la syntaxe : elle est « intentionnelle ». Ainsi, il est légitime de se demander comment l'alignement d'ontologies peut permettre de lever les ambiguïtés sémantiques [9]. Pour autant, il ne faut pas en conclure que les modèles à base d'ontologies ne présentent pas d'intérêt pour la modélisation sémantique des informations : l'enjeu est de comprendre précisément ce qu'il est.

1.4.2 La problématique de recherche

Dans ce travail, nous proposons d'évaluer, à travers l'étude de la littérature mais aussi une application industrielle concrète, les apports potentiels des ontologies afin de lever les verrous de l'unification. Nous évaluons aussi les limites et les freins à leur utilisation

9. Un exemple d'ambiguïté sémantique est l'assertion suivante : « prendre un verre ». S'agit-il du contenant ou du contenu ?

pour des applications industrielles de grande ampleur et couvrant l'ensemble du cycle de vie du produit.

Ainsi, la problématique de ce travail est de comprendre si et comment les ontologies, en tant que nouveau paradigme de modélisation sémantique, permettent de dépasser les limites des approches actuelles d'unification, afin de modéliser les informations produit sur l'ensemble du cycle de vie.

1.4.3 Questions et démarche de recherche

Répondre à cette problématique implique de répondre à deux questions de recherche sous-jacentes :

Comment doit-on modéliser sémantiquement les informations produit dans une approche cycle de vie ?

Cette première question implique de définir un cadre générique de modélisation, applicable dans différents domaines industriels, et qui permette d'assurer la captation et l'échange de la sémantique entre les différents acteurs du cycle de vie. Ce n'est qu'une fois ce cadre générique posé qu'il est possible de répondre à la deuxième question de recherche :

Quels sont les apports et les limites potentiels des ontologies pour réaliser ce cadre de modélisation sémantique générique ?

Il s'agit d'évaluer les potentialités des ontologies, considérées alors comme des outils sémantiques, capables ou non de réaliser l'approche envisagée.

Afin de répondre à ces différentes questions, la démarche de recherche suivante est envisagée :

Tout d'abord, un état de l'art est réalisé. Celui-ci a pour objectif de comprendre les propriétés des ontologies qui pourraient être sources d'amélioration vis-à-vis des langages existants. Notamment, deux aspects sont discutés : l'expressivité des langages et le raisonnement. Cet état de l'art décrit également les objectifs de modélisation exposés dans la littérature qui justifient ce transfert vers des modèles ontologiques. Ces objectifs sont de trois natures principales : le besoin de modèles sémantiquement riches afin d'intégrer les connaissances, un échange d'information dynamique qui conserve le flux sémantique, et un partage de sémantique pour une mutuelle compréhension.

Ensuite, nous proposons un cadre de modélisation global des informations. Celui-ci se réalise par trois éléments clés : une unification des données grâce à un méta-modèle partagé ; une intégration d'un plus large spectre d'informations par des modèles métiers spécifiques et compatibles avec le modèle unifié ; et la définition de modèles applicatifs interopérables et cohérents entre eux, grâce au modèle unifié. L'enjeu est alors de comprendre en quoi les ontologies permettent de mieux réaliser ces trois types de modélisation, et ce, au travers d'illustrations issues du cas d'étude industriel : la modélisation des centrales nucléaires.

Enfin, l'implémentation de la modélisation proposée permet, sur le cas d'étude industriel, de lever les difficultés et les limites d'une modélisation fondée sur les ontologies, et de comparer concrètement ce paradigme de modélisation avec celui utilisé pour l'implémentation du démonstrateur : UML/OCL. Ces limites, qui sont révélées grâce au cas d'application ont cependant des portées génériques, qui permettent de définir des perspectives de recherche afin d'améliorer les fonctionnalités des ontologies, mais aussi la modélisation des informations de manière plus générale.

CHAPITRE 2

L'APPORT DES ONTOLOGIES POUR LA MODÉLISATION DES INFORMATIONS : ÉTAT DE L'ART

Résumé du chapitre

Ce chapitre étudie, à travers une revue de la littérature et un cas d'illustration, l'apport potentiel des ontologies dans la réalisation d'une interopérabilité fédérative. Plus particulièrement :

La section 2.1 définit les ontologies en présentant leurs langages et caractéristiques, et notamment, l'expressivité des langages DL et l'inférence. Ensuite, l'analyse de la littérature permet de dégager trois objectifs majeurs de l'emploi des ontologies dans les applications industrielles : créer des modèles riches en sémantique, qui ont vocation à modéliser une information source de connaissance (métier), comme expliqué dans la section 2.2 ; générer un échange plus dynamique et efficace des informations, à partir de ces modèles sémantiques, notamment en revoyant la définition des modèles standards suivant une approche ontologique ; et enfin partager une sémantique commune entre systèmes afin d'uniformiser la compréhension de l'information qui est échangée par les différents acteurs.

La section 2.3 illustre ces trois objectifs au travers d'un cas d'illustration : une ontologie pour l'interface conception/fabrication. Enfin la section 2.4 propose un état de l'art des méthodologies existantes pour la conception de modèles à base d'ontologies.

2.1 Les potentialités des ontologies d'inférence

2.1.1 Les langages OWL

Le Web Ontology Language (OWL) est un langage ontologique standard, recommandé par le W3C depuis 2004, dont la création fait suite à l'émergence simultanée d'une multitude de langages ontologiques disparates [Matsokis and Kiritsis, 2010], comme DAML ou OIL. Il se décompose en trois sous-langages : le OWL-Lite, le OWL-DL et le OWL-Full (voir figure 2.1).

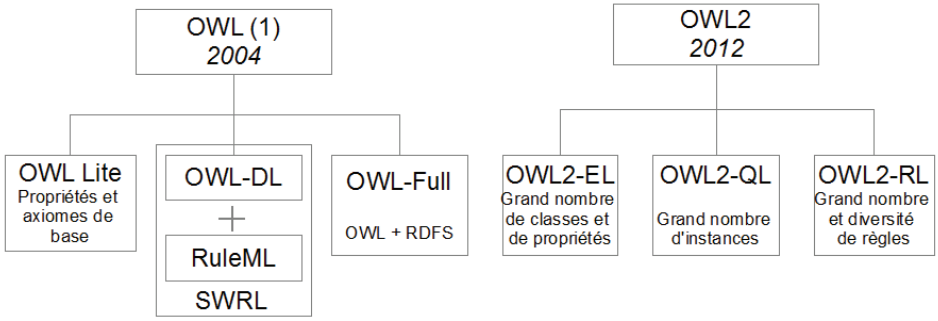


FIGURE 2.1 – Les langages et sous-langages OWL

Depuis décembre 2012, une extension de OWL (dès lors appelé OWL1) est également devenue un standard du W3C : OWL2. OWL2 est compatible avec OWL1 (toute ontologie OWL1 est exprimable en OWL2). L'une des caractéristiques principales de différenciation des langages OWL est leur niveau d'expressivité, qui est graduel (le plus faible pour OWL-Lite et le plus grand pour OWL-Full et OWL2).

2.1.2 L'expressivité

L'expressivité : définition

L'expressivité d'un langage ontologique est sa capacité à représenter des concepts, des propriétés et des restrictions variés. La logique descriptive (DL), sur laquelle reposent les langages OWL, est la source de leur expressivité. Le noyau expressif sur lequel reposent tous ces langages est la logique Attribute Language (AL), dont les éléments sont donnés dans le tableau 2.1. Elle permet la définition de rôles et de concepts atomiques ; la négation, la restriction et l'existence des concepts atomiques ; l'intersection de concepts

quelconques ; et une première hiérarchisation des concepts, avec les concepts « bottom » et « top ».

Syntaxe	Définition
A	concept atomique
R	rôle atomique
$\top$	concept général (top)
$\bot$	concept spécifique (bottom)
$\neg A$	négation d'un concept atomique
$C \sqcap D$	intersection
$\forall A$	restriction d'un concept
$\exists A$	existence limitée

Table 2.1: Logique Attribute Language (AL), issu de [Obitko, 2013]

La logique ALC (parfois appelée ALUEC) est une extension de la logique AL, qui apporte l'union de deux concepts (U), la quantification (ϵ ou E) et la négation de n'importe quel concept (C) [Baader et al., 2005], comme représenté dans la partie supérieure du tableau 2.2.

nom	Syntaxe	Définition
U	C⊔D	union de deux concepts
E	∃R.C	existence complète
C	¬C	négation d'un concept
N	≥nR, ≤nR	restriction numéraire d'un rôle
S	Trans(R)	transitivité d'un rôle
H	R⊆S	taxonomie de rôles
I	R^{-}	rôle inverse
F	≤ 1R	fonctionnalité (unicité) d'un rôle
O	$\{a_1, ..., a_n\}$	classe énumérée
R		rôles complexes
Q		cardinalité exacte d'une relation instanciée

Table 2.2: Expressivité des DLs

Si les langages OWL et RDFS reposent sur la logique ALC, il peuvent avoir d'autres extensions d'expressivité, que l'on désigne communément par les lettres *N, S, F, H, I, O, R, Q* et dont la signification est donnée dans le tableau 2.2. L'expressivité d'un langage DL s'exprime alors par un ensemble de lettres, en fonction des fonctionnalités du langage (en plus de ALUENC, commun à tous les langages DL). Ainsi, OWL-Lite, qui est le moins expressif des langages OWL, est *SHIF*. OWL-DL, qui est le plus répandu dans les applications industrielles, est d'expressivité SHOIN. Cela signifie qu'en plus de la logique ALUENC, il permet d'exprimer la transitivité (S), la hiérarchie (H), la fermeture

des classes (O), l'inverse (I) et la restriction (N). OWL-DL est souvent appelé SHOIN(D), le « D » signifiant qu'il distingue deux types de propriétés : des propriétés de type *objet* qui relient deux instances entre elles, et des propriétés de type *donnée* qui attribuent des valeurs aux instances [Yang et al., 2008]. Enfin, OWL-2 a une expressivité SROIQ et permet ainsi de complexifier les rôles (passage de H à R) et les cardinalités (passage de N à Q). Pour une description complète de la logique SROIQ, voir [Horrocks et al., 2006]. Le tableau 2.3 résume l'expressivité des langages OWL, du moins expressif (OWL-Lite) au plus expressif (OWL2).

Langage	Expressivité
OWL-Lite	SHIF
OWL-DL	SHOIN
OWL-Full	SHOIN+RDFS
OWL2	SROIQ

TABLE 2.3: L'expressivité des langages OWL

Illustration de l'expressivité

On peut illustrer l'expressivité du langage OWL-DL grâce à l'ontologie « famille » définie précédemment. Pour cela, on modélise l'ontologie famille en OWL-DL en utilisant le logiciel Protégé. OWL-DL est dit d'expressivité SHOIN, cela signifie :

- S (transitivité) : définir la propriété *a_frere_soeur* comme transitive, c'est exprimer le fait que si Pierre a pour sœur Claire qui elle-même a pour sœur Julie, alors Pierre a aussi pour sœur Julie ;
- H (taxonomies de rôles) : dans l'ontologie famille, il existe une taxonomie des relations. Les propriétés *a_fils* et *a_fille* sont en effet des sous-propriétés de *a_parent* : $a\_fils \subseteq a\_parent$;
- O (énumération) : Les membres de la famille étant en nombre limité, on déclare la classe **humain** comme classe énumérée : elle est donc composée uniquement de l'ensemble des membres de la famille. Déclarer une classe fermée permet de « fermer le monde », c'est-à-dire requêter par la négative ;
- I (inverse) : la propriété *a_parent* est déclarée inverse de la propriété *a_enfant*. Ainsi, si Sophie est l'enfant de Claire, alors Claire est le parent de Sophie ;
- N (cardinalité de rôles) : Pour exprimer le fait qu'on ne peut avoir que deux parents au maximum, on applique la restriction ≤ 2 à la propriété *a_parent*.

En termes d'expressivité, OLW2 apporte de nouvelles fonctionnalités par rapport à OWL1, et notamment :

- OWL2 permet d'intégrer l'approche méta-modèle en autorisant qu'une classe puisse également être une instance, c'est le « punning ». Par conséquent, avec OWL2, il est possible d'instancier des taxonomies à plusieurs niveaux, puisqu'il est possible de traiter des classes comme des instances. Il est également possible d'étendre une taxonomie existante, puisque toute instance peut à son tour devenir une classe ;
- OWL2 permet d'exprimer l'union disjointe (« disjoint union of »), qui permet d'exprimer le « ou » absent de OWL1. Ainsi, si un **humain** peut être un **homme** ou une **femme** et rien d'autre, on déclare que **homme** est l'union disjointe de **femme** et **homme**. Les unions disjointes peuvent s'additionner (un humain est un homme ou une femme, enfant ou adulte) ;
- OWL2 permet d'instancier des propriétés négatives entre deux instances. Ainsi, on peut définir que Sophie *n'est pas la sœur de* Pierre en niant la propriété *a_soeur* entre les instances `Pierre` et `Sophie` ;
- OWL2 étend les cardinalités des propriétés avec les notions *min*, *max* et *exact* ;
- OWL2 ajoute la disjonction de propriétés. Par exemple, les mêmes instances ne peuvent être reliées par les propriétés *a_parent* et *a_enfant*. Ces propriétés sont disjointes ;
- OWL2 introduit les chaînes de propriétés. Par exemple, la propriété *a_oncle* est la chaîne *a_parent*$\circ$ *a_frere* ;
- les types de valeur sont étendus dans OWL2, avec par exemple la possibilité de définir des plages de valeurs (age >= 18) ;
- OWL2 étend les annotations.

La logique des langages ontologiques peut également être représentée par deux boîtes : la Tbox et la Abox [Kontchakov et al., 2011]. Dans ce cas, la Tbox représente l'ensemble des relations (Rbox) ainsi que les taxonomies entre concepts : c'est le squelette de l'ontologie. La Abox représente l'instanciation de l'ontologie. Cette représentation est plutôt utilisée dans le cas du raisonnement, car un moteur d'inférence vérifie la consistance de la Abox vis-à-vis de la Tbox.

2.1.3 Le raisonnement

Le raisonnement est la capacité à inférer des informations. On peut différencier le raisonnement sur les ontologies de type OWL en quatre catégories [Fiorentini et al., 2010] :

- **la subsomption** *(subsumption)* explicite les liens taxonomiques ;

- **la récupération** *(retrieving)* permet de recenser toutes les instances d'une classe, ou ayant une propriété donnée ;
- **la vérification de la cohérence** *(consistency checking)* consiste à vérifier que l'ontologie définie n'a pas d'incohérence. Une façon de vérifier la cohérence est de vérifier que chaque classe peut avoir au moins une instance selon les axiomes définis. La vérification de la cohérence se fait à la fois sur le modèle et sur les instances ;
- **la réalisation** *(realization)* permet de trouver toutes les classes auxquelles une instance appartient.

Les requêtes de subsomption et de cohérence sont dites *extensionnelles*, car elles découlent d'une interrogation directe de la base de données, tandis que celles de type récupération et réalisation sont *intentionnelles*, car elles nécessitent une réflexion préalable sur les données [Tsarkov and Horrocks, 2006]. Le raisonnement des algorithmes les plus utilisés de OWL-DL est dit *sound* and *complete*. Cela signifie que pour une requête donnée, toutes les assertions trouvées justes seront véritablement justes (*sound*) et que toutes les assertions qui sont justes seront trouvées (*complete*).

Illustration des mécanismes de raisonnement

On peut illustrer les quatre mécanismes de raisonnement avec l'ontologie « famille » définie précédemment :

- subsomption : sur l'exemple de l'ontologie « famille », le raisonnement par subsomption aboutirait à déterminer par exemple que **fille** qui est sous-classe de **femme** est aussi une sous-classe de la classe **humain** ;
- récupération : on peut, à l'aide d'une requête, rechercher toutes les instances de la classe **fille**. Le raisonneur répond alors : Sophie et Claire.
- vérification de la cohérence : si l'on définit une instance comme étant à la fois **homme** et **fille** dans l'ontologie « famille », alors le raisonneur détecte une incohérence car toute **fille** est une **femme** et qu'une **femme** ne peut pas être un **homme** (les classes sont disjointes) ;
- réalisation : permet de déterminer que l'instance Claire qui est définie comme un **humain** appartient également aux classes **femme**, **mère**, **sœur** et **fille** de par ses liens avec les autres instances.

Ainsi, si l'on instancie l'ontologie famille telle qu'elle est définie dans la section 1.3.1 et l'annexe 7.1 à partir de l'instance Claire (définie comme une femme), on obtient :

- `Claire` *a_mere* `Sylvie` ;
- `Claire` *a_fille* `Sophie` ;

– Claire *a_fils* Simon ;
– Claire *a_frere* Pierre.

Alors, les rôles de chaque membre de la famille (c'est-à-dire les classes auxquelles les instances appartiennent) ainsi que les relations familiales implicites sont inférées par raisonnement. Par exemple, pour Claire, l'ontologie infère que :

– Claire *a_enfant* Simon ;
– Claire *a_enfant* Sophie ;
– Claire *a_parent* Sylvie ;
– Claire *a_frere_soeur* Pierre ;
– Claire $\sqsubseteq$ **mere** ;
– Claire $\sqsubseteq$ **fille** ;
– Claire $\sqsubseteq$ **sœur** ;

Le langage OWL2 présente trois profils d'implémentation qui conviennent chacun à des types d'applications :

– les ontologies dites « lourdes », c'est-à-dire composées d'un grand nombre de propriétés et de classes, peuvent se modéliser en OWL2-EL et permet un raisonnement DL ;
– les ontologies « légères » (peu de classes) mais possédant un grand nombre d'instances peuvent se modéliser avec OWL2-QL, lié à SQL, et permettant une implémentation sur des bases de données relationnelles ;
– Les ontologies avec un grand nombre et une grande diversité de restrictions se modélisent en OWL2-RL, permettant de raisonner par des approches fondées sur les règles (Jess, Prolog).

Le fonctionnement des raisonneurs

Dans les faits, pour les langages DLs, la vérification des deux types de requêtes (extensionnelles et intentionnelles) par le raisonneur peut se résumer à la vérification de la (non)-satisfiabilité de l'instanciation (la Abox) par rapport au modèle de concepts et de relations (la Tbox). La satisfiabilité est la vérification de la véracité ou de la fausseté d'une formule booléenne à partir de variables d'entrée et est à la base du raisonnement sur les DLs. Cela se fait au moyen d'algorithmes prédéfinis [Tsarkov and Horrocks, 2006] [Motik et al., 2007]. A l'échelle du concept, cela revient à vérifier sa non-contradiction (c'est-à-dire la possibilité d'existence d'une instance). Par conséquent, les raisonneurs fonctionnent traditionnellement suivant le même schéma [Kontchakov et al., 2011] : une phase de pré-traitement qui est une sorte de réécriture et de préparation de l'ontologie

Raisonneur	Implémentation	Expressivité	Outil
Pellet	Java	SROIN	Protégé
HermiT	Java / OWL API	SROIN	Protégé
FACT++	FACT + C++		Protégé
Racer	FACT + C++ + nRQL	SROIQ	-

Table 2.4: Tableau comparatif des principaux raisonneurs

et de ses instances en un langage optimisé pour le raisonnement ; le raisonneur fait ensuite une *classification* pendant laquelle il ordonne les concepts par subsumption ; enfin le raisonneur vérifie la *satisfiabilité* de la Abox au regard de la Tbox.

Les raisonneurs principaux sont Pellet, FACT++, HermiT et RACER. Les algorithmes d'un raisonneur sont toujours définis en fonction de l'expressivité du langage DL choisi, et sont inutilisables sur un langage possédant une plus grande expressivité. Les trois raisonneurs Pellet, FACT++ et HermiT sont implémentés dans l'outil d'édition Protégé et permettent de raisonner sur OWL-DL. Ils incluent également certaines fonctionnalités de OWL2. RACER, lui, inclut le langage nRQL, permettant de gérer également SPARQL, langage de requêtes. Pellet utilise un langage d'implantation JAVA, tandis que FACT++ utilise les algorithmes FACT et le C++, tout comme Racer. Le tableau 2.4 propose un comparatif synthétique de ces raisonneurs.

2.1.4 L'expressivité et le raisonnement : un antagonisme

La décidabilité

L'expressivité des DLs et l'inférence sont donc les deux caractéristiques spécifiques des ontologies en tant que paradigme de modélisation. Cependant, l'expressivité et le raisonnement sont également deux caractéristiques antagonistes. En effet, le plus expressif des langages est celui qui permet le moins bon raisonnement [Baader et al., 2008]. Par exemple, OWL-Full est un langage qui a la même expressivité de base que OWL-DL, mais auquel on a adjoint celle de RDF : cela le rend alors indécidable [Antoniou and van Harmelen, 2009].

Définition de la décidabilité [Motik et al., 2007]

« La décidabilité est la capacité à réaliser un raisonnement dans un temps fini et limité. »

Ainsi, plus l'expressivité du langage augmente et plus le raisonnement s'en trouve complexifié et peut devenir indécidable. Par conséquent, le sous-langage OWL le plus souvent utilisé dans les applications industrielles est OWL-DL, parce qu'il représente un compromis idéal entre expressivité du langage et capacités de raisonnement. Récemment, des applications OWL2 émergent également, mais l'expressivité de OWL2 n'est pas encore pleinement supportée par les outils de modélisation, et c'est donc souvent une partie limitée de OWL2 qui est réellement utilisée. Par exemple, l'outil Protégé, développé par l'université de Standford, supporte le fait qu'une instance et une classe aient le même identifiant, mais aucun des raisonneurs associés (pellet, FACT++, HermiT) ne le prend en compte dans le raisonnement.

L'open World Assumption

L'inférence ontologique repose sur l'Open World Assumption (OWA) [Drummond and Shearer, 2006], qui s'oppose à la Closed World Assumption (CWA), et qui signifie que l'information contenue est considérée comme incomplète, c'est-à-dire que l'absence d'information ne conduit pas à sa négation. Les systèmes à monde fermé, à l'inverse, présupposent que l'absence d'information équivaut à une information fausse, ce qui n'est pas le cas de la plupart des applications industrielles [Matsokis, 2010].

Définition de l'Open World Assumption [Drummond and Shearer, 2006]

« si une proposition ne peut pas être prouvée comme vraie avec la connaissance disponible, le système ne peut pour autant conclure que cette proposition est fausse »

---

l'OWA a pour conséquence que les langages DLs gèrent difficilement la négation [Fiorentini et al., 2010]. En effet, puisqu'il est impossible de requêter sur l'absence d'une information, il est également impossible d'exprimer des axiomes "négatifs" ni d'exprimer des règles par la négative. Les applications qui nécessitent plus particulièrement de raisonner en monde fermé sont celles qui intègrent des règles complexes et notamment, des règles d'accès ou d'utilisation des données, comme par exemple l'application de [Beimel and Peleg, 2011]. Pour ce type d'applications, des travaux sont en cours pour permettre de raisonner sur des modèles OWL mais en suivant la CWA, comme les travaux de [Wang and Yu, 2011] qui proposent un nouveau raisonneur appelé BCAR.

Illustration de l'OWA et de la CWA

De simples requêtes sur l'ontologie famille permettent d'illustrer la différence entre un raisonnement OWA et un raisonnement CWA : dans cette ontologie, Sylvie est définie comme la mère de Claire, elle-même la mère de Simon. Aucun parent ne lui a été associé. A la question :

Sylvie a-t-elle une mère ?

Les réponses varient suivant que l'on raisonne sur l'OWA ou la CWA :

OWA : « ne sais pas »
CWA : « non »

Sylvie, comme tout être humain, doit bien avoir une mère. Dans ce cas, l'OWA assumption semble plus pertinente. Cependant, si la requête est cette fois :

Sylvie a-t-elle un frère ?

La réponse de la CWA est plus juste, puisque que Sylvie est fille unique. Le choix du mode de raisonnement a donc d'importantes répercussions sur le résultat de l'inférence, dont il faut tenir compte.

2.1.5 Les rêgles, les requêtes et la syntaxe

L'expression des règles

L'expression et l'exécution de règles sur les modèles OWL se fait principalement de deux manières : soit directement par l'expression d'axiomes OWL, comme nous l'avons vu précédemment, qui permettent de restreindre un concept ou d'y associer des instances automatiquement ; soit, si la règle est soumise à des conditions plus complexes, par l'écriture d'une règle SWRL.

Le Semantic Web Rule Language (SWRL) est aujourd'hui le seul langage compatible avec OWL qui intègre des règles [Fiorentini et al., 2010]. En réalité, ce langage est la

réunion de la sémantique d'OWL-DL et du langage de règle Rule Markup Language (RuleML). Les règles SWRL sont de type "implication", c'est-à-dire qu'elles suivent la syntaxe suivante [Mun and Ramani, 2011] :

$$antecedent \rightarrow consequent \tag{2.1}$$

Dans cette implication, le conséquent est réalisé (vrai) si l'antécédent est satisfait. L'antécédent et le conséquent sont tous les deux des conjonctions d'expressions OWL [Hirankitti and Xuan, 2011].

Exemple de règle SWRL sur l'ontologie « famille »

On peut écrire une règle SWRL qui permet d'exprimer que deux enfants qui ont un parent en commun sont frères et sœurs :

$$a\_parent(?x, ?y) \wedge a\_parent(?z, ?y) \rightarrow a\_frere\_soeur(?x, ?z) \tag{2.2}$$

Cette règle s'applique sur les instances de la relation *a_parent* et se lit comme suit : si x a pour parent y et z a aussi pour parent y, alors x et z sont frere et sœur. Les règles SWRL permettent donc d'instancier de nouvelles relations entre instances à partir de déductions sur les propriétés existantes.

L'expression des requêtes

L'intérêt des ontologies est également de pouvoir aisément émettre des requêtes sur les éléments du modèle. Le langage de requêtes aujourd'hui normalisé auprès du W3C (depuis 2008) est SPARQL. Il s'agit d'un langage encapsulant des fonctions, proche de la syntaxe SQL, mais dédié au langage RDF. SPARQL permet donc de requêter uniquement sur les « taxonomies », c'est-à-dire la partie RDF de OWL. Ainsi, il faut exécuter le raisonnement OWL afin d'établir l'ensemble des liens taxonomiques entre instances avant de pouvoir requêter en SPARQL. SPARQL s'impose maintenant comme l'un des standards pivot du web.

En 2011, SPARQL Inferencing Notation (SPIN), a été soumis auprès du W3C comme nouveau standard de requête. Les personnes qui ont officiellement soumis cette standardisation sont : Holger Knublauch de la société Top Quadrant -qui développe entre autre l'éditeur d'ontologies TopBraid qui intègre déjà SPIN ; James A. Hendler, professeur au Rensselear Polytechnic Institute aux USA ; et Kingsley Idehen de la société Openlinks

Software. Selon le document officiel de soumission [1], les caractérisitiques de SPIN sont les suivantes : « *fondé sur RDF, RDFS et SPARQL, SPIN implémente un langage de règles et de contraintes qui est aligné avec la vision du Semantic Web et des Linked Data, et qui est facile à comprendre et à implémenter. SPIN utilise une syntaxe pour SPARQL, basée sur RDF, qui permet de partager des requêtes SPARQL et de mettre à jour des opérations avec d'autres ressources RDF. Plus particulièrement, il est alors possible de lier des classes RDFS/OWL, par des règles et contraintes sur le formalisme de leurs instances. Ces règles s'exécutent sur des moteurs de requêtes SPARQL classiques, ce qui limite les coûts d'implémentation et d'apprentissage. SPIN propose également un vocabulaire qui permet de décrire de nouvelles fonctions SPARQL, et ainsi étendre le périmètre actuel des fonctions SPARQL tout en le rendant indépendant de la plateforme d'utilisation. Enfin, SPIN inclut un mécanisme qui permet de formaliser et de partager des templates SPARQL. Avec ces templates, il est possible de construire des modèles de plus haut niveau, qui réutilisent des blocs standards* ». SPIN est uniquement implémenté dans l'outil commercial TopBraid, de la société Topquadrant.

La syntaxe et l'identification des objets

Les langages ontologiques utilisent une syntaxe XML (eXtensible Markup Language). XML est un langage à balise, qui plus est extensible. Quelques-unes de ses extensions les plus connues sont : XHTML (pour les pages web), XSLT (pour l'interprétation et la transformation des fichiers XML), RSS (pour la mise à jour automatique des ressources web), SVG (format graphique), etc. L'avantage de XML par rapport à d'autres syntaxes -outre le fait qu'il est un standard répandu- est que, grâce à son formatage par balises, il est facilement compréhensible par un être humain.

Enfin, dans les modèles ontologiques, chaque objet (classe, instance ou propriété) est identifié de façon unique par son International Resource Identifier (IRI), qui est une généralisation des plus connus Uniform Resource Identifiers (URIs). L'IRI est un identifiant unique qui permet de représenter une ressource sur le web. Le fait d'identifier chaque ressource dans une ontologie permet le partage aisé de ces ressources, ainsi que le requêtage à distance. Cela facilite, par exemple, la mise à jour et la synchronisation des données entre plusieurs ontologies. D'ailleurs, le langage RDF a été l'un des développements fondateurs du web des données (Linked data), qui propose de mettre des données (et non plus des documents) sur le web, et de lier ces données entre elles par les URIs et des requêtes utilisant le protocole Hypertext Transfert (HTTP) [2].

1. http ://www.w3.org/Submission/2011/02/
2. http ://www.w3.org/standards/semanticweb/data

Conclusion de la section 2.1

Les langages ontologiques, puisque fondés sur la logique descriptive, présentent deux caractéristiques intéressantes : une expressivité large, ainsi que la capacité d'inférer des informations. De plus, ils s'appuient sur une syntaxe XML, largement répandue et supportée par de nombreux outils. Enfin, chaque objet est identifié de façon unique grâce aux IRIs.

Dans l'article [Fortineau et al., 2013b], nous avons réalisé un état de l'art des ontologies pour les applications industrielles en fonction des apports constatés dans la littérature : la complétude des modèles, la possibilité d'automatiser l'information grâce au raisonnement, et la flexibilité des modèles et de l'accès aux informations. Dans cette étude, nous cherchons à comprendre comment les ontologies peuvent permettre de parvenir à une fédération des informations liées au produit. Ainsi, les sections qui suivent étudient les contributions de la littérature selon un point de vue légèrement différent : il s'agit de comprendre les objectifs des ontologies pour la modélisation sémantique, et l'apport des ontologies pour dépasser les limites des approches unificatrices. Il se dégage de cette étude trois objectifs majeurs : permettre un partage d'information qui soit « sémantique » entre collaborateurs ; échanger des informations source de connaissance entre systèmes sans perte de sémantique ; et proposer des modèles riches en sémantique, capables d'intégrer un large spectre d'informations.

2.2 Les ontologies et la modélisation sémantique : état de l'art

2.2.1 Les ontologies : un moyen de procurer une sémantique riche

La richesse sémantique et les ontologies

Le terme de *richesse* sémantique est apparu assez récemment dans le domaine des ontologies pour le PLM. La première occurrence de ce terme survient dans un article de conférence en 2011 [Nazarenko et al., 2011], rédigé par l'équipe d'ONTORule [3], où il est question « d'enrichir sémantiquement » la base de gestion des règles par des annotations ontologiques. Cette notion est reprise dans [Wicaksono et al., 2011], puis dans [Landherr and Constantinescu, 2012] et enfin dans [van Ruijven, 2013]. Pour ce qui concerne les articles de revue, la première occurrence a lieu dans la contribution de [Al-Ashaab et al., 2012] qui précisent que le but est de produire un environnement « riche,

3. http ://ontorule-project.eu/

basé sur la logique et les faits » pour la chaîne logistique automobile. Le terme n'est alors utilisé qu'une seule fois. Il revient ensuite dans la contribution de [Barbau et al., 2012], rédigée par une équipe du NIST, et qui considère que le langage OntoSTEP permet « l'enrichissement des modèles de données produit par les ontologies ». Le terme apparaît également dans [Giovannini et al., 2012] et [Lee and Jeong, 2012] qui emploient la notion de « sémantiquement riche ». Cette notion est souvent reprise par la suite, notamment dans [Bechhofer et al., 2013], [Chandrasegaran et al., 2013] et [Song et al., 2013].

A la lecture de ces différentes contributions, et selon notre compréhension personnelle, nous définissons la richesse sémantique d'un modèle comme :

Définition de la richesse sémantique (proposée)

« la quantité, la qualité et la diversité de sens (en général métier) que le modèle est capable d'exprimer et de procurer aux informations qu'il contient. »

Nous avons identifié 13 articles, depuis 2008, qui traitent particulièrement de la « richesse sémantique » des ontologies, et ce, pour des applications industrielles. Comme le montre le tableau 2.5, la majeure partie des contributions se focalisent sur la phase de conception (10 articles sur 13), où il s'agit surtout d'améliorer la captation et la réutilisation de la connaissance métier, au travers d'ontologies. En effet, durant cette phase où convergent les informations de différents métiers, et où la prise de décision est importante et conséquente pour la suite du cycle de vie du produit, la bonne gestion des connaissances joue un rôle crucial sur les choix de conception, ainsi que la réduction du temps de développement des produits. Il n'est donc pas si surprenant que la plupart des applications des ontologies, liées à la richesse sémantique, se focalisent sur la gestion de la connaissance, en phase de conception. C'est en effet pour exprimer des informations complexes (comme la connaissance métier peut l'être), de sources hétérogènes, que la richesse sémantique du modèle est nécessaire.

Cependant, les travaux de [Giovannini et al., 2012], [Blomqvist and Ohgren, 2008] et de [Landherr and Constantinescu, 2012] montrent que la richesse sémantique procurée par les ontologies peut également permettre de mieux gérer la multitude d'informations qui co-existent pendant la phase de fabrication, afin d'optimiser les prises de décisions durant cette activité.

Contribution	Domaine	Phase concernée
[Song et al., 2013]	CAD/CAM	conception
[Lee and Jeong, 2012]	BIM/IFC	conception
[Giovannini et al., 2012]	fabrication durable	fabrication
[Landherr and Constantinescu, 2012]	KM	fabrication
[Barbau et al., 2012]	OntoSTEP	BOL
[Afacan and Demirkan, 2011]	connaissance universelle	conception
[Raza et al., 2011]	re-configuration	BOL
[Han and Park, 2009]	processus métier	conception
[Chen et al., 2009]	KM	cycle de vie
[Chang et al., 2008]	e-design	conception
[Brandt et al., 2008]	processus de conception	conception
[Blomqvist and Ohgren, 2008]	supply chain	fabrication
[Yang et al., 2008]	configuration de produit	conception

TABLE 2.5: Contributions scientifiques sur les ontologies et la richesse sémantique

La richesse sémantique au service de l'intégration des connaissances

Selon [Dekkers et al., 2013], l'un des obstacles majeurs à la mise en place du PLM est que les méthodes et outils actuellement développés n'ont pas encore intégré la gestion de la connaissance métier. En effet, les modèles ne sont pas assez matures pour intégrer toutes les informations nécessaires à cette captation de la connaissance métier. Toujours selon [Dekkers et al., 2013], les ontologies, notamment grâce à leur grande expressivité, pourraient permettre d'établir ce pont entre les modèles PLM actuels et la gestion de la connaissance métier, notamment en apportant de la richesse sémantique aux modèles, comme évoqué précédemment. Ainsi, le plébiscite des ontologies en phase de conception que nous avons observé a essentiellement pour but de permettre d'intégrer un plus large spectre d'information dans le SI, mais aussi, une information plus « intelligente » et contextualisée : c'est-à-dire, passer d'une gestion de l'information à une gestion de la connaissance.

L'ensemble des travaux du tableau 2.5 montre que pour parvenir à une gestion de la connaissance, il faut modéliser une sémantique métier. Le processus de gestion de cette sémantique métier est composé de deux actions réflexives [Landherr and Constantinescu, 2012] : la captation de la connaissance auprès des experts, puis sa réutilisation, que l'on souhaiterait la plus "intelligente" et automatisée possible. Entre les deux, un modèle métier permet de représenter l'ensemble des informations capitalisées. Comme le montre la figure 2.2, les apports des ontologies sont aux trois niveaux du processus :

- elles permettent d'améliorer l'intégration de la connaissance métier ;
- elles proposent des modèles métier qui représentent un plus large spectre d'infor-

mations ;
– elles permettent une réutilisation automatisée et intelligente des informations.

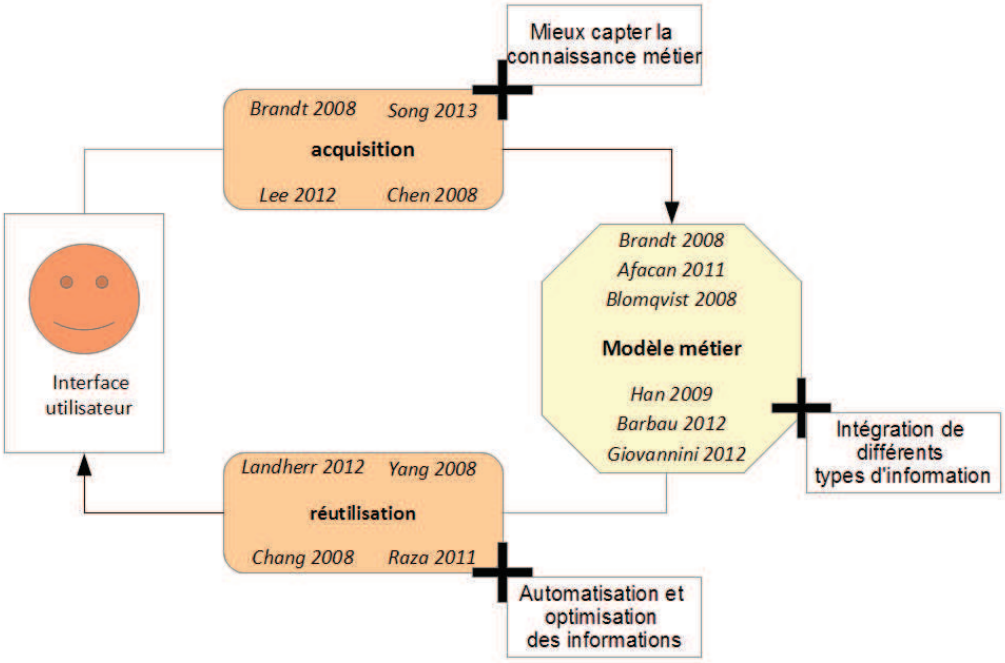


FIGURE 2.2 – L'apport des ontologies sur la gestion de la connaissance métier

Formaliser un plus large spectre d'information

La gestion d'une connaissance métier passe donc par la modélisation du domaine, autrement dit, une conceptualisation de l'expertise métier. Cette conceptualisation du métier prend tantôt le nom d'*ontologie d'entreprise* [Han and Park, 2009], [Blomqvist and Ohgren, 2008], d'*entrepôt des données* [Brandt et al., 2008] ou encore de *modèle unique de la connaissance* [Afacan and Demirkan, 2011]. Dans tous ces cas, l'apport des ontologies sur ce modèle métier est la possibilité, grâce à l'expressivité des langages DLs, de modéliser des informations de différents niveaux d'abstraction, et d'établir des liens sémantiquement riches entre ces informations. Alors, le modèle métier couvre un plus large spectre d'informations, et permet de modéliser des interactions plus complexes entre ces informations. Ainsi [Barbau et al., 2012], à travers la transformation du modèle produit STEP en ontologie (OntoSTEP), ont pu augmenter la sémantique du modèle, en y incluant entre autre les aspects « assemblage » de l'Open Assembly Model (OAM) et des meta-informations permettant de contextualiser le produit. Dans un autre contexte - celui de la fabrication durable - [Giovannini et al., 2012] utilisent les ontologies pour modéliser

les interactions entre le produit, les processus et les ressources (PPR). Ces interactions fines sont indispensables à la modélisation de la « durabilité », qui, en réalité, dépend de la possibilité ou non d'effectuer des changements sur la situation initiale, en fonction des relations du modèle PPR. L'ontologie proposée est centrée sur le produit, mais permet de modéliser les relations entre le produit, les processus et les ressources sous forme de fonctions. Cette ontologie permet donc finalement de faire converger et interagir des informations de sources hétérogènes.

L'information porteuse de sens peut être mieux exploitée

Sur ces informations variées et leurs propriétés sémantiquement riches, stockées dans l'ontologie d'entreprise, il est possible de faire des déductions : de raisonner. La richesse sémantique permet alors d'améliorer et d'automatiser le processus de récupération et de réutilisation des informations [Landherr and Constantinescu, 2012]. Ainsi, l'objectif de [Chang et al., 2008] est de modéliser les processus métiers par les ontologies afin d'optimiser la prise de décisions pendant la phase de conception, tandis que [Raza et al., 2011] et [Yang et al., 2008] construisent des ontologies de domaine dont l'objectif unique est d'automatiser et d'optimiser une activité métier : il s'agit de re-configuration de ligne chez Ford pour [Raza et al., 2011] et de configuration de produit à la demande pour [Yang et al., 2008]. C'est la richesse sémantique, qui permet d'inclure dans les modèles des *meta*-informations, c'est-à-dire des informations autour du produit, qui procurent dans les deux cas ces résultats optimisés et automatisés.

Un modèle porteur de sens est mieux adapté aux métiers

Enfin, la richesse sémantique a pour objectif une meilleure captation de la connaissance métier auprès des experts. Comme le précisent [Chen et al., 2009], le passage à une approche ontologique pour l'intégration des connaissances permet, dans un contexte de collaboration - et contrairement aux méthodes heuristiques, de logique floue ou multi-agent- de réellement manipuler de la sémantique. Grâce à l'expressivité des ontologies, il est possible d'établir des modèles de connaissance orientés métier, ainsi que d'implémenter des mécanismes d'inférence qui déduiront de nouvelles informations à partir de l'instanciation du modèle par les utilisateurs métiers. Il n'est donc pas nécessaire de renseigner le modèle de façon exhaustive, ce qui permet de gagner du temps, et rend moins pénible cette tâche pour l'utilisateur. De plus, comme l'expliquent [Lee and Jeong, 2012], le problème est que les modèles actuels sont souvent si complexes qu'ils ne permettent pas à un utilisateur unique de comprendre entièrement l'information

qu'ils contiennent. Les ontologies facilitent alors la définition de systèmes distribués, qui permettent de s'adapter réellement au besoin de chaque type d'utilisateur, qui sera alors plus enclin à le renseigner, et pourra y intégrer plus d'information.

L'un des domaines majeurs de l'utilisation des ontologies est donc la gestion de la connaissance métier en phase de conception. Plus généralement, ce qui est mis en avant dans la littérature est la capacité des ontologies à modéliser une sémantique riche, et donc, à faire converger de façon sémantique (conceptuelle) des informations de types et de sources hétérogènes. Alors, les ontologies permettent une représentation plus fidèle du domaine métier, mais facilitent également la captation et la réutilisation des informations par ces mêmes utilisateurs métier. La gestion de la connaissance n'est cependant pas l'unique domaine dans lequel on note une utilisation croissante des ontologies. Pour ce qui concerne l'échange d'information, dans un contexte économique poussant fortement à la collaboration intra ou inter-entreprise, les ontologies sont également utilisées pour définir des langages unificateurs et standards.

2.2.2 Les ontologies pour l'échange d'information et la collaboration

L'échange d'information par les modèles d'unification

Le deuxième objectif majeur de l'utilisation de modèles ontologiques est donc l'échange d'information. En effet, une information du PLM a pour vocation d'être échangée : entre les métiers, entre les phases du cycle de vie, entre les applications. Elle permet de transmettre des données porteuses de sens. Comme nous l'avons précédemment évoqué, les problématiques d'interopérabilité sont par conséquent importantes dans une approche PLM.

Pour pallier le manque d'interopérabilité sémantique des systèmes applicatifs actuels, des modèles pivots ont été développés, éléments de base des approches d'unification. Parmi ces modèles, certains ont été standardisés, afin de permettre un partage générique entre les systèmes, sans connaissance a priori des systèmes tiers. Récemment, de nombreux travaux ont visé à transformer ces standards vers des modèles fondés sur les ontologies, forts de l'idée que le paradigme de modélisation des ontologies permettait d'améliorer l'échange sémantique des informations. De plus, la modélisation par les ontologies apportant de nouvelles possibilités, des modèles *ad hoc* ont été développés, et ont vocation à devenir des standards.

La figure 2.2.2 présente une cartographie globale de l'ensemble de ces modèles, qu'il

s'agisse des transformations de standards vers des modèles ontologiques (représentées par des rectangles) ou de modèles produit ontologiques *ad hoc* (représentés par des cercles). Les modèles y sont positionnés en abscisse en fonction de la (ou des) phase(s) du cycle de vie qu'ils couvrent, et en ordonnée en fonction du niveau d'abstraction, qui va, depuis l'origine, du niveau applicatif (niveau de détail maximum, spécifique à une application en particulier) vers le niveau syntaxe (niveau d'abstraction d'un langage), en passant par les niveaux de domaine (regroupement de plusieurs métier connexes), et le niveau méta-modèle, qui est une vue globale sur plusieurs phases du cycle de vie.

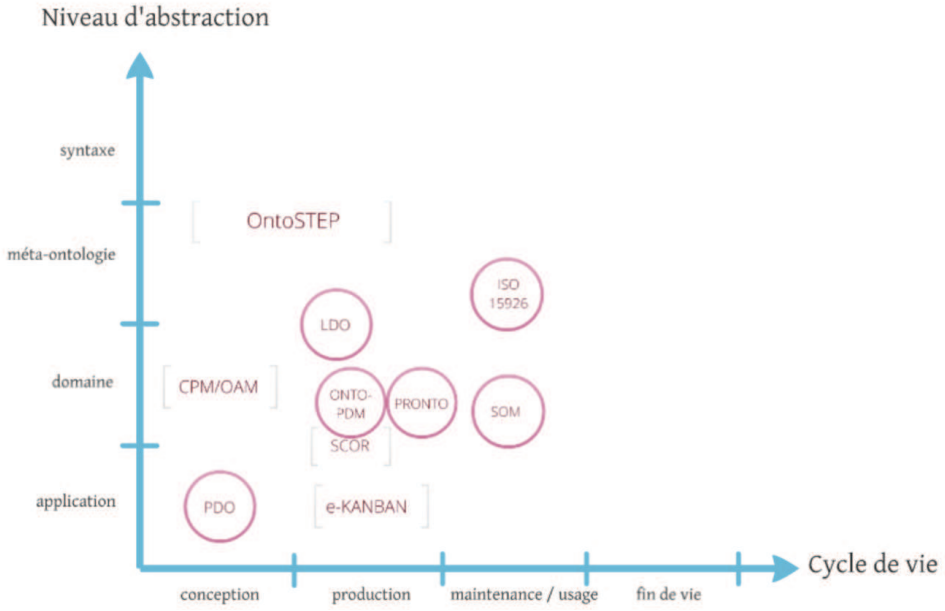


FIGURE 2.3 – Cartographie des modèles ontologiques d'unification

Les transformations de modèles d'unification en ontologies

La plupart des travaux sur les transformations de modèles existants sont menés par des équipes du National Institute for Standard and Technology (NIST) aux Etats-Unis, et notamment, ceux sur le CPM/OAM, l'E-Kanban et OntoSTEP.

CPM/OAM : une ontologie pour les modèles de conception.

Le Core Product Model (CPM), dont le développement a débuté il y a un peu plus de 10 ans, est un modèle produit de conception, exprimé dans un premier temps en UML

[Fenves, 2001]. Comme expliqué par [Fenves et al., 2007], « les objectifs premiers de la création du CPM étaient de proposer un modèle de données commun pour les recherches internes au NIST ainsi qu'une représentation des données au niveau opérationnel pour modéliser le flux d'information de conception qui est multi-niveaux ». Le CPM est un modèle générique pour représenter le produit en conception, qui repose sur l'entité originelle de *l'artéfact*, qui représente de façon équivalente un composant, une partie ou un sous-ensemble : il s'agit d'un « morceau » du produit. Cet artéfact peut ensuite avoir trois types d'attributs : un forme, une fonction ou un comportement. Par ces trois attributs, le CPM entend adresser le début du cycle de vie : les fonctions permettent en pré-conception de modéliser les exigences, la forme permet ensuite de représenter le produit pendant sa conception, et enfin, le comportement est utile lors de la phase de fabrication (*usinable*, par exemple, peut être un comportement d'un composant).

L'Open Assembly Model (OAM) est une extension du CPM pour gérer plus particulièrement la sémantique des assemblages [Baysal et al., 2004]. En 2007, [Fiorentini et al., 2007] présentent l'ébauche d'une transformation de l'OAM en ontologie. Cette transformation exige de proposer une version ontologique du CPM, et a pour objectif d'augmenter l'OAM, des capacités d'inférence des ontologies. Des axiomes OWL sont donc adjoints aux classes de l'OAM. Cependant, le raisonnement DL étant limité, des règles SWRL sont utilisées pour augmenter le raisonnement. Ces règles sont de trois types : association (pour contourner le fait que les relations sont définies dans le modèle comme des instances d'une classe et non comme des propriétés), PartOf (pour permettre d'exprimer la transitivité des assemblages) et les restrictions (qui expriment les impossibilités d'assemblages). Il est à noter que puisque SWRL ne permet pas d'exprimer des règles négatives, cette dernière catégorie de règles est obtenue par contournement, en créant des classes et propriétés artificielles.

OntoSTEP : la transformation de STEP en ontologie.

Nous en avons déjà brièvement parlé, OntoSTEP est la transformation de STEP en ontologie. STEP étant une norme dont le champ est assez vaste, les travaux de transformation ont commencé en 2008 par la transformation du langage de modélisation EXPRESS, sur lequel repose la norme STEP en langage ontologique [Zhao and Liu, 2008], plus précisément par la transformation des fonctionnalités d'EXPRESS en OWL-DL [Krima et al., 2009]. Dès ces premiers travaux, l'objectif de la transformation est que OntoSTEP propose une sémantique plus grande que STEP. La première étape de cette extension étant l'intégration de l'OAM/CPM. La transformation sémantique réalisée n'est pas exactement complète : certaines fonctionnalités d'EXPRESS ne sont pas modélisables en OWL (par exemple, le

concept d'abstraction) ou doivent être modélisés de manière détournée (par exemple, le concept de duplication). Cependant, ces verrous ne s'avèrent pas limitant pour la transformation. Dans ces premiers travaux, seul l'AP 203 de la norme était concerné.

Plus récemment, [Barbau et al., 2012] ont réalisé l'ambition de [Krima et al., 2009] en intégrant de nouveaux AP dans OntoSTEP, ansi que la version ontologique du modèle CPM/OAM, proposant ainsi un modèle produit de conception assez large et extensible. Dans ce travail, les auteurs rencontrent les mêmes problèmes de transformation, mais qu'ils concluent ainsi : « *traduire une structure d'information basée sur un paradigme de modélisation et un langage particulier présentera toujours des problèmes en termes d'inexactitude des concepts et de perte sémantique à la traduction, mais, malgré ce problème, traduire les entités d'EXPRESS en concepts OWL est un moyen d'exploiter la rigueur de OWL. L'OntoSTEP présentée dans ce papier est un nouveau pas vers un enrichissement sémantique des informations produit. OntoSTEP peut facilement être intégré dans d'autres ontologies OWL* ». Les auteurs précisent également, qu'OntoSTEP n'est pas un remplacement de STEP, mais une autre manière de représenter des modèles compatibles avec STEP, ce qui permet, à nouveau, de relativiser les différences sémantiques ainsi que les limites d'échelle qu'imposent le raisonnement OWL, mais entre en contradiction avec l'idée selon laquelle OntoSTEP augmente la sémantique de STEP : en effet, si ces deux modèles ne sont pas contraints à posséder une sémantique commune, et peuvent cohabiter, il est possible à terme, que leurs sémantiques diffèrent finalement d'une façon qui ne permette plus de garantir leur compatibilité réciproque.

L'E-Kanban : une ontologie pour modéliser les flux de production.

L'E-Kanban est l'implémentation numérique du système de gestion des flux : le KANBAN. Pour modéliser ce système KANBAN numérique, le NIST a proposé en 2007 de représenter le système KANBAN par une ontologie [Barkmeyer and Kulvatunyou, 2007]. Cette ontologie permet de représenter le processus industriel contrôlé par le KANBAN, ainsi que les rôles de chaque partie prenante. L'intérêt selon Barkmeyer de modéliser ce système par une ontologie est que, au-delà de modéliser une réalité de façon numérique, l'ontologie peut représenter la logique du système et en tirer des raisonnements. Dans cette application ce sont à nouveau les capacités d'inférence des ontologies qui sont plébiscitées. Le système E-Kanban est aujourd'hui mis en place dans de nombreuses entreprises, en témoignent notamment les travaux de [Al-Hawari and Aqlan, 2012, Hu et al., 2012, Oh and Shin, 2012].

SOM : une ontologie pour le milieu de vie et pour le PLM.

Le Semantic Object Model est un modèle produit exprimé en premier lieu selon la syntaxe UML, dont l'objectif est de modéliser le produit tel qu'il est fabriqué ainsi que son évolution au cours de son utilisation. Ainsi, le SOM est un modèle qui se concentre surtout sur la phase du milieu de vie (MOL), ce qui en soit le différencie déjà des autres modèles produits. Mais le SOM a vocation à modéliser l'ensemble du cycle de vie, et donc est étendu aux autres phases. Ce modèle a été développé dans le contexte du projet PROMISE FP6 [4].

En 2011, [Matsokis and Kiritsis, 2011] proposent une transformation et une extension du modèle SOM qui se fonde cette fois sur la syntaxe OWL (et non plus UML). Les classes étendues sont alors conçues directement en fonction des fonctionnalités de OWL, afin d'optimiser les capacités de raisonnement des DLs. SOM est donc le premier modèle où les ontologies sont employées en tant que nouveau paradigme de modélisation, impliquant une nouvelle façon de concevoir, et non seulement comme un langage de représentation classique à forte sémantique. Dans le cas du modèle SOM, un autre intérêt de l'utilisation des ontologies est de pouvoir étendre et partitionner le modèle. En effet, en fonction des applications, une partie limitée du modèle est utilisée, et puisque la sémantique employée est compréhensible facilement par un utilisateur, le modèle peut alors être étendu et diversifié en fonction des besoins et spécificités de chaque application. Cependant, il n'est pas précisé comment ces extensions inter-opèrent, et comment les correspondances sémantiques entre représentations de conception, de fabrication ou de maintenance par exemple (que l'on sait extrêmement différentes) sont effectuées.

Les nouveaux modèles d'unification fondés sur des ontologies

Product Design Ontology : une ontologie de conception.

La Product Design Ontology (PDO) est une ontologie à destination des concepteurs, plus particulièrement dédiée à l'échange d'éléments surfaciques de conception (objets CAO) [Catalano et al., 2009]. La PDO a été conçue dans le cadre du projet Aim@Shape [5]. L'originalité de la PDO est d'ajoindre à la définition surfacique des objets de conception des méta-informations telles que les fonctions ou l'utilisation. L'objectif est d'embarquer avec la définition du produit la connaissance métier sous-jacente. Ainsi, les informations

4. http ://www.promise.no/
5. http ://www.aimatshape.net/

échangées entre les concepteurs sont riches de sens.

OntoPDM et PRONTO : deux ontologies produit pour l'ensemble de la chaîne logistique

La Product Ontology PRONTO est une ontologie définie par [Gimenez et al., 2008] et [Vegetti et al., 2011]. Elle nait d'une réflexion sur les nomenclatures de conception et de fabrication, étendue ensuite à la problématique de la planification et la fabrication des produits complexes de manière générale. PRONTO utilise des règles et l'inférence pour optimiser les activités de planification et de commande, si bien qu'elle est considérée comme une ontologie de la supply-chain.

Onto-PDM est une ontologie de domaine, centrée sur le produit, et qui sert de modèle noyau pour l'interopérabilité en phase de production [Panetto et al., 2012]. Onto-PDM a le même périmètre que PRONTO, mais la particularité du modèle Onto-PDM est de se fonder sur deux standards : STEP (ISO 10303) et IEC 62264, afin de permettre l'interopérabilité entre les données de CAO, les PDM, les MES et les ERP. Onto-PDM est exprimée en syntaxe UML, et n'utilise donc pas les capacités d'inférence des ontologies.

Linked Design Ontology : un modèle générique pour la gestion des connaissances.

La Linked Design Ontology (LDO) est une ontologie développée dans le cadre du projet européen FP7[6] dont l'objectif est de permettre l'extraction, la structuration, l'échange et la réutilisation de la connaissance, ce, de manière générique, en s'appuyant sur l'analyse de trois cas industriels très différents [Kiritsis et al., 2012]. L'ambition de la LDO est ainsi d'être réutilisable dans toutes entreprise de conception ou de production avec un très faible effort d'adaptation. Pour ce faire, la LDO est organisée comme un réseau d'ontologies unitaire, dont chacune remplit une fonction particulière. Pour unifier ce réseau, une ontologie noyau est définie. Elle repose sur les concepts suivants :

- *Task* : ensemble d'opérations planifiées ;
- *Process* : ensemble des processus ;
- *Product* : ensemble des produits ;
- *Subsystem* : ensemble des sous-systèmes composant un produit ;
- *Component* : ensemble des composants des sous-systèmes ;
- *Resource* : ensemble des éléments utiles à la fabrication d'un produit ;
- *Agent* : ensemble des agents ;

6. http ://cordis.europa.eu/fp7/home_en.html

- *LCP* : ensemble des phases du cycle de vie ;
- *Event* : liste des évènements ;
- *TypeEvent* : décrit les types d'évènements ;
- *Factor* : contraintes relatives au produit qui doivent être examinées ;
- *Datatype* : décrit les données obtenues ;
- *Indicator* : indicateur.

Il est remarquable que la LDO - qui est le projet le plus récent - repose sur une nouvelle façon de percevoir le standard : seuls des concepts fondamentaux sont génériques, afin d'être échangés et même partagés, tandis que leur spécification au travers d'ontologies différentes reste spécifique au domaine concerné.

L'ISO 15926 : une norme pour l'interopérabilité

La norme ISO-15926, qui s'intitule « *Systèmes d'automatisation industriels et intégration - Intégration des données cycle de vie pour les usines de process, incluant les usines de de production de pétrole et de gaz* »[ISO/15926-1, 2004], est un standard d'échange des données structuré en 11 « fascicules ». La part 4 de cette norme prévoit la construction d'une bibliothèque de données de référence (RDL), support sémantique à l'échange des données. Pour la formalisation de cette RDL, la part 8 de la norme préconise le langage OWL. Ainsi, l'ontologie partagée qu'est alors la RDL permet une convergence sémantique et métier entre les différents partenaires. L'emploi du langage OWL se justifie alors pour la richesse sémantique qu'il procure, mais aussi par la gestion possible de données non-canoniques. Une application de cette norme, notamment la méthodologie de construction de la RDL pour un cas industriel, est décrite par [Fiorentini et al., 2013].

2.2.3 Les ontologies pour le partage de la sémantique

Le partage de sémantique entre les utilisateurs

La définition de l'ontologie stipule qu'il s'agit d'une représentation formelle d'une conceptualisation *partagée*. Le terme « partagée » est important : il rappelle que les informations représentées formellement par des ontologies ne doivent pas seulement être échangées : il doit exister un partage mimimum des concepts de l'ontologie entre les différents utilisateurs afin qu'ils comprennent les informations échangées de la même manière. Sans quoi, les bénéfices amenés par les ontologies se trouvent limités [Staab and Studer, 2009]. Ce besoin de « partager » une sémantique commune correspond à la notion de commensurabilité des schémas interprétatifs en gestion des connaissances, que nous avons évoquée à la section 1.1. Dans les deux cas, il est nécessaire, pour qu'il y

ait une interprétation juste et constante des données échangées, de partager des notions principales entre les différents utilisateurs.

Face aux problématiques d'interopérabilité, et depuis l'émergence des outils ontologiques, le partage d'une sémantique commune se révèle un levier potentiel important. Comme exposé par [Chen et al., 2009], « *comparé aux efforts conjoints de différentes entreprises pour collaborer sur les activités du cycle de vie du produit, comme la coordination, la communication ou le contrôle, la capacité à partager de manière effective l'information autour du produit ainsi que la connaissance est plus importante* ». La question qui se pose alors est : comment faire en sorte que l'information échangée ait un sens univoque pour l'ensemble des collaborateurs ?

Les modèles sémantiques de partage

L'ISO 15926 part 4 : définition d'une RDL

L'une des initiatives qui tente de lever ce verrou est la norme ISO 15926. L'originalité de cette norme face à d'autres modèles standards est qu'elle inclut dans sa part 4 la définition d'une bibliothèque de référence (RDL). L'objectif de cette RDL est de définir de manière unique et partagée les éléments-types qui seront utilisés par l'ensemble des protagonistes. La norme prévoit également que l'échange d'information entre utilisateurs fasse systématiquement appel à cette RDL, afin de guider l'interprétation des informations. Comme nous l'avons vu auparavant, l'ISO 15926 préconise de modéliser la RDL et le modèle d'échange sous forme d'ontologie.

La définition d'une RDL a aussi été considérée pour le standard Product Life Cycle Support (PLCS) afin d'améliorer la représentation des informations échangées. Pour PLCS, comme pour tous ces modèles standards, l'ISO recommande de formaliser la RDL dans un langage ontologique, et plus particulièrement en OWL [Price and Bodington, 2004]. Cette unification de la formalisation des RDL participe à l'ambition de permettre, un jour, d'échanger des données aisément entre les différentes RDL ISO.

Les raisons invoquées par [Price and Bodington, 2004] pour choisir OWL comme langage de formalisation sont :

- la sémantique de OWL satisfait les exigences de la RDL ;
- l'Object Management Group travaille au développement d'un outil de représentation graphique de RDL fondé sur OWL ;
- de par sa syntaxe XML, OWL est facilement implémentable ;
- en choisissant OWL, la réutilisation de données de la RDL 15926 est envisageable ;

- l'industrie et le monde académique supportent et développent des outils (open source) pour OWL ;
- OWL supporte de grandes librairies ;
- les technologies du web sémantiques vont avoir une utilisation toujours plus grande dans les applications « cycle de vie », choisir OWL permet de pousser dans cette direction.

Les ontologies d'entreprise et les modèles métier de partage

La définition d'un modèle métier, ou du moins d'un dictionnaire de termes, dans le domaine de la gestion des connaissances participe du même objectif : s'accorder sur les notions fondamentales afin de donner du sens à l'information échangée. Cet objectif est celui du travail de [Young et al., 2010] et [Chungoora et al., 2013] : pour améliorer l'interopérabilité sémantique entre les phases de conception et de fabrication, un cadre de modélisation multi-niveaux est proposé, s'appuyant sur la définition d'ontologies de domaine (de conception et de fabrication), qui partagent une sémantique commune, modélisée dans une « core ontology ». Ce cadre de modélisation permet alors la vérification de la connaissance modélisée.

Les modèles partagés n'ont cependant pas pour unique but d'améliorer la compréhension des informations échangées : ils cherchent également, comme la Linked Design Ontology [Kiritsis et al., 2012], à diminuer la charge de conception et de population de l'ontologie en réutilisant et spécialisant, pour des applications proches, une même ontologie générique partagée. Cette idée de réutilisation des efforts de conceptualisation, en plus de l'amélioration de la qualité de l'information échangée est également se qui a justifié la définition de dictionnaires de termes et autres *thesaurus*.

Conclusion de la section 2.2

De par leur expressivité et l'inférence qu'elles permettent, les ontologies ont dans un premier temps trouvé des applications dans le domaine des modèles pour la gestion des connaissances. Cependant, la richesse sémantique étant un des freins à l'application des modèles d'unification, des transformations ontologiques de ces modèles ont été envisagées. On remarque au fur et à mesure du développement de standards ontologiques un changement d'approche : au standard d'unification comme modèle effectif d'échange se substituent des modèles génériques partagés, qui permettent d'unifier des modèles plus spécifiques. Les exemples notoires de ce type de démarche sont le projet européen FP7, qui donna jour à la Linked Design Ontology, ainsi que l'ISO 15926. Dans la section

suivante, nous illustrons ce type de démarche ainsi que l'exploitation possible des capacités expressives et d'inférence des ontologies au travers de l'étude de l'interface conception/fabrication.

2.3 Une illustration : l'interface conception / fabrication

Afin d'illustrer les conclusions de l'état de l'art, cette section présente de façon synthétique un premier travail exploratoire, réalisé en 2011 et présenté à la conférence PLM 11 [Fortineau et al., 2011] sur l'interface conception/fabrication.

2.3.1 Présentation du cas d'étude

Au début de ces travaux de thèse, nous nous sommes intéressés à une problématique industrielle récurrente, qui est l'interface entre conception et fabrication. Cette problématique a déjà été traitée par Thomas Paviot dans sa thèse [Paviot, 2010], notamment par le spectre de l'interopérabilité PDM/ERP. Dans cette contribution, nous avons prolongé la réflexion en nous focalisant sur les liens entre nomenclatures de conception (EBOM) et nomenclatures de fabrication (MBOM). Les significations de ces deux vues d'un même produit, bien qu'elles semblent très proches d'un point de vue graphique, sont éminemment différentes : l'EBOM représente un unique produit virtuel, tandis que la MBOM représente un processus de production, utilisé pour le calcul des besoins en fabrication [Stonebraker, 1996, Orlicky, 1973]. De manière générale, la représentation par nomenclature des produits de conception et de fabrication souffre des limites suivantes :

- un manque de sémantique : les nomenclatures sont constituées d'objets (les cadres) et de liens (les traits) ce qui leur procure une structure d'arbre (on parle souvent « d'arbre de conception »). Cependant, la signification d'un trait ou d'un cadre n'est pas la même dans les deux nomenclatures, et n'est pas claire dans chacune : il y a quantité d'informations implicites dans ces deux représentations [Paviot et al., 2010] ;
- un manque d'échange dynamique entre les deux vues : puisque la sémantique des représentations est faible, le mapping sémantique entre les deux nomenclatures est difficile, et ne permet pas l'actualisation aisée et automatique des informations. Cet état de fait explique en partie les difficultés d'interopérabilité entre systèmes de conception et systèmes de fabrication ;
- les deux nomenclatures ne partagent pas une sémantique commune : entre EBOM et MBOM, seule la représentation graphique est commune. Mais mêmes les objets graphiques ne portent pas le même sens : par exemple, un objet de conception,

bien que virtuel, est toujours l'expression d'une partie du produit, tandis que la nomenclature de fabrication inclut des articles fantômes. Ainsi, l'une et l'autre des nomenclatures ne connaît rien de la seconde : elles sont incapables de dialoguer, puisqu'elles ne partagent pas la même sémantique.

Pour illustrer ces limites, nous nous intéressons à une nomenclature extrêmement simple, extraite de la représentation d'une voiture, et donnée sur la figure 2.4. Du point de vue du concepteur (à gauche), le morceau de voiture (*c_voiture*) est constitué d'une porte - elle même constituée d'une plaque de métal et d'une vitre - d'un châssis et d'un moteur. Lors de la fabrication cependant [7], deux tôles sont assemblées de part et d'autre de la vitre pour former l'assemblage *f_assemblage2*, monté ensuite sur le châssis et le moteur, soudés ensemble auparavant, pour former le *f_assemblage1*. Cet exemple très simple permet alors de mieux comprendre les différences sémantiques entre EBOM et MBOM.

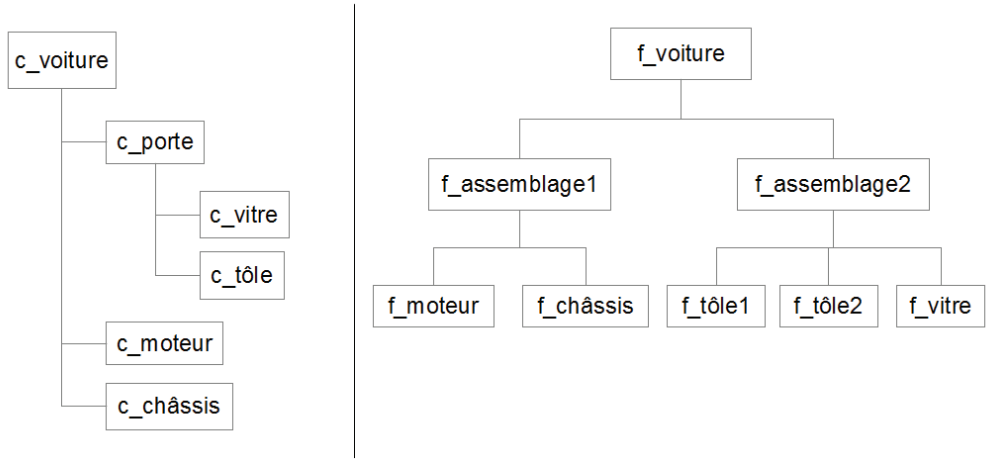


FIGURE 2.4 – Les nomenclatures de conception et de fabrication du cas d'école.

2.3.2 Une ontologie pour l'interface conception / fabrication

La sémantique de l'ontologie d'interface

Pour palier aux limites sémantiques des deux représentations actuelles du produit, nous avons conçu une ontologie d'interface, qui établit un mapping sémantique entre les deux nomenclatures en les exprimant dans une sémantique commune et en modélisation les « meta-informations » des deux représentations. Le modèle de cette ontologie est présenté de manière synthétique sur la figure 2.5, et plus en détail à l'annexe 7.2.

7. Les cardinalités de la MBOM sont toutes égales à 1

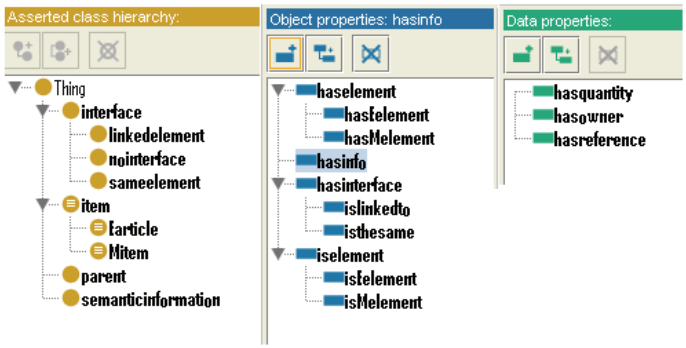


FIGURE 2.5 – L'ontologie pour l'interface conception/fabrication

Cette ontologie permet de représenter le produit en conception (classe **Eitem**) et en fabrication (classe **Mitem**), mais modélise également l'interface entre ces deux représentations (classe **interface**) :

- **LinkedElement** est une classe contenant les `items` reliés entre eux par la propriété *is_linked_to*.
- **SameElement** est une classe contenant les `items` reliés entre eux par la propriété *is_the_same*, c'est-à-dire les éléments qui ont leur équivalent direct dans la nomenclature opposée. Cette propriété est donc de type *functional* et a pour inverse elle-même.
- **NoInterface** est une classe représentant les `items` auxquels aucune interface n'est attribuée. Cette classe a pour unique objectif de pouvoir raisonner « par la négative » afin de retrouver les items potentiellement problématiques : il n'y a normalement aucun item dans cette classe, si l'interfaçage est complet. Elle ne peut être peuplée que par une requête SPARQL.

Enfin, cette ontologie représente les meta-informations nécessaires à un bon échange entre les deux vues, via la classe **SemanticInformation**. Ces informations exploitent les propriétés datatype des ontologies.

Les apports de l'inférence

Grâce au raisonnement ontologique, il est alors possible de faciliter le travail d'interface entre conception et fabrication, puisque des règles de cohérence sont ajoutées et que les requêtes « métier » sont rendues possibles par la sémantique du modèle. Par exemple, la propriété *is_the_same* est de type functional et réflexive, c'est-à-dire qu'un article d'une des deux nomenclatures ne peut être relié qu'à un et un seul article de l'autre

nomenclature par cette relation, et de manière bijective. Il peut être également intéressant d'utiliser des règles SWRL pour ajouter automatiquement des informations entre les instances. Imaginons par exemple que certains articles de conception soient définis comme « prioritaires », c'est-à-dire qu'ils ont une place cruciale dans le produit et nécessitent un suivi particulier. Une règle SWRL de type : $priority\_item(?x), has\_interface(?x, ?y) \rightarrow priority\_item(?y)$ permet de transmettre cette priorité aux articles de la nomenclature de fabrication qui y sont liés. Par ce genre de modélisation, on permet, au sein même du modèle, un véritable dialogue entre les deux phases.

Cependant, il faut tout de même veiller à ne pas mélanger modèle d'information et exploitation des informations, mélange que l'ontologie rend plus aisé et malheureusement fort attrayant. Ce premier cas d'étude nous a permis de détecter ce biais, qui peut amener à de mauvaises modélisations des informations, car trop « comportementales » plutôt que représentatives. Ainsi, il semble important que l'inférence ontologique serve essentiellement à vérifier la cohérence des informations, et à automatiser leur instanciation par des axiomes logiques, et non à interpréter ou exploiter l'information selon le point de vue de l'utilisateur. Pour cela, un modèle d'exploitation est indispensable.

Nous avons donc réfléchi sur le cas d'étude à des règles de cohérence des informations. Il s'agit là d'un travail de réflexion sur le sens des nomenclatures et les liens entre conception et fabrication, que nous ne prétendons pas avoir traité de manière exhaustive. Du point de vue « modèle », ce qui est ressorti de cette réflexion est le besoin d'exprimer un grand nombre de règles par la négative, car la recherche des incohérences s'est souvent exprimée sur l'absence de propriétés des instances, ou bien sur les instances manquantes d'une classe. Or, nous en avons déjà parlé et en reparlerons dans cette étude, l'open world assumption limite considérablement la possibilité d'exprimer ce type de règles ou de requêtes, dites « par la négative ». Il est également apparu que le langage SWRL était limité pour exprimer les règles nécessaire à ce cas d'étude pourtant simple.

Illustration des limites de OWL et SWRL sur le cas d'étude

Nous avons souhaité par exemple exprimer une règle qui précise le domaine de définition des propriétés de type *hasinterface*. L'expressivité de OWL permet de définir que les domaines de définition (départ et arrivée) de cette propriété sont identiques : il s'agit de la classe **Item**. En effet, la propriété *has_interface* et ses sous-propriétés expriment les liens sémantiques entre deux articles de nomenclature. Nous voulions exprimer dans le modèle que le domaine de départ et d'arrivée étaient liés entre eux : en effet, si au départ de la relation se trouve un **Eitem**, à l'arrivée, il ne peut y avoir qu'un **Mitem** et vice et versa. La propriété *has_interface* relie certes deux articles entre eux, mais surtout

deux articles de nomenclature différente. OWL ne permet pas d'exprimer cette restriction.

Nous nous sommes alors orientés vers SWRL, mais avec la même difficulté : le langage SWRL ne permet pas la comparaison directe d'instance. La seule façon d'exprimer cette restriction est alors d'utiliser un moyen détourné : il faut créer une classe artificielle (appelée **error**), dans laquelle sont placées les instances d'articles s'ils sont reliés par la relation *has_interface* alors qu'ils sont de même nomenclature. Cela se fait à l'aide de deux règles SWRL :

$$Eitem(?x), Eitem(?y), hasinterface(?x, ?y) \rightarrow error(?x), error(?y)$$

$$Mitem(?x), Mitem(?y), hasinterface(?x, ?y) \rightarrow error(?x), error(?y)$$

L'incohérence du modèle est alors détectable au travers de la présence d'articles dans la classe **error**, donc par l'utilisation d'une requête. Si l'on souhaite s'affranchir de la requête, il suffit de déclarer la classe **error** disjointe de la classe **Item**, et l'incohérence sera directement détectée.

Une autre limite que nous avons pointée sur ce cas d'étude est le fait que dès que l'on complexifie le raisonnement, cela nécessite des requêtes multi-niveaux. Cela implique un biais, car la superposition de règles et de requêtes conduit rapidement à une exploitation des données, alors que l'objectif premier est de modéliser ces données. Un exemple de ce type de vérification complexe est la recherche des articles (items) qui n'ont pas d'enfants et pas d'interface. Ces articles sont des « puits » dans la nomenclature, et il est donc potentiellement faux (mais pas systématiquement) qu'ils n'aient pas de lien d'interface avec les articles de l'autre nomenclature. Il serait donc tentant d'automatiser un repérage de ces articles, et de forcer une instanciation de leur interface. Mais, les règles et requêtes nécessaires au repérage de ces articles sont négatives, donc difficiles à mettre en oeuvre, et, puisque le traitement n'est pas identique pour tous les articles, l'intervention de l'utilisateur est nécessaire : on franchit la barrière du modèle pour procurer une exploitation des informations.

2.3.3 Une nouvelle représentation graphique de la nomenclature

La nouvelle modélisation de l'interface possible par l'ontologie nous a également permis de repenser la façon d'utiliser les informations de cette nomenclature étendue, et en premier lieu, la représentation graphique des nomenclatures proposée à l'utilisateur.

Pour trouver l'élément graphique primaire qui permettra de représenter les éléments de la nomenclature, il faut chercher le sens et les ressemblances dans les éléments de nomenclatures actuels. Ces ressemblances sont les suivantes :

- que ce soit en conception ou en fabrication, les niveaux de nomenclature correspondent toujours à des niveaux de parentalité : l'élément père est alors « constitué de » l'ensemble de ses éléments fils ;
- dans les deux cas également, il existe un lien physique entre un ensemble d'éléments fils d'un même père. Par exemple, les 4 roues et le châssis d'une voiture ne sont pas liés physiquement deux à deux, mais il existe une boucle physique, permettant de les relier physiquement les uns aux autres.

Ces deux similitudes permettent de définir un élément primaire de représentation de l'ensemble $pere + fils$: il doit être constitué d'une boucle (qui représente le lien physique entre fils), reliée à un sommet (qui représente la parentalité). Le cône semble donc être la représentation idéale.

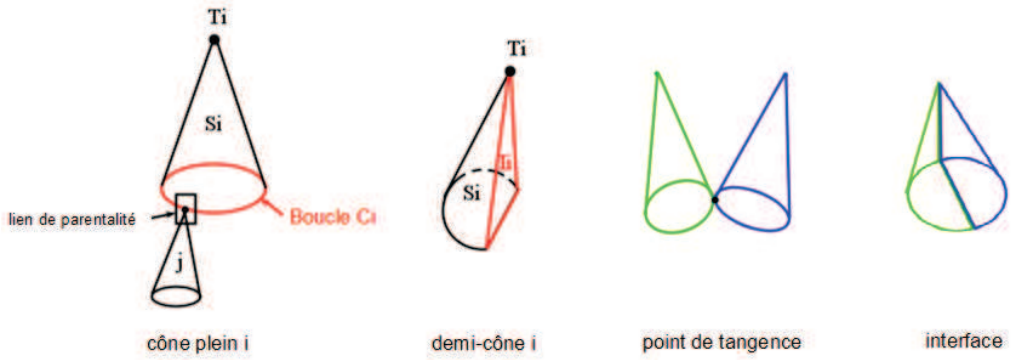


Figure 2.6 – Représentation des éléments sous forme de cônes

Dans cette représentation, visible sur la figure 7.6, tout élément devient donc un cône, dont le sommet est rattaché à la boucle de base de l'élément père (tout comme les autres éléments fils, qui sont alors mobiles sur un même cercle). Il est possible de complexifier cette représentation afin d'y inclure les éléments d'interface ainsi que les meta-informations du modèle ontologique. L'interface entre conception et fabrication se représente par des demi-cônes : un demi-cône représente un élément de conception ou de fabrication. La surface de découpe verticale du demi-cône devient alors surface d'interface. Lorsqu'un élément équivalent existe dans la nomenclature opposée, les deux surfaces d'interfaces se réunissent, et cela forme un cône plein. Si les deux éléments ne sont que partiellement liés, il y a alors une contrainte de tangence entre leurs boucles.

La représentation des nomenclatures prend alors une troisième dimension, et il reste la surface du cône pour y porter les meta-informations de l'élément. Une nomenclature

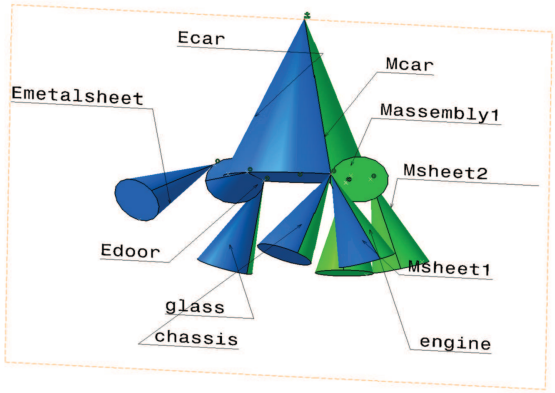


FIGURE 2.7 – Représentation des nomenclatures interfacées

complète de conception ou de fabrication est alors une suspension à plusieurs niveaux de demi-cônes. Cette suspension présente de nombreux degrés de liberté puisque les cônes fils peuvent se déplacer tout le long des boucles pères. Mais lorsque l'on fait intervenir les conditions d'interface, les deux suspensions se figent dans une position contrainte par leur interactions mutuelles, comme l'illustre la figure 2.7. Dans cette représentation, chaque élément graphique est alors porteur de sémantique.

Conclusion de la section 2.3

Ce cas d'étude à portée volontairement restreinte permet déjà d'appréhender à la fois le potentiel de modélisation qu'apportent les ontologies :

- une approche conceptuelle de la modélisation, ou les classes sont des « *propriétés* » qui affectent les instances ;
- la gestion de données non-canoniques, qui permet une description aisée de l'interface ;
- l'inférence comme support à l'interfaçage des informations, et à la vérification de la cohérence des données ;

mais aussi certaines limites, qu'il convient de confirmer sur un cas d'étude plus large :

- la difficulté à raisonner « par la négative » à cause de l'Open World Assumption ;
- le biais impliqué par les ontologies, qui poussent à réunir modélisation et exploitation des informations, ce qui peut conduire à des erreurs de modélisation.

Cependant, il est tout de même intéressant de remarquer par cette expérience que le changement de paradigme, que proposent les ontologies, permet de repenser les modèles, mais aussi l'exploitation des modèles sous un nouveau jour, et par là, peut être source d'innovation.

Dans cet exemple, nous avons créé l'ontologie de manière *ad hoc*, à partir du besoin qu'elle doit exprimer. Un autre concepteur aurait donc pu obtenir une ontologie différente, puisque nous n'avons pas suivi de méthodologie particulière. La section suivante présente les méthodologies de conception d'ontologies existant dans la littérature.

2.4 Les méthodologies de modélisation ontologique : état de l'art

Il existe plusieurs méthodologies pour créer des modèles ontologiques. A vrai dire, le terme « directives » semble plus adapté, car bien souvent, ces méthodologies proposent seulement une suite d'étapes génériques à suivre par les utilisateurs.

Methontology

Parmi toutes les méthodologies existantes, une des plus citées est la méthode METHONTOLOGY [Fernández-Lopéz et al., 1999], définie pourtant il y a plus de 10 ans. METHONTOLOGY a été établie pour la création d'ontologies dans le domaine de la chimie. Cette méthodologie se fonde sur une suite d'étapes décrites sur la figure 2.8, chacune de ces étapes nécessitant une acquisition, une documentation et une évaluation. La méthodologie commence par une spécification de l'ontologie, fondée sur l'approche « middle out » de [Uschold and King, 1995] puis une conceptualisation reposant sur un glossaire de termes métier. Les concepts ainsi définis doivent ensuite être formalisés afin d'y intégrer l'information. Enfin, l'ontologie doit être implémentée et maintenue. [Fernández-Lopéz et al., 1999] préconisent fortement la réutilisation d'ontologies existantes.

On-To-Knowledge

Plus récemment, [Staab et al., 2001] proposent avec On-To-Knowledge quatre étapes pour créer des ontologies de gestion de la connaissance. La proposition principale de On-To-Knowledge est de rompre dans le domaine de la gestion de la connaissance avec un paradigme de modélisation fondé habituellement sur la gestion documentaire, afin de tendre vers un paradigme fondé sur la connaissance elle-même. Alors, les quatre étapes de modélisation sont :

- identifier les besoins primordiaux de la connaissance ;
- mettre en évidence les concepts métiers de la connaissance ;
- décrire les processus de la connaissance ;

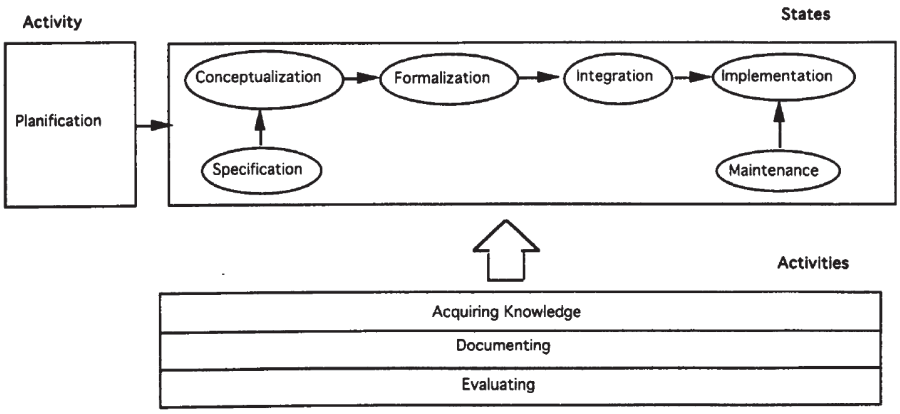


FIGURE 2.8 – Etapes de methontology [Fernández-Lopéz et al., 1999]

– construire une infrastructure pour gérer la connaissance.

Le processus de conception d'ontologies de [Staab et al., 2001], passe par une phase lancement au cours de laquelle l'objectif est de définir des concepts métiers fondamentaux, grâce à des interviews d'experts, concepts qui seront ensuite déclinés en sous-concepts. Mais comme le précise [Ottens, 2007], « ces méthodes (METHONTOLOGY et On-To-Knowledge) fournissent peu d'aide sur les modalités d'extraction des connaissances et sur la manière de choisir le contenu de l'ontologie. »

Anemone et Diligent

ANEMONE [Özacar et al., 2011] est une méthodologie plus spécifique que les précédentes. Bien qu'elle concerne elle-aussi la gestion des connaissances, elle s'adresse plus particulièrement à la définition d'ontologies modulaires et distribuées, c'est-à-dire d'ontologies multiples développées de manière collaborative en général. Anemone propose donc un cadre d'ontologies interopérables pour modulariser de façon unique des ontologies. Anemone présente les différents modules ontologiques envisagés mais ne propose pas de processus de conception de l'ontologie. A l'inverse, DILIGENT qui s'adresse également aux ontologies distribuées développées par plusieurs acteurs dont les objectifs et besoins peuvent différer [Pinto et al., 2004], établit un processus itératif en cinq étapes :

– construction collaborative d'une ontologie générique ;
– adaptation locale de l'ontologie générique aux besoins de chaque acteur ;
– analyse des adaptations spécifiques ;
– révision de l'ontologie générique partagée en fonction des analyses précédentes ;
– mise à jour locale des révisions ;

NeON

Finalement, le projet NeON[8] a pour ambition de réunir l'ensemble des méthodologies existantes dans une seule approche. Pour cela, Neon, qui se focalise sur la conception d'ontologies distribuées à large champ sémantique, propose une méthodologie qui repose sur des scénarios d'application [Suárez-Figueroa, 2010]. Neuf scénarios génériques ont été identifiés (construire un réseau ontologique de zéro, construire un réseau à partir de modèles non-ontologiques, etc.), et pour chacun de ces neuf scénarios, un cadre de modélisation est proposé, à partir d'étapes génériques. C'est essentiellement l'ordre d'application de ces étapes qui varie en fonction des scénarios. Neon propose également une vision « cycle de vie » du modèle ontologique, dont les étapes principales et leurs répétitions varient en fonction de l'activité de modélisation (itérative ou non).

Conclusion de la section 2.4

Les méthodologies existantes traitent des champs assez variés, mais aucune ne traite d'ontologies produit en particulier. Il est cependant remarquable que ces méthodologies présentent des similitudes dans leur approche :

- une première étape de focalisation sur la sémantique, qui conduit donc à la définition de concepts fondamentaux : le premier travail n'est donc pas sur l'inférence ou la logique du modèle, mais sur ses concepts et le sens qui leur est donné ;
- lorsque qu'il s'agit de modèles à plusieurs ontologies, leur réalisation passe nécessairement par la définition d'une ontologie partagée.

Surtout, l'un des apports principaux du projet Neon est qu'il est difficile de définir une méthodologie type pour la définition d'ontologies. Cela dépend notamment des pré-recquis du projet (y a-t-il un modèle existant ? Suit-on une approche montante -réunion d'ontologies d'application en une ontologie commune, ou descendante - définition d'ontologies spécifiques à partir d'une ontologie générique ?, etc.) En réalité, la méthodologie dépend surtout du type de modélisation que l'on souhaite définir, les ontologies n'étant qu'un outil pour formaliser ce modèle. Vouloir définir des méthodologies de construction d'ontologie de manière générique, et sans prendre en compte le domaine modélisé, ne peut alors conduire qu'à une approche biaisée et peu précise. Ainsi, dans le chapitre suivant, nous présentons le cadre de modélisation pour des modèles produits dans une vision cycle de vie, et nous adossons par la suite une méthodologie à ce cadre de modélisation particulier.

8. http ://www.neon-project.org

CHAPITRE 3

VERS UNE MODÉLISATION ONTOLOGIQUE DES INFORMATIONS

Résumé du chapitre

Ce chapitre décrit les propositions générales de cette thèse, qui sont un cadre et une méthode de modélisation des informations. Tout d'abord un cadre générique de modélisation sémantique des informations liées au produit est présenté dans la section 3.1. Il se décompose en trois niveaux de modélisation : un méta-modèle pivot pour unifier les données, des modèles de domaine supports à la capitalisation et à la réutilisation des connaissances métier, et enfin des modèles applicatifs pour instancier le modèle produit et mettre en oeuvre les informations partagées.

La section 3.2 présente ensuite une méthode générique d'application du cadre de modélisation, fondée sur l'hypothèse d'une modélisation ontologique des informations. Cette méthode traite des trois niveaux d'abstraction : de la spécification, de la conceptualisation et de la formalisation, tout en tenant compte des contraintes d'implémentation. Cette méthode permet ensuite de proposer un cadre ontologique de modélisation, qui s'appuie sur RDF et OWL.

3.1 La modélisation des informations du cycle de vie dans une approche fédérative

3.1.1 De l'unification à la fédération : approche générale

Proposition d'un cadre général de modélisation des informations

Les limites des approches d'unification montrent la nécessité de proposer un partage plus large de l'information, grâce à des modèles fondés sur des sémantiques riches et sur du raisonnement. On se situe alors à un niveau de granularité supérieur : au lieu d'établir directement les correspondances (mapping) entre les systèmes, comme c'est le cas dans une approche d'unification grâce aux transformations de langage, le but est de créer le lien entre les informations via l'utilisation d'un méta-modèle qui rapproche les sémantiques de chaque système. Une partie de la sémantique doit alors être commune, et communément partagée. Si l'on compare avec le domaine de la linguistique, la méthode d'unification revient à effectuer la traduction littérale des termes de chaque langue, à la manière d'un dictionnaire, alors que la méthode que nous envisageons, qui se rapproche de l'approche fédérative, est fondée sur le partage de sémantique et s'attache au sens des mots (concepts), définis dans un méta-modèle partagé. Ainsi, peu importe que le mot *métro* se dise *tube* en anglais ou *U-bahn* en allemand, l'important est que, pour chaque utilisateur, et quelle que soit sa langue, cela représente un *moyen souterrain de transport des personnes*. De plus, si un espagnol entre dans la conversation, il n'est pas nécessaire de faire de nouvelles traductions, il comprendra directement le sens du concept, et pourra y apposer son terme. Le but du méta-modèle est donc de prendre en compte le sens des données pour pouvoir s'affranchir de l'étape de traduction et transmettre automatiquement le flux sémantique d'un système à l'autre.

Le périmètre du cadre général de modélisation proposé dans cette thèse est représenté par le cadre bleu de la figure 3.1. Il s'agit de proposer une modélisation sémantique de l'ensemble des informations liées au produit, tout au long du cycle de vie. L'approche proposée doit permettre : de construire des modèles riches en sémantique, d'intégrer de nouvelles connaissances, de partager une sémantique commune et de générer un échange plus dynamique et efficace des informations entre les modèles applicatifs. Ainsi, elle entend lever les limites des approches unificatrices, tout en conservant une approche méta-modèle, absente de la vision fédérative. Nous proposons pour cela une solution sur trois niveaux de modélisation, représentés sur la figure 3.1. Cette division en trois niveaux existe déjà dans le domaine des ontologies, et a été formalisée par [Ishak, 2010] qui sépare les ontologies selon trois niveaux d'application :

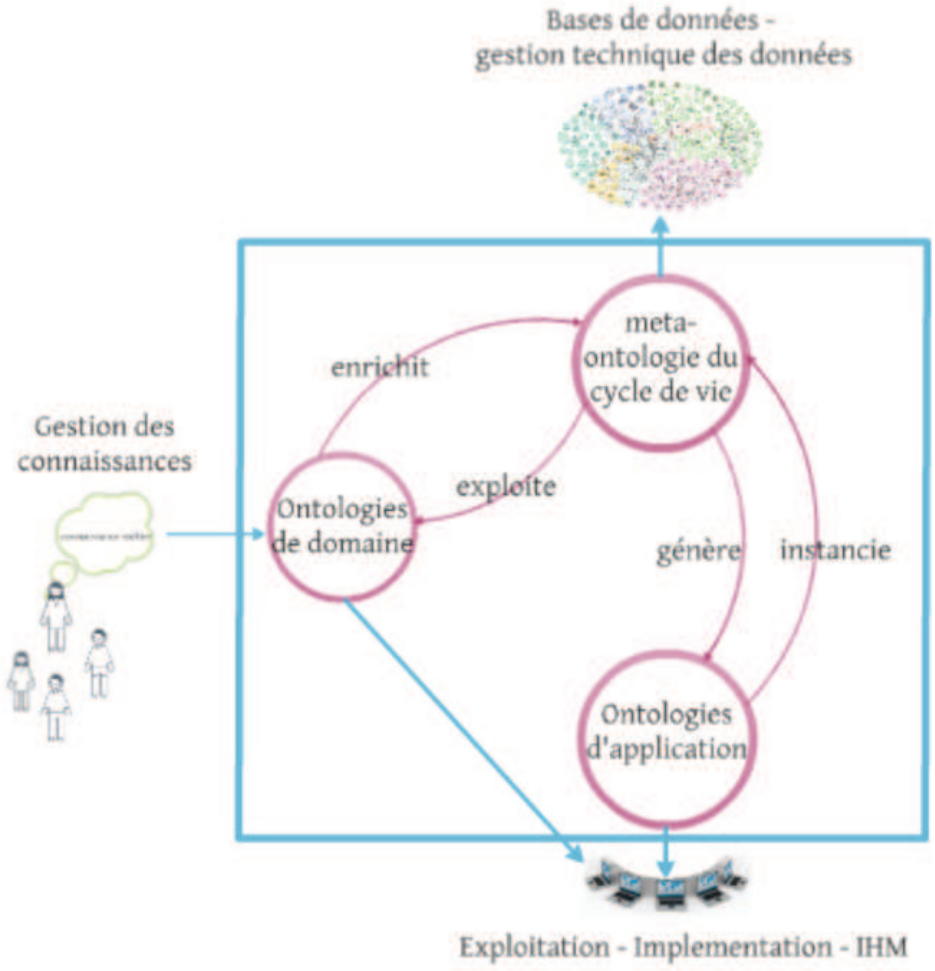


FIGURE 3.1 – Modélisation des données du cycle de vie dans une approche ontologique

- les *méta*-ontologies sont les ontologies de plus haut niveau. Elles sont utilisées pour définir le méta-domaine, c'est-à-dire l'ensemble des concepts partagés par l'ensemble des applications du domaine ;
- les ontologies de *tâche* ou de *domaine* sont des ontologies intermédiaires. Elles permettent de spécifier une tâche donnée ou un périmètre métier limité ;
- les ontologies *d'application* sont les ontologies de plus bas niveau : elles sont dédiées à un groupe d'utilisateurs, partageant un même point de vue.

Ces trois niveaux, bien qu'ils soient liés à l'emploi d'ontologies, sont bien des niveaux de modélisation. Bien souvent, ces niveaux sont simplement hiérarchiques : une méta-ontologie permet de définir plusieurs ontologies de domaine, qui seront ensuite instanciées à travers diverses ontologies d'application.

La proposition de cette étude est fondamentalement différente car elle associe des rôles sémantiques différents à ce niveaux de modélisation, en lieu et place d'un unique niveau de spécialisation. Les rôles sémantiques envisagés sont alors les suivants :

- le niveau « cycle de vie », qui correspond au niveau d'abstraction le plus élevé, permet de représenter et d'unifier toutes les informations nécessaires pendant l'ensemble du cycle de vie, et de les partager entre l'ensemble des acteurs. Les informations communes y sont donc représentées suivant un méta-modèle commun, permettant ainsi d'unifier leur représentation et exploitation. On associera alors les termes « ontologie du cycle de vie », « méta-modèle » et « modèle unifié » ;
- le niveau de domaine n'a pas pour vocation d'être un niveau de spécialisation intermédiaire, mais est un niveau de modélisation dédié à l'expert métier et cohérent avec le niveau « cycle de vie », qui permette d'intégrer ou de réutiliser de manière intelligente, et sur un domaine limité, les informations partagées ;
- enfin les modèles d'application sont instanciés par les applications métier. Ils portent en eux les informations du méta-modèle, et permettent ainsi de garantir la consistance et l'interopérabilité des données, et d'exécuter les règles définies au niveau supérieur. Il s'agit du niveau de mise en œuvre des informations qui étaient représentées dans le modèle « cycle de vie ».

Ainsi, ce qui est particulièrement novateur dans cette approche de modélisation, est qu'il n'y a pas de lien direct entre les modèles de domaine et les modèles applicatifs. Ils ne peuvent en effet dialoguer qu'au travers du modèle partagé, qui unifie la représentation de leurs informations respectives. Il garantit ainsi leur mutuelle compréhension sémantique. De plus, la captation et l'exploitation des informations métier se fait via des modèles de domaine spécifiques, et non directement via le modèle partagé. Cela permet de conserver le point de vue générique au modèle partagé, ainsi que sa simplicité de mise en œuvre : il devient un modèle de représentation des informations, tandis que leur exploitation et capitalisation sont reportées à un niveau plus spécifique, qui permet le raisonnement, et sert de filtre à l'utilisateur métier.

Ce qui n'est pas dans le périmètre de l'approche proposée

Il y a certains domaines dont le cadre de modélisation peut dépendre, ou sur lesquels il peut avoir une influence, mais qui sont tout de même hors périmètre, et qu'il convient, pour plus de clarté, de rappeler ici.

Tout d'abord, la gestion des connaissances, en ce qu'elle est un processus permettant d'extraire et de formaliser la connaissance métier tacite dans une forme explicite, n'est pas dans le cadre de cette recherche. En effet, l'enjeu est bien d'intégrer plus de connaissance,

ou du moins d'information source de connaissance dans les modèles, mais le présent travail se centre sur la définition de modèles de données à ces fins. En aucun cas et en aucun instant, les problématiques de processus et d'exploitation liées à la gestion des connaissances ne sont prétendues être traitées.

De même, la gestion des bases de données et des requêtes sur ces bases, tout comme l'actualisation et la synchronisation technique des données ne sont pas considérées. La présente recherche entend répondre à la question : quel modèle pour le partage des informations unifiées ?, et non envisager l'architecture et les requêtes nécessaires à la vie des informations du modèle.

Enfin, l'implémentation des modèles, qu'ils soient de domaine ou applicatifs, ainsi que les processus d'exploitation des données par l'utilisateur qui requièrent un modèle de type « processus » et des interfaces homme-machine ne sont pas débattus dans la proposition du cadre de modélisation. Les sous-sections qui suivent visent donc à spécifier les trois niveaux de modélisation proposés sur la figure 3.1.

3.1.2 Partager une sémantique commune via un modèle unifié pivot

Afin de fédérer les informations des différents métiers ou étapes du cycle de vie, il faut modéliser les informations selon un méta-modèle commun. L'approche méta-modèle est une approche de modélisation qui permet d'unifier les représentations des vues applicatives et la mutuelle compréhension entre ces différentes représentations métier. Alors, le modèle unifié, appelé « ontologie du cycle de vie » devient le modèle pivot pour l'échange des informations. Une ontologie du cycle de vie a déjà été proposée par [Chen et al., 2009]. Selon ces travaux, l'ontologie du cycle de vie a l'avantage de permettre « un véritable partage sémantique », c'est pourquoi elle est à la base du modèle d'organisation proposé pour la collaboration inter-entreprise : « *une entreprise dominante établit les exigences du cycle de vie du produit, qu'elle transforme en une ontologie du cycle de vie, qui sert de base à la modélisation* ». Cependant, [Chen et al., 2009] ne proposent qu'un modèle de collaboration, et le contenu sémantique de l'ontologie du cycle de vie n'est pas abordé dans l'étude.

L'ontologie du cycle de vie doit pouvoir contenir l'ensemble des informations liées au produit pendant la totalité du cycle de vie de ce produit. Elle doit également être en mesure de modéliser les interactions entre ces données, qui sont nécessaires à la définition des ontologies de domaine et d'application. Ainsi, les exigences sous-jacentes à la définition de l'ontologie du cycle de vie sont les suivantes :

- le méta-modèle de l'ontologie du cycle de vie doit être générique : pas spécifique à une phase ou un métier ;
- le méta-modèle de l'ontologie du cycle de vie doit être porteur de sens pour tous les métiers : l'utilisation d'une sémantique orientée métier est préconisée.

Le contenu de l'ontologie du cycle de vie

L'objectif du modèle partagé (l'ontologie du cycle de vie) est de faire converger sémantiquement les différentes vues métiers. La difficulté est de savoir comment spécifier cette ontologie. Le contenu complet de l'ontologie du cycle de vie dépend en effet en grande partie du périmètre de l'entreprise et du type de produits considéré. Cependant, une base commune peut être trouvée, indépendamment des produits et des processus particuliers. Pour y parvenir, nous proposons de suivre une approche nucléaire, c'est-à-dire une approche fondée sur la définition de concepts racine, ou *core concepts* en anglais. Cette approche est préconisée par [Assmann et al., 2006] pour qui « si une ontologie de haut niveau et standardisée existait, toutes les ontologies de domaines pourraient se référer aux mêmes concepts et au même vocabulaire ». Alors, définir des concepts racine et génériques est un moyen d'améliorer la sémantique du modèle et la cohérence des ontologies de domaine en se fondant sur le même noyau sémantique. Par la suite, la spécialisation de ces concepts racine permet de décrire l'ensemble des informations requises dans le modèle partagé.

Ce type d'approche a déjà été utilisé par [Brandt et al., 2008], [Lee et al., 2010], et [Tao et al., 2011]. Cependant, aucun de ces travaux n'a un périmètre qui concerne l'ensemble du cycle de vie :

- [Tao et al., 2011] définissent en effet une meta-ontologie d'éléments cliniques, fondée sur 5 concepts disjoints : *component*, *statement*, *panel*, *attribution* et *modifier* ;
- [Brandt et al., 2008] proposent un noyau ontologique pour la gestion de la connaissance fondé sur 4 concepts : *product*, *description*, *storage*, *process*. Cette ontologie ne s'adresse cependant qu'au domaine de la chimie industrielle, et au début du cycle de vie ;
- [Lee et al., 2010] proposent un méta-modèle pour la conception de produits, qui se fonde sur 5 classes fondamentales : *attribute*, *behavior*, *entity*, *property* et *objectrelationship*.

Contrairement à ces travaux, nous considérons l'ensemble du cycle de vie du produit. Les exigences à la définition de ces concepts sont les suivantes :

- les concepts racines doivent être sémantiquement disjoints. Cela signifie qu'il doit être possible de spécialiser les concepts racines sans ambiguïté d'appartenance : une classe spécialisée ne saurait être sous-classe de deux concepts racine différents ;
- La sémantique de l'ensemble des concepts racine doit être exhaustive. Ils doivent permettre de représenter l'ensemble des informations requises pour la modélisation des produits ;
- les concepts racine doivent être suffisamment génériques pour parler aussi bien aux ontologies de domaine qu'aux applications métier. Ils doivent donc s'affranchir du domaine d'application [Lee et al., 2009], mais être tout de même porteur de sens, même aux niveaux applicatifs les plus bas [Fiorentini et al., 2010] ;
- enfin les concepts racines doivent être porteurs de sens pour l'ensemble des acteurs concernés [Lee et al., 2010].

Suivant ces exigences, cinq concepts racine (que nous définissons plus en détail ci-après) ont été envisagés : le produit, le processus, les ressources, les règles et le métier.

Le concept de produit

Selon [Fiorentini et al., 2010], il est difficile de modéliser un produit. L'une des raisons de cette difficulté est que le produit embarque en lui-même différentes notions, dont il paraît nécessaire de tenir compte, comme : *la fonction*, *le comportement*, *la structure*, *la géométrie*, *le matériau*, *les caractéristiques*, et *les tolérances géométriques*. En réalité, ces différentes notions ne sont pas toutes intrinsèques au produit, bien qu'elles y soient fortement liées. Pour construire la facette produit du modèle, nous proposons une vue minimaliste de ce concept : elle ne contient que les éléments propres au produit, comme les pièces constitutives, habituellement appelées « composants » et les caractéristiques indispensables à leur définition (matériau, couleur, géométrie, identifiant, etc.), habituellement appelées « attributs ». La facette produit décrit par exemple également les relations entre ces pièces et les autres pièces.

Pour décider de ce qui est ou non intrinsèque au produit, il faut se poser la question de l'impact de l'élément qui est mis en question. Par exemple, la machine d'usinage sur laquelle est réalisée une pièce du produit est bien sûr très importante, mais il est possible de la remplacer, ou d'en modifier les paramètres de réglage sans transformer le produit, autrement dit, de manière transparente. En revanche, si l'on change le matériau ou la couleur d'une pièce, le produit n'est plus intrinsèquement le même.

Le concept de processus

Le concept de processus est celui qui intègre la variable temps. Comme l'a montré [Matsokis, 2010] lors de travaux sur le milieu de vie (MOL), il est très difficile de modéliser dans un système d'information la notion d'écoulement du temps. Pourtant, dans le modèle Time-Product, Process, Organization, Agent (T-PPOA) proposé par [Kiritsis, 2011], le temps est vu comme l'élément pivot des modèles PLM. C'est une des raisons pour lesquelles nous affirmons que les processus doivent être modélisés séparément des autres éléments du modèle. Une définition largement répandue d'un processus est une séquence d'activités, de tâches ou d'évènements, qui consomme ou produit des éléments (pièces physiques, données numériques, documents, etc.). Il peut bien sur y avoir coexistence de différents processus autour d'un même produit, par exemple, le processus de conception, le processus de fabrication, les processus de maintenance, le cycle de vie du produit, etc.

Le concept de ressource

Une ressource est ce qui est *utilisé* au cours d'un processus, afin de réaliser quelque chose. Une ressource est quelque chose de factuel, de concret, car à disposition des utilisateurs. Par exemple, un document formalisant une connaissance métier dans une base de données est une ressource, tandis les compétences d'un expert métier n'en sont pas : si l'expert s'en va, la compétence disparaît avec lui. L'expert en lui-même est en revanche une ressource de la tâche qu'il réalise. Le but de la facette ressource est de modéliser l'organisation autour du produit, au service des processus, et qui est nécessaire à la réalisation des tâches et livrables. Il peut donc y avoir différents types de ressources : les ressources humaines, les bases de documents, les outils, les machines, etc. L'une des spécificités de la ressource, c'est qu'elle est utilisée par le processus sans être transformée (nous avons donc bien une vision SADT de la ressource). Elle peut également être modifiée sans impacter directement le processus ou le produit (mais potentiellement l'organisation et sa capacité à réaliser le processus voulu).

Jusque-là, la définition des concepts racine ne diffère pas du modèle Produit-Process-Ressource (PPR), largement diffusé dans le domaine de l'ingénierie. Seulement, ce modèle est limité pour représenter l'ensemble des informations nécessaires. Beaucoup de travaux tentent par exemple d'y adjoindre la notion de comportement ou encore de fonction (SysML, [Barbau et al., 2012, Catalano et al., 2009, Cai et al., 2012]) sans pour autant parvenir à une description exhaustive des types d'information contenus dans les modèles industriels. Nous proposons, pour atteindre cette exhaustivité, de rajouter deux concepts définis ci-après : le concept de règle, et le concept métier.

Le concept de règle

Selon le Business Rule Manifesto[1] écrit par le Business Rules Group (groupe interdisciplinaire ayant travaillé sur les règles métier et aboutit, entre autre, à la définition du langage Semantic Business Vocabulary and Business Rule (SBVR)), les règles métier doivent être modélisées à part du modèle produit. Cette recommandation, qui peut paraître simple, est très peu appliquée en réalité. Par exemple, les modèles de conception modélisent souvent les tolérances géométriques comme des caractéristiques des pièces, alors qu'il s'agit en réalité des règles fonctionnelles liées au produit, exprimée dans un langage normalisé, et qui devraient être modélisées comme telles. Le but de cette modélisation séparée des règles est double : le premier objectif est que les règles métier puissent s'adresser à l'utilisateur métier, grâce à une modélisation spécifique et différente de celle du produit ; le second objectif est d'assurer la durabilité des règles ainsi que leur traçabilité : si les éléments du produit changent, les règles ne sont impactées que de manière indirecte, et un même ensemble de règles peut éventuellement être redéployé sur un autre produit. Puisque les règles contiennent l'expression d'une expertise métier, les modéliser de manière indépendante peut participer à l'amélioration de la gestion des connaissances métier. Dans la facette règle doivent alors être représentés tous les types de règles, qu'ils soient appelés directives, contraintes, tolérances, exigences, ou encore procédures de contrôle.

Le concept métier

L'existence du concept métier est due au besoin de spécifier, même au niveau de l'ontologie du cycle de vie, des éléments dépendants de domaines d'utilisation particuliers. Par exemple, dans beaucoup de modèles de conception, les fonctionnalités sont modélisées comme des caractéristiques des parties du produit. Seulement, ces fonctionnalités ne sont pas génériques : elles dépendent d'un point de vue métier particulier, et ne devraient donc pas être intégrées à la description du produit. De plus, il est tentant avec les ontologies, puisqu'elles permettent de gérer des données non-canoniques, d'établir des classifications complexes des éléments produit. Cependant, une classification est forcément l'expression d'un point de vue spécifique : il est donc préférable de chercher à limiter ces classifications sur les éléments de l'ontologie du cycle de vie. Nous proposons alors, comme pour les règles, que les éléments métier (comme les critères de classification) soient modélisés de manière indépendante. Alors, les éléments génériques des autres facettes sont *colorés* par les éléments définis dans la facette métier, à l'aide de relations sémantiques particulières, à la façon des *tags sémantiques* proposés par [Paviot, 2010]. Ces tags sont

1. http ://www.businessrulesgroup.org/brmanifesto.htm

déjà utilisés dans de nombreuses applications de gestion de données (on peut citer en exemple le module bibliographique Zotero de firefox, qui propose de tagger les références bibliographiques par mot clef, afin d'éviter les taxonomies complexes de références). Des exemples de spécifications métier peuvent être : les éléments environnementaux influents, les compétences des ressources humaines, les fonctionnalités du produit, le vocabulaire métier, etc.

Liens sémantiques entre les concepts

Une fois les concepts racines définis, se pose une question fondamentale : quelles sont les relations sémantiques entre ces concepts ? L'instanciation des relations entre les concepts est l'expression de liens spécifiques, qui, de fait, peuvent difficilement se généraliser. Par conséquent, la figure 3.2 définit uniquement les relations entre concepts. Ces relations sont orientées, et exprimées à l'aide de verbes d'action :

- *contraint* est le verbe associé à l'action des règles sur l'ensemble des autres facettes ;
- *spécifie*, *catégorise*, *classifie* et *exprime* sont les verbes associés aux actions de la facette métier. Comme nous l'avons vu précédemment, la facette métier sert à « tagger » les autres facettes, pour colorer leurs éléments ;
- *réalise* et *transforme* sont les verbes qui expriment l'action du processus sur les éléments du produit. En effet, si le produit évolue au fur et à mesure de son cycle de vie, c'est sous l'action des processus métier, qui, eux-mêmes, *utilise* et *produisent* les ressources à leur disposition.

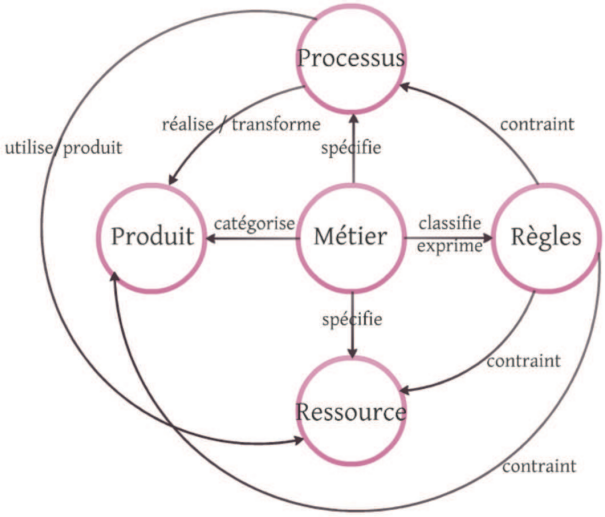


Figure 3.2 – Les liens sémantiques entre les concepts racines

L'évaluation des concepts racines de l'ontologie du cycle de vie

Pour évaluer les concepts racines définis, nous avons comparé notre approche avec les modèles industriels de la littérature. Le tableau 3.1 classifie en fonction des cinq concepts racines les concepts de plus haut niveau des modèles suivants :

- [Kiritsis, 2011] et [Matsokis and Kiritsis, 2011] proposent une ontologie pour le cycle de vie du produit, dans le cadre du projet PROMISE. Il s'agit de l'ontologie du modèle SOM ;
- [Catalano et al., 2009], au travers du projet AIM@SHAPE proposent la Product Design Ontology (PDO), qui décrit l'assemblage du produits, les surfaces et les tolérances ;
- [Kiritsis et al., 2012] proposent la Linked Design Ontology (LDO), ontologie orientée production ;
- [Fiorentini et al., 2007] proposent une ontologie pour l'Open Assembly Model (OAM), qui est une extension du Core Product Model (CPM). Il s'agit de modèles de conception ;
- Concernant les activités logistiques, [Vegetti et al., 2011] et [Gimenez et al., 2008] proposent l'ontologie PRONTO ;
- Dans le même domaine, [Lu et al., 2010] travaillent sur l'ontologie ONTO-PDM, définie auparavant par [Tursi et al., 2009], et étend les concepts de processus et d'organisation avec une ontologie basée sur le modèle SCOR (modèle de la supply chain).

Il est remarquable dans cette comparaison qu'il est possible de classer les éléments de chaque modèle en fonction des cinq concepts racines sans ambiguïté sémantique. L'examen de chaque facette permet d'affiner cette conclusion :

- concernant la facette produit, on remarque que les modèles de la littérature ont des approches assez similaires : ils décomposent le produit en un ensemble de composants ou de systèmes, qui possèdent des attributs. Ce résultat est probablement en partie dû au fait que les modèles sélectionnés représentent en majorité des produits manufacturiers. Cependant, quelques modèles, comme SOM, proposent plusieurs points de vue du même produit : *as_designed* ou *as_manufactured* ;
- le processus, dans la majorité des contributions, est une séquence d'activités ou de tâches, qui consomment et produisent des éléments (articles, données ou encore connaissance). Ces activités peuvent être séparées et séquencées par des jalons. Quelquefois, le processus prend la forme d'un flux ;

Les notions de produit et de processus présentent donc des représentations assez bien établies et constantes. En revanche, les trois autres facettes (ressource, règle et métier) ont des expressions très diverses, car liées au contexte choisi :

Matsokis and [Kiritsis, 2011] *PROMISE-SOM* **facette règle** valid field datatype, access rights, condition	**facette ressource** resource, document **facette métier** property	**facette processus** activity, event, lifecycle phase **facette produit** physical product, as_designed
[Fiorentini et al., 2007] *CPM/OAM* **facette produit** associations, geometry, form material, port	**facette métier** properties, function **facette ressource** documentation	**facette règle** tolerance requirement, constraint, **facette process** flow
[Vegetti et al., 2011] *Product Ontology (PRONTO)* **facette produit** product, variant set, structure	**facette processus** change **facette ressource** -	**facette règle** restriction **facette métier** family
[Lu et al., 2010] *ONTO-PDM* **facette produit** product, material	**facette processus** process **facette ressource** person, equipment	**facette règle** - **facette métier** organization
[Catalano et al., 2009] *PDO* **facette produit** PD Model	**facette processus** task **facette ressource** algorithm, software	**facette règle** condition type **facette métier** shape role, shape type
[Kiritsis et al., 2012] *Linked Design Ontology (LDO)* **facette produit** product, component, subsystem	**process facet** process, LCP, task, event **facette ressource** resource, agent	**facette règle** - **facette métier** datatype, eventtype

TABLE 3.1: Comparaison des concepts racines à la littérature

- dans la littérature, sauf chez [Yang et al., 2008], les ressources ne sont pas modélisées comme un ensemble à part entière, mais par quelques concepts spécifiques au modèle. Cependant, la catégorisation des ressources est assez constante. Elle se fait par rapport à leur nature : humaines, documentaires, outils et équipement, spatiales, etc. ;
- de la même manière, les concepts métier ne sont jamais modélisés explicitement. Cependant, les notions de *fonction*, *comportement* ou *propriétés* sont récurrentes. De plus, il existe des catégorisations par typage des rôles. Mais le plus souvent, les aspects métier s'expriment implicitement au sein des autres concepts, au travers de classification ou d'attributs. Dans les modèles de la littérature, la facette métier existe donc implicitement, mais n'est pas encore explicitement définie, ce qui conduit

à des divergences de représentation et quelquefois un manque de clarté du modèle.
– enfin, dans la littérature les règles ne sont pas modélisées explicitement. Elles s'expriment directement dans la structure produit. Cela empêche une gestion séparées et une traçabilité des règles. Par exemple, pour [Giovannini et al., 2012] les règles sur le modèle sont exprimées en SWRL, mais ne sont pas identifiées comme « objets » du modèle, ne permettant donc pas leur traçabilité ni leur classification. Dans certains cas, les règles sont modélisées, sous différents termes, dont les définitions ne sont pas claires : droits, contraintes, conditions, ou encore restrictions. Il arrive que ces différents termes soient utilisés pour les mêmes règles.

3.1.3 Capter et réutiliser l'information par les ontologies de domaine

Pour capter la connaissance auprès des métiers, et permettre d'intégrer les informations dans le modèle partagé, notre approche propose de construire des ontologies de domaine adaptées au métier, et « intelligentes ». Ces ontologies de domaine ont alors un rôle singulier : elles ne sont plus, comme c'est souvent le cas dans la littérature, des intermédiaires de spécialisation du modèle pivot à destination des applications métier, mais des portes d'entrée de la connaissance métier vers le modèle unifié et le moyen de réutiliser l'information contenue dans le modèle pivot.

Ces ontologies étant dédiées à des tâches particulières, leur définition peut être très variable, tout comme l'expressivité ou l'inférence qu'elles requièrent. On voudra, par exemple, établir un dictionnaire de termes métier pour l'annotation sémantique ou encore inférer des correspondances entre éléments de différents domaines. La conceptualisation de l'ontologie de domaine doit donc se faire à partir d'une analyse fine du contexte, et peut fortement varier d'un besoin à l'autre.

L'inférence à ce niveau a pour vocation à la fois d'alléger la charge d'instanciation des utilisateurs métier en automatisant certaines tâches, mais aussi d'exploiter au maximum l'information modélisation en proposant des déductions logiques et génériques sur ces informations. Enfin, et surtout, l'inférence permet de garantir la cohérence des informations qui sont exportées vers l'ontologie du cycle de vie.

Cependant, la spécification de ces ontologies de domaine est généralement tirée par deux besoins a priori antagonistes :
– elles doivent être conformes au modèle pivot, c'est-à-dire qu'elles doivent pouvoir importer et exporter des informations avec ce modèle pivot ;
– elles doivent également être orientées métier, pour permettre d'exprimer au mieux le besoin et capter efficacement l'information source de connaissance.

Nous envisageons la définition de ces ontologies de domaine dans la section 4, à travers le cas de l'expression des règles métier.

3.1.4 La définition des ontologies d'application

La question de l'interopérabilité des systèmes applicatifs

Le passage du méta-modèle aux modèles applicatifs pose la question de l'interopérabilité entre les applications et le modèle unifié, qui permet l'échange des informations. L'interopérabilité sémantique à ce niveau peut se réaliser de différentes manières. Nous nous plaçons dans le cas d'une interopérabilité directe et *a priori*, c'est-à-dire que La définition des modèles applicatifs se fait du haut vers le bas : du modèle partagé vers les ontologies d'application. L'application, qui intègre alors cette ontologie, peut l'instancier et ainsi « produire » un ensemble d'occurrences conformes au modèle unifié. Cette solution consiste à forcer les applications à se conformer au modèle de l'ontologie d'application. Même si le modèle complet de d'application est plus large que l'ontologie proposée, le but est qu'il en intègre les éléments et puisse donc l'instancier de manière bijective et directe.

Ce cas est idéal, mais peu fréquent en réalité. La plupart des outils applicatifs utilisés dans l'industrie sont commerciaux, et possèdent leur propre modèle propriétaire, défini a priori et indépendamment du modèle métier de l'entreprise. On se place alors dans le cas d'une interopérabilité *a posteriori*, qu'elle soit directe ou indirecte. Cette interopérabilité est complexe à réaliser, en raison de la distance entre le modèle unifié et le modèle applicatif (propriétaire ou standard), distance qui est de deux ordres :

- une distance syntaxique, qui doit être comblée par une transformation de modèles sous forme de mappings ;
- une distance sémantique, qui pose problème car elle est un frein à la bijectivité voire à l'établissement des mappings. Imaginons par exemple que le modèle applicatif propriétaire ne définit qu'un seul type de robinet, appelé *robinet*, alors que l'ontologie d'application définit deux types de robinets, les robinets à deux voies, appelés *Rob2V* et les robinets à trois voies, appelés *Rob3V*. Alors, il est possible à l'outil applicatif d'importer les données de l'ontologie : les *Rob2V* et *Rob3V* deviendront des instances de la classe *robinet*. Cependant, au moment de transmettre à nouveaux les données de l'outil vers l'ontologie d'application, il sera impossible de distinguer les deux types de robinets : l'échange n'est donc pas bijectif.

Dans cette thèse, nous nous affranchissons donc de cette complexité en posant l'hypothèse forte d'une interopérabilité *a priori*, où l'ontologie d'application peut être définie

conforme au modèle unifié, et à laquelle l'outil applicatif se conforme lui-même. Le bien fondé de cette hypothèse repose sur le fait que nous avons pour objectif d'évaluer les capacités et l'apport des ontologies pour la modélisation des informations. Pour ce faire, il est nécessaire dans un premier temps de s'affranchir des difficultés liées aux distances syntaxiques et sémantiques entre outils commerciaux. Il faudra néanmoins s'y reporter par la suite.

La formalisation des ontologies d'application

L'information qui est « exprimée » dans le modèle pivot doit donc être mise en œuvre dans un modèle d'application, et être instanciée. Nous entendons par « mise en œuvre » le fait que l'ensemble des concepts et des relations, dont la sémantique est exprimée dans l'ontologie du cycle de vie, doivent maintenant incarner cette sémantique en leur sein, dans le modèle applicatif. Imaginons par exemple que l'ontologie du cycle de vie modélise les relations entre composants d'une ligne de tuyauterie comme suit : les éléments dans la ligne de tuyauterie sont des instances d'une classe **composant** et les relations entre composants sont les instances d'une classe **relation**. S'il s'avère que l'instanciation du modèle unifié établit qu'un *robinet* est relié par la relation *Rob2Res* à un *reservoir* et que cette relation a une cardinalité 1, alors au niveau applicatif, le robinet et le réservoir sont des éléments instanciables (par des occurrences), qu'une propriété de type « functional » relie. Ainsi, la sémantique métier modélisée dans l'ontologie du cycle de vie à été *mise en œuvre* dans l'ontologie d'application.

Il y a donc une transformation de modèle, depuis l'ontologie du cycle de vie vers les modèles applicatifs. Cette transformation est de deux ordres.

- un changement de niveau sémantique : les instances de l'ontologie du cycle de vie, reliées entre elles par les éléments du méta-modèle, doivent maintenant devenir des éléments instanciables, afin de représenter « la réalité » au travers des occurrences ;
- une restriction du champ d'information : si l'ontologie du cycle de vie contient l'ensemble des informations unifiées, chaque modèle applicatif est une restriction de cet ensemble, dédiée à l'application souhaitée.

L'interopérabilité entre les modèles applicatifs, et surtout leurs occurrences -qui sont l'aboutissement finalement de toute cette modélisation amont- est alors réalisée a priori, par le fait que l'ensemble des modèles applicatifs dérivent du même modèle unifié, et que, soit ils partagent une même modélisation des informations qu'ils doivent échanger (le reste de la modélisation pouvant différer entre les deux modèles), soit les mappings entre les éléments des deux modèles ont été établis a priori, au niveau de l'ontologie du cycle de vie.

Conclusion de la section 3.1

Le cadre générique de modélisation envisagé dans cette étude est fait de trois niveaux : un niveau générique, un niveau de domaine et un niveau applicatif. Ces différents niveaux ont des périmètres bien délimités, en fonction des rôles sémantiques qu'ils remplissent. L'ontologie du cycle de vie permet le partage de sémantique et l'échange d'information ; les ontologies de domaine permettent la capitalisation et l'exploitation métier des informations ; et les ontologies d'application mettent en œuvre la sémantique sur les éléments concrets du produit. Dans la section suivante, nous proposons une méthode générique qui permet de mettre en place ce cadre de modélisation avec des modèles ontologiques. Par la suite, l'application à un cas d'étude industriel dans le domaine de l'ingénierie nucléaire permet d'illustrer et de valider l'approche proposée.

3.2 Une méthode générique pour le cadre de modélisation

Cette section formalise la démarche à suivre pour mettre en place le cadre de modélisation proposé. Cette formalisation prend le nom de « méthode », du fait qu'elle est une démarche structurée et reproductible mais non une « méthodologie » complète au sens de [Zellner, 2011]. Le but de cette méthode générique est de guider un modélisateur qui souhaiterait mettre en place le cadre générique proposé sur un domaine particulier, étape après étape. Elle est représentée sur la figure 3.3 et s'appuie sur les niveaux d'abstraction définis dans le projet Neon [Suárez-Figueroa, 2010], qui sont :

- la spécification, qui est le plus haut niveau d'abstraction. A ce niveau sont définis le but, le périmètre, ou encore les exigences de sémantique ;
- la conceptualisation, qui est le niveau où les caractéristiques du modèle, tels que la structure ou les éléments constitutifs sont définis. L'information que le modèle doit contenir est représentée suivant des primitives (concepts, relations, etc.), sans pour autant intégrer des aspects spécifiques à tel ou tel langage de formalisation ;
- la formalisation, qui est le niveau où l'information est représentée dans un langage formel spécifique ;
- l'implémentation, qui est le niveau où la formalisation est mise en œuvre dans des outils, pour permettre l'édition et l'exploitation des données.

Puisque cette méthode vise à appliquer un cadre de modélisation, l'implémentation peut être considérée comme hors périmètre. Cependant, les contraintes d'implémentation influencent la formalisation des modèles ainsi que la conceptualisation du besoin sémantique. Cette étape peut donc entraîner des boucles itératives dans la méthode proposée, c'est pourquoi elle est considérée ici, bien que le cadre de modélisation en lui-même se limite aux trois niveaux d'abstraction supérieurs.

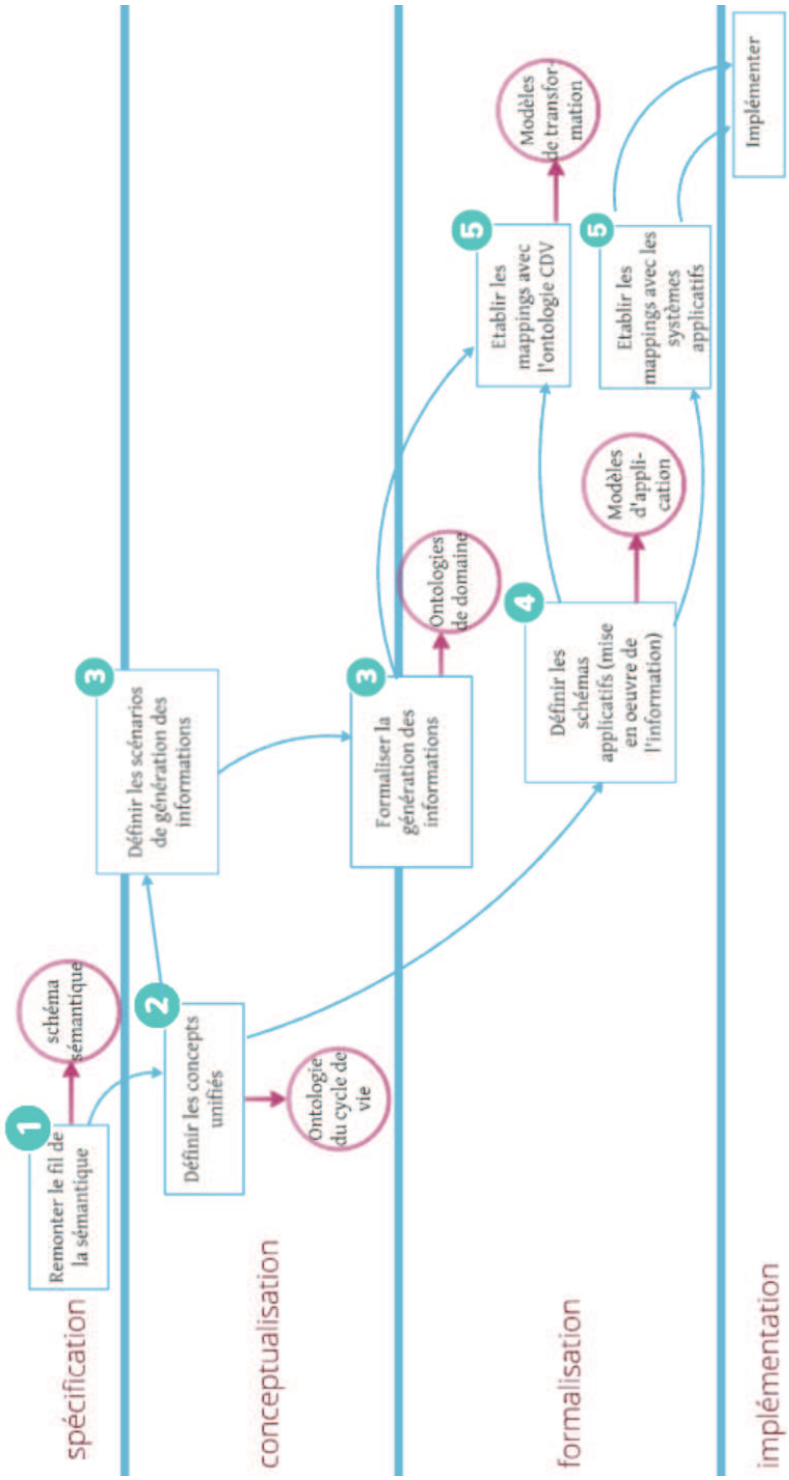


FIGURE 3.3 – La méthode générale de modélisation ontologique des informations

3.2.1 L'étape 1 : remonter le fil de la sémantique

Les ontologies sont des outils permettant d'exprimer une grande sémantique. Surtout, ce sont des moyens de lier les concepts entre eux. Ainsi, l'ontologie du cycle de vie pourrait être un ensemble de données connectées, alimenté par les différents utilisateurs, et que chacun pourrait à loisir et selon ses besoins propres exploiter et interpréter dans un modèle dédié (modèle de domaine ou d'application). Tout le problème est alors de savoir comment organiser ce grand « réseau sémantique ». Et même plus que cela, car, avant de poser la question de la conception du modèle, il est nécessaire de répondre à la question suivante : « quelle est donc cette sémantique que l'on veut exprimer ? ». Et surtout, il est très important de distinguer, lors de ce travail, quelle sémantique l'on souhaite représenter de l'exploitation que l'on souhaite en faire par la suite. En effet, il s'agit dans un premier temps de délimiter le périmètre sémantique, afin de produire l'ontologie partagée du cycle de vie, qui ne tient pas encore compte de l'utilisation a posteriori des informations.

Il s'agit aussi se demander s'il est possible de définir les concepts de façon unique et partagée. L'expérience du projet PLM-EDF, présenté plus en détail dans la section 4.1, montre qu'un même concept peut être compris de manière extrêmement différente par les différents utilisateurs métier : un robinet n'est pas le même objet pour un ingénieur de circuit fluide et un installateur. Quel sens donner dans ce cas à la donnée « robinet » et peut-on la lier de façon pertinente et non-contradictoire aux autres concepts du domaine ? De plus, à quelle fin veut-on modéliser ces informations, c'est-à-dire, quelle sémantique porte la relation qui les lie ?

Seul un tel travail de haut niveau et en amont de la modélisation peut conduire à la définition de modèles robustes et pertinents. Il faut donc s'efforcer à « remonter le fil de la sémantique » afin de comprendre ce qui doit ou non être représenté dans le modèle unifié, avant de chercher à savoir comment le représenter. De ce travail doit émerger un schéma sémantique qui décrit explicitement et de manière unique l'information qui devra être modélisée.

Cependant, la définition de ce schéma répond à une problématique particulière, à un objectif de modélisation. Le méta-modèle « cycle de vie » ne saurait se construire de manière exhaustive et directe, par la simple étude du domaine : au contraire, il se construit de manière itérative, et sous le prisme de problématiques délimitées et identifiées, qui permettent alors de réellement « creuser le besoin » de modélisation, et non simplement représenter un existant. La méthodologie proposée doit donc s'appliquer et se répéter pour chacune de ces problématiques identifiées. D'où la nécessité de définir des modèles extensibles.

3.2.2 L'étape 2 : conceptualiser et formaliser la sémantique partagée

Une fois le besoin sémantique explicité, il faut définir les concepts et les relations qui permettent d'exprimer ce besoin. Il n'est pas encore question de définir une ontologie dans un langage formel, mais de penser les éléments constitutifs du modèle et leurs relations sémantiques. Cette conceptualisation spécialise les concepts racines de l'ontologie du cycle de vie. La spécialisation relevant d'un besoin particulier est par conséquent souvent *ad hoc*. Cependant, afin d'optimiser cette tâche, il peut être pertinent de réutiliser des modèles existants ou des portions de modèles, et de les faire fusionner avec l'ontologie du cycle de vie. On entre alors dans la phase de formalisation.

La formalisation dans un langage logique doit permettre par la suite l'implémentation du modèle, car elle exprime la sémantique dans un langage compréhensible par la machine. Le modèle obtenu est une partie de l'ontologie du cycle de vie, permettant de répondre au besoin sémantique identifié préalablement. A cette étape de la formalisation, le langage OWL peut être employé, bien qu'il ne soit pas vraiment question d'inférer des informations au niveau du modèle partagé. Ainsi, le langage RDF(S) paraît suffisant : il s'agit de représenter des relations sémantiques entre concepts. RDF est même pertinent, car il permet d'établir des liens spécifiques entre données de différents niveaux d'abstraction, faisant de l'ontologie du cycle de vie est véritable « réseau » sémantique. Mais s'il est nécessaire d'étendre la sémantique de cette formalisation, il peut être judicieux d'utiliser OWL, notamment pour pouvoir restreindre les domaines d'application des propriétés.

L'une des difficultés de la formalisation des informations dans l'ontologie du cycle de vie est d'évaluer la justesse du contenu. Les méthodologies existantes traitent très peu du contenu sémantique de l'ontologie. Seule l'approche d'OntoClean permet d'évaluer la formalisation réalisée. Pour développer OntoClean, [Guarino and Welty, 2009] se sont inspirés des techniques des philosophes pour évaluer leurs ontologies (philosophiques), afin de proposer une méthode de validation du contenu d'une ontologie. OntoClean repose sur quatre meta-propriétés : Identité, Unité, Rigidité et Dépendance, auxquelles se rajoutent Permanence et Réalité. La mise en application de cette méthode est réalisée à la section 4.2.1.

3.2.3 L'étape 3 : conceptualiser et formaliser les ontologies de domaine

L'étape 2 a permis de conceptualiser et formaliser la sémantique des informations que doit contenir le modèle partagé. Cependant, ce modèle est une vue statique des informations : il représente les concepts et les relations instanciées nécessaires à la

représentation d'une information dans un état donné. La conception des ontologies de domaine passe alors par deux étapes importantes :

- la définition des scénarios de génération et d'exploitation des informations, c'est-à-dire comment approvisionner le modèle unifié, et quelles informations nouvelles vont émerger de l'exploitation des informations partagées ;
- la formalisation de ces modèles, de manière conforme à l'ontologie du cycle de vie. Cette étape peut générer des itérations et amener à revoir la formalisation du modèle partagé.

Il y a donc à nouveau une étape de spécification et de conceptualisation de la sémantique. Cependant, le point de vue est différent : il s'agit d'envisager les processus de génération et d'exploitation des informations qui vont pouvoir être représentées dans l'ontologie du cycle de vie. Ensuite, il faut formaliser cette conceptualisation, en exploitant au mieux l'expressivité et l'inférence permises par des langages ontologiques.

Ces deux étapes génèrent souvent des boucles d'itération : en effet, les possibilités ou les limites de la formalisation peuvent conduire à repenser la conceptualisation de l'ontologie de domaine. Elles peuvent également remettre en cause la conceptualisation de l'ontologie du cycle de vie. Par exemple, dans le cas d'étude sur les règles (voir section 4), dans un premier temps, il était apparu que seuls les triplets métier étaient nécessaires à l'expression des règles métier. Cependant, le travail sur la génération de ces règles, et notamment l'exigence de vérification de la cohérence des règles a permis de comprendre qu'il était également nécessaire de représenter formellement dans l'ontologie du cycle de vie les éléments constitutifs des triplets - à savoir les concepts et les relations.

Les ontologies de domaine sont souvent des ontologies de gestion de connaissance, ou du moins, d'informations source de connaissance, qui nécessitent une sémantique riche. Le langage OWL2 est le candidat idéal pour formaliser ce types de modèles : en effet, OWL2 est riche sémantiquement, afin d'exprimer la complexité du modèle, mais permet également de raisonner, pour automatiser en partie l'instanciation des informations et vérifier la cohérence des données. Cependant, OWL2 n'est pas pleinement décidable et/ou supporté par les raisonneurs actuels. Il convient de se limiter à une restriction « stable » d'OWL2, généralement appelée OWL2-DL.

3.2.4 L'étape 4 : formaliser les modèles applicatifs

Suite à la définition de l'ontologie du cycle de vie et des ontologies de domaines, l'étape 4 consiste à « mettre en œuvre » la sémantique de l'ontologie du cycle de vie pour chaque

application. A cette étape, il est nécessaire de définir une ontologie d'application, qui, sur le périmètre choisi, doit re-formaliser la sémantique du modèle partagé, afin qu'elle soit implémentable et instanciable par les applications.

La formalisation des ontologies d'application nécessite une sémantique riche, des bonnes capacités d'inférence, mais aussi d'exécuter des règles. Le meilleur compromis sur ces trois critères, parmi les langages ontologiques, est l'association OWL-DL/SWRL. En effet, OWL-DL est le meilleur compromis entre capacité d'inférence et expressivité, et SWRL est le seul langage de règles entièrement conforme à OWL [Fiorentini et al., 2010].

3.2.5 L'étape 5 : établir les mappings sémantiques entre les différents modèles

Enfin, une fois chaque niveaux de modélisation conceptualisé et formalisé, il faut établir les mapping sémantiques entre les différents modèles, ou du moins, les règles d'échange de données entre ces modèles. Ces transformations doivent être automatisées et bi-directionnelles. Leur spécification doit être modélisée dans un langage formel afin d'en permettre la maintenance et l'évolution.

L'établissement des mappings sémantique est fortement guidé par les contraintes d'implémentation et les syntaxes des langages formels choisis. A cette étape, l'implémentation peut soulever des difficultés qui remettent en cause les choix de formalisation antérieur. A nouveau donc, des boucles d'itération sont possibles.

3.2.6 Le bilan : un cadre ontologique de formalisation des informations

La méthode proposée permet ainsi d'aboutir à une conceptualisation et une formalisation ontologique de chacun des niveaux de modélisation du cadre général proposé. Selon les contraintes et les enjeux de chaque niveau de modélisation, la figure 3.4 dresse, en guise de synthèse, le bilan des échanges et des syntaxes nécessaires à la formalisation ontologique de ces modèles.

On y voit que l'ontologie du cycle de vie est un réseau sémantique RDF dont les éléments sont modélisés suivant un schéma RDFS : en effet, ce niveau de modélisation n'exploite pas l'inférence OWL, mais l'approche méta-modèle, la légèreté et la distributivité de RDF. Ensuite, les ontologies de domaine, grâce à l'expressivité d'OWL2 et au raisonnement, formalisent l'information source de connaissance des métiers qu'elles exportent sous forme de triplets RDF, conformes au schéma de l'ontologie du cycle de vie. Enfin, par une transformation sémantique, l'ontologie d'application OWL met en

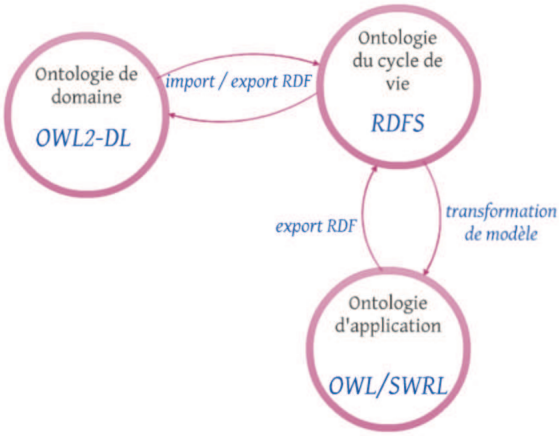


FIGURE 3.4 – Le cadre de modélisation ontologique des informations

œuvre la sémantique de l'ontologie du cycle de vie, exécute les règles (SWRL), et est par conséquent en mesure d'exporter un réseau RDF d'occurrences, réputées conforme à l'ontologie du cycle de vie, puisque produites par l'ontologie d'application.

Conclusion de la section 3.2

La méthode proposée s'attache à définir le besoin sémantique avant de prendre en compte les contraintes de formalisation et d'implémentation. Elle propose également une définition des modèles doublement récurrente : tout d'abord, la méthode doit s'appliquer dans son ensemble pour chaque problématique de modélisation, les modèles n'étant jamais définis de manière complète à chaque fois ; ensuite, il existe des boucles d'itération au sein de la méthodologie, dues au fait que les niveaux de formalisation et d'implémentation présentent des contraintes logiques ou techniques qui peuvent demander des modifications aux niveaux d'abstraction supérieurs. Ainsi, les étapes considérées ne doivent pas être vues comme un processus chronologique mais comme un ensemble de réflexions nécessaires à la définition du cadre de modélisation. Dans le chapitre suivant, nous mettons en place le cadre de modélisation, en suivant la méthode proposée, sur un cas d'application particulier : la modélisation des informations des centrales nucléaires.

CHAPITRE 4

L'APPLICATION AU CAS DES RÈGLES MÉTIER

Résumé du chapitre

Dans ce chapitre, le cadre de modélisation proposé précédemment est mis en place sur un cas industriel. Cette mise en place suit la méthode générale proposée et permet la définition de modèles ontologiques. Par conséquent, le chapitre s'articule comme suit :

La section 4.1 présente le cas d'étude industriel, issu de l'ingénierie nucléaire, et plus précisément l'expression et l'exécution de règles métier sur un modèle produit. Ainsi, la modélisation existante, fondée sur un modèle unifié des informations, est décrite, ainsi que les enjeux et les verrous liés à l'expression et l'exécution des règles.

Tout au long de la section 4.2 sont définis les différents modèles spécifiques au cas d'étude. Cette définition suis la méthode générique proposée, et abouti à la définition d'un méta-modèle « cycle de vie », d'un modèle de domaine appelé « ontologie métier » support à l'expression des règles, et à envisager l'exécution des règles dans un modèle applicatif OWL/SWRL.

Enfin, La section 4.2.5 relève les limites des ontologies que l'application industrielle a permis de poser ou de confirmer, selon le point de vue de la modélisation. Ces limites sont notamment liées à l'Open World Assumption, aux langages de règles mais aussi à des problématiques d'implémentation.

4.1 Le cas d'étude industriel

4.1.1 L'ingénierie de la prochaine génération de centrales nucléaires

Nous présentons ici le contexte général du cas d'étude traité par la suite, et qui va permettre d'illustrer chacun des trois niveaux du cadre de modélisation. Il s'agit de la modélisation des informations des centrales nucléaires. Dans cette étude, nous travaillons en collaboration avec Électricité de France (EDF), entreprise française qui conçoit, exploite, maintient et démantèle les centrales nucléaires. Pour ce qui concerne la modélisation des informations, le cas des centrales nucléaires est singulièrement complexe et varié, pour les raisons suivantes :

- la nécessité d'une préservation à long terme des données : l'unité de temps des centrales nucléaires est la décennie, avec un cycle de vie de l'ordre du siècle ;
- les effets d'échelle : une centrale nucléaire est composée d'environ un milliard de composants. Cela implique une explosion du nombres d'instances, de documents, de métiers, d'applications et d'acteurs ;
- l'importance des exigences et des contrôles de sécurité : la vie et le comportement d'une centrale nucléaire sont soumis à une grande quantité d'exigences de différents niveaux, et émanant de diverses autorités (agences nationales, internationales, etc.). A tout moment, EDF doit pouvoir garantir le respect de ces exigences, et ce même pendant la phase d'exploitation, car, contrairement à d'autres industries comme l'aéronautique ou l'automobile, EDF, en tant qu'exploitant est responsable du comportement de la centrale, bien qu'elle ne l'ait pas construite ;
- le va-et-vient des acteurs au cours du cycle de vie : si EDF intervient du début à la fin du cycle de vie de la centrale, l'entreprise passe plusieurs fois le relais à ses partenaires au cours de ce cycle. L'échange d'informations avec les partenaires est alors un défi en soi (par exemple, Bouygues construit l'enveloppe en béton, AREVA est responsable de l'installation des circuits nucléaires, etc.).

Dans ce contexte, la division de l'ingénierie nucléaire (DIN) d'EDF a compris l'enjeu que représente la maîtrise de l'information dans une approche cycle de vie et a entamé un projet PLM il y a maintenant deux ans. Les objectifs stratégiques de ce projet sont :

- d'unifier l'ensemble des informations des différents métiers au travers d'un modèle unifié commun, et ainsi, garantir l'interopérabilité des informations entre toutes les applications métier ;
- de rationaliser et de maîtriser le stockage des informations à long terme et celui des informations de maintenance ;

– d'être capable d'automatiser et de tracer les processus métier au travers d'une modélisation des exigences et des règles métier. L'ingénierie des exigences, dans ce contexte, permettra d'amorcer la démarche d'ingénierie système.

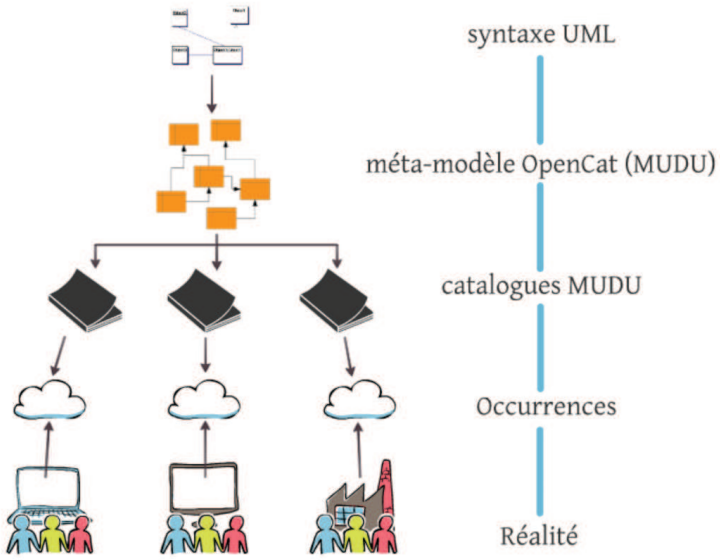


FIGURE 4.1 – Cas d'application : la modélisation des centrales nucléaires

La mise en place du PLM à EDF se fait sur la base d'une modélisation existante, représentée sur la figure 4.1.1. Au niveau applicatif de cette modélisation, les informations sont éparpillées en différentes applications métier, qui possèdent toutes un modèle de données différent, et souvent peu pérenne (à l'échelle de la vie d'une centrale). Pour éviter les limites liées à ces modèles propriétaires, EDF a développé un modèle unifié, appelé MUDU (Modèle Unifié Des Données Utilisateur). MUDU est en réalité un ensemble de « catalogues » (dédiés chacun à des applications particulières), unifiés puisqu'ils reposent tous sur le même méta-modèle de données, le méta-modèle Opencat, dont la syntaxe est de type UML. Ce méta-modèle définit un certain nombre de classes génériques (composants, caractéristiques, représentation graphique, etc.) qui permettent de constituer des dictionnaires d'instances. Ces instances sont ensuite regroupées par catalogue. Ensuite, chaque catalogue doit lui-même être « instancié » par les applications métier. Les instances de ces catalogues sont appelées occurrences. Elles font référence aux instances de MUDU, mais sont modélisées suivant un modèle d'occurrences unifié, définit dans OpenCat.

A l'heure actuelle, la solution envisagée fait face aux limites suivantes :

– l'interopérabilité entre les applications métier et les catalogues MUDU n'est pas réalisée : en effet, il faut pouvoir exprimer les fichiers applicatifs selon le modèle du catalogue pour parvenir à générer les occurrences ;

- le modèle unifié MUDU n'est pas complet : il ne contient que des éléments produit, ainsi qu'une modélisation sommaire des activités et des règles. De plus, il ne couvre pas encore l'ensemble des métiers et les correspondances entre catalogues sont incomplètes. Il est nécessaire de l'enrichir ;
- le langage d'implémentation est OTScript, un langage spécifique, peu flexible, et qui n'est plus développé ni maintenu. Une transformation de l'implémentation s'impose ;
- la modélisation actuelle ne permet pas d'exécuter les règles ni de vérifier la cohérence des données.

Pour la bonne compréhension des sections qui suivent, il est important de retenir quelques notions et termes de la modélisation actuelle, auxquels il sera souvent fait référence. Le modèle actuel, appelé **MUDU** est constitué de deux niveaux sémantiques :

- le niveau **« instance »** définit les éléments génériques de la centrale. Ils sont référencés dans un **dictionnaire** générique (qui définit tous les éléments possibles) et dans des catalogues métier, qui sont une spécification partielle et subjective du dictionnaire ;
- le niveau **« occurrence »** est le niveau d'instanciation de MUDU. Les occurrences sont en effet des instances d'instances de catalogue.

Illustration des niveaux de modélisation existant

Le méta-modèle OpenCat définit des les classes **composant** et **caractéristique**. *Regl* est une caractéristique de dictionnaire exprimant le caractère réglant ou non d'un matériel. Elle est associée dans un catalogue métier à l'instance de composant *Rob2V* qui définit tous les robinets deux voies.

Le robinet deux voies « réel » et unique `rob2V_1` spécifié sur le schéma mécanique 2D est une occurrence, liée à l'instance de MUDU *Rob2V* : c'est un type de *Rob2V*. L'occurrence de caractéristique *Regl* pour le robinet `rob2V_1` vaut `faux`.

Ainsi le niveau occurrences est bien l'instanciation des instances de MUDU, définie suivant le méta-modèle OpenCat.

4.1.2 L'expression et l'exécution des règles métier

Les règles métier : de la connaissance à la donnée

L'intégration de connaissances métier dans un système d'information est en fait un transfert d'une information intelligible par le métier vers un modèle d'information conçu

par les experts SI, dans un langage formel. Puisque l'enjeu du PLM est, comme nous l'avons vu dans la section 1.2, de modéliser une information source de connaissance, il est nécessaire de procurer des modèles complexes et riches. De cet état de fait naît alors une contradiction : les experts SI n'ont pas la possibilité de comprendre pleinement les concepts du métier, et les experts métier rencontrent des difficultés à s'approprier le SI. Cela engendre des pertes sémantiques potentielles et est un frein à la modélisation des connaissances métier. De plus, l'activité de formalisation des connaissances est souvent perçue comme chronophage et peu utile par le métier. Alors, le système mis en place doit être orienté métier et permettre une automatisation complète, ou à défaut partielle, de l'instanciation des informations afin de limiter les actions répétitives aux utilisateurs métier et les guider au mieux vers une représentation la plus homogène possible de l'information.

Ainsi, pour intégrer efficacement la connaissance métier dans les modèles d'information, il est nécessaire de :

- modéliser des informations de différents niveaux d'abstraction, puisque la connaissance naît en réalité de la convergence de différentes informations et de leur contexte ;
- proposer un modèle sémantique compréhensible par le métier, afin de faciliter leur appropriation du SI ;
- accompagner l'expression de l'information depuis la vision métier vers le SI, pour éviter la perte de flux sémantique entre les acteurs ;
- permettre une automatisation, et une vérification de la cohérence des informations, qui jouent le rôle de poka yoke (système anti-erreur) dans la formalisation des connaissances et réduit l'implication métier dans le travail de formalisation.

Un des moyens pour formaliser la connaissance métier dans le modèle d'information est de l'exprimer sous forme de règles métier, qui sont exécutées sur les instances du modèle. Selon le Business Rules Group, qui a mis fin à des divergences importantes dans les définitions de la règle métier[1], cette dernière peut se définir selon deux perspectives [Hay and Healy, 2000] :

Définition de la règle métier (Business Rules Group)

Du point de vue SI, une règle métier est « une assertion qui définit ou contraint un ou plusieurs aspects du métier. Elle a vocation à établir une structure métier, ou de contrôler ou influencer le comportement du métier ».

Du point de vue métier, une règle métier est « l'expression qu'il y a une obligation

1. notamment les définitions de [Herbst, 1997, Van Assche et al., 1988, Morabito et al., 2001, Ceri and Fraternale, 1997]. Voir [Fortineau et al., 2013a] pour une confrontation de ces définitions.

concernant une conduite, une pratique, une action ou une procédure, au sein d'une activité ou d'une sphère particulière ».

Il est très intéressant que la définition proposée par le groupe inter-disciplinaire BRG soit double : elle confirme que l'implémentation de règles métier est un processus qui fait se percuter deux points de vues : la règle est à la fois un élément du modèle d'information, et l'expression d'une connaissance métier. Alors, l'implémentation des règles métier lève la question de l'intégration de connaissances dans un système d'information. La problématique de l'étude sur les règles est par conséquent de savoir comment modéliser les règles métier, pour en permettre l'expression depuis leur forme naturelle jusqu'à leur forme formelle et leur exécution. Des exemples de types de règles métier que nous traitons dans cette étude se trouvent en annexe 7.4.

4.2 La mise en place du cadre de modélisation

4.2.1 La définition du besoin sémantique

La problématique de l'étude est de savoir comment modéliser et exécuter les règles métier, depuis leur forme naturelle (telle qu'exprimée par l'expert métier) jusqu'à leur forme formelle en vue de leur exécution. La première étape du travail est de définir le besoin sémantique nécessaire à l'expression et la validation des règles métier. Afin d'accompagner l'expression des règles métiers depuis l'expert métier jusqu'à leur forme formelle, le modèle de transformation sémantique de la figure 4.2 est envisagé. Ce modèle de transformation a été présenté plus en détail lors de la conférence MIM 2013 [Fortineau et al., 2013a]. Il s'agit d'un modèle en quatre étapes, expliquées ci-après. Chacune des étapes a un périmètre sémantique bien défini, un groupe d'utilisateurs particulier, et, comme le montre la figure 4.2 une durée de conservation propre. Ce modèle a été conçu pour répondre à trois objectifs majeurs :

- permettre la traçabilité des règles depuis les documents techniques jusqu'à l'exécution formelle ;
- assurer une transformation du besoin métier sans perte de sémantique ;
- permettre une expression homogène et une vérification de la cohérence sémantique des règles.

Pour atteindre ces trois objectifs, quatre étapes - donc trois transformations- sont nécessaires.

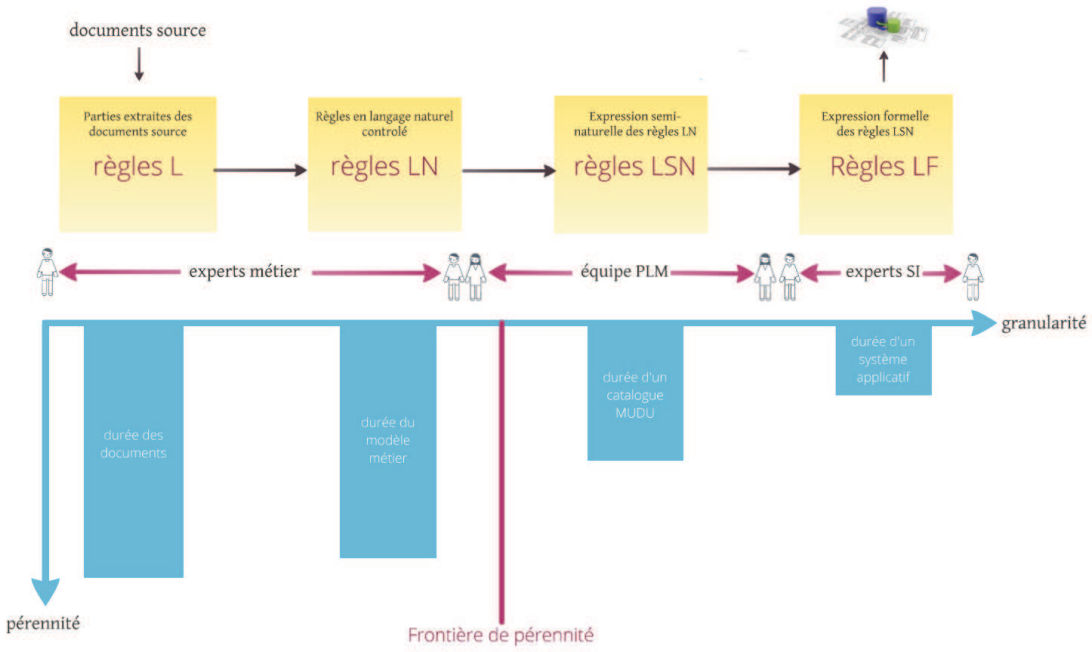


FIGURE 4.2 – Le modèle de transformation sémantique des règles métier

La transformation de L vers LN : une homogénéisation du besoin métier

Les éléments ressource du modèle, points d'entrée de la connaissance métier, sont des documents textuels divers (documents de doctrine, exigences de conception, etc), qui contiennent, entre autre, des règles métier. Ces documents ont des contenus extrêmement hétérogènes, si bien qu'il n'y a aucun standard ni sur la forme que prennent les règles, ni sur le vocabulaire employé. Au niveau métier, il faut par conséquent réaliser deux actions, que nous avons choisi de séparer dans le modèle :

- extraire des documents ressources les parties contenant des règles métier, les tracer vis-à-vis du modèle de documents et modéliser leurs interactions mutuelles : c'est l'objectif de l'étape L ;
- réécrire les règles métier suivant une syntaxe et un vocabulaire homogène, afin de produire un ensemble de règles homogènes, non-redondantes, non-ambiguës et cohérentes entre elles : c'est l'objectif de l'étape LN. Les règles sont alors écrites en langage naturel, mais contrôlé.

La transformation de LN à LSN : une conformation au SI

Les règles LN obtenues lors de la transformation sont garanties cohérentes, porteuses de sens et homogènes. Cependant, elles ne sont pas exprimées suivant la syntaxe et le point de vue du modèle d'information (dans notre cas d'application, le modèle MUDU). Elles ne sont donc pas implémentables en l'état par un expert SI. Le rôle de l'équipe PLM à cet instant est de transformer les concepts métier qui constituent l'expression des règles LN en des concepts MUDU, afin de pouvoir proposer une expression de la règle qui, d'un point de vue syntaxique, soit toujours de type « naturelle contrôlée », mais qui, d'un point de vue sémantique, utilise les concepts et les relations du modèle MUDU. Alors, cette règle LSN pourra être comprise sans ambiguité par l'expert SI, qui pourra en proposer des versions formelles exécutables. Les règles LSN portent donc sur les éléments du modèle de données partagé, « l'ontologie du cycle de vie ».

La transformation de LSN vers LF : l'écriture de la règle dans un langage formel

Une règle LSN porte sur les éléments du modèle de données partagé. Elle peut donc être implémentée au niveau des modèles applicatifs, puisque ceux-ci sont réputés cohérents vis-à-vis du modèle partagé. Alors, les règles LF générées pourront s'exécuter sur les instances des modèles d'application, et finalement, valider le besoin métier sur une représentation du produit réel. L'expression de règles en langage formel à partir de LSN est multiple, car elle dépend de plusieurs paramètres :

- le langage d'implémentation choisi pour le modèle applicatif. La règle LF devra alors être exprimée en fonction de ce langage, de sa syntaxe et de l'expressivité qu'il autorise ou non ;
- le processus de mise en œuvre de la règle. En effet, une règle n'aura pas la même expression si, par exemple, elle doit être exécutée *a priori* (l'application propose une valeur par défaut), instantanément (l'utilisateur doit entrer la valeur au moment où il crée l'instance), ou *a posteriori* (la règle est exécutée une fois le modèle instancié sur un ensemble de données).

Illustration du modèle de transformation sémantique sur une règle métier

Dans le document de doctrine sur la robinetterie, il est stipulé que :
« Les matériels non concernés par la fiche sont les suivants : réservoirs de transport destinés à être transportés, après remplissage, à un autre endroit pour y être vidés ; Bacs de rétention (voir fiche DRI B4) ; Bâches intégrées à la structure des bâtiments.

Alors : les réservoirs dont la température du fluide est supérieure à 60 degrés sont calorifugés ou entourés d'un grillage protecteur afin de protéger le personnel. »

Cet extrait contient une règle métier, c'est donc une règle L. Mais cette règle est mal formulée. En réalité, le besoin exprimé est le suivant : les réservoirs sont des composants destinés à contenir un fluide. Ce fluide a une température variable. Les réservoirs ont également des fonctions différentes : ils peuvent par exemple servir à transporter le fluide uniquement, ou bien servir de réservoir de rétention (appelés alors « bâches »). De plus, les réservoirs peuvent être localisés à différents endroits : à l'intérieur de l'enceinte, à l'extérieur de l'enceinte, ou directement dans la structure de l'enceinte. La règle métier LN, est alors la suivante :

L'ensemble des réservoirs, sauf ceux ayant pour fonction la rétention ou le transport temporaire du fluide, et ceux directement localisés dans la structure béton, doivent posséder un dispositif de protection si le fluide qu'ils contiennent est de température supérieure à 60 degrés. Ce dispositif de rétention est de type calorifuge ou grillage. L'objectif de cette règle est de protéger le personnel en contact avec le réservoir de potentielles brûlures.

Pour cette règle LN, la règle LSN n'est pas directe, car le concept de « réservoir » n'existe pas en tant que tel dans MUDU. En effet, les réservoirs sont définis en fonction de leurs dispositifs d'isolation : BA_flot est par exemple un réservoir à toit flottant. De plus, le fluide du réservoir est modélisé via les points de connexion, qui permettent de relier le réservoir au circuit fluidique. La règle LSN serait alors du type :

les composants BA_ToitFlot et BA_Reac dont les points de connexion E1 ont pour caractéristique TempFlui de valeur $\geq$ 60 doivent être composants père de la relation AssoDispoProt dont les fils sont Calo ou GrillProt.

Plusieurs règles LF peuvent être envisagées. Si la vérification se fait au moment de l'instanciation, et que le modèle applicatif est de type OWL/SWRL, la règle LF a l'expression suivante :

BA_ToitFlot (?x), a_pt_cnx (?x, ?y), a_caract (?y, ?z), TempFlui (?z), a_valeur (?z, ">60") $\longrightarrow$ AssoDispoProt (?x, ?v), Calo (?v)

or

BA_ToitFlot (?x), a_pt_cnx (?x, ?y), a_caract (?y, ?z), TempFlui (?z), a_valeur (?z, "sup60") $\longrightarrow$ AssoDispoProt (?x, ?v), GrillProt (?v)

La question qui se pose, une fois le besoin sémantique explicité, est de savoir comment le représenter à travers le cadre de modélisation proposé. C'est à cette question que ré-

pondent les sous-sections suivantes. Dans un premier temps, une conceptualisation et une formalisation des informations partagées (dans l'ontologie du cycle de vie) est proposée. Dans un second temps, la définition d'ontologies de domaine, pour produire et exploiter ces informations partagées est réalisée, et enfin, l'exécution des règles au niveau applicatif est envisagée.

4.2.2 La définition du méta-modèle sémantique (l'ontologie du cycle de vie)

La formalisation ontologique des règles métiers

Dans la littérature, d'autres travaux ont proposé des transformations sémantiques pour implémenter la connaissance métier. L'étude qui est sans doute la plus aboutie à l'heure actuelle sur le sujet est le projet européen OntoRule [2]. Ce projet envisage l'utilisation de technologies du web sémantique pour gérer les règles métier, et notamment leur expression et implémentations depuis les politiques générales exprimées en langage naturel, jusqu'à leur expression formelle. Puisque cette transformation ne peut être automatisée, de par la complexité des langages mis en jeu, le projet OntoRule propose de passer par une transformation intermédiaire des règles, sous forme de validation de règles candidates [Guissé et al., 2012]. Cette validation est de plusieurs ordres et se fait à partir de la plate-forme Semex, et par l'intermédiaire du langage semi-naturel SBVR-SE (Semantic Business Vocabulary and Business Rules - Structured English). La validation est de plusieurs ordres :

- lexical : un remplacement des termes de la règle par un terme référent est réalisé ;
- contextuel : rend explicite les éléments de contexte, afin que les règles puissent être comprises par elles-mêmes ;
- syntaxique : une simplification de la syntaxe est effectuée afin de limiter les difficultés d'interprétation ;
- sémantique : le but est de rendre explicites des éléments implicites de la règle. Cette vérification est purement manuelle.

Les règles ainsi produites grâce au langage SBVR-SE, peuvent ensuite être implémentées dans un langage formel. D'ailleurs, des travaux ont montré la possibilité de transformation automatique entre SBVR et le langage de règles standard Object Constrain Language (OCL) [Bajwa et al., 2011].

L'automatisation du traitement des règles dans le projet OntoRule est donc essentiellement syntaxique. Cependant, comme nous l'avons déjà évoqué dans le chapitre 1, la sémantique est différente de la syntaxe. Les approches comme OntoRule sont des approches

2. http ://ontorule-project.eu/

fondées sur l'analyse de termes. Or, le but de l'ontologue est de formaliser un réseau de concepts, qui différent fondamentalement des termes, car ils sont définis en « intension » [Ottens, 2007]. Les règles obtenues dans OntoRule sont donc homogènes et bien écrites. Des correspondances syntaxiques peuvent être établies entre ces règles, mais rien ne garantit qu'elles soient porteuses de sens : leur sémantique n'est pas évaluée. Dans notre projet, nous voulons proposer des transformations sémantiques, c'est-à-dire des transformations qui préservent le sens métier de la règle. Il ne s'agit donc plus de gérer et stocker des documents ou des phrases porteuses de connaissance : aussi « intelligent » que puisse être ce catalogage, il ne peut en résulter qu'un ensemble cohérent de termes. Notre but est au contraire de modéliser la connaissance métier au travers de concepts porteurs de sens et d'interactions expressives entre ces concepts.

La modélisation des règles LN en fonction des concepts métier

La modélisation d'une règle est en général de type « condition - implication ». C'est par exemple le cas dans le langage Rule Interchange Format (RIF), où la règle est constituée de deux éléments : l'antécédent et le conséquent. Nous pensons que cette représentation, bien que largement répandue, ne répond pas entièrement aux exigences liées à l'expression des règles métier, et notamment, à celle d'une règle indépendante et suffisante par elle-même. Alors, en plus des deux éléments que sont la condition et l'implication, nous proposons d'ajouter à la définition de la règle LN un troisième élément : le contexte.

Ces trois éléments constitutifs de la règle LN : le contexte, la condition, l'implication, doivent faire référence aux concepts métier préalablement définis, afin que la règle soit modélisée de façon sémantique. Le lien entre les règles et les concepts métier n'est pas aisé à conceptualiser : la seule réalisation d'un tel lien dans la littérature a été faite par l'intermédiaire du langage SBVR, langage syntaxique fondé sur un dictionnaire de termes. Encore une fois, cette approche n'est pas *sémantique*. La question fondamentale est de savoir sur quels concepts la règle porte, et comment déduire de ces concepts une expression générique de la règle. La réponse à cette question semble, à l'expérience inextricable. Sûrement parce qu'une règle ne porte pas sur des concepts, ni sur des relations d'ailleurs. Une règles LN ne porte sur aucun des éléments du modèle métier en particulier : elle porte sur des faits, des situations, en d'autres termes, l'instanciation de liens entre relations et concepts, et non simplement les relations et les concepts eux-mêmes.

Lorsque Tim Berners-Lee présenta son projet « Linked Data » à l'émission TED [3], il déclara « data is a relationship », exprimant le fait que l'information se trouve dans

3. http ://www.ted.com/

l'instanciation d'une relation entre deux données. C'est d'ailleurs l'enjeu du projet Linked data : mettre sur le web des relations entre données, et non plus entre documents. Nous pensons également que l'information se trouve dans l'instanciation d'une relation. Le « fait » auquel la règle LN est relié est en réalité l'existence d'une ou plusieurs relations particulières entre des concepts. Par conséquent, l'élément fondamental auquel une règle doit faire référence est le triplet, constitué de deux concepts (un antécédent et un conséquent), et d'une relation instanciée entre ces concepts. La règle LN, ou plus précisément, les éléments fondamentaux constitutifs d'une règle LN (le contexte, la condition, l'implication) sont alors exprimés sous la forme d'une liste algébrique de triplets métier.

Illustration de l'expression d'une règle LN sous forme de triplet

Il est possible d'exprimer la règle L « le code matériel d'un robinet dépend de la nature du fluide *BA* s'il s'agit d'eau de circulation et *BR* s'il s'agit d'eau brute » en règle LN faisant référence à des triplets métier. Le tableau ?? liste les concepts métiers, relations et triplets utiles dans cet exemple.

concept	relation	triplet
robinet	a identifiant	t_1 =robinet - a identifiant - code matériel
fluide	vaut	t_2 =robinet - est traversé par - fluide
eau borée	est traversé par	t_3 =fluide - est de type - eau brute
eau brute	est de type	t_4 =fluide - est de type - eau borée
code matériel		t_5 =code matériel - vaut - BA
BA		t_6 =code matériel - vaut - BR
BR		

TABLE 4.1: La liste des concepts, relations et triplets de l'exemple

A partir du modèle métier ainsi défini, la règle L de départ peut être transformée en règle NL, comme illustré dans le tableau ??.

règle LN		
contexte	condition	implication
$t_1 + t_2$	$condition_1 = t_3$ $condition_2 = t_4$	$implication_1 = t_6$ $implication_2 = t_5$

TABLE 4.2: L'expression de la règle LN exemple

La règle LN ainsi obtenue est un cas particulier : il s'agit d'une règle multi-conditions.

Elle s'exprime sans difficulté dans un modèle « contexte - condition - implication », dès lors que l'implication est un élément lié à la condition et non à la règle en elle-même : chaque condition génère une implication particulière.

Enfin, afin de contextualiser complètement la règle LN, plusieurs attributs lui sont adjoints. Ces attributs sont optionnels, et définissent :

- l'activité du processus pendant laquelle la règle doit s'exécuter. Il s'agit bien de l'activité pendant laquelle la règle doit être prise en compte, et non celle pour laquelle elle s'applique. Par exemple, une règle de conception peut avoir pour objectif de faciliter la maintenance. Son attribut activité sera alors « conception » car c'est à ce moment du processus qu'il faut la prendre en compte ;
- la raison pour laquelle la règle s'applique. Dans notre exemple, la raison est « faciliter la maintenance » ;
- la méthode spécifique à employer pour appliquer la règle, si elle existe.

Les règles LN : une expression graphique fondée sur des triplets

Pour représenter la règle LN de manière univoque et non ambiguë, nous avons envisagé, sans succès, le traitement syntaxique automatique des éléments constitutifs de la règle. Le langage naturel étant extrêmement complexe, l'approche qui consiste à écrire directement un énoncé de la règle à partir de sa modélisation ne nous a pas parue pertinente, car il faut envisager toutes les subtilités de langage. A contrario, une approche graphique s'avère plus directe, et lève les problématiques d'interprétations multiples du modèle. La figure 4.3 présente les deux représentations graphiques (une par condition) de la règle exemple. On voit bien sur cet exemple, que la représentation graphique lève les problèmes syntaxique de « mise en phrases » de la règle, proposant une sémantique unique pour chaque triplet, dont le dessin oriente la lecture, plutôt que la grammaire. Ainsi, la dimension supplémentaire offerte par le dessin dépasse les problématiques d'interprétation, et garantit la préservation du flux sémantique métier.

Vers une expression automatique de LSN

La règle LN ainsi obtenue est donc une règle sémantique, dont la cohérence avec d'autres règles peut être garantie du fait que la règle est modélisée, et non plus simplement exprimée. Cependant, cette règle est orientée métier, et l'enjeu est de définir le plus facilement possible, et sans erreur, son expression dans le modèle de données. En d'autres termes, quels éléments du modèle MUDU sont concernés, et de quelle manière ? Cette

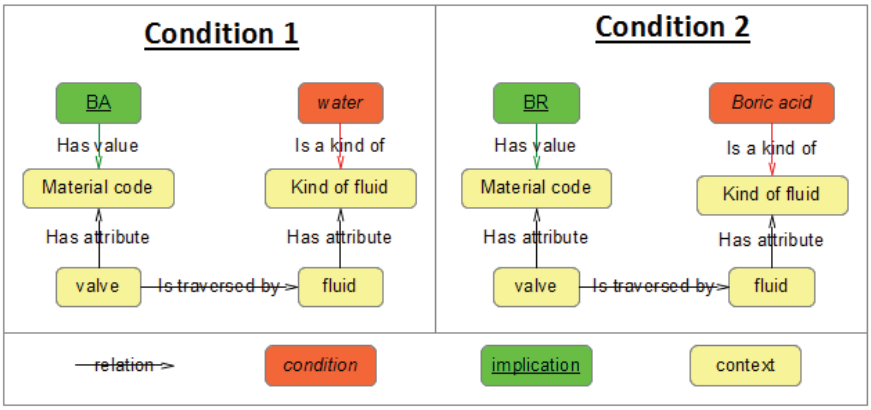


FIGURE 4.3 – Les représentations graphiques de la règle LN exemple

étape de transition de LN vers LSN est à faible valeur ajoutée. En effet, LN contient déjà toute la connaissance métier correctement formalisée, et la transformation LN vers LSN est alors une simple traduction de langage. C'est pourquoi, cette transformation doit se faire de la manière la plus automatique possible. Pour parvenir à cette automatisation, deux pré-requis s'imposent :

- la règle LSN doit avoir un modèle proche de LN, afin de pouvoir faire correspondre des éléments de l'une et de l'autre de manière directe ;
- il doit exister a priori un mapping entre les éléments métier et ceux du modèle MUDU, comme un « dictionnaire » de traduction entre ces deux points de vue afin que l'automatisation soit réalisable.

Par conséquent, les règles LSN ont une structure semblable à celle de LN : elles sont constituées d'un contexte, d'une condition et d'une implication de cette condition. Elles s'expriment également sous forme de triplets, dont le niveau sémantique est proche de celui des triplets métier. Simplement, ces triplets sont constitués d'éléments des dictionnaires MUDU, et des relations du méta-modèle OpenCat. La figure 4.2.1 présente les deux expressions LN et LSN de la règle exemple appliquée aux robinets et soupapes pour l'une des conditions, ainsi que le mapping entre les triplets métier et MUDU nécessaire à cette transformation. On remarque sur cet exemple que ce mapping n'est pas direct. Un triplet métier peut correspondre à plusieurs triplets MUDU, et vice versa. Par exemple, l'expression du fait qu'un fluide traverse un robinet, un clapet ou une soupape, en MUDU, se fait par l'intermédiaire des points de connexion, qui possèdent une caractéristique NatFluid permettant de décrire le fluide. Le triplet métier t_{60} *soupape - est traversé par - fluide* correspond alors à plusieurs triplets MUDU : *Soup - a connexion - Point de connexion* + *Point de connexion - a caracteristique - NatFluid*. Il est à noter que dans notre cas d'étude la transformation LN - LSN n'est pas unique, car le modèle MUDU est un ensemble de

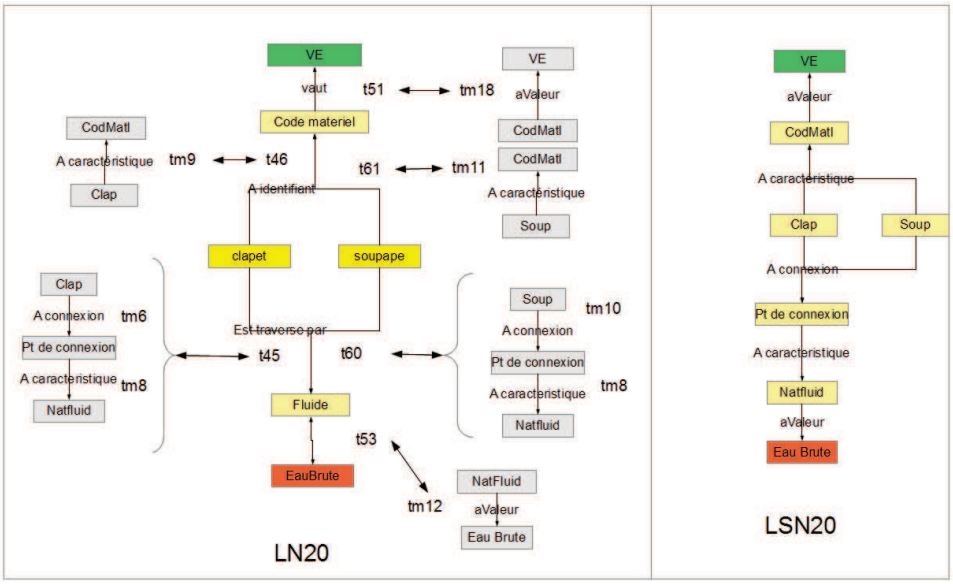


FIGURE 4.4 – Mapping entre LN et LSN

catalogues métier, et chaque catalogue a une représentation différente du produit. Ainsi, il faut établir un mapping par catalogue, et une règle LSN par catalogue. Pour plus de facilité, nous avons en premier lieu travaillé sur un unique catalogue, avant d'étendre la démarche à MUDU entier.

L'expression de la sémantique des règles LSN est donc finalement identique à celle des règles LN : un contexte, une condition, une implication. Seulement ces éléments de la règle LSN sont liés à des triplets MUDU. L'expression du mapping entre le modèle métier et le modèle MUDU est également assez simple dans l'ontologie du cycle de vie : chaque instance de mapping relie un ou plusieurs triplets métier à un ou plusieurs triplets MUDU.

La formalisation des règles dans l'ontologie du cycle de vie

Compte-tenu de la modélisation présentée précédemment, la figure 4.2.1 présente la taxonomie du sous-ensemble de l'ontologie du cycle de vie nécessaire à la formalisation des règles métier [4]. Ce modèle, décrit plus en détail dans l'annexe 7.3, permet de représenter complètement les règles et leurs liens sémantiques, mais ne modélise pas du tout les règles et restrictions nécessaires à leur édition ou exploitation. Ainsi, cette ontologie contient complètement et uniquement la sémantique des règles. Ce modèle est alors suffisant pour échanger les informations liées aux règles. En revanche, il ne permet aucun raisonnement

4. En annexe 7.3 se trouve une description complète de cette ontologie

sur leur structure. En revanche, la sémantique des ontologies est pleinement exploitée quand cela est nécessaire. Par exemple, plusieurs règles L peuvent être exprimées par la même règle LN : il y a des redondances possibles au niveau L, mais pas au niveau LN. Alors, la propriété *est_exprimee_par* qui relie une règle L à une règle LN possède une sous-propriété : *est_maitre_de* permettant de déclarer parmi les règles L, celle qui est l'origine véritable de la règle LN (remonter à la source). Cette propriété est définie comme *functional*, non pour vérifier la cohérence du modèle par raisonnement, mais pour exprimer le fait que la relation est de cardinalité 1.

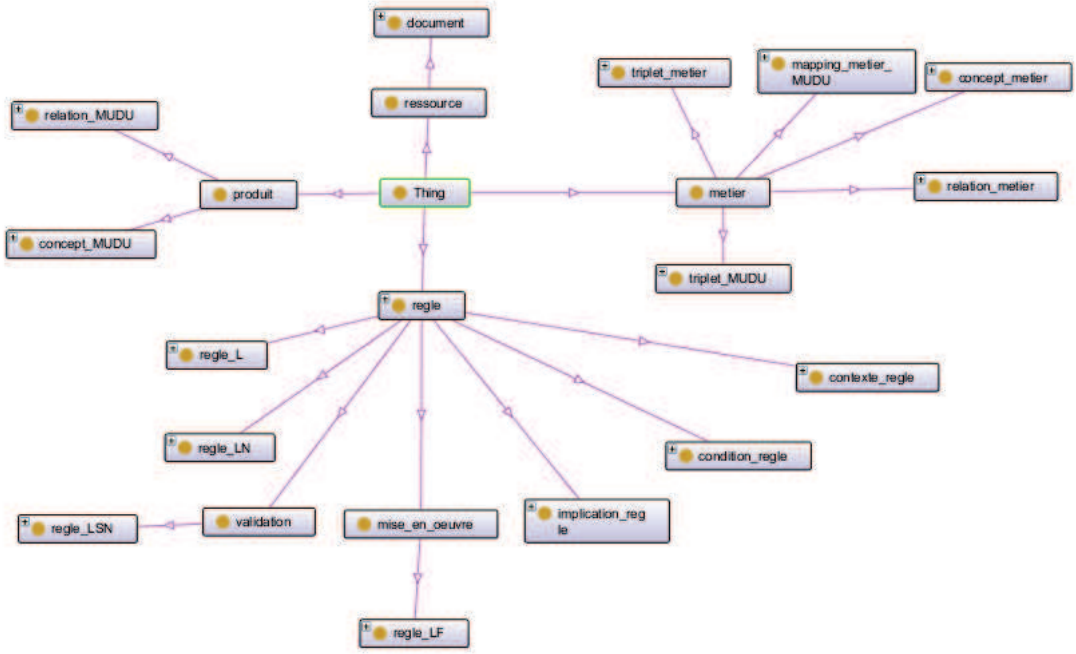


FIGURE 4.5 – Méta-modèle formalisant la sémantique des règles

La validation de la formalisation proposée

Valider une modélisation est difficile. la méthode OntoClean [Guarino and Welty, 2009] propose de vérifier qu'il n'y a pas d'incohérence « philosophique » dans la description des classes et relations. Pour cela, on définit les différentes propriétés de l'ontologie suivant quatre méta-propriétés. Il est à entendre ici par « propriété » le sens philosophique, c'est-à-dire aussi bien une classe qu'un rôle. En effet, une classe n'est autre que l'expression d'une propriété : par exemple la classe **regle_L** exprime la propriété *être une règle textuelle en langage naturel*. Les quatre méta-propriétés sont :

- l'identité (I), qui exprime le fait de porter un axiome d'identité, autrement dit, les conditions nécessaires et suffisantes de la propriété. Quand une propriété porte sa propre identité, elle est représentée par un O (pour « own ») ;
- l'unité (U), qui exprime le fait que les instances de la propriété sont un ensemble délimité (par exemple un océan est unique, mais pas une étendue d'eau, qui peut être de différentes natures et qu'on ne peut délimiter) ;
- la rigidité (R) : une propriété est rigide si les instances de cette propriété ne peuvent être autre chose que cela. Par exemple, un moteur, qu'il soit dans un bateau ou une voiture est toujours un moteur, tandis qu'un élément de voiture n'est pas nécessairement et intrinsèquement toujours un élément de voiture (le même moteur peut être un élément de voiture ou un élément de bateau) ;
- la dépendance (D) : deux propriétés sont dépendantes quand elles partagent des instances.

Ces quatre méta-propriété sont valorisées : elles peuvent être positives (« + », toutes les instances correspondent à la méta-propriété), négatives (« − » toutes les instances ne correspondent pas nécessairement à la méta-propriété) ou *anti-* (« ∼ », aucune instance ne correspond à la méta-propriété).

A partir de ces propriétés, il est possible d'établir des règles de conception, fondées sur la subsomption (hiérarchie) entre propriétés, notamment, dans le cas ou p est sous-classe de q (q *subsome* p) :

- si q est anti-rigide (∼R), alors p l'est aussi ;
- si q porte un axiome d'identité (+I) alors p en hérite ;
- si q porte l'unité (+U) alors p en hérite ;
- si q porte l'anti-unité (∼ U) alors p également ;
- toute instance doit appartenir à une seule sur-propriété +I.

La figure 4.2.1 présente la catégorisation des classes de l'ontologie de règles vis à vis des trois méta-propriétés U, I et R. Cette évaluation de l'ontologie permet de détecter une erreur de modélisation (encadrée en rouge sur la figure 4.2.1). En effet, nous avons défini conceptuellement une règle LSN comme étant un moyen de valider une règle LN, et une règle LF comme étant un moyen de mettre en oeuvre (exécuter) une règle LSN. Nous avons formalisé cette conceptualisation en désignant **regle_LSN** comme une sous-classe de **validation** et **regle_LF** comme une sous classe de **mise_en_oeuvre**. Nous rappelons ci-après les propriétés de ces classes :

- **regle_LSN** : *être une règle exprimée en langage semi-naturel à partir des éléments du produit* ;
- **regle_LF** : *être une règle exprimée en langage formel* ;
- **validation** : *être un moyen de valider une règle* ;
- **mise_en_oeuvre** : *être un moyen d'exécuter une règle* ;

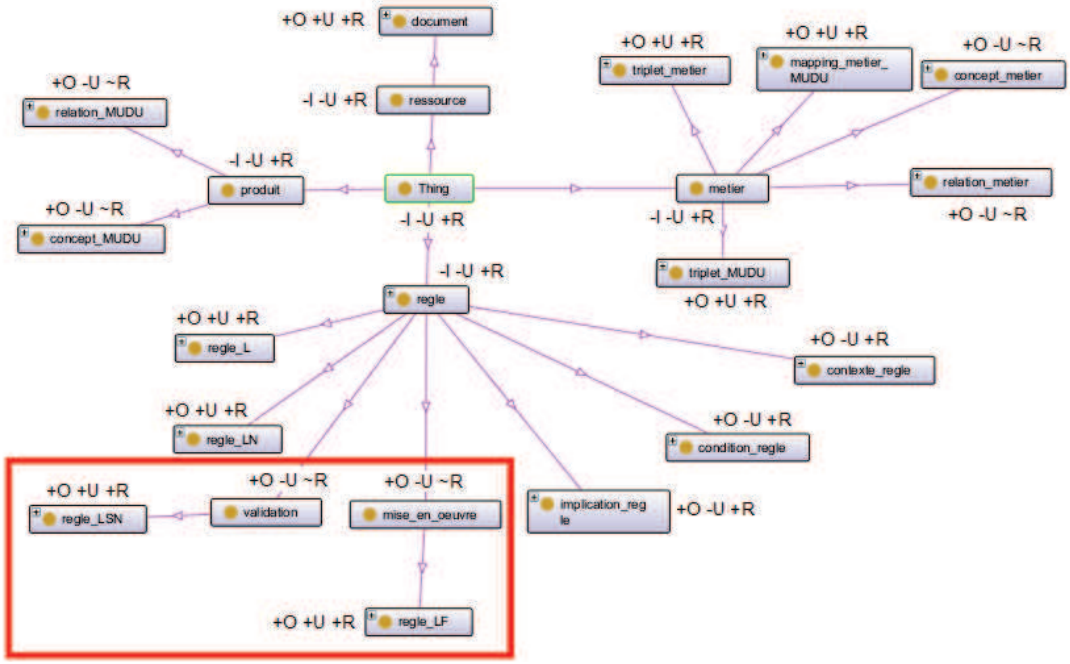


FIGURE 4.6 – Evaluation OntoClean de l'ontologie de règles

Selon ces propriétés, aucune instance de validation n'en est intrinsèquement une, donc la propriété est anti-rigide (∼R). En effet, le fait qu'une règle soit un moyen de valider une autre règle est une caractéristique particulière de la règle, qui ne la définit pas spécifiquement, et qu'elle peut perdre ou gagner au fil du temps. En revanche, le fait d'être une règle LSN est rigide (+R). **Regle_LSN** n'est donc pas sous-classe de **validation**. Idem pour **mise_en_oeuvre** et **regle_LF**. Ces quatres classes sont sous-classes de **regle**, et les couples **mise_en_oeuvre**/**regle_LF** et **validation**/**regle_LSN** sont liés (ils partagent des instances).

4.2.3 L'utilisation des ontologies de domaine

L'ontologie du cycle de vie définie précédemment doit être alimentée par les experts métier : il faut éditer les règles, à partir de triplets métier pré-établis, et de mapping entre les éléments métiers et le modèle MUDU. Pour générer les triplets métier de manière cohérente et semi-automatique, l'utilisation d'une ontologie de domaine comme modèle métier semble pertinente pour deux raisons principales :

– l'ontologie définie sera porteuse d'une sémantique riche et intelligible par les personnes du métier ;

– l'inférence permettra d'automatiser une partie de l'instanciation des informations, afin de minimiser l'engagement des experts métier dans le travail de captation, mais aussi de vérifier la cohérence des informations modélisées.

La définition du modèle métier comme ontologie de domaine

Le « modèle métier » est donc une ontologie de représentation d'une connaissance métier partagée. Pour concevoir (mais aussi évaluer) ce type d'ontologies, [Gruber, 1995] propose cinq critères de conception, qui sont :

– la clarté : les définitions de concepts se doivent d'être les plus objectives possibles. Il est donc nécessaire de définir par des axiomes logiques non seulement les conditions nécessaires mais aussi les conditions suffisantes d'appartenance à un concept. Il est également important de définir chaque concept dans un langage naturel ;
– la cohérence : il faut mettre en place un mécanisme d'inférence permettant de garantir la cohérence des axiomes logiques définis ;
– l'extensibilité : il s'agit d'anticiper sur l'utilisation future du vocabulaire partagé dans l'ontologie, en s'appuyant sur un ensemble de tâches à remplir. De plus, il faut s'assurer qu'il est possible de définir de nouveau termes sans remettre en question le sens et la définition des termes existants ;
– les biais d'encodage : ils doivent être minimisés. Il ne faut pas, par exemple, faire dépendre la définition des termes de symboles spécifiques. Le but est de maintenir un niveau de description de la connaissance, sans inclure un formalisme particulier ;
– l'engagement ontologique : il doit être minimum. Il s'agit de définir uniquement les termes et concepts nécessaires à la réalisation des tâches prédéfinies ou à la communication entre experts.

La conceptualisation du domaine métier

La première étape pour modéliser l'ontologie métier est donc de définir les tâches qui lui seront attribuées. Le but de cette ontologie est de permettre la définition de triplets métier, qui serviront de support à l'expression des règles métier. Il faut permettre :

– l'édition aisée et assistée (par des automatismes) de ces triplets ;
– de vérifier la cohérence sémantique des triplets pour chaque règle ;
– de définir un modèle suffisamment riche pour exprimer toutes les règles métier ;
– de faire converger tous les points de vue métier en une seule représentation ;
– de restreindre l'expression des règles uniquement à ce qui a été explicitement défini (ne pas laisser de liberté à l'éditeur de règles).

Pour construire l'ontologie métier, nous avons identifié les concepts nécessaires ainsi que les relations sémantiques entre ces concepts. Cette conceptualisation s'est limitée dans un premier temps au champ de la robinetterie, si bien que l'ontologie définie sera incomplète et doit être extensible.

La formalisation des concepts métier

Une fois la conceptualisation faite, la formalisation du modèle métier peut être envisagée. Nous avons choisi de formaliser les concepts métier comme des instances, instances auxquelles les règles sont liées via les triplets. Ainsi, le niveau de formalisation des concepts est le même que celui des règles. La figure 4.7 présente l'ontologie métier suite à la définition du modèle métier pour un ensemble de règles portant sur la robinetterie. Cette ontologie est exprimée en OWL2-DL/SWRL. Pour assurer la clarté et la cohérence du modèle, les instances sont réunies dans des classes métier, qui sont reliées par des propriétés spécifiques. Chaque instance possède une propriété objet qui permet de définir le concept en langage naturel. Les classes elles, sont définies par des axiomes et des restrictions sur les propriétés. En suivant le critère d'engagement ontologique minimum, nous n'avons défini pour le moment que les classes nécessaires au domaine restreint que nous étudions (la robinetterie). L'ensemble des propriétés et classes sont présentées en détail dans l'annexe 7.5.

La première classe définie est la classe **composant**. Cette classe ne fait pas uniquement référence aux composants physiques de l'installation, mais à l'ensemble des éléments tangibles qui la composent. Ainsi, une tranche de l'installation, un fluide, ou encore une ligne entière de tuyauterie sont considérés comme des composants. Par conséquent, il est possible de relier les composants entre eux par tout type de propriétés. Nous les appelons « propriétés spécifiques ». La liste de ces propriétés peut être étendue selon le besoin. Chaque composant possède également des éléments particulier, définis dans des classes disjointes :

- un état particulier : par exemple, un fluide peut être liquide ou gazeux. Pour un composant physique, « maintenu sous-pression » est un état particulier. La relation associée est *a_etat* ;
- une fonction : par exemple, un réservoir peut servir à transporter un liquide ou à l'entreposer. La relation associée est *assure* ;
- un matériau : il s'agit de matériau constitutif de composants physiques. La relation associée est *est_fait_de* ;
- des propriétés : un fluide peut par exemple être inflammable. La relation associée est *a_propriete* ;

– une localisation particulière : les réservoirs peuvent par exemple être positionnés à l'intérieur ou à l'extérieur de l'enceinte. La relation associée est *a_localisation* ;
– une orientation : par exemple, horizontale ou verticale. La relation associée est *a_orientation* ;
– des attributs : ils peuvent être de différents types (identifiants, classement, quantité, température). La particularité des attributs est que ce sont les seules instances qui prennent des valeurs particulières (définies par une dataproperty). La relation associée est *a_attribut*.

La définition des conditions nécessaires et suffisantes des différentes classes n'est pas aisée, car elle entre en contradiction avec la nécessité d'extensibilité et d'engagement ontologique minimal. Nous nous sommes donc limités à la définition des conditions nécessaires, au travers de la définition des domaines de propriétés et de la disjonction des classes.

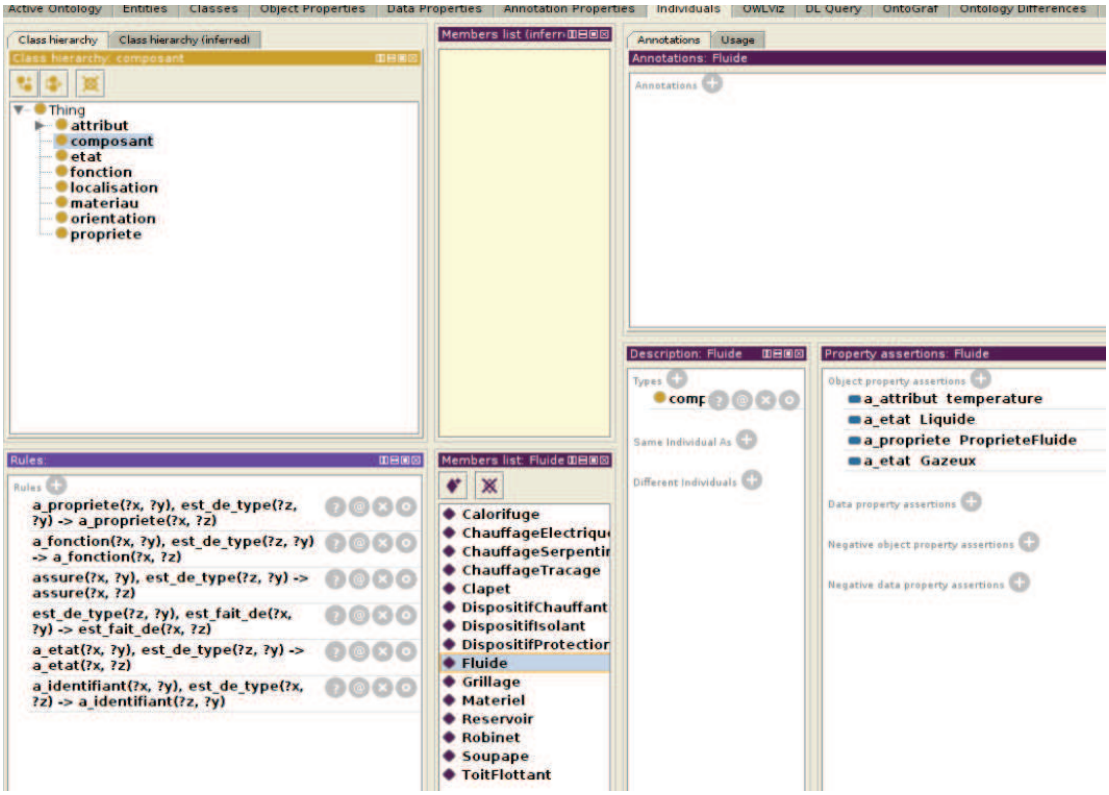


FIGURE 4.7 – L'ontologie métier : classes, instances et règles SWRL

Pour permettre une spécialisation des concepts (taxonomie à plusieurs niveaux), nous utilisons la propriété *est_de_type*. Par exemple, `propriété fluide` est une instance de la classe **propriété**, et cette instance *est_de_type* `radioactif` ou `inflammable` (qui sont

également des instances de la classe **propriété**). Cette propriété permet donc d'exprimer le domaine métier à différents niveaux d'abstraction, et par conséquent, de faire converger différentes vues métier. Elle est définie comme transitive pour permettre l'héritage.

Afin à la fois d'assurer la cohérence du modèle, et de permettre d'automatiser l'instanciation de relations (nous rappelons que l'une des exigences de ce modèle est de générer tous les triplets métier à partir d'un travail de formalisation métier minimum), le modèle est contraint par des règles SWRL. Un des aspects de l'automatisation du modèle est de permettre de remonter les liens d'héritage. Plus précisément, il faut pouvoir exprimer que chaque fois qu'un composant est lié à un élément particulier (fonction, propriété, etc.), il est également lié à l'ensemble des concepts supérieurs de cet élément (les « parents » de la propriété *est_de_type*). Par exemple, si un `fluide` *a_propriete* `inflammable` alors il a aussi pour propriété `propriete fluide` qui est le parent d'`inflammable` par la relation *est_de_type*. Cette règle d'héritage s'exprime par plusieurs règles SWRL (une pour chaque propriété), selon le type générique suivant :

$$a\_propriete(?x, ?y),\ est\_de\_type(?z, ?y) \rightarrow a\_propriete(?x, ?z)$$

Enfin, l'une des vérifications de cohérence importante est de vérifier que les instances liées par la propriété *est_de_type* sont bien de la même classe. SWRL ne permet pas d'exprimer cette règle en l'état. Il est possible de l'exprimer par des moyens détournés :

- en définissant une propriété *est_de_type* pour chaque classe du modèle. La vérification se fait alors en limitant les domaines de définition de chaque propriété ;
- en créant une classe **« incohérence »** dans laquelle sont placées les instances qui posent problème, via l'application d'une règle SWRL pour chaque classe du modèle.

Pour ne pas alourdir le modèle, nous avons choisi de conserver une vérification manuelle.

Une fois l'ontologie instanciée par les métiers, l'ensemble des liens entre instances représentent l'ensemble des triplets métier autorisés pour l'expression des règles métier. Alors l'export RDF de la Abox de l'ontologie métier permet donc d'obtenir l'ensemble des triplets de l'ontologie du cycle de vie. Cependant, pour l'expression des règles, il parait également important de conserver la sémantique des concepts et relations de l'ontologie métier, et on seulement les triplets RDF. Alors, des mappings peuvent être établis entre l'ontologie métier et l'ontologie du cycle de vie. Ces mappings sont envisagés dans la section 4.2.5.

4.2.4 La mise en oeuvre des règles au niveau applicatif

A partir d'une même règle LSN, qui est une règle en langage semi-naturel intelligible par l'expert SI, il existe plusieurs règles LF. En effet, il y a autant de règles LF qu'il y a

de langages applicatifs exécutant les règles. Il y a également autant de règles LF qu'il y a de démarches de mise en oeuvre de la règle. La règle LF a donc une pérennité faible, et n'est pas unique. On fait face à ce niveau à un éclatement du nombre de règles.

Dans l'hypothèse où la vérification de la règle se fait a posteriori, et par le biais d'une ontologie d'application, la règle LF doit être exprimée en SWRL, seul langage de règles conforme à OWL-DL. Dans la contribution soumise à l'IJPLM, insérée en annexe 7.6, nous avons évalué sur un cas d'application les capacités de SWRL pour exprimer toutes les règles métier nécessaires, ce qui a conduit à émettre des critiques quant à l'expressivité de SWRL. Dans cette contribution, une ontologie d'application a été définie à partir de la sémantique de MUDU, le modèle unifié d'EDF. Cette ontologie porte uniquement sur les informations liées à l'alimentation électrique d'un composant. Les règles, définies de manière textuelles au niveau de MUDU doivent être implémentées de manière formelle dans l'ontologie d'application, afin d'être exécutées sur les occurrences. Nous avons donc sélectionné les règles portant sur le cas d'étude, seulement, certaines de ces règles n'ont pu être exprimées du fait de limites sémantiques de OWL/SWRL sur lesquelles nous revenons dans la section 4.3.2.

4.2.5 Les correspondances entre les modèles

Les ontologies de domaines sont pour le moment décorrélées de l'ontologie du cycle de vie, si bien qu'il n'est pas possible d'échanger des informations entre ces ontologies. Il faut procéder pour cela à une transformation de modèles entre l'ontologie du cycle de vie et les ontologies de domaine. Cette transformation de modèles peut se faire de deux manières :

- par l'utilisation de SPARQL, langage de requête sur RDF, qui permet, notamment via la fonction CONSTRUCT et les blank nodes (en français « ressources anonymes », qui permettent de lier et d'identifier un ensemble cohérent de ressources RDF, par exemple, une liste de triplets métier) de transformer les données d'un schéma RDF à un autre ;
- par l'utilisation de XSLT, en supposant que le format d'échange de données soit XML.

Dans les deux cas, il faut spécifier les transformations de modèle pour chacune des ontologies de domaine et d'application. Cela peut représenter un effort conséquent a priori, mais si les règles de transformation sont écrites au niveau des méta-modèles, alors, ces transformations sont pérennes dans le temps.

Illustration de règles de transformation de modèles

Pour illustrer le type de transformation de modèle nécessaire pour faire dialoguer l'ontologie du cycle de vie avec les ontologies de domaine, on peut prendre l'exemple simple de la transformation avec l'ontologie métier.

Règles de transformation depuis le modèle métier vers l'ontologie du cycle de vie :

- pour chaque classe de l'ontologie métier, créer un concept métier ;
- pour chaque instance de l'ontologie métier, créer un concept métier ;
- pour chaque propriété objet de l'ontologie, créer une relation métier ;
- pour chaque instance i, lister les propriétés instanciées. Pour chaque instanciation de propriété p, rechercher l'instance liée j, et créer une instance de triplet métier. Lier le triplet créé à l'instance i par la relation *a_antecedent*, à la relation métier p par la relation *a_relation* et à l'instance j par la relation *a_consequent* ;
- chaque fois qu'une instance i appartient à la classe c, créer un triplet avec pour antécédent c, pour relation métier *est_de_type* et pour conséquent i.

Ces quelques règles permettent de transformer les instances de l'ontologie métier en une représentation sous forme de triplets dans l'ontologie du cycle de vie.

Conclusion de la section 4.2

L'exemple traité dans cette section permet d'illustrer le cadre de modélisation proposé dans la section 3.1. Les différents niveaux de modélisation y sont spécifiés sur un cas d'étude, et les échanges entre les niveaux sont également envisagés : ils peuvent être une contrainte à la définition des modèles ontologiques.

Dans cette application, l'ontologie du cycle de vie a un rôle de modèle pivot, qui permet d'exprimer sémantiquement les informations qui doivent être partagées et échangées entre les différents domaines et les applications. Cette ontologie, pour être robuste, pérenne, et s'adapter à chaque besoin doit être uniquement descriptive : elle représente la sémantique des informations. Aucune vérification ou inférence n'est faite à ce niveau. En effet, le raisonnement dépend toujours d'un point de vue particulier : il est le reflet d'un parti pris, tandis que l'ontologie du cycle de vie doit être générique.

Le raisonnement est alors reporté aux niveaux de domaine et d'application, où les capacités d'inférences des ontologies sont pleinement exploitées, mais à des fins différentes. Dans le cas des modèles de domaine l'inférence des langages OWL permet de faire porter une partie de la cohérence métier par le modèle lui-même, conduisant ainsi à une semi-

automatisation du traitement de l'information, soit par des axiomes OWL, soit par des règles et requêtes. De plus, OWL proposant une sémantique riche, le modèle est directement compréhensible par le métier, qui peut être partie prenante dans sa conception et son évolution. Au niveau applicatif, l'inférence OWL est utilisée pour « mettre en œuvre » la sémantique du modèle unifié, et garantir la conformité des occurrences à ce modèle, grâce à l'exécution de restrictions ou de règles.

La mise en œuvre des informations au niveau applicatif n'est pas chose aisée. Malgré la richesse sémantique autorisée par les ontologies et l'inférence, la modélisation ontologique des informations au niveau applicatif n'est pas pleinement robuste, notamment en raison des faibles capacités expressives des langages de règles, et de l'Open World Assumption, sur lesquelles nous revenons dans la section suivante, qui traite des apports et des limites des ontologies pour le cadre de modélisation proposé.

4.3 Les apports et limites du cadre de modélisation ontologique

4.3.1 L'apport du cadre de modélisation : une discussion

Malgré les limites des ontologies identifiées dans ces travaux et synthétisées dans la section suivante, la mise en place du cadre de modélisation ontologique sur un cas d'application permet d'identifier ou de confirmer les apports des ontologies dans une démarche de modélisation. Nous développons plus particulièrement deux points essentiels en termes de modélisation :

- le premier point est que la distributivité des modèles introduite par la modélisation ontologique permet de différencier la représentation des informations - dans une logique d'échange - du raisonnement sur ces informations, permettant d'exploiter au mieux, et selon un besoin précis les capacités des ontologies. L'enjeu est alors de savoir où placer le « curseur » entre représentation d'une sémantique générique et exploitation de l'information ;
- le deuxième point réside dans la conceptualisation, et l'analyse qui en résulte. Les ontologies, puisqu'elles manipulent des donnée non-canoniques et reposent sur la définition d'une Tbox expressive, forcent à une démarche fondée sur les concepts, et de plus, permettent la validation logique de cette conceptualisation.

Placer le curseur entre représentation et exploitation de l'information

Le modèle partagé, au-delà des concepts racines, doit permettre de représenter l'ensemble des informations selon le même paradigme de modélisation. Cependant, il ne pro-

cure qu'une représentation statique des informations : cela est nécessaire pour garantir sa généricité et sa pérennité. L'enjeu est alors d'amener dynamiquement des informations vers le modèle pivot, et, par exemple, d'exploiter les capacités d'inférence des ontologies. Dans notre approche, ce travail d'inférence sur les informations, qui permet à la fois de capter la connaissance, mais aussi de réutiliser « intelligemment » les informations a posteriori, doit être dé-corrélé de la représentation unifiée des informations proposée dans l'ontologie du cycle de vie. Outre les problématiques de performance posées par le raisonnement, sur lesquelles nous reviendrons, la raison fondamentale de cette séparation est conceptuelle. En effet, une inférence est toujours dépendante d'un point de vue particulier : il s'agit de « faire parler » la donnée, en d'autres termes, de lui conférer un point de vue particulier. Nous pensons que cela se fait toujours dans un périmètre restreint, et pour une application particulière, et ne peut donc se faire au niveau du modèle partagé. De plus, limiter l'inférence à ce niveau de modélisation permet de s'affranchir des problèmes de performance, de promouvoir une simplicité du modèle et de garantir par conséquent son extensibilité. Il peut être cependant difficile de déterminer où « placer le curseur ». Nous pensons que ce sont les deux étapes de conceptualisation de la méthode générale proposée (étapes 2 et 3) qui permettent, après itérations, d'aboutir à un compromis, et donc à la définition de l'ontologie du cycle de vie et des ontologies de domaine.

Une réflexion fondée sur les concepts

L'un des points essentiels de cette étude et du cas d'application des règles, est que, pour permettre d'exploiter au mieux les capacités d'inférence des ontologies - c'est-à-dire pour parvenir à faire interagir les informations entre elles, et notamment les informations métier sources de connaissance comme les règles - il faut sortir du paradigme actuel qui propose une gestion des supports de connaissance (bases documentaires, lien hypertexte), voire au mieux, de termes et de syntaxe, pour tendre vers un modèle conceptuel de la connaissance. De cette représentation fondée sur des concepts sémantiques et leurs interactions pourra naître une intelligence de la donnée, ou plutôt, des données, puisque leur structure ne sera plus seulement l'expression d'une organisation ou de ressemblances terminologiques. Les relations porteuses de sens, établies entre les données, en feront au contraire des informations contextualisées. Dès lors, le raisonnement, l'automatisation, et, dans une certaine mesure, « l'intelligence » des modèles métier sera possible.

Pour exprimer les règles LN par exemple, il aurait été envisageable (outre le problème linguistique), d'utiliser un traitement du même type qu'OntoloRule, fondé sur un langage structuré, SBVR et un traitement synonymique des termes. Ainsi, les termes les règles auraient été bien écrites. Certains liens auraient pu être établis entre ces règles, si elle

employaient par exemple des termes ou des structures syntaxiques similaires. Cependant, il aurait été impossible de requêter par exemple sur l'ensemble des composant dont le classement de sureté inclut une clause de séisme (appelé dans le vocabulaire nucléaire les « composants classés séisme »), car en aucun cas SBVR ne modélise le fait qu'une relation d'association existe entre les concepts composants et classement de sûreté. Sans doute, plusieurs expressions syntaxiquement correcte permettent de l'écrire, et rien ne les lie sémantiquement entre elles.

La modélisation ontologique en soit est une démarche qui se fonde sur des concepts. La démarche de [Gruber, 1995] et celle d'OntoClean par exemple incluent une réflexion sur les concepts, qui est poussée par la structure même de l'ontologie. En ce sens, la démarche ontologique force à l'innovation et à la réflexion conceptuelle lors de la définition des modèles. Sans cette réflexion conceptuelle, notamment, l'inférence se trouve vide de sens.

4.3.2 Les limites des ontologies pour la modélisation envisagée

Dans cette section, nous évaluons les modèles ontologiques proposés et exprimons les limites des ontologies à chaque niveau de la modélisation, identifiées au travers du cas d'application.

Les limites d'expressivité des ontologies pour les modèles de domaine

La première limite des langages OWL au niveau de domaine est l'absence de support de la fonction de méta-modélisation d'OWL2 (fonction de « punning »). Les ontologies OWL(1) sont des modèles à deux niveaux : le niveau de classes, qui catégorise les éléments du modèle, et le niveau d'instances sur lequel il est possible d'instancier des relations, de raisonner, et d'appliquer des règles. Cette décomposition en deux niveaux est à la base du traitement des raisonneurs, qui manipulent séparément la Abox de la Tbox. Seulement, nombres d'applications industrielles ont besoin de plusieurs niveaux d'instanciation. L'exemple de l'ontologie métier pour l'expression des règles en langage naturel en est une bonne illustration : le domaine métier est instancié à plusieurs niveaux, car il unifie plusieurs points de vues métiers et à plusieurs niveaux d'abstraction. Souvent donc, pour les ontologies de domaine, une approche dite de « méta-modélisation » est nécessaire.

OWL2 permet normalement cette approche méta-modèle, par la fonction de « punning ». Cette fonction permet en effet que le même élément de l'ontologie puisse être à la fois une classe et une instance (et aussi une propriété objet). Grâce à cette double-représentation, il est envisageable de représenter et d'instancier une taxonomie à plusieurs niveaux : les

classes portent la hiérarchie des concepts, tandis que les instances portent les relations. Cependant, dans les faits, cette propriété d'OWL2 est implémentée comme suit par les raisonneurs : une classe et une instance peuvent porter le même nom, mais elles n'héritent par de leurs propriétés respectives. Le raisonneur les traite comme deux objets différents ; il faut transmettre manuellement les propriétés de l'une à l'autre. Ainsi implémenté, le punning ne permet pas à l'heure actuelle de lever le problème de méta-modélisation, qui est fréquent pour les ontologies de domaine, et notamment les ontologies « métier » comme celle du cas d'étude.

Pour contourner l'incomplétude du « punning » nous avons choisi une approche qui consiste à modéliser tous les objets métier comme des instances. Elle privilégie donc le niveau « instance », même si elle conserve un méta-modèle générique permettant une relative inférence. Alors, des propriétés particulières et des règles SWRL permettent de *re-créer* en partie les fonctionnalités du punning en automatisant partiellement l'héritage des propriétés. Cependant, la perte d'inférence est notable. Une autre approche est possible, dont nous discutons dans la section 5.2

Les limites des ontologies pour les modèles d'application

Les limites des ontologies au niveau applicatif portent sur la modélisation des règles, et à nouveau, sur l'Open World Assumption.

Pour les modèles applicatifs, pouvoir exécuter des règles sur les instances du modèle est une exigence très fréquente dans les applications industrielles. En effet, tout produit répond à des exigences (de fonctionnement, de sécurité, etc.), qui doivent être garanties. Pouvoir garantir ces exigences par l'exécution de règles est stratégique pour l'industrie, puisque cela permet à la fois d'automatiser des processus métier (délai raccourcis), d'éviter les erreurs humaines (qualité augmentée), et d'alléger les processus de vérification (coût baissé).

Comme nous l'avons montré dans la contribution [Fortineau et al., 2013c] - reportée en annexe 7.6 - au travers de l'étude d'une portion de centrale nucléaire, mais aussi au travers de l'étude de la littérature, le langage SWRL n'est pas adapté à des applications industrielles. Voici une liste (non-exhaustive) des règles que SWRL ne permet pas d'exprimer, et qui pourtant sont fréquentes dans l'industrie :

- les règles fondées sur des axiomes négatifs. Par exemple, une règle qui dit « si le robinet n'est pas de type réglant, alors... » ne peut pas être exécutée en SWRL, à moins que l'absence de propriété réglante sur un robinet ne soit formellement modélisée. Cette limite est dûe à l'Open World Assumption ;

- les règles de comparaison, du type : « si deux alimentations électriques se trouvent sur la même branche électrique, alors la tension de celle en amont doit être égale ou supérieure à celle de l'alimentation en aval ». SWRL en effet ne permet pas la comparaison de valeurs ni dans son antécédent, ni dans son conséquent. Il ne permet pas non plus les opérations de base ;
- les règles multi-conditions, du type « si le fluide traversant le robinet est radioactif, alors son classement sismique est K3, et s'il est de plus inflammable, le robinet doit posséder un dispositif de protection ». En SWRL, il est nécessaire d'écrire deux règles : une pour chaque combinaison de conditions. Il arrive qu'il y ait des règles qui encapsulent un plus grand nombre de conditions, qui, si elles se conjuguent bien, font exploser le nombre de combinaisons possibles. Il serait plus intéressant de pouvoir écrire toutes ces conditions en une seule règle ;
- les règles contenant un « ou », du type : « si la température traversant le robinet est supérieure à 60°C, alors le robinet doit être isolé par un calorifuge ou un grillage de protection ». En effet, la syntaxe de SWRL ne permet que la conjonction d'axiomes.

Malgré tout, SWRL est le seul langage de règles qui soit à la fois standard et conforme à OWL. Son manque d'expressivité est donc une limite importante à la diffusion et l'implémentation à large échelle d'ontologies, plus particulièrement d'ontologies d'application. Mais ce n'est pas la seule limite de ce langage.

Il n'est pas possible non plus avec le langage SWRL de réaliser les opérations CRUD (créer, lire, mettre à jour et effacer). Cela est une limite forte de SWRL pour l'expression des règles métier. Plus particulièrement, SWRL ne permet pas de modifier des informations existantes, ce qui est souvent nécessaire pour exploiter l'inférence ontologique. Cette constatation est par exemple une limite au travail de [Rossellò-Busquet et al., 2011] qui utilisent des ontologies pour gérer la dépense d'énergie de l'entreprise. Dans cette application, des déductions sont faites automatiquement à partir de données environnementales. Pour cela, il est nécessaire de mettre à jour les caractéristiques des machines ou encore des ouvertures de l'usine en fonction des changements des données d'entrée. Le langage SWRL ne permet pas cette mise à jour, car il n'est possible ni de créer des instances, ni de les effacer. Il est seulement possible d'établir de nouveaux liens entre les instances existantes du modèle. Imaginons par exemple qu'il faille implémenter une règle qui change le statut d'une machine de *on* à *off* en fonction de paramètres externes. SWRL pourra attribuer la valeur *on* ou *off* à la machine, mais pas actualiser ou supprimer cette affectation. SPARQL permet, grâce notamment à la fonction CONSTRUCT, de contourner ce problème. Cependant, SPARQL est un langage qui requête des modèles RDF, et non directement sur OWL. Ainsi, lorsque certaines règles sont définies en SWRL et d'autres en SPARQL, leur exécution n'est pas simultanée : il faut d'abord raisonner sur le modèle OWL/SWRL afin d'en extraire un graphe RDF inféré, avant d'appliquer les

requêtes SPARQL. Cette désynchronisation de l'exécution des règles peut être problématique pour des applications industrielles. Les travaux sur SPIN tentent de lever cette limite [Krima et al., 2012].

Enfin, une autre limite d'OWL/SWRL affecte son utilisation dans notre cas d'application : l'absence d'identification des règles SWRL comme des objets du modèle. Les modèles ontologiques trouvent en grande partie leur intérêt dans le fait que tous les objets du modèle (instances, classes, relations) possèdent un identifiant unique (l'IRI). Cela permet de mettre en place des architectures distribuées, puisqu'il est possible de retrouver tout objet, à tout moment, et quelle que soit sa localisation grâce à son unique IRI. C'est un avantage important pour le développement par exemple de bases de données web collaboratives et distribuées. Mais les règles SWRL ne possèdent pas d'IRI. Ainsi, OWL (comme d'autres langages de modélisation) ne propose pas une véritable approche intégrée des règles, permettant leur classification et leur traçabilité en sus de leur exécution. Pourtant, ce besoin se fait sentir dans l'industrie, comme nous l'avons vu dans le cas d'application et il pourrait être un frein important au développement de modèles applicatifs fondés sur les ontologies.

Conclusion de la section 4.2.5

Le cas d'étude sur l'ingénierie nucléaire a permis de mettre en lumière les apports des ontologies dans une démarche de modélisation. Outre l'expressivité et l'inférence, dont la valeur ajoutée a déjà été identifiée dans la littérature, les ontologies permettent d'améliorer la démarche de modélisation, notamment sur deux points :

- les ontologies sont des outils structurés afin de permettre une modélisation distribuée des informations. Alors, en séparant le partage d'information, l'inférence et leur mise en œuvre, il est possible d'améliorer chaque niveau de modélisation, puisque son périmètre s'en voit restreint. L'enjeu est alors de bien délimiter les frontières entre chaque modèle ;
- la décomposition Abox / Tbox permet de réellement penser la Tbox comme un ensemble de concepts expressifs, avant d'envisager l'instanciation du modèle. Cette approche « conceptuelle » de la modélisation permet de définir des modèles plus riches de sens, et surtout, permet de valider la structure ontologique du modèle (notamment par l'évaluation OntoClean).

Cependant, cette mise en application a aussi révélé les limites des ontologies en termes de modélisation. L'Open World Assumption est une hypothèse dont les conséquences affectent à la fois les ontologies de domaines et d'application. L'impossibilité de restreindre le champ du domaine (ou en ayant recours à des raisonnements « par l'absurde ») est un frein à la pleine exploitation de l'inférence pour des applications industrielles. De plus, l'absence d'un langage de règles suffisamment expressif, et même, d'une démarche inté-

grée de gestion des règles remet fortement en cause le potentiel d'OWL/WSRL pour les ontologies d'application.

En plus de ces limites de modélisation, le chapitre suivant, qui valide l'approche par l'implémentation d'un démonstrateur, révèle également des limites techniques à la mise en place de modèles ontologiques pour une application industrielle large.

CHAPITRE 5

L'IMPLÉMENTATION DU CADRE DE MODÉLISATION ET LA VALIDATION DU MODÈLE

Résumé du chapitre

Ce chapitre a pour objectif d'évaluer et de valider les propositions de ces travaux de recherche, qui sont de trois ordres :

- le cadre de modélisation général, séparant les informations partagées (ontologie du cycle de vie), les informations de domaine, sources de connaissance, et la mise en œuvre des informations au niveau applicatif ;
- la méthodologie générale, liée au cadre de modélisation ;
- la formalisation ontologique des trois niveaux de modélisation du cadre.

Evaluer un cadre de modélisation n'est pas chose aisée. En effet, il n'existe pas de critères objectifs et quantitatifs permettant de valider ce type d'approche. De plus, envisager de valider le cadre de modélisation par son application exhaustive (multiple) sur des cas industriels est complexe, car cela impliquerait de disposer d'un grand nombre de cas comparables entre eux. Pour les mêmes raisons, il est difficile d'évaluer une méthodologie autrement qu'en la mettant en œuvre sur un cas d'étude industriel, ce qui reste une validation limitée par les particularités du cas choisi.

Dans la section 5.1, la validation de la modélisation proposée est réalisée au travers de l'implémentation sous forme d'un démonstrateur. Puis, dans la section 5.2, une comparaison entre formalisations UML et OWL est présentée.

5.1 La validation par un démonstrateur

5.1.1 Le cas d'application industriel

L'évaluation des modèles définis précédemment se fait par leur implémentation dans un démonstrateur, dont la validité est elle-même évaluée par l'instanciation d'un cas d'application. Dans le démonstrateur, les considérations d'IHM et de maintenance des données ne sont pas prises en compte. Le cas d'application choisi est la conception du schéma mécanique fonctionnel d'une installation hydraulique. Comme le montre la figure 5.1, cette installation est composée de différentes lignes de tuyauteries, de moto-pompes, de robinets, de valves et d'une bâche.

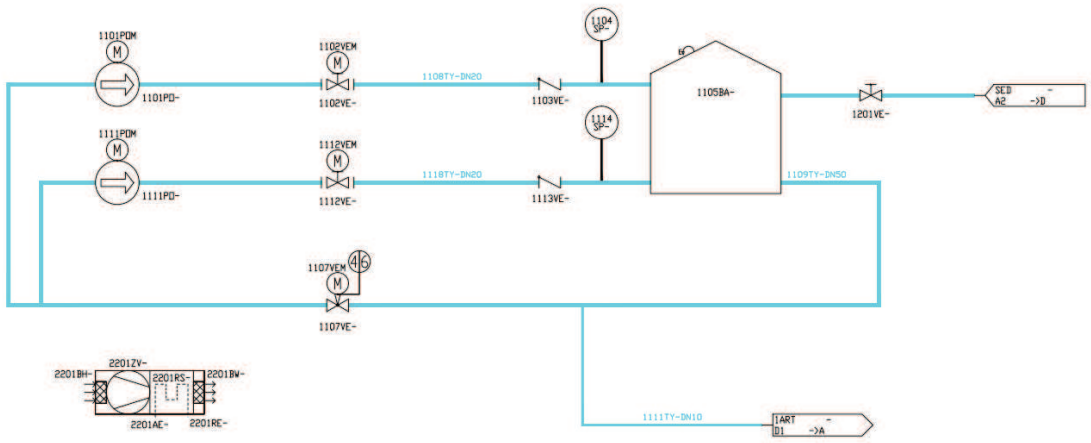


FIGURE 5.1 – Le cas d'étude envisagé : un schéma 2D d'une ligne de tuyauterie

Ce cas a été choisi pour sa simplicité (en termes de quantités et de types d'éléments), mais est d'une complexité quasi-exhaustive (il contient tous les éléments de modélisation qui peuvent poser problème pour l'interopérabilité entre le modèle applicatif et MUDU). L'outil applicatif sur lequel l'étude porte est un outil de schématique 2D, (« SMNGC », surcouche métier développée spécifiquement pour EDF au-dessus d'un noyau AutoCAD) dont la spécification a été fortement inspirée par MUDU. De ce fait, ce modèle est sémantiquement proche de MUDU, même si quelques différences (notables) subsistent, notamment pour des raisons de performance.

La figure 5.2 schématise la situation initiale de modélisation (exposée de façon générique à la section 4.1) relative à l'outil de schématique 2D. Les verrous qui subsistent dans l'approche actuelle sont représentés par les flèches bleues sur la figure et sont les suivants :

- assurer l'interopérabilité entre le modèle applicatif - qui n'est pas complètement conforme à MUDU - et le modèle neutre d'occurrences, lié à MUDU, que l'application doit instancier ;
- valider les occurrences produites, notamment en exécutant les règles métier sur ce modèle ;
- proposer une implémentation robuste (on rappelle que l'implémentation actuelle se fonde sur un langage qui n'est plus développé ni maintenu) ;
- enrichir le modèle MUDU qui ne couvre pas l'ensemble des métiers ni l'ensemble des informations nécessaires. Plus particulièrement, il ne possède pas pour le moment de modèle de règles.

Sur ce cas d'application doivent être exécutées différentes règles métier, dont une partie se trouve en annexe 7.4.

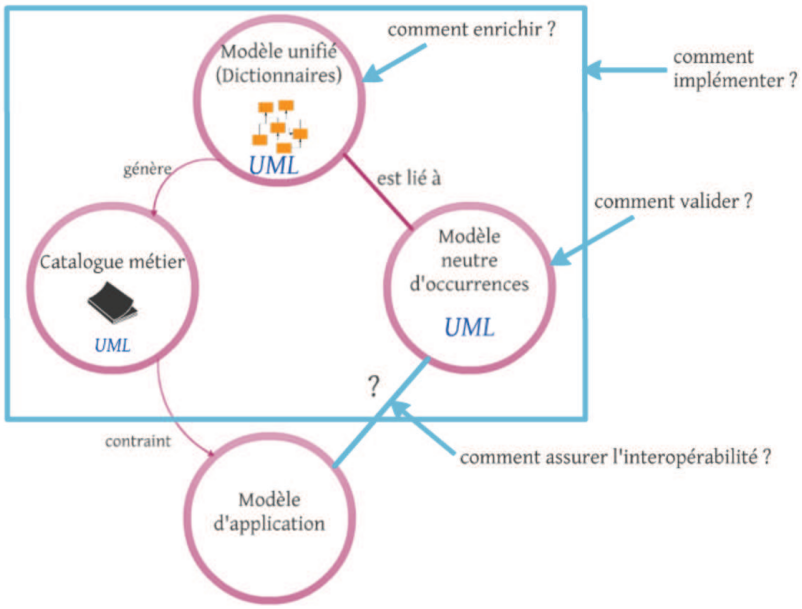


FIGURE 5.2 – La modélisation existante du cas d'étude industriel

Par rapport aux propositions de cette thèse, le démonstrateur doit permettre de valider :

- la modélisation envisagée sur les règles métier ;
- en conséquence du premier point, la démarche de modélisation employée.

Il permet également d'évaluer les potentialités des ontologies face aux contraintes d'implémentation.

Le développement du démonstrateur répond à une problématique plus large que celle exposée sur les règles. En effet, le projet en collaboration avec la DIN d'EDF porte sur

trois problématiques principales :

- la conservation à long terme (et hors de leur format propriétaire) des données de CAO3D ;
- l'interopérabilité entre les applications et MUDU, qu'il s'agisse d'interopérabilité directe, ou indirecte, via le standard ISO 15926 ;
- l'expression et l'exécution des règles métier.

La problématique des règles n'est donc qu'une part du projet, réalisé dans son ensemble par l'équipe « ARTS ». Le schéma d'architecture complet du démonstrateur est visible à l'annexe 7.8.

5.1.2 L'implémentation réalisée

Les choix d'implémentation

Le contexte du développement de ce démonstrateur est particulier : il se fait alors qu'une solution partielle (MUDU) existe et est implémentée. L'existence de MUDU, même si son implémentation doit être repensée, contraint les choix de modélisation et d'implémentation du démonstrateur : chaque changement doit présenter une valeur ajoutée notoire pour être validé. Les changements peuvent être de deux types :

- changements technologiques d'implémentation ;
- changements sémantiques de modélisation.

De plus, la modélisation des règles doit s'intégrer dans un projet plus large, au travers d'une implémentation commune.

C'est sous ce jour particulier qu'a été évaluée la transformation du modèle MUDU existant - modélisé en UML - en une modélisation ontologique. D'un point de vue technique, les outils d'implémentation d'ontologies font face à des lacunes de robustesse et de maturité. Cette composante technique, si elle peut être levée avec le temps, n'en reste pas moins majeure dans un projet de développement. D'autant que les difficultés d'implémentation des ontologies ne sont pas seulement liées à un manque d'efforts de développement par rapport aux technologies UML : OWL possède pas moins de quatre syntaxes (OWL/XML, RDF/XML, OWL functional syntax et Manchester OWL syntax). Cette multiplicité syntaxique freine le développement d'outils, limité également par la performance des moteurs d'inférence.

Du point de vue de la modélisation, une transformation d'UML vers une solution ontologique doit s'examiner de manière globale, sur les trois niveaux de modélisation. Au niveau de l'ontologie du cycle de vie (MUDU), la valeur ajoutée de RDFS par rapport

à UML, sur laquelle nous reviendrons par la suite, n'est pas suffisante comparée à l'effort que représente une re-formalisation complète du modèle, et les faiblesses des outils d'implémentation. Cela s'ajoute aux problèmes de robustesse des raisonneurs OWL2 (pour les ontologies de domaine) et aux limites d'expressivité de OWL/SWRL sur les ontologies d'application. Ce dernier point évacue la possibilité d'une ontologie d'application comme modèle d'occurrences qui, dans ce contexte, ajouterait des transformations de modèles sans donner une satisfaction pleine en termes de vérification de cohérence et de règles comme nous l'avons vu au chapitre précédent.

Pour le développement du démonstrateur, la technologie Java [1] a été choisie pour les raisons suivantes :

- c'est le langage usuellement utilisé dans l'intégration d'entreprise ;
- il existe de nombreuses bibliothèques disponibles, de qualité industrielle, libres et open source ;
- le SGDT choisi (Play) est programmé en java ;
- l'environnement de développement Eclipse Modeling Framework se prête bien aux transformations de modèles.

Ce choix a conduit à conserver la modélisation UML de MUDU. Il doit être précisé ici que l'implémentation - pour sa partie technique - a été réalisée par Roch Bertucat, cofondateur de la société ENGISIS, et Thomas Paviot, chercheur associé à Arts et Métiers ParisTech, membres de l'équipe ARTS.

Pour configurer la base de données java, la plateforme Play, open source, a été utilisée [2]. L'utilisation d'outils open source est une priorité du projet, qui permet, en sus, de personnaliser l'outil aisément. Comme le montre la figure 5.3, Play permet d'accéder à la base de données par une interface par défaut appelée *CRUD*, qui réalise les quatre opérations *create*, *retrieve*, *update* et *delete*, par l'intermédiaire d'une interface dédiée dans un client Web (voir figure 5.4).

En plus de l'interface CRUD et dans l'optique de l'intégration d'applications tierces, une autre interface, appelée « REST » a été spécifiquement développée en extension de Play (voir figure5.5). REST est une architecture pour les services WEB [Fielding, 2000], qui, appliquée au protocole HTTP et aux URI, permet d'interfacer des applications distantes via des services web au travers des quatre opérations *get*, *post*, *put* et *delete*. Une requête « REST » est alors constituée d'une adresse HTTP et de l'URI unique de l'instance.

Cette combinaison URI / http est à la base du web des données [Bizer et al., 2009]. Elle permet, dans le cadre du démonstrateur, d'envisager la création de modules indépendants

1. http ://www.java.com/fr/download/whatis_java.jsp
2. http ://www.playframework.com/documentation/2.0/JavaDatabase

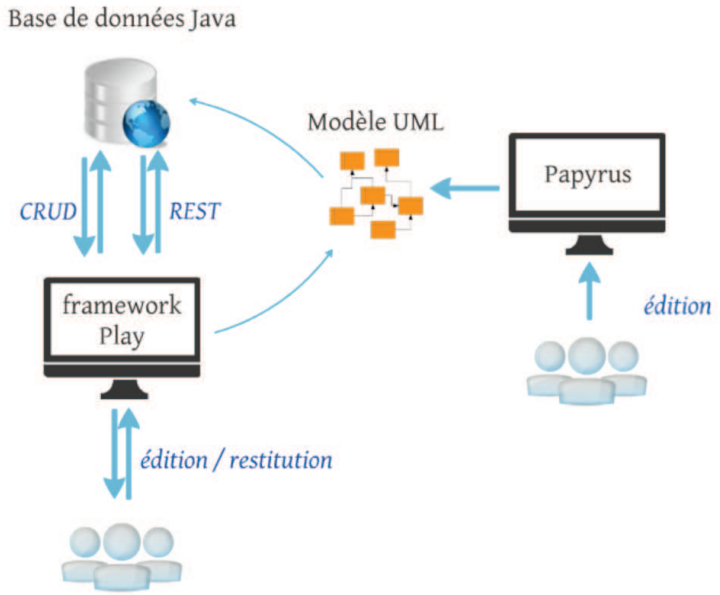


FIGURE 5.3 – Les choix d'implémentation réalisés

FIGURE 5.4 – L'interface « CRUD » de Play

développés sur d'autres technologies qui inter-agissent avec le SGDT par cette interface « standard ». Enfin, la spécification UML du modèle MUDU - qui doit être étendu - se fait au travers de l'outil d'édition Papyrus [3] qui est un projet Eclipse développé par le Centre pour l'Energie Atomique (CEA).

3. http ://www.eclipse.org/papyrus/

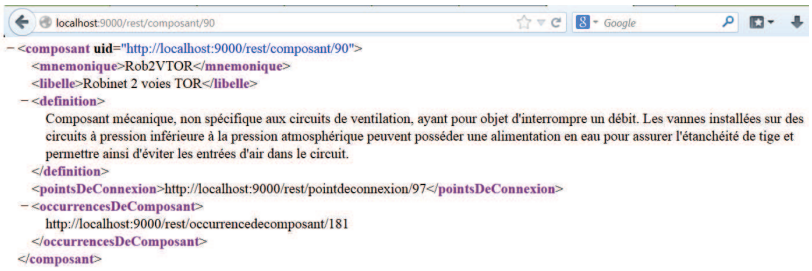


FIGURE 5.5 – La définition « REST » du composant Rob2VTOR

Le modèle UML pour les règles métier

Afin d'intégrer le modèle générique des règles à l'implémentation envisagée, la modélisation de l'ontologie du cycle de vie doit être transformée en modèle UML. En effet, ce modèle de règles est une extension du modèle MUDU existant.

La formalisation UML complète du modèle de règles, dont la figure 5.6 donne les éléments principaux (« l'arête dorsale »), est présentée de manière exhaustive dans l'annexe 7.7. Les différences de modélisation entre l'ontologie du cycle de vie et le modèle UML ne sont pas fondamentales, du fait que l'ontologie du cycle de vie n'utilise pas l'inférence ni les classifications multiples (elle est un réseau RDF, structurée par un schéma RDFS). Cependant, une différence importante, liée à des limites d'implémentation, a conduit à des modifications dans la formalisation : Java ne gère pas l'héritage multiple, et donc, ne peut traiter que des données canoniques.

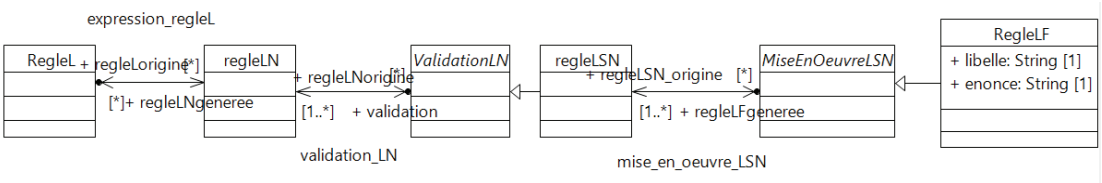


FIGURE 5.6 – « L'arête dorsale » du modèle UML de règles

L'implémentation et l'exécution formelle des règles métier

Comme il a été exposé précédemment, le passage d'une règle LSN (en langage semi-naturel) à une règle LF fait exploser le nombre de règles. Dans ces conditions, il est important d'en faciliter l'expression et la maintenance. Ce constat est d'autant plus vrai dans le contexte de cette application car, pour dépasser le caractère propriétaire du modèle

applicatif, les règles doivent s'exécuter sur un modèle neutre d'occurrences lié à MUDU dont le parcours est complexe. La question de la génération et de l'exécution des règles sur les occurrences restent donc entière.

Pour faciliter l'expression des règles, il est intéressant de masquer et d'automatiser la génération du code formel de la règle à l'utilisateur. Ainsi, l'expertise du langage formel n'est plus nécessaire pour générer les règles LF. Le scénario idéal est que l'expert PLM qui reçoit la règle LSN choisit le mode d'exécution de la règle, le langage applicatif et écrit la règle LSN « telle quelle » dans un outil qui génère la règle formelle LF correspondante de manière automatique. En allant plus loin, il peut même être pertinent de générer automatiquement les règles LF à partir des expressions LSN. Ce point cependant est complexe et non résolu à l'heure actuelle.

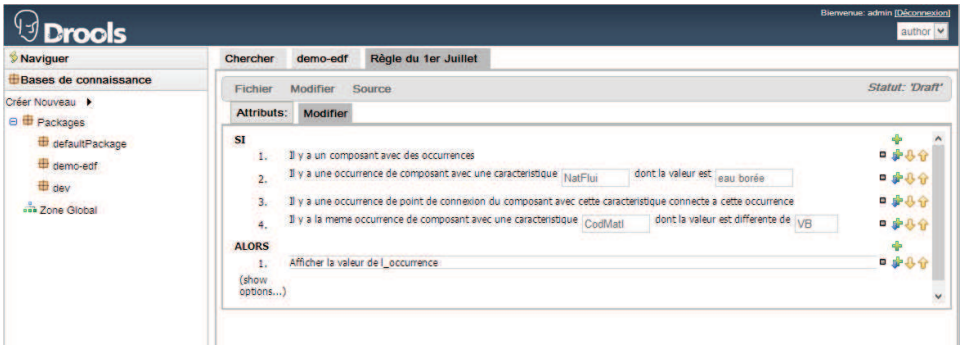


Figure 5.7 – L'outil Drools pour l'édition et l'exécution des règles LF

L'outil Drools, édité par Jboss, va dans ce sens. Comme le montre la figure 5.7, cet outil de gestion des règles permet de définir des fonctions, comme des « macro », qui servent ensuite à l'expression de règles formelles. Ainsi, il permet de créer des éléments de langage naturel qui encapsulent un ensemble de fonctions permettant de générer le code correspondant. Sur la figure 5.7, on voit par exemple qu'a été définie la fonction « occurrence de composant » qui permet de coder automatiquement l'appel à n'importe quelle occurrence de composant dans le modèle (Drools importe le modèle de la base de données au préalable). La règle Drools s'écrit comme une règle de type implication : si...alors, la condition et l'implication étant la concaténation de fonctions en langage naturel. Drools génère le code dans son propre langage mais puisqu'il s'agit d'un outil open source, il est envisageable de l'étendre et de le paramétrer afin de générer des expressions LF dans des langages différents. Surtout, grâce à l'import des données de la base, Drools peut exécuter les règles directement sur les occurrences. Un test sur la règle « si le fluide qui traverse le robinet est de type *eau borée* alors le code matériel du robinet vaut *VB* » a permis de confirmer ce potentiel d'exécution des règles par Drools.

Du point de vue de la modélisation, et pour faciliter la maintenance, la règle LF est définie comme un élément de la mise en œuvre d'une règle LSN, et non comme la fille directe de cette règle. Le fait de passer par un modèle de mise en œuvre, qui précise d'autres éléments - comme l'application concernée et la façon d'exécuter la règle - permet de faire évoluer l'expression de la règle si les conditions de sa mise en œuvre évoluent. L'intégration de Drools dans le démonstrateur est illustrée à l'annexe 7.8.

L'implémentation des modèles métier

La figure 5.8 synthétise la solution implémentée dans le démonstrateur. On y voit notamment que le modèle métier n'est pas encore pris en charge par la plateforme play (le périmètre de la plateforme est délimité par le cadre jaune). En effet, la construction et l'implémentation de ce modèle métier pose les limites d'une modélisation UML et il est envisagé de maintenir la modélisation ontologique de ce modèle. Dès lors, l'articulation avec la plateforme de technologie Java doit être pensée et n'est pas encore finalisée.

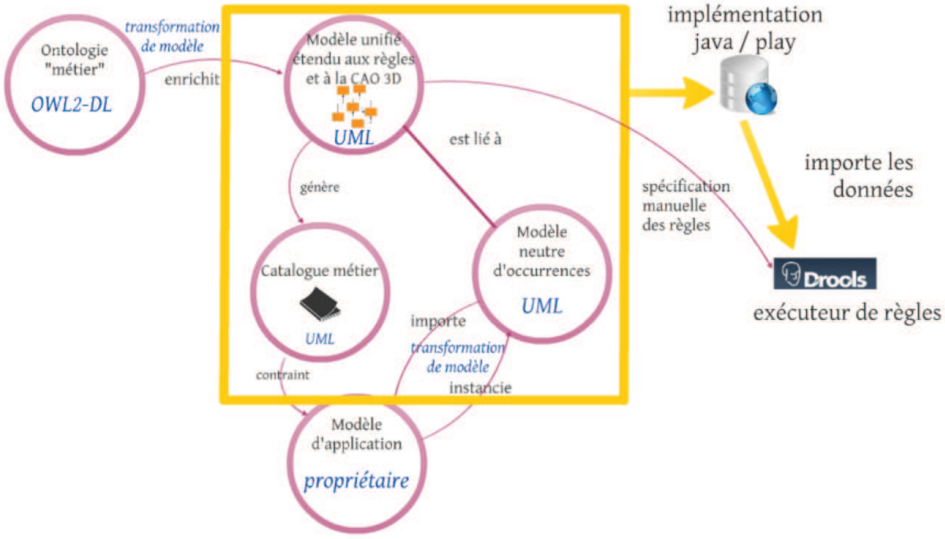


FIGURE 5.8 – La modélisation et l'implémentation réalisées

Dans l'état actuel des choses, l'instanciation des triplets métier et des mappings métier/MUDU dans l'ontologie du cycle de vie est donc manuelle. Par conséquent, elle se limite au cas d'application envisagé.

Conclusion de la section 5.1

Le démonstrateur a permis d'intégrer à MUDU une véritable modélisation des règles.

Le lien avec l'ontologie métier, ainsi que les mappings métier/MUDU sont, sur le cas d'étude, manuels. Malgré ces quelques interventions manuelles, le démonstrateur permet d'exprimer et d'exécuter des règles métier depuis leur expression textuelle jusqu'à leur exécution formelle. Ainsi, l'implémentation permet de valider la modélisation multi-niveaux des règles ainsi que la démarche employée. Il reste maintenant à automatiser l'ensemble des transformations à développer l'IHM.

La confrontation de la modélisation des règles (décrite au chapitre 4) aux contraintes d'implémentation ainsi qu'aux exigences du projet PLM global ont conduit à des changements, notamment sur la formalisation de l'extension de MUDU. Le langage UML a été conservé, parce qu'il est le meilleur compromis entre expressivité et facilité d'implémentation. Cette transformation en UML nous permet, dans la section suivante, de comparer les démarches de modélisation UML et OWL/RDF, et plus généralement, de déterminer la place des ontologies dans un projet PLM global.

5.2 La place des ontologies dans l'implémentation : une comparaison UML/OWL

Pour les raisons exprimées précédemment, le choix de modélisation UML de MUDU a été conservé. Afin d'étendre MUDU aux règles, une transformation des ontologies de règles en UML est donc nécessaire. Cela permet alors de dresser un bilan comparatif entre modélisation ontologique et modélisation UML. Ce bilan, sans être exhaustif, porte sur les trois niveaux de modélisation : application, domaine et méta-modèle.

5.2.1 L'indispensable inférence des modèles « métier »

Le modèle métier, qui permet la génération des triplets sur lesquels repose l'expression des règles, n'est pas encore intégré au démonstrateur. Cela s'explique par le fait que le démonstrateur est pour le moment fondé sur une modélisation UML, tandis que le modèle métier a été défini suivant le langage OWL2-DL. Après une tentative de transformation du modèle métier en UML - tentative dont le bien fondé nait de l'ambition de ne pas multiplier les langages de modélisation et par conséquent de ne pas complexifier l'implémentation - il apparaît que les apports des modèles ontologiques pour ce type de modélisation sont indéniables, et qu'un passage à UML fait perdre beaucoup, tant en termes d'expressivité, de conceptualisation que d'inférence. Pour les modèles de domaine, qui modélisent une information source de connaissance, les ontologies présentent en effet des avantages importants, notamment :

- l'expressivité et surtout la gestion de données non-canoniques permet une définition aisée d'un modèle de données muti-vues, nécessairement complexe ;
- le raisonnement et l'automatisation partielle induite par l'inférence constituent un gain en temps et en engagement de formalisation qui n'est pas négligeable.

Dans le cas d'étude, le modèle métier proposé exploite peu l'expressivité d'OWL2. Mais cela est du au fait qu'il est restreint pour le moment à un périmètre d'étude très limité. Il est certain que l'agrandissement de ce périmètre conduira à la définition de nouvelles classes utilisant pleinement l'expressivité d'OWL2. De plus, même sur son champ limité, le modèle actuel exploite l'inférence ontologique qu'il est difficile voire impossible de reproduire dans le cas d'une formalisation UML.

La transformation de modèles qu'implique l'utilisation d'une ontologie de domaine sur la base d'un méta-modèle UML doit être pensée et ses conséquences évaluées. Seule une analyse profonde du rapport bénéfices / risques d'une telle solution « hybride » peut permettre de trancher définitivement cette décision. Cette analyse doit intégrer les problématiques d'implémentation et de maintenance des échanges d'information - et non, seulement la problématique de modélisation. Les travaux de recherche sont en cours sur ce point, mais semblent s'acheminer sur le maintien de l'utilisation des ontologies pour les modèles de domaine. Ce choix est facilité par le développement de l'interface REST qui par sa généricité permet de « personnaliser » les différents modules du démonstrateur. Par exemple, sur la question de la conservation à long terme des données de CAO 3D - qui est partie-prenante de la problématique PLM globale du projet - le développement d'un module spécifique sur une base python a pu être réalisé, bien que la base de données repose sur une implémentation java, ce grâce à la flexibilité de l'interface REST.

En revanche, l'OWA reste une limite à la représentation de modèles de domaine avec OWL2. Dans le cas d'étude, pour contourner l'absence de « punning » une approche privilégiant le niveau instance a été proposée. Cependant, ce n'est pas le seul choix possible. Une seconde approche consiste à privilégier le niveau de classes, définies à partir d'axiomes OWL. Alors, l'expression de chaque règle devient une instanciation du modèle de classes. Cette deuxième option optimise les capacités d'inférence des ontologies et donc l'automatisation de sa définition.

On peut illustrer ces deux options, qui sont fondamentalement différentes, par un exemple volontairement restreint, illustré sur la figure 5.9 : un composant de type robinet est traversé par un fluide ; ce fluide a pour attribut une température, pouvant prendre différentes valeurs réelles. Nous pouvons définir les classes **robinet**, **composant** et **fluide**, ainsi que la propriété *est_traverse_par*. L'axiome « **robinet** ≡ *est_traverse_par* only **fluide** » permet de spécifier qu'un robinet est traversé par un fluide. Cependant, rien n'empêche que le robinet `r1`, instance de la classe **robinet**, ait d'autres relations que celle-ci. En

effet, l'Open World Assumption implique que les axiomes OWL définissent les conditions nécessaires et suffisantes pour appartenir à une classe. En revanche, l'instanciation de ces classes n'est pas limitée aux axiomes prédéfinis. Ainsi, le modèle de classes ne réduit pas le champ des possibles et autorise des instanciations qui ne sont pas explicitement prévues. Cet exemple illustre une différence d'ordre conceptuel entre une formalisation UML : avec UML, on spécifie tout ce qui est possible, alors qu'avec OWL, il faut spécifier ce qui n'est pas possible, ce qui peut, dans des applications industrielles, rapidement s'assimiler à l'infini. En effet dans un modèle OWL toute instanciation est autorisée tant qu'elle n'est pas explicitement interdite. Ce « raisonnement par l'absurde » auquel une formalisation ontologique contraint peut être difficile à mettre en place, voire, dans certains cas, inextricable.

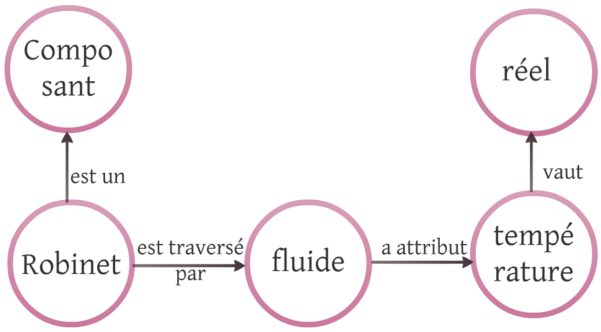


Figure 5.9 – Exemple de conceptualisation métier

Dans le cas d'étude, le modèle métier doit répondre au critère de restriction de l'expression des règles : il est important de restreindre l'édition des triplets. Alors, il est nécessaire de formaliser les concepts métier comme des instances : seules les instances et propriétés explicitement définies, ou obtenues par un raisonnement maitrisé, seront autorisées. Tout ce qui n'aura pas été instancié sera considéré comme impossible pour l'éditeur de règles. Cependant, ce choix restreint fortement le champ de raisonnement et d'inférence possible. Ainsi, l'OWA et l'absence de réel « punning » limitent fortement l'utilisation de l'inférence sur des ontologies de domaine.

5.2.2 La conceptualisation ontologique face à la formalisation UML

Dans le cas d'application, la transformation nécessaire de l'ontologie du cycle de vie en modèle UML a montré qu'UML a une expressivité suffisante, et tous les éléments initialement formalisés en RDFS ont pu l'être en UML. Cependant, cette re-formalisation du

modèle de règles a permis de révéler des différences dans les démarches de modélisation.

La démarche ontologique est fortement dirigée par une approche « conceptuelle » du modèle. L'ensemble des concepts sont définis, leurs liens taxonomiques évalués. Ensuite seulement sont envisagées les relations possibles et enfin, l'instanciation qui en sera faite. La démarche UML est différente : les classes UML ne représentent pas forcément des « concepts », surtout quand leur création n'a pour objectif que de contourner le problème de canonicité des données. De plus, les relations n'ont pas d'existence propre en UML. Elles appartiennent à une classe, et il ne peut être défini une même relation entre plusieurs classes différentes. Ainsi, les relations n'ont pas de valeur ontologique propre en UML.

Le modèle UML de la figure 5.6 en est une illustration : la mise en œuvre d'une règle LSN y est définie comme une généralisation des règles LF. Cela représente le fait qu'il y a plusieurs façons de mettre en œuvre formellement une règle LSN, et que la règle LF est un des moyens possibles. Cependant, comme il a été montré par l'analyse OntoClean de l'ontologie des règles, conceptuellement, cette représentation n'est pas juste. Si une règle LF peut être vue - du point de vue de la modélisation des règles - comme un moyen de mise en œuvre formelle d'une règle métier, elle n'est pas uniquement cela si on la regarde d'un point de vue plus général : si un lien existe entre les deux, il ne saurait être un lien taxonomique direct. Ce « raccourci » sémantique est nécessaire car UML ne gère pas les données non-canoniques. Du point du résultat d'implémentation, il n'y a pas de différence notoire, en revanche, du point de vue de la démarche de modélisation, la différence est réelle.

5.2.3 L'apport de RDF et du web des données pour la gestion des occurrences

Il est un apport de la modélisation ontologique qui peut être conservé : la distributivité des modèles et des bases de données. L'approche de modélisation ontologique proposée dans cette thèse permet notamment une gestion séparée du modèle partagé et des occurrences (les instances des ontologies d'application). A l'heure actuelle, la problématique d'interopérabilité entre les applications et le modèle MUDU a conduit à la définition d'un modèle neutre d'occurrences, cohérent avec MUDU, et instancié par des mappings avec les outils applicatifs. Les liens entre le modèle neutre d'occurrences et MUDU ont conduit à une modélisation et une implémentation liées de ces deux modèles. Pourtant, ils représentent des niveaux foncièrement différents.

Pour des raisons à la fois conceptuelles et techniques, un traitement disjoint de ces deux niveaux de modélisation est pertinent :

- l'évolutivité du modèle MUDU et des occurrences n'est pas la même : un ensemble d'occurrences doit être conforme à une « version » de MUDU ;
- les acteurs et outils qui instancient les deux modèles sont différents ;
- d'un point de vue technique le nombre d'occurrences peut poser des problèmes d'échelle (une centrale nucléaire est composée d'environ un milliard de composants). Il convient de ne pas impacter MUDU par ces problèmes techniques.

Une modélisation et une architecture distribuées sont possibles en utilisant le langage RDF. La figure 5.10 présente une évolution possible de l'implémentation actuelle vers une telle solution distribuée. Le modèle d'occurrences, sorti du cadre de la base de données MUDU, est alors structuré comme un graphe RDF, toujours à l'aide de mappings avec l'outil applicatif. Ces éléments RDF pointent alors vers les instances de MUDU, ou plutôt, vers les instances d'une version figée de MUDU. Alors, MUDU peut continuer d'évoluer sans impacter la cohérence des occurrences. En plus d'une solution distribuée, RDF apporte une légèreté qui est importante vue la quantité de données à stocker.

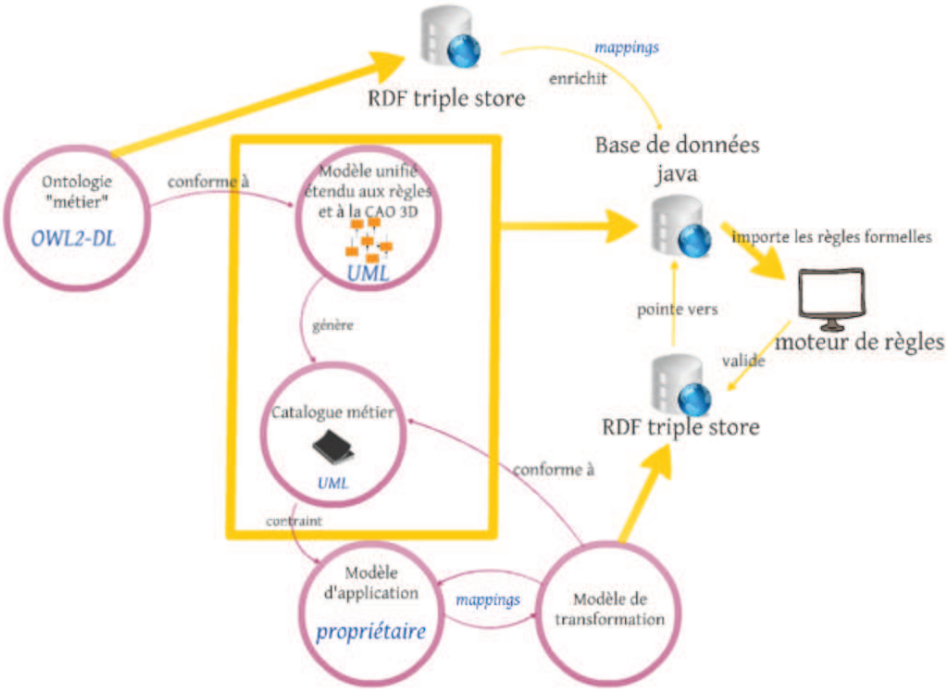


Figure 5.10 – Les extensions possibles de l'implémentation réalisée

Dans ce contexte, l'ontologie métier pourrait également être stockée sous forme d'un magasin RDF et instancier MUDU via l'interface REST. Enfin, le moteur de règles

autonome importerait automatiquement les règles spécifiées dans MUDU pour les exécuter sur le réseau RDF d'occurrences, réalisant alors une véritable gestion séparée des règles.

Conclusion de la section 5.2

L'apport des ontologies en terme de modélisation se situe donc à deux niveaux :

- une formalisation *conceptuelle* du modèle, malgré les limites engendrées par l'open world assumption ;
- une formalisation *intelligente* des modèles métier, grâce à l'inférence.

Cependant, les contraintes d'implémentation ont remis en cause le cadre de modélisation envisagé. Notamment, l'inférence n'est pas assez robuste à l'heure actuelle pour permettre une application industrielle à grande échelle. Et même au-delà des problématiques techniques, l'OWA est encore une forte limite à l'utilisation de OWL pour les ontologies d'application. Cependant, les ontologies peuvent trouver leur place dans une démarche de modélisation PLM des informations. La distributivité et la flexibilité de RDF permettent de dépasser les limites d'échelle et de séparer les niveaux de modélisation. De plus, pour les modèles de domaine, particulièrement les modèles métier qui représentent des informations sources de connaissance, l'inférence ontologique peut tout de même être exploitée en tenant compte des limites qui ont été énoncées dans ces travaux. Alors, une solution « hybride », exploitant au mieux les avantages des différents types de modélisation est envisageable.

CHAPITRE 6

LA CONCLUSION ET LES PERSPECTIVES DE RECHERCHE

6.1 La conclusion générale

Le titre de cette thèse est « Contribution à une modélisation ontologique des informations au long du cycle de vie du produit ». Dans ces travaux, nous nous sommes efforcés à chercher comment modéliser les informations du cycle de vie. Le premier objectif de la modélisation sémantique est de proposer une représentation juste des informations. Mais ce n'est pas le seul. Il faut pouvoir échanger ces informations entre acteurs du cycle de vie, en l'occurrence, entre systèmes d'information. C'est en effet par l'échange que l'information vit. Pour réaliser cet échange, il pourrait suffire de transmettre les instances de concepts et leurs relations à l'utilisateur concerné. Enfin presque : encore faut-il que le receveur et l'envoyeur de l'information s'accordent sur la syntaxe des données qui sont transmises. C'est le problème auxquels les travaux sur l'interopérabilité ont tenté de répondre jusqu'à présent, notamment en proposant des modèles standards d'unification. En revanche, il n'est pas encore transmis via ces standards *comment* interpréter l'information : pour qu'il y ait un échange sémantique bijectif, sans perte de sens, il faut que les deux systèmes (ou plus) s'accordent sur la compréhension qu'ils ont de l'information échangée. La solution envisagée est alors de partager une sémantique commune, qui contextualise l'information afin d'en guider l'interprétation : c'est ainsi l'information source de connaissance qui est transmise, et non plus seulement les données et leur syntaxe.

Dans cette thèse, nous proposons un cadre de modélisation qui réponde aux enjeux suivants : modéliser une information riche, l'échanger sans perte entre systèmes, et enfin, définir une sémantique partagée afin d'interpréter les informations de façon homogène.

Ce cadre repose alors sur trois niveaux de modélisation : des ontologies de domaine exploitant l'expressivité et l'inférence des langages DL pour formaliser une information riche de manière semi-automatique ; un modèle pivot, appelé « ontologie du cycle de vie » pour partager une sémantique entre utilisateurs : les informations de chaque phase et chaque SI y sont représentées suivant le même méta-modèle. Cette ontologie du cycle de vie permet par la suite de définir des modèles applicatifs dans une approche descendante, qui garantit l'échange syntaxiquement et sémantiquement correct des informations entre applications, puisque les modèles eux-mêmes sont syntaxiquement et sémantiquement cohérents. Une spécialisation de ces ontologies de domaine conduit enfin à l'adaptation exacte au besoin utilisateur.

Cette approche soulève cependant des difficultés nouvelles. Tout d'abord, il est nécessaire que l'ensemble des acteurs s'accordent sur une sémantique commune. Cet accord est d'autant plus difficile à réaliser que la sémantique commune définie est large et riche. Ensuite, il faut réussir à formaliser la sémantique, c'est-à-dire, ajouter de la logique et de l'inférence dans le modèle, sans générer d'erreurs ni faire perdre sa généricité au modèle pivot : l'enjeu est de bien placer le curseur entre représentation des informations partagées et édition, interprétation et exploitation de ces informations par l'inférence. Enfin, il faut instancier les modèles applicatifs, définis en cohérence avec l'ontologie du cycle de vie, alors même que dans la majorité des cas, le SI utilisateur est fondé sur son propre modèle, souvent commercial - donc propriétaire - et souvent moins riche que le modèle unifié : une distance sémantique s'instaure et doit être comblée.

Dans ce cadre de modélisation, les ontologies permettent de lever certaines de ces difficultés. L'expressivité ontologique, fondée de plus sur des concepts porteurs de sens, constitue un apport dans la formalisation des informations partagées, et surtout, dans l'examen et la validation sémantique du modèle. De plus, l'inférence permet de proposer des modèles de domaine dédiés « intelligents », qui vérifient la cohérence des données métier selon des schémas pré-établis et permettent de déduire des informations implicites. Cependant, quelques difficultés subsistent, notamment au niveau applicatif. Les ontologies ne permettent pas de mettre en œuvre toute la sémantique nécessaire, particulièrement en ce qui concerne les règles. De plus, elles n'apportent pas de solution supplémentaire au problème de conformité aux modèles applicatifs propriétaires. Enfin, le manque de robustesse des outils d'implémentation en limite l'exploitation à grande échelle. Cette dernière difficulté est cependant temporaire, si les efforts de la communauté dans le développement d'outil d'implémentation se poursuivent. Ainsi, aux antipodes de la mouvance actuelle, qui tend à privilégier et à développer les modèles OWL, l'application industrielle développée dans cette thèse tend à démontrer que le langage RDF(S) présentent des capacités qui, faces aux problématiques de performance d'OWL, restent très valables.

Il est important de noter que ces travaux ne prétendent pas proposer une méthode d'ingénierie robuste et éprouvée : il s'agit de l'expression, comme une photographie à un instant donné, de travaux et de réflexions de recherche. Si un cas industriel - ayant conduit au développement d'un démonstrateur - est ici largement décrit, c'est d'abord par souci didactique, gageant que l'application est un moyen performant de faire comprendre des idées. C'est aussi parce que la confrontation au terrain que cette application a impliquée a permis d'éclairer les propositions de cette thèse, établies à la lumière également de lectures, de discussions avec des chercheurs de différentes communautés et de réflexions personnelles. Tous ces éclairages ont permis de dessiner un chemin, formalisé par ce manuscrit. Cependant, il n'est pas certain que cela soit l'unique chemin. Au cours de ces réflexions et de ces recherches ont également émergé des obstacles, dont nous avons discuté dans ce document, et des zones d'ombre qu'il convient, en guise de perspectives, d'explorer. Nous envisageons les perspectives de recherche selon deux points de vue. Dans un premier temps, nous exposons les perspectives de recherche liées à l'application à l'ingénierie nucléaire, et dans un second temps, nous proposons des perspectives de recherche sur la modélisation ontologique et la modélisation des informations en général.

6.2 Les perspectives de recherche relatives au cas d'application

6.2.1 Les méthodes de traitement automatique de corpus linguistiques

Dans le cas d'étude industriel, nous avons défini une ontologie métier partielle à partir d'un corpus réduit de documents. La restriction du périmètre d'étude a permis de conceptualiser le modèle métier de manière empirique, par l'analyse manuelle des documents. Cependant, la conceptualisation de l'ensemble du domaine métier est une tâche de grande envergure, particulièrement dans le domaine de l'ingénierie nucléaire. Afin d'aider les concepteurs dans cette tâche, les technologies de traitement automatique de corpus documentaires peuvent être envisagées.

L'idée n'est pas de remplacer l'activité de conception du modèle par une construction automatique de l'ontologie à partir de l'analyse de documents : cela conduirait certainement à des imprécisions, l'analyse automatique ne pouvant être que syntaxique (fondée sur les termes), et n'incluant donc pas la dimension sémantique des concepts. En revanche, ces méthodes peuvent faciliter le travail de récupération et de catégorisation des termes, qui est long et fastidieux, afin de focaliser le travail du concepteur sur une tâche à haute valeur ajoutée : la construction du réseau de concepts et sa formalisation ontologique.

Il est à noter cependant que ces d'approches présentent des limites, synthétisées par

[Ottens, 2007], et doivent donc être employées en connaissance de cause :

- les ontologies obtenues sont très dépendantes du corpus linguistique choisi. Par conséquent, elles ne sont donc ni complètes, ni raffinées. Ce sont des « brouillons d'ontologie » ;
- les ontologies obtenues ne sont révélatrices que de la connaissance explicite, puisque le traitement automatique ne permet pas d'extraire les connaissances implicites comme un être humain pourrait le faire ;
- l'intervention d'un expert métier pour critiquer le résultat obtenu et identifier les erreurs est indispensable ;
- la maintenance des liens entre le corpus textuel (qui évolue) et l'ontologie est difficile à réaliser.

Nous pouvons ajouter à ces limites que le traitement automatique linguistique implique également que le contenu du corpus de texte soit juste, cohérent et à jour, puisqu'aucune pondération ou analyse critique de son contenu n'est possible. Cette hypothèse, au vu de l'application dans l'ingénierie nucléaire, nous paraît forte.

6.2.2 L'évaluation sémantique du contenu de l'ontologie métier et son évaluation

A l'heure de l'évaluation du contenu sémantique de l'ontologie métier, nous sommes restés penauds devant les méthodes d'évaluation à notre disposition. Outre la méthode OntoClean [Guarino and Welty, 2009] qui a permis une réflexion sur les concepts formalisés par l'ontologie, les méthodologies existantes ne proposent qu'une suite d'étapes génériques, et aucune approche, hormis quelques techniques de traitement syntaxique fondées sur l'analyse de termes, ne permet de valider la sémantique du modèle. L'intervention d'un expert métier est toujours requise, avec tout ce que cela implique d'imperfection, de subjectivité, et de dépense en temps. Pourtant, cette évaluation est indispensable. Dans notre application par exemple, la pertinence du modèle métier détermine l'expression des règles métier, dont dépend également leur cohérence et a fortiori leur exécution formelle. De plus, assurer la pertinence et la justesse du modèle par une évaluation de son contenu et de sa formalisation est nécessaire pour garantir l'évolution du modèle, condition indispensable à la conception des modèles de domaine notamment : ceux-ci doivent évoluer en même temps que l'environnement qu'ils modélisent.

6.2.3 Des règles aux exigences et aux processus : vers une ingénierie système

Le travail sur les règles métiers proposé dans cette thèse peut (et doit) être étendu du point de vue métier à la notion d'exigence, et aux problématiques d'ingénierie des

exigences. La différence conceptuelle entre règle et exigence n'est d'ailleurs pas claire, la tendance étant ces dernières années à placer sous le terme « exigence » des objets qui s'intitulaient auparavant « règle ». Malgré cette confusion sémantique, qu'il faudra lever, il existe tout de même des problématiques différentes quoique connexes entre gestion et traçabilité des règles et ingénierie des exigences. On sent entre ces deux notions des différences concernant l'impériosité et la généricité de la contrainte exprimée, allant jusqu'à envisager que la règle peut être l'application générique d'une exigence particulière liée à un fonctionnement ou un contexte donné.

En effet, la notion d'exigence étend celle des règles au moins du fait qu'elle y associe un contexte, qui est d'ordre fonctionnel, environnemental ou procédural. La règle métier, générique et imposée, serait la conséquence de l'analyse d'une exigence. Quoiqu'il en soit, la traçabilité des exigences et leur validation formelle sur des données numériques est un aspect indispensable d'une démarche PLM : cela permet d'intégrer la connaissance métier dans la modélisation et l'exploitation des informations.

Alors, l'ingénierie des exigences établit un pont entre PLM et ingénierie système. Les aspects fonctionnels, environnementaux et procéduraux pris en charge par l'ingénierie système produisent un ensemble d'exigences qui, si elles sont modélisées et tracées impactent les informations produit. A l'inverse, l'exploitation de ces informations métier peut être un moyen de valider la prise en compte d'exigences. L'ingénierie des exigences dans ce contexte est alors le moyen qui permet d'intégrer PLM et ingénierie système.

Dans ce domaine, il existe des normes et des langages qu'il convient d'étudier, et dont l'applicabilité et la pertinence doivent être évaluées. Plus particulièrement, le protocole d'application AP 233 de la norme STEP envisage la modélisation des exigences et des fonctionnalités d'un produit, tandis que le modèle SysML propose un modèle de relation entre exigences qui inclut la notion de fonction.

Enfin, en intégrant l'ingénierie système dans une démarche PLM, on apporte une vision « processus » à la modélisation des informations, que nous avons jusqu'à présent considérée de façon plus statique. Or, il paraît important pour enrichir la sémantique des modèles et augmenter leur champ d'application, d'appuyer les démarches d'intégration des connaissances sur des processus (voir [Chen et al., 2009]).

6.3 Les perspectives de recherche sur la modélisation ontologique des informations

6.3.1 Les limites d'implémentation des ontologies sont-elles insurmontables ?

Lors de diverses conférences ou colloques, durant lesquels nous avons exprimé nos réserves quant à la possibilité d'implémenter à grande échelle une solution fondée sur les ontologies, ils nous a été opposé certains arguments légitimes, dont nous souhaiterions traiter ici.

OWL2 n'est pas l'unique moyen de représenter la connaissance

L'une de nos hypothèses de travail est que les ontologies de domaine ne sont « optimales » que si elles exploitent le paradigme de modélisation OWL2, qui élargi l'expressivité de OWL(1), et surtout, par la fonction de « punning » permettant de modéliser un concept à la fois comme une classe et une instance. Alors une véritable modélisation multi-niveaux est envisageable, car à une taxonomie de classes s'ajoute l'instanciation des relations à chaque niveau taxonomique. Or, nous l'avons vu, cette fonction n'est pas encore réalisée par les raisonneurs et le langage OWL2 n'est pas pleinement décidable. Seuls des « profils », restrictions de la sémantique d'OWL2 face aux contraintes d'implémentation, sont aujourd'hui exploitables.

Une telle représentation multi-niveaux (également appelée « méta-modèle ») est possible avec d'autres langages que OWL2. En effet, RDFS permet d'instancier des relations entre tout niveau de concepts, et d'exprimer en parallèle une taxonomie entre ces concepts. Idem pour OWL-full, qui est la concaténation de OWL-DL et de RDF. Le modèle d'ontologie d'entreprise proposé dans cette thèse pourrait donc tout-à-fait être formalisé par ces langages ontologiques. Il faut cependant rappeler une limite importante des langages RDFS et OWL-Full : leur absence de décidabilité. En d'autres termes, ils ne permettent pas de raisonner. Or le raisonnement est une des raisons majeures du choix des ontologies dans notre approche. Il est possible alors d'argumenter qu'aucune autre approche de modélisation ne permet de raisonner sur ce type de représentation. Cela est juste, et explique sûrement que des solutions RDFS / OWL-Full existent tout de même dans la littérature pour des applications de modélisation de la connaissance. Cependant, dans notre étude, nous nous positionnons dans un contexte particulier, et qui nous semble être représentatif de nombreux cas d'application : un modèle d'information existe déjà, et celui-ci est modélisé en UML. Alors, une question très importante se pose : quelle est la valeur ajoutée de RDFS / OWL-Full qui pourrait justifier une transformation de modèle par rapport à UML ?

A la lumière en sus des contraintes d'implémentation, nous n'avons pu trouver de réponse convaincante à cette question.

Il n'existe aucun langage qui identifie les règles comme des objets

L'une des limites à l'utilisation de SWRL ou SPARQL comme langage de règles que nous avons soulevée est que ces langages, pourtant adossés à des modèles ontologiques, ne permettent pas d'identifier les règles formelles par des URI (identifiants des instances d'ontologies), et donc, ne permettent pas d'assurer une traçabilité aisée de ces règles vis-à-vis des exigences dont elles découlent, ce qui a également comme conséquence de complexifier leur maintenance. Il est vrai que cette identification des règles n'est pas non plus possible avec OCL, ni avec aucun des langages de règles que nous avons envisagés. Ce point nous parait être une limite importante, car les attentes industrielles sur la traçabilité, la maintenance et l'automatisation des règles sont grandes. Les travaux sur SPIN en la matière méritent d'être suivis, car ils pourraient apporter un début de réponse à ce problème.

Dans ce contexte, envisager l'exécution des règles « à part » du modèle, comme proposé au travers de l'outil Drools paraît être un compromis acceptable. Si Drools ne règle pas le problème de modélisation des règles formelles du moins, le fait que l'outil permette - sur n'importe quel modèle - une exécution autonome et séparée des règles est un apport en termes de traçabilité et d'évolution du corpus de règles.

L'OWA peut être contournée

Nous avons pointé dans cette thèse l'Open World Assumption comme un frein à l'implémentation de modèles ontologiques pour l'industrie, notamment pour ce qui concerne le niveau applicatif. Les ontologies d'application doivent en effet « mettre en œuvre » la sémantique du méta-modèle et exécuter les règles sur les instances réelles. Pour cela, il est nécessaire d'avoir un modèle à sémantique riche, mais aussi un langage de règle intégré qui permette d'exprimer différents types de règles, et notamment, les règles par la négative. Il faut également pouvoir vérifier la cohérence des informations instanciées. Or l'OWA empêche de bien gérer la négation, si bien que ces règles sont rarement exprimables et que la vérification de la cohérence est complexifiée, puisqu'elle doit se faire par « l'absurde », comme nous l'avons évoqué dans la section 5.2.

Il existe cependant des moyens de « fermer » le modèle, donc de contourner l'OWA. Nous avons relevé trois moyens différents :

– utiliser les classes énumérées ;
– définir des propriétés ou des classes à sémantique négative ;
– utiliser un raisonneur CWA.

Les classes énumérées sont un moyen de « fermer localement le monde ». En effet, le principe d'une classe énumérée est de définir les instances de cette classe comme une liste fermée : il ne peut y avoir dans la classe énumérée que les instances que l'on a définies a priori. Alors, il est possible de requêter par la négative sur cette classe, et de retrouver les instances qui n'en font pas partie (ce qui est normalement impossible avec l'OWA). En revanche, cette solution est extrêmement limitée, car, une fois définie comme énumérée, il n'est plus possible de rajouter des instances dans la classe. Cela n'est donc applicable que lorsque l'on connaît toutes les instances a priori (par exemple, dans le cas d'un catalogue fini).

La solution la plus flexible est donc d'exprimer la négation au sein même des classes et des relations. Ensuite, on peut raisonner par la négative en appliquant un raisonnement ouvert sur ces classes et propriétés négatives. OWL2 permet d'ailleurs de nier explicitement l'existence d'une relation entre instances (*negative object property*). La rigueur sémantique d'un tel procédé ainsi que la maintenance du modèle ainsi défini peuvent cependant être remises en cause. En termes de rigueur, le modèle se retrouve alors « bricolé » afin de pallier à la limite de la syntaxe. Il ne représente plus uniquement la vérité des informations : on perd le sens métier. De plus, cela peut conduire à des dérives inextricables, qui font grossir artificiellement le nombre de classes et de propriétés simplement pour contourner l'OWA. Ces classes et propriétés ne sont alors plus intrinsèques au modèle, mais des éléments d'exploitation de l'information. La maintenance d'un tel modèle s'en retrouve complexifiée.

Enfin, des travaux sont en cours pour proposer un raisonneur CWA pour les ontologies. Il ne s'agit cependant plus d'une fermeture locale du monde, mais d'un choix à effectuer a priori entre OWA et CWA. Nous discutons de la difficulté de ce choix ci-après.

6.3.2 Les systèmes industriels : des mondes fermés ou ouverts ?

L'une des questions récurrentes en ce qui concerne la modélisation des informations liées à un produit est de savoir si les systèmes et donc les modèles industriels sont des mondes fermés ou ouverts. La réponse à cette question est importante, puisqu'elle restreint ensuite le choix du paradigme de modélisation : OWL et SWRL, puisque fondés sur la logique descriptive, raisonnent suivant l'OWA et considèrent donc le monde ouvert, tandis que SysML par exemple est une technologie UML qui suppose le monde fermé.

Les détracteurs des mondes fermés et de la CWA argumentent souvent qu'il est inexact de considérer l'absence d'une information comme une négation : ainsi, si je n'ai pas renseigné le nombre de roues de la voiture lors de l'instanciation du modèle, cela ne veut pas dire que la voiture n'a pas de roue : cela signifie que je ne sais pas encore combien elle en a. Puisque l'état du produit au cours de son cycle de vie est par nature évolutif, le modèle qui le formalise ne saurait être fermé.

A l'opposé, les détracteurs de l'OWA soulignent son incapacité à gérer la négation, et le besoin d'exprimer des requêtes sur l'absence d'une information, besoin récurrent dans les applications industrielles. On voit alors poindre des modèles qui expriment la négation ou l'absence d'une propriété par des moyens détournés, afin de permettre tout de même le raisonnement. Cela conduit donc à des « bricolages » de modélisation, dont la véracité sémantique peut être remise en question.

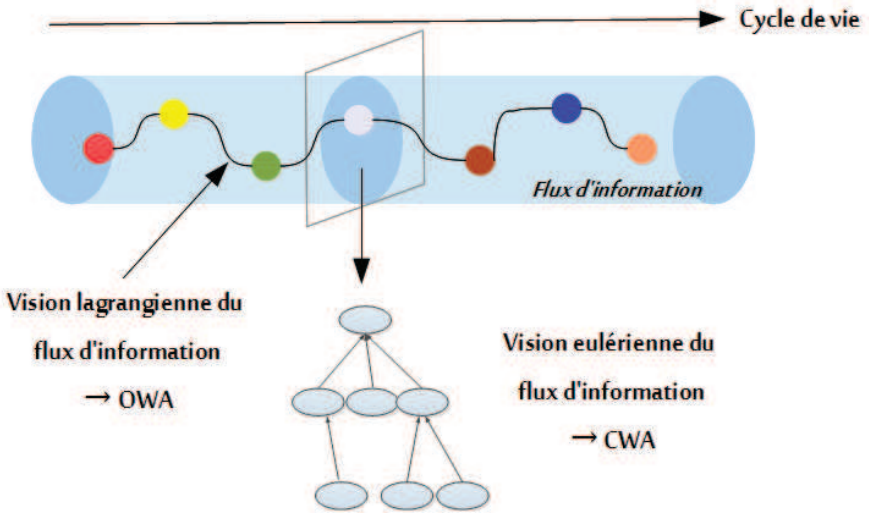


FIGURE 6.1 – Le flux d'information : analogie avec la mécanique des fluides

Chaque fois donc, les arguments opposés par les deux parties pour défendre l'un ou l'autre des paradigmes (OWA ou CWA) semblent pertinents, et se vérifient par des exemples concrets. Alors, on peut présumer qu'il ne reste qu'une solution à ce paradoxe : les systèmes industriels sont fermés et ouverts à la fois, et corollairement, ni l'un ni l'autre. Cette idée peut s'illustrer par une analogie avec la mécanique des fluides (voir figure 6.3.2) : si l'on considère que le flux d'information qui circule entre les SI est un fluide s'écoulant dans un tuyau, le choix entre l'OWA et la CWA revient à vouloir choisir, parmi les deux représentations d'Euler ou de Lagrange, la plus juste pour représenter l'écoulement du fluide. Or, ces deux représentations expriment des points de vue, qui, selon les hypothèses données et surtout le résultat recherché, ont des légitimités tout à fait équivalentes. C'est

le cas par là-même pour l'OWA et la CWA en ce qui concerne le flux d'information.

Ainsi, dans une vision lagrangienne du flux d'information, l'OWA semble pertinente pour raisonner sur des informations qui s'écoulent de systèmes en systèmes, car, d'un point de vue processus, le modèle d'information est effectivement un monde ouvert : il porte en lui l'absence de toutes les informations à venir au cours du cycle de vie. Cependant, si l'on se positionne à une phase donnée, à un instant donné, il est important de pouvoir connaître le flux d'information exactement en présence : l'absence est alors la négation d'une information. La CWA est ainsi plus juste pour requêter sur cette image figée du flux d'information. Des travaux en cours sur les raisonnements localement fermés, comme ceux de [Knorr et al., 2011, Grimm et al., 2006], pourraient permettre de trouver un compromis entre mondes fermés et ouverts.

Dans le cas d'application de cette thèse, il semble par ailleurs que la problématique d'OWA/CWA diffère selon le niveau de modélisation. Au niveau du méta-modèle, qui a une portée transverse entre les métiers et dont le rôle est de représenter la sémantique partagée, le fait d'opter pour une modélisation ontologique qui implique la définition ouverte des conditions nécessaires et suffisantes n'est pas problématique. Au contraire, la conceptualisation des éléments du modèle par une définition claire du contenu des catégories (même non fermée) constitue un apport sémantique indéniable. En revanche, au niveau applicatif, travailler en monde fermé paraît indispensable : d'abord pour vérifier la cohérence des données sans devoir « raisonner par l'absurde », mais aussi pour garantir une exécution pleine et juste des règles.

6.3.3 Quelle place pour les standards d'interopérabilité ?

Dans l'état de l'art de cette thèse, nous avions abordé les standards du PLM, mais aussi l'effort qui était déployé pour les transformer en modèles ontologiques. La question qui se pose est de savoir quelle est la place des standards dans le cadre de modélisation proposé. On ne peut cependant apporter une réponse unique à cette question car les standards du PLM ne sont pas tous du même niveau de modélisation et n'utilisent pas tous la même syntaxe. Alors, on peut différencier deux cas.

Dans le premier cas, illustré par la figure 6.2, il existe un standard d'unification dont la richesse sémantique est suffisante pour qu'il puisse être utilisé comme un modèle d'application. Au lieu de créer un modèle applicatif ad hoc à partir de l'ontologie du cycle de vie, il s'agit de créer les mappings sémantiques entre ce standard et l'ontologie du cycle de vie. Chaque utilisateur métier du niveau applicatif concerné instancie alors ce standard, tout comme un éventuel partenaire, avec qui l'entreprise travaillerait de manière collaborative.

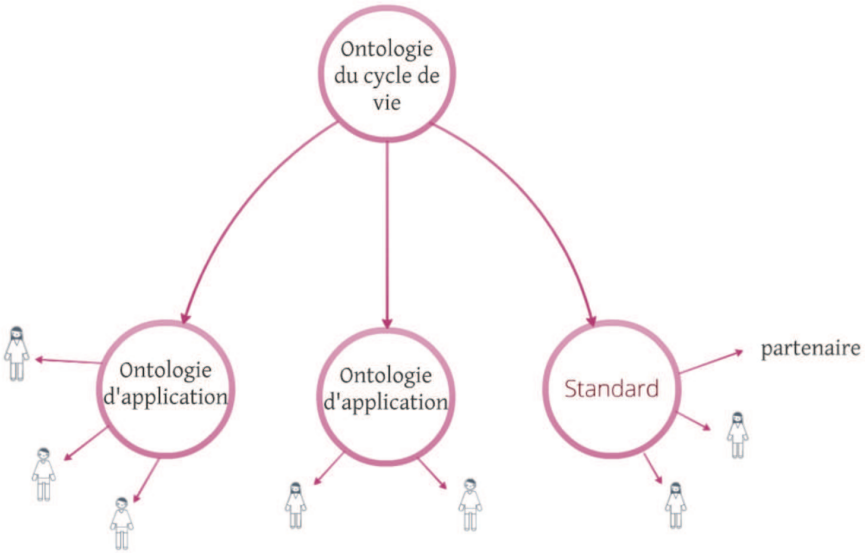


FIGURE 6.2 – L'inclusion d'un standard dans la démarche de modélisation

Par exemple, si le domaine d'application est restreint au champ de la CAO 3D, on peut imaginer qu'IGES en tant que standard de CAO soit le modèle applicatif. Si une partie de cette CAO est confiée à un sous-traitant, il utilisera également IGES pour échanger des informations. Puisque IGES est un modèle connu de l'ontologie du cycle de vie, le partage des informations avec les autres niveaux applicatifs ne posera pas de problème.

Il arrive cependant souvent que le modèle de l'entreprise possède une sémantique plus riche que les standards à disposition, et que l'intégration au sein du cadre comme modèle applicatif ne soit pas possible. Comme illustré sur la figure 6.3, le standard est alors uniquement le vecteur d'échange avec les partenaires. Dans le cas d'application de cette étude par exemple, a été menée une réflexion sur l'utilisation du standard ISO 15926 et son intégration au modèle de données. Il est apparu à l'issue de cet étude que l'ISO 15926, de part sa sémantique notamment, n'est qu'un standard d'échange, vecteur d'information avec les partenaires, et s'intègre dans la démarche envisagée comme sur la figure 6.3.

Du point de vue de l'échange d'information, une question intéressante est de se demander si l'ontologie du cycle de vie peut elle-même être un standard, afin de faciliter les échanges, et limiter les problématiques de mappings entre modèles. A l'heure actuelle, la littérature pointe la faiblesse de la sémantique des standards comme limite à cette approche. Mais à vrai dire, il existe une autre limite importante à cette idée : elle pose le problème de la préservation du savoir-faire de l'entreprise. En effet, le but des entreprise est d'intégrer le plus d'informations dans leurs modèles, afin de pouvoir capitaliser

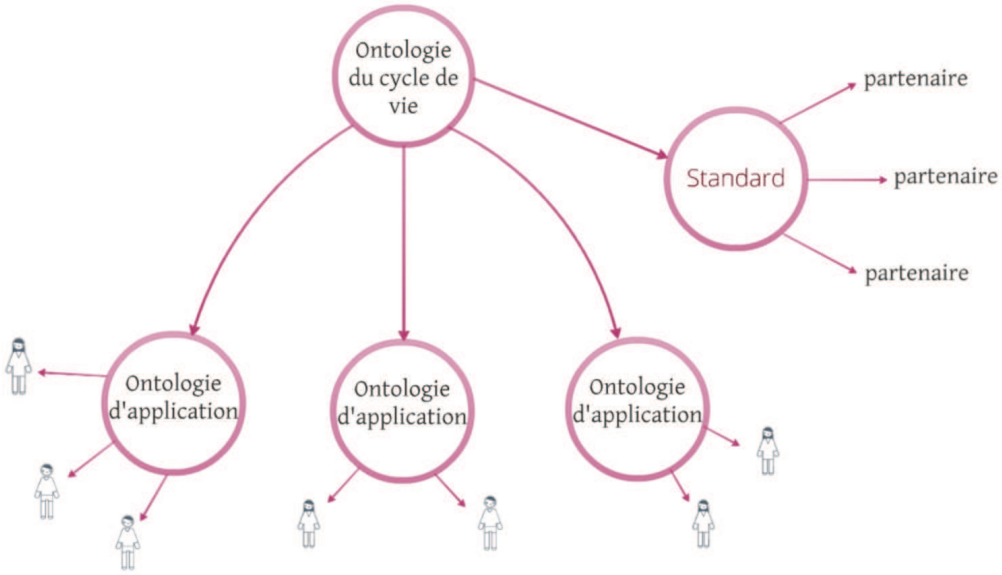


FIGURE 6.3 – Le standard comme vecteur d'échange avec les partenaires

et réutiliser la connaissance métier. Alors, partager l'ontologie du cycle de vie n'est pas envisageable : cela reviendrait à partager le savoir-faire métier de l'entreprise. Seule une portion amaigrie du modèle peut être échangée avec les partenaires. La problématique de distance sémantique entre le standard et les modèles d'entreprise n'est donc pas une problématique sémantique ou technique, il s'agit d'une problématique stratégique.

6.3.4 De la philosophie des ontologies

Dans ces travaux de recherche, nous sommes partis d'un postulat : que les ontologies sont des outils, en l'occurrence des outils de modélisation -plus précisément de formalisation, fondés sur la logique descriptive et le raisonnement. De par cette vision « outil », nous avons pu mettre à jour les avancées que ces outils permettent, tout comme leurs limites, en termes de performance, de fiabilité et de maturité. Cependant, l'ontologie est également une notion de philosophie. Et la modélisation ontologique -et les travaux qui en découlent- ont aussi une composante philosophique sur la formalisation, l'échange et le partage des informations. « L'entreprise sémantique » est une notion que l'on voit éclore et qui propose de repenser les démarches actuelles. Nous avons esquissé ces aspects plus conceptuels et philosophiques en nous posant la question du contenu sémantique des informations ; en tentant de « remonter le fil » de la sémantique. Mais nous n'avons pas creusé plus en détail la démarche ontologique en soi.

Le terme ontologie est à la mode, et, dans toutes les modes, il faut savoir s'affranchir des extrêmes. Des deux extrêmes : de ceux qui ne jurent que par ce nouveau concept, et qui voudraient faire croire que les ontologies sont le seul et unique moyen de modéliser correctement l'information. Ceux là en général sont aussi véhéments à l'égard des quelques-uns qui, comme nous avons pu le faire, soulèvent quelques verrous et limites de ces approches, qu'ils sont ambitieux sur les résultats à venir de ces outils « miracles » ; mais aussi des sceptiques, qui, parce qu'il subsiste des verrous non levés (et nous en avons pointé quelques-uns dans ces travaux), refusent de s'y pencher, et y collent justement cette étiquette de *mode*. Entre ces deux eaux tourmentées, il y a un travail de réflexion presque conceptuel à mener, car, de tout nouveau paradigme peut émerger une innovation, dès lors qu'on l'analyse suffisamment profondément pour en comprendre la philosophie. Nous avons souvent entendu les défenseurs des ontologies argumenter que, si certains y trouvent des limites, c'est que le changement de paradigme vis-à-vis des approches plus classiques, telles que UML par exemple, est trop grand, et qu'il est trop difficile de modifier les façons de penser. A la réflexion, cet argument n'est pas aussi fallacieux qu'il y paraît : changer les structures mentales est difficile, et, si les ontologies sont réellement aussi conceptuellement différentes des approches actuelles, il est pertinent de s'attacher à en comprendre la mécanique, pour permettre l'innovation dans les méthodes de modélisation. Innovation qui aboutira peut-être à autre chose que les ontologies telles qu'on les connait actuellement.

Nous avons touché du doigt cette réflexion « ontologique » de la modélisation sémantique avec l'application de la méthode OntoClean, qui, à l'instar des autres méthodes d'évaluation du contenu sémantique, ne s'attache pas aux termes métier mais à l'analyse des niveaux et natures sémantiques des concepts. Cette démarche nous a permis de porter un regard neuf sur la formalisation d'ontologies, et mérite d'être approfondie.

CHAPITRE 7

ANNEXES

7.1 L'ontologie Famille

L'ontologie famille est constituée de trois classes principales :

Classe	Signification
humain	regroupe toutes les personnes de la famille
femme	regroupe les humains de sexe féminins
homme	regroupe les personnes de sexe masculin. Classe disjointe de **femme**

TABLE 7.1: Liste des classes principales de l'ontologie famille

A ces classes principales s'ajoutent des propriétés objet qui permettent de définir les liens familiaux, listées dans le tableaux 7.2. Ces propriétés ont pour domaines de définition des classes de spécialisation de l'ontologie, qui sont définies à partir d'axiomes OWL dans le tableau 7.3. Enfin pour augmenter l'expressivité du modèle, on peut ajouter les règles SWRL du tableau 7.4.

Propriété	Domain	Range	type	Inverse
a_enfant	parent	enfant		a_parent
a_fille	parent	fille	ss-prop de a_enfant	
a_fils	parent	fils	ss-prop de a_enfant	
a_parent	enfant	parent		a_enfant
a_mere	enfant	mere	ss-prop de a_parent	
a_pere	enfant	pere	ss-prop de a_parent	
a_frere_soeur	humain	humain	réflexive	elle-même
a_soeur	humain	soeur	ss-prop de a_frere_soeur	
a_frere	humain	frere	ss-prop de a_frere_soeur	

Table 7.2: Liste des propriétés objet de l'ontologie famille

Classe	Axiome de définition
parent	définie par les domaines de propriété
enfant	définie par les domaines de propriété
soeur	≡ **femme** that *a_frere_soeur* some **humain**
frere	≡ **homme** that *a_frere_soeur* some **humain**
mere	≡ **femme** that *a_enfant* some **humain**
pere	≡ **homme** that *a_enfant* some **humain**
fils	≡ **homme** that *a_parent* some **humain**
fille	≡ **femme** that *a_parent* some **humain**

Table 7.3: Liste des classes spécifiques de l'ontologie famille

Règles SWRL	Signification
a_parent(?x, ?y) ∧ a_parent(?z, ?y) → a_frere_soeur(?x, ?z)	deux enfants qui ont le même parent sont frères et soeurs
a_frere_soeur(?x, ?y) ∧ homme(?y) → a_frere(?x, ?z)	spécifie la propriété *a_frere_soeur* en *a_frere* quand il s'agit d'un homme
a_frere_soeur(?x, ?y) ∧ femme(?y) → a_soeur(?x, ?z)	spécifie la propriété *a_frere_soeur* en *a_soeur* quand il s'agit d'une femme

Table 7.4: Liste des règles de l'ontologie famille

7.2 L'ontologie d'interface conception/fabrication

L'ontologie d'interface conception/fabrication est constituée de neufs classes principales disjointes :

A ces classes sont associées des propriétés objet, qui permettent d'exprimer :

– la taxonomie entre articles d'une même nomenclature (propriétés *hasE(M)element*, inverses de *isE(M)element* ;

Classe	Signification
Item	contient l'ensemble des articles des deux nomenclatures, divisés en deux classes disjointes : Eitem (conception) et Mitem (fabrication)
Interface	les sous-classes de la classe interface permettent de discriminer les articles selon les types de liens qui existe avec l'autre nomenclature. Ces classes sont instanciées par des axiomes OWL qui raisonnent sur les classes item.
Parent	il s'agit d'une classe qui exprime la position des articles dans la nomenclature. Elle permet de distinguer les articles qui ont des enfants de ceux qui n'en ont pas.
SemanticInformation	est une classe disjointe des autres, qui regroupe l'ensemble des informations liées aux articles et que l'on souhaite modéliser (c'est-à-dire qui ne sont pas exprimées sous forme de dataproperty).

Table 7.5: Liste des classes de l'ontologie d'interface conception/fabrication

- les liens entre les nomenclatures (propriété *hasinterface*, divisée en *isLinkedto* (lien non unique) et *isthesame* (functional)) ;
- l'attribution d'information d'autres type par la propriété *hasinfo* ;

Les éléments caractérisant ces différentes propriétés sont donnés dans le tableau 7.6.

Propriété	Domain	Range	type	Inverse
haselement	item	item	transitive	iselement
hasEelement	Eitem	Eitem	transitive	isEelement
hasMelement	Mitem	Mitem	transitive	isMelement
hasinfo	item	SemanticInformation		
hasinterface	item	item		hasinterface
islinkedto	item	item		islinkedto
isthesame	item	item	functional	isthesame

Table 7.6: Liste des propriétés objet de l'ontologie d'interface conception/fabrication

Les axiomes OWL utilisés pour le raisonnement ontologique sont résumés dans le tableau 7.7.

Classe	Axiome OWL
Interface	≡ item that hasinterface some item
Linkeditem	≡ item that islinkedto some item
Sameitem	≡ item that isthesame some item

Table 7.7: Liste des axiomes OWL de l'ontologie d'interface conception/fabrication

7.3 L'ontologie du cycle de vie : cas des règles métier

L'ontologie du cycle de vie pour exprimer la sémantique des règles métier est constituée des classes suivantes :

Classe	Signification
ressource	concept racine
document de doctrine	liste les documents dont sont issues les règles L
metier	concept racine
triplet_MUDU	ensemble des triplets constitués des éléments du modèle produit MUDU
triplet_metier	ensemble des triplets constitués des éléments du modèle métier
concept_metier	concept du modèle métier formant les triplets
relation_metier	relations des triplets métier
regle	concept racine
regle L	règles issues des documents de doctrine
regle LN	règles métier
regle LSN	règles conformes à MUDU
regle LF	règles formelles
condition	conditions des règles LN et LSN
implication	implication des règles LN et LSN
contexte	contexte des règles LN et LSN
mapping_metier_MUDU	mappings entre les triplets métier et les triplets MUDU

Table 7.8: Liste des classes de l'ontologie du cycle de vie (règles métier)

Les propriétés objet suivantes ont été définies :

Propriété	Domain	Range
a_antecedent	triplet_metier	concet_metier
a_consequent	triplet_metier	concet_metier
a_relation	triplet_metier	relation_metier
a_condition	regle_LN	condition
	regle_LSN	
a_contexte	regle_LN	contexte
	regle_LSN	
a_implication	condition	implication
a_triplet_neg	contexte	triplet_metier
	condition	
	implication	
	mapping_metier_MUDU	
a_triplet_pos	contexte	triplet_metier
	condition	
	implication	
	mapping_metier_MUDU	
a_triplet_MUDU	contexte	triplet_MUDU
	condition	
	implication	
	mapping_metier_MUDU	
a_expression_formelle	regle_LSN	regle_LF
est_exprimee_par	regle_L	regle_LN
est_maitre_de	regle_L	regle_LN
est_extraite_de	regle_L	document_de_doctrine
est_validée_par	regle_LN	regle_LSN

Table 7.9: Liste des propriétés de l'ontologie du cycle de vie (règles métier)

7.4 Liste des règles métier du cas d'application

Le cas d'application qui sert de base à la validation du démonstrateur est contraint par des règles métier dont des exemples sont donnés ci-après sans ordre spécifique.

Le code matériel des robinets, et soupapes est déterminé par la nature du fluide véhiculé et de sa phase.

- Pour de l'Air, le code matériel est VA.
- Pour de l'Eau borée et non primaire, il vaut VB.
- Pour de l'Eau de circulation, il vaut VC.
- Pour de l'Eau déminéralisée, il vaut VD.
- Pour de l'Eau brute, il vaut VE.
- Pour un Combustible gazeux, il vaut VF.
- Pour du CO2, il vaut VG.
- Pour del'Huile, il vaut VH.
- Pour de l'Air de ventilation, il vaut VI.
- Pour un Effluent gazeux, il vaut VJ.
- Pour un Effleuent liquide, il vaut VK.
- Pour de l'Eau condensée, il vaut VL.
- Pour un combustible d'allumage, il vaut VM.
- Pour de l'Eau de circuit Noria, il vaut VN.
- Pour de l'Eau primaire, il vaut VP.
- Pour un liquide organique, il vaut VQ.
- Pour un réactif, il vaut VR.
- Pour un Effluent solide, il vaut "VS".
- Pour de l'Eau de nappe - Eau potable, il vaut VT.
- Pour de la vapeur, il vaut VV.
- Pour de l'Argon, il vaut VX.
- Pour de l'Hydrogène, il vaut VY.
- pour de l'Azote, il vaut VZ.

Calcul du repère fonctionnel d'une occurence conformément à la règle ENG 2.01 indice A (code ECS)

Le repère fonctionnel est composé comme suit :

- Si le SE d'appartenance est corrélé et :
  - si SE_Tr
    - "-"

        - trigramme du SE_Tr d'appartenance
        - numéro d'ordre de la SF d'appartenance
        - numéro d'ordre de la FB d'appartenance
        - numéro d'ordre d'occurrence
        - code matériel
        - code matériel extension
      - Si SE_Site
        - repère fonctionnel du SE_Site d'appartenance
        - numéro d'ordre de la SF d'appartenance
        - numéro d'ordre de la FB d'appartenance
        - numéro d'ordre d'occurrence
        - code matériel
        - code matériel extension
    - Si le SE d'appartenance est décorrélé et :
      - Si SE_Tr
        - "-"
        - trigramme du SE_Tr d'appartenance
        - numéro d'ordre d'occurrence
        - code matériel
        - code matériel extension
      - Si SE_Site :
        - repère fonctionnel du SE_Site d'appartenance
        - numéro d'ordre d'occurrence
        - code matériel
        - code matériel extension

Les matériels concernés sont les suivants :

- Réservoirs horizontaux et verticaux.
- Réservoirs extérieurs ou intérieurs calorifugés ou non.
- Réservoirs reliés directement à l'atmosphère.
- Réservoirs isolés de l'atmosphère :
  - soit par un dispositif à toit flottant,
  - soit par ciel gazeux.
- Réservoirs sous pression de liquide ou de gaz.
- Réservoirs de stockage d'effluents radioactifs.
- Réservoirs métalliques ou en composites.

Les matériels non concernés par la fiche sont les suivants :

- Réservoirs de transport destinés à être transportés, après remplissage, à un autre

endroit pour y être vidés.
- Bacs de rétention (voir fiche DRI B4 [40]).
- Bâches intégrées à la structure des bâtiments.

Chauffage par serpentin :
- Utilisation possible sur une bâche non calorifugée,
- Installation possible sur une bâche stockant un fluide inflammable.

Chauffage par traçage :
- Ces deux types de traçage ne peuvent être installés que sur des bâches prévues pour être calorifugées.

Chauffage par résistances électriques
- Installation possible sur une bâche non calorifugée.

Les réservoirs dont la température du fluide est supérieure à 60 degrés C sont calorifugés ou entourés d'un grillage protecteur afin de protéger le personnel.

7.5 L'ontologie métier OWL2/SWRL

L'ontologie métier OWL/SWRL est constituée de neufs classes principales disjointes :

Classe	Signification
Composant	contient tous les éléments ou ensembles constitutifs de la centrale
Orientation	
Localisation	décrit la localisation d'un élément dans la centrale, ou par rapport à d'autres éléments
Fonction	décrit les fonctions que peut remplir un composant
Etat	décrit les états particuliers dans lesquels se trouvent les composants
Attribut	un attribut est une caractéristique d'un autre concept qui peut prendre des valeurs spécifiques
Identifiant	attribut particulier, sous forme de code
Materiau	liste l'ensemble des matériaux constitutifs des composants
Propriete	décrit les propriétés particulières que peuvent avoir les composants

Table 7.10: Liste des classes de l'ontologie métier

Ces classes sont reliées par des propriétés spécifiques, afin de pouvoir « ranger » les instances de manière automatique en fonction des propriétés qu'elles possèdent, mais aussi détecter a priori d'éventuelles incohérences dans le modèle. Les propriétés définies sont :

Propriété	Domain	Range
a_attribut	thing	Attribut
a_etat	Composant	Etat
assure	Composant	Fonction
a_orientation	Composant	Orientation
a_localisation	Composant	Localisation
relativement_a	Localisation	Composant
a_identifiant	Composant	Identifiant
a_propriété	Composant	Propriete
est_fait_de	Composant	Materiau
est_de_type	thing	thing
propriété spécifique	composant	composant

Table 7.11: Liste des propriétés de l'ontologie métier

Les propriétés spécifiques sont l'ensemble des propriétés qui permettent de relier les

composants entre eux. Par exemple, la propriété *contient* permet d'exprimer qu'un réservoir contient un fluide. Ces propriétés sont aussi nombreuses que le métier l'exige, et peuvent être amenées à évoluer. Ce sont les seuls types de propriétés que l'utilisateur métier peut ajouter a posteriori.

7.6 Expressing formal rules within ontology-based models using SWRL : an application to the nuclear industry

Cette section est une contribution soumise à l'International Journal On Product Lifecycle Management (JPLM).

Abstract : Ontology-based models propose a new modelling paradigm for industrial applications. In this study, we explore the maturity of ontology-based models for industrial applications from a special point of view : the expression of business rules. Indeed, expressing and implementing rules is a crucial criterion for industrial models. We therefore propose an overview of the use of OWL-DL/SWRL approaches in the industry, as well as a critical overview of the execution of SWRL rules on an industrial use case : the modelling of a nuclear power plant.

Keywords : Rules ; ontology ; OWL ; SWRL ; SPARQL ; nuclear industry.

1. Context of the study

The French nuclear plant designer, Electricité de France (EDF), is currently designing a global Product Lifecycle Management (PLM) approach for the whole nuclear power plant lifecycle, from design to disposal. PLM is a global methodology that aims to manage the complexity surrounding the product during its whole lifecycle [Terzi et al., 2010]. One aspect of PLM is to manage information systems (IS) in order to provide efficient information sharing, access, and storage at each stage of the product lifecycle. Nuclear power plants are products that present specifities [Mun et al., 2008] : they are made of billions of elements ; they involve many actors, data, and businesses ; they require long term data storage ; and they are constrained by a large number of norms, security requirements, and business procedures. In this context, expressing business rules in an automatical and rigorous manner is a preriquisite to a collaborative and efficient information management along the whole nuclear plant lifecycle. The expression and execution of formal rules is therefore the scope of this study.

Since the early 2000's, there has been a growing interest in semantic web technologies that enhance information management of industrial IS. In the area of PLM, ontology-based models have recently emerged. A full literature review of ontology-based applications in PLM can be found in [Fortineau et al., 2013b]. Some examples of ontology-based models follow : [Matsokis and Kiritsis, 2010] propose an ontology-based translation of the SOM model, [Vegetti et al., 2011] provide a product ontology (PRONTO) for the whole product

lifecycle and[Krima et al., 2009] propose OntoSTEP as an OWL-DL ontology for the STEP standard (ISO 10303). In the nuclear sector, [Mun et al., 2006] design an OWL ontology to model a common data repository. Moreover, the ISO 15926 standard (part 8) recommends the use of ontologies to implement the shared Reference Data Library.

In this paper, we explore the ability of semantic web languages to express business rules within an ontology-based product model. The Web Ontology Language (OWL) and the Semantic Web Rule Language (SWRL) are studied and tested on a use case extracted from the nuclear industry. The aim of the study is to explore the abilities and limits of SWRL to fully comply with industrial needs in term of formal rule implementation and execution within an OWL-based product model. This would hardly influence the possibility of using ontology-based models for the management of industrial information.

As a consequence, the paper is organized as follows. Section 2 presents a literature review about business rules, ontology applications in PLM, and about previous works using SWRL as a rule language. Then Section 3 describes the industrial need and the use case studied. Section 4 presents the OWL/SWRL ontology developed and focuses on two particular business rules that raise implementation issues. Finaly, section 5 discusses the limits of OWL/SWRL to express business rules, compared to other rule approaches, and presents further works. Section 6 concludes this paper.

2. Literature review

2.1 Ontology-based model in the aera of PLM

Ontologies are, according to the widely accepted definition given by Gruber [Gruber, 1995], “an explicit specification of a conceptualization". In mathematical words, an ontology is a set of classes, properties connecting classes to one another, restrictions on properties, and axioms [Maedche, 2002]. As [Fankam, 2008] explain, there exist two kinds of ontologies : storage ontologies and inference ontologies. Storage ontologies are used for information capture, storage, classification and re-use from heterogeneous sources. Contrarily to storage ontologies, inference ontologies can infer information. Inference is the ability to make deductions about instances, regarding the classes and the defined properties and restrictions [Krima et al., 2009], and is performed by inference engines.

Expected and noticed benefits of inference ontology-based models in the PLM literature are of various kinds, that can be classified into three categories :

– **Integration and completeness :** Current PLM models suffer from a lack of content. It has been estimated that around 50 % of the available information is not stored in

current information systems [Ben Kheder et al., 2010], information retrieval and reuse are not efficient [Louis-Sidney et al., 2012]. Inference ontologies provide a great expressivity that makes it possible to integrate information from different abstraction levels [Fiorentini and Rachuri, 2009]. Illustrations of these abilities are the works of [Yang et al., 2008] and [Dong et al., 2011] in product configuration, and the works of [Jeong et al., 2010] and [Matsokis and Kiritsis, 2011] in integrated product model.

- **Reasoning and automation :** The explosion of the number of data and the constant reduction of processes duration require automated and intelligent systems. Inference ontology-based models provide intelligence, in the sense of the "ability to explicit implicit informations from the model specification" [Baader et al., 2005]. [Vegetti et al., 2011] use for instance the inference mechanism in order to automate Bill of Materials (BOM) generation for product families. [Lim et al., 2011] automatically generate design annotations with inference ontologies to improve decision making at the design stage. According to [Brandt et al., 2008] and [Raza et al., 2011], deductions that an inference engine provides can be better than that of humans, and transform the model into a source of intelligent data.
- **Flexibility, extensibility and collaboration :** Current PLM tools fail to adapt to the perpetual evolutions of product environment [Yang et al., 2008]. PLM systems need for instance a "timely and precise information retrieval" [Lim et al., 2011]. Ontologies, as semantic tools, are well adapted to web applications and collaboration and, moreover, are extendable, which prevents the models from premature obsolescence [Gimenez et al., 2008]. It is therefore possible to continuously adapt the model to product and process evolutions.

These noticed benefits of inference ontologies are relevant for the nuclear industry, where automation, collaboration and information storage are crucial issues. Moreover, the usage of OWL is considerably growing in the nuclear power plant sector thanks to the ISO 15926 standard [Kim et al., 2011]. This standard formalizes in OWL the concepts used for the integration of life-cycle data for process plants. Hence, ontologies are a modelling paradigm that should be explored to model nuclear plant information. These facts led us to investigate the potential of ontologies to model nuclear plants. However, what limits the use of ontologies is their ability to express and execute business rules, which represent a growing interest in industrial applications.

2.2 Expressing business rules : a requirement in industrial applications

The Business Rules Group [1] is an organization composed of IS professionnals working on rule classification, formalization, and expression, whose work lead to the development of

1. http ://www.businessrulesgroup.org/home-brg.shtml

the Semantics of Business Vocabulary and Business Rules (SBVR) language. They propose the following definition of a business rule : "*a business rule is a statement that defines or constraints some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business.*" [Hay and Healy, 2000]. The Business Rules Group also published a "Business rules manifesto", a set of recommendations for the expression and management of business rules, where they recommend that business rules be expressed in a natural language, in order to address the business experts. Formal rules implementation is therefore only one aspect of business rules expression. As explained in [Fortineau et al., 2013a], the translation from natural language assertions into formal rules is a challenge in itself, but is not the scope of this study, which will only consider formal rules.

2.3 Why is OWL-DL/SWRL a widely adopted ontology-based approach in industry

2.3.1 Standard languages for inference ontologies

OWL is the only standard language (recommended by the World Wide Web Consortium (W3C) since 2004) that expresses inference ontology-based models. Figure 7.1 presents the two main sub-languages of OWL : OWL1 and OWL2, and their own sub-languages. OWL-DL is the most widespread sub-language of OWL in industrial applications, but the number OWL2 applications is increasing nowadays, since support tools are becoming more stable.

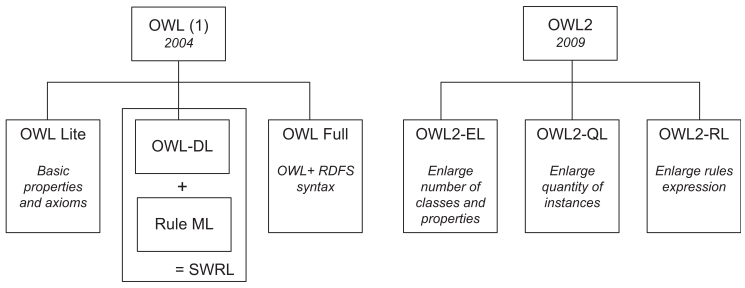


Figure 7.1 – Standardized OWL Languages

2.3.2 SWRL : a rule language for ontology-based models

Nowadays, the Semantic Web Rule Language (SWRL) is the most appropriate formal rule language to comply with OWL-based models, because it is integrated with OWL, i.e.,

it is syntactically and semantically coherent with OWL. Indeed, SWRL is a standard language based on OWL-DL and on the Rule Markup Language (RuleML), which provides both OWL-DL expressivity and rules from RuleML [Horrocks et al., 2010] (see again figure 7.1). It is the only approach that gathers ontology and rules in product development [Fiorentini et al., 2010]. Rules in SWRL are implication rules. Hence, the syntax of a SWRL rule is of the following form [Mun and Ramani, 2011] :

$$antecedent \rightarrow consequent \tag{7.1}$$

This syntax implies that the consequent must be true when the antecedent is satisfied. OWL expressions can occur in both antecedent and consequent [Hirankitti and Xuan, 2011]. Different kinds of rules can be expressed in SWRL according to the literature. The next section presents only the ones that are involved in case of power plant modelling, like control and access policies, which ensure legal requisitions about data access, such as configuration, association and naming rules, which enhance interoperation between actors and information retrieval, as well as automation and strategic rules, in order to improve business processes and to reduce Time-to-Market, and finally assembly rules, related to product assembly constraints, and design rules, like design checking or product consistency checking.

2.3.3 SWRL as a rule language : a literature review

Control policies have already been implemented using SWRL in healthcare. In the work of [Beimel and Peleg, 2011], the hospital organization is modeled in an OWL-DL ontology and control policies are expressed as implication rules on classes and properties from the OWL-DL model. Therefore, in this case, the organization surrouding the product must be modeled to perform rule implementation. It increases the kind of information required in the model.

Naming and classification rules are particulary well adapted to SWRL syntax because OWL expressions are based on classes and properties. As explained [Fiorentini et al., 2010], SWRL provides association rules, and allows to "associate instances to new classes and to create properties between instances". Examples of the use of SWRL in such cases can be found in [Zhao and Liu, 2008] who provide a methodology to translate EXPRESS-driven models to OWL and SWRL, and in [Dong et al., 2011] and [Yang et al., 2008] in case of product configuration.

Process automation is important to reduce time-to-market and to manage complex product design. As [Raza et al., 2011] experimented in the case of ERP systems,

ontologies and SWRL rules allow knowledge management by providing intelligent data that “feed automated processes to aid agile manufacturing". In another context, [Rossellò-Busquet et al., 2011] achieve automated actions with a SWRL-based model involving energy management. [Elenius et al., 2009] automate military events analysis with a SWRL-based reasoning. Finally, [Matsokis and Kiritsis, 2011] and [Lim et al., 2011] enhance the extendability of OWL models with SWRL rules by automating design annotations or knowledge (i.e. classes and properties) creation under a “learnable" approach.

The design of complex products, like power plants, involves a lot of components and assembly relationships. Design **assembly rules** can be expressed in SWRL. For instance [Kim et al., 2006] use SWRL rules to express features relations in product design. In the same way, the National Institute of Standards and Technology (NIST) recently transformed the Open Assembly Model in a OWL-DL based model with SWRL rules [Fiorentini et al., 2007]. Finally, [Mun and Ramani, 2011] have expressed dependancy relations in SWRL in case of injection mould design.

Design checking rules and business rules are probably the most complex rules because they depend on a lot of conditions. However, [Wicaksono et al., 2011] perform SWRL rules to detect design conflicts in product configuration. Even if SWRL is able to provide *restrict* rules, *partof* relationships and other design oriented axioms [Fiorentini et al., 2010], few applications of SWRL in design checking or modelling exist. Moreover, they involve mostly services or processes than the product itself.

According to the literature, SWRL seems in compliance with the industrial requirements that involve formal rules within product models. The rest of the study proposes a critical overview of SWRL rules implementation within an industrial use case extract from the nuclear industry. The next Section describes the chosen use case.

3. Industrial application

3.1 The existing modelling paradigm and industrial needs

Our industrial partner EDF achieves the power plant design and installation, and ensures maintenance of the plant. It is also responsible for the plant disassembly. Currently, several dedicated softwares are used during the nuclear power plant lifecycle. Most of them are commercial softwares, which are poorly interoperable. To overcome this issue, EDF has developed its own business model, to which each application should comply. This shared model is called MUDU (French acronym for *user data unified model*) and is a set

of coherent user dictionnaries that are used to generate business catalogs for each nuclear power plant project and each business activity. These dictionnaries are constrained and designed following a meta-model composed of physical or abstract objects (components and connection points), attributes (relations between components, characteristics, graphical representations, and connection types), and rules.

The current rule classification in MUDU (called later “EDF classification") is the following (see figure 7.2) :

- **Meta-rules**, represented at the bottom in Figure 7.2, are of two kinds. The so-called “design rules" are a set of norms, procedures, and security requirements, while the “activities requirements" describe each step of the design process. Meta-rules are currently textual rules only. They are non-applicable rules, but general rules, from which business rules then derivate.
- **Business rules** : these rules directly address the components of the model. They are of three kinds : value rules add values to the characteristics, connexion rules define the connexion possibilities between the components, while constraints on relations constrain the relationships between components.

The current classification is based on the object that the rule affects and not on the rule type. This can be explained by the fact that very few business rules are formaly expressed and implemented at the moment. Most of them are written in natural language expressions, or non-consistent logical expressions. The industrial need is therefore to provide a formal expression and implementation of all the business rules and a rule classification, while taking into consideration the following requirements :

- rules might be defined and executed in different applications ;
- standard languages are preferred ;
- rule management might be considered ;
- the proposed solution might be based on the existing model.

3.2 Description of the use case

The use case presented in figure 7.3 is a subset of components, characteristics, and relations taken from a catalog (i.e. an application model) of MUDU. This subsect defines the functional specification of the electrical power supply of the nuclear plant. Indeed, the component *AlimElecSpecFonct* represents the functional specification of the electrical power supply, including its safety classification (*ClasFonctSur*), the voltage quality (*QualU*), and the need for an alternative power supply during maintenance (*RealimMaintElec*). The functional specification of the power supply installation is represented by the component

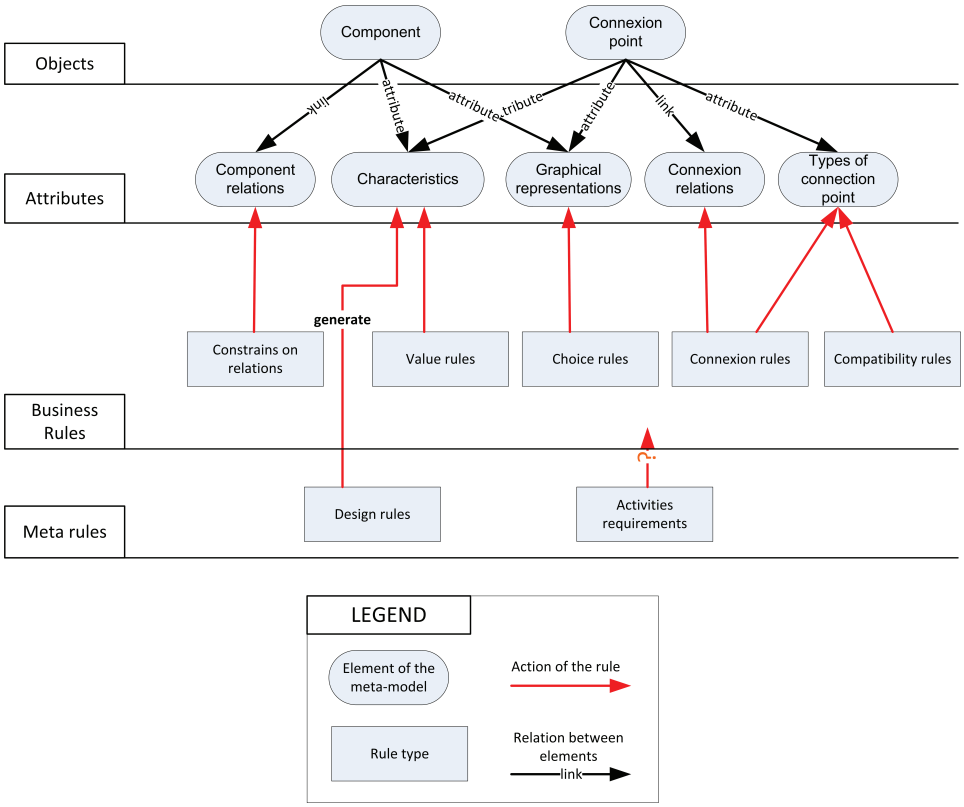


FIGURE 7.2 – Current rules classification

IN_SpecFonct : its characteristic *ChutAvionPec_Agre* indicates if some security measures need to be taken in case of a plane crashes on the nuclear power plant installation. The component *AlimSpecFonctIN* associates the functional specifications of the electrical power supply and of its installation. These specifications are encoded in a string represented by the characteristic *RepFonct*. The component *SpecU* indicates the voltage specification, i.e., the minimum and maximum variation of the voltage in exceptional or transitional cases. The characteristics of this component indicate the location of the voltage (*Bigr*), its nominal value (*U_Nom*), and its quality (*QualU*). The component AlimElecSpec associates the voltage specification to the functional specifications. This component indicates which functionalities are needed for a specific voltage. The functionalities are encoded in a string (*RepFonct*), while the voltage is represented by its nominal value (*U_Nom*).

In this use case, there exist several business rules, mostly value rules (rules to calculate the value of component characteristic). In order to ease the reading of the paper we focus on two specific rules that are typical rules of the use case and which raise critical issues of SWRL :

– Rule 1 `RepeFonctAlimElecSpec` is a value rule according to the EDF classification.

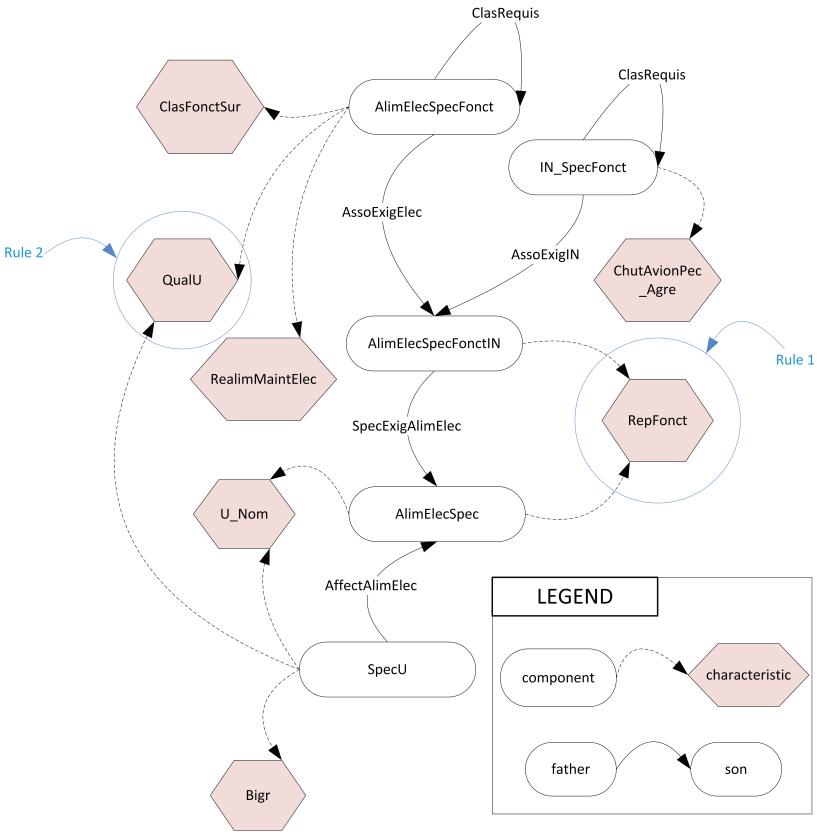


FIGURE 7.3 – Description of the use case

It codifies the functional specification into the *RepFonct* string. The code indicates the location of the voltage (*Bigr*), the nominal value of the voltage (*U_Nom*), the need for an alternative power supply during maintenance (*RealimMaintElec*), the safety classification (*ClasFonctSur*), and the security measures in case of a plane crash (*ChutAvionPec_Agre*). For each *AlimElecSpec* Rule 1 assigns a value to the related *RepFonct* string by concatenating the values of the encoded characteristics. The value to concatenate depends on the value of the characteristics. For example, if the boolean value of *RealimMaintElec* is equal to true, the rule concatenates the string .Realim ; otherwise it concatenates the string . only.

– Rule 2 `AffectAlimElecSpecExigAlimElecQualU` is a constraint on relation according to the EDF classification. It constrains the voltage quality of the voltage specification and of the functional specification depending on their relationship. This rule checks that the voltage quality (characteristic *QualU*) of the voltage specification (component *SpecU*) and the voltage quality (characteristic *QualU*) of the functional specification (component *AlimElecSpecFonct*) related to the same component

AlimElecSpec are equal.

4. An OWL/SWRL solution for the use case

The present use case has been modeled in an OWL-DL ontology, and formal rules expressed as SWRL assertions. The modelling and rules have then been tested using the Protégé software. Protégé is an ontology editor developed by the University of Standford enabling to define OWL and OWL2 ontologies. Several reasoners are included in Protégé, like FACT++ or Pellet. For the use case Pellet 2.3 and Protégé 4.2 were used. Protégé and Pellet are free tools.

4.1 Presentation of the OWL-DL ontology

The OWL ontology has been designed in order to comply with the existing EDF model MUDU, expressed in UML. These was therefore a model transformation, but we kept the semantic of the existing model. More precisely, we modeled the following elements :

- The concepts *Composant* (components), *Relation*, and *CaracteristiqueComposant* (characteristic) are expressed as OWL disjoint classes to represent the meta-model.
- The various components, relations, and characteristic from the use case are defined as OWL classes and also subclasses of the *Composant*, *Relation*, and *CaracteristiqueComposant* classes. Hence, they can be instancied as well. Figure 7.4 shows the proposed model.
  - *AlimElecSpec*, *SpecU*, *AlimElecSpecFonctIN*, *INSpecFonct*, and *AlimElecSpecFonct* are disjoint sub-classes of the class *Composant*. Each of these five classes also owns one individual, application instanciation of the product model ;
  - *AffectAlimElec*, *SpecExigAlimElec*, *AssoExigElec*, and *AssoExigIN* are disjoint sub-classes of the class *relation*. Each of these four classes also owns one individual ;
  - *Bigr*, *U_Nom*, *QualU*, *RealimMaintElec*, *ClasFonctSur*, *RepFonct*, and *ChutAvion-Pec_Agre* are disjoint sub-classes of the class *CaracteristiqueComposant*. These classes are themselves instancied.
- Three object properties are created
  - **caracteristique_du_composant** connects a characteristique to a component ;
  - two properties **fils**, and **pere** : **fils** connects a relation to the component in the role of son, while **pere** connects a relation to the component in the role of father. Hence, the component taxonomy is expressed with these two properties.

– One datatype property is created : **valeur**. This property connects a characteristic to its datatype value. The ontology also contains owl :allValuesFrom constraints and cardinality constraints.

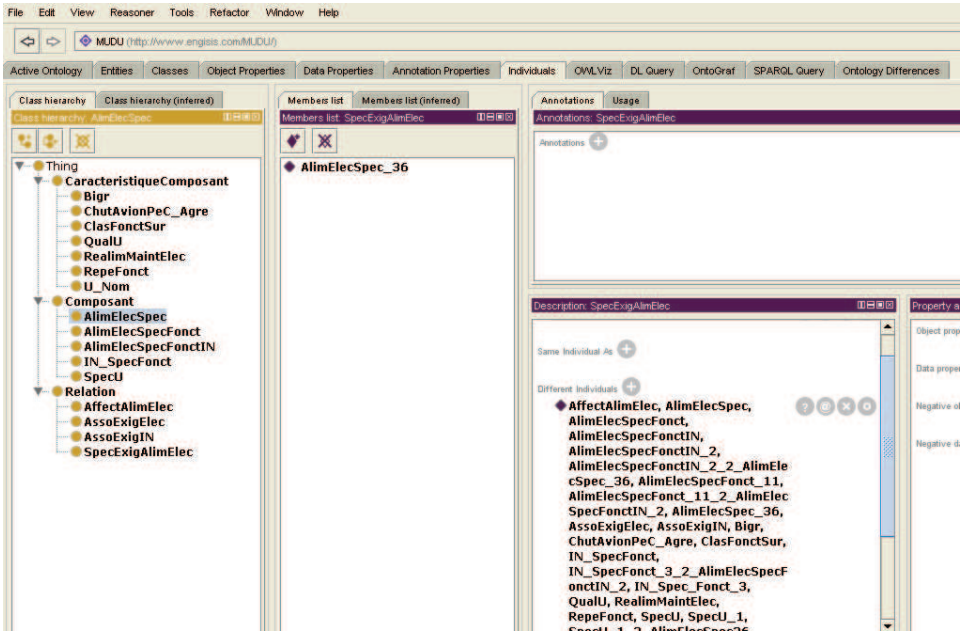


FIGURE 7.4 – Ontology of the use case

4.2 Rule expression in SWRL

According to the EDF classification, rule 1 `RepeFonctAlimElecSpec` is a derivation rule. This rule includes conditionnal instructions (like IF...THEN). These instructions also evaluate the value of the characteristics *ChutAvionPeC_Agre* and *RealimMaintElec*, and concatenate a different string based on those values. **The problem with rule 1** is that SWRL is not designed for this kind of instruction : SWRL doesn't support multiple conditions in the antecedent. Hence, one rule for each value of the conditions *ChutAvionPeC_Agre* and *RealimMaintElec* is required. Figure 7.5 presents the SWRL expression of rule 1 when *ChutAvionPeC_Agre* and *RealimMaintElec* are both false. The inference engine Pellet executes rule 1 and infers the value of *RepFonct* associated to AlimElecSpec36 : LA220 V.NS.F1. Finally, the rule is executable, but has to be divided into several rules.

Rule 2 `AffectAlimElecSpecExigAlimElecQualU` is an integrity constraint, according to the EDF classification. This rule is a restriction on two components related to *AlimElecSpec* : *SpecU* and *AlimElecSpecFonc*. They must have the same *QualU* value whenever they are related to the same *AlimElecSpec*. **The problem with rule 2** is that it can not

```
AffectAlimElec(?aae), AlimElecSpec(?aes), AlimElecSpecFonct(?aesf),
AlimElecSpecFonctIN(?aesfIN), AssoExigElec(?aee), AssoExigIN(?aeIN), Bigr(?b1),
ChutAvionPeC_Agre(?capa), ClasFonctSur(?cfs), IN_SpecFonct(?INsf), QualU(?qu1),
RealimMaintElec(?rme), RepeFonct(?rf1), SpecExigAlimElec(?seae), SpecU(?su),
U_Nom(?un), characteristique_du_composant(?INsf, ?capa),
characteristique_du_composant(?aes, ?rf1), characteristique_du_composant(?aes, ?un),
characteristique_du_composant(?aesf, ?cfs), characteristique_du_composant(?aesf,
?qu1), characteristique_du_composant(?aesf, ?rme),
characteristique_du_composant(?su, ?b1), fils(?aae, ?aes), fils(?aeIN, ?aesfIN), fils(?aee,
?aesfIN), fils(?seae, ?aes), pere(?aae, ?su), pere(?aeIN, ?INsf), pere(?aee, ?aesf),
pere(?seae, ?aesfIN), valeur(?b1, ?x), valeur(?capa, false), valeur(?cfs, ?w), valeur(?qu1,
?y), valeur(?rme, false), valeur(?un, ?p), stringConcat(?z, ?x, ?p, ".", ?y, "", ".", ?w, "") ->
valeur(?rf1, ?z)
```

Figure 7.5 – SWRL expression of rule 1

```
AffectAlimElec(?aae), AlimElecSpec(?aes), QualU(?qu1), SpecU(?su),
characteristique_du_composant(?su, ?qu1), fils(?aae, ?aes), pere(?aae, ?su), valeur(?qu1, ?x) -
> artificial_property(?aes, ?x)
```

```
AlimElecSpec(?aes), AlimElecSpecFonct(?aesf), AlimElecSpecFonctIN(?aesfIN),
AssoExigElec(?aee), QualU(?qu1), SpecExigAlimElec(?seae),
characteristique_du_composant(?aesf, ?qu1), fils(?aee, ?aesfIN), fils(?seae, ?aes), pere(?aee,
?aesf), pere(?seae, ?aesfIN), valeur(?qu1, ?x) -> artificial_property(?aes, ?x)
```

Figure 7.6 – SWRL expressions of rule 2

be directly expressed in SWRL. The following solution is employed : a data property (**artificial_property**) is defined in the OWL ontology to assign a string to *AlimElecSpec*. **artificial_property** is a functional property, e.g. it may assign only one string to each instance of *AlimElecSpec*. Figure 7.6 presents the two SWRL rules defined :

- the first rule assigns the *QualU* value of the component *SpecU* to the related *AlimElecSpec*, via the **artificial_property** ;
- the second rule assigns the *QualU* value of the component *AlimElecSpecFonc* to the same related *AlimElecSpec*, via the **artificial_property**.

Since **artificial_property** is functional, if the various *QualU* values assigned are different, an inconsistency is detected. The present solution is a workaround, which is possible because the two values of *QualU* are constrained to be equal. It wouldn't be possible to express that *QualU* of *SpecU* is higher than *QualU* of *AlimElecSpecFonc*.

5. Limits and further works

In this section, we discuss the limits of an OWL-DL/SWRL solution for industrial application, and present research prospects. Some limitations explained later are specific to the use case, while others are more generally related to product modelling.

5.1 Specific limits of the use case

In rule 1 of the use case, the limits of SWRL are associated to the impossibility of expressing IF...THEN conditions within the antecedent of a SWRL rule. To overcome this limitation, a rule for each case of the IF condition was created. The adopted solution is not sustainable, since a unique rule in the EDF model had to be translated into several rules and the relationship between those rules has been lost.

In rule 2 of the use case, the limits of SWRL are associated to the impossibility of using OWL to compare data literals connected to different related classes ; and declaring a constraint that is attached to a set of related classes, and not to one particular class. In the use case, the equivalence between the data literals had to be checked. To overcome this issue, a functional artificial property was created. This property was populated through SWRL rules and its functionality was checked by the reasoner. This was not an optimal solution for four reasons.

- Firstly, even if annotations were used to explain the implementation, the rule was not clearly readable : a single constraint was implemented through two SWRL rules and a property, which was not part of the initial model.
- sSecondly, if rule 2 had included a complex comparison between the data literals, the solution would have appeared even more intricate.
- Thirdly, rule 2 was not intended to be implemented in SWRL since it is a constraint, and not an implication.
- Fourthly, the SWRL rules connected the artificial property to an instance of the AlimElecSpec class, even if rule 2 was not related to any particular class.

5.2 Limits of SWRL as rule language

More generally we identified major issues that could limit the use of OWL-DL/SWRL as modelling paradigm for industrial applications, when rule implementation is required. These issues are : the Open World Assumption, the complexity of rule management, and information updates.

5.2.1 Open World Assumption

OWL-DL, and consequently SWRL, works usually under the Open World Assumption (OWA) [Fiorentini et al., 2010], because they originally focus on the Semantic Web, that deals with an unlimited knowledge resource (internet). The OWA means that “if a proposition cannot be proved to be true with the current knowledge, the system cannot declare this proposition as false" [Drummond and Shearer, 2006]. Unlike OWA, Closed World systems assume that empty knowledge is false, in fact that the available knowledge in the model is exhaustive. Industrial information systems can be considered on one side as closed worlds in which information should be threated under the Closed World Assumption (CWA) [Wang and Yu, 2011], since their information sources are finite. But on the other side, the interest of using ontologies is the extendability of the model, which is based on OWA. This is an important paradox that constrains model design. The same query, on the same model, could give different results regarding if the reasoning is made under a closed or an open world. Expressing control rules in SWRL implies for instance to reason in a closed world [Beimel and Peleg, 2011]. Works are going on to reason on SWRL-based models, under the CWA, like the work of [Wang and Yu, 2011] who propose a new reasoner called BCAR that for.

5.2.2 Rule complexity

SWRL has difficulties dealing with complex rules. For instance, it is not possible in theory to create rules with numbered predicates [Elenius et al., 2009]. This limitation can be overtaken with SWRL built-ins [Rosselló-Busquet et al., 2011]. Built-ins allow to define a rdf list of OWL arguments, that then feed a SWRL rule as an unique antecedent. However, this method is complex. This difficulty in expressing complex rules call into question the ability of SWRL to expressing design rules, that can depend on a lot of arguments. Because of it, preference is sometimes given to C++ language to express design checking rules of nuclear plants.

5.2.3 Information updates

Finally, “SWRL rules cannot be used to modify existing information in an ontology" [Rosselló-Busquet et al., 2011]. Indeed, SWRL rules originally create new knowledge, i.e. new instances or new properties between existing instances. It cannot change the value of an instance. For example, let’s imagine a class expressing a machine status, with instances “on" and “off" expressing if the machine is turned on or off. If the machine status changes

from on to off, an SWRL rule can express that “off" is a new instance connected to “machine". Then the machine status owns both “on" and “off" instances, since the SWRL rule does not replace “on" by “off". As for complex rules, this drawback can be avoided by using Built-ins. But it demands an SWRLbuiltinbridge, and a specific OWL model describing built-ins, which is an awkward and complex solution.

5.3 Comparison to other rule languages

Other rule languages showed various capacities in modelling and checking rules in information systems.

5.3.1 OCL : a rule language for UML

As the case in point, The OMG supports the Object Constraint Language (OCL) [OMG, 2012]. It can be used to describe additional constraints about objects in a model and makes it possible to check intrinsic model coherency and coherency between model and model instantiation. The language being particularly expressive, its use without restriction leads in practice to undecidability when the checking operations are performed automatically. Nevertheless, operating over on an appropriate fragment insure decidability with actual DL reasoners [Queralt et al., 2012]. Model upon which OCL constraints apply are UML models, however translation works have already begun and revealed a compatibility between UML and OWL [OMG, 2009] along with OCL and SWRL [Milanovic et al., 2007]. The possibilities of expressing rules provided by OCL, as well as its close world orientation (more in line with industrial information system context, as presented above) could lead to a joint use with SWRL.

5.3.2 EXPRESS : an integrated language

At last, another possibility would be to express rules using the EXPRESS modelling language [ISO-10303-11, 1994]. The EXPRESS-G graphical part of the language looks similar to UML regarding modelling possibilities (e.g. attributes, inheritance, aggregation, polymorphism). However EXPRESS-M provides a rich set of rule modelling capabilities, based upon the concept of local and global rules. Local rules constrain the values of individual attributes or combinations of attributes for every entity instance. Global rules can be applied to check the consistency of a set of instances using queries, functions and procedures. [Barbau et al., 2012] as well as [Zhao and Liu, 2008] study how to convert an

EXPRESS model to an OWL/SWRL representation, but none of them managed to convert all the semantics of rule restrictions.

5.4 Further steps : SPARQL, SPIN, or integrity constraints ?

Future researches might include the usage of integrity constraints, SPARQL 1.1, and SPIN. Integrity constraints (ICs) are used to validate instance data using the Closed World Assumption (CWA) and the Unique Name Assumption (UNA (http ://clarkparsia.com/files/pdf/ic-owled09.pdf)) [Tao, 2012]. The usage of CWA and UNA is well suited to the industrial case, where the constraints of the meta-model have to be imposed against a fully defined set of instances. In the use case, for example, ICs might check the minimum cardinality constraints. If an instance of *SpecU* needs to be connected to at least one instance of *AlimElecSpec*, it is necessary that this connection exists among instances, in other words, that the CWA applies. Under the OWA this connection would not exist primarily : it could be possible that the connection does not exist yet, as the instances are created, but only later, while checking the ontology, which is not acceptable in the use case. The final ontology would contain both common OWL axioms, interpreted in the OWA, and ICs, interpreted in the CWA. ICs are syntactically defined as OWL axioms, but semantically interpreted as constraints by the reasoner. Pellet has developed an Integrity Constraint Validator (http ://clarkparsia.com/pellet/icv), which validates the ICs by translating them into SPARQL (SPARQL Protocol and RDF Query Language) query-answering. SPARQL is a language, which defines queries as functions over an RDF dataset [Hartig, 2012].

In the use case, the newest version of SPARQL, v. 1.1 seems relevant. With this new version, SPARQL becomes much more powerful than a normal query language [Buil-Aranda et al., 2011]. It includes the possibility of :

- using a large variety of built-ins ;
- executing CRUD (Create, Read, Update, and Delete) operations ;
- defining local variables.

In the use case, for example, rule 1 `RepeFonctAlimElecSpec` can be expressed with a SPARQL query. This query would contain a built-in function to concatenate the strings (fn :concatenate), the INSERT operation to connect AlimElecSpecFonctIN to the right concatenated string, and one local variable for each case of the IF clause, thus resolving the issue related to the number of SWRL rules needed to express rule 1.

SPARQL 1.1 has been a W3C specification since 03/23/2013. It is already implemented by some of the most common reasoners and software APIs (e.g., Pellet and Jena). Unfortunately, SPARQL is based on RDF, and not on OWL. As a consequence, it might be

explored if these reasoners actually incorporate the semantics of OWL into their APIs. Pellet, for example, does not execute TBox queries. The impact of this limitation on the use case might be verified. If SPARQL is used, it must be decided if the APIs related to the DL inference shall be called first and then the APIs related to the rules execution, or vice versa. Different choices might bring different results.

SPARQL has been incorporated into the SPARQL Inferencing Notation (SPIN) [Callahan and Dumontier, 2012]. SPIN (http ://www.w3.org/Submission/2011/SUBM-spin-overview-20110222/) is a SPARQL-based rule and constrained language for the Semantic Web, which has been submitted to the W3C but has not been accepted yet for standardization. SPIN has been applied by Krima et al. [Krima et al., 2012] to validate product data models. In SPIN, SPARQL queries are stored as RDF triples. Because the queries are stored as RDF, they are attached to a specific class of the ontology. This is very similar to the programming logic in Java or C++, where methods are declared within a class. Another similarity with programming logic is related to the usage of the keyword " ?this", which refers to each member of the class. In the industrial case, rules are not attached to specific classes : this makes SPIN unsuitable for this particular need. SPIN allows writing the business rules as *constructors*, *rules*, or *constraints*. *Constraints* are invariants : they need to be always valid for each member of the class. If a constraint is not valid, then an invidual of the type *spin :ConstraintViolation* is created and the reasoner shows this individual. This is logically very different from the consistency checking mechanisms in DL. The usage of this kind of inference needs to be further investigated.

6. Conclusion

In this study, we explore the possibilities of OWL/SWRL as modelling paradigm for cases involving rule implementation. The syntax of SWRL rules and the limits of SWRL are presented on an application case, taken from a real industrial case : a nucelar power plant model. More general issues about SWRL as a rule language are also exposed in this study. They are mainly raised by the Open World Assumption and the difficulties to manage rule complexity and information update.

This study focuses on rule expression, because business rules are crucial in product modelling : business rules embed business knowledge and enable automation. This is clear that business rule expression will be more and more required in industrial applications. In this context, the lack of an integrated rule approach within OWL based models could be a great issue to the development of ontology-based models.

Further researches should focus on the use of SPARQL/SPIN as a rule language for

OWL based models, and of the development of reasoners able to locally close the world, to overcome the OWA issue, that we identified as one of the major issue of the OWL/SWRL modelling paradigm.

7.7 La modélisation UML des règles

Cette annexe a été supprimée pour raisons de confidentialité.

Figure 7.7 – Diagramme de classes UML des relations entre règles

Figure 7.8 – Diagramme de classes UML des règles L

Figure 7.9 – Diagramme de classes UML des règles LN

Figure 7.10 – Diagramme de classes UML des règles LSN

Figure 7.11 – Diagramme de classes UML du mapping métier/MUDU (partie métier)

Figure 7.12 – Diagramme de classes UML du mapping métier/MUDU (partie MUDU)

Cette annexe a été supprimée pour raison de confidentialité.

7.8 Le schéma d'architecture du démonstrateur

BIBLIOGRAPHIE

[Afacan and Demirkan, 2011] Afacan, Y. and Demirkan, H. (2011). An ontology-based universal design knowledge support. *Knowledge-based systems*, 24 :530–541.

[Ahmed et al., 2007] Ahmed, S., Kim, S., and Wallace, K. (2007). A methodology for creating ontologies for engineering design. *Journal of Computing and Information Science in Engineering*, 7 :132–140.

[Al-Ashaab et al., 2012] Al-Ashaab, A., Molyneaux, M., Doultsinou, A., Brunner, B., Martínez, E., Moliner, F., Santamaría, V., D., T., Ewers, P., and Knight, G. (2012). Knowledge-based environment to support product design validation. *Knowledge-based systems*, 26 :48–60.

[Al-Hawari and Aqlan, 2012] Al-Hawari, T. and Aqlan, F. (2012). A software application for E-Kanban-based WIP control in the aluminium industry. *International Journal of Modelling in Operations Management*, 2 :119–137.

[Albani and Dietz, 2011] Albani, A. and Dietz, J. L. (2011). Enterprise ontology based development of information systems. *International Journal of Internet and Enterprise Management*, 7(1) :41–63.

[Alsafi and Vyatkin, 2010] Alsafi, Y. and Vyatkin, V. (2010). Ontology-based reconfiguration agent for intelligent mechatronic systems in flexible manufacturing. *Robotics and Computer-Integrated Manufacturing*, 26(4) :381–391.

[Ammar-Khdoja, 2007] Ammar-Khdoja, S. (2007). *Processus d'aide à la spécification et à la vérification d'application d'ingénierie à base de connaissances expertes.* PhD thesis, Ecole Centrale de Nantes, France.

[Antoniou and van Harmelen, 2009] Antoniou, G. and van Harmelen, F. (2009). Web Ontology Language : OWL. *Handbook on ontologies, International Handbooks on Information Systems*, 1 :91–110.

[Assmann et al., 2006] Assmann, U., Zschaler, S., and Wagner, G. (2006). *Ontologies, Metamodels, and the Model-Driven Paradigm*, chapter 9, pages 249–273.

[Baader et al., 2008] Baader, F., Calvanese, D., McGuiness, D., Nardi, D., and Patel-Schneider, P. (2008). *The Description Logics Handbook.*

[Baader et al., 2005] Baader, F., Horrocks, I., and Sattler, U. (2005). Description Logics as ontology languages for the semantic web. *Journal of Computer science*, 2605 :228–248.

[Bajwa et al., 2011] Bajwa, I., Bordbar, B., and Lee, M. (2011). SBVR vs OCL : A comparative analysis of standards. pages 261–266. 14th IEEE International Multitopic Conference (INMIC 2011), Karachi, Pakistan.

[Barbau et al., 2012] Barbau, R., Krima, S., Rachuri, S., Narayanan, A., Fiorentini, X., Foufou, S., and Sriram, R. (2012). OntoSTEP : Enriching product model data using ontologies. *Computer Aided Design*, 44(6) :575–590.

[Barcikowsky, 2006] Barcikowsky, M. (2006). *Vers une évaluation de la robustesse des connaissances au sein d'une base de connaissances*. PhD thesis, Université Claude Bernard - Lyon 1, France.

[Barkmeyer et al., 1999] Barkmeyer, E., Denno, P., Feng, S., Jones, A., and Wallace, E. (1999). NIST response to MES request for information. *NIST Response to RFI-3 MES Models*, pages 2–4.

[Barkmeyer and Kulvatunyou, 2007] Barkmeyer, E. and Kulvatunyou, B. (2007). An ontology for the e-Kanban business process. Technical report, National Institute of Standards and Technology.

[Baysal et al., 2004] Baysal, M., Roy, U., Rachuri, S., Sriram, R., and Lyons, K. (2004). The Open Assembly Model for the exchange of assembly and tolerance information : overview and example. ASME International design engineering technical conference.

[Bechhofer et al., 2013] Bechhofer, S., Buchan, I., De Roure, D., Missier, P., Ainsworth, J., Bhagat, J., Couch, P., Cruickshank, D., Delderfield, M., Dunlopa, I., Gamblea, M., Michaelides, D., Owena, S., Newmanc, D., Sufi, S., and Goble, C. (2013). Why linked data is not enough for scientists. *Future Generation Computer Systems*, 45 :204–228.

[Beetz et al., 2005] Beetz, J., van Leeuwen, P., J., and de Vries, B. (2005). An ontology web language notation of the industry foundation classes. CIB W78 Conference on Information Technology in Construction.

[Beimel and Peleg, 2011] Beimel, D. and Peleg, M. (2011). Using OWL and SWRL to represent and reason with situation-based access control policies. *Journal of Data and Knowledge Engineering*, 70 :596–615.

[Ben Kheder et al., 2010] Ben Kheder, A., Henry, S., and Bouras, A. (2010). An analysis of the interaction among design, industrialization and production. International Conference on Product Lifecycle Management, Bremen, Germany.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web : A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, pages 1–7.

[Bisseret, 1995] Bisseret, A. (1995). *Représentation et décision experte : Psychologie cognitive de la décision chez les aiguilleurs du ciel*. Toulouse : Octarès.

[Bizer et al., 2009] Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, 5(3) :1–2.

[Blomqvist and Ohgren, 2008] Blomqvist, E. and Ohgren, A. (2008). Constructing an enterprise ontology for an automotive supplier. *Engineering Applications of Artificial Intelligence*, 21 :386–397.

[Borst and Akkermans, 1997] Borst, P. and Akkermans, H. (1997). An ontology approach to product disassembly. *Journal of Computer Science*, 1319 :33–48.

[Bréal, 1897] Bréal, M. (1897). *Essai de Sémantique (sciences des significations)*. Librairie Hachette et Cie.

[Brandt et al., 2008] Brandt, S., Morbach, J., Matiadis, M., Theissen, M., Jarke, M., and Marquardt, W. (2008). An ontology-based approach to knowledge management in design processes. *Computers and Chemical Engineering*, 32 :320–342.

[Braun et al., 2005] Braun, C., Wortmann, F., Hafner, M., and Winter, R. (2005). Method construction - a core approach to organizational engineering. pages 1295–9. Proceedings of the 2005 ACM Symposium on Applied Computing, SAC'05, Santa Fe, NM, ACM, New York (USA).

[Buil-Aranda et al., 2011] Buil-Aranda, C., Arenas, M., and Corcho, O. (2011). Semantics and optimization of the SPARQL 1.1 federation extension. *Lecture Notes in Computer Science*, 6644 :1–15.

[Cai et al., 2012] Cai, H., Xu, B., and Bu, F. (2012). A conceptual ontology-based resource meta-model towards business-driven information system implementation. *Journal of Universal Computer Science*, 18(17) :2493–2513.

[Callahan and Dumontier, 2012] Callahan, A. and Dumontier, M. (2012). Evaluating scientific hypotheses using the SPARQL Inferencing Notation. *Lecture Notes in Computer Science*, 7295 :647–658.

[Cantamessa et al., 2012] Cantamessa, M., Montagna, F., and Neirotti, P. (2012). An empirical analysis of the plm implementation effects in the aerospace industry. *Computers in industry*, 63 :243–251.

[Cao et al., 2004] Cao, L., Luo, C., Luo, D., and Zhang, C. (2004). Integration of business intelligence based on three-level ontology services. pages 17–23. Web Intelligence.

[Catalano et al., 2009] Catalano, C., Camossi, E., Ferrandes, R., and Cheutet, V. (2009). A product design ontology for enhancing shape processing in design workflows. *Journal of Intelligent Manufacturing*, 20 (5) :553–567.

[Ceri and Fraternale, 1997] Ceri, S. and Fraternale, P. (1997). Designing database applications with objects and rules : The idea methodology. *Series on Database Systems and Applications, Addison Wesley Longman*.

[Chandrasegaran et al., 2013] Chandrasegaran, S., Ramania, K., Sriram, R.D.and Horváth, I., Bernard, A., Harik, R., and Gaoa, W. (2013). The evolution, challenges, and future of knowledge representation in product design systems. *Computer-aided design*, 45 :204–228.

[Chang et al., 2008] Chang, X., Sahin, A., and Terpenny, J. (2008). An ontology-based support for product conceptual design. *Journal of Robotics and Computer-Integrated Manufacturing*, 24 :755–762.

[Chen and Vernadat, 2004] Chen, D. and Vernadat, F. (2004). Standard on enterprise integration and engineering-state of the art. *International Journal of Computer Integrated Manufacturing*, 17 (3) :235–253.

[Chen et al., 2009] Chen, Y., Chen, Y., and Chu, C. (2009). Development of a mechanism for ontology-based product lifecycle knowledge integration. *Expert Systems with Applications*, 36(2) :2789–2779.

[Chungoora et al., 2013] Chungoora, N., Young, R. I., Gunendran, G., Palmer, C., Usman, Z., Anjum, N. A., and Case, K. (2013). A model-driven ontology approach for manufacturing system interoperability and knowledge sharing. *Computers in Industry*, 64 :392–401.

[Cruz et al., 2004] Cruz, I., Sunna, W., and Chaudhry, A. (2004). Ontology alignment for real-world applications. Annual national conference on Digital government research. Digital Government Society of North America.

[David and Sutter, 1985] David, A. and Sutter, E. (1985). *La gestion de l'information dans l'entreprise*. BBF.

[Dekkers et al., 2013] Dekkers, R., Chang, C., and Kreutzfeldt, J. (2013). The interface between "product design and engineering" and manufacturing : A review of the literature and empirical evidence. *International Journal of Production Economics*, 63 :749–755.

[D'Elia et al., 2010] D'Elia, A., Roffia, L., Zamagni, G., Vergari, F., Toninelli, A., and Bellavista, P. (2010). Smart applications for the maintenance of large buildings : How to achieve ontology-based interoperability at the information level. IEEE Symposium on Computers and Communications.

[Demoly et al., 2012] Demoly, F., Matsokis, A., and Kiritsis, D. (2012). A mereotopological product relationship description approach for assembly oriented design. *Robotics and Computer-Integrated Manufacturing*, 28 :681–693.

[Dong et al., 2011] Dong, M., Yang, D., and Su, L. (2011). Ontology-based service product configuration system modeling and development. *Expert systems with applications*, 38 (9) :11770–11786.

[Dori and Shpitalni, 2005] Dori, D. and Shpitalni, M. (2005). Mapping knowledge about product lifecycle engineering for ontology construction via object-process methodology. volume 54, pages 117–122. CIRP Annals-Manufacturing Technology.

[Drummond and Shearer, 2006] Drummond, N. and Shearer, R. (2006). The Open World Assumption. Technical report, University of Manchester, UK.

[EIF, 2004] EIF (2004). European interoperability framework - White paper. pages 1–40.

[El-Gohary and El-Diraby, 2010] El-Gohary, N. M. and El-Diraby, T. E. (2010). Domain ontology for processes in infrastructure and construction. *Journal of Construction Engineering and Management*, 136(7) :730–744.

[Elenius et al., 2009] Elenius, D., Martin, D., Ford, R., and Denker, G. (2009). Reasoning about resources and hierarchical tasks using OWL and SWRL. International Semantic Web Conference, Chantilly, USA.

[Fankam, 2008] Fankam, C. (2008). Ontodb2 : support of multiple ontology models within ontology based database. EDBT Ph.D. workshop, New York, USA.

[Fankam, 2009] Fankam, C. (2009). *OntoDB2 : un système flexible et efficient de Base de Données à Base Ontologique pour le Web Sémantique et les données techniques*. PhD thesis, ENSMA, Futuroscope, France.

[Fenves, 2001] Fenves, S. (2001). A Core Product Model for representing design information. Technical report, National Institute of Standards and Technology.

[Fenves et al., 2007] Fenves, S., Foufou, S., Bock, C., and Sriram, R. (2007). Cpm : a Core Product Model for representing design information. Technical report, Manufacturing Systems, Integration, Division, National Institute of Standards and Technology, Gaithersburg, USA.

[Fernández-Lopéz et al., 1999] Fernández-Lopéz, L., Gomez-Perez, A., and Sierra, J. (1999). Building a chemical ontology using METHONTOLOGY and the ontology design environment. *IEEE Intelligent Systems*, 14(1) :37–46.

[Fielding, 2000] Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, USA.

[Fiorentini et al., 2007] Fiorentini, X., Gambino, I., Liand, V., Foufou, S., Rachuri, S., Bock, C., and Mani, M. (2007). Towards an ontology for Open Assembly Model. International Conference on Product Lifecycle Management, Milan, Italy.

[Fiorentini et al., 2013] Fiorentini, X., Paviot, T., Fortineau, V., Goblet, J., and Lamouri, S. (2013). Modeling nuclear power plants engineering data using ISO 15926. International Conference on Industrial Engineering and Systems Management.

[Fiorentini and Rachuri, 2009] Fiorentini, X. and Rachuri, S. (2009). STEP-OAGIS harmonization joint working group. Technical report, National Institute of Standards and Technology.

[Fiorentini et al., 2008] Fiorentini, X., Rachuri, S., Mani, M., Fenves, S., and Sriram, R. (2008). An evaluation of description logic for the development of product models. volume 7481. US National Institute of Standards and Technology Interagency Report.

[Fiorentini et al., 2010] Fiorentini, X., Sudarsan, R., Suh, H., Lee, J., and Sriram, R. (2010). An analysis of description logic augmented with domain rules for the development of product models. *Journal of Computing and Information Science in Engineering*, 10 :1–13.

[Fortineau et al., 2013a] Fortineau, V., Paviot, T., Guissé, A., and Lamouri, S. (2013a). A transformation model to express business rules from natural language to formal execution : an application to nuclear power plant. Conference on Manufacturing, Modeling, Management and Control (MIM), Saint-Petersburg, Russia.

[Fortineau et al., 2013b] Fortineau, V., Paviot, T., and Lamouri, S. (2013b). Improving the interoperability of industrial information systems with description logic-based models - the state of the art. *Computers in Industry*, 64 :363–375.

[Fortineau et al., 2011] Fortineau, V., Paviot, T., Lamouri, S., and Cheutet, V. (2011). The cone-bom model for consistent and minimal product structures representation. International Conference on Product Lifecycle Management, Eindhoven, Netherlands.

[Fortineau et al., 2013c] Fortineau, V., Paviot, T. Fiorentini, X., Louis-Sidney, L., and Lamouri, S. (2013c). Expressing formal rules within ontology-based models using SWRL : an application to the nuclear industry. *International Journal On Product Lifecycle Management*, sousmis.

[Galeta et al., 2006] Galeta, T., Klajin, M., and Karakasic, M. (2006). Product model suited for the ERP system. 9th International Design Conference-Design.

[Garetti et al., 2005] Garetti, M., Terzi, S., Bertacci, N., and Brianza, M. (2005). Organisational change and knowledge management in plm implementation. *International Journal of Product Lifecycle Management*, 1(1) :43–51.

[Genin et al., 2005] Genin, P., Lamouri, S., and Thomas, A. (2005). Planification avancée : Aps. *Techniques de L'ingénieur*, AG5120 :1–13.

[Gimenez et al., 2008] Gimenez, D., Vegetti, M., Leone, H., and Henning, G. (2008). PRONTO : An ontology for comprehensive and consistent representation of product information. *Computers in Industry*, 59 :231–241.

[Giovannini et al., 2012] Giovannini, A., Aubry, A., Panetto, H., Dassisti, M., and El Haouzi, H. (2012). Ontology-based system for supporting manufacturing sustainability. *Annual Reviews in Control*, 36 :309–317.

[Grabot et al., 2008] Grabot, B., Mayère, A., and Bazet, I. (2008). ERP systems and organizational change : a socio-technical insight. *Springer Verlag*.

[Grimm et al., 2006] Grimm, S., Motik, B., and Preist, C. (2006). Matching semantic service descriptions with local closed-world reasoning. *The Semantic Web : Research and Applications*, pages 575–589.

[Gruber, 1995] Gruber, T. (1995). Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human Computer Studies*, 43 (5) :907–928.

[Grundstein, 2000] Grundstein, M. (2000). *Knowledge Management, classic and contemporary works*. MIT Press, published by Morey, D., Maybury, M. and Thuraisingham, B.

[Guarino and Welty, 2009] Guarino, N. and Welty, C. (2009). *Handbook on Ontologies*, chapter An overview of OntoClean, pages 201–220.

[Guissé et al., 2012] Guissé, A., Lévy, F., and Nazarenko, A. (2012). From regulatory texts to BRMS : How to guide the acquisition of business rules ? International Symposium on Rules : Research based and industry focused, Montpellier, France.

[Han and Park, 2009] Han, K. and Park, J. (2009). Process-centered knowledge model and enterprise ontology for the development of knowledge management system. *Expert Systems with Applications*, 36 :7447–7447.

[Hartig, 2012] Hartig, O. (2012). SPARQL for a Web of Linked Data : Semantics and computability. *Lecture Notes in Computer Science*, 7295 :8–23.

[Hay and Healy, 2000] Hay, D. and Healy, K. (2000). Defining business rules : What are they really ? Technical report, The Business Rules Group.

[Herbst, 1997] Herbst, H. (1997). *Business Rule-oriented Conceptual Modeling*. Physica Verlag, Springer Verlag.

[Hirankitti and Xuan, 2011] Hirankitti, V. and Xuan, T. (2011). A meta-reasoning approach for reasoning with SWRL ontologies. International Multiconference of Engineers.

[Hoffmann, 2008] Hoffmann, P. (2008). *Similarité sémantique inter-ontologies basée sur le contexte*. PhD thesis, Université Claude-Bernard - Lyon 1.

[Horrocks et al., 2006] Horrocks, I., Kutz, O., and Sattler, U. (2006). The even more irresistible SROIQ. *KR*, 6 :57–67.

[Horrocks et al., 2010] Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B., and Dean, M. (2010). SWRL : A Semantic Web Rule Language combininig OWL and RuleML.

[Hu et al., 2012] Hu, R., Gao, F., and Gong, C. (2012). An E-Kanban system for fork truck assembly lines. *Applied Mechanics and Materials*, 220-223 :259–262.

[Ishak, 2010] Ishak, K. (2010). *Integration of Semantic Interoperability in a Distributed Architecture for Multi-Site Planning*. PhD thesis, Université de Toulouse.

[ISO-10303-11, 1994] ISO-10303-11 (1994). Industrial automation systems and integration - Product data representation and exchange - Part 11 : Description methods : The EXPRESS language reference manual. International Standard Organization.

[ISO-14258-1998, 1998] ISO-14258-1998 (1998). Industrial automation systems : Concepts and rules for enterprise models. International Standard Organization.

[ISO/15926-1, 2004] ISO/15926-1 (2004). Industrial automation systems and integration - integration of life-cycle data for process plants including oil and gas production facilities. Part 1 : Overview and fundamental principles.

[Jeong et al., 2010] Jeong, S., Wegner, D., and Noh, S. (2010). Validation of an ontology-based approach for enhancing human simulation in general assembly environments. volume 3. Proceedings of the WCE 2010, London, UK.

[Jun et al., 2007] Jun, H., Kiritsis, D., and Xirouchakis, P. (2007). Research issues on closed-loop PLM. *Computers in Industry*, 58 :855–868.

[Kim et al., 2011] Kim, B., Teijgeler, H., Mun, D., and Han, S. (2011). Integration of distributed plant lifecycle data using ISO 15926 and Web services. *Annals of nuclear energy*, 38 :2309–2318.

[Kim et al., 2008] Kim, B., Teijgeler, H., Mun, D., Sun, D., Hwang, J., and S., H. (2008). A representation of implementation of process plants models using OWL and ISO 15926. International Conference on Product Lifecycle Management, Seoul, Korea.

[Kim et al., 2006] Kim, K., Manley, D., and Yang, H. (2006). Ontology-based assembly design and information sharing for collaborative product development. *Computer-Aided Design*, 38 :1233–1250.

[Kiritsis, 2011] Kiritsis, D. (2011). Closed-loop plm for intelligent products in the era of the internet of things. *Computer-Aided Design*, 43 :479–501.

[Kiritsis et al., 2012] Kiritsis, D., El Kadiri, S., Perdikakis, A., and Milicic, A. (2012). Design of fundamental ontology for manufacturing product lifecycle applications. International Conference on Advances in Production Management Systems, Rhodes Island, Greece.

[Knorr et al., 2011] Knorr, M., Alferes, J. J., and Hitzler, P. (2011). Local closed world reasoning with description logics under the well-founded semantics. *Artificial Inteligence*, 175(9) :1528–1554.

[Kontchakov et al., 2011] Kontchakov, R., Wolter, F., and Zakharyaschev, M. (2011). Logic-based ontology comparison and module extraction, with an application to DL-Lite. *Journal of Artificial Intelligence*, 174 :1093–1141.

[Kossman et al., 2010] Kossman, M., Odeh, M., Watts, S., and Gillies, A. (2010). Ontology-driven requirements engineering with reference to the aerospace industry. *Journal of E-Technology*, 1(1) :22–30.

[Koukias et al., 2012] Koukias, A., Nadoveza, D., and Kiritsis, D. (2012). Semantic data model for operation and maintenance of the engineering asset. International Conference on Advances in Production Management Systems, Rhodes Island, Greece.

[Krima et al., 2009] Krima, S., Barbau, R., Fiorentini, X., R., S., Foufou, S., and Sriram, R. (2009). OntoSTEP : OWL-DL ontology for STEP. International Conference of Product Lifecycle Management, Bath, UK.

[Krima et al., 2012] Krima, S., Feeney, A., and Foufou, S. (2012). Dynamic customization and validation of product data models using semantic web tools. International Conference on Product Lifecycle Management, Montreal, Canada.

[Landherr and Constantinescu, 2012] Landherr, M. and Constantinescu, C. (2012). Intelligent management of manufacturing knowledge : Foundations, motivation scenario and roadmap. volume 3, pages 269–274. 45th CIRP Conference on Manufacturing Systems.

[Lee et al., 2005] Lee, I. H., Lee, S., Lee, T., Lee, S. G., Kim, D., Chun, J., and Shim, J. (2005). Practical issues for building a product ontology system. International Workshop on Data Engineering Issues in E-Commerce.

[Lee et al., 2009] Lee, J., Chae, H., Kim, C., and Kim, K. (2009). Design of product ontology architecture for collaborative enterprises. *Expert Systems with Applications*, 36 :2300–2309.

[Lee et al., 2010] Lee, J., Fenves, S., Bock, C., Suh, H., Rachuri, S., Fiorentini, X., and Sriram, R. (2010). A semantic product modeling framework and language for behavior evaluation. Technical report, National Institute of Standards and Technology.

[Lee and Jeong, 2012] Lee, J. and Jeong, Y. (2012). User-centric knowledge representations based on ontology for aec design collaboration. *Computer-aided design*, 44 :735–748.

[Lee and Jeong, 2006] Lee, S. and Jeong, Y. (2006). A system integration framework through development of ISO 10303-based product model for steel bridges. *Automation in construction*, 15 (2) :212–228.

[Lim et al., 2011] Lim, S., Liu, Y., and Lee, W. B. (2011). A methodology for building a semantically annotated multy-faceted ontology for product family modeling. *Advanced Engineering Informatics*, 25 :147–161.

[Lin and Harding, 2007] Lin, H. K. and Harding, J. A. (2007). A manufacturing system engineering ontology model on the semantic web for inter-enterprise collaboration. *Computers in Industry*, 58(5) :428–437.

[Louis-Sidney, 2011] Louis-Sidney, L. (2011). *Modèles et outils de capitalisation des connaissances en conception : Contribution au management et à l'ingénierie des connaissances chez Renault - DCT*. PhD thesis, Ecole Centrale Paris, France.

[Louis-Sidney et al., 2012] Louis-Sidney, L., Cheutet, V., Lamouri, S., Puron, O., and Mezza, A. (2012). A conceptual model for the implementation of IKOES (an inter-knowledge objects exchange system) in automotive industry. *Journal of Engineering Applications of Artificial Intelligence.*, 25(5) :1090–1101.

[Lu et al., 2010] Lu, Y., Panetto, H., and Gu, X. (2010). Ontology approach for the interoperability of networked enterprises in supply chain environment. 5th IFAC/IFIP EI2N Conference, Hersonissou, Greece.

[Maedche, 2002] Maedche, A. (2002). Ontology learning for the semantic web. *Journal of Intelligent Systems, IEEE*, 16 (2) :72–79.

[Matsokis, 2010] Matsokis, A. (2010). *An Ontology-Based Approach for Closed-Loop Product Lifecycle Management*. PhD thesis, Ecole polytechnique fédérale de Lausanne.

[Matsokis and Kiritsis, 2010] Matsokis, A. and Kiritsis, D. (2010). An ontology-based approach for product lifecycle management. *Computers in industry*, 61 :787–797.

[Matsokis and Kiritsis, 2011] Matsokis, A. and Kiritsis, D. (2011). Ontology applications in PLM. *International Journal of Product Lifecycle Management*, 5 :84–97.

[Merdan et al., 2010] Merdan, M., Lepuschitz, W., Meurer, T., and Vincze, M. (2010). Towards ontology-based automated disassembly systems. pages 1392–1397. Annual Conference on IEEE Industrial Electronics Society, Glendale, AZ, USA.

[Mikos et al., 2011] Mikos, W. L., Ferreira, J. C., Botura, P. E., and Freitas, L. S. (2011). A system for distributed sharing and reuse of design and manufacturing knowledge in the pfmea domain using a description logics-based ontology. *Journal of Manufacturing Systems*, 30(3) :133–143.

[Milanovic et al., 2007] Milanovic, M., Gasevic, D., Giurca, A., Wagner, G., and Devedzic, V. (2007). Sharing ocl constraints by using web rules. International Workshop on Modelling Systems with OCL in the MODELS'07 Conference.

[Morabito et al., 2001] Morabito, J., Sack, I., and Bhate, A. (2001). Organisation modelling, innovative architectures for the 21st century. NJ : Prentice Hall.

[Morbach et al., 2009] Morbach, J., Wiesner, A., and Marquardt, W. (2009). OntoCAPE : A (re)usable ontology for computer-aided process engineering. *Computers and Chemical Engineering*, 33(10) :1546–1556.

[Motik et al., 2007] Motik, B., Shearer, R., and Horrocks, I. (2007). Optimized reasoning in description logics using hypertableaux. *Journal of Automated Deduction - CADE*, 4603 :67–83.

[Mun et al., 2008] Mun, D., Hwang, J., Han, S., Seki, H., and Yang, J. (2008). Sharing product data of nuclear power plants across their lifecycles by utilizing a neutral model. *Anals of Nuclear Energy*, 35 :175–186.

[Mun et al., 2006] Mun, D., Lee, S., Kim, B., and Han, S. (2006). ISO 15926-based data repository and related web services for sharing lifecycle data of process plants. volume 38, pages 713–725. International Conference on Product Lifecycle Management, Bath, UK.

[Mun and Ramani, 2011] Mun, D. and Ramani, K. (2011). Knowledge-based part similarity measurement utilizing ontology and multi-criteria decision making technique. *Advanced Engineering Informatics*, 25 :119–130.

[Nanda et al., 2006] Nanda, J., Simpson, T., Kumara, S., and Shooter, S. (2006). A methodology for product family ontology development using formal concept analysis and web ontology language. *Journal of Computing and Information Science in Engineering*, 6 :103–113.

[Nazarenko et al., 2011] Nazarenko, A., Guissé, A., Lévy, F., Omrane, N., and Szulman, S. (2011). Integrating written policies in business rule management systems. Rule-Based Reasoning, Programming, and Applications. 5th International Symposium, RuleML 2011-Europe, Barcelona, Spain.

[Noël and Roucoules, 2008] Noël, F. and Roucoules, L. (2008). The PPO design model with respect to digital enterprise technologies among product life cycle. *International Journal of Computer Integrated Manufacturing*, 21 (2) :139–145.

[Nonaka and Takeuchi, 1995] Nonaka, I. and Takeuchi, H. (1995). *The Knowledge-Creating Company*. Oxford University Press.

[Noy, 2003] Noy, N. (2003). What do we need for ontology integration on the Semantic Web position statement. International Semantic Web Conference, Florida, USA.

[Noy and McGuiness, 2001] Noy, N. and McGuiness, D. (2001). *Ontology development 101 : A guide to creating your first ontology*. Standford University.

[Obitko, 2013] Obitko, M. (2013). www.obitko.com.

[Oh and Shin, 2012] Oh, S.-C. and Shin, J. (2012). A semantic e-kanban system for network-centric manufacturing : technology, scale-free convergence, value and cost-sharing considerations. *International Journal of Production Research*, 50 (19) :5292–5316.

[OMG, 2009] OMG (2009). Ontology Definition Metamodel (ODM). Object Management Group.

[OMG, 2012] OMG (2012). Object Constraint Language (OCL), version 2.3.1. Object Management Group.

[Orlicky, 1973] Orlicky, J. (1973). Net change material requirements planning. *IBM Systems Journal*, 12 (1) :2–29.

[Ottens, 2007] Ottens, K. (2007). *Un système multi-agent adaptatif pour la construction d'ontologies à partir de textes*. PhD thesis, Université Paul Sabatier, Toulouse III, France.

[Özacar et al., 2011] Özacar, T., Öztük, O., and Ünalir, M. (2011). ANEMONE : an environement for modular ontology development. *Journal of Data and knowledge engineering*, 70 :504–526.

[Pandit and Zhu, 2007] Pandit, A. and Zhu, Y. (2007). An ontology-based approach to support decision-making for the design of ETO (Engineer-To-Order) products. *Journal of Automation in construction*, 16 :759–770.

[Panetto, 2006] Panetto, H. (2006). Meta-modèles et modèles pour l'intégration et l'interopérabilité des applications d'entreprises de production. Mémoire d'Habilitation à Diriger des Recherches. Université Henri Poincaré - Nancy 1.

[Panetto et al., 2012] Panetto, H., Dassisti, M., and Tursi, A. (2012). ONTO-PDM : Product-driven ONTOlogy for Product Data Management interoperability within manufacturing process environment. *Knowledge based engineering to support complex product design*, 26(2) :334–348.

[Paquet, 2006] Paquet, P. (2006). *De l'information à la connaissance*. Université d'Orléans - Cahiers de recherche.

[Pardo et al., 2012] Pardo, C., Pino, F.J., G. F., Piattini, M., and Baldassarre, M. (2012). An ontology for the harmonization of multiple standards and models. *Computer Standards and Interfaces*, 18(17) :2493–2513.

[Paviot, 2010] Paviot, T. (2010). *Méthodologie de résolution des problèmes d'interopérabilité dans le domaine du Product Lifecycle Management*. PhD thesis, École Centrale Paris.

[Paviot et al., 2010] Paviot, T., Cheutet, V., and Lamouri, S. (2010). Semantic tags for generative multiview product beakdown. International Conference on Product Lifecycle Management, Bremen, Germany.

[Paviot et al., 2011] Paviot, T., Lamouri, S., and Cheutet, V. (2011). A generic multi-CAD/multiPDM interoperability framework. *International Journal of Services Operations and Informatics*, 6(1-2) :124–137.

[Pinto et al., 2004] Pinto, H., Staab, S., and Tempich, C. (2004). Diligent : Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In *ECAI*, volume 16, page 393.

[Pintzos et al., 2012] Pintzos, G., Matsas, M., and Chryssolouris, G. (2012). Defining manufacturing performance indicators using semantic ontology representation. CIRP Conference on Manufacturing Systems.

[Polanyi, 1966] Polanyi, M. (1966). *The tacit dimension*. The University of Chicago Press.

[Pratt et al., 2005] Pratt, M., Anderson, B., and Ranger, T. (2005). Towards the standardized exchange of parameterized feature-based CAD models. *Computer-Aided Design*, 37(12) :1251–1265.

[Preuveneers et al., 2004] Preuveneers, D., Van Den Bergh, J., Wagelaar, D., Georges, A., Rigole, P., Clerckx, T., Berbers, E., Coninx, K., and De Bosschere, K. (2004). Towards an extensible context ontology for ambient intelligence. Second European Symposium on Ambient Intelligence.

[Price and Bodington, 2004] Price, D. and Bodington, R. (2004). Applying semantic web technology to the life cycle support of complex engineering assets. *The Semantic Web*, 3298 :812–822.

[Queralt et al., 2012] Queralt, A., Artale, A., Calvanses, D., and Teniente, E. (2012). OCL-Lite : Finite reasoning on UML/OCL conceptual schemas. *Data and Knowledge Engineering*, 73 :1–22.

[Rachuri et al., 2008] Rachuri, S., Subrahmanian, E., Bouras, A., Fenves, S., Foufou, S., and Sriram, R. (2008). Information sharing and exchange in the context of product lifecycle management : Role of standards. *Computer-Aided Design*, 40(7) :789–800.

[Ramos-Garcia, 2010] Ramos-Garcia, L. (2010). Ontological CAD data interoperability framework. International Conference on Advances in Semantic Processing.

[Raza et al., 2011] Raza, M., Kirkham, T., Harrison, R., and Reul, Q. (2011). Knowledge-based flexible and integrated PLM system at Ford. *Journal of Information and Systems management*, 1 (1) :8–16.

[Rossellò-Busquet et al., 2011] Rossellò-Busquet, A., Brewka, J., and Dittman, L. (2011). OWL ontologies and SWRL rules applied to energy management. International Conference on Modelling and Simulation.

[Segonds, 2012] Segonds, F. (2012). *Contribution à l'intégration d'un environnement collaboratif en conception amont de produits*. PhD thesis, Arts et Métiers ParisTech, France.

[Song et al., 2013] Song, F., Zacharewicz, G., and Chen, D. (2013). An ontology-driven framework towards building enterprise semantic. *Advanced Engineering Informatics*, 27 :38–50.

[Sorensen et al., 2010] Sorensen, K. B., Christiansson, P., and Svidt, K. (2010). Ontologies to support RFID-based link between virtual models and construction components. *Computer Aided Civil and Infrastructure Engineering*, 25(4) :285–302.

[Staab et al., 2001] Staab, S., Schnurr, H., Studer, R., and Sure, Y. (2001). Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16(1) :26–34.

[Staab and Studer, 2009] Staab, S. and Studer, R. (2009). *Handbook on Ontologies*.

[Stark, 2011] Stark, J. (2011). *Product Lifecycle Management - 21st Century Paradigm for Product Realisation*. Springer Verlag - London.

[Stonebraker, 1996] Stonebraker, P. (1996). Restructuring the bill of material for productivity : A strategic evaluation of product configuration. *International Journal of Production Economics*, 45 :251–260.

[Suárez-Figueroa, 2010] Suárez-Figueroa, M. (2010). *NeOn Methodology for building ontology networks : specification, scheduling and reuse*. PhD thesis, Informatica, Spain.

[Sun et al., 2012] Sun, H., Fan, W., Shen, W., and Xiao, T. (2012). Ontology-based interoperation model of collaborative product development. *Journal of Network and Computer applications*, 1 :132–144.

[Tao et al., 2011] Tao, C., Parker, C., Oniki, T., Pathak, J., Huff, S., and Chute, C. (2011). An owl meta-ontology for representing the clinical element model. pages 1372–1381. AMIA annual symposium.

[Tao, 2012] Tao, J. (2012). *Integrity constraints for the semantic web : an OWL2 DL extension*. PhD thesis, Faculty of Rensselaer Polytechnic Institute.

[Terzi et al., 2010] Terzi, S., Bouras, A., Dutta, D., Garetti, M., and Kiritsis, D. (2010). Product lifecycle management - from its history to its new role. *International Journal of Product Lifecycle Management*, 4 (4) :360–389.

[Tsarkov and Horrocks, 2006] Tsarkov, D. and Horrocks, I. (2006). Fact++ description logic reasoner : System description. *Journal of Automated Reasoning*, 4130 :292–297.

[Tsuchiya, 1993] Tsuchiya, S. (1993). Improving knowledge creation ability trough organizational learning. *Proceedings of International Symposium on the Management of Industrial and Corporate Knowledge, ISMICK, Compiègne, France*.

[Tursi et al., 2009] Tursi, A., Panetto, H., Morel, G., and Dassisti, M. (2009). Ontological approach for product-centrics information system interoperability in networked manufacturing enterprises. *Annual Reviews in Control*, 33 (1) :238–245.

[Uschold and King, 1995] Uschold, M. and King, M. (1995). Towards a methodology for building ontologies. IJCAI95's workshop on basic ontological issues in knowledge sharing.

[Van Assche et al., 1988] Van Assche, F., Layzell, P., Loucopoulos, P., and Speltincxs, G. (1988). Information systems development : a rule-based approach. *Knowledge-based Systems*, 1(4) :227–234.

[van Ruijven, 2013] van Ruijven, L. (2013). Ontology for systems engineering. volume 16, pages 383–392. Conference on Systems Engineering Research (CSER?13), Georgia Institute of Technology, Atlanta, GA,.

[Vegetti et al., 2011] Vegetti, M., Leone, H., and Henning, G. (2011). PRONTO : An ontology for comprehensive and consistent representation of product information. *Engineerings Applications of Artificial Intelligence*, 24(8) :1305–1327.

[Wang and Yu, 2011] Wang, Q. and Yu, X. (2011). Reasoning over OWL/SWRL ontologies under CWA and UNA for industrial applications. *Advances in Artificial Intelligence*, 7106 :789–798.

[Wegner, 1996] Wegner, P. (1996). Interoperability. *ACM Computing Survey*, 28 (1) :258–287.

[Wicaksono et al., 2011] Wicaksono, H., Schubert, V., Rogalski, S., Ait Laydi, Y., and Ovtcharova, J. (2011). Ontology-driven requirements elicitation in product configuration systems. International Conference on Changeable, Agile, Reconfigurable and Virtual Production, Canada.

[Wilson and Schooler, 1991] Wilson, T. and Schooler, J. (1991). Thinking too much : introspection can reduce the quality of preferences and decisions. *Journal of personality and social psychology*, 60 :181–192.

[Winter and Schelp, 2006] Winter, R. and Schelp, J. (2006). Reference modeling and method construction : a design science perspective. pages 1561–2. Proceedings of the 2006 ACM Symposium on Applied Computing, Dijon, France.

[Yan et al., 2010] Yan, J., Ye, K., Wang, H., and Hua, Z. (2010). Ontology of collaborative manufacturing : Alignment of service-oriented framework with service-dominant logic. *Expert Systems with Applications*, 37(3) :2222–2231.

[Yang et al., 2008] Yang, D., Miao, R., Wu, H., and Zhou, Y. (2008). Product configuration knowledge using ontology web language. *Expert Systems with Applications*, 40 :863–878.

[Young et al., 2010] Young, R., Chungoora, N., Anjum, Z. U. N., Gunendran, G., Palmer, C., Harding, J. A., and Cutting-Decelle, A. F. (2010). An exploration of foundation ontologies and verification methods for manufacturing knowledge sharing. page 83. Proceedings of the Workshops and the Doctorial Symposium of the I-ESA International Conference.

[Zellner, 2011] Zellner, G. (2011). A structured evaluation of business process improvement approaches. *Business Process Management Journal*, 17(2) :203–237.

[Zhang and Yin, 2008] Zhang, W. Y. and Yin, J. W. (2008). Exploring semantic web technologies for ontology-based modeling in collaborative engineering design. *International journal of advanced manufacturing technology*, 36 :833–843.

[Zhao and Liu, 2008] Zhao, W. and Liu, J. (2008). OWL/SWRL representation methodology for EXPRESS-driven product information model. *Computers in industry*, 59 :580–589.

[Zimmermann, 2008] Zimmermann, A. (2008). *Sémantique des réseaux de connaissances : gestion de l'hétérogeneite fondée sur le principe de médiation*. PhD thesis, Université Joseph Fournier - Grenoble 1.

TABLE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ACRONYMES

APS : Advanced Planning System - *Système d'information de plannification (production).*
DAML : DARPA Agent Markup Language - *Langage ontologique précurseur de OWL.*
DRP : Distribution Resource Planning - *Système d'information pour la gestion de la distribution.*
ERP : Enterprise Resource Planning - *Système d'information de production.*
IHM : Interface Homme-Machine.
IRI : Internationalized Resource Indentifier - *Généralisation et internationalisation de l'URI.*
MES : Manufacturing Execution System - *Système d'information de production.*
NIST : National Institute for Standard and Technology - *Agence gouvernementale américaine travaillant sur les standards et les techonologies de l'information.*
OCL : Object Constraint Language - *Langage de règles utilisé pour les modèles UML.*
OIL : Ontology Inference Layer ou Ontology Interchange Language - *Langage ontologique précurseur de OWL.*
OWL : Web ontology language - *Langage standard pour la modélisation d'ontologies d'inférences, fondé sur une logique DL.*
PDM : Product Data Management - *Type de système d'information regroupant l'ensemble des informations de conception, et notamment les nomenclatures produit.*
PLCS : Product Life Cycle Support - *Protocole d'application AP 239 de la norme STEP.*
PLM : Product Lifecycle Management - *Méthodologie visant à gérer l'ensemble des informations liées au produit tout au long de son cycle de vie.*
PSL : Process Specification Language - *Modèle standard pour la description des activités de production.*
RDF : Resource Description Framework - *Modèle de graphes pour la représentation d'ontologies.*
RDFS : RDF Schema - *Langage standard pour la modélisation d'ontologies sous forme de triplets.*
RDL : Reference Data Library - *Bibliothèque de données de référence, construite pour supporter sémantiquement l'échange d'informations.*
Rule ML : Rule Markup Language - *Langage de règles fondé sur une syntaxe XML.*

SBVR : Semantics of Business Vocabulary and Business Rules - *Standard OMG pour la formalisation de règles en langage semi-naturel.*
SCOR : Supply Chain Operations Reference - *Modèle standard pour la supply chain globale.*
SI : Systèmes d'Information.
SPARQL : SPARQL Protocol and RDF Query Language - *Langage de requête pour RDF.*
SPIN : SPARQL Inferencing Notation - *Extension de SPARQL, qui permet notamment de maintenir et de partager les requêtes SPARQL, ainsi que de construire des templates de requêtes.*
STEP : - *Modèle ISO 10303 pour la modélisation standard du produit.*
SWRL : Semantic Web Rule Language - *Langage alliant la modélisation OWL-DL et les règles Rule ML.*
TMS : Transport Management System - *Système d'information pour la gestion du transport.*
UML : Unified Modeling Langage - *Langage standard de modélisation.*
URI : Uniform Resource Identifier
W3C : World Wide Web Consortium - *International community that standardizes web languages*
XML : Extensible Markup Language - *Langage informatique à balises.*

Contribution à une modélisation ontologique des informations tout au long du cycle de vie du produit

RESUME : Les travaux de recherche de cette thèse portent sur la modélisation sémantique des informations industrielles, dans une approche « cycle de vie » de gestion des informations. Dans ce type d'approche, lever le verrou lié à l'interopérabilité des modèles d'information est une condition sine qua non à un échange d'information sans perte de flux sémantique. Jusqu'alors, des méthodes d'unification étaient envisagées, reposant sur l'utilisation mutuelle de modèles standards, tels que la norme STEP par exemple. Cependant, l'unification fait face à des limites en termes d'expressivité des modèles, de rigidité, et de perte de sémantique. Afin de lever ces limites, les paradigmes de modélisation évoluent et se tournent vers les ontologies d'inférence, outils du web sémantique.

Dans cette thèse, nous proposons un cadre de modélisation sémantique général et une méthodologie de mise en place de ce cadre, qui reposent sur l'utilisation d'ontologies d'inférence. L'application du cadre de modélisation à un cas d'étude industriel, issu de l'ingénierie nucléaire (plus particulièrement l'expression et l'exécution des règles métier), permet alors d'évaluer les apports et limites des ontologies en tant que paradigme de modélisation. Les limites les plus importantes que nous identifions sur l'Open World Assumption, le manque de langage de règles performant et le manque d'outils d'implémentation robustes pour des applications à large échelle. Le développement d'un démonstrateur pour le cas d'étude industriel permet finalement de tendre vers une solution mixte, où les ontologies sont utilisées localement, afin d'en exploiter les divers avantages de manière optimale.

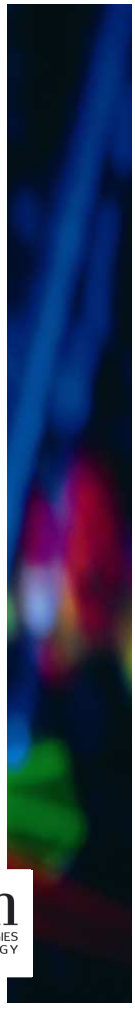
Mots clés : ontologies, PLM, modélisation sémantique, règles métier.

Towards an ontology-based model for product information along the product lifecycle

ABSTRACT : The present research study deals with industrial information modelling in a lifecycle management approach. In this context, achieving semantic interoperability is an important issue to guaranty the quality of information storage, exchange and reuse. Current solutions are based on unification approaches and use standard models as shared information representation. However, unification approches suffer from a lack of expressivity and flexibility. To overcome this issue, ontologies are proposed as a new modelling paradigm in order to perform federativ interoperability.

This research focuses on inference ontologies and investigates the use of those semantic web-based technologies for a large scale industrial application. An ontology-based framework and a modelling methodology are therefore proposed and evaluated on a industrial use case in the nuclear industry. The application raises modelling issues like the Open World Assumption, the lack of a real integrated rule approach, and the robustness of implementation tools. The industrial use case is implemented within a demonstrator, that finally allows to propose an hybrid solution, where ontologies are locally used.

Keywords : ontology, PLM, semantic modeling, business rules.



ARTS ET MÉTIERS ParisTech

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY