Table des matières

AXIS-CSRM 04 : Le modèle de DONNÉES AXIS-CSDM (Conceptual Semantic Data Model) : 2

Introduction : 2

Les constructions conceptuelles de base et leurs représentations graphiques. 7

Le concept de 'Construct' : 7

Le concept de Classes : 8

Les CLASSES dans CSDM : 10

Les Super CLASSES dans CSDM : Entity et Configuration : 10

Les Technologies de Modélisation (à valider) : 13

Instances de classes et utilisation des Technologies de Modélisation 13

Propriétés des classes et utilisation des Technologies de Modélisation 14

Réification de Properties 17

Les relations 20

Réification de Relationships 20

Autres Constructions conceptuelles des Entités et de leurs représentations graphiques. 22

Modélisation générale des « Entités » 22

Représentations graphiques plus détaillées 26

Déclaration d'Existence des « Entités » & de leurs « Modèles » 29

Les "Registres" : 33

La modélisation à « États Finis » : les « Segments » & les « Points » 35

La modélisation des « Entités » et des « Métadonnées » 37

La modélisation des « Entités » : 40

Modèles sémantiques 41

Modèles sémantiques hybrides : 42

Association du « Modèle de données » avec le « Modèle Fonctionnel » 43

Actuality 45

Working 45

Concepts <à développer> 45

Détails des Constructs <à développer> 46

Introduction à la définition formelle des Constructs 46

Définition formelle des Constructs d'AXIS-CSDM : <à développer> 46

Suggestions de choix pour les Technologies de Modélisation <à développer> 46

Exemple d'application des Constructs <à développer> 47

AXIS-CSRM 04 : Le modèle de DONNÉES AXIS-CSDM (Conceptual Semantic Data Model) :

Le modèle conceptuel de données sémantiques // **C**onceptual **S**emantic **D**ata **M**odel [CSDM] est l'un des huit modèles d'AXIS-CSRM. Comme son nom l'indique, le CSDM est une ontologie conceptuelle. C'est une définition conceptuelle des **Constructs [de Classes, de Relations, de Propriétés** ... et de **Vocabulaire]** qui pourrait servir de base à la définition d'Ontologies Racines (Upper-Ontologies) selon les normes spécifiques de représentation des technologies de l'information (TIC) et à travers des outils appropriés.

Le modèle de données est vraiment synergétique avec le modèle fonctionnel. De la même façon que dans la norme OAIS le modèle fonctionnel et le modèle de données sont indissociables, mais ils correspondent chacun à une façon de modéliser la réalité, de regarder les moyens informatiques de deux façons différentes, mais tout à fait complémentaires.

Le modèle de données se préoccupe d'exprimer comment organiser les données pour représenter les concepts. Ce modèle conceptuel va organiser comment représenter les données de manière sémantique au niveau des EXPRESSIONs, des MANIFESTATIONs et enfin des ITEMs.

Dans AXIS-CSRM il ne s'agit nullement de produire un modèle de données lié à l'exploitation proprement dite. La focalisation porte sur le moment où s'effectue l'échange des données. L'échange porte sur des Packaged-Autonomous eXchange Entities (ou P-AXES) ou des AXEs car ces derniers sont toujours ''packagés'' d'une façon spécifique.

Dans la terminologie OAIS, les AXEs représentent à la fois le DIP (Dissemination Information Package) quand c'est un des ARD (Autonomous Resource Domain) qui émet, et il est vu comme un SIP (Submission Information Package) par le domaine qui le reçoit. Dans le cas inverse, lorsqu'un domaine réagit il renvoie à l'autre un DIP qui est vu par le domaine qui le reçoit comme un SIP.

Dans le cas où l'ARD réclame des informations, le dialogue est géré par des protocoles. Les données relatives au Configuration Management Protocol sont aussi des éléments du modèle de données d'AXIS-CSRM.

Enfin, il est nécessaire que les citoyens (citizen) puissent voir, écouter, entendre, donc avoir des perceptions de ce qui est représenté à l'intérieur de l'informatique. L'objet même du modèle de données est de fournir les moyens de traitement des données en vue de réaliser la perception. Dans le cas d'un fichier.wav en vue de reproduire le son via un haut-parleur, il faut suivre un ensemble de processus liés à l'usage de protocoles. Idem pour la représentation d'un fichier MPEG2 en vue d'un affichage sur un écran.

Dans AXIS-CSRM la logique vise à savoir comment créer des AXEs qui puissent être à la fois des SIP et des DIP tout en sachant qu'à l'intérieur des ARD il puisse exister des structures de données opérationnelles propriétaires.

Lors d'échange d'AXEs entre domaines, il subsiste des traces de l'origine des informations extraites. En effet, les protocoles requièrent d'échanger des informations relatives à des emplacements (URL) alors que dans les AXEs, à priori, il n'y a que des URN. Les seules URL dont on dispose sont celles qui indiquent où se situent les Configuration Management Services nécessaires pour retrouver les bits en question.

Combinés, le 'Modèle Conceptuel Sémantique de Données' [CSDM et le 'Modèle Fonctionnel' [FRAR] constituent le noyau dur, la racine des huit modèles d'AXIS-CSRM.

Il est pertinent de reproduire ici préalablement les définitions ISO de "Données" et "Information":

- **DONNÉES** : une représentation de faits, de concepts ou d'instructions, d'une manière formalisée, appropriée pour la communication, l'interprétation ou le traitement par des moyens **humains** ou des **moyens automatiques**.
- **INFORMATION :** le sens que l'humain assigne à la DONNÉE au moyen de conventions appliquées aux DONNÉES.

Introduction :

Un bon «Data Model» est essentiel pour tout modèle conceptuel lié à l'interchange et à l'archivage : en effet, ce sont des données que l'on échange entre domaines ce que dans AXIS-CSRM nous appelons des ARD ou ce sont des données qui sont présentées en perception aux humains. Le data model d'AXIS-CSRM ne gère pratiquement que des concepts d'organisation des données, de leur évolution, de leur structuration, de matérialisation / incorporation [embodiement] (de Manifestation à Item et vice-versa). Il tient compte du besoin

de déclarer l'existence de choses physiques, de pouvoir les représenter de nombreuses manières et du besoin que ces choses puissent être «statiques» ou en «flux».

À ce propos du flux, le poète Henry Bauchau[1] a un vers fameux qui a influencé bien des artistes (et, en particulier, Pierre Bartolomée[2] dans son Requiem [CYP-1655]) : « Le chant de l'alouette ne vieillit pas ! »). En effet le son entendu quand l'alouette chante évolue dans la durée, alors que son enregistrement est statique ! Pour que l'enregistrement puisse s'exprimer, il doit être transformé en flux sonore : à peine émis, le son s'éteint dans l'espace. Si vous vous bouchez les oreilles en l'écoutant, il ne reste rien.

AXIS-CSRM ne se préoccupe pas d'imposer une façon particulière de modéliser les Entités. Au contraire, AXIS-CSRM a été construit pour accueillir autant de façons de modéliser les Entités que l'imagination ou le besoin établira. Dans certains cas, ces modèles seront exprimés suivant les technologies sémantiques ; dans d'autres cas, AXIS-CSRM déclarera l'existence et la disponibilité d'un modèle non exprimé sémantiquement : il y aura l'identification du modèle ainsi que les paramètres d'instanciation (s'il y en a).

Le Data Model d'AXIS-CSRM ne peut pas, à proprement parler, être considéré comme une « upper ontology », au sens où il s'efforce d'être orthogonal de ce qui veut être déclaré ou représenté. C'est ce qui fait sa force. Il possède par ailleurs tous les crochets sémantiques pour intégrer toutes les façons voulues de déclarer quelque chose de spécifique suivant toute approche pertinente dans le cas. C'est dans les 'Profiles' que cette vision particulière sera exprimée et liée à l'ensemble des instances de classes mobilisées pour exprimer formellement cette vision.

Cette section vise à introduire et motiver les concepts principaux de ce Modèle de Données. Ces concepts seront détaillés ensuite dans les sous-sections qui suivent.

La mise en œuvre du modèle de donnée s'entendra sous forme de définitions et d'occurrence des constructs des ontologies : **Classes** ; **Relationships** et **Properties**. Il sera montré plus loin que cette mise en œuvre pourra être réalisée sur base de normes internationales. Plusieurs expressions de mise en œuvre du modèle de données conceptuel « AXIS-CSDM » peuvent coexister : en effet, pour des raisons de performance (par exemple) certains aspects conceptuels pourraient être implémentés via des structures de données plus simples qui seraient complétées par des processus ou des mécanismes d'inférences tels qu'en pratique les fonctions soient remplies correctement, mais au prix d'une diminution de la lisibilité. Cet aspect sera illustré plus loin dans le cas de la réification des Properties.

Cependant, comme on le verra dans le modèle d'Import & Export, AXIS-CSRM recommande que pour la création des Packages d'Exports [les **DIP** dans la terminologie de la norme OAIS] l'ontologie utilisée soit la plus proche possible du modèle conceptuel, de telle sorte que l'interopérabilité soit privilégiée par rapport à la performance. La clarté et la lisibilité sont les garants de transcodages guidés par la sémantique, c'est-à-dire que les Guichets d'Interopérabilité n'auront pas de perte sémantique dans la création des AIP dans le module fonctionnel d'Ingest (voir la norme OAIS **CCSDS 651.0-B-1**) au départ des **DIP** dont question ci-devant considérée par le module d'Ingest comme des **SIP**.

Une des originalités d'AXIS-CSDM est d'organiser, de structurer les données en explicitant la déclaration d'existence des **[Quelques Choses]** indépendamment de leurs représentations (des **Artefacts**) et de leurs activations. À leur tour, les Artefacts peuvent être vus comme des **[Quelques Choses]**. Il faut insister sur le fait que les [Quelques Choses] et les Artefacts peuvent être des moyens qui peuvent être activés pour produire des effets, des transformations des [Quelques Choses] ou des Artefacts (ce qu'usuellement l'on appelle des processus).

Les deux figures qui suivent illustrent ce schéma racine de CSDM, d'abord par un exemple ; puis par la généralisation :

[1] Henry Bauchau, né à Malines le 22 janvier 1913 et mort à Louveciennes (France) le 21 septembre 2012, est un psychanalyste belge, également poète, dramaturge et romancier de langue française.
[2] Le chevalier Pierre, Georges, Édouard Bartholomée, né le 5 août 1937 à Bruxelles est un chef d'orchestre, compositeur et pianiste belge.

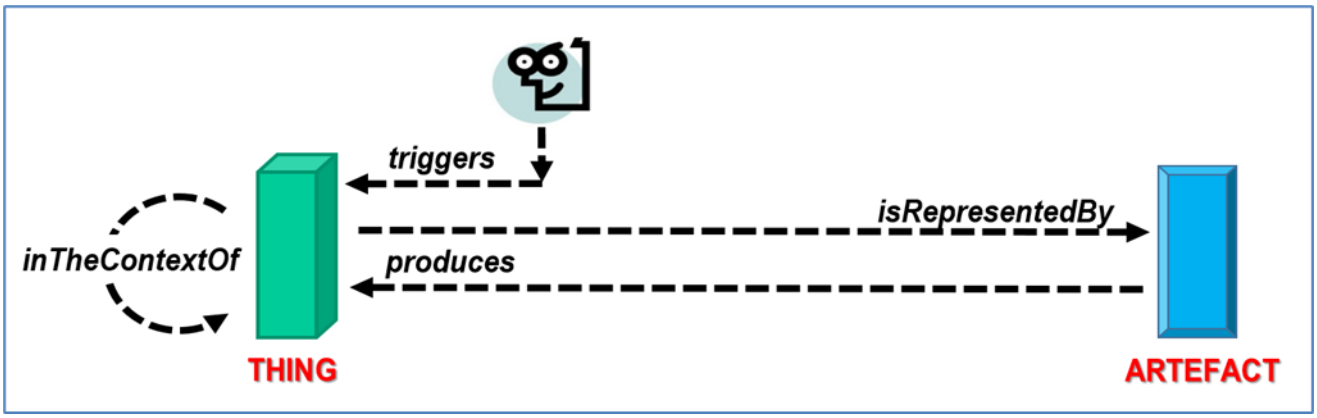

Figure 1 : le schéma ontologique racine de CSDM (exemple)

L'exemple montre que [Quelque Chose] [Thing] peut être représentée [*isRepresentedBy*] par un Artefact [Artefact] ; que [Quelque Chose] peut être activée [triggers] par une action Humaine ; qu'un Artefact peut être tel que si des ressources adéquates sont disponibles [inTheContextOf] et activées [triggers], [Quelque Chose] [Thing] peut être produit [produced].

La figure qui suit généralise l'exemple pour exprimer que la Relationship 'triggers' de l'exemple est une occurrence particulière ; de même pour 'produces' ou pour 'inTheContextOf' ou pour 'isRepresentedBy'.

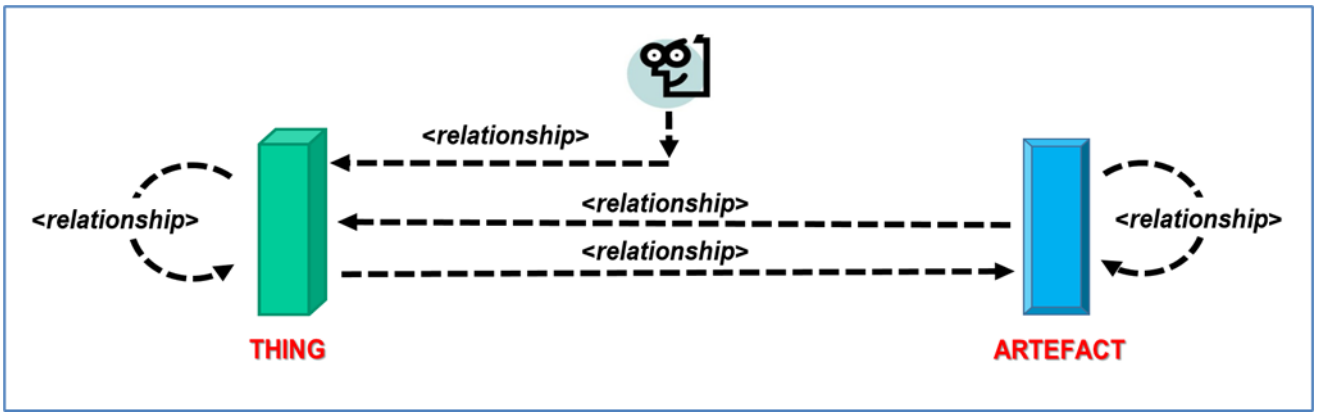

Figure 2 : le schéma ontologique racine CSDM (généralisation)

Lors des occurrences, il faut remplacer l'expression générique <relationship> par le terme de la Relationship sélectionnée.

AXIS-CSDM permet donc de modéliser la définition de l'existence des Choses et de les modéliser selon plusieurs Technologies de Modélisation. La définition (ontologie, base de connaissances associée ...) de ces technologies de modélisation [MT] est hors de la portée de la modélisation CSDM. Les illustrations des concepts induiront l'existence de ces cas triviaux.

AXIS-CSDM est un cadre de travail commun. Les utilisateurs sont invités à, et devraient, enrichir davantage le modèle avec des Classes et des Propriétés répondant plus précisément à leurs besoins.

AXIS-CSDM a été conçu pour être directement lié aux autres modèles AXIS-CSRM, en particulier les modèles Fonctionnel et d'Identification.

La spécification AXIS-CSRM/CSDM permet aussi la réconciliation et la migration sémantique, au départ de bases de métadonnées existantes, potentiellement incohérentes. Les métadonnées dédiées de plusieurs bases de données pourraient être migrées et « élevées » sémantiquement : un exemple est donné dans le cas des métadonnées basées sur les éléments Dublin Core.

La seconde des originalités principales d'AXIS-CSDM est d'organiser, de structurer les données en **Constructs** permettant l'expression de deux structures distinctes :

1. La première concerne la modélisation des Choses : les **Entités**
2. La seconde concerne la structuration des Choses : les **Configurations**

La figure qui suit illustre cette approche :

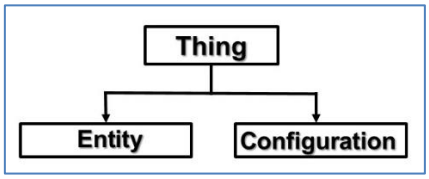

Figure 3: les Classes racines dans CSDM

Le concept de Classe sera explicité en 6.5.2.B. La figure qui suit illustre les articulations et les rôles respectifs des concepts d'Entité et de Configuration en vue de déclarer l'existence d'Entités ou de modèles d'Entités et de lier les modèles eux-mêmes et l'expression des structures internes et externes.

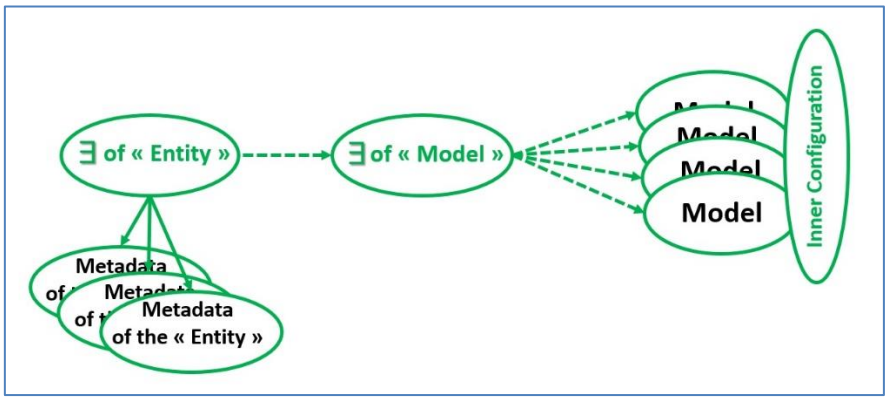

Figure 4 : la structure racine de la configuration des modèles dans CSDM

La modélisation des Choses est représentée par la couleur **noire** ! D'une part les Modèles [Model] qui représentent des Choses et d'autre part les ensembles de Métadonnées [Metadata] qui représentent les données qui sont nécessaires à modéliser la déclaration d'existence de choses. AXIS-CSDM ne se préoccupe pas d'imposer des façons de modéliser ; il considère que les façons de modéliser les Choses sont tout à fait libres. AXIS-CSDM se borne simplement à reconnaitre l'existence des façons de modéliser. Chaque façon de modéliser est appelée « Modelling Technology ».

La structuration des Choses est exprimée par la couleur **verte**. Les données représentant les structures sont disjointes des données qui représentent les modèles. AXIS-CSDM propose une Ontologie Racine [Upper Ontology] pour représenter la structuration des choses. Cette structuration commence toujours par une déclaration d'existence des Choses et déclaration d'existence de Modèles. Au niveau de détail de la structuration, AXIS-CSDM propose un mécanisme de représentation de structuration interne aux modèles ou de synchronisation entre les modèles. Ceci est représenté dans la figure par l'ellipse labélisée 'Inner Configuration'.

Après avoir défini les constructs de configuration dans AXIS-CSDM, les « Entités » sont les choses dont on parle à un moment donné dans une « Autonomous Exchange Entity ». La structure racine devient :

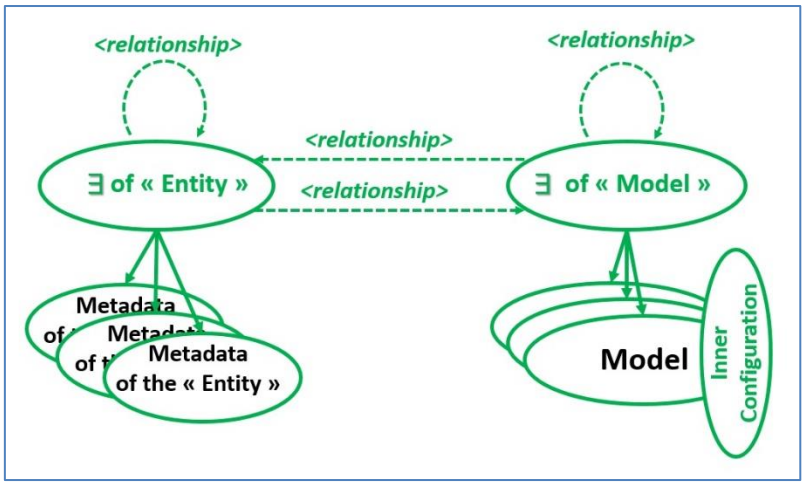

Figure 5 : la structure racine de la configuration des modèles des « Entités » dans CSDM

Notes sur les mises en œuvre d'AXIS-CSDM :

Comme cela a été dit plus avant, la mise en œuvre d'AXIS-CSDM implique de définir des ontologies telles que les AXEs puissent exporter des DIP suivant des formats les plus proches possible des Constructs conceptuels d'AXIS-CSDM. C'est pourquoi, inversement, AXIS-CSRM s'efforce de présenter les Constructs conceptuels de manière qu'un tel mapping du conceptuel sur le réel soit le plus facile.

Comme cela a été dit aussi, les Constructs permettant d'exprimer les 'Configurations' des Choses sont fondamentaux ! Une telle ontologie est introduite conceptuellement dans les sections qui suivent et son expression en OWL est présentée en annexe.

Pour réaliser des exemples, il est nécessaire de choisir une Ontologie des 'Entités'. Celle qui est introduite conceptuellement dans les sections qui suivent et son expression en OWL est présentée en annexe. Il faut insister sur le fait qu'AXIS-CSDM est ouvert à d'autres façons de modéliser les Entités.

La représentation des DIP au plus proche des concepts conduit à des Packages plus volumineux que ceux que l'on pourrait générer au départ des AIP ! Mais cela impliquerait que l'on suppose que les destinataires qui prendraient ces DIP comme des SIP dispose des mêmes outils, ressources, contextes issus de compromis de performance et de ciblage sur une « Designated Community ». Ce point sera illustré au point traitant de la 'Réification' des 'Properties'.

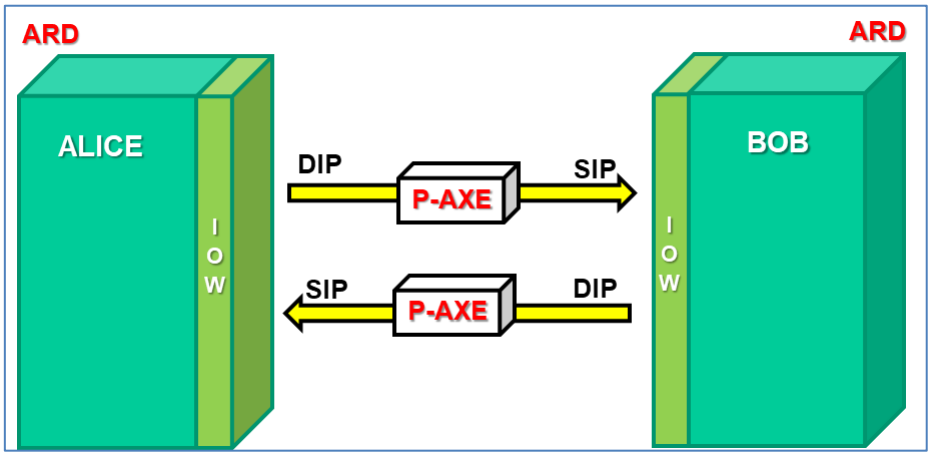

Figure 6 : l'interopérabilité entre deux ARD indépendants

La section 6.9 sera entièrement consacrée à détailler la structure des P-AXE et à la façon de les générer. Le caractère autonome des P-AXE implique que l'encapsulation incorporerait non seulement les données représentant les contenus transmis, mais aussi les données représentant les modèles de données définissant les technologies de représentation utilisées pour représenter les contenus et, le cas échéant, les outils applicables (tels que les outils d'ouverture du package et ceux d'expression des représentations des contenus).

La figure illustre le transfert d'un P-AXE [«package» encapsulant un AXE (Entité d'échange autonome)] du domaine de ressources autonome «ALICE» vers le domaine de ressources autonome «BOB»; et vice versa !

ALICE, par l'intermédiaire de son IOW [Inter Operability Wicket], et via un AIP [Archival Information Packages], publie un P-AXE au format AXIS CSRM; l'IOW de BOB traite le P-AXE pour le vérifier et l'inclure dans sa base de données AIP. Pour ALICE, le P-AXE est un DIP; et pour BOB, le P AXE est un SIP [paquet d'informations sur la soumission] SIP.

La mise en œuvre du Modèle Conceptuel Sémantique des Données consiste à exprimer les concepts via un mappage concret des concepts sur les Constructs offerts par les normes telles que RDF/OWL et Web 3/Données liées ouvertes. Cela n'implique pas que le mappage doit couvrir complètement le modèle de données conceptuel AXIS, ni construire l'ontologie IT avec un mappage où à chaque concept est associé une et une seule représentation technique.

La représentation de la «connaissance simple» (Thésaurus et Taxonomies) pourrait également être exprimée avec SKOS.

Protégé est un outil typique: https://protege.stanford.edu/

La spécification AXIS-CSRM/CSDM combine plusieurs aspects des spécifications et modèles existants dans un cadre de travail commun. En particulier:

- UER-Core EBU-CCDM: voir http://tech.ebu.ch/
- Le FRBR: https://www.ifla.org/publications/functional-requirements-for-bibliographic-records
- Le CIDOC-CRM: http://www.cidoc-crm.org/
- Le FRSAD: https://www.ifla.org/files/assets/classification-and-indexing/functional-requirements-for-subject-authority-data/frsad-final-report.pdf
- L'Ontologie ABC: voir http://dcpapers.dublincore.org/pubs/article/view/655/651

En outre, une introduction aux architectures CIM-OSA et SAP est présentée en annexe. L'un des auteurs (Guy Noël Maréchal) a été le coordinateur scientifique de ce projet CIM OSA, parrainé par l'Union européenne, qui a largement influencé la création de l'activité SAP. Il est important de mentionner qu'à ses débuts SAP n'a couvert ni la généricité[3] ni la récursivité[4] de la gestion des processus. Toutes ces limitations et ces pièges ont été résolus dans AXIS CSDM.

Les constructions conceptuelles de base et leurs représentations graphiques.

Le concept de 'Construct' :

CSDM déclare trois types de **Constructs** : les '**Classes**', les '**Propriétés**' et les **Relations** :

- Les **Classes** sont les Constructs porteurs de la déclaration d'existence des Choses, des modèles des Choses et de la définition des Concepts.
- Les **Propriétés** sont les Constructs qui vont documenter les Classes, en quelque sorte, les liens vers les métadonnées des Classes
- Les **Relations** sont les Constructs qui permettent de relier les Classes.

Ces trois types de Constructs permettent de modéliser sémantiquement les 'Déclarations d'existence' des Choses, les 'Modèles' des Choses et les définitions des Concepts.

L'approche d'AXIS-CSDM n'impose pas de structuration et définition particulière des Constructs, mais bien une approche conceptuelle basée sur un ensemble de Classes, Relationships et Properties conceptuelles qui, pour des applications spécifiques, pourraient être exprimées de manière adaptée aux contextes visés, en particulier pour des raisons de performance. Ce point sera analysé et illustré plus en détail plus loin.

La structuration et la définition des Constructs ne sont pas évidentes et ont des fondements philosophiques.

Afin de faciliter la compréhension et les élaborations d'expressions particulières, une structuration exemplative des concepts via des Constructs est présentée ci-après.

[3] La généricité (ou programmation générique), consiste à définir des algorithmes identiques opérant sur des données de types différents.

[4] La récursivité est une démarche dont la description mène à la répétition d'une même règle : Un dictionnaire de définitions constitue un bel exemple.

Le concept de Classes :

L'étape de base dans tout processus de description d'un modèle conceptuel tel qu'AXIS-CSRM est de formuler de façon générique le contenu et la représentation interne du modèle. Il s'agit de formuler une forme de dictionnaire (au service de l'Humain comme de la Machine), qui n'est pas construit sur la syntaxe des termes, mais bien sur l'interconnexion sémantique des concepts.

Un concept associe un double contenu sémantique : d'une part, l'ensemble des prédicats, dont la conjonction définit l' « Étant »[5] en évidence (le concept d'un triangle); et d'autre part, l'ensemble des objets dans le cas d'un triangle auxquels on peut faire référence via l'expression linguistique du terme triangle.

Pour la linguistique, il s'agit du concept de Signifié (la représentation mentale) qui se décline en « Dénotation » et « Connotation ».

En logique, l'intention (ou «compréhension») et l'extension sont deux façons de définir un concept :

- L'intention repose sur l'ensemble des caractères qui permettent de définir un concept, comme l'ensemble des prédicats qui appartiennent à un concept. L'intention définit les conditions nécessaires et suffisantes pour qu'un « Étant» (ex: la Belgique) appartiennent à l'extension d'un terme («Pays»).
- L'extension est la description de l'ensemble des «Étant» auxquels l'intention (la définition) peut s'appliquer, qui sont dotés de la forme signifiée/requise par le concept.

Passer d'une représentation conceptuelle à une représentation utilisable par les systèmes de technologies de l'information implique de choisir un langage technologique. À titre d'exemple, le Resource Description Framework [RDF] organise les définitions via des 'Schémas' qui désignent les « Choses » comme étant des CLASSES, ce sont des ensembles d'instances (le concept Pays est constitué de la France, l'Italie, les USA...). Une CLASSE est donc la description formelle d'un ensemble d'objets ayant une sémantique et des caractéristiques communes. C'est une Entité discrète dotée d'une identité, de propriétés, d'un état et d'un comportement que l'on peut invoquer.

Cette approche est équivalente à la notion de **classe** utilisée dans la modélisation web sémantique (voir RDF et OWL Primers), également appelée « Business Objects » ou « Concepts » dans certains projets, voir aussi :

- http://protege.stanford.edu/publications/ontology development / ontology101.pdf
- Media-Ontology du W3C (http://www.w3.org/ns/ma-ont.rdf) basé sur l'EBU-CCDM.

Une classe en RDF Schema correspond au concept générique d'un type ou d'une catégorie (proche de la notion de CLASSE dans les langages de programmation orientée objet). Les CLASSES RDF peuvent servir à représenter pratiquement n'importe quelle catégorie de Choses, telles que des concepts abstraits, des pages web, des personnes, des types de documents, des bases de données .... Les CLASSES sont construites à partir de descriptions qui contraignent les conditions d'appartenance d'une instance à une CLASSE ou d'une SOUS/SUR-CLASSE à une CLASSE.

Comme AXIS-CSRM a la prétention de définir des Concepts, le choix d'une forme de représentation est nodal en vue de pouvoir formuler de façon générique le contenu et la représentation interne du modèle. La représentation graphique des classes dans CSDM est un rectangle avec le nom de la classe inclus :

Figure 7 : La représentation de la Classe « PhysicalPerson »

Illustration : la CLASSE "PhysicalPerson" représentée suivant le modèle ci-dessus.
La CLASSE « PhysicalPerson » désigne collectivement tous les humains capables d'agir. La CLASSE « PhysicalPerson » peut être conçue comme un raffinement d'une (Super) CLASSE qui est « Person ».

[5] Dans AXIS-CSRM, l'on désigne par « Étant » (en majuscule) les choses (avec une minuscule) et les concepts (avec une minuscule) qui **sont**, ... que nous les percevions, concevions … ou non ! Lorsque nous les percevons ou que nous pouvons les conceptualiser, nous pouvons décider d'en déclarer formellement l'existence pour pouvoir décrire, représenter, partager, agir ….

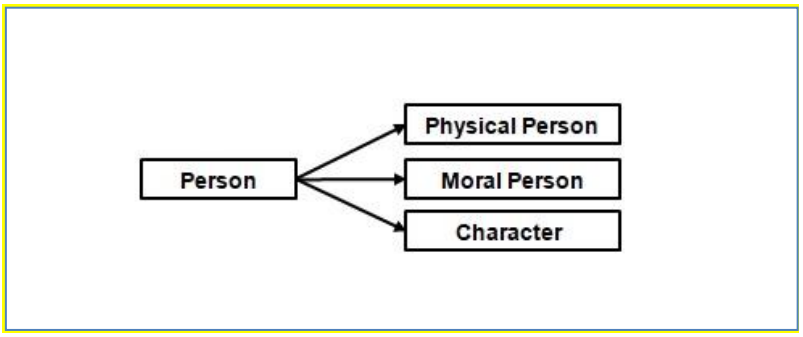

Figure 8 : La spécialisation de la Classe « Person »

Les flèches indiquent que « PhysicalPerson » est une spécialisation de « Person ». De même d'autres CLASSES peuvent être liées à « Person » comme :

- «MoralPerson» représentant les organisations et les entreprises
- « Character » les personnages fictifs (rôles dans les films, les romans, au théâtre, ...)

Pour AXIS-CSRM, chaque instance de PhysicalPerson est soumise dans le contexte des "droits de l'homme", c'est-à-dire des sujets de devoirs et de droits et, en tant que tels, responsables de leurs actes dans chacun des contextes spécifiques où ils sont exécutés.

Pour les besoins du modèle conceptuel, et comme d'autres CLASSES de ressources sont susceptibles d'agir, il est pratique de créer une super classe d'agents, appelée « Agent » dont les CLASSES « ActivatedResource » (représentant des objets habilités à effectuer des tâches comme des robots ...) et « Person » sont des raffinements d'« Agent » :

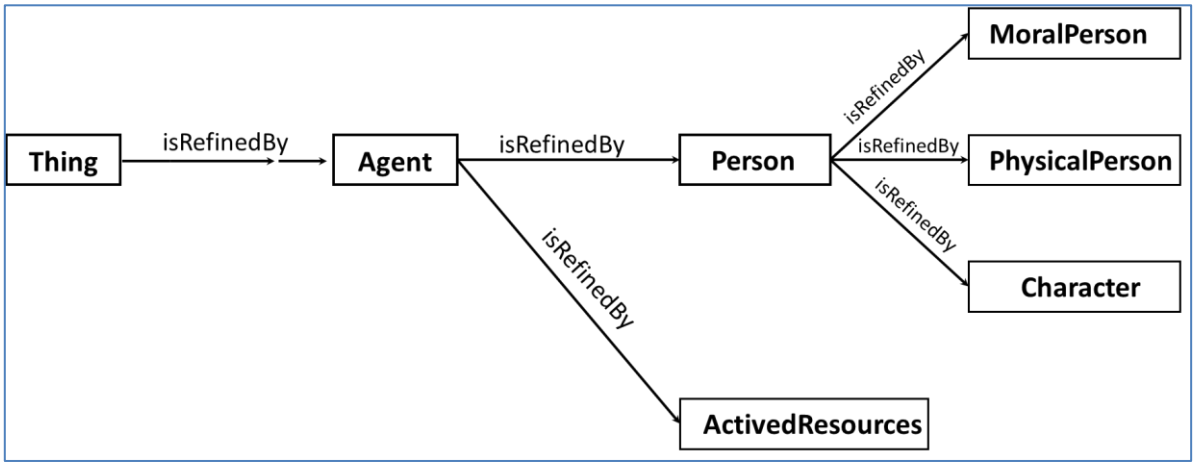

Figure 9 : La spécialisation progressive à partir de 'Thing'[Quelque Chose]

La figure ci-dessus illustre la spécialisation progressive à partir de « Thing » jusqu'à « PhysicalPerson » et avec les CLASSES associées. La flèche en pointillé entre «Thing» et «Agent» signifie que la spécialisation se produit à travers une chaîne de CLASSES dont certaines ne sont pas indiquées pour des raisons de simplification du schéma.

Comme d'autres Classes de ressources sont capables d'agir, il est pratique de créer une super Classe d'agents, nommée **Agent**! La Classe PhysicalPerson est un raffinement de l'agent de Classe. Les autres Classes en cours de perfectionnement d'Agent pourraient être :

- La Classe **MoralPerson** représentant des organisations et entreprises capables d'agir
- La classe **Character** représentant des personnes physiques fictives (rôles dans le théâtre, les films, les romans…)
- La classe **ActivatedResource** représentant des artefacts habilités à effectuer des tâches (robots ...).

Comme l'objectif est bien de constituer un dictionnaire dynamique basé sur les interrelations sémantiques des instances, il est nécessaire de définir les propriétés ou relations (Properties) sous la forme de relations binaires (Transitive, Symétrique, ...) afin de disposer d'une représentation générique.

Les CLASSES dans CSDM :

Dans CSDM, les Classes sont des raffinements de la CLASSE [Thing] [Quelque Chose]. Les Classes héritent des caractéristiques de leurs super CLASSES. Chaque Classe a un nom unique. Chaque instance d'une classe est supposée avoir un identifiant, unique dans le contexte local; les instances peuvent également avoir un ou plusieurs Noms qui ne sont pas nécessairement uniques dans le contexte. Les Noms des classes sont des chaînes de caractères commençant par des lettres majuscules. L'approche utilise la convention de nommage habituelle dans la définition des ontologies.

Les classes sont considérées comme des Ressources dans le monde des TIC.

Remarque: le niveau de détail de l'ontologie: illustration de la classe «PhysicalPerson»

Une des classes de CSDM appelée «PhysicalPerson» désigne collectivement tous les humains. En général, nous comprenons ce qui est désigné par humain et nous avons trouvé pratique de nommer la classe «PhysicalPerson».

Cependant, pour certaines «communautés désignées», il est nécessaire de modéliser le tout avec des détails et des améliorations, par exemple pour distinguer les humains capables d'agir de ceux qui sont handicapés et qualifier les maladies par propriétés. Chaque instance de PhysicalPerson est soumise dans le contexte des «Droits de l'homme», c'est-à-dire des sujets de devoirs et de droits dépendant de leurs capacités d'agir et en tant que tels, responsables de leurs actes dans chacun des contextes spécifiques où ils sont exercés.

Si l'on a l'intention de représenter les cycles de vie de l'être humain, il faudra envisager de créer des Classes telles que les **Embryons** et les **Cadavres** en tant que spécialisation de la Classe **HomoSapiens**, une spécialisation de la Classe « **Nature** ».

Le cas de AnonymousPhysicalPerson sera traité plus loin dans le texte.

Dans certaines illustrations et exemples, la généricité est requise. Dans ce cas, le nom de la classe n'est pas mentionné, mais simplement le fait que la construction de classe est utilisée. Dans ce cas, le symbolisme utilisé est le suivant :

Figure 10 : Symbolisme générique de « Class »

Cela signifie que pour l'utiliser, il est nécessaire de substituer la chaîne <Class> par le nom réel de la classe, dans l'exemple par PhysicalPerson.

Les Super CLASSES dans CSDM : Entity et Configuration :

Dans CSDM (Conceptual Semantic Data Model) d'AXIS-CSRM, la classe **Thing** est d'abord spécialisée par deux superclasses : les **Entités** et les **Configurations**:

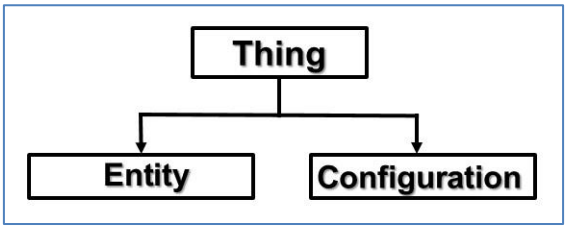

Figure 11 : AXIS-CSRM : Spécialisation de 'Thing' en 'Entity' et 'Configuration'

La Classe « Configuration »

La Classe « Configuration » qui couvre les constructions du modèle conceptuel des données d'AXIS-CSRM, exprime les Classes de gestion utilisées pour exprimer la structure interne et entre les Entités; pour gérer leurs cycles de vie; pour regrouper Entités et Configuration; pour la construction des Droits, de la vie privée, les Présentations et d'autres outils à des fins de gestion similaires; pour contrôler les importations et les exportations;

pour conserver la définition des Ontologies, les formats, le thésaurus, les taxonomies et autres constructions de profils.

Le diagramme de niveau supérieur de Configuration est le suivant:

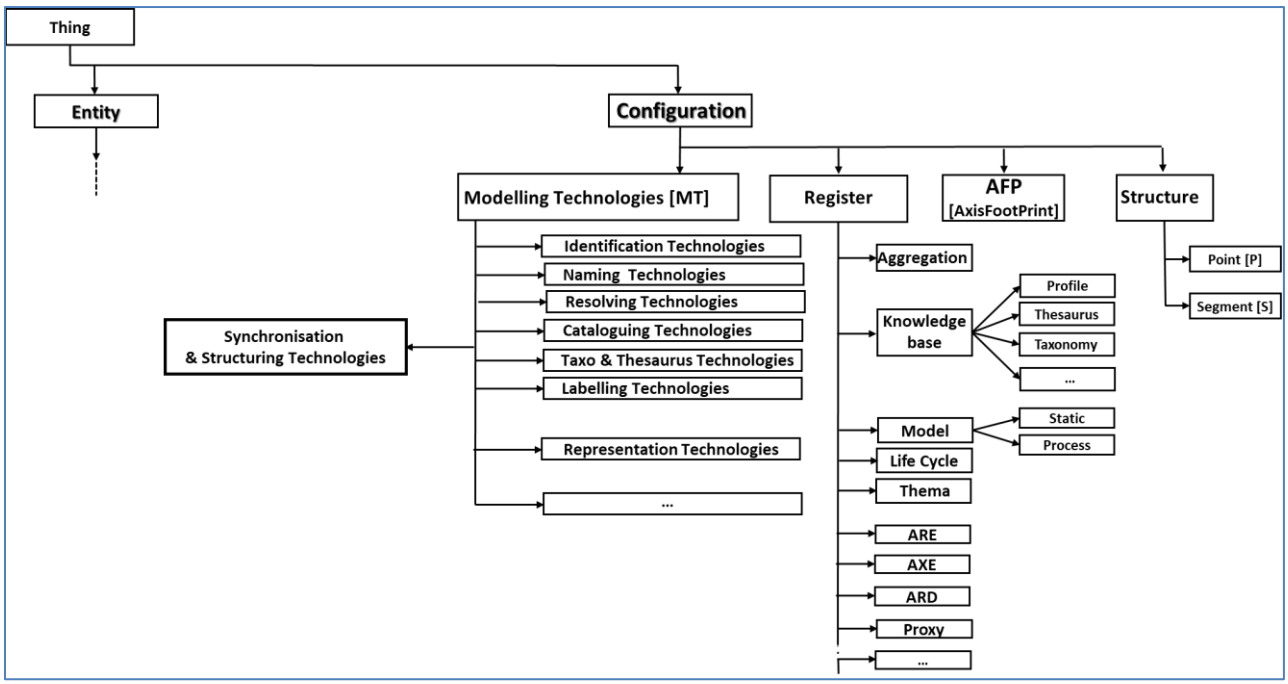

Figure 12: Schéma générique des 'Configurations'

Ces classes sont des mappings directs des concepts sous-jacents.

Chacune des Classes indiquées sur la figure sera définie plus en détail et illustrée spécifiquement.

La Classe « Entity »

La Classe « **Entity** » couvre toutes les choses qui sont des sujets potentiels de déclaration d'existence; de représentations et d'expression des relations sémantiques entre Entités. Pour le modèle de données d'AXIS-CSRM, c'est aux utilisateurs de structurer la Classe « Entity » en fonction de leurs besoins spécifiques. Cependant, le modèle de données d'AXIS-CSRM détaille la Classe « Configuration » de telle sorte que les Classes qui y sont définies puissent gérer les instances des constructs disponibles dans « Entity ». Néanmoins, le modèle de données d'AXIS-CSRM présente une structuration illustrative de « Entity » afin de pouvoir présenter des exemples de combinaison de constructs d'« Entity » avec des constructs de « Configuration ». Cette structuration illustrative peut évidemment être exploitée telle quelle lorsque l'on préfère éviter de définir une structuration sémantique complètement personnalisée, pour se borner à ne définir que ce qui est vraiment spécifique et à choisir ce que l'on importe comme ontologies.

Voici le diagramme de niveau supérieur des Entités qui sera utilisé dans les illustrations :

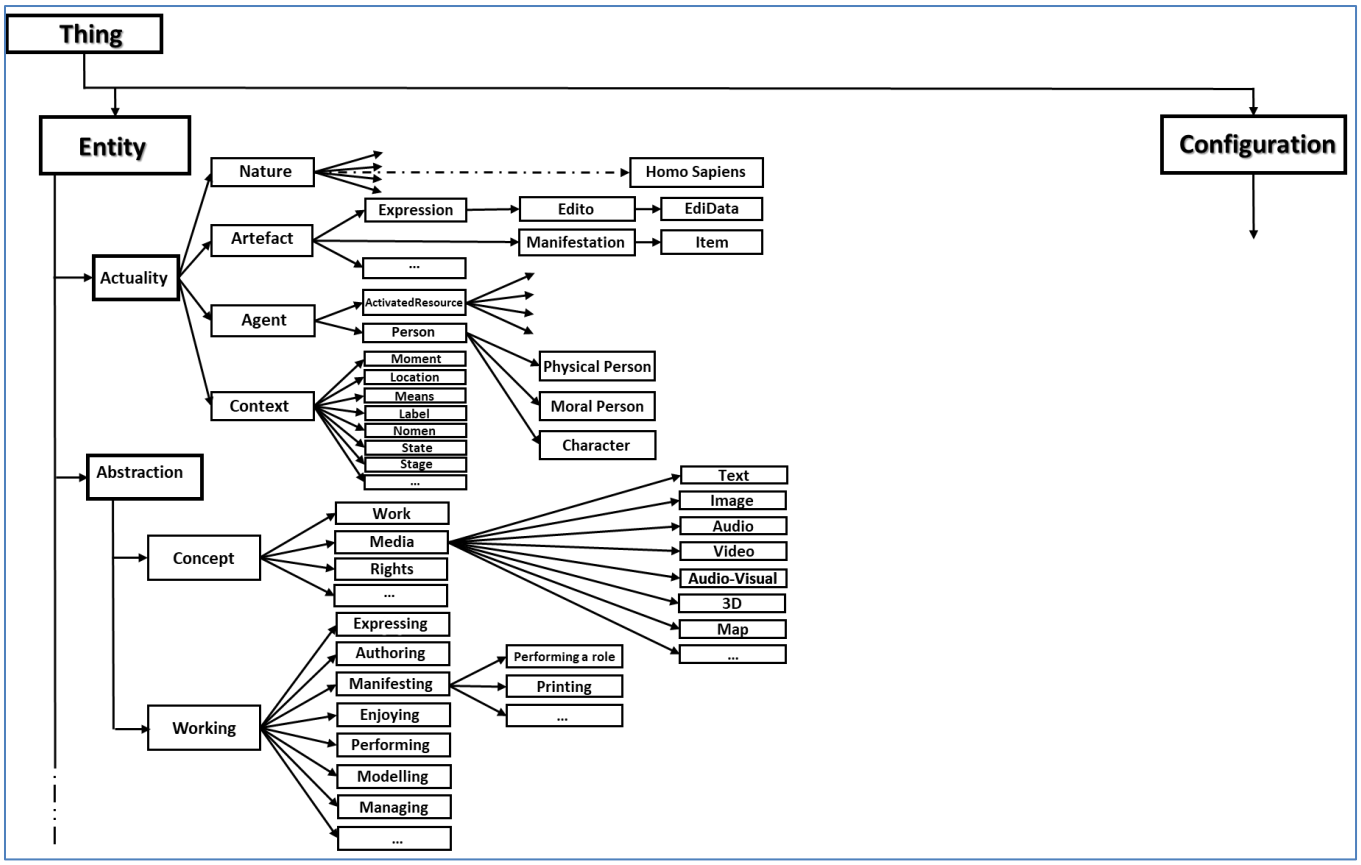

Figure 13: Schéma générique des 'Entity'

Chacune des classes indiquées sur la figure illustre les concepts sous-jacents. Elles seront définies plus en détail plus tard

Cependant, la compréhension des sections suivantes implique une introduction plus poussée du concept de « **Technologie de Modélisation** » et de chacune des classes de « **Configuration** ».

Les Technologies de Modélisation (à valider) :

Le concept de «Technologie de Modélisation» est très général dans les modèles conceptuels. Lorsqu'il sera mis en œuvre, pour des raisons pragmatiques, la Classe «Technologie de Modélisation» sera exprimée par des spécialisations, comme cela est illustré ci-dessous :

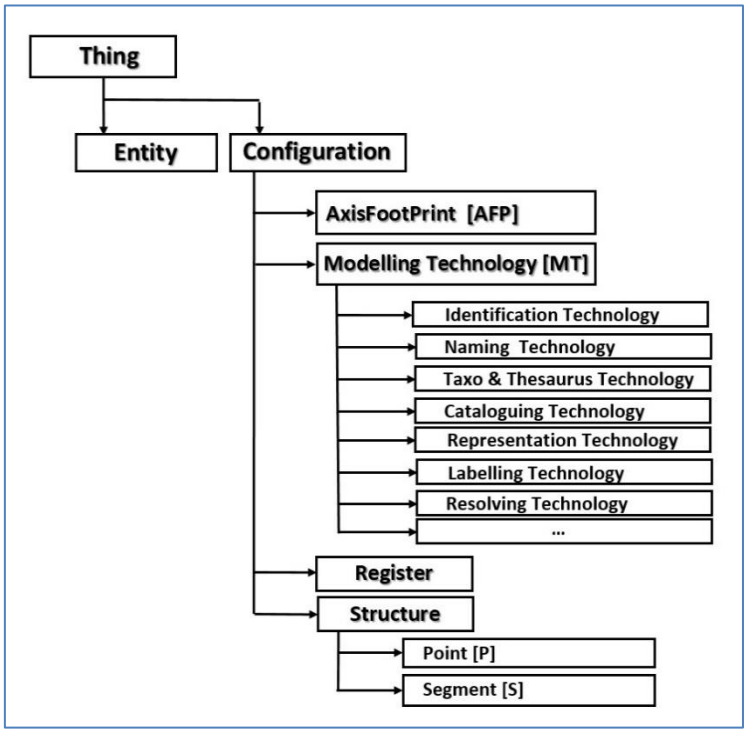

Figure 14: Les principales classes de «Configuration» et les Technologies de Modélisation

Ces spécialisations de la Classe «technologie de modélisation» seront utilisées lors de l'introduction de la Classe «Axis_Foot_Print».

Mais, pour simplifier, dans la plupart des cas, la référence générique à la Technologie de Modélisation sera utilisée, au lieu du raffinement contextuel applicable.

Instances de classes et utilisation des Technologies de Modélisation

À l'usage, les Classes sont instanciées: cela signifie qu'une instance d'une Classe reçoit un identifiant tel que les instances de classes deviennent des ressources.

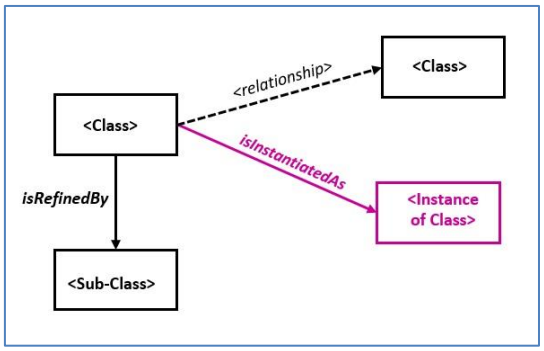

Figure 15 : Expression des Relations et Instanciation des Classes

L'exemple ci-dessous représente graphiquement un exemple d'une instance de Classe PhysicalPerson:

Figure 16 : Instance de la Classe 'PhysicalPerson'

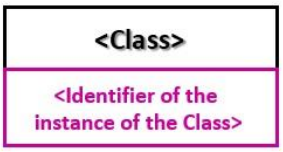

Figure 17 : Représentation générique d'une instance de Classe

Il existe plusieurs façons d'exprimer les identifiants des instances des Classes. La contrainte est que chaque identificateur doit être unique dans le contexte local. Chacune des façons d'exprimer l'identificateur doit avoir un moyen managérial de le générer (protocole) et une façon de représenter l'identificateur comme une chaîne de caractères (syntaxe de codage). Chacune de ces façons (protocole / outils / syntaxe / ...) est supposée s'exprimer formellement en utilisant une modélisation sémantique jusqu'à un niveau de détails ultimement ou utilement possibles. Cela sera détaillé plus tard.

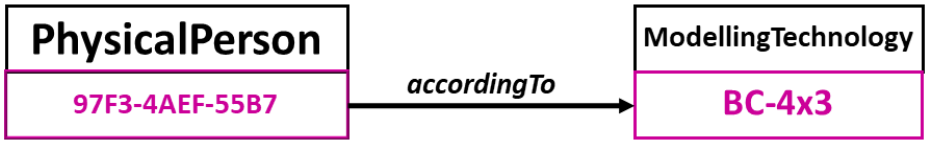

Figure 18: Codage suivant une Technologie de modélisation

La chaîne de caractères **97F3-4AEF-55B7** exprime la valeur de l'URI. La couleur violette exprime que la chaîne représente une instance de la classe 'PhysicalPerson'. La flèche noire intitulée « ***accordingTo*** » représente la relation avec la classe ***BC 4x3***, ce qui signifie que le codage de l'URI a été exprimé selon les règles et le protocole définis par BC 4x3, une spécialisation de la «Technologie de Modélisation» de la Classe. L'unicité des noms des Classes et de l'URI des instances doit être assurée dans le contexte local (l'**ARE**).

Propriétés des classes et utilisation des Technologies de Modélisation

Dans le CSDM, les propriétés sont des constructions conceptuelles dédiées à la définition des caractéristiques intrinsèques des instances de Classes. Les propriétés sont Typées (par exemple, "Moment" ou "Lieu" ... "Date de naissance" ...). L'instance d'une Propriété est représentée selon une Technologie de Modélisation (par exemple «bitrate» serait exprimée comme un entier (integer) ou «CityName» exprimée en tant que chaîne de caractères ; ou une «date» exprimée en tant que concaténation de trois propriétés ['Année' - ' Mois '-' Jour']). Certaines propriétés peuvent être exprimées dans plus qu'un même langage humain. Dans CSDM, les Technologies de Modélisation sont supposées exister dans un contexte spécifique: un Modèle Conceptuel doit simplement supposer l'existence des Technologies de Modélisation capables de définir les formats, le codage et les protocoles.

Dans l'exemple précédent, une instance de la classe 'PhysicalPerson' a reçu l'ID: 97F3-4AEF-55B7: c'est une manière persistante d'identifier quelqu'un.

La première intention évidente est de nommer les instances de [Things]:

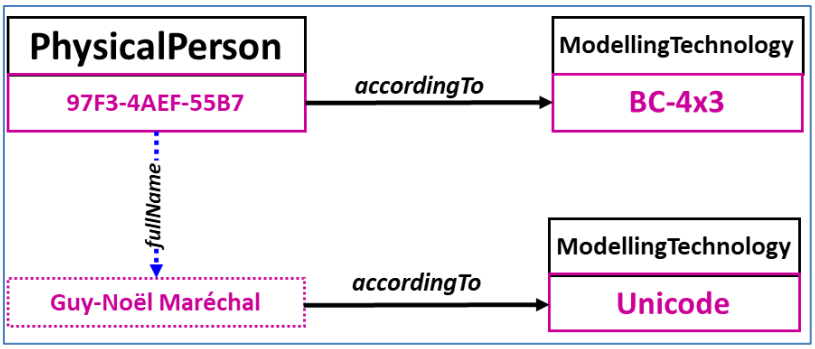

Figure 19 : La vision humaine vs la vision Machine

La propriété '**Name**' associera un nom à l'Identificateur de l'instance de la classe '**PhysicalPerson**'. Pour la compréhension humaine, le nom «Guy Noël Maréchal» est sémantiquement parlant tandis que, pour le système informatique, 97F3-4AEF-55B7 possède les caractéristiques appropriées pour un traitement correct.

La Technologie de Modélisation '**Unicode**' est appropriée pour illustrer le concept de la technologie de modélisation dans les modèles conceptuels : elle possède toutes les caractéristiques requises pour coder des chaînes de caractères et relier les ***règles et les protocoles sémantiques.*** Pour plus de détails, voir l'Annexe 4.

Sur les figures, le lien vers les Propriétés est représenté par de fines lignes pointillées et le 'Nom de Propriété' est présenté en rectangle avec des bordures en pointillés; et la **Valeur** de la Propriété est représentée en violet.

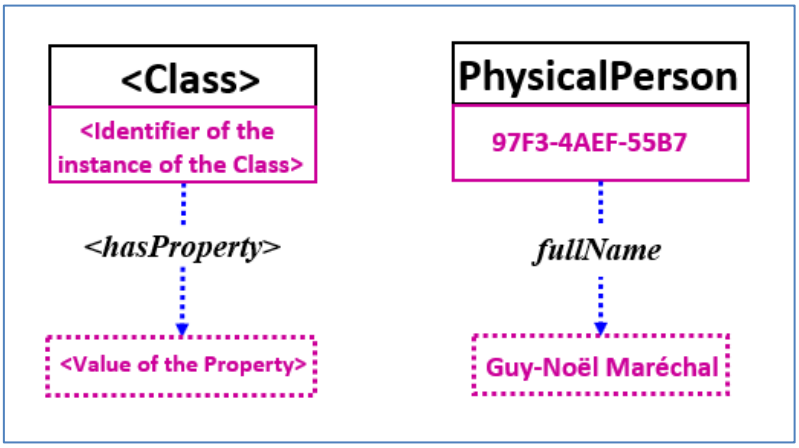

Figure 20 : Valeurs des propriétés des Classes

Cependant, l'allocation de valeur à une propriété doit conceptuellement couvrir la possibilité d'exprimer la même réalité ou la même cible en fonction des différentes Technologies de Modélisation (ou expression paramétrique de la Technique de Modélisation encadrée) ou de différents contextes (langages naturels, moments, usages ...).

L'expression d'un moment (**Moment**) et de lieux (**PLACES**) illustre bien les différentes Technologies de Modélisation :

la date de naissance de Guy-Noël Maréchal peut être exprimée conformément à la norme ISO 8601 (en sélectionnant la variante AAAA-MM-JJ, où AAAA représente l'année de naissance; MM le mois de naissance; et où DD le jour de naissance): exemple **1937-12-25**. Mais cette date ne fait référence à un moment unique que si le «lieu» de naissance est connu; dans ce cas, en Belgique. En outre, elle fait référence au «Calendrier Grégorien».

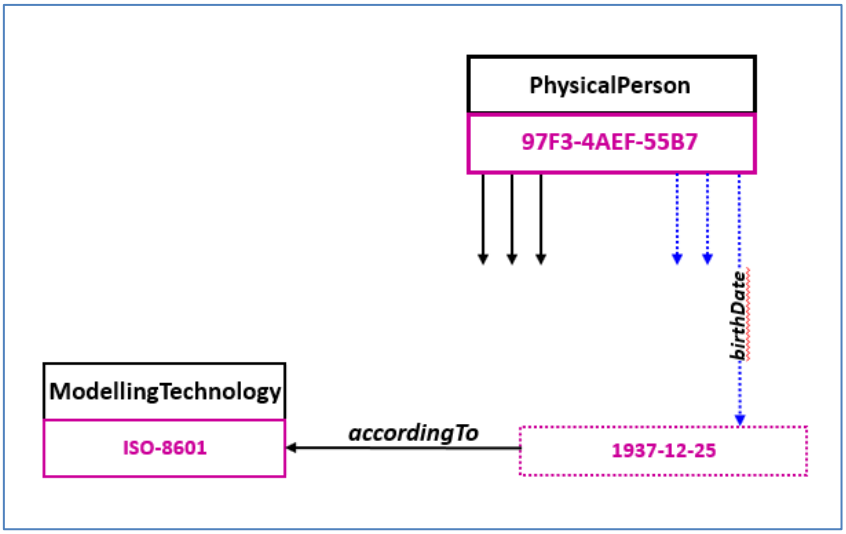

Figure 21 : Codage des valeurs des propriétés de Classes

Figure 22: Représentations compactes

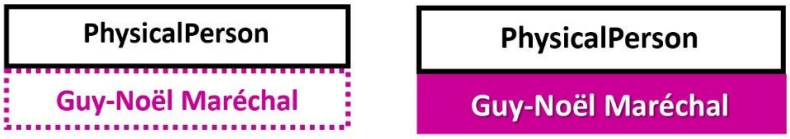

Figure 23 : deux représentations ultra compactes équivalentes

En Belgique, chaque citoyen a un identifiant légal unique: il est formaté selon la règle suivante: une séquence de cinq champs, avec les séparateurs syntaxiques en fonction du contexte:

1. L'année de naissance codée selon YYYY
2. Le mois de naissance codé selon MM
3. Le jour de naissance codé selon DD
4. L'identifiant du Livre d'Enregistrement de l'Autorité Publique: dans le cas 317
5. Le numéro de séquence des enregistrements de l'année de naissance sur le registre: dans le cas 85

Sur les cartes d'identité, les séparateurs syntaxiques sont comme illustré: **1937.12.25-317.85**!

Sur les passeports, la représentation est **1937-12-25 & 317,85**

Réification de Properties

La **réification** de Properties consiste à créer un porteur des différentes représentations de Properties visant à exprimer le même signifié. Le porteur sera une instance de Classe. Dans le cas des Properties, la réification est purement liée à l'expression cohérente des properties ; l'on verra plus loin que la réification des Relations / Relationships est conduite en vue d'étendre les signifiés de manière radicalement nouvelle tout en restant compatible et évolutive.

Le mot 'réification' est dérivé du latin où « res » désigne « chose » : dans ces cas, on regarde donc une Property comme une « chose » et donc l'élever au rang de Classe.

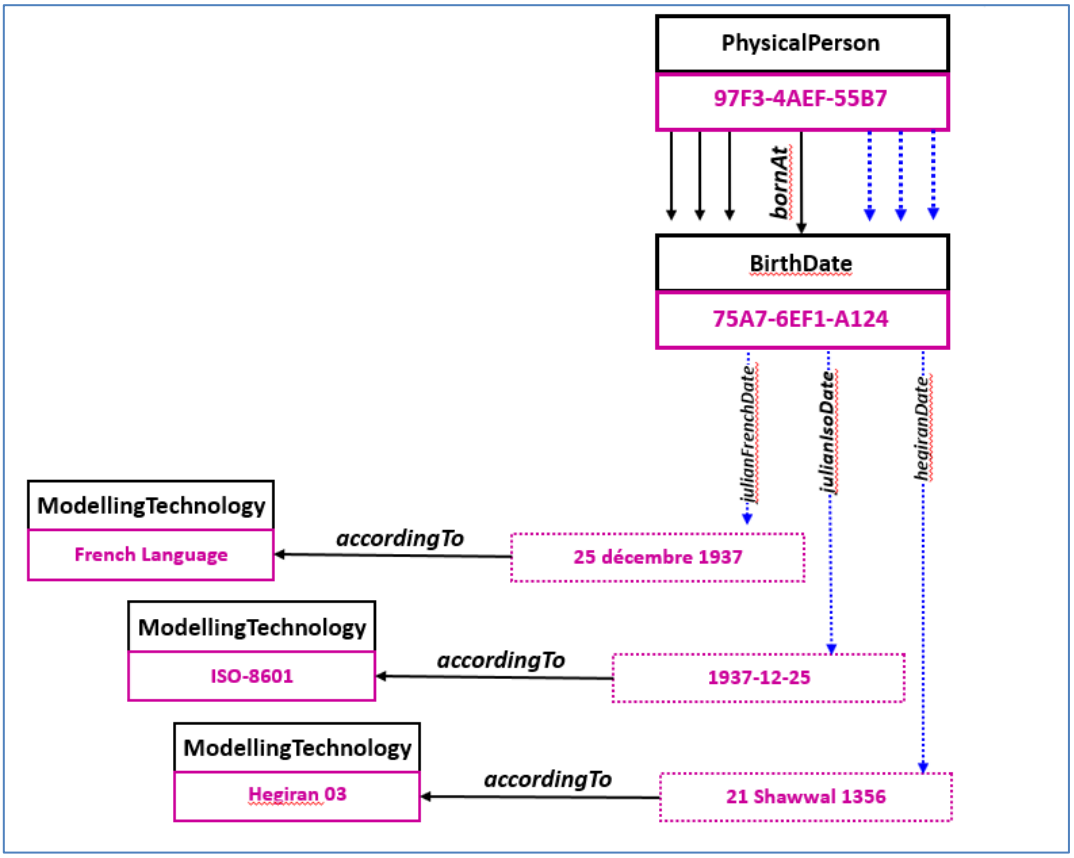

Figure 24: Réification de «*datedenaissance*» selon ISO 8601 et d'autres technologies de modélisation

L'expression de la date de naissance pourrait être faite suivant différents calendriers : Par exemple, Si la référence doit être donnée par le calendrier de l'Hégire (version actuelle 03), à Bruxelles, elle deviendra **'21 10 1356** 'ou '**21 Shawwal 1356'**.

Le calcul de l'expression de la date suivant le calendrier de l'Hégire peut être fait au départ de la valeur de la date en calendrier Grégorien. Dans ce cas, l'on pourrait se borner à ce que les AIP ne représentent les dates qu'en mode Grégorien et que la valeur de la date serait présentée suivant d'autres calendriers choisis par la 'Designated Target Community'. Cependant, dans de nombreux cas, l'expression de la valeur d'une propriété ne peut être déduite par calcul. L'identifieur légal d'une personne est défini de façon différente suivant les pays.

La figure qui suit illustre l'usage systématique de la réification :

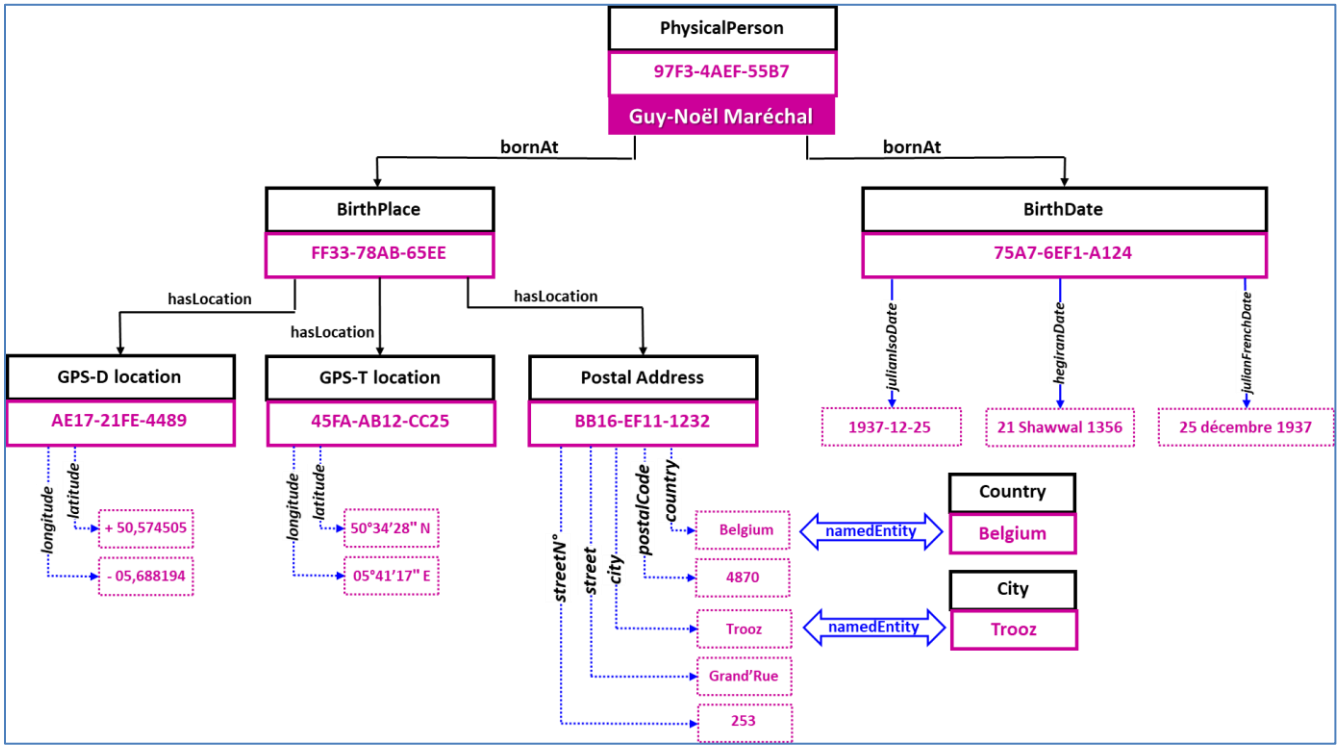

Figure 25 : Illustration de la réification de Propriétés

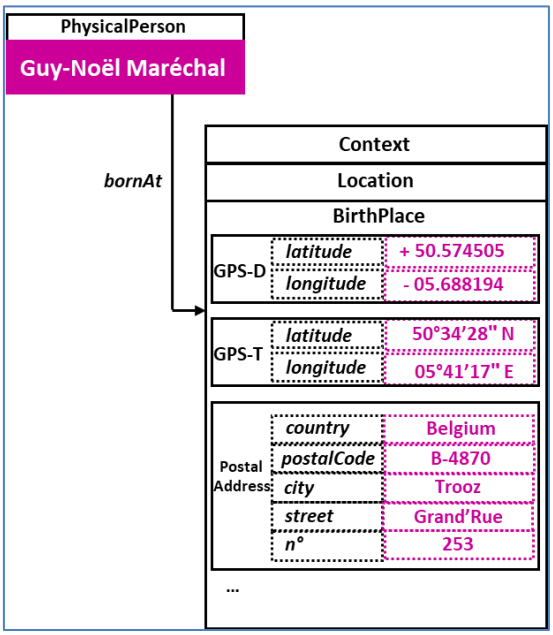

Figure 26 : Représentation compacte d'une Propriété réifiée

L'expression des **Noms** est une bonne illustration de l'influence du Contexte. Le nom de la célèbre peinture de "**Mona Lisa**" est généralement connu sous le nom de "**La Joconde**" dans les textes en français.

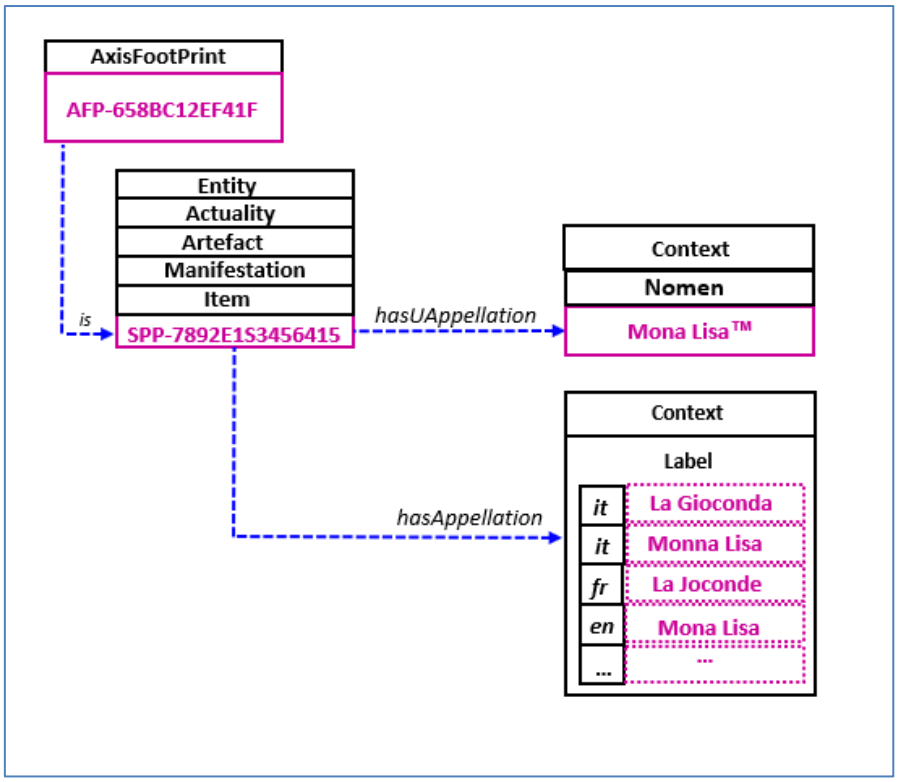

Figure 27 : Nom de la personne physique : plusieurs syntaxes et langages

Parfois, les autorités publiques changent les noms de rue pour des raisons pragmatiques (en évitant le même nom pour différentes rues après la fusion des municipalités), mais aussi pour des raisons politiques : le "**Rond-Point des Champs Élysées**" de Paris a été rebaptisé "**Place Charles de Gaulle**". Les Constructs sont capables de faire face à ces exigences (si nécessaire), elles seront introduites plus tard.

La propriété Construct peut être mappée à la structure dataProperty utilisée dans la modélisation du web sémantique (voir RDF et OWL Primers). En particulier, CSDM cache les constructions de OWL pour définir les 'formats de données' par une construction générale abstraite appelée "Modelling Technology". Dans la représentation graphique, CSDM représente le Construct "Object Property" OWL par un rectangle noir et son lien vers l'instance de la classe par une flèche noire pointillée (signifiant "*has*"). Cette approche conceptuelle permet d'utiliser le même symbolisme pour exprimer les Classes et leurs Instances que pour exprimer les Propriétés et leurs Instances.

L'expression graphique des structures de données pourrait être simplifiée ou présentée de manière plus compacte quand il n'est pas nécessaire de disposer de la représentation complète pour le message prévu. Un exemple :

Full **Compact 1** **Compact 2**

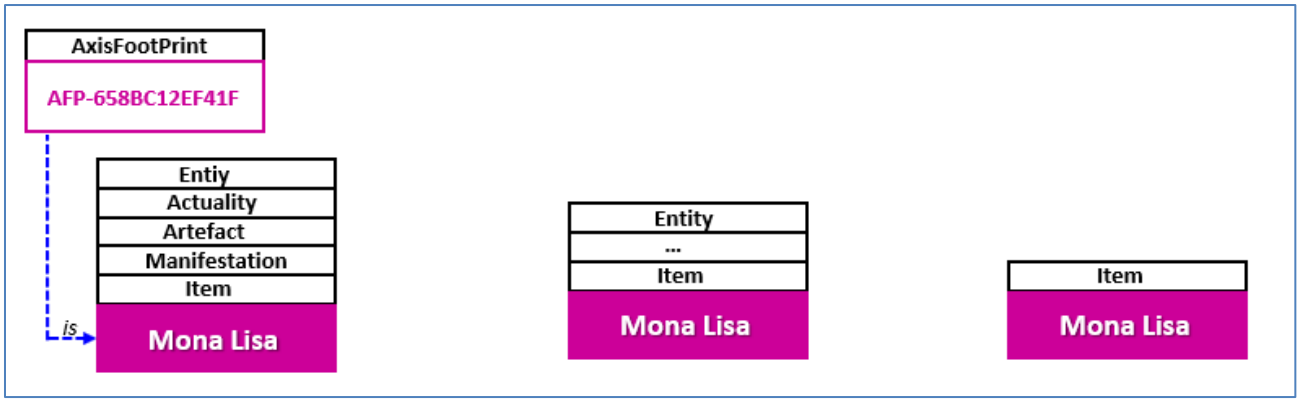

Figure 28 : Simplification des représentations

Le "Compact 1" doit être utilisé lorsque la référence à la chaine de Classes a de l'importance pour l'illustration; tandis que le 'Compact 2' devrait être utilisé lorsque l'instance de la propriété Name est suffisante.

Les relations

La Relation est l'un des trois types de Constructs. Elle exprime une manière typée de lier des instances de Classes :

Exemple de l'expression qu'une personne physique spécifique est le père d'une autre personne physique sera représenté par la relation ***isFatherOf.***

Dans le CSDM, les relations expriment les types de liens entre les classes et/ou entre les instances des classes.

Les noms des Relations sont des chaînes de caractères présentées en lettres italiques commençant par une lettre minuscule et permettant des lettres majuscules dans la chaîne de caractères pour faciliter la lecture et la compréhension du sens exprimé par le type de relation. Le nom de la relation est mentionné en haut d'une flèche pointillée noire. Si les liens sont affichés dans les deux sens, avec la même signification, alors une seule ligne pointillée est dessinée et le nom est mentionné une seule fois.

L'approche est fort proche de la convention de nommage de la définition des Ontologies.

L'exemple suivant est auto-explicatif :

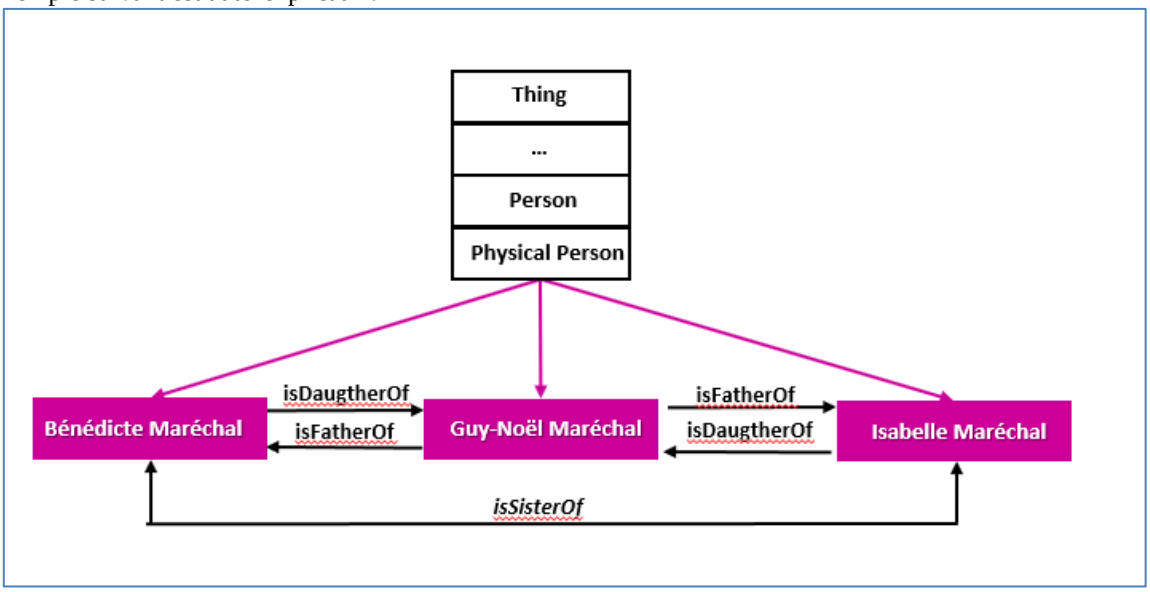

Figure 29: Représentations graphique des 'Relations'

Les Relations '*isSisterOf*' et '*isDaughterOf*' peuvent être déduites des relations '*isFatherOf*'. La relation '*isSisterOf*' est réflexive, ce qui signifie qu'une flèche représente les deux côtés.

Réification de Properties

La **réification** de Properties consiste à créer un porteur des différentes représentations de Properties visant à exprimer le même signifié. Le porteur sera une instance de Classe. Dans le cas des Properties, la réification est purement liée à l'expression cohérente des Properties ; l'on verra plus loin que la réification des Relations / Relationships est conduite en vue d'étendre les signifiés de manière radicalement nouvelle tout en restant compatible et évolutive.

Réification de Relationships

La **réification** de Relationships consiste à créer un porteur des modèles représentant les processus et moyens mis en œuvre pour réaliser ce qui est exprimé par la Relationship !

Le porteur sera une instance de Classe. Dans le cas des Relationships, la réification est conduite en vue d'étendre les signifiés de manière radicalement nouvelle tout en restant compatible et évolutive. Cela dépasse de loin les motivations et les capacités d'expression liées à la réification des Properties.

À titre d'exemple, l'on peut voir le modèle fonctionnel FRAR comme une réification du modèle FRBR :

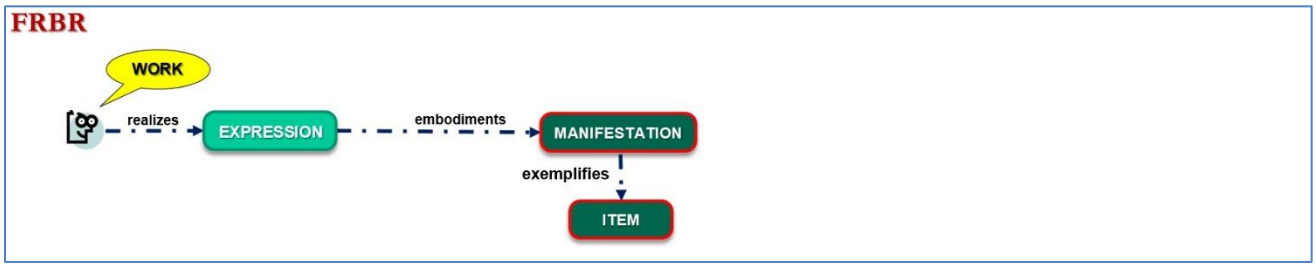

Figure 30 : FRBR (Entités du groupe 1)

Le modèle fonctionnel FRAR (Entités du groupe 1) réifie non seulement « realizes », « embodiments » et « exemplifies » ; mais encore, différencie les divers Agents et Contextes impliqués ; rend possible les cas où les processus ne sont pas complets, introduit les possibilités de boucles, de dérivations, de fusion au départ de plusieurs 'Works', de création de variantes …

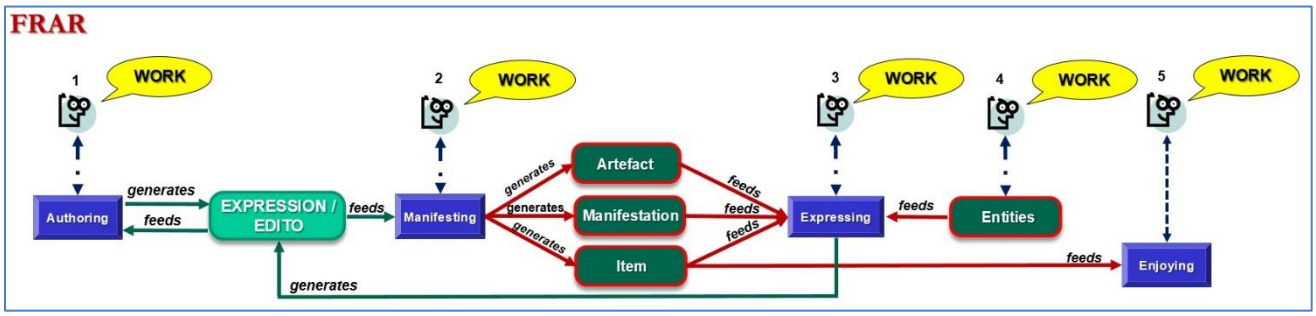

Figure 31 : FRAR (Entités du groupe 1)

Les processus par lesquels l'auteur du FRBR créée une 'Expression' du 'Work' (ce que FRBR exprime sous la forme de la Relationship « *realizes* »), devient un modèle porté par les constructs liés à une instance de la Classe « Authoring » ou de la Classe « Expressing » suivant les phases des processus de création d'existence, d'expression des auteurs et de modification de contenus ou de formats.

De même pour « *Embodiements* » qui est réifié via la Classe « Manifesting ».

La réification permet d'identifier les Agents impliqués à chaque étape, ce qui conduit à avoir des porteurs des droits associés aux agents et aux processus ancillaires associés.

De plus, si les archives disposaient des représentations de « Manifestations » dans la ligne de FRBR, la production de représentations migrées conformes à FRAR pourrait être réalisée de manière structurellement compatible et sémantiquement identique à la source. L'enrichissement y serait exprimé par ajout de structures et de l'expression explicite des processus.

Dans la foulée, l'enrichissement lui-même peut être documenté afin d'assurer la traçabilité ; et l'enrichissement peut concerner la réification des Properties et leur enrichissement.

Une réification plus complexe sera introduite plus loin concernant les Classes « Point » et « Segment : elle combine une réification de Properties avec une réification des Relationships.

Autres Constructions conceptuelles des Entités et de leurs représentations graphiques.

Modélisation générale des « Entités »

La section 6.5.3 détaille la modélisation des « Entités » et leur structuration via les Constructs de « Configuration ».

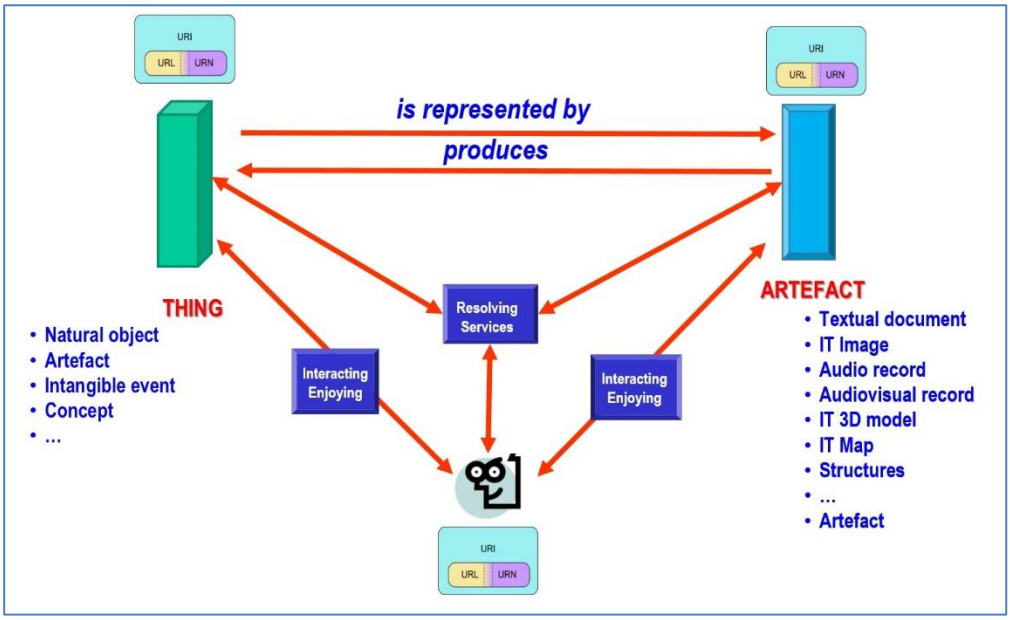

Figure 32 : La modélisation générique d'AXIS-CSRM

Comme cela a été introduit à la section précédente, le modèle de données est bâti sur deux types de **Constructs : « Entité » et « Configuration »**.

Le terme 'construct' désigne les moyens conceptuels utilisés pour construire une représentation de quelque chose.

Dans AXIS-CSRM, l'approche conceptuelle générale du modèle de données s'organise comme suit :

1. Déclaration d'existence[6] d'une instance d'une « Entité »
2. À cette déclaration d'existence, accrochage d'un (ou plusieurs) groupe (s) de « Métadonnées »
3. Déclaration d'existence d'une instance d'un « Model » d'une Entité
4. À cette déclaration d'existence, accrochage d'un (ou plusieurs) modèle(s) de cette Entité
5. Expression des liens entre ces déclarations d'existence

Cette approche est illustrée par l'exemple représenté ci-dessous. L'intention est de représenter un Objet physique réel dans le système informatique.

- L'Objet peut être un Artefact (dans ce cas la statue de Lincoln, ou un livre par exemple), un Objet naturel (un parc naturel protégé ...) ou une Personne physique.
- à cet Objet est associé un I&T Token [Jeton] d'Identification et/ou de Traçabilité. Ce Token serait, par exemple, un code-barres ou un QR-code et/ou un badge RF-Id (voir figure suivante)
- La flèche verticale représente le lien physique qui est établi entre l'Objet et le I&T Token : cela peut prendre la forme d'un Token collé sur l'Objet ; variante, devant un paysage naturel, sur un panneau ; ... Pour une Personne physique, c'est le Passeport ou la Carte d'Identité.

La figure montre la déclaration d'existence de cet Objet et sa modélisation dans le système informatique: en premier lieu la déclaration de la Base de Connaissance [KB] associée à l'objet, au centre la déclaration d'existence de la représentation de l'objet et de même pour le Token (Note: la Base de Connaissance associée au Token n'est pas montrée).

[6] Le symbole mathématique ∃ signifie 'déclaration d'existence' !

Chaque Entité (Knowledge base – Object – Token) est décrite par des Métadonnées spécifiques et dispose d'une déclaration d'Existence des modèles de ces Entités. Ces nouvelles entités fédèrent les différents modèles.

La figure montre enfin les liens entre les Entités (Base de Connaissance, la représentation de l'objet et l'I&T Token) en Outer Configuration et les liens entre les divers modèles représentant les Entités (Inner Configuration)

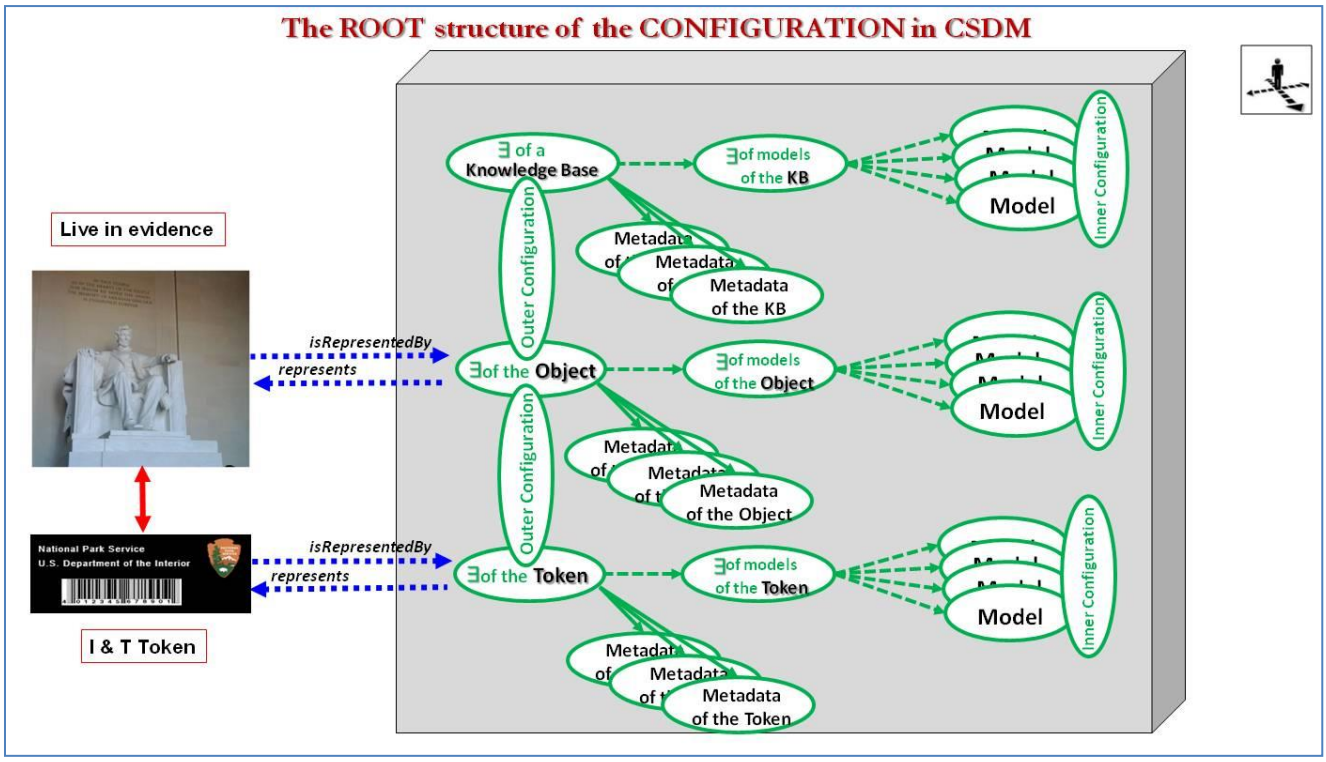

Figure 33 : Application exemplative de l'approche du modèle de données[7]

La figure suivante montre l'interaction entre les Modèles, les Humains et le monde réel : L'humain voit en évidence

- Perception [A] : l'Objet;
- Perception [B] : l'I&T Token 'Jeton d'Identification & Traçabilité';
- [C] manipule l'I&T Sensor (agent du protocole [C]) afin de permettre au système informatique (IT System) d'interagir avec l'I&T Token [4] ;
- [D] et interagit enfin avec le 'Graphical & Audio & Vidéo Human Interface' pour chercher, trouver, activer … cet IT System. Ces interactions concernent non seulement les Représentations des Objets, mais aussi l'accès aux Objets eux-mêmes (localisation et droits).

[7] <Cette illustration devra être expliquée plus en détail> !!!!

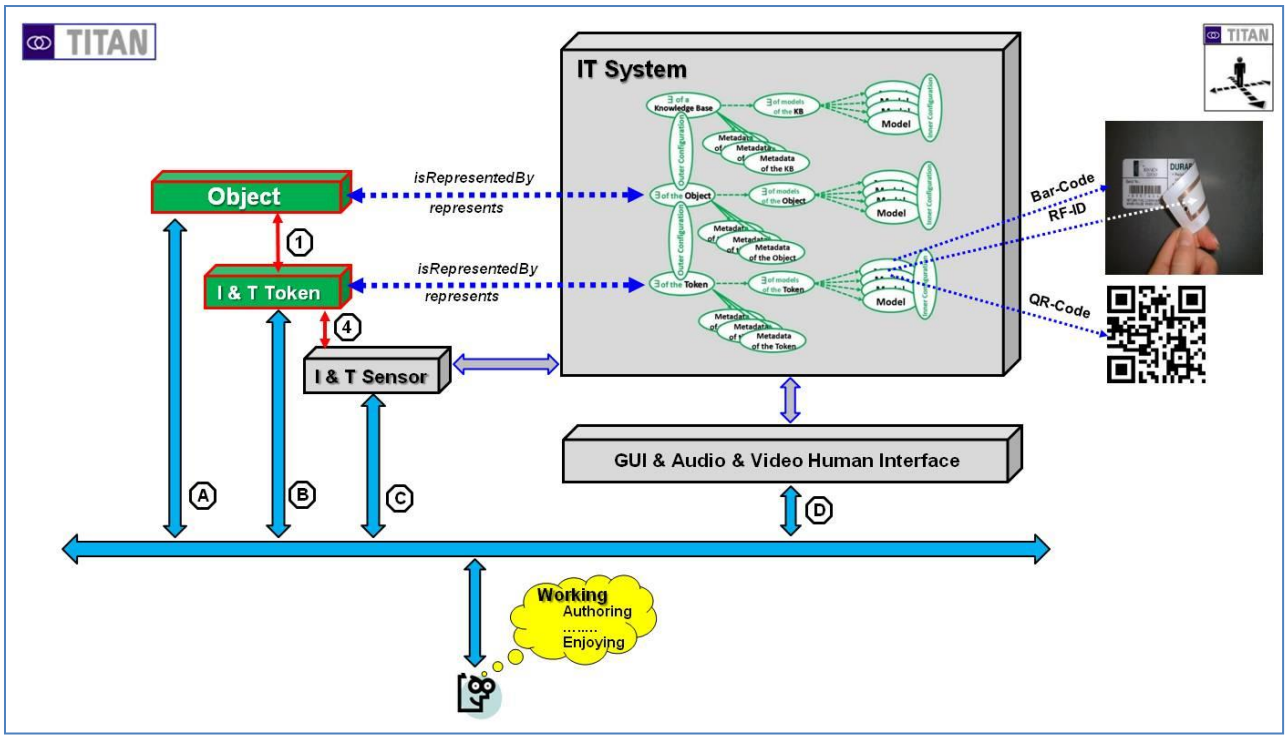

Figure 34 : Application exemplative de l'approche en contexte IT

L'approche conceptuelle de la modélisation avait été représentée par la figure qui suivante :

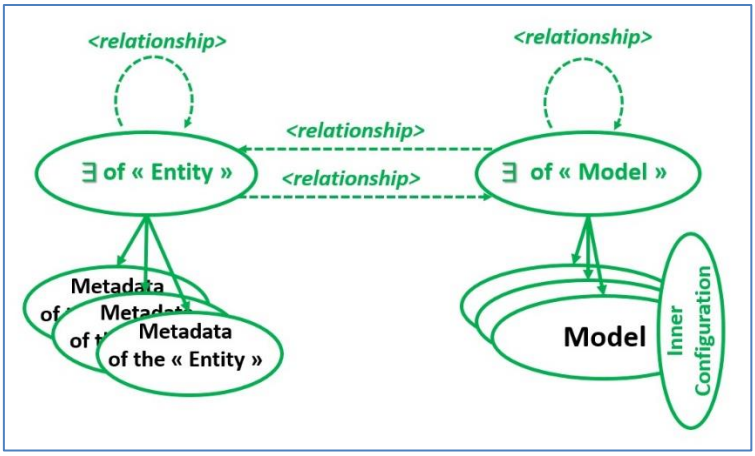

Figure 35 : Approche conceptuelle générique du modèle de données

Cette figure peut être synthétisée suivant la figure qui suit où <**ID**> désigne le cas général et **A** l'occurrence particulière !

Figure 36: Représentation compacte de la figure précédente : les « Entités » dans CSDM

Ainsi que cela a été introduit plus tôt, cette approche permet de représenter les **Traitements effectués par des processus lors de leur exécution entre deux États.** Il s'agit de **représentation d'étapes particulières** de classes qui sont des spécialisations de la classe « **Working** », laquelle est une spécialisation de la classe « **Abstraction** », laquelle est une spécialisation de la classe « **Entité** ». « **Working »** est bien une Abstraction, mais les Modèles représentent les États et les Traitements effectués par le processus entre ces États.

La figure qui suit illustre l'interaction de « A » avec « B » via l'exécution du processus « P » entre deux de ces états :

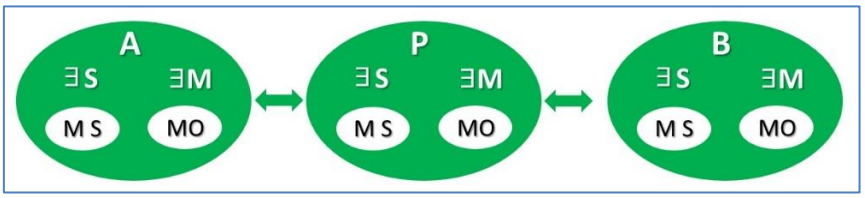

Figure 37 : l'interaction de "A" avec "B" à travers l'exécution du processus "P"

Ce schéma représente, par exemple, que Madame « B » écoute le concerto pour piano représenté sur le CD-A « A », grâce au Processus « P ». Ce Processus « P » d'écoute, dans l'exemple, impliquerait un lecteur de CD, des haut-parleurs … et d'avoir activé la lecture à 10h22 qui se terminera à 11h17.

Le symbole ⟷ représente l'interaction.

L'approche couvre aussi évidemment les évolutions d'une « Entité » particulière par un chainage d'états. Ce chainage peut être **temporel** (de la naissance à la mort, par exemple) ou **cyclique** (production en série d'objets, par exemple) ou tout autre chainage désigné habituellement comme « Machines à États Finis ». Il y a de nombreuses façons de représenter ces machines ! La plus connue est celle des « Réseaux de Petri ». AXIS-CSRM n'est pas impliqué dans ces façons et se borne à s'y référer via le lien vers une « Modelling Technology ».
La figure qui suit représente le passage de « A » de son état « A (n) » à son état « A (n+1) » via le processus « P ».

Le symbole ➡ représente l'enchainement.

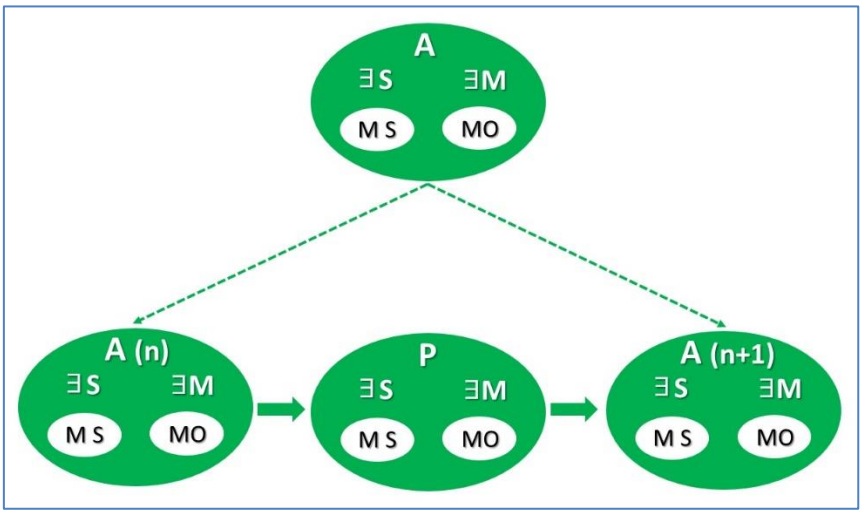

Figure 38 : le passage de « A » de son état « A (n) » à son état « A (n+1) » via le processus « P »

Représentations graphiques plus détaillées :

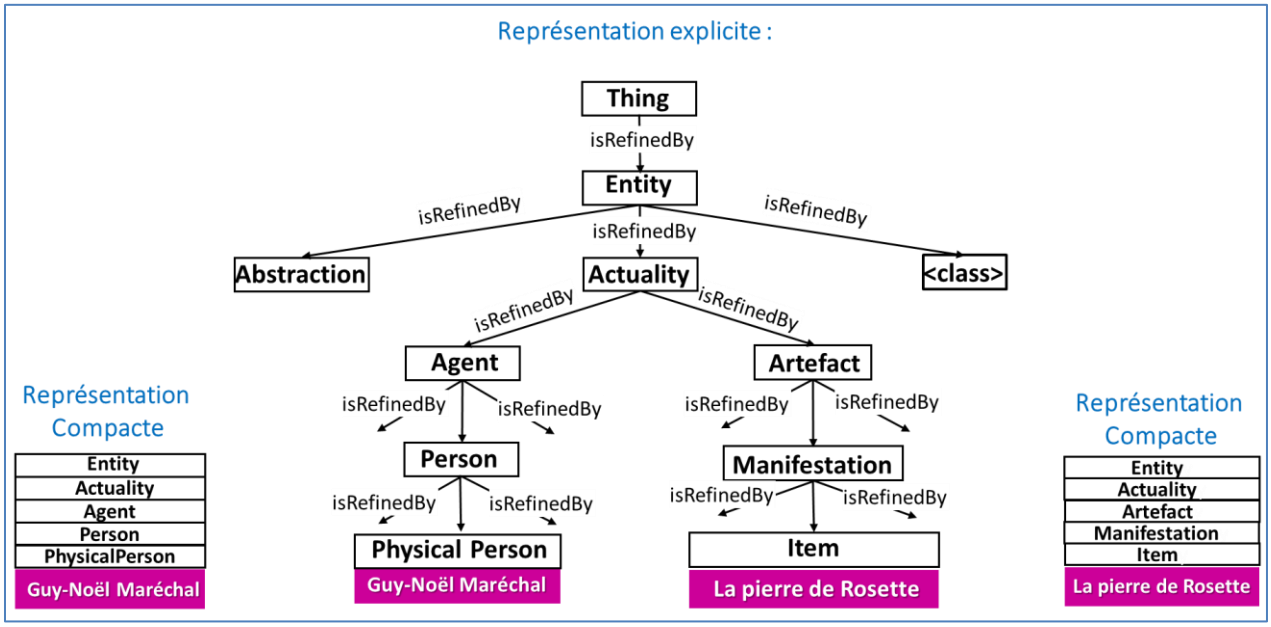

Figure 39: Représentations compactes de « raffinement » de Classes:

L'expression graphique des structures de données pourrait être encore simplifiée ou présentée de manière plus compacte quand la représentation complète pour le message prévu n'est pas indispensable.

Une représentation compacte des Classes et des Propriétés peut être organisée à travers quelques exemples auto-explicatifs:

Le symbole du **Ou-Exclusif** est représenté par le signe mathématique : ⊕

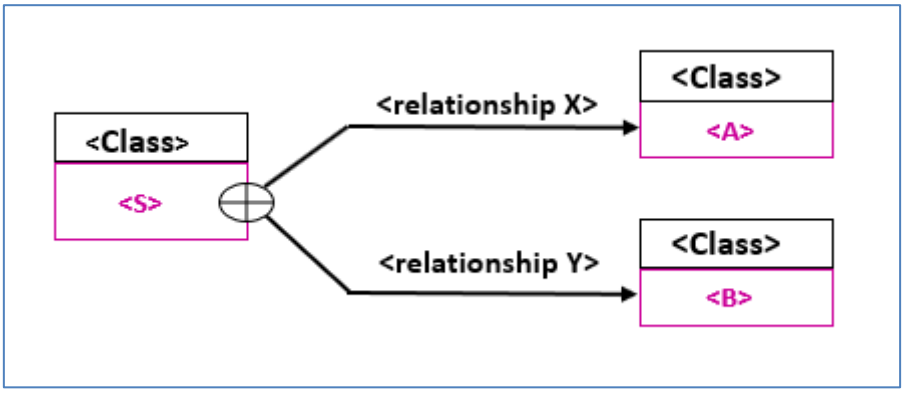

Figure Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.**40 : La relation sera soit entre <S> et <A> ; soit entre <S> et <B>**

Le symbole du Ou-Exclusif indique que la fin de la ligne fléchée représente une situation où toute instance de l'entité située au début de la ligne peut être associée, par l'intermédiaire de l'une des relations, à une instance de l'Entité et à une seule, lié par la ligne fléchée. Lorsque l'association relie deux entités à travers une liste de relations, une, et une seule, doit être sélectionnée.

L'illustration exprime que l'instance <S> peut être associée soit à l'instance <A> par la Relation <X>; soit avec l'instance <B> via la Relation <Y>.

Un cas similaire se réfère à la sélection exclusive des Classes :

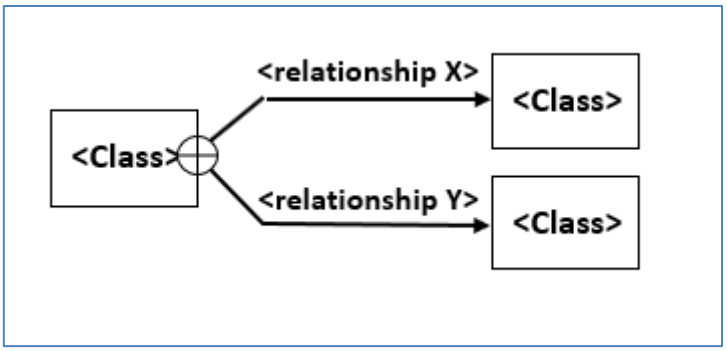

Figure 41 : sélection exclusive d'une Classe

Un cas similaire se réfère à la sélection exclusive de Relation. La figure ci-dessous exprime que les instances de l'association impliquent de sélectionner soit la relation "X" soit la relation "Y". Un cas typique sera introduit plus tard dans le contexte du registre de la classe où il est nécessaire de sélectionner la relation «possède» ou la relation «utilise».

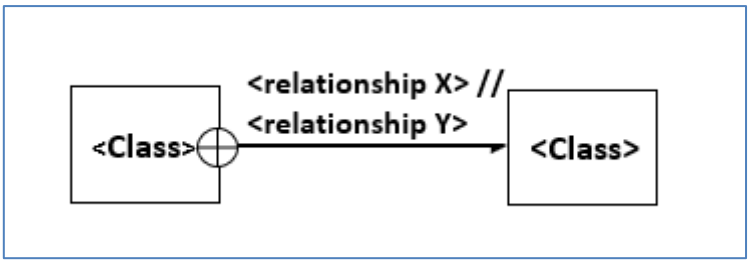

Figure 42: sélection exclusive d'une 'Relationship'

Une ligne pointillée avec une double flèche à la fin représente une relation où toute instance de l'entité située au début de la ligne peut être associée à une ou plusieurs instances de l'entité liées par la double flèche :

Le **lien vers l'emplacement d'une ressource** sera représenté par le symbole suivant :

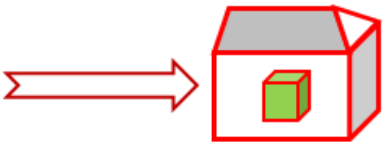

Figure 43 : représentation de la localisation d'une Ressource

Le **lien vers l'emplacement d'un fichier sur une ressource** sera représenté par le symbole suivant:

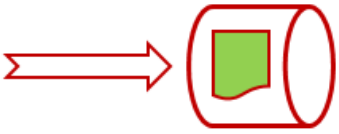

Figure 44 : Emplacement d'un fichier sur une ressource

Le symbole suivant représente un '**Agent**'. La flèche pointillée illustre l'interaction (par exemple, le déclenchement d'un processus).

Figure 45 : le symbole d'un Agent

Le symbole suivant représente un '**Processus**':

Figure 46 : le symbole d'un "Processus"

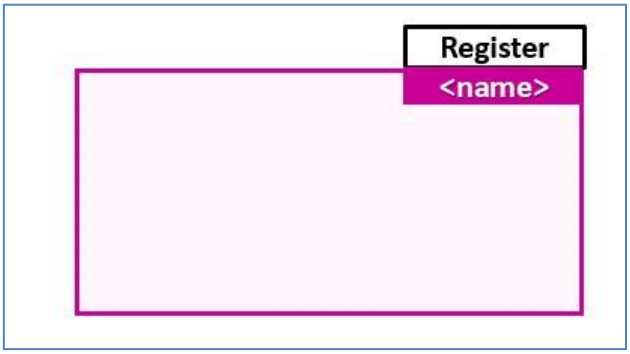

Figure 47: le symbole compact d'un 'Registre'

Déclaration d'Existence des « Entités » & de leurs « Modèles »

La déclaration de l'existence des « Entités » et de leurs « Modèles » s'exprime via la classe « **AxisFootPrint** » abréviée « **AFP** ». Cette déclaration va être le pivot, le support de leur 'régistration'/'catalogage' ; de leurs représentations ; de leur localisation et de la localisation de leurs représentations ; de la représentation de l'histoire, des étapes de la 'vie' des Entités ; …

Ce Construct porteur est très général. Il est le support de **ce qui, dans le contexte, est persistant par nature** des Entités et des liens vers les diverses représentations des Entités dans leur cycle de vie et des liens et évolution des liens entre les Entités. Comme introduit à la section précédente, les Entités dont on déclare l'existence sont considérées comme stables ou factuelles : l'on représente des États des Entités ou des Traitements effectués par des Entités entre deux États. Cela signifie que ce qui est considéré comme stable dans un contexte pourra être différent dans un autre contexte. Une Personne physique particulière reste elle-même de sa naissance à sa mort ; mais si elle est décrite dans le contexte social marital, la relation « est l'époux de » n'existe que dès le moment du mariage au moment du décès ou au moment du divorce. La déclaration d'existence d'une personne physique s'appuiera donc sur des éléments stables ou factuels, tels la 'Date de Naissance', le 'Nom d'État civil', la référence au registre du 'Lieu de naissance' ….

À titre d'exemple, dans la déclaration d'existence de Monsieur Wolfgang Amadeus Mozart l'on mentionnera sa date de naissance « 1756-01-27 » à « Salzbourg » et décédé le « 1791-12-05 » à « Vienne » ; mais l'on ne mentionnera pas qu'il fut « Compositeur de musique », si l'intention est de décrire en détail sa vie.

La figure qui suit représente le détail de l'expression de la déclaration d'existence **d'un tableau** le Léonard de Vinci connu sous divers noms, dont « Mona Lisa » et « La Joconde » : il ne s'agit pas de la déclaration d'existence du modèle du tableau ; mais bien du tableau lui-même.

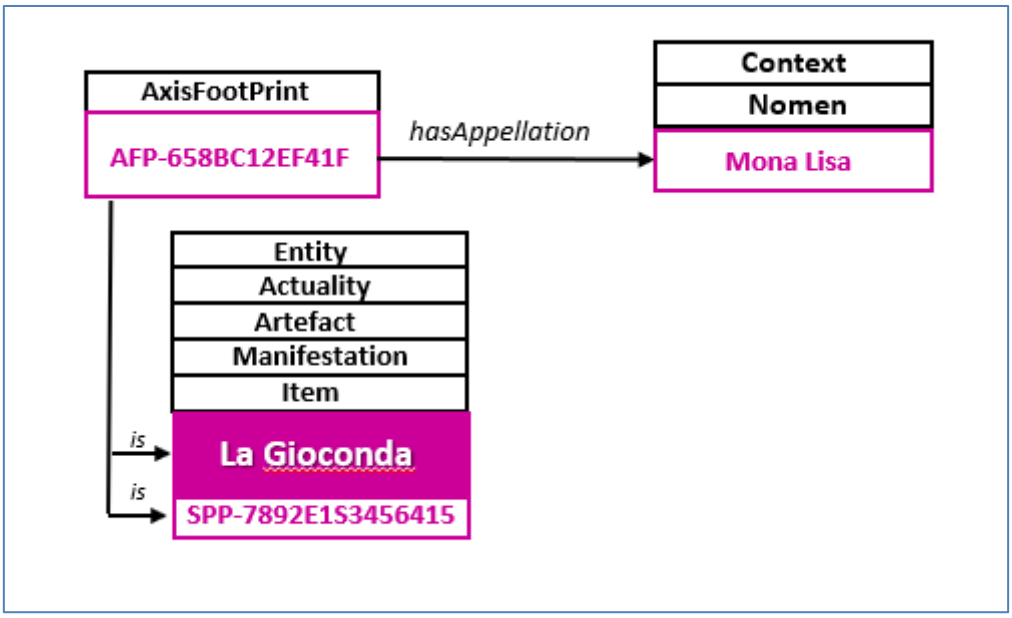

Figure 48: la déclaration d'existence de "Mona Lisa" et "La Joconde"

- L'instance de la classe « AxisFootPrint » a pour l'URI la valeur « AFP-658BC12EF41F ».
- L'instance de classe dont l'on déclare l'existence est un « Item » qui a pour URI la valeur « SPP-789E1S3456415 »
- Deux des métadonnées sont représentées :
  - La Property « Unique Name » qui a pour valeur « Mona Lisa ».
  - L'instance de classe « Nomen » qui imagine que cet 'Item' a été muni d'une protection juridique exprimée par la référence à un « Trade-Mark ».
- D'autres métadonnées devraient être ajoutées pour signifier qu'il s'agit d'une peinture à l'huile, peinte sur des panneaux de bois … mais l'illustration suffit à elle-même.

La figure qui précède peut-être représentée de manière plus compacte ou simplifiée :

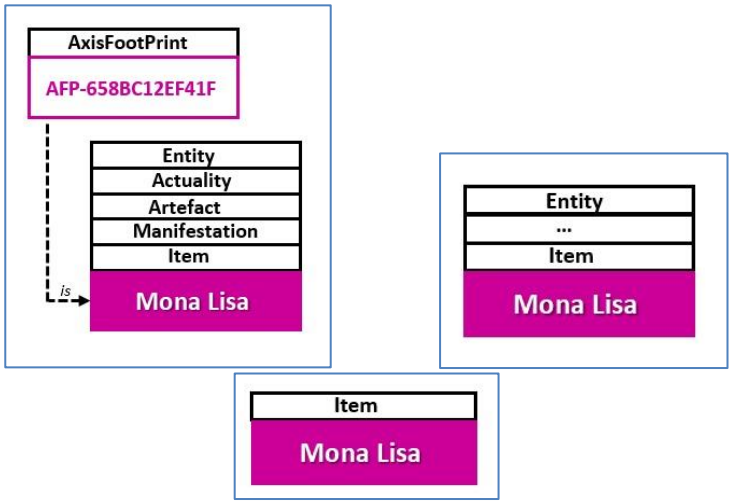

Figure 49: Représentation compacte de la déclaration d'Existence

Le Construct AFP n'est pas d'application à l'intérieur d'une instance d'une « Modelling Technology ».

Le besoin d'ajouter une déclaration explicite de l'existence des Entités n'est pas évident ! Mais la nécessité d'un Construct capable de gérer la diversité des types de données associées à quelque chose a rapidement montré la pertinence, l'adéquation et la souplesse qu'apporte l'introduction de cette Classe 'AFP' « AxisFootPrint ».

La figure qui suit va introduire cette problématique et expliciter les usages de la Classe 'AFP'. Il faut rappeler que la Classe 'Modelling Technology' peut avoir des raffinements :

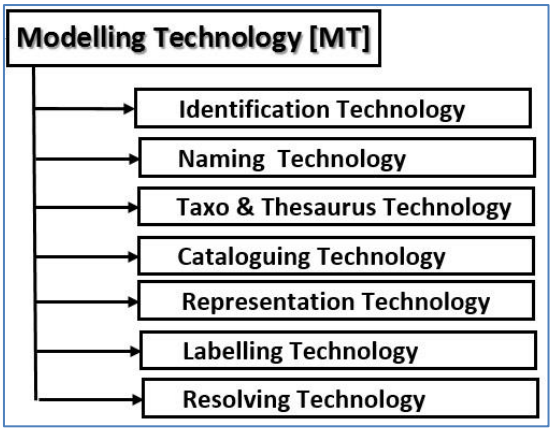

Figure 50: les "raffinements" de la classe 'Modeling Technology'

Le schéma générique d'usage de la Classe 'AFP' est repris à la figure qui suit :

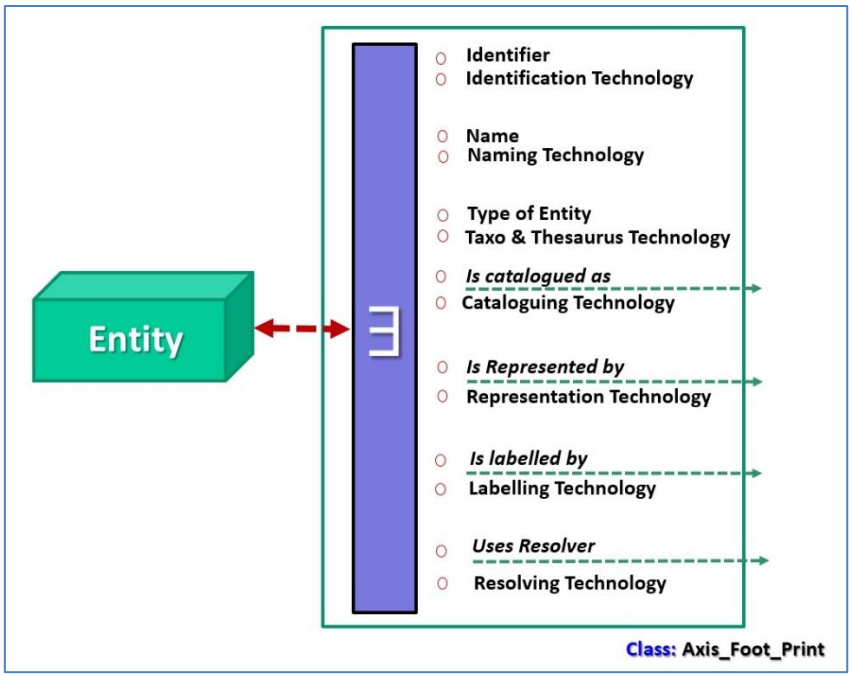

Figure 51 : La Classe générique AFP

Le symbole mathématique ∃ signifie la déclaration de l'existence d'une Entité.

Les instances de la classe 'AFP' sont les porteurs des liens vers les diverses façons d'exprimer des données au sujet des Entités.

Les figures qui suivent expriment le fait que les AFP peuvent être les porteurs des relations qui existent entre les Entités.

La figure qui suit illustre deux types de liens : la présence d'une Entité dans un catalogue (comme 'Parts' dans un 'Registre') ; l'existence de plusieurs représentations de la même Entité.

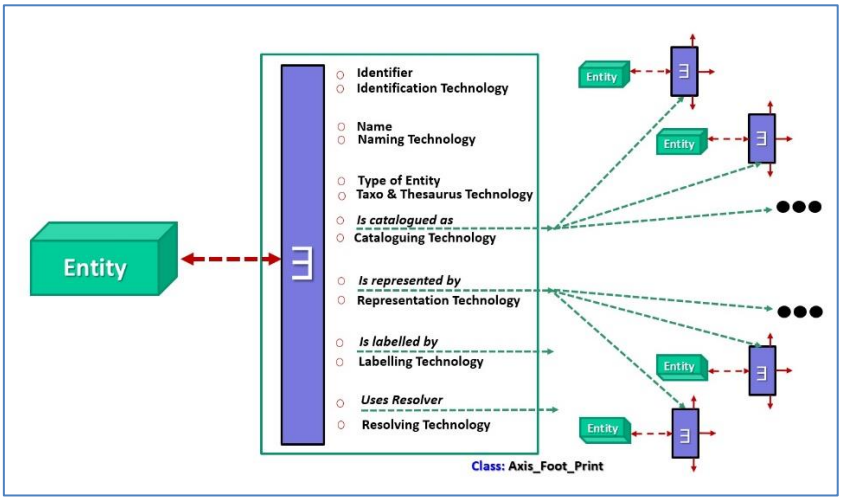

Figure 52 : Existence de plusieurs représentations de la même Entité.

La figure qui suit illustre que l'Entité concernée est équipée de trois types de labels (Bar code // QR code // RFID) et exprime les liens vers les ensembles de données qui les définissent.

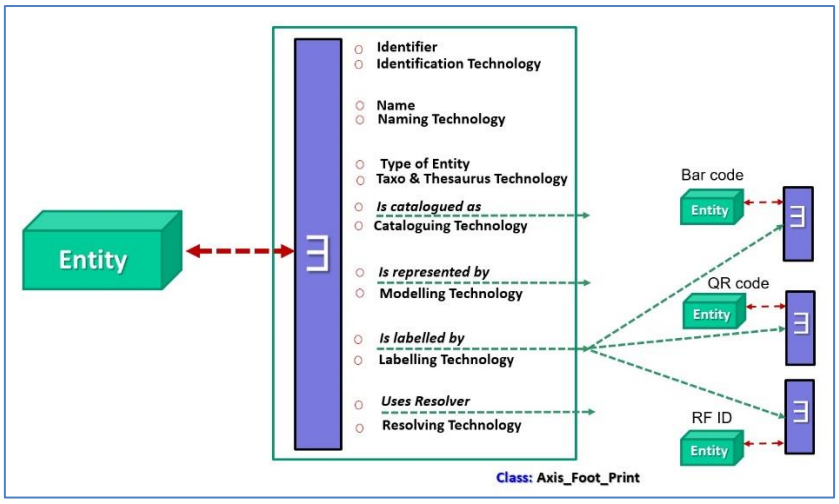

Figure 53 : trois types de Labels (Bar code // QR code // RFID)

La figure qui suit illustre que l'Entité concernée est munie de métadonnées de catalogage suivant plusieurs normes.

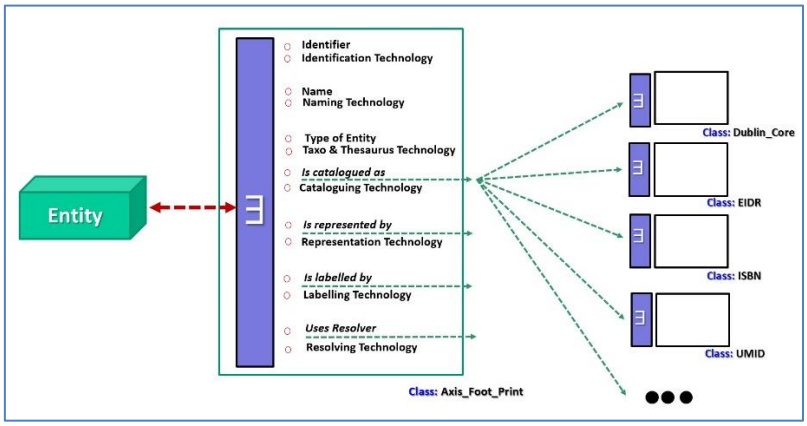

Figure 54 : Métadonnées cataloguées selon plusieurs standards

La figure qui suit illustre par des exemples la souplesse du concept.

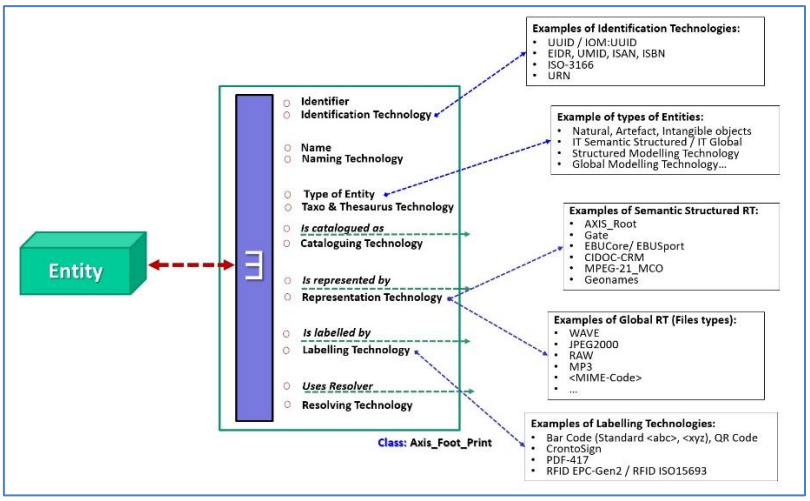

Figure 55 : exemples la souplesse du concept.

Les “Registres” :

AXIS-CSRM organise le fait de pouvoir faire des ensembles cohérents des données représentant les déclarations d’existence des Entités, représentant les Entités et leurs liens. Les registres regroupent les Entités en spécifiant quel est le registre qui est le ‘chef d’orchestre’ dans la gestion de ces représentations. Cette propriété rend possible la gestion des droits, les exports et les évolutions dans la gestion elle-même des life-cycle des ressources portant les bases de données, en particulier les migrations.

La figure qui suit illustre la volonté de regrouper des instances de Constructs pour exprimer un ensemble sémantique cohérent des processus de montage vidéo.

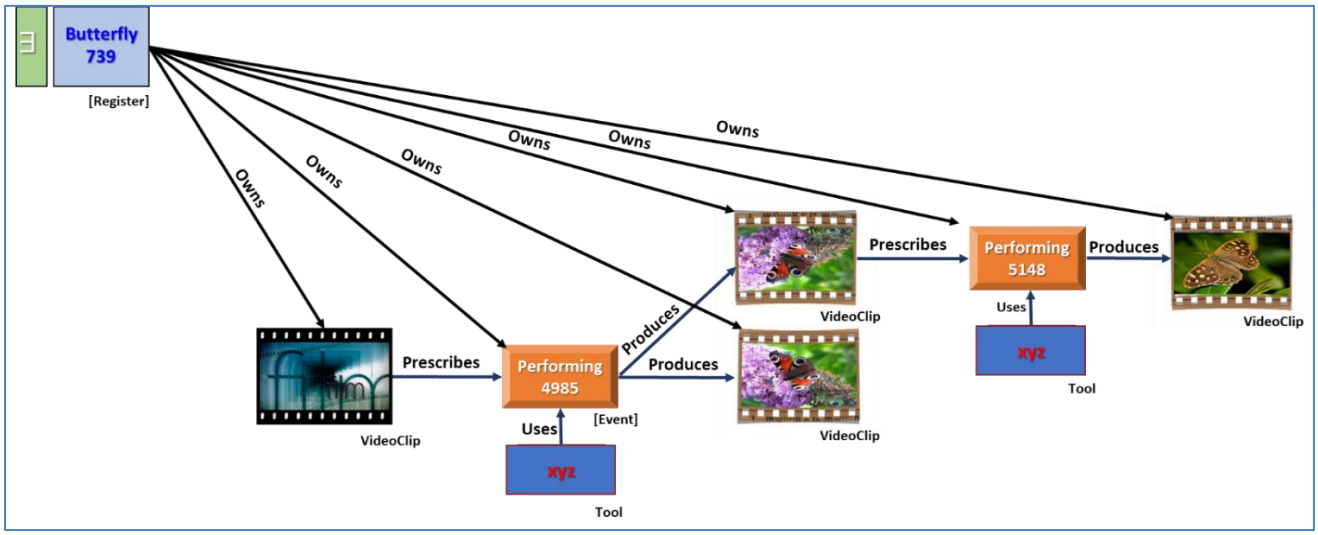

Figure 56 : Registre explicite (Owns - Uses)

Un autre exemple concerne la définition de la sélection de ce qui doit être extrait pour obtenir des « Packages » autonomes et cohérents sémantiquement, ce qui s’exprime aisément par le mécanisme des Registres.

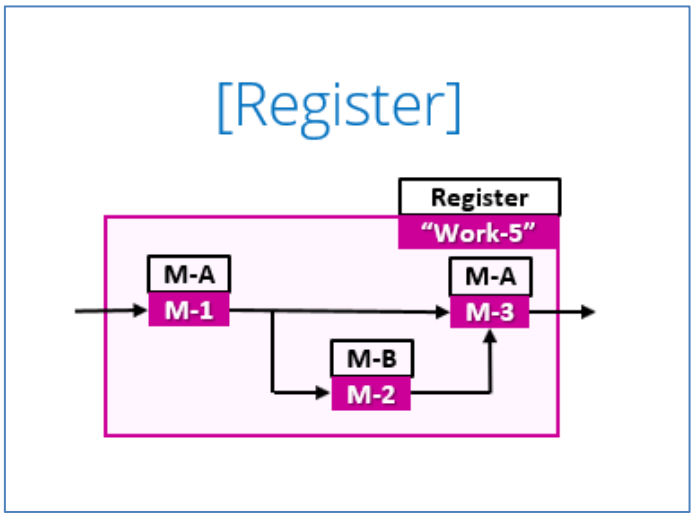

Figure 57 : Symbolisme du regroupement de Registres

La figure 6-72 illustre le symbolisme pour représenter **le regroupement** des instances de **plusieurs modèles** [chacune avec toutes leurs instances de classes, propriétés et relations définies par une des Technologies de Modélisation] et les liens entre ces modèles comme appropriés pour représenter (dans l'exemple) le 'Work-5'. C'est la manière de regrouper tous les Modèles et leurs Relations relatives à une intention. Il faut remarquer que les types de ‘Relationships’ qui lient les instances de Registres à des instances de classes ne peuvent être que de trois sortes :

1. Has
2. Owns

3. Uses

Une instance de classe ne peut être liée à un registre que par un seul « Owns ». Les instances de Constructs importées sont placées dans une instance de la spécialisation de Registre nommée « Proxy ».

Les trois figures qui suivent illustrent des usages typiques du concept de « Register ».

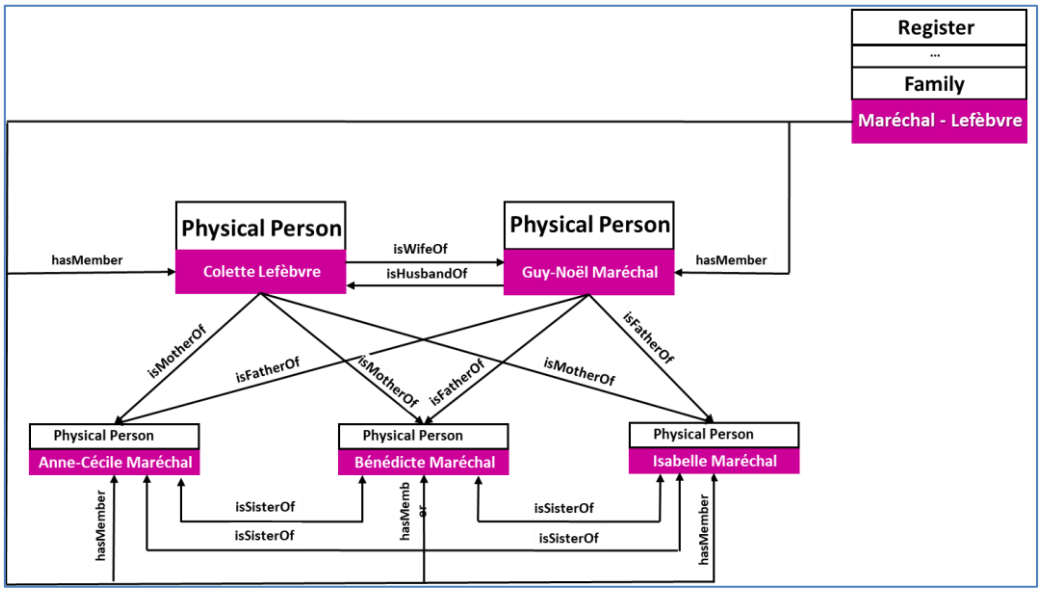

Figure 58 : Illustration du Concept de Registre

La figure montre l'existence d'une instance de la Classe « Family » (une spécialisation de la Classe « Registre »). Cette instance porte le nom de « Maréchal-Lefèbvre » ! La figure montre les cinq 'membres' de ce Registre sous la forme de Relationships du type « *has* ».

- Il faut noter que l'instance de la Classe « PhysicalPerson » dénommée « Colette Lefèbvre » peut aussi être membre d'un autre Registre « Family » dénommé « Lefèbvre-Nols » .
- Il faut noter que l'instance de la Classe « PhysicalPerson » dénommée « Guy-Noël Maréchal » peut aussi être membre d'un autre Registre « Family » dénommé « Maréchal-Rennotte ».
- Il faut aussi noter que l'on peut aussi attacher à ces Registres les scans du 'Livret de Mariage' et le modèle de resolving permettant de savoir dans quel endroit ce 'Livret de Mariage' se trouve.

La figure montre aussi les Relationships existant entre ces cinq membres : « *isWifeOf* » … « *isSisterOf* ». Cet ensemble permet de réaliser des inférences.

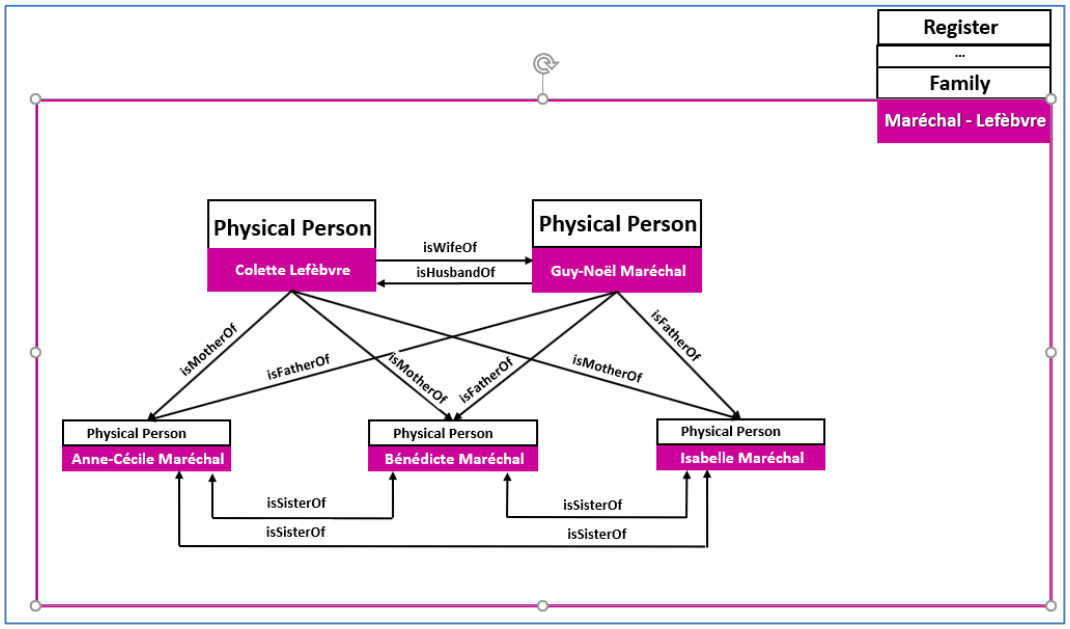

Figure 59 : Illustration du symbolisme grappe pour les registres

Cette figure a le même signifié que la figure précédente, mais en omettant de représenter les Relationships du type « Has ». Le cadre violet rassemble tous les membres du Registre.

La figure qui suit étant la précédente en montrant un autre registre, une carte d'identité et sa représentation. Cet autre Registre est un 'proxy' du Registre officiel tenu par l'autorité communale ; laquelle est l'entité productrice de la Carte d'Identité (comme objet) qui la dépose en usage à son titulaire (dans l'exemple, Guy-Noël Maréchal).

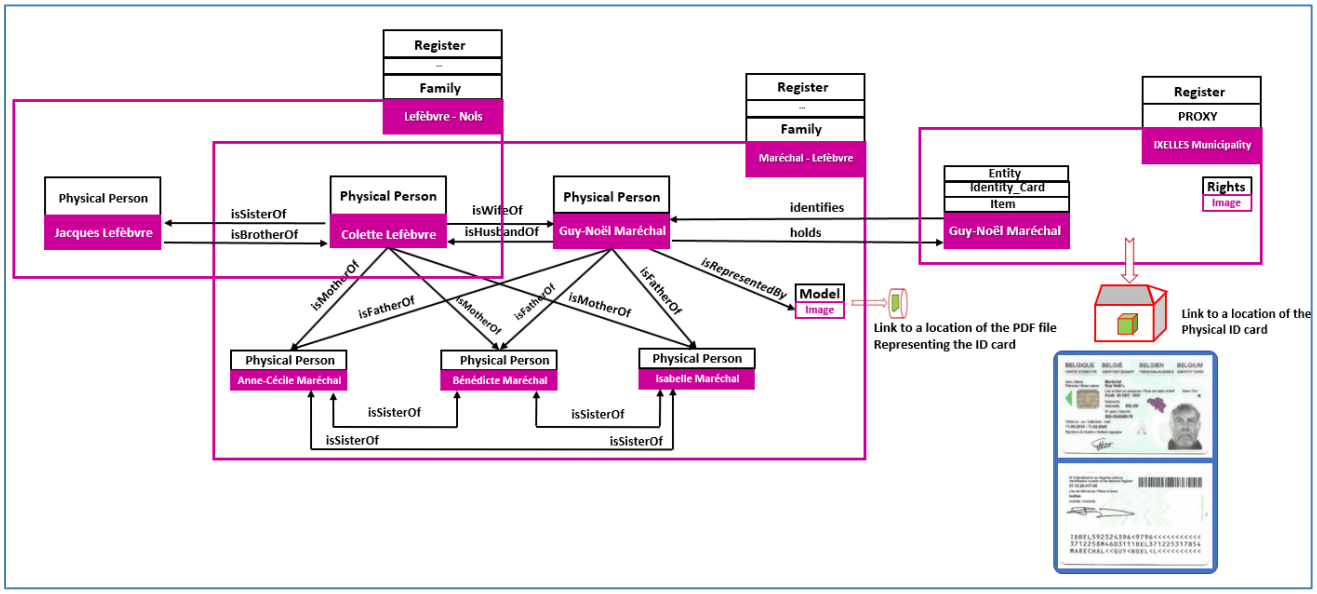

Figure 60: Illustration de la flexibilité du concept de registre

La modélisation à « États Finis » : les « Segments » & les « Points »

À la section 6.5.3.1 [General Modelling of the "Entities"], le concept de modélisation à « États Finis » a été introduit pour tenir compte de la gestion et de la représentation des processus et de leurs interactions avec des « Entités » et des « Modèles ».

Dans le « Life-Cycle » des processus, il faut habituellement distinguer plusieurs phases :

- Définition du processus
- Présentation de cette définition à des ressources (humaines et/ou machines) et un contexte capable de transformer cette définition en une action. Cette définition porte le nom de « Prescription ».
- Activation (manuelle ou automatique) de la Ressource de telle sorte que le processus s'exécute, réalisant le traitement qui a été prévu ou rencontrant des difficultés à réaliser ce traitement et/ou ne le réalisant pas du tout.
- Usuellement, le processus commence par identifier et enregistrer un 'Vecteur d'état' de départ et un 'Vecteur d'État' de fin de processus. Ces vecteurs d'États contrôlent les enchaînements des processus (via des activations conditionnelles) suivant un enchainement programmé de processus [WorkFlow]. En particulier, en cas de problèmes, un processus d'Exception Handling va activer soit l'arrêt des traitements, soit un traitement de sauvegarde adéquat.
- Ces processus conduisent à des résultats suivant diverses formes : ces activations de ressources adéquates, suivant des protocoles associés, pourront générer un flux artefact (le son issu d'un haut-parleur …) ou un objet artefact (une impression 3D …) ou la modification d'une « Entité » ou la création d'un nouveau modèle ou … .
- Enregistrement des flux de traitements [WorkFlow] et des 'Vecteurs d'États' en vue d'archives, de traçabilité, d'analyse de qualité … .

L'on notera que l'informatique ne peut représenter que des **faits**, des **concepts** ou des **instructions** en vue de déclarer l'existence de choses ou de les modéliser à autant d'états de leurs évolutions que de nécessaire, souhaité,

pertinent, réaliste … dans le contexte courant. L'informatique gère donc cette capacité suivant l'approche dite à 'États Finis' et elle est elle-même une machine à états finis.

Pour que l'on puisse représenter et gérer les évolutions des Entités (dans le temps et dans l'espace) et de leurs liens, il faut que les 'Constructs' informatiques représentant ces évolutions aient la même puissance de représentation que les 'Constructs' nécessaires pour représenter les Entités elles-mêmes. La puissance et les propriétés de représentations des évolutions-spatio-culturo-socialo-…-temporelles des Entités trouvent leur source dans la modélisation des processus tels que les « Réseaux de Pétri » ou des extensions à la logique du premier-ordre ou ses extensions dans le calcul des prédicats..

Dans AXIS-CSRM, les porteurs de cette modélisation sont les Classes « Point » et « Segment ». À un « Point » seront associés des « états » (Espace ; Temps ; Concepts ; …) et à un « Segment » seront associées des représentations des Entités (soit qu'elles soient considérées comme 'statiques' ; soit qu'elles soient associées à la représentation des 'processus').

Usuellement ces concepts sont représentés par des combinaisons de Properties et de Relationships. Dans AXIS-CSRM ces concepts sont réifiés doublement : à la fois porteurs de l'existence d'un état ou d'un processus et porteurs de modèles définissant ces états et processus.

- Le "**Point**" est le porteur de l'existence d'un « État » et des « Modèles » qui le définissent
- Le "**Segment**" est le porteur des sous-modèles représentant une « Entité ».

Les «**Points**» et «**Segments**» expriment la structure reliant les sous-modèles et les états. Ils sont liés par des relations telles que «isFollowedBy» ou «triggers» qui constituent l'expression de la structure (temporelle ou topologique ou ...) reliant les modèles et les états. Lorsque le modèle se rapporte à une «Entité» statique, le segment sera qualifié de «**stable**»; alors que lorsque le modèle se rapporte à une «substance» dynamique, le segment sera qualifié d'«**évolutif**».

Le modèle d'un son pourrait être représenté comme un fichier MP3; il sera tenu par un «**segment stable**».
Le modèle du processus de génération de «son en perception» à partir de ce fichier MP3 sera tenu par un «**segment en évolution**»

Les paramètres d'activation et les aspects contextuels seront maintenus par un «**point**» lié au «segment stable» pour exprimer la nature «prescriptive» du fichier MP3 et lié au «segment évolutif» pour exprimer «l'activation» du processus.

Les concepts de "**Point**" et "**Segment**" sont exprimés par Classes! Ils peuvent être utilisés après que les modèles ont été mis à disposition! Cela signifie que l'approche est «ascendante»:

- Ces concepts pourraient être liés dans la hiérarchie à travers les relations entre les «Points».
- Ces concepts pourraient être utilisés comme la structure des flux de travail, les réseaux de Petri et la modélisation similaire des processus
- Ces concepts pourraient être utilisés pour les représentations managériales des projets

Des exemples seront présentés ci-après illustrant ces capacités et modalités. Le « Point » est la généralisation du concept d'événement et le « Segment » la généralisation du concept de situation pour couvrir les représentations des processus eux-mêmes. Par exemple, le « Point » pourra représenter une 'borne' (milestone) dans la programmation d'un projet. Le « Segment » peut être orienté (pour représenter des graphes, par exemple) : ce sont les relations (relationships) qui lient les extrémités des « Segments » à des « Points » ou à des « Segments » qui expriment ces vecteurs potentiels.

Les classes de « Segment » et « Point » sont aussi les porteurs des synchronisations entre des détails à l'intérieur des Modèles (par exemple, la synchronisation entre la vidéo et le son ; de même, pour synchroniser les sons et images quand un événement est filmé dans un lieu et que les commentaires associés sont faits par un journalise dans un autre lieu. De même « Segment » et « Point » peuvent être les porteurs de la gestion des évolutions ressources qui génèrent données. Dans la figure qui suit, les divers 'Fragments' sont accrochés à des 'Points' [fragments ponctuels] ou à des 'Segments' [fragments de période].

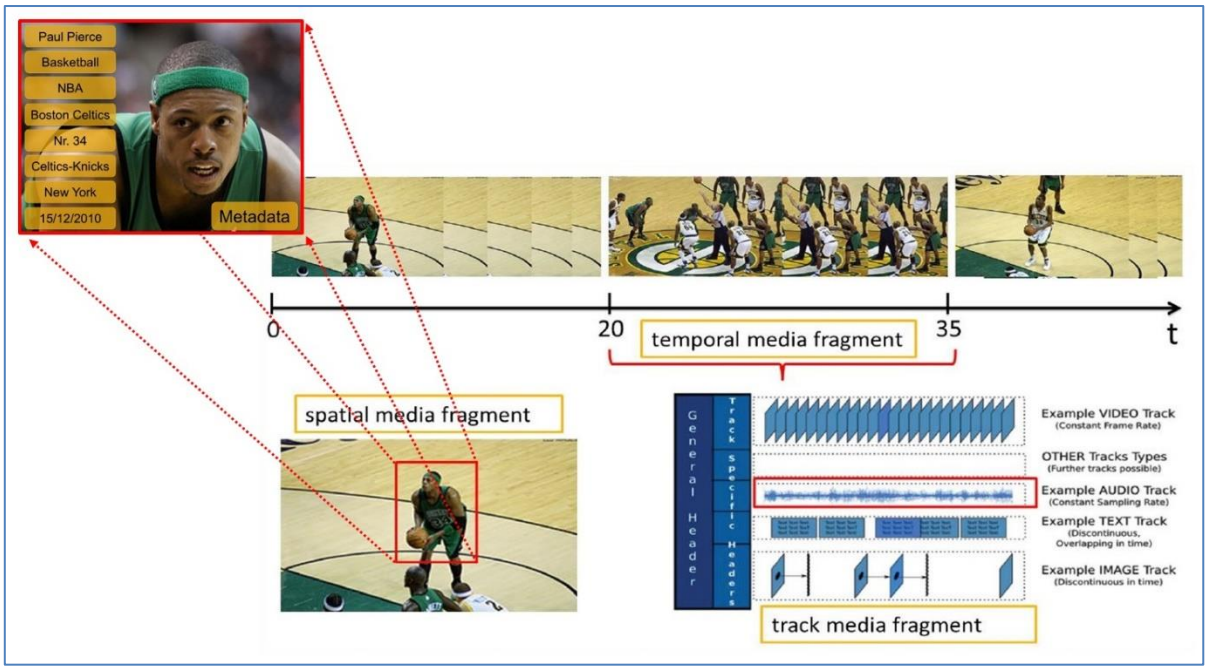

Figure 61 : Exemple d'utilisation de 'Point' et 'Segment' pour la structuration d'un média

La figure suivante montre l'articulation de structure des Constructs. Les exemples des sections suivantes illustreront des usages types de ces Constructs :

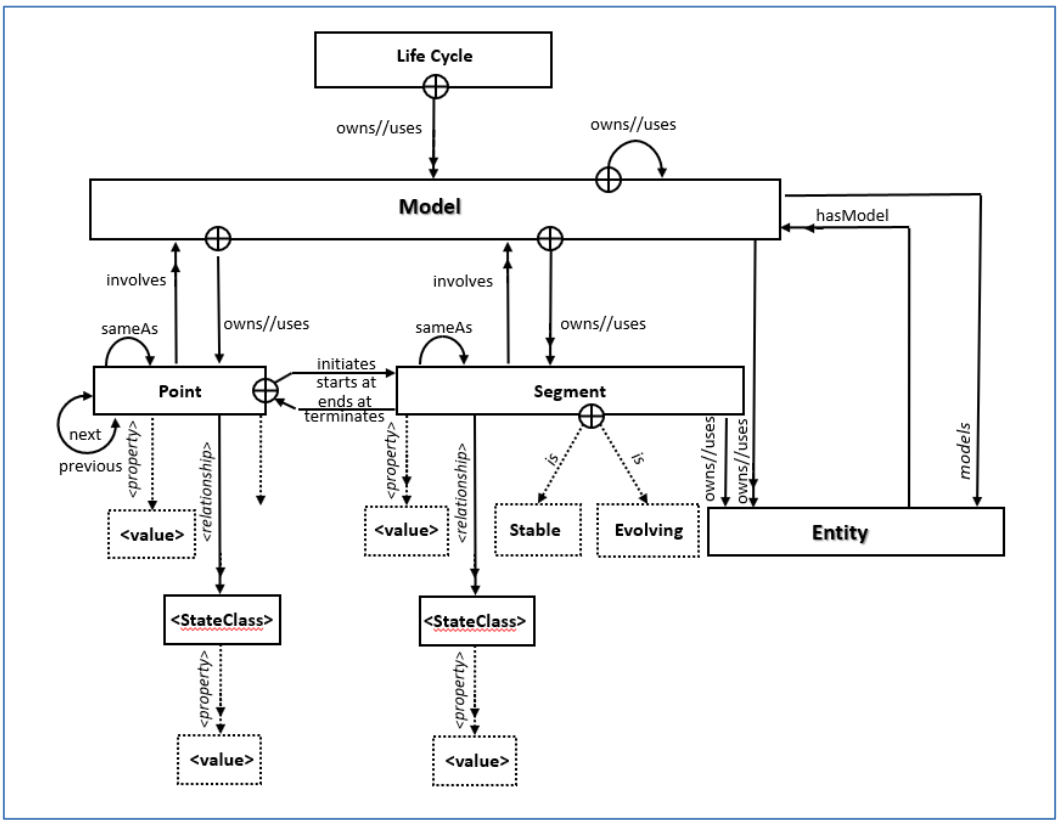

Figure 62 : Modèle de données générique pour les classes 'Point' et 'Segment'

La modélisation des « Entités » et des « Métadonnées »

Les Entités ne sont pas indépendantes entre elles et ce monde est en perpétuelle évolution. Usuellement, et en particulier dans le monde des archives (la norme OAIS en est un exemple significatif) chaque chose est sensée

être munie de métadonnées et d'un fichier média de représentation. En fait, les représentations sont des métadonnées plus détaillées. La nature profonde du monde est évolutive dans un espace-temps et toutes les choses que l'on reconnait comme disjointes sont reliées entre elles. De plus, les déclarations d'existence des choses et les modèles des choses ne sont pas stables (erreurs ; contextualisation ; enrichissement des représentations et des liens …). Nous sommes donc contraints de modéliser les choses de diverses façons.

Le concept de « **Modelling Technology** » a été déjà rencontré pour définir les formalismes utilisés, d'abord pour coder les **URI**s identifiant les instances de Classes ; puis pour représenter et coder les instances des '**Properties**' et enfin dans le contexte des déclarations d'existence.

Le concept de « **Modelling Technology** » va être utilisé pour définir les formalismes utilisés pour représenter les « Entités » et les « Métadonnées » qui s'accrochent aux déclarations d'existence.

La modélisation des « Métadonnées ».

La modélisation des « Métadonnées » a été rencontrée de manière simple dans le cas des instances de Properties directement associées à des instances de Classes des Entités. La représentation des métadonnées via les ontologies est bien connue et disponible en open source. L'essentiel est de ne choisir que les Properties et Classes qui ont un caractère stable dans le Contexte donné.

Un exemple est illustré par la figure qui suit :

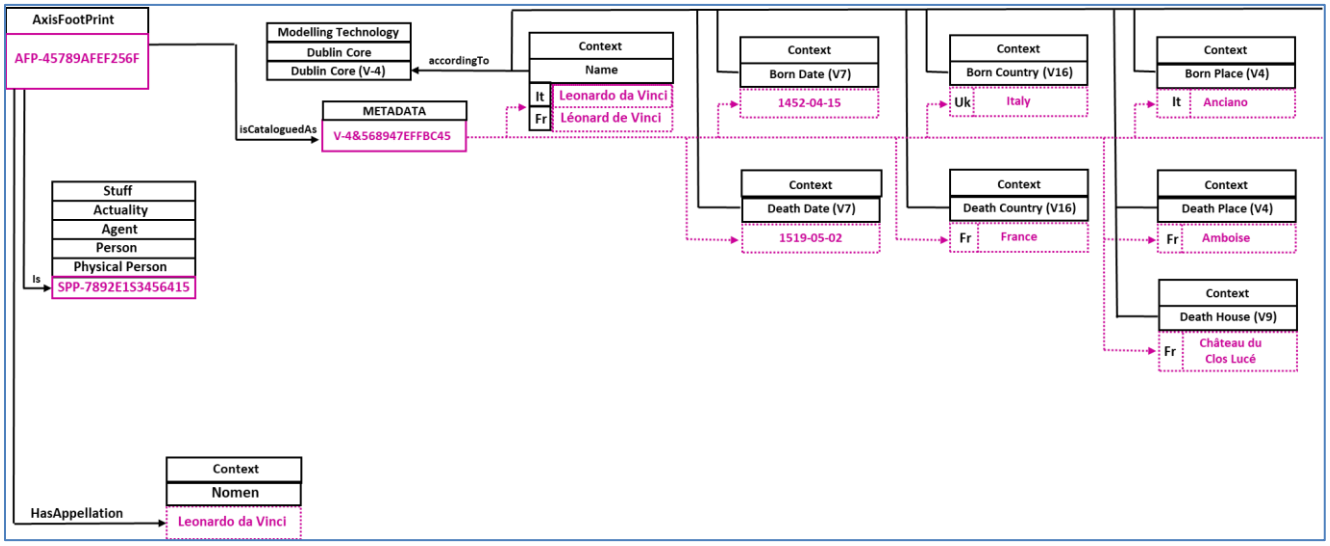

Figure 63 : Exemple d'une déclaration d'existence de «Leonardo da Vinci»

Une seconde façon de déclarer l'existence dans le contexte local est d'ajouter la référence au Registre des Baptêmes.

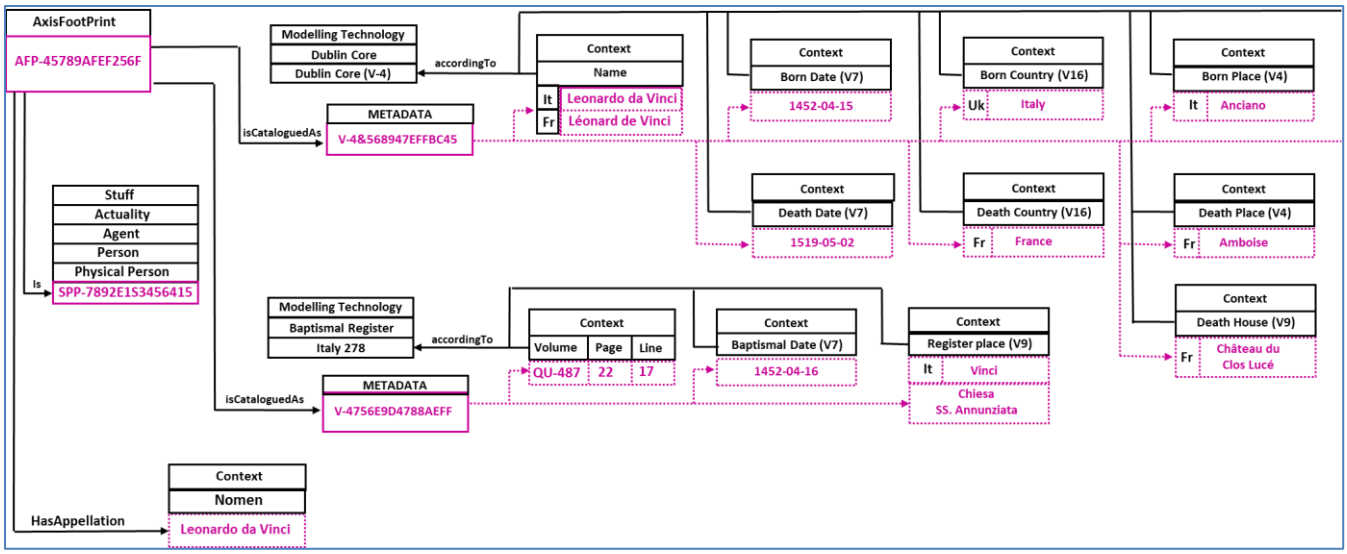

Figure 64 : Exemple de deux manières différentes de déclarer l'existence de «Leonardo da Vinci»

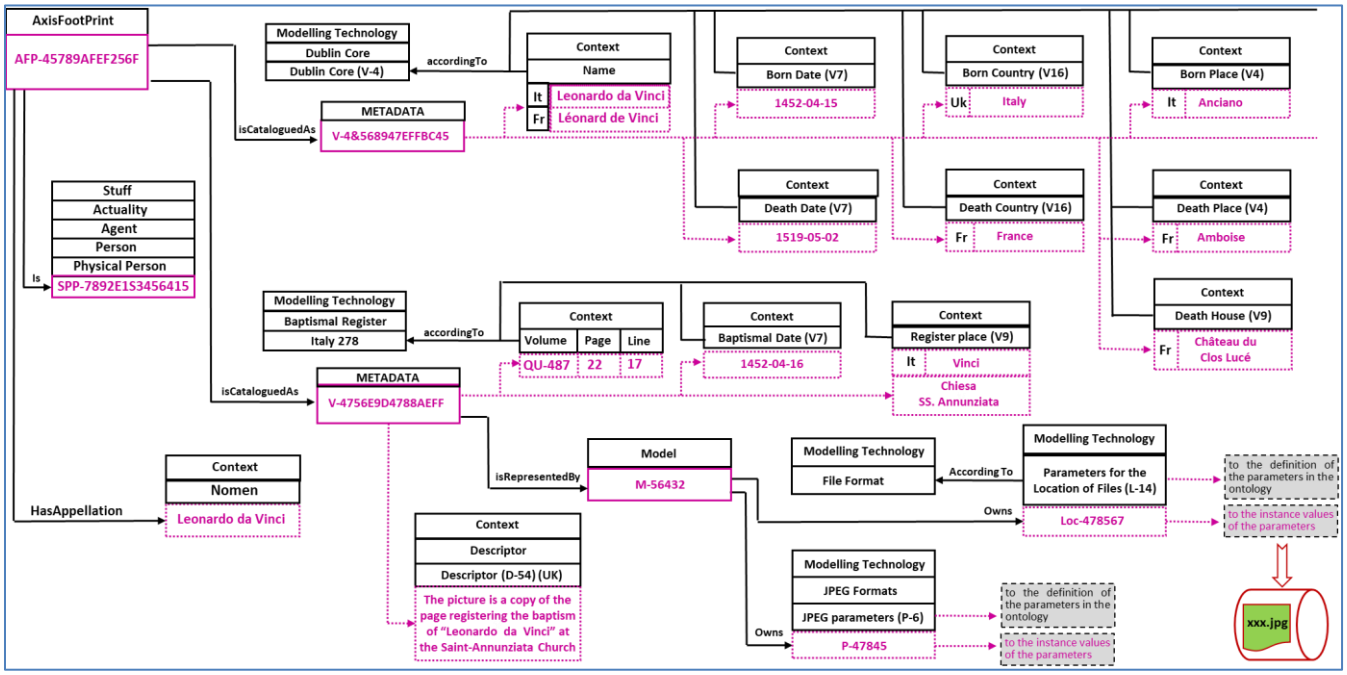

Figure 65 : Exemple de la façon de lier à la représentation de la page du registre de baptême

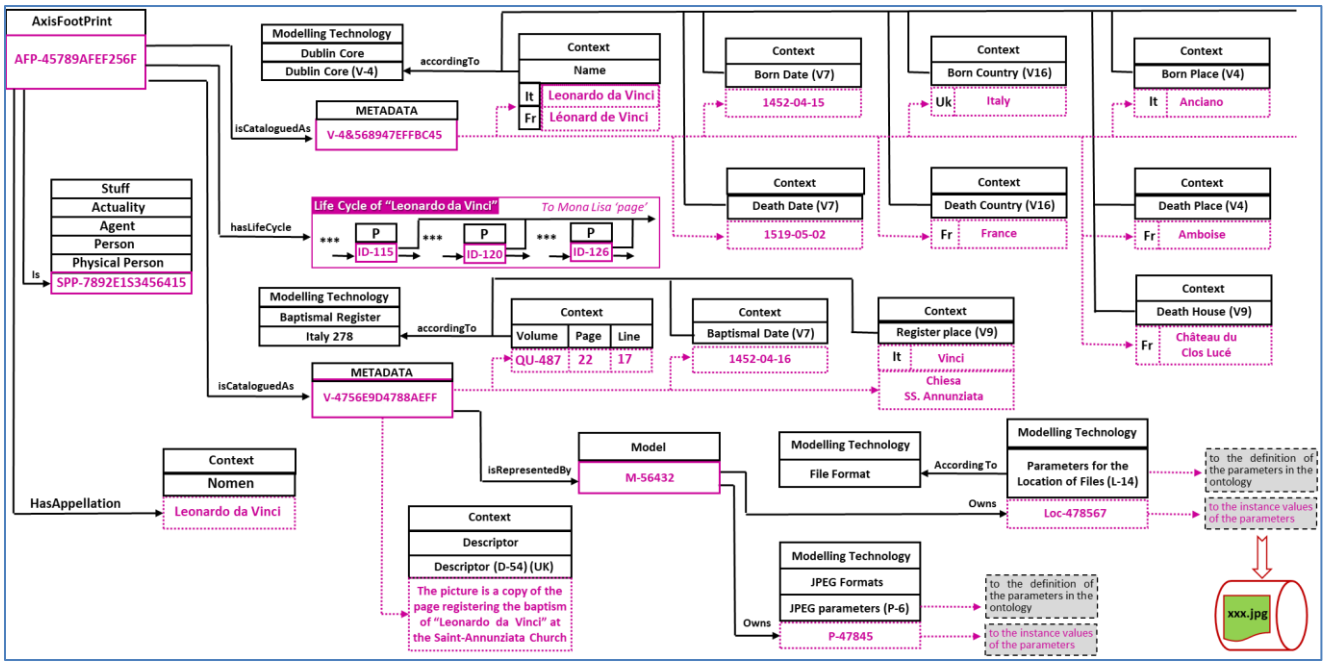

Figure 66 : Lien entre le «Registre» contenant le «cycle de vie» et la personne «Leonardo da Vinci»

La modélisation des « Entités » :

La figure qui suit va être progressivement expliquée :

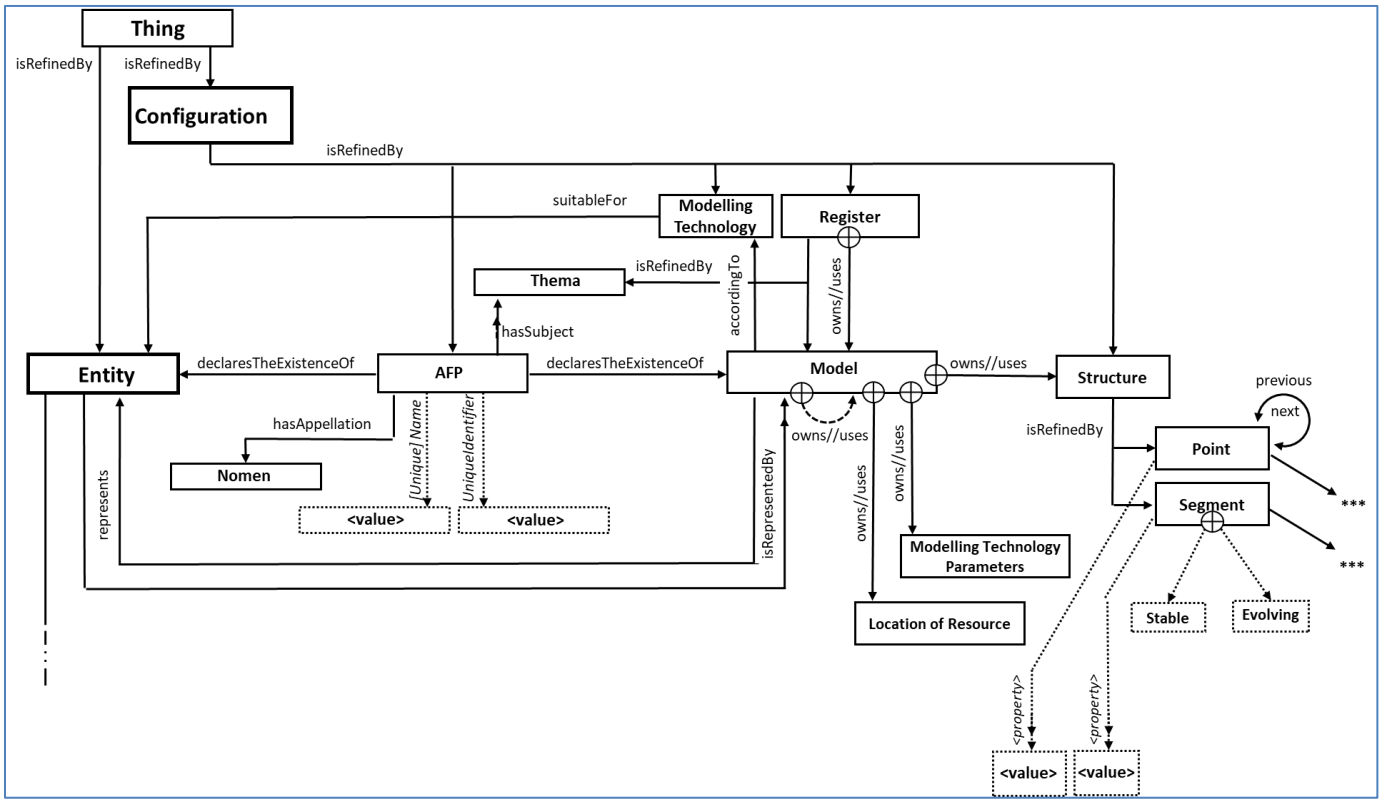

Figure 67: la modélisation des "Entités"

La figure qui suit illustre la modélisation du 'Life Cycle' d'une Entité avec des Points et des Segments :

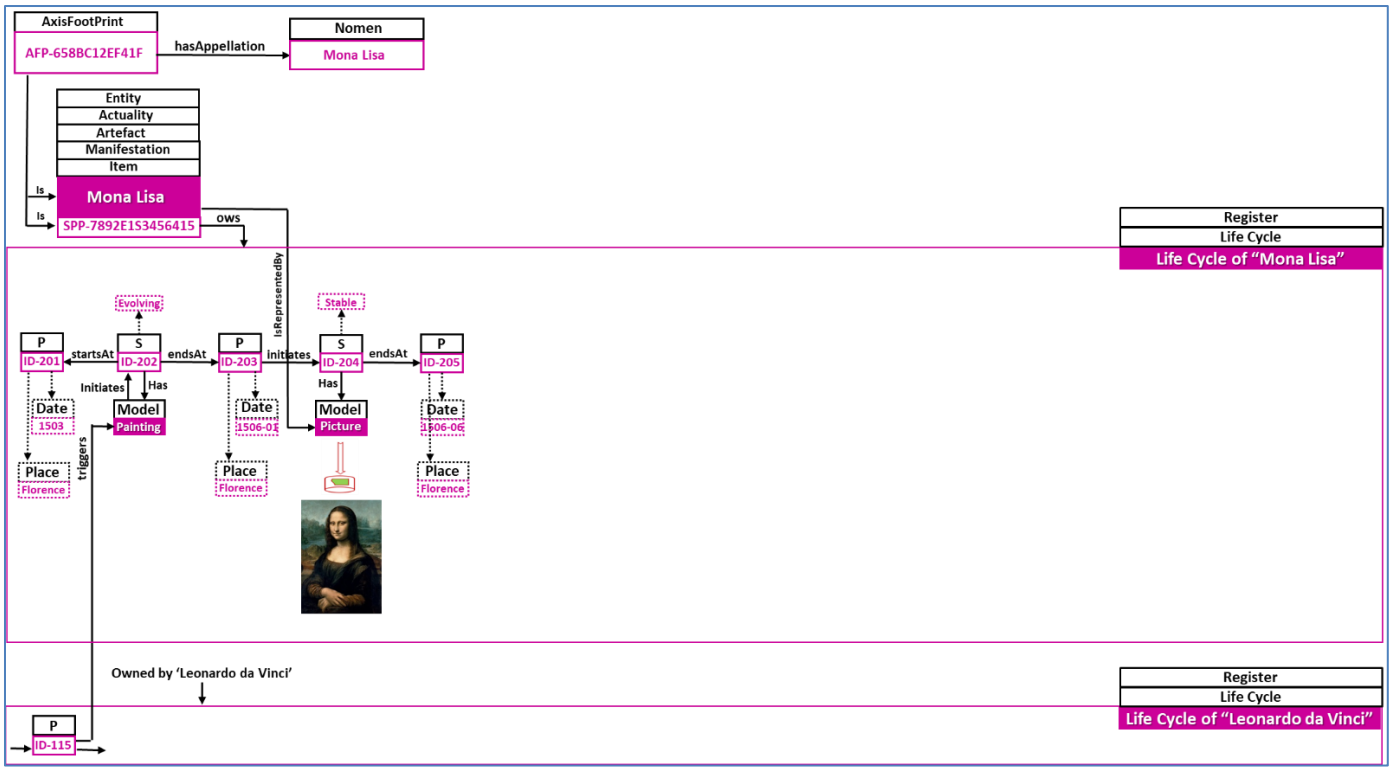

Figure 68 : La modélisation du Cycle de vie des Entités[8]

Pour la modélisation des Entités, deux cas peuvent être distingués :

Modèles sémantiques

Il s'agit des cas où la définition de la « Modelling Technology » peut s'exprimer exclusivement en utilisant les trois types de Constructs ontologiques (Classes, Relationships et Properties).

Un exemple d'un tel cas serait la **transcription sémantique** complète d'une pièce de théâtre : l'ontologie définissant cette façon de modéliser la complexité d'un tel texte est assez immédiate à imaginer. Un exemple illustratif d'un tel cas sera proposé dans l'une des sections suivantes.

Un autre exemple serait de représenter une **photo** (en s'inspirant du format de fichier DNG d'ADOBE [une version ouverte du format RAW]) comme :

- une instance de la Classe '**Photo Properties**' agglomérant un groupe de métadonnées exprimées comme des Properties : Appareil / Focale / Ouverture / Coordonnées du lieu / Moment / Vitesse d'obturation / …
- une instance d'une spécialisation de la Classe '**Matrice**' qui représenterait une matrice de pixels [« Pixel Matrix »] : par exemple, le format de télévision, cinéma et photo 4K a une résolution de 4 096 pixels par ligne et 2 160 lignes par matrice. La couleur de chaque pixel serait codée par trois nombres, par exemple :

  Vert: 16 bits - Bleu : 16 bits - Rouge : 16 bits

  Une telle résolution donne donc 4096 x 2160 = 8 847 360 pixels par 'Photo' ; soit 8 847 360 x (3x16) = 4 246 733 280 bits ou encore en consommant six Bytes par pixel (48 bits pour coder 48 bits) 53 084 160 MB par photo.
- Une instance d'une spécialisation de la Classe '**Matrice**' qui représenterait une '**Proxy Vignette**' de la photo comme une matrice de pixels [« Pixel Matrix »] à faible résolution: par exemple, 256 pixels par ligne et 135 lignes par matrice. La couleur de chaque pixel serait codée par trois nombres, par exemple :

  Vert: 4 bits - Bleu : 4 bits - Rouge : 4 bits

[8] Cette figure est développée dans le Tutoriel 3 : gestion des Patrimoines Documentaires

Une telle résolution donne donc 256 x 135 = 34 560 pixels par 'Vignette' ; soit 34 560 x (3 x 4) = 414 720 bits ou encore en consommant deux Bytes par pixel (16 bits pour coder 12 bits) : 34 560 x 2 = 69 kB par Vignette.

- Une instance d'une spécialisation de la Classe 'Matrice' qui représenterait la table de calibration des couleurs.

Modèles sémantiques hybrides :

Il s'agit des cas où la définition de la « Modelling Technology » s'exprime en utilisant les trois types de Constructs ontologiques (Classes, Relationships et Properties) pour renvoyer vers une ressource externe à la Base de données du contexte (usuellement vers un Fichier situé sur un Volume ; pour donner les valeurs des paramètres de format associés à ce fichier ; pour définir les ressources capables d'ouvrir ce fichier ; pour donner les liens vers la définition de ce format de ressource [fichier …] externe (typiquement la référence à une Norme ISO ou W3C ou …) et pour assurer l'indirection vers cette ressource via un Résolveur (les ressources peuvent changer de place ou être remplacées ou …).

Il existe de nombreuses raisons pour garder des modèles hybrides : des représentations partiellement non sémantiques. La plus évidente est celle de l'efficacité ! Dans l'exemple précédent, la taille de représentation d'une image est 53 MB ; si elle est représentée comme un fichier de format JPEG, elle sera typiquement cinq fois moins lourde sans perte significative [10 MB] et vingt fois moins lourde avec une perte de qualité encore raisonnable [3 MB].

Une autre raison est que dans bien des cas, les originaux des fichiers existent et doivent être sauvegardés pour des raisons légales.

Pour des raisons de sécurité, les modèles sont sauvegardés en plusieurs copies en des lieux différents. Certains des modèles sont des avatars analogiques, artefacts issus de modèles …Certains des modèles sont représentés conformément à des formats propriétaires. Il est particulièrement pertinent ici de mentionner les ressources (CODEC ou autres) capables de les ouvrir. En variante ou en complément, l'expression du lien vers des 'Proxy' de qualité adaptée pourra offrir un palliatif en cas d'impossibilité d'exploiter l'original …

Dans de nombreux des cas, les modèles ne couvrent pas uniquement des représentations des « Entités », mais aussi les procédures permettant à accéder à une '**Actuality**'. Une image de très haute résolution, représentant « La Joconde », ne remplace pas l'acte de se rendre au Louvre pour voir le tableau. Par ailleurs, accéder à un incunable très fragile ou très précieux demande une compétence, des précautions, une traçabilité … qui doivent être codifiées.

Des exemples vont illustrer ces concepts dans les sections qui suivent.

La représentation d'un modèle passe toujours par l'usage d'une instance de la Classe '**Modèle**' qui est une spécialisation de la classe '**Registre**'. Ce point va être explicité dans la section suivante.

Une illustration du cas hybride est donnée en liaison avec l'exemple précédent ; on y présente le lien vers un modèle du Life Cycle de Léonard de Vinci et une représentation de la page du Registre de Baptême de Léonard de Vinci :

La figure qui suit montre l'articulation qui va être expliquée plus tard :

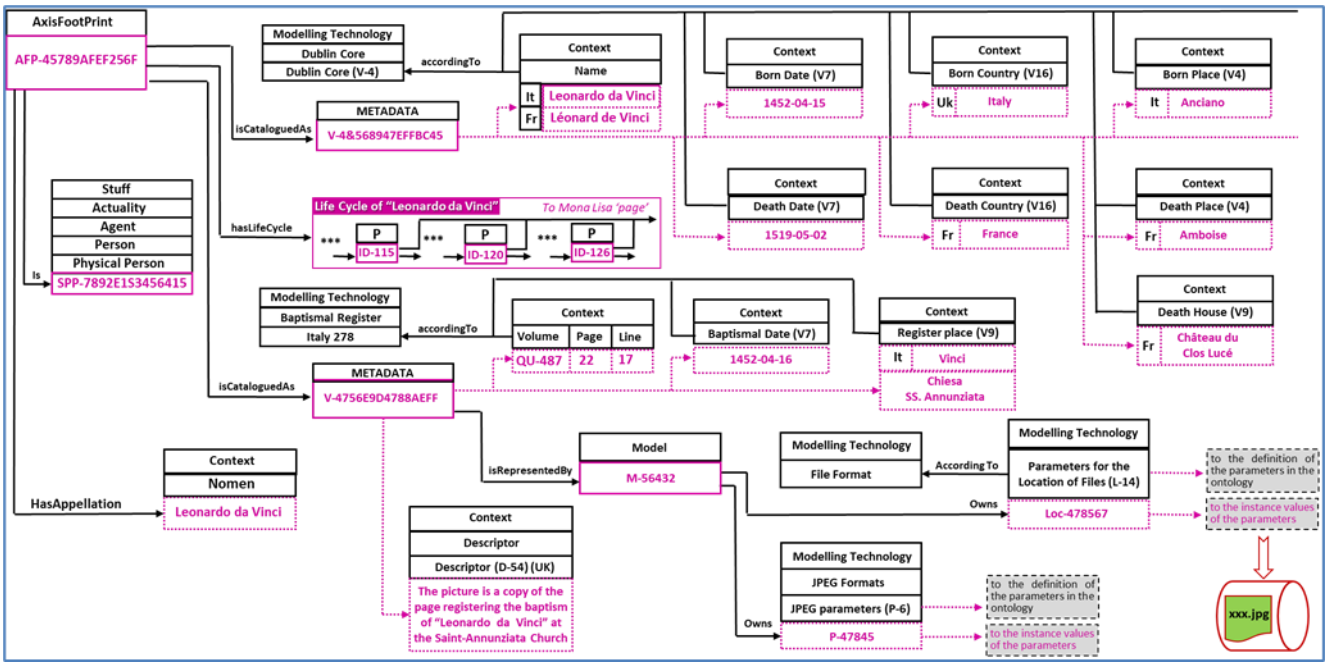

Figure 69 : Le modèle intégré de la déclaration d'existence de «Leonardo da Vinci»

Association du « Modèle de données » avec le « Modèle Fonctionnel »

Ces deux modèles sont inters corrélés !

Les modélisations classiques se bornent à déclarer l'existence des 'Entités' via des métadonnées et, quand cela s'y prête à mémoriser des « EdiData » sous la forme de fichiers annexés.

Dans AXIS-CSRM, la déclaration d'existence des 'Entités' possède ses propres Constructs et de même pour la représentation des 'Entités'!

Le terme « Construct » désigne tous les moyens par lesquels il est possible de représenter les 'Choses' dans AXIS-CSRM : essentiellement :

- les 'Classes',
- les 'Properties'
- et les 'Relationships',
- leurs spécialisations et leurs occurrences.

De plus, la représentation des 'Choses' peut couvrir les processus associés (en particulier les processus ayant conduit à l'existence d'une 'Entité', à son 'Enjoying' et aussi aux processus qui ont conduit aux évolutions des 'Entités' dont on a déclaré l'existence). Il est même possible de déclarer l'intention d'existence d'une 'Entité' avant qu'elle n'existe, soit en réalité soit en modèle. Il s'agit en fait d'une 'Entité' impliquée dans la déclaration d'existence et de la modélisation de l'intention de créer cette 'Entité'. C'est la modélisation du 'Projet'.

Voici comment représenter le Modèle de données d'une des figures du Modèle fonctionnel.

La plus simple, mais aussi la plus abstraite est représentée par :

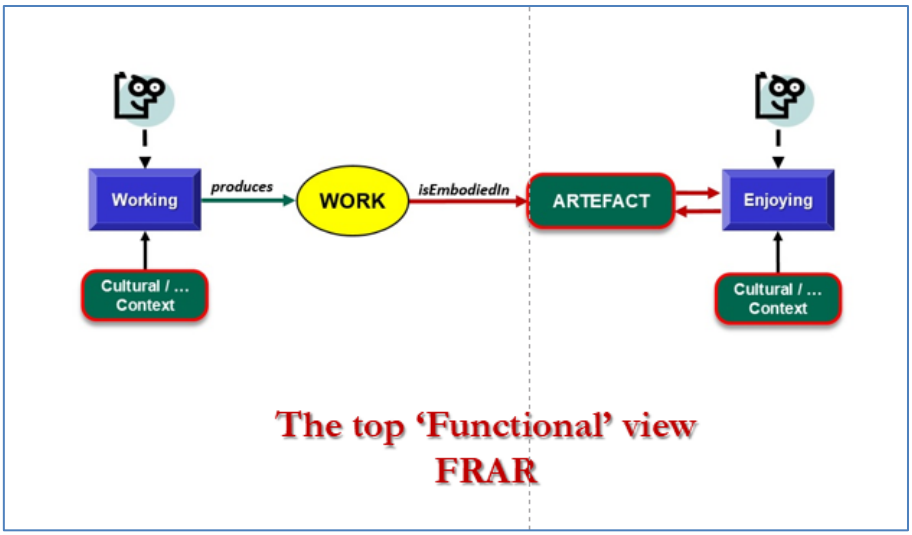

Figure 70: le Modèle de Données de l'une des figures du Modèle Fonctionnel

Ce modèle de données va être décomposé en trois étapes :

I. La modélisation d'un <Working> via le processus de 'Working' et l'intervention de l'Agent': un registre ; deux points ; un segment, un modèle du processus 'Working', deux états et des Relationships.

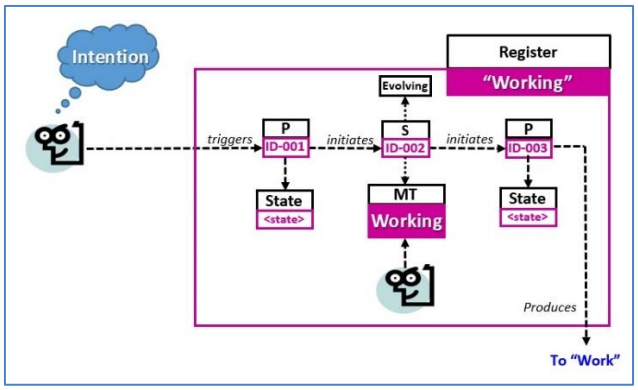

Figure 71 : la modélisation 'données' d'un 'Working'

II. La modélisation d'un <Work> par un modèle, deux points, un segment et des Relationships : un registre ; deux points ; un segment, le modèle d'un artefact 'Work', deux états et des Relationships.

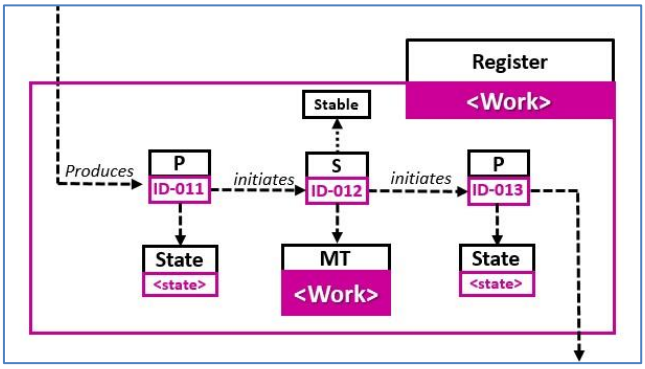

Figure 72 : la modélisation 'données' d'un 'Work'(oeuvre)

III. La modélisation d'un <Enjoying> par deux modèles (le processus d'Enjoying' et l' 'Agent'), deux points, un segment et des Relationships : un registre ; deux points ; un segment, le modèle d'un processus 'Enjoying, deux états et des Relationships.

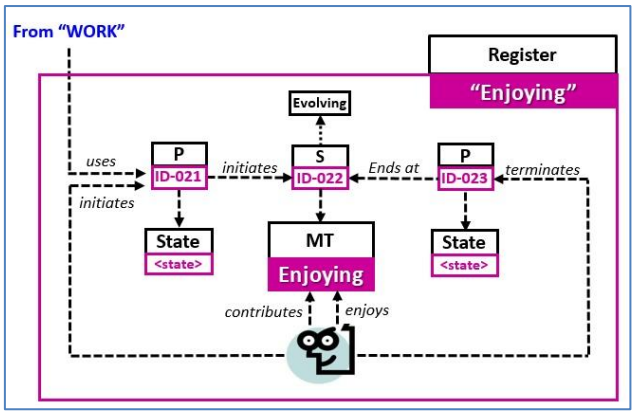

Figure 73: la modélisation 'données' d'un 'Enjoying'

La modélisation d'ensemble devient :

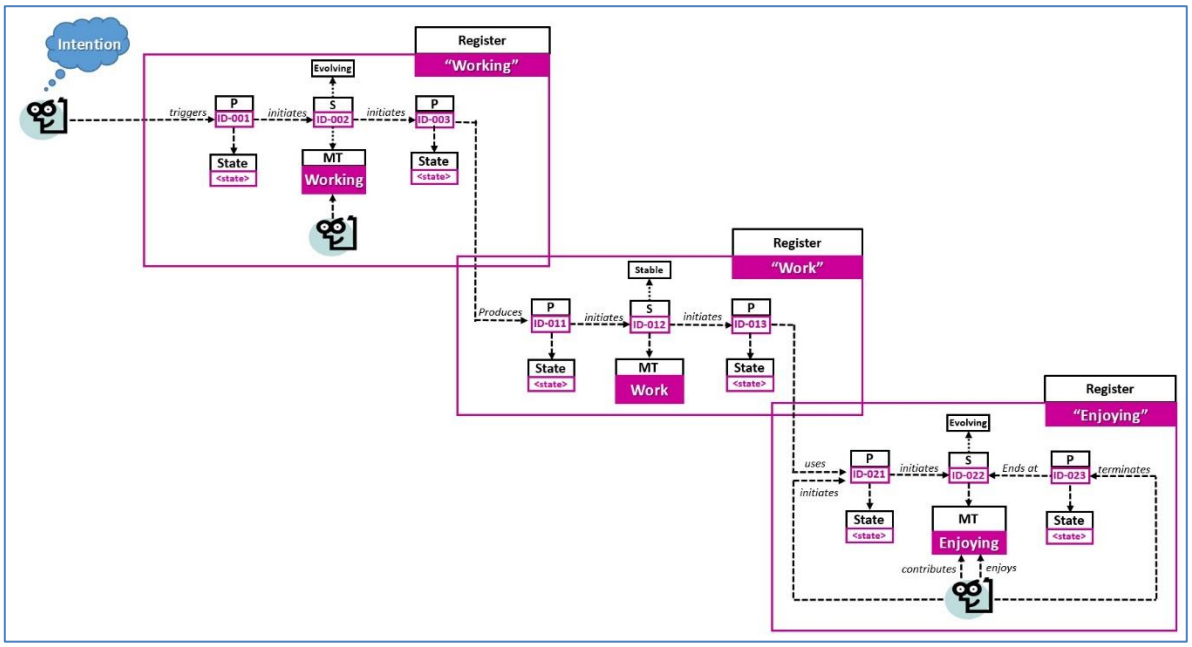

Figure 74 : la modélisation 'Working', 'Work' et 'Enjoying.'

Actuality

AXIS-CSRM organise la séparation de tout ce qui a à voir avec les choses « **stables** » du réel.

<à développer>

Working

AXIS-CSRM organise la séparation de tout ce qui a à voir avec les choses « **évolutives** » du réel.

<à développer>

Concepts <à développer>

AXIS-CSRM organise la séparation de tous les aspects qui ne sont **pas du domaine du réel**. Par exemple, le concept d'**« Œuvre »**au sens de FRBR.

<à développer>

Détails des Constructs <à développer>

Introduction à la définition formelle des Constructs

<A rédiger : une motivation de choix de technologie de modélisation des Constructs (Ontologie, Base de Connaissance racine …). Dans l'état actuel, la représentation de ces constructs s'impose d'être faite suivant la modélisation en normes structurées normalisée par le W3C [utilisation en particulier des langages normalisés RDF et OWL].>

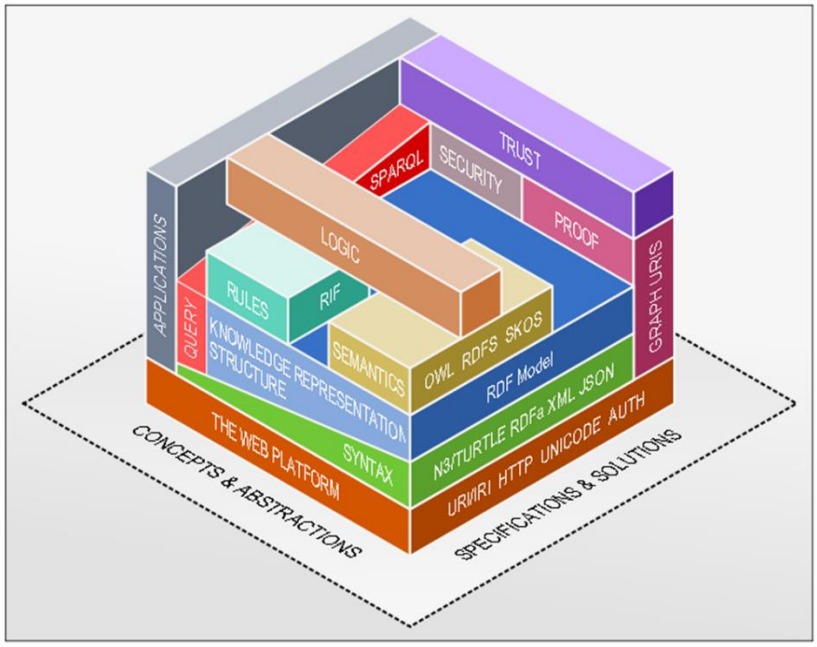

Figure 75 : La pile des normes du Web sémantiques du W3C

La représentation sémantique est normalisée et mature : il est possible de représenter la sémantique de manière normalisée, avec un ensemble de normes possédant un nombre extrêmement restreint de caractères (en l'occurrence la liste de caractères ASCII et rien de plus), quelques méta-symboles séparateurs et c'est tout ! La normalisation de la représentation est complètement achevée et elle a été bien conçue. [9]

Définition formelle des Constructs d'AXIS-CSDM : <à développer>

- **Classes** <à placer ici une telle représentation formelle>
- **Relationships** <à placer ici une telle représentation formelle>
- **Property** <à placer ici une telle représentation formelle>
- **Root Knowledge Base:** <à placer ici une telle représentation formelle>>

Suggestions de choix pour les Technologies de Modélisation <à développer>

<à placer ici ces suggestions de MT formelles & informelles>

[9] Cependant il n'existe pas encore une normalisation complète de l'exploitation de la sémantique. Dans les parties ''inférence'', il reste des choses à normaliser mais cela n'est pas essentiel pour la représentation des informations à transmettre (il s'agit simplement de représentations sémantiques à produire).

Exemple d'application des Constructs <à développer>

<L'intention est de placer ici des exemples de mise en œuvre des Constructs en montrant tous les cas introduits à la Section 6.5.2 et les cas complémentaires introduits aux sections 6.5.3 et 6.5.4>.

À rédiger ...